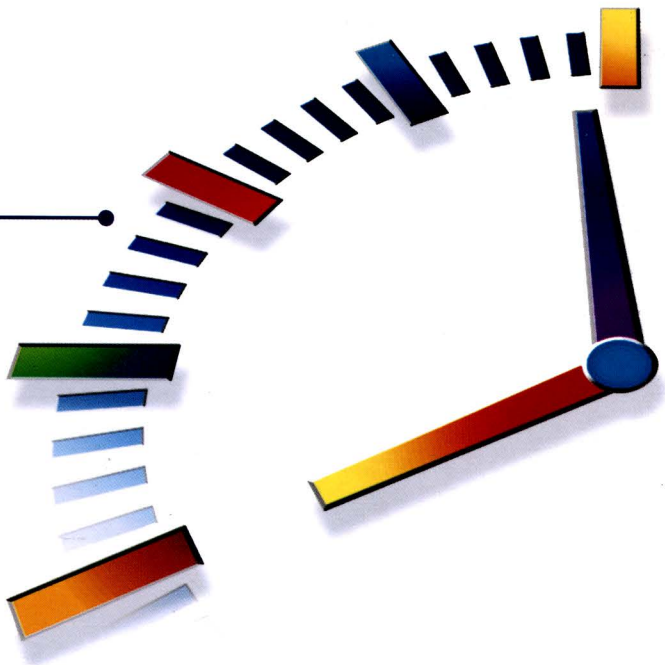
SQL
精華盡在其中

保證24小時學會的課程



第三波

Teach Yourself SQL

24小時自學手冊

原著：Stephens & Plew
編譯：佐登

SAMS

SQL ™

24小時自學手冊

原著

Ryan K. Stephens

Ronald R. Plew

編譯

佐登

第三波資訊股份有限公司 印行

Copyright

Trademarks

簡介

誰應該讀這本書？

歡迎來到關連式資料庫和 SQL 的世界！這本書寫給自動自發，想要藉由學習結構查詢語言 SQL 得到關連式資料庫技術方面知識的讀者，這本書主要爲偶而使用或從未使用 SQL 的關連式資料庫管理系統的人而寫，這本書也適用於那些對關連式資料庫有一些經驗，但是需要學習該如何在資料庫裡面瀏覽、對資料庫執行查詢、建立資料庫結構、處理資料庫裡的資料等，這本書對於使用 SQL 關連式資料庫有非常經驗的人，可能會感到不夠。

這本書所計劃提供的是什麼？

這本書是針對 SQL 方面沒有經驗的人寫的，或已經用關連式資料庫，但是他們在 SQL 裡的工作任務是非常有限的，這本書嚴格來的說應該是提供知識的機器。而我們所提供的教材是從最基礎開始，而且已經應用各章節所教導的知識，提供範例與練習題，這本書不是參考書，而且不應該只被參考而已。

您所需要的是什麼？

您可能會想，要使這本書爲我工作，我需要什麼？理論上，您應該拾起這本書，研究目前小時所談的教材，研究哪些例子和寫出練習題的答案，或在關連式資料庫上試著執行，然而，應用每小時裡的材料對您存取關連式資料庫系統將會是有益的，因爲 SQL 是爲所有關連式資料庫的標準語言，所以對您存取關連式資料庫不是主要的影響，您可能使用的資料庫系統，包括 Oracle、Sybase、Informix，Microsoft SQl Server、Microsoft Access 和 dBase。

在這本書所用的符號

大體而言，在這本書中我們已經盡可能使用簡單的符號。

在項目清單裡，您輸入的程式碼以黑體字、單一空格出現，輸出以標準的單一空格出現。如：

```
SELECT * FROM PRODUCTS_TBL;

PROD_ID    PROD_DESC                      COST
---------- ------------------------- ----------
11235      WITCHES COSTUME                29.99
222        PLASTIC PUMPKIN 18 INCH         7.75
13         FALSE PARAFFIN TEETH             1.1
90         LIGHTED LANTERNS                14.5
15         ASSORTED COSTUMES               10
9          CANDY CORN                      1.35
6          PUMPKIN CANDY                    1.45
87         PLASTIC SPIDERS                  1.05
119        ASSORTED MASKS                   4.95

9 rows selected.
```

下列特別設計特徵以提高本文：

有語法方格可以引起您注意到，在每個小時所討論相關指令的語法。

```
select [ all | * | distinct column1, column2 ]
from table1 [ , table2 ];
```


註解

一分鐘的文字說明，用以澄清前一主題的重要概念，如果您對前一主題已清楚的理解，您便可以跳過此部份內容。


注意

警告用來提醒使用者哪邊該特別注意與小心。

提示

本書提供大量的「提示」以幫助讀者解決一般常見的問題。

ANSI SQL 和廠商所用的 SQL

寫本像這樣相關 SQL 的書，是很困難的，雖然有 ANSI 爲標準的 SQL，但每個資料庫廠商有它的 SQL，要針對使用每個廠商變更標準、增強標準和自標準遺失一些元件等，得特別小心。

因爲有 ANSI 爲標準的 SQL，所以可預期的問題是，爲什麼教標準的 SQL 是很困難的一件事？這個問題的答案開始於 ANSI SQL 指令，它只是個標準而已，ANSI SQL 不是真的程式語言，爲了要教您 SQL，我們必須提出例子和練習，例子和練習提到包括使用一個或多個不同加廠商的 SQL，而且因爲每個廠商用不同 SQL 的語言，這些不同的變更，有它自己的規格與用法，如果不在本書適當地處理，實際上各種不同的 SQL 指令語法，容易引起混淆，因此，我們嘗試盡可能接近 ANSI 標準，如果與 ANSI 規定的正確語法不同，再討論 ANSI 標準與不同廠商所應用的例子。

然而，我們在旁邊有註解與提示，提醒讀者注意不同廠商之中例子的變化與應該注意的地方，只是請記得，每個廠商之間的差異是很小的，最重要的是，您能了解 SQL 和它指令所隱藏的觀念，雖然輕微的差異確實存在，但是 SQL 基本上是相同。

了解範例和練習題

我們選擇這本書裡的大部份例子使用 Oracle；然而，我們也顯示來自 Sybase、Microsoft SQL Server 和 dBase 的例子，使用 Oracle 有各種不同的理由，包括 Oracle 符合 ANSI SQL 和 Oracle 是今天其中最最流行的一種關連式資料庫產品等事實，正如所說的，在 SQL 的廠商之中的語法裡的確有一些不同，舉例來說，如果您嘗試執行這本書裡的一些例子，您可能必須修改來符合您所使用的廠商正確語

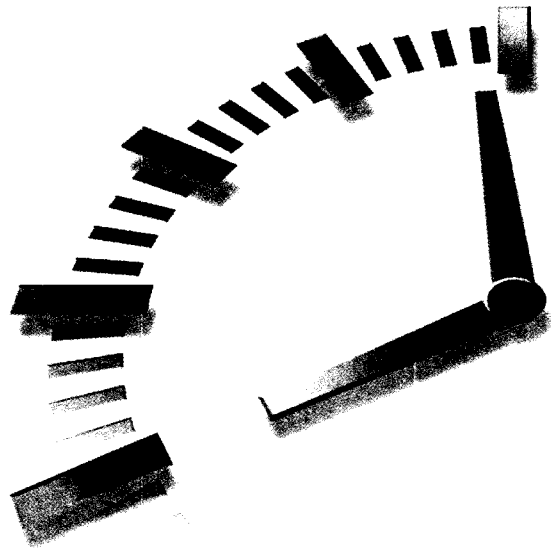
法，我們試著保存所有的例子符合標準；然而，我們也企圖地顯示一些不完全符合的例子，但是所有指令的基本結構是相同的，爲了要學習 SQL，您必須使用實際廠商的例子，應該沒有很大的困難，學習應用本書所提到的資料庫範例，因爲使用不同的 SQL，您可能必須更改本書裡所提到的範例，以適合您的廠商，任何的調整將會幫助您了解您的廠商特別的語法和特徵。

祝您好運了！

目錄

第 I 單元　SQL 概念總覽

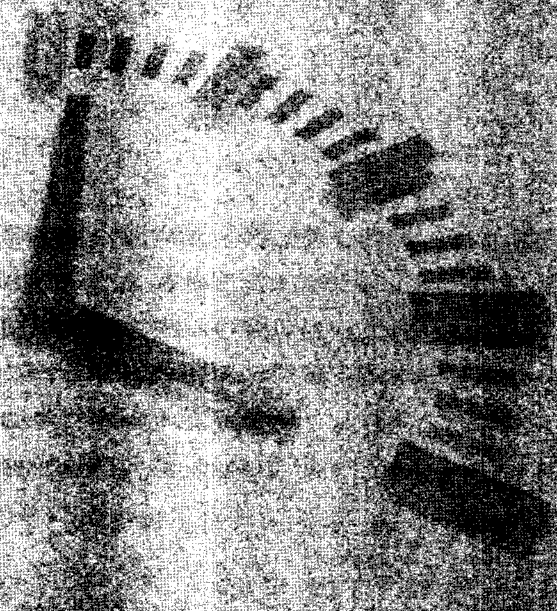
第 II 單元　建立您的資料庫



第 1 單元

SQL 概念總覽

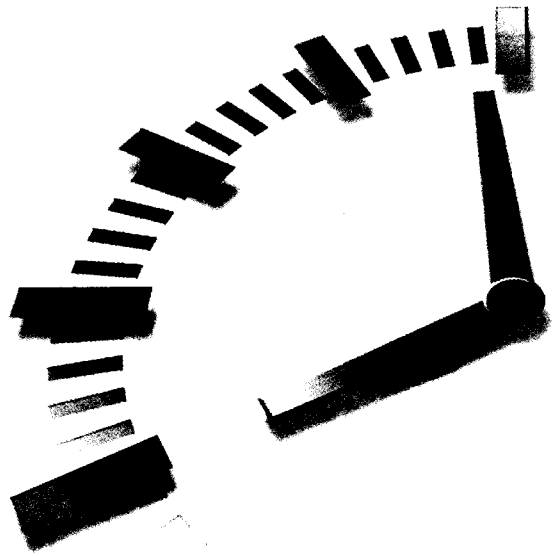
第 1 小時　歡迎來到 SQL 的世界



天道 | 酬

[illegible] 102

[illegible]



第 1 小時

歡迎來到 SQL 的世界

歡迎來到 SQL 的世界，一個在今日全球商業界快速成長的資料處理技術。在閱讀這本書時，您要有心理準備，準備接受一種即將在今日關連式資料庫和資料管理所必須了解的知識，因爲這是讓您瞭解 SQL 的背景與概念的第一步，在本章節這樣概念大多採用文字敘述讓您了解，而第一小時絕對讓您有如獲至寶的感覺。

本小時的重點包括：

- 介紹 SQL 與其歷史
- 介紹資料庫管理系統
- 說明基礎名詞和概念
- 介紹本書所提到的資料庫廠商

1-1 SQL 的定義與歷史

每一個企業都有資料，不論利用人工或機械式，這些資料都須要利用某些方法來管理。這樣機械式的管理方式，通稱爲資料管理系統（Database Management System, DBMS）。資料管理系統已經行之有年，剛開始許多都採用大型主機（Mainframe）的平面檔（Flat-File）系統，因爲企業的成長、型態的改變與 Internet 的技術，產生大量的資料而導致今日資料管理系統的技術才開始轉型。

現代資訊管理的潮流是使用關連式資料管理系統（Relational Database Management System, RDBMS），它不同於傳統的資料管理系統。目前大多數的企業都利用關連式資料庫和 Client/Server（主從式系統）的組合，很成功的管理它們的資料，同時也使它們在同業間維持競爭性。爲了讓您對有關連式資料庫的語言—SQL 有扎實的基礎，下幾小時將介紹關連式資料庫和 Client/Server 的技術。

1-1-1 何謂 SQL？

SQL，Structured Query Language，是一個用來和關連式資料庫溝通的標準語言。它原本是 IBM 採用 Dr. E.F. Codd 的論文（“A Relational Model of Data for large Shared Data Banks”）爲雛型來開發。在 1979，在 IBM 開發雛型後沒多久，第一套 SQL 的產品－Oracle（甲骨文），正式被 Relational Software Inc.（後來更名爲 Oracle Corporation）公佈上市發行，今日它是關連式資料庫技術的領導者之一。SQL 有二種念法：只唸英文字母 S-Q-L 或是“sequel”；二種發音都被接受與採用。

1-1-2 何謂 ANSI SQL？

美國國家標準局（ANSI）是一個專門爲不同產業制定標準的機構。SQL 被認爲是一個用來和關連式資料庫溝通的標準語言，原本在 1986 年時已核准 IBM 所採用的 SQL 爲標準語言，在 1987 年，ANSI

SQL 標準被國際標準組織（International Organization for Standardization, ISO）認定爲國際標準，目前的標準被稱爲 SQL/92，和 1986 的版本差異很多。

NEW TERM ANSI（American National Standards Institute）爲美國國家標準局專司爲不同產品和觀念制定標準的機構。

面對著不同的標準，當然有好處也有壞處。在業界居主流的廠商有著絕對領導開發的地位，同時也主導開發的方向。對 SQL 而言，誰能在必需的基本架構下提供固定、完整與輕巧的程式變成是重點（這不僅指程式而已，而是包含資料庫和對管理資料庫的人而言）。

有些人可能對被制定的標準有意見，認爲不夠好、有太多限制與缺乏彈性，而且在特定的安裝與運用時，會出現不相容的問題。但是大部份的廠商依然根據標準來設計程式，同時爲了提高標準 SQL 的效能，會設計其他功能來加強其產品的功能並解決上述的問題。

有一定的標準畢竟是好的，它同時考量了好處與壞處。期待中的標準所指定的特性應被任何 SQL 程式應用，除了強迫不同競爭廠商的 SQL 保持一貫性，並同時描述出大概的基本概念，也增加今日市場上 SQL 程式設計師和關連式資料庫使用者的價值。

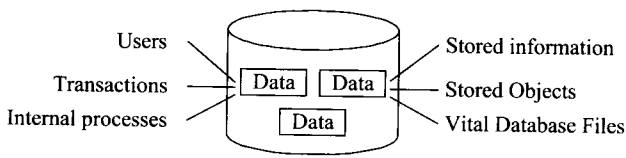
1-1-3 何謂資料庫（Database）？

從字面上來定義，資料庫（Database）是資料的收集；另一方面也可以想成使用者用有效率的方式讀取被整理好的資料庫。

我們每天都用到資料庫而不自覺。電話簿就是一個很好的例子，它包含了每一個人的姓名和電話號碼。名單依字母次序排列或索引的方式讓使用者能很容易的找到想要的資料。這資料是儲存在電腦內的某一個資料庫，而且每年新版本的每一頁都不再需要人工鍵入了！

資料庫也是要維護的，像人們搬遷到不一樣的地址時，他在電話簿的記錄就必需要更動。相同的道理，人們可能更改姓名、住址及電話，在這時候，這些資料就必需被更改。

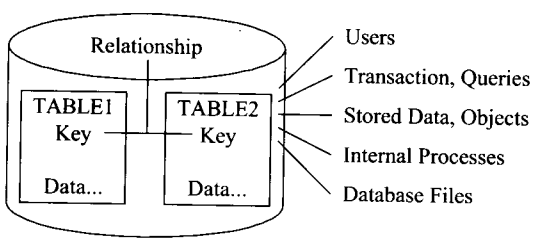
圖 1.1.
資料庫



1-1-4 介紹關連式資料庫（Relational Database）

關連式資料庫被分成邏輯單位稱爲“資料表格”而這資料表格在資料庫內和另一個資料表格有關連。在關連式資料庫內允許資料被分成至邏輯更小或更容易管理的單位，如此一來資料庫將更容易維護，另一方面也可依照企業層次提供最佳化的資料庫。在圖 1.2，您可以看到資料表格透過共同索引鍵形成相互關連。

圖 1.2.
關連式資料庫



由於資料表在關連式資料庫有相互關連，因此能讓適當的資料被查詢檢索出來（儘管資料可能儲存在多個資料表內）。利用共同索引或欄位，在關連式資料庫之間的資料表與在多重資料表的資料可以被結合並輸出到表單。當您繼續閱讀本書，您將會發現關連式資料庫的優點，例如整體的效能與存取資料簡易。

NEW TERM 關連式資料庫（Relational Database）是由相關的物件與主要的資料表組合而成的。

NEW TERM 在資料庫中資料表格（Table）通常是被指爲儲存資料的地方。

1-1-5 介紹主從式系統（Client/Server）技術

在過去，電腦市場的規則都被大型主機所決定，大而快速的電腦可以儲存很大的容量與快速處理資料的能力。使用者用終端機和大型主機溝通，終端機是一台不會處理任何事情、所有動作都在大型主機上處理，使用大型主機的 CPU、硬碟空間與記憶體，每台終端機有一條資料傳輸線連著大型主機。在今日大型主機提供了不少的服務給許多不同的企業界，但是將有更新的技術導入：就是主從式系統（Client/Server）模式。

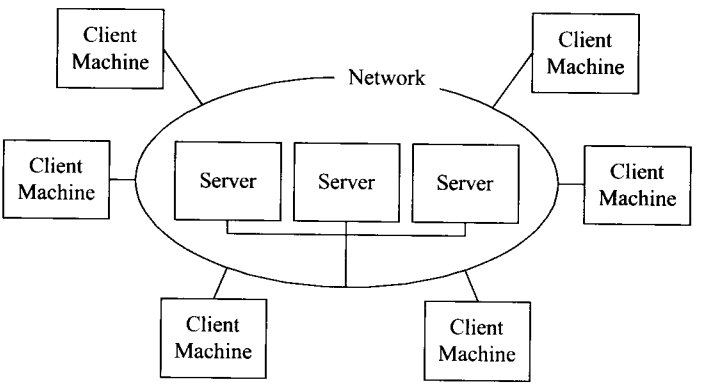
在主從式系統（Client/Server）內，主要的電腦稱爲伺服器（Server），它可以透過網路來讀取，如本地網路（Local Area Network, LAN）或廣域網路（Wide Area Network, WAN）。個人電腦（PCs）或另一台伺服器取代了終端機來存取伺服器，每一台 PC 被稱爲用戶端（Client），用來提供存取網路，讓伺服器和用戶端可以溝通，則稱爲主從式系統（Client/Server）。Client/Server 與 Mainframe 最大的差異在於，在 Client/Server 的環境是使用 PC 爲用戶端，PC 有能力自己思考，可以用自己的 CPU 和記憶體來執行應用軟體，通過網路來存取伺服器的資料。對今日的企業界而言，Client/Server 系統是比較有彈性的，同時也是較受喜愛的。

關連式資料庫可同時被 Client/Server 與 Mainframe 使用。雖然 Client/Server 系統比較被喜愛，但公司還是會依照自己的需求來選擇是否繼續採用 Mainframe。不少公司正在把資料從 Mainframe 轉到 Client/Server，利用先進的技術，提供更有彈性且更適合自己公司需要的資料，同時也讓後端的系統能解決千禧年相容的問題。

不少公司成功的轉到 Client/Server 系統，也因此利用到它的優點。當然也有不少公司花了上千萬來導入 Client/Server 系統，但是失敗了，而繼續使用 Mainframe；此外還有其它公司在觀望中。對於新的技術，缺乏訓練與適當的專家，導致失敗的主要原因。在今日的企業界，了解 Client/Server 系統的技術，就像了解開發 Internet 與

Network Computing 的技術一樣重要，而且市場也急需這方面的人才。圖 1.3 顯示 Client/Server 技術的觀念。

圖 1.3.
Client/Server 模式



1-1-6 較熱門的資料庫廠商

一些主要的資料庫廠商包含 Oracle、Microsoft、Informix、Sybase 與 IBM。雖然，還有不少其他廠商，但這所提到的大多是您從書架、報紙、雜誌、股票市場與 Internet 上都可以聽到。

產品之間的差異

正如世界上的每一個人，天生的條件都有差異，而每一家廠商的 SQL 也不盡相同。資料庫和其他在市面上的產品一樣，被許多廠商製造。對廠商較有利的是確保他們的產品能符合現今 ANSI 輕巧與對使用者方便的標準。比方說，一家公司正要從現有的資料庫系統轉移到另一個資料庫系統，他一定不想資料庫使用者再花時間去學另一種程式語言來更新、維護新的資料庫系統的功能。

當然您也可以發現，在每一家廠商的資料庫中，都會內建一些新的加強功能。這些加強或延伸的功能是額外的指令，也只存於特定廠商的資料庫系統內，同時對標準的 SQL 而言也能提供額外的優勢。

1-2 SQL 指令種類

我們將在下列章節討論基本的 SQL 指令表現不同的功能。這些功能包含建立資料物件、製造物件、用資料建立資料庫表格、在表格內更新資料、刪除資料、查詢資料庫、控制資料的存取與資料庫所有的管理。

主要的種類爲：

- DDL，Data Definition Language（資料定義語言）
- DML，Data Manipulation Language（資料處理語言）
- DQL，Data Query Language（資料查詢語言）
- DCL，Data Control Language（資料控制語言）
- Data administration commands（資料管理指令）
- Transactional control commands（異動控制指令）

1-2-1 定義資料庫的結構

Data Definition Language（DDL），資料定義語言是 SQL 的一部份，他允許資料庫使用者去建立與重建資料庫物件，比如說建立或刪除表格。

下列的 DDL 指令將於後面的小時中介紹：

```
CREATE TABLE
ALTER TABLE
DROP TABLE
CREATE INDEX
ALTER INDEX
DROP INDEX
```

這些指令在第 3 小時“管理資料庫物件”和第 17 小時“增進資料庫工作效率”會詳細的介紹。

1-2-2 資料處理語言（DML）

Data Manipulation Language（DML），資料處理語言是 SQL 的一部份，用來製造在關連式資造庫物件的資料。

三種基本 DML 的指令：

```
INSERT
UPDATE
DELETE
```

這些指令在第 5 小時“處理資料”會詳細的介紹。

1-2-3 資料查詢語言（DQL）

很直接的只有一個的指令，Data Query Language（DQL）資料查詢語言，是針對 SQL 中的關連式資料庫使用者。其指令如下：

```
SELECT
```

這指令是必須和許多其他的選項和規定一起使用，來查詢關連式資料庫。查詢任何從簡單到複雜，模糊不定到特定的資料。SELECT 指令在第 7 小時到第 16 小時會有詳細精彩的介紹。

NEW TERM 查詢（Query），是從資料庫調閱資料出來。

1-2-4 資料控制語言（DCL）

資料控制語言讓您在資料庫內能控制資料的存取，DCL 指令通常使用在建立讓使用者可存取的物件與散佈特有的權利給全部的使用者。

一些 DCL 指令如下：

```
ALTER PASSWORD
GRANT
REVOKE
CREATE SYNONYM
```

您將會發現這些指令是和其他指令一起使用的，並且在不同的章節都會出現。

1-2-5 資料管理指令

資料管理指令讓使用者對在使用中的資料庫產生稽核與分析，它也用來幫助分析系統的效益。

二種主要的資料管理指令如下：

```
START AUDIT
STOP AUDIT
```

不要將資料管理與資料庫管理搞混了，資料庫管理是全盤的資料庫管理並且使用到不同層次的指令。

1-2-6 異動控制指令

除了上述的三種指令外，還有一種指令用來管理資料的異動。

- COMMIT　用來儲存資料庫的異動。
- ROLLBACK　用來復原資料庫的異動。
- SAVEPOINT　在群組內建立記號，方便復原。
- SET TRANSCATION　幫每一次異動定名。

異動控制指令在第 6 小時 “管理資料庫異動” 會詳細的介紹。

1-3 介紹本書所使用的資料庫

在您瞭解 SQL 的基本概念前，下一步是介紹您在下 23 個小時的課程會使用到表格與資料。下二個段落提供一份使用資料庫的總覽，如彼此的關係、架構與使用的範例資料。

1-3-1 本書的表格圖表

在圖 1.4 中，顯示本書的範例、測驗、問題所使用的表格間的關係。每一個表格與欄位都給予特定的名稱。下列的關連線是連結特定表格的通用欄位，通常被稱爲主索引。

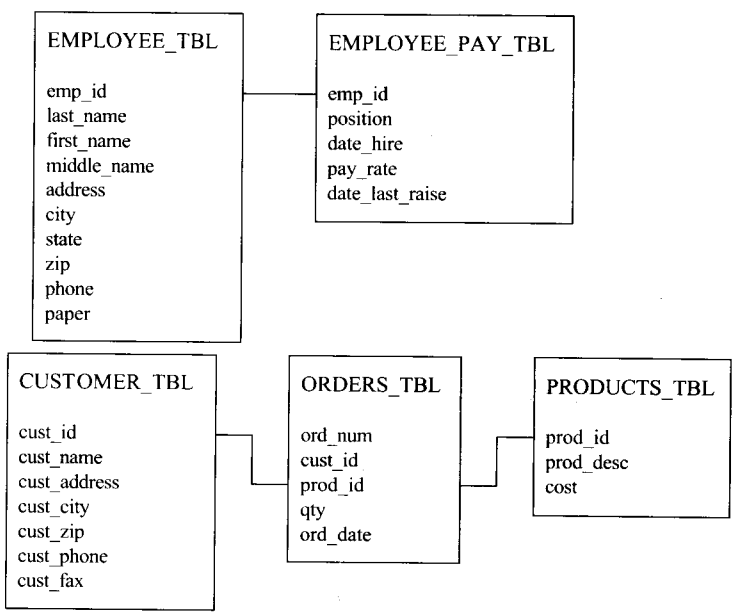


圖 1.4.
本書的表格圖表

1-3-2 表格命名的標準

表格命名的標準，正像企業內的任何標準一樣，必須很小心的管理。在前面學習過表格和資料後，您應該知道每一個表格的結尾是 _TBL，_TBL 用來告知您這個物件是表格，在關連式資料庫中有許多種

類的物件，這是被選擇在這使用的命名標準且也使用在不同的用戶端。比方說，在往後的章節您會看到用來認定表格索引的結尾是 _INX。表格命名標準在企業內部是很嚴格的，也因此對任何關連式資料庫的管理有很大的幫助。

> 您除了要注意 SQL 的命名文法外，對於企業的文化與規定也必須遵從。

1-3-3 檢視資料

在本章將讓您預覽本書所提到的每一個表格與所包含的資料，請花數分鐘瞭解這些資料、變數與資料間的關係。請注意有些特定的欄位可能不需要資料，這是當時建立資料庫設定的。

EMPLOYEE_TBL

```
EMP_ID    LAST_NAM FIRST_NA M ADDRESS          CITY          ST ZIP   PHONE
--------- -------- -------- - ---------------- ------------- -- ----- ----------
311549902 STEPHENS TINA     D RR 3 BOX 17A     GREENWOOD     IN 47890 3178784465

442346889 PLEW     LINDA    C 3301 BEACON      INDIANAPOLIS  IN 46224 3172978990

213764555 GLASS    BRANDON  S 1710 MAIN ST     WHITELAND     IN 47885 3178984321

313782439 GLASS    JACOB      3789 RIVER BLVD  INDIANAPOLIS  IN 45734 3175457676

220984332 WALLACE  MARIAH     7889 KEYSTONE    INDIANAPOLIS  IN 46741 3173325986

443679012 SPURGEON TIFFANY    5 GEORGE COURT   INDIANAPOLIS  IN 46234 31756790047
```

EMPLOYEE_PAY_TBL

```
EMP_ID    POSITION        DATE_HIRE   PAY_RATE DATE_LAST     SALARY      BONUS
--------- --------------- --------- ---------- --------- ---------- ----------
311549902 MARKETING       23-MAY-89            01-MAY-97       4000
442346889 TEAM LEADER     17-JUN-90      14.75 01-JUN-97
213764555 SALES MANAGER   14-AUG-94            01-AUG-97       3000       2000
313782439 SALESMAN        28-JUN-97                            2000       1000
220984332 SHIPPER         22-JUL-96            11 01-JUL-97
443679012 SHIPPER         14-JAN-91            15 01-JAN-97
```

CUSTOMER_TBL

CUST_ID	CUST_NAME	ADDRESS	CUST_CITY	ST	ZIP	CUST_PHONE	CUST_FAX
232	LESLIE GLEASON	798 HARDAW AY DR	INDIANAPOLIS	IN	47856	3175457690	
109	NANCY BUNKER	APT A 4556 WATERWAY	BROAD RIPPLE	IN	47950	3174262323	
345	ANGELA DOBKO	RR3 BOX 76	LEBANON	IN	49967	7658970090	
090	WENDY WOLF	3345 GATEW AY DR	INDIANAPOLIS	IN	46224	3172913421	
12	MARYS GIFT SHOP	435 MAIN S T	DANVILLE	IL	47978	3178567221	3178523434
432	SCOTTYS MARKET	RR2 BOX 17 3	BROWNSBURG	IN	45687	3178529835	3178529836
333	JASONS AND DALL AS GOODIES	LAFAYETTE SQ MALL	INDIANAPOLIS	IN	46222	3172978886	3172978887
21	MORGANS CANDIES AND TREATS	5657 W TENTH ST	INDIANAPOLIS	IN	46234	3172714398	
43	SCHYLERS NOVELT IES	17 MAPLE ST	LEBANON	IN	48990	3174346758	
287	GAVINS PLACE	9880 ROCKV ILLE RD	INDIANAPOLIS	IN	46244	3172719991	3172719992
288	HOLLYS GAMEARAM	567 US 31	WHITELAND	IN	49980	3178879023	
590	HEATHERS FEATHE RS AND THINGS	4090 N SHA DELAND AVE	INDIANAPOLIS	IN	43278	3175456768	
610	RAGANS HOBBIES	451 GREEN	PLAINFIELD	IN	46818	3178393441	3178399090
560	ANDYS CANDIES	RR 1 BOX 34	NASHVILLE	IN	48756	8123239871	
221	RYANS STUFF	2337 S SHELBY ST	INDIANAPOLIS	IN	47834	3175634402	

ORDERS_TBL

```
ORD_NUM    CUST_ID PROD_ID          QTY ORD_DATE
---------- ------- ---------- ---------- --------
56A901     232     11235               1 22-OCT-97
56A917     12      907               100 30-SEP-97
32A132     43      222                25 10-OCT-97
16C17      090     222                 2 17-OCT-97
18D778     287     90                 10 17-OCT-97
23E934     432     13                 20 15-OCT-97

PRODUCTS_TBL

PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
11235      WITCHES COSTUME                29.99
222        PLASTIC PUMPKIN 18 INCH         7.75
13         FALSE PARAFFIN TEETH            1.10
90         LIGHTED LANTERNS               14.50
15         ASSORTED COSTUMES              10.00
9          CANDY CORN                      1.35
6          PUMPKIN CANDY                   1.45
87         PLASTIC SPIDERS                 1.05
119        ASSORTED MASKS                  4.95
```

1-3-4 表格的組成

資料庫的存在是爲了保存與整理寶貴的資料，您剛看過本書用來解釋 SQL 概念的資料。下列章節將讓您進一步瞭解組成表格的原件，請記住，表格是關連式資料庫最重要也最簡單的資料儲存格式。

欄位（Field）

每一格表格是由許多稱爲欄位的小項目所組成的，組成 PROUCTS_TBL 的欄位是 PROD_ID，PPROD_DESC 與 COST，這些欄位在指定的表格中記錄特別的資訊。

NEW TERM 欄位（field）是在表格內用來整理表格中特別記錄的每一筆資訊。

資料記錄（Record，or Row of Data）

表格中每一筆輸入項被稱爲記錄，再看上一個表格，PRODUCTS_TBL，第一筆資料如下：

```
11235       WITCHES COSTUME                    29.99
```

很清楚的可以看出這記錄是產品的編號、敘述與單價，在每一個單一的產品，都必須有相關的記錄在 PRODUCTS_TBL 表格內，每一筆記錄的輸入項在表格內是呈水平排列的。

NEW TERM 每一筆資料“row of data“在關連式資料庫內即是一筆記錄。

欄（Column）

欄，在表格內的輸入項是呈垂直列排的，它包含了以表格爲基準，特定欄位的所有資料。比方說：在 PRODUCTS_TBL 的欄應和產品的敘述有關，包含如下：

```
WITCHES COSTUME
PLASTIC PUMPKIN 18 INCH
FALSE PARAFFIN TEETH
LIGHTED LANTERNS
ASSORTED COSTUMES
CANDY CORN
PUMPKIN CANDY
PLASTIC SPIDERS
ASSORTED SPIDERS
ASSORTED MASKS
```

這欄是產品的敘述，以 PROD_DESC 的欄位爲基準，在表格內，欄針對每一欄位拉出特定的資訊。

NEW TERM “欄”（column）包含表格內特定欄位的所有資料。

主索引（Primary Key）

在關連式資料庫內主索引（Primary Key）是用來區別表格內資料的單一性，PRODUCTS_TBL 的主索引是 PROD_ID，這是在最初建立表格時設定的。主索引的功能是用來確定所有產品的編號是唯一的，因此在 PRODUCTS_TBL 的每一筆資料有他們自己的 PROD_ID，主索引避免表格產生重複的資料，同時我們也會介紹其他方面的應用。

NEW TERM “主索引”（primary key）是使表格內欄的資料記錄是唯一。

Null 值

在表格內 NULL 值的欄位表示空白，欄位有 NULL 值是指欄位不包含任何值，釐清 NULL 值、零數值或欄位只有空格的關係是非常重要的。當欄位包含 NULL 值是指欄位在建立時，沒有鍵入任何資料，像在 EMPLOYEE_TBL 的表格內，並不是每一位員工都都有中間的縮寫名，也由於他們沒有中間的縮寫名，所以在欄位內是 NULL 值。

NEW TERM NULL 是代表缺少值的資料。

其他表格的元件將會在下二個小時詳細介紹。

1-4 摘要

我們已經介紹了 SQL 的標準語言，也介紹了它的歷史與簡介，在過去幾年間，標準是如何形成的。資料庫系統和現在的技術也介紹過，其中包含關連式資料庫、伺服器與用戶端系統，這對您瞭解 SQL 是非常重要的。SQL 語言主要的元件與關連式資料庫的市場和許多廠商的產品都有介紹，不管 ANSI SQL 如何變動，大部分的廠商均與目前的標準相容並做了適當的延展功能設計，也因此使在 SQL 上的開發能保持一致性，同時也更輕便。

本書所使用的資料庫也已經介紹過了，您所看到的資料庫包含互相有關聯的表格與和本小時有關的相關資料。您應該吸收了一些 SQL 的整體基本概念同時也應該瞭解關連式資料庫的概念，在本章的綜合練習將會做個複習，讓您更有信心進行下一小時的課程。

1-4-1 Q&A

Q 如果我學習 SQL，我將會使用任何版本的 SQL 嗎？

A 是的，您將可以與所有和 ANSI SQL 相容的資料庫溝通，就算產品沒有完全相容，您也會很快的上手並做適當的更正。

Q 在伺服器與用戶端系統環境，個人電腦是伺服器或用戶端？

A 個人電腦是用戶端。

Q 在建立每一個表格時，我需要都使用 _tbl 嗎？

A 當然不用，使用 _tbl 是爲了讓您的資料庫內表格命名更簡單與方便，您也可以用 table 代替 _tbl，或您不想使用結尾都可以。如 employee_tbl 可用 employee 來表示。

Q 當我插入一筆新的記錄到表格時，員工電話欄為什麼一定要鍵入資料（顯示 not null）？

A 有二種可能，因爲那欄位被設定必須鍵入資料，因此您必須收集相關資料後再插入那筆資料；或者是，更改欄位的定義，從 not null 改成 null，在第 3 小時將會教您如何更改欄位定義。

1-5 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成，隨堂測驗是用來測驗您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

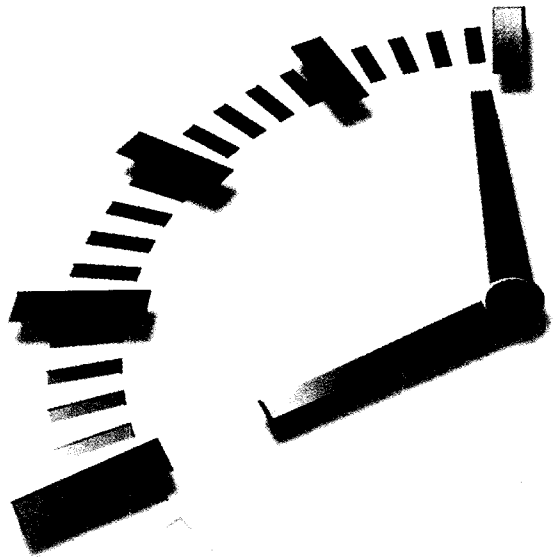
1-5-1 隨堂測驗

1. SQL 的縮寫代表什麼？
2. SQL 指令的六個主要類別為何？
3. 四種異動控制指令是指那四種？
4. 主從式系統與大型主機的主要差異為何？
5. 如果欄位定義為 null，是否代表這欄位一定要輸入資料？

1-5-2 練習題

1. 請說明下列 SQL 指令屬於那一種指令類別？

```
create table
delete
select
insert
alter table
update
```



第 II 單元

建立您的資料庫

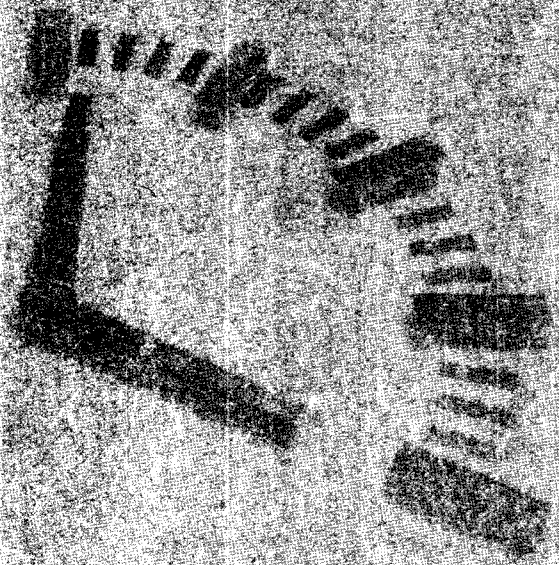
第 2 小時　資料結構的定義

第 3 小時　管理資料庫物件

第 4 小時　正常化程序

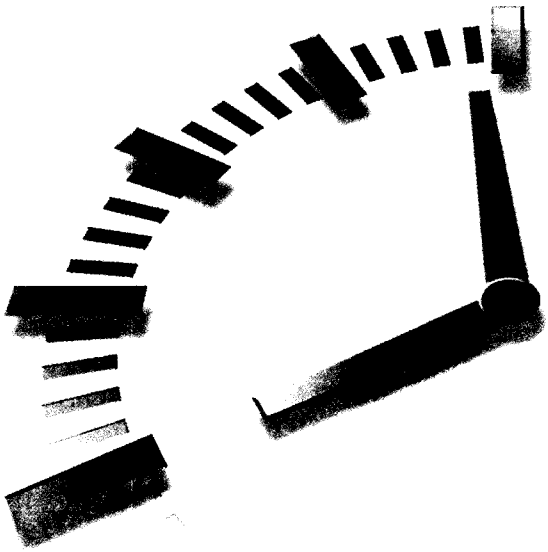
第 5 小時　處理資料

第 6 小時　管理資料庫異動



工業設計

概論與應用



第 2 小時

資料結構的定義

在第 2 小時，您將會瞭解更多在第 1 小時所看到的資料，包括資料本身的特性、瞭解資料是如何儲存在關連式資料庫中，另外您也會發現許多不同的資料格式。

本小時的重點包括：

- 瀏覽表格內的基本資料
- 介紹基本的資料格式
- 介紹如何使用不同的資料格式
- 舉例敘述資料格式的差異

2-1 何謂資料？

所謂資料是收集不同的資訊然後儲存在資料庫內，資料包含姓名、數字、金額、文字、圖表、小數、計算、摘要與任何您想得到的

物件。資料可儲存爲英文大寫、小寫或混合都可以，資料可以被刪除、更改，大多數的資料都不會永遠保持固定。

NEW TERM 資料格式（Datatypes）是用來規定在特定欄位的資料。資料格式牽扯到資料儲存的方式與在欄位的長度，資料值可以爲數字、日期、時間等。

資料是屬於資料庫的而且是必須被保護的。

2-2 基本的資料格式

下列將討論 ANSI SQL 所支援的基本資料格式，資料格式說明了儲存在表格欄位內資料本身的特性。舉例說，您可以指定一個欄位只能輸入數字而不能同時輸入數字與文字，也不可以在金額欄位中打入文字。


註解

不同的 SQL 廠商也提供自己的資料格式，瞭解並使用每個廠商所提供的資料格式，可以知道每個廠商如何儲存資料，但每個廠商的基本資料格式還是相同的。

如同其他電腦語言，最基本的資料格式爲：

- 字元串（Character strings）
- 數字串（Numeric Strings）
- 日期與時間值（Date and time values）

2-2-1 固定長度字元

固定字元（Constant characters），這些字元總是有一定的長度並使用固定長度字元來儲存，SQL 固定長度字元的標準爲：

```
CHARACTER(n)
```

n 代表在定義特定的欄位時，分配或最大長度。

有些廠商用 CHAR 來表示固定長度字元，數字與文字的資料可以被儲存在這類資料格式，最好的例子就是郵遞區號，所有的郵遞區號皆爲三碼。

空格通常都用來填滿固定長度字元不足的地方，如果一個欄位的長度爲十，資料只佔去了七，那其餘的部分將被認定與記錄成空格，這樣的處理方式是確保欄位中的每筆資料都是固定長度。


注意

若資料的長度不一，如人名、住址時，請小心使用固定長度字元的資料格式。如果不正當的使用固定長度資料格式，將會引起不必要的問題，如浪費空間與無法使用資料來做進一步的比較。

2-2-2 變數字元（Variable Characters）

SQL 支援變化長度的字串，所有資料的字串長度不是固定的，SQL 的標準變化長度字元爲：

```
CHARACTER VARYING (n)
```

n 代表在定義特定的欄位時，分配或最大長度。

最普遍的變數長度字元值的資料格式是 VARCHAR 與 VARCHAR2。Microsoft SQL Server 使用 ASNI 的標準－VARCHAR；VARCHAR2 只能被 Oracle 使用，因爲未來會更改 VARCHAR 的使用方式，資料可儲存爲文數字。

固定長度的資料格式都用空格來填滿固定長度字元不足的地方，但是變化長度的資料格式並不採用這方法。比方說：如果一個欄位的長度爲十，資料只佔去了五，那麼那筆資料的長度就是五，不用空格來填滿變化長度字元不足的地方。


提示

為了節省資料庫的空間，對於一些不固定的字元，請多使用變化長度的資料格式。

2-2-3 數值（Numeric Values）

數值是儲存於欄位中被定義爲某種格式的數字，如 NUMBER、INTEGER、REAL、DECIMAL 等。

SQL 標準的數值爲：

```
BIT(n)
BIT VARYING(n)
DECIMAL(n,n)
INTEGER
SAMLLINT
FLOAT(p)
REAL(s)
DOUBLE PRECISION(p)
```

n 代表在定義特定的欄位時，分配或最大長度。

在 SQL 最普遍的數值資料格式爲 NUMBER，是依照 ANSI 所提供的值來使用，數值可儲存零、正數、負數與浮點數（floating-point），如以下使用 NUMBER 的範例：

```
NUMBER (5)
```

以上範例是限制這欄位最大值只到 99999。

2-2-4 小數位值（Decimal Values）

小數位值是數值的一種，可以使用小數點，SQL 的小數值標準如下，第一個 *n* 是精準值，第二個 *n* 是小數比例。

```
DECIMAL(n,n)
```

NEW TERM 精準值（precision）是數值的全部長度。比方說在定義數值 DECIMAL(4,2)，精準值爲 4，是指數值的全部長度爲 4。

NEW TERM 比例（scale）是指有幾位小數點數。上一個範例，小數比例爲 2，是指二位小數點。

範例：34.33 在 DECIMAL(3,1) 的指令後變成爲 34.3。

如果一個欄位的數值定義爲 DECIMAL(4,2)，那麼最大值爲 99.99。

精準值爲 4，代表數值的全部長度爲 4，比例爲 2，代表在小數點右邊的位置或位元，小數點本身不能算成一個字元。

下列是欄位的數值都可定義成爲 DECIMAL(4,2)：

```
12
12.4
12.44
12.449
```

最後的一個數值爲 12.449，會被四捨五入爲 12.45。

2-2-5 整數（Integers）

整數是指數值不包含任何小數點，但可爲正、負數。

合法的整數爲：

```
1
0
-1
99
-99
199
```

2-2-6 浮點小數值（Floating-Point Decimals）

浮點小數值是指小數點的精準值與比例的值是變化長度且無限的。任何精準與比例的值都可以被接受。實數（REAL）的資料格式是給欄位只包含單一精準浮點數值，雙倍精準值（DOUBLE PRECISION）的資料格式是給欄位只包含雙倍精準浮點數值。要成爲單一精準浮點數值，精準值必須包含 1 到 21；要成爲雙倍精準值，精準值必須包含 22 到 53。

```
FLOAT
FLOAT(15)
FLOAT(50)
```

2-2-7 日期與時間（Date and Time）

日期與時間的資料格式是很容易瞭解的，用來查詢、追蹤有關日期與時間的資料，標準 SQL 支援 DATETIME 的資料格式，包含了下列特別的資料格式：

```
DATE
TIME
INTERVAL
TIMESTAMP
```

DATETIME 資料格式包含下列元件：

```
YEAR
MONTH
DAY
HOUR
MINUTE
SECOND
```

註解

第二個元件可以簡化至秒的小數值，範圍為 00.000 到 61.999。

請小心，每一家 SQL 廠商有他們自己設定的日期與時間資料格式，之前的資料格式與元件，每一家廠商都有保持原本的標準，但對

於日期與時間的資料格式，每一家廠商都有不同的規格，而且實際呈現方的式與內部儲存的格式也不盡相同。

日期與時間的資料格式通常不會限制長度，等會兒您會學習到更多有關日期的資料格式，知道在其他廠商日期是如何儲存的，利用轉換功能來製造日期與時間，與學習日期與時間如何應用在實際情況。

2-2-8 文字字串（Literal Strings）

文字字串是一系列的字元，如姓名、電話，被使用者或程式明確的定義。文字字串與前面提到的資料格式有著相同的屬性，但是字串的值是事先知道的；因爲表格內的資料，每一欄是對應著每一列，所以通常每一列的本身值是未知的。

在文字字串內您無須特別指定資料格式，您只要簡單的指定字串就可以了。請參考下列的例子：

```
'Hello'
45000
'45000'
3.14
'November 1, 1997'
```

文數字串必須寫在二個單引號內，但是數值 45000 就不用。請注意第二個 45000 是寫在單引號內。一般而言，字元字串是必須寫在單引號內，數值字串就不用，您等會兒會看見文字字串如何與資料庫查詢一起使用。

2-2-9 NULL 資料格式

本小時已經大約提過 NULL 的概念，您應該知道 NULL 值是一個遺漏的值或是在欄位中的資料未被指定。NULL 值應用在 SQL 很多地方，如建立表格、條件式查詢與文字字串等，下列二個是使用 NULL 值的參考方式：

- NULL（NULL 本身就是關鍵字）
- '' 單引號之間沒有任何值

下列並不代表 NULL 值，但代表字元 N-U-L-L：

```
'NULL'
```

2-3 摘要

在 SQL 中有許多資料格式，如果您有程式開發的經驗，您應該對所提到的資料格式感覺很熟悉。資料格式允許不同形式的資料儲存在資料庫內，資料的形式從簡單的文字、小數點、日期與時間。資料格式的概念和其他電腦語言相同，比如在像使用 C 這種第三代語言來傳遞變數或是在 SQL 上開發關連式資料庫，當然各加廠商有自己一套的資料格式，但再使用的基本方式是相同的。

在規劃決定資料的格式、長度、與精準值時，都必須爲現在與未來的狀況做詳細的考慮。企業內部的規定與使用者將如何存取資料，都必須列入考慮，您應該知道資料本身的特性與資料在資料庫時正確的資料格式。

2-3-1 Q&A

Q 我要如何輸入個人社會福利號碼這方面的資料，到已定義為文字的欄位呢？

A 數值也是文數值的一種，而且可以輸入到文字的資料格式。通常需要運算的資料才會儲存爲數值。但有時爲了確定輸入的資料爲數值，因此也可以把一些欄位的資料格式定義爲數值。

Q 我還是不太瞭解固定與變化長度字元的資料格式的差異，請再多解釋？

A 比方說您有一個欄位爲記錄每個人的英文姓，定義爲固定長度 20 位元。當爲了記錄 Smith 這筆資料時，它佔用了 20 位元的資料，其中 5 位元個給 Smith，另外 15 位元記錄爲空格（請注意是固定長度資料格式），如果您是使用變化長度的話，它只佔用 5 位元的資料。

2-4 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

2-4-1 隨堂測驗

1. 是非題：個人社會福利號碼可屬於下列任何一種資料格式：固定長度字元、變化長度字元與數值。
2. 是非題：數值內的比例值是代表整個數值的長度。
3. 全部廠商的 SQL 都使用同樣的資料格式嗎？
4. 下列的精準值與比例值為何？

```
DECIMAL(4,2)
DECIMAL(10,2)
DECIMAL(14,1)
```

5. 下列哪一個數字可以被用在欄位定義為 DECIMAL(4,1)？

   a. 16.2

   b. 116.2

   c. 16.21

   d. 1116.2

   e. 1116.21

2-4-2 練習題

1. 使用下列欄位名稱，定義他們的資料格式與決定長度：

   a. ssn

   b. state

   c. city

   d. phone_number

   e. zip

   f. last_name

   g. first_name

   h. middle_name

   i. salary

   j. hourly_pay_rate

   k. date_hired

2. 使用同樣的欄位名稱並定義他們為 null 或 NOT NULL，有些欄位通常是 NOT NULL，有些欄位通常是由應用程式決定為 NULL 或兩者皆可。

   a. ssn

   b. state

   c. city

   d. phone_number

   e. zip

   f. last_name

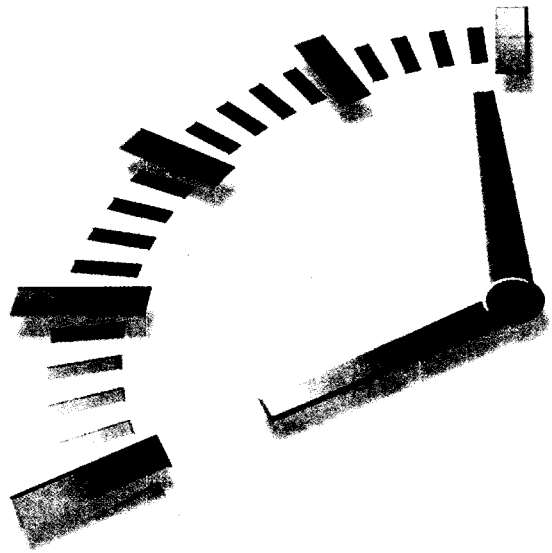
   g. first_name

   h. middle_name

   i. salary

   j. hourly_pay_rate

   k. date_hired



第 3 小時

管理資料庫物件

在這小時內您會學到有關資料庫物件：如什麼是物件資料庫、如何使用、互相的關聯是什麼等，資料庫物件是整個關連式資料庫的骨架。這些物件是資料庫邏輯單位，用來儲存資料且被稱爲後端資料庫（Back-end database），在這小時主要說明重點將環繞在表格，但請留意在其他小時將會提到許多資料庫物件。

本小時的重點包括：

- 資料庫物件介紹
- 資料庫結構（Schema）介紹
- 表格介紹
- 討論表格的屬性
- 表格處理與建立範例

- 討論表格儲存選項
- 參考完整性與資料一貫性的概念

3-1 什麼是資料庫物件（Database Object）

資料庫物件是在資料庫任何已定義的物件，用來儲存或參考資料，資料庫物件的範例如：表格、檢視、叢集、序列、索引與同義字。在第 3 小時將會著重在表格，因爲它是關連式資料庫儲存資料的最基本格式。

3-2 什麼是資料庫結構（Schema）

資料庫結構是集合資料庫物件（在本小時以討論表格爲主）並有特定資料庫使用者名稱，這使用者名稱爲資料庫結構擁有者，或爲相關物件群組使用者。在資料庫內，您或許有一個或多個資料庫結構，基本上，當任何使用者建立了一個物件時，他也建立了自己的資料庫結構，除非在各廠商的資料庫中有特別規定，不然資料庫結構包含一個表格與無數個資料物件，。

NEW TERM 資料庫結構（Schema）是集合資料庫內相同的資料庫使用者帳戶的相關物件。

比方說資料庫管理員提供您一個資料庫的使用者名稱與密碼，您的使用者名稱爲 USER1，現在您登錄到資料庫，建立了一個表格爲 EMPLOYEE_TBL，您的表格確實名稱爲 .EMPLOYEE.EMPLOYEE_TBL。表格的資料庫結構名稱爲 USER1，也是表格擁有者名稱，您剛爲資料庫結構建立了第一個表格。

使用資料庫結構的好處，是當您存取自己的表格時，您不用再參考資料庫結構名稱，例如，您可以用下列任何一種方式來參考您的表格：

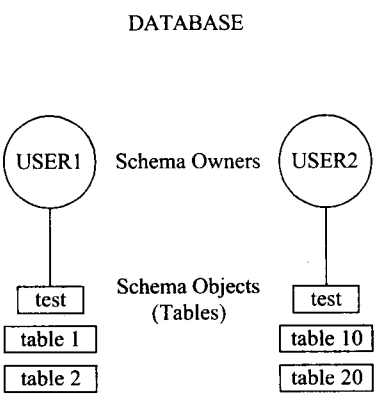
```
EMPLOYEE_TBL
USER1.EMPLOYEE_TBL
```

一般比較傾向使用第一行的方式，因爲打的字母較少，如果其他使用者想要查詢您其中的一個表格，他們就要指定資料庫結構，方式如下：

```
USER1.EMPLOYEE_TBL
```

在第 20 小時“建立與使用檢視和同義字”，您會學到有關分散權限的方法，讓其他使用者可以存取您的表格。您也可以學到有關同義字，您可以給表格另一個名稱，讓在做存取時，您不用去特定資料庫結構名稱。圖 3.1 顯示在關連式資料庫的二個資料庫結構。

圖 3.1.
資料庫中的資料庫結構



在圖 3.1 在資料庫內有二個使用者帳戶個別擁有自己的表格，USER1 與 USER2，每一個使用者帳戶有自己的資料庫結構，在下列一些範例讓您瞭解二位使用者如何存取自己的表格與其他使用者表格。

USER1 access own table1:	TABLE1
USER1 access own test:	TEST
USER1 access USER2's table10:	*USER2.TABLE10*
USER1 access USER2's test:	USER2.TEST

二個使用者都把表格名稱定爲 TEST，在資料庫中表格可有相同的名稱，只要他們屬於不同資料庫結構。由此可見，資料庫內表格名稱必須單一化，因爲資料庫結構擁有者就是表格名稱的一部份。例如，USER1.TEST 和 USER2.TEST 就不相同，如果您不特別指定資料庫結構與表格名稱，在存取資料庫表格時，資料庫伺服器會自動使用您自己擁有的表格，也就是說如果 USER1 試著存取 TEST，資料庫伺服器會先檢視 USER1 擁有的表格名稱 TEST，然後再檢視 USER1 擁有的其他物件，與在其他資料庫結構相同名稱的表格。在第 21 小時“系統型錄運作”希望您能徹底瞭解同義字的運用。

每一個資料庫伺服器在命名方面有一定的規則，好比欄位名稱，您可以詢問廠商有關命名或縮寫標準。

3-3 表格：儲存資料的主要地方

在關連式資料庫，表格主要是儲存資料、物件的地方。表格包含欄與列，表格的確會佔據資料庫的空間，這空間可爲永久或暫時的。

3-3-1 欄位與欄（Fields and Columns）

在關連式資料庫中欄位與欄是代表相同意思，都是表格內特定的資料格式，欄位名稱應和輸入欄資料的資料格式互相對應，欄可以設定爲 NULL 或 NOT NULL，如果設定爲 NOT NULL 爲則代表必須輸入部分資料，相對的如果設定爲 NULL 則不用輸入任何資料。

每一個資料庫表格最少必須有一行欄，欄所含的元件是表格內含有特定格式的資料，如個人名稱或住址，舉例說：在客戶表格一個合法的欄，可能爲客戶名稱。

一般而言，一個名稱必須爲一個連續性的字串，一個物件名稱也必須爲一個連續性的字串，依照每一家廠商的 SQL，字元長度可能

有限，通常在 SQL 都用底線來區別字元，比方說：客戶名稱可能被命名爲 CUSTOMER_NAME 而不是 CUSTOMERNAME。


註解

請和您的 SQL 廠商確認物件與其他元件命名規則。

3-3-2 列（Row）

在資料庫表格中，列用來是記錄資料，舉例說：在客戶表格內，列可能代表爲客戶 ID、名稱、住址、電話、傳真等。列構成了資料庫表格內的欄位，而欄位包含了每一列的資料，一個表格包含了一列的資料到千、百萬列的資料。

3-3-3 建立表格（CREATE TABLE STATEMENT）

從字面來看，CREATE TABLE 是用來建立一個表格，雖然建立一個表格很簡單，但在實際執行指令之前，表格的規劃、結構都必須花相當時間與精神。

建立一些基本表格之前，先依下列一些基本問題構思表格架構：

- 表格含有哪一些資料？
- 表格的名稱？
- 哪一欄（columns）將被設定為主索引（primary key）？
- 欄位（fields）的名稱？
- 每一欄的資料格式？
- 每一欄位長度？
- 哪一個欄必須包含資料？

等待您回答這些問題後，真正執行 CREATE TABLE 就很簡單了：

基本語法（syntax）如下：

```
CREATE TABLE table_name
( field1   datatype   [ not null ],
  field2   datatype   [ not null ],
  field3   datatype   [ not null ],
  field4   datatype   [ not null ],
  field5   datatype   [ not null ] )
```


註解

在本章的範例中，最常被使用的資料格式為 CHAR（固定長度字元）、VARCHAR（變化長度字元）、NUMBER（數值、小數和非小數值）、DATE（日期與時間）。

利用下列的範例，您可以建立一個表格命名爲 EMPLOYEE_TBL：

```
CREATE TABLE EMPLOYEE_TBL
(EMP_ID          CHAR(9)          NOT NUL,
EMP_NAME         VARCHAR(40)      NOT NULL,
EMP_ST_ADDR      VARCHAR(20)      NOT NULL,
EMP_CITY         VARCHAR(15)      NOT NULL,
EMP_ST           CHAR(2)          NOT NULL,
EMP_ZIP          NUMBER(5)        NOT NULL,
EMP_PHONE        NUMBER(10)       NULL,
EMP_PAGER        NUMBER(10)       NULL);
```

八個不同的欄組成了這個表格，請注意看如何使用底線來分開欄的名稱，如 EMPLOYEE ID 被稱爲 EMP_ID。使用 NULL/NOT NULL 的條件來指定每一個欄特別的資料格式與長度，您必須指定在表格內的每一列、哪一欄必須含有資料，如 EMP_PHONE 被定爲 NULL，代表資料不一定要被輸入，因爲不一定每一個員工都有電話。每一欄的資訊用逗號來區隔，所有欄都包含在括弧號內（左括弧在第一欄的前面，右括弧在最後欄的後面）。

分號是每一段指令（statement）的最後的字元，大多數的廠商都有一些字元來表示結束指令或送出指令給資料庫伺服器。

每一筆資料在表格內應包含：

```
EMP_ID, EMP_NAME, EMP_CITY, EMP_ST, EMP_ZIP, EMP_HOME, EMP_PAGER
```

在表格內會因資料查詢或異動的要求，EMP_ID 爲一個員工或許多員工的代號，**Column** 是垂直的資料而 **Row** 是平行的資料。


註解

Column 的預設值為 NULL；因此在 CREATE TABLE 定義時不用特別設定。

3-3-4 storage 子句

在許多家廠商的 SQL 都有一些 storage 子句格式，storage 子句通常是建立表格時，使用在設定表格大小，下列 storage 子句的語法範例，是被其中的一家廠商所使用：

```
CREATE TABLE EMPLOYEE_TBL
(EMP_ID          CHAR(9)          NOT NULL,
EMP_NAME         VARCHAR(400      NOT NULL,
EMP_ST_ADDR      VARCHAR(20)      NOT NULL,
EMP_CITY         VARCHAR(15)      NOT NULL,
EMP_ST           CHAR(2)          NOT NULL,
EMP_ZIP          NUMBER(5)        NOT NULL,
EMP_PHONE        NUMBER(10)       NULL,
EMP_PAGER        NUMBER(10)       NULL)
STORAGE
    (INITIAL      3K
     NEXT         2K );
```

在一些廠商中，有提供許多 storage 子句選項，INITIAL 分配一組資料空間，用來設定最初表格所使用空間，單位爲 byte、kilobyte 等。NEXT 是用來說明比當初預設表格時多使用的空間，您會發現 storage 子句有其他許多的選項，也請記住這些選項會因廠商不同而有所更動。如果一些重要的廠商忽略了 storage 子句，而只有使用預設變數，這樣的程式不被認爲是最好的應用程式。

請注意陳述 CREATE TABLE 時，第一行要縮排而且儘量簡潔，方便閱讀與日後更改與修正。


註解

storage 子句在每個 SQL 關連式資料庫使用方法都不同，上一個範例是使用在 Oracle，加在 CREATE TABLE 陳述中。請注意，ANSI 對 SQL 所設的標準，只是一個標準而已，標準本身並不是一種程式語言，但卻是廠商開發設計自己 SQL 時所採用的基本法則，就好比每一家廠商的資料格式也不盡相同，大多數的廠商都考慮如何把資料做最好的儲存與處理。

3-3-5 命名方式

當選擇物件、表格名稱時，名稱通常代表被儲存資料的特性，比方說：員工資料附屬表格，就應被命名爲 EMPLOYEE_TBL，同樣邏輯應用在命名欄位，如員工電話的欄位就應被命名爲 PHONE_NUMBER。


註解

由於每家廠商名稱長度不盡相同，請先查詢您所使用廠商所設定名稱的長度。

3-3-6 ALTER TABLE 指令

表格建立後，可用 ALTER TABLE 來進行更改，您可以增加欄位、修改欄位、更改欄位定義，增加、修改條件，有些廠商更提供如何更改表格儲存值，ALTER TABLE 指令標準語法爲：

```
alter table tale name [modify] [column column_name]
[datatype|null not null][restrict|cascade]
                      [drop]    [constraint constraint_name]
                      [add]     [column] column definition
```

3-3-7 修改表格元件

使用 ALTER TABLE 指令，您可以修改欄位屬性，屬性的定義爲：

- 欄位資料格式
- 欄位長度、精準性或比例
- 欄位有無包含 NULL 值

NEW TERM 欄位的屬性（attributes）爲欄位中資料的規則和行爲。

下列範例是使用 ALTER TABLE 指令來修改 EMPLOYEE_TBL 內，EMP_ID 的屬性：

```
ALTER TABLE EMPLOYEE_TBL MODIFY (EMP_ID CARCHAR2(10))

TABLE ALTERED
```

這個欄位原本被定義爲 VARCHAR2 資料格式（變動長度字元），但已經把最大長度從 9 增加到 10。

3-3-8 增加必須欄位到表格中

增加欄位到現有表格的基本條件之一是：如果現有表格含有資料，您要增加的欄位不得定義爲 NOT NULL。NOT NULL 是表示表格內每一欄位一定包含一些資料，如果您新增的欄位定義爲 NOT NULL，那麼您就違背了 NOT NULL 條件，先前表格內的欄位就不一定有資料給對應的新欄位。

但是如果必須加入欄位到表格內，可參考下列方法：

1. 加入被定義為 NULL 欄位（欄位不一定要有資料）。
2. 插入新資料到每一個表格內的新欄位。
3. 在確定表格內每一欄位包含了資料後，您可以更改欄位的屬性為 NOT NULL。

3-3-9 修改欄

在修改現有表格欄時，要考慮許多地方，如下：

修改表格常用規則：

- 欄位長度可以增加到資料格式所限定最大長度。
- 欄位長度只能縮短，如果欄內最長資料的長度，小於或等於新欄的長度。
- 資料格式為數值時，數位可以永遠增加。
- 資料格式為數值時，數位只能被縮短如果欄內最長數值的長度，小於或等於特定新欄的長度。
- 小數點數可以隨意被增加或縮短。
- 通常欄的資料格式可以被更改。

部分廠商會限制使用 ALTER TABLE 選項，有時您可能無法取消表格內的欄位，要做到這樣，您可以先取消表格本身，然後再重建表格與想要保留的欄位，如果您取消欄位內的資料是參考或依賴其他表格內欄位的資料，您可能會遇到一些問題，請參考廠商所提供的文件來解決。

3-3-10 從現有表格建立表格

一起使用 CREATE TABLE 和 SELECT 指令可以複製現有表格，新表格有著相同欄位定義，部分或全部的欄位可以被選取，新欄位可以透過函數或自動連結欄位被建立，並預設大小會自動設定，從其他表格建立表格基本語法如下：

```
create table new_table_name as
select [ *|column1, column2 ]
from table_name
[ where ]
```

請注意語法中所使用的關鍵字，特別是 SELECT，SELECT 是資料庫查詢語言，等會將加以介紹，但重要的是您可以使用查詢的方式來建立一個表格。

首先，建立一個簡單查詢去檢視 PRODUCT_TBL 表格的資料。

TYPE
```
select * from products_tbl;
```

Output
```
PROD_ID    PROD_DESC                   COST
---------- ---------------------- ---------
11235      WITCHES COSTUME            29.99
222        PLASTIC PUMPKIN 18 INCH     7.75
13         FALSE PARAFFIN TEETH        1.1
90         LIGHTED LANTERNS           14.5
15         ASSORTED COSTUMES          10
9          CANDY CORN                  1.35
6          PUMPKIN CANDY               1.45
87         PLASTIC SPIDERS             1.05
119        ASSORTED MASKS              4.95
```

註解

SELECT * 表示選取表格內所有欄位的資料。

下一步，使用上個查詢來建立一個爲 PRODUCT_TMP 的表格：

```
create table product_tmp as
select * from product_tbl;

table created
```

現在，如果您執行查詢，您的結果會和您選取原來表格一樣。

TYPE
```
select *
from products_tmp;
```

Output

```
PROD_ID    PROD_DESC                    COST
---------- ---------------------------- ---------
11235      WITCHES COSTUME              29.99
222        PLASTIC PUMPKIN 18 INCH       7.75
13         FALSE PARAFFIN TEETH          1.1
90         LIGHTED LANTERNS             14.5
15         ASSORTED COSTUMES            10
9          CANDY CORN                    1.35
6          PUMPKIN CANDY                 1.45
87         PLASTIC SPIDERS               1.05
119        ASSORTED MASKS                4.95
```

註解

當從現有存在表格建立另一個表格時，新表格會承繼原有表格相同的儲存屬性。

3-3-11 取消表格（Drop Tables）

取消表格是最容易做的事，當使用 restrict 選項與表格被條件或檢視使用時，使用 drop 指令會產生錯誤訊息，當使用 cascade 選項時，drop 指令可以順利執行，所有條件或檢視也會被取消。其語法如下：

```
drop table table_name [ restrict | cascade ]
```

在下列範例，您取消了您剛建立的表格：

```
drop table products_tmp

table dropped.
```

注意

不論是否取消表格，在送出指令前，請確認有指定資料庫結構或表格擁有者，您有可能取消不正確的表格，如果您可以存取許多使用者帳號，請確認您透過正確帳號去取消您所想取消的表格。

3-4 完整性條件（Integrity Constraints）

完整性條件是用來確認在關連式資料庫的資料正確性與一致性，在關連式資料庫內，資料完整性是透過參考完整性觀念，在參考完整性（Referential Integrity, RI）內，有許多完整性條件的格式佔有重要地位。

3-4-1 主索引條件（Primary Key Constraints）

主索引是用來識別表格內，一個或多個資料單一的欄位，雖然主索引通常是指表格內的一個欄位，多於一個欄位也可以組成一個主索引，舉例說：在員工資料表格內，可以同時選員工社會福利號碼或員工編號爲主索引，主要目的讓每一筆記錄，有個單一性主索引或值給員工編號，因爲在每一個員工資料表格中，每一個員工資料不需要被重複，員工編號就可以成爲主索引，主索引在建立表格時就被設定了。

在下列範例中，EMP_ID 是員工資料表格的主索引。

TYPE
```
CREATE TABLE EMPLOYEE_TBL
(EMP_ID          CHAR(9)          NOT NULL PRIMARY KEY,
EMP_NAME         VARCHAR2(40)     NOT NULL,
EMP_ST_ADDR      VARCHAR2(20)     NOT NULL,
EMP_CITY         VARCHAR2(15)     NOT NULL,
EMP_ST           CHAR(2)          NOT NULL,
EMP_ZIP          NUMBER(5)        NOT NULL,
EMP_PHONE        NUMBER(10)       NULL,
EMP_PAGER        NUMBER(10)       NULL);
```

這方式是在建立表格時，先定義主索引，在這主索引有被設定條件，如下，在設定表格時，您可以設定主索引特定的條件。

TYPE
```
CREATE TABLE EMPLOYEE_TBL
(EMP_ID          CHAR(9)          NOT NULL,
EMP_NAME         VARCHAR2(40)     NOT NULL,
EMP_ST_ADDR      VARCHAR2(20)     NOT NULL,
EMP_CITY         VARCHAR2(15)     NOT NULL,
EMP_ST           CHAR(2)          NOT NULL,
EMP_ZIP          NUMBER(5)        NOT NULL,
```

```
EMP_PHONE       NUMBER(10)      NULL,
EMP_PAGER       NUMBER(10)      NULL);
PRIMARY KEY (EMP_ID);
```

主索引條件在這範例中，是在 CREATE TABLE 時，在逗號之後的敘述所定義。

3-4-2 單一性條件（Unique Constraints）

表格內欄單一性條件和主索引是非常類似，表格內每行與列的資料必須是單一性的，當主索引條件被放置在一個欄時，您可以放一個單一條件在另一個欄，不管它是不是被當作主索引來使用。

請參考下列範例：

```
CREATE TABLE EMPLOYEE_TBL
(EMP_ID         CAHR(9)         NOT NULL      PRIMARY KEY,
EMP_NAME        VARCHAR2(40)    NOT NULL,
EMP_ST_ADDR     VARCHAR2(20)    NOT NULL,
EMP_CITY        VARCHAR2(15)    NOT NULL,
EMP_ST          CHAR(2)         NOT NULL,
EMP_ZIP         NUMBER(5)       NOT NULL,
EMP_PHONE       NUMBER(10)      NULL          UNIQUE,
EMP_PAGER       NUMBER(10)      NULL);
```

在這範例中，EMP_ID 是主索引，代表員工編號是用來確保表格內每一筆記錄都是單一的。主索引通常都是用在查詢，特別是連結表格。EMP_PHONE 被指定爲 UNIQUE 單一值，代表沒有二位員工可有相同電話號碼，二者並沒有很大差異，除了主索引是用來排序資料與連結相關表格。

3-4-3 外部索引條件（Foreign Key Constraints）

外部索引條件是主要結構確認關連式資料庫表格的參考完整性，當一個欄被定義爲外部索引時，代表在參考其他表格時，那個欄將被認定爲主索引。

NEW TERM 外部索引（Foreign Key）是指子表格的欄參考父表格的欄。

請參考下列 Foreign Key 範例：

```
CREATE TABLE EMPLOYEE_PAY_TBL
(EMP_ID              CHAR(9)          NOT NULL,
POSITION             VARCHAR2(15)     NOT NULL,
DATE_HIRE            DATE             NULL,
PAY_RATE             NUMBER(4,2)      NOT NULL,
DATE_LAST_RAISE      DATE             NULL,
FOREIGN KEY EMP_ID_FK (EMP_ID) REFERENCES EMPLOYEE_TBL (EMP_ID);
```

在這範例中，EMP_ID 被定義爲和表格 EMPLOYEE_PAY_TBL 的外部索引，您可以看出 EMP_Id 參考 EMPLOYEE_TBL 爲外部索引，外部索引確認在 EMPLOYEE_PAY_TBL 表格的每一個 EMP_ID 都和 EMPLOYEE_TBL 的 EMP_ID 相應，這被稱爲父子關係（parent/child relationship），父表格爲 EMPLOYEE_TBL，子表格爲 EMPLOYEE_PAY_TBL。看圖 3.2 會對父子關係（parent/child relationship）有更清楚的概念。

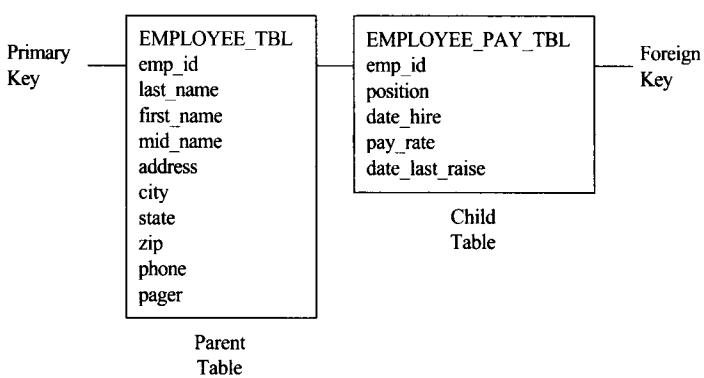


圖 3.2. 父子關係（parent/child relationship）

在這圖表，子表格 EMP_ID 參考著父 EMP_ID 表格，當要插入一個值至子表格 EMP_ID 時，必須在父表格 EMP_ID 有個值存在，同樣的道理，當要從父表格 EMP_ID 移除資料時，所有在子表格 EMP_ID 的相關資料必須先移除，這就是參考完整性（Referential Integrity）。

外部索引可利用 ALTER TABLE 指令被加入表格，範例如下：

```
alter table employee_pay_tbl
```

```
add constraint ad_fk foreign key (emp_id)
references employee_tbl (emp_id);
```


註解

由於廠商的不同，alter table 指令選項也跟著不同，特別是和條件一起使用時，此外真正的使用方式與條件的定義也不同，但所有關連式資料庫基本參考完整性的概念卻是一樣的。

3-4-4 NOT NULL 條件

前個範例使用關鍵字 NULL 與 NOT NULL 放在每一個欄的資料格式後面，NOT NULL 可以放在表格欄內成爲一個條件，這條件使得資料不一定要被輸入到欄位內，換句話說當欄位設定爲 NOT NULL 時，資料一定要被輸入到表格的欄位中，NULL 通常爲預設值，讓欄位不一定要輸入資料。

3-4-5 使用檢查條件（Using Check Constraints）

檢查條件用來檢查被輸入表格的資料是合法的，檢查條件用來提供資料庫後端（back-end）編輯，雖然編輯通常也在應用程式的前端使用，一般編輯在資料庫本身或視窗應用程式，會限制被輸入欄位或物件的值，檢查條件是另一種方式來保證輸入資料格式。

下列範例展示了檢查條件的使用方式：

```
CREATE TABLE EMPLOYEE_TBL
(EMP_ID          CHAR(9)          NOT NULL,
EMP_NAME         VARCHAR2(40)     NOT NULL,
EMP_ST_ADDR      VARCHAR2(20)     NOT NULL,
EMP_CITY         VARCHAR2(15)     NOT NULL,
EMP_ST           CHAR(2)          NOT NULL,
EMP_ZIP          NUMBER(5)        NOT NULL,
EMP_PHONE        NUMBER(10)       NULL,
EMP_PAGER        NUMBER(10)       NULL),
PRIMARY KEY (EMP_ID),
CONSTRAINT CHK_EMP_ZIP CHECK ( EMP_ZIP = '46234');
```

在這表格內檢查條件被放置在 EMP_ZIP，確認所有員工輸入到這表格有著郵遞區號 46234，雖然這只是小小的示範，但您應該可以看出它是如何工作的。

如果您想要使用檢查條件來確認郵遞區號的值是在清單內，您應設定如下條件：

```
CONSTRAINT CHK_EMP_ZIP CHECK ( EMP_ZIP in (`46234`,`46227`,`46745`));
```

如果員工最少應付的稅也要被設計進去，您的條件應設定如下：

```
CREATE TABLE EMPLOYEE_PAY_TBL
(EMP_ID              CHAR(9)          NOT NULL,
POSITION             VARCHAR2(15)     NOT NULL,
DATE_HIRE            DATE             NULL,
PAY_RATE             NUMBER(4,2)      NOT NULL,
DATE_LAST_RAISE      DATE             NULL,
FOREIGN KEY EMP_ID_FK (EMP_ID) REFERENCES EMPLOYEE_TBL (EMP_ID),
CONSTRAINT CHK_PAY CHECK ( PAY_RATE > 12.50 ) );
```

在這個範例，表格所輸入的任何員工每小時一定要付多於 $ 12.50 的稅，您可以使用任何條件在檢查條件，相同的在任何 SQL 查詢也可以，等會您會學到更多。

3-5 摘要

您已經學到有關資料庫物件的概念，特別是有關表格。在關連式資料庫中，表格是最簡單儲存資料的格式，表格包含群組邏輯資料，如員工、客戶與產品資料，表格是由取多欄位所組成，每個欄位有著不同屬性，每個屬性代表資料的資料格式與設定的條件如 NOT NULL，主索引、外部索引與單一值。您學到了 CREATE TABLE 指令與選項，如儲存參數與其他可和這指令一起用的參數。您也學到如何利用 ALTER TABLE 指令更改現有表格結構，雖然在 SQL 管理資料庫或許不是最基本的，卻是最重要的，如果您先學表格的結構和屬性，您會比較容易領會如何存取表格，不論是透過資料處理或是資料庫查詢，在下個小時，您會學到如何管理 SQL 內的其他物件如與檢視。

3-5-1 Q&A

Q 當我為我建立的表格命名時，字尾一定要用 _tbl 嗎？

A 當然不用，您甚至不用使用字尾，比方說，當表格用來記錄員工資料時，您可以使用下列名稱或任何您所喜歡名稱：

```
employee
emp_tbl
employee_tbl
employee_table
worker
```

Q 為什麼在取消表格時，要使用資料庫結構名稱？

A 下列是有關一位新的資料庫管理員要取消一個表格的真實故事。一位程式設計師在他資料庫結構下建立個和產品表格名稱相同的表格，當這位程式設計師離職時，他的資料庫帳號被刪除，但是 DROP USER 回報錯誤訊息，因爲程式設計師所擁有的物件依舊存在，在詳細調查後，發現程式設計師的表格式不需要的，所以使用 DROP TABLE 來取消表格。

看起來很迷惑，但問題在資料庫管理員在執行 DROP TABLE 時登錄在產品資料庫結構下，資料庫管理員在取消表格時，必須指定資料庫結構名稱或擁有人。從上看來，大約要 8 小時來復原產品資料庫。

3-6 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

3-6-1 隨堂測驗

1. 下列 CREATE TABLE 指令能否正常執行？如果不行，請修正該修正的地方。

   Create table employee_table as :

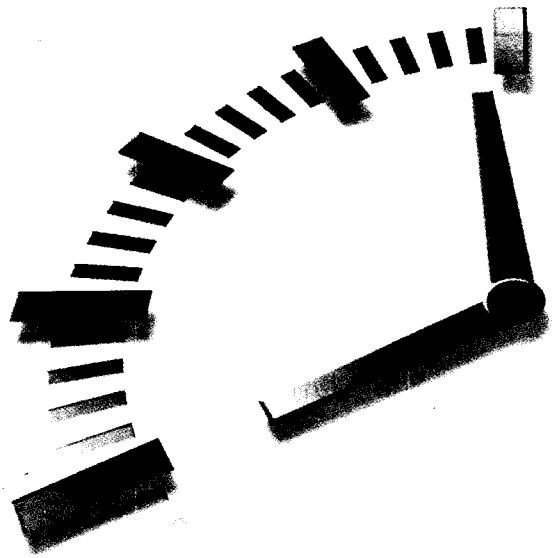
```
( ssn              number(9)        not null,
 last_name         varchar2(20)     not null,
 first_name        varchar2(20)     not null,
 middle_name       varchar2(20)     not null,
 st address        varchar2(30)     not null,
 city              char(20)         not null,
 state             char(2)          not null,
 zip               number(4)        not null,
 date hired        date)
 storage
     (initial         3k,
      next          1k);
```

2. 您可以從表格取消一個 column 嗎？

3. 如果您沒有放入 storage clause 在 create table 指令內，會發生什麼？

3-6-2 練習題

1. 到附錄 D 用本書所使用的 DLL，建立一個表格。



第 4 小時

正常化程序

在這小時，您會學到如何將原始資料轉換成邏輯單位 - 表格，這程序稱爲正常化程序。

資料的正常化程序與解除正常化程序優、缺點，資料完整性對執行效能等問題在這都會被提到。

本小時的重點包括：

- 什麼是正常化程序？
- 正常化程序益處
- 解除正常化程序優點
- 正常化程序的技巧
- 正常化程序指引

- 三種正常形式
- 資料庫設計

4-1 正常化資料庫(Normalizing Database)

正常化是一個縮減資料庫內重複資料的程序，此外，資料庫的資料名稱、物件名稱與形式也可以被正常化。

NEW TERM 正常化（normalization）是一個縮減關連式資料庫內重複資料的程序。

4-1-1 原始資料庫（Raw Database）

一個資料庫若沒有被正常化那麼資料可能無故出現一個或多個的表格，這樣對安全考量、磁碟使用、查詢速度、資料更新與最重要的-資料完整性都非常不好，一個資料庫在尚未被正常化前，是被認定爲尚未被分割成更小的邏輯與方便管理的表格。圖 4.1 展示了本書所使用的資料庫在尚未被正常化時的情況。

圖 4.1. 原始資料庫

COMPNAY_DATABASE	
emp_id	cust_id
lasy_name	cust_name
first_name	cust_address
middle_name	cust_city
address	cust_state
city	cust_zip
state	cust_phone
zip	cust_fax
phone	ord_num
pager	qty
position	ord_date
date_hire	prod_id
pay_rate	prod_desc
date_last_raise	cost

4-1-2 邏輯資料庫設計

在設計任何資料庫時，應先考慮使用者的立場，邏輯資料庫設計也被稱爲邏輯模式，是一種安排資料成爲有邏輯的程序，被整理過的群組物件可以很容易被維護，資料庫的邏輯設計應該減少資料的重複性或到完全杜絕，爲什麼要重複儲存呢？所以資料庫命名應有固定正常與邏輯。

什麼是使用者需求？

在設計資料庫時，使用者需求應該是優先考慮的條件之一，請注意，使用者是最終使用資料庫的人，這應該是讓使用者容易使用的前端工具（程式讓使用者可以存取資料庫），如果使用者需求沒被考慮進去，就不能達到最佳執行效能。

一些使用者相關的考量爲：

- 什麼資料要被儲存到資料庫？
- 使用者如何存取資料庫？
- 使用者需要什麼權限？
- 在資料庫內，資料要如何被分群組？
- 哪些資料最常被存取？
- 資料庫內所有資料的關係如何？
- 應採取什麼方式來確保資料被正確鍵入？

資料重複（Data Redundancy）

資料不應該有重複，所以在任何狀況下資料應盡量避免被重複鍵入，比方說，員工住家地址應該被儲存一個表格就可以了，因爲資料重複會占去不必要的磁碟空間，常常會被下列狀況搞混：員工住家地址與另一個表格內資料不符合，哪一個才是對的？有任何文件可以確

認員工目前住址嗎？因爲如果資料管理不夠嚴謹，重複資料就成爲一種災難了！

4-1-3 正常形式（Normal Form）

下個章節會討論正常形式，整個概念涉及到資料庫正常化。

NEW TERM 正常形式（Normal Form）是測量資料庫被正常化的程度與深度，資料庫被正常化的程度是由正常形式所決定。

三種在正常化程序最常用的正常形式：

- 第一正常形式
- 第二正常形式
- 第三正常形式

在這三正常形式中，每種正常形式在於上一個正常形式正常化所採取的步驟，比如說：要正常化第二正常形式前，一定要先正常化第一正常形式。

第一正常形式

第一正常形式的主要目就是將基本資料分割成爲邏輯單位-表格，當設計每一個表格時，主引索大多也被設定好了，檢視一下圖 4.2，顯示如何將上圖所用的原始資料發展成第一正常形式。

您可以看見要完成第一正常形式，資料一定要被分割成爲邏輯單位，每一個表格都有主引索來確保在任何表格內，沒有一個群組是重複的。在這採用了幾個較小、較容易管理的表格 EMPLOYEE_TBL、CUSTOMER_TBL 與 PRODUCTS_TBL 來取代一個大表格，主引索通常是表格的第一個 column，在這爲 EMP_ID、CUST_ID 與 PROD_ID。

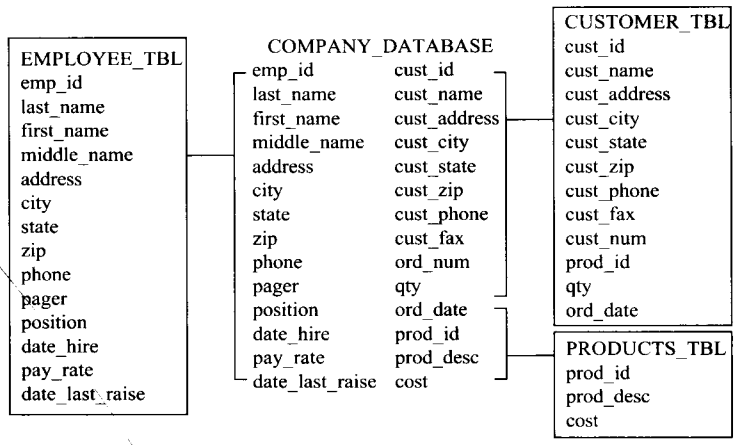


圖 4.2.
第一正常形式

第二正常形式

第二正常形式的目的是將部分只依賴主引索資料並將那資料輸入到另外表格。

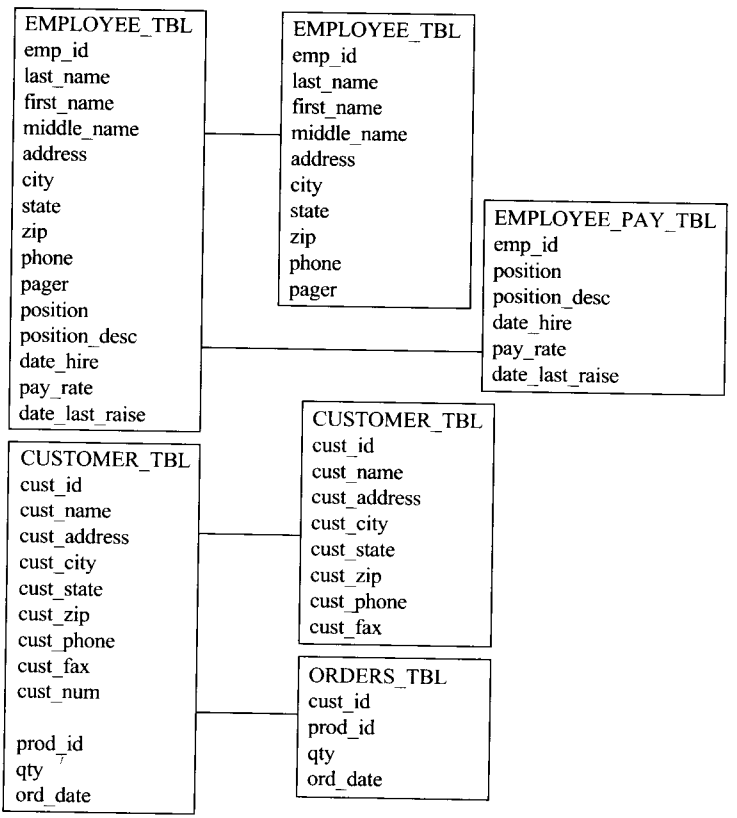
依照圖，第二正常形式起源於第一正常形式，把二個表格分裂稱更小的特定單位。

EMPLOYEE_TBL 分裂到二個表格之內稱爲 EMPLOYEE_TBL 和 EMPLOYEE_PAY_TBL。員工私人資訊依賴著主引索，或 EMP_ID，所以訊息依然保持在 EMPLOYEE_TBL（EMP_ID、LAST_NAME、FIRST_NAME、MIDDLE_NAME、ADDRESS、CITY、STATE、ZIP、PHONE **and PAGER**）。另一方面，只有部分訊息依靠 EMP_ID（每位個別的員工）被公佈在 EMPLOYEE_PAY_TBL（EMP_ID、POSITION、POSITION_DESC、DATE_HIRE、PAY_RATE、DATE_LAST_RAISE），請注意因爲兩個表格都包含 **column** CUST_ID。這是每個表的主引索並且被用在對應在二個表格之間的資料。

CUSTOMER_TBL 分裂爲二個表格稱爲 CUSTOMER_TBL 和 ORDERS_TBL，發生什麼東西在 CUSTOMER_TBL 也會發生在 EMPLOYEE_TBL，部分依賴

的主引索的 column 被指到另一個不同的表格，客戶資訊是依賴著每個 CUST_ID，但是不直接地依賴最初一般客戶表格的資訊。

圖 4.3.
第二正常形式

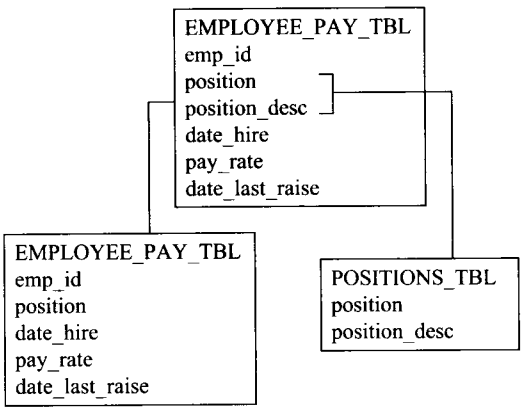


第三正常形式

第三正常形式目的是遷除表格內的資料是不依靠主引索上，圖 4.4 舉例說明第三正常形式。

另外的表格被建立來顯示第三正常形式，EMPLOYEE_PAY_TBL 是分裂到二個表格內，一個表格包含真實的員工薪資資訊息和另一個表格包含職位說明，實際上職位說明不需要放在 EMPLOYEE_PAY_TBL，POSITION_DESC column 完全地是獨立的，不依靠主引索 EMP_ID。

圖 4.4.
第三正常形式



4-1-4 命名協定

命名協定是之前作業考慮因素之一，當您正常化資料庫的時候，您希望您的表格名稱是敘述他們所包含的資訊類型，公司內的命名協定應該被設定，在資料庫表格內，不只能提供命名標準，也是使用者，檔案名稱和其他相關物件的命名協定，設計與強迫執行命名協定是一家公司邁向實施成功的資料庫的第一步驟。

4-1-5 正常化的好處

正常化提供很多好處給資料庫，一些主要的好處包括：

- 組織比較主要的資料庫
- 縮減多餘的資料
- 在資料庫裡面的資料是一致
- 更便利的資料庫設計
- 較好處理資料庫的安全性

組織被正常化程序使得每個人工作更容易，從對使用者存取表格到資料庫管理員所負責管理的資料庫全部的物件，多餘資料被減少，那會簡單化資料結構而且保存磁碟空間，因爲重複的資料被減到最

少，不合理的資料發生的可能性也就相對的減少，舉例來說，在一個表格內，個人名稱可能讀爲 STEVE SMITH，相同人的名稱可能外部表格稱爲 STEPHEN R. SMITH，因爲資料庫已經被正常化而且強行分裂成比較小的表格，您會有比較多彈性來修改現有結構，修改較小的表格含有較少的資料是比修改一個保存所有重要資料的表格與較大資料庫來得容易的。最後，安全性使得資料庫管理員能授與某一個使用者有限制的存取特定表格，當正常化已經存在的時候，安全性的控制也就比較容易。

NEW TERM 資料完整性是保証在資料庫裡面的資料是一致的。

參考完整性

參考完整性只是表示表格內一個 **column** 的數值是因另外一個表格 **column** 數值而定，舉例來說，若要找一個客戶在 ORDERS_TBL 表格裡的記錄，就先必須在 CUSTOMER_TBL 表格裡有存在那個客戶的記錄，完整性的條件限制也能限定 **column** 的數值在一個可範圍控制，完整性條件限制應該在建立表格時就被建立，參考完整性是典型地透過主引索與外部引索來控制的。

NEW TERM 在表格裡，外部引索通常是單一欄位，直接地參考另外表格裡的欄位來執行參考完整性，在前述的段落，ORDERS_TBL 裡的 CUST_ID 是外部引索，參考著 CUSTOMER_TBL 裡的 CUST_ID 的資料。

4-1-6 正常化的壞處

雖然大多成功的資料庫是做了某程度的正常化，但是一個實際上的壞處是針對正常化後的資料庫：降低資料庫執行效率。要瞭解會降低執行效率的原因，當資料庫提出異動、查詢等需求時，必須考慮到像是中央處理器、記憶體的使用量與，資料輸入／輸出等因素，爲了要使冗長的故事變短，被正常化的資料庫比 **denormalized** 資料庫，需要更多的中央處理器、記憶體、資料輸入／輸出來處理異動與資料

庫查詢，被正常化的資料庫必須找出哪些需求的表格，然後加入來自哪些表格的資料得到被請求的資訊或處理被要求更改的資料，關於較深入的討論資料庫執行效率，在第 18 個小時“管理資料庫使用者”會提到。

4-1-7 Denormalizing 資料庫

Denormalizing 資料庫：爲什麼您會這麼做？企圖改善資料庫執行效率是 denormalize 的唯一理由，denormalized 資料庫是與未被正常化的資料庫是不同的，Denormalizing 資料庫是在資料庫正常化的程序裡面下降一或二個層次，請記得，在經常性表格結合運算中，正常化實際上可能降低執行效率（在第 13 小時“結合查詢表格”會討論），Denormalization 可能包括再結合分開的表格，或在表格裡面建立複製資料來減少被要求資料的需求，因此造成比較少的輸入/輸出和中央處理器所需要的時間來讀取所需要的資料。

NEW TERM Denormalization 是用被正常化的資料庫和修改表格結構來控制增加資料庫執行效率的程序

然而 denormalization 是有代價的，多餘的資料在 denormalized 資料庫會被增加，denormalized 資料庫可以改良執行效率，但是更需要外來的努力，來明瞭追蹤有關連的資料，資料已經在各種不同的表格展開，所以應用程式的程式碼比較難找出特定的位置而導致更複雜化，除此之外，參考完整性更是屬於繁瑣的工作；有關連的資料已經被分散在許多表格之間，在正常化和 denormalization 裡有個良好的媒介，但是兩者需要對實際的資料 和公司業務特定的需求有徹底的瞭解。

4-2 摘要

對於決定資料庫是否要正常化非常困難的，在某程度上，您會希望正常化資料庫，在不破壞執行效率之下，您要正常化資料庫到甚麼

程度，真正決定在應用程式本身。資料庫多大？它的目的是什麼？什麼樣類型的使用者將要存取資料？這個小時涵蓋三種最通常的正常形成，哪些是在正常化程序之後的觀念與資料的完整性。正常化程序包括許多步驟，大部份最正常程度爲高效總能有個基本的設定，不論是質疑在簡單的維護和較差的執行效率，或是複雜的維護與較好的執行效率較好。在最後，設計資料庫的人員必須做決定，所以他們的責任更重大的。

4-2-1 Q&A

Q 當設計資料庫的時候，為什麼我必須以一般使用者的需要為考量？

A 因爲一般使用者是使用資料庫的行家，也因此，在設計資料庫時，應尊重他們的意見，資料庫設計者只能幫助他們組織資料。

Q 對我而言，正常化似乎比 denormalization 更有利的，您同意嗎？

A 它可能是更有利的，然而，有時 denormalization 可能是更有利的，請記得，有許多因素決定應該選那一個方向，您或許可以減少在正常化後資料庫裡資料的重複，但是相反的可能會 denormalize 可能可以改良執行效率。

4-3 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

4-3-1 隨堂測驗

1. 是非題：正常化是將群組資料分成為合乎邏輯與有關連群組的程序。
2. 是非題：資料庫裡沒有重複或多餘的資料，資料庫裡每件事物被正常化是最好方法。
3. 是非題：如果資料是在第三正常形式，它自動地是在第一與第二正常形式。
4. 什麼是正常化資料庫大於 denormalized 資料庫的主要優點？
5. Denormalization 的主要缺點是什麼？

4-3-2 練習題

1. 您正在為一間小的公司發展新的資料庫，使用下列各項資料做正常化，請注意您除了在它所提供的項目外，小公司還有許多的項目。

   員工：

   Angela Smith, secretary, 317-545-6789, RR 1 Box 73, Greensburg, Indiana, 47890, $9.50 hour, date started January 22, 1996, SSN is 323149669.

   Jack Lee Nelson, salesman, 3334 N Main St, Brownsburg, IN, 45687, 317-852-9901, salary of $35,000.00 year, SSN is 312567342, date started 10/28/95.

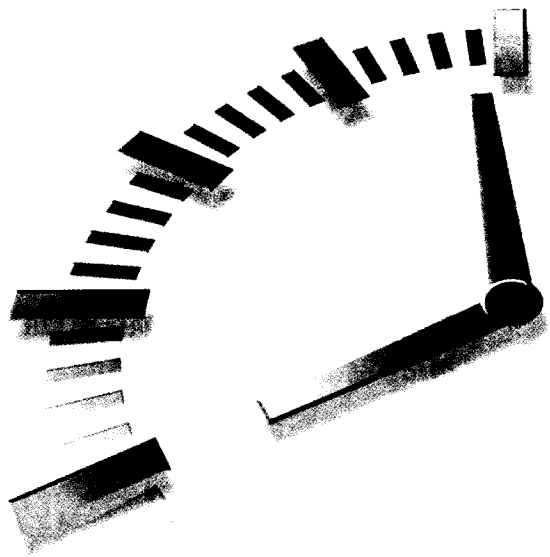
   客戶：

   Roberts Games and Things, 5612 Lafayette Rd, Indianapolis, IN, 46224, 317-291-7888, customer ID is 432A.

   Reeds Dairy Bar, 4556 W 10th St, Indianapolis, IN, 46245, 317-271-9823, customer ID is 117A.

   客戶訂單：

   Customer ID is 117A, date of last order is February 20, 1997, product ordered was napkins and the product ID is 661.



第 5 小時

處理資料

在這個小時，您將學習 SQL 的另一部份，稱爲資料處理語言（Data Manipulation Language DML），是用來更改表格內資料和關連式資料庫裡的表格。

本小時的重點包括：

- 資料處理語言的概觀
- 該如何處理表格裡的資料
- 表格資料背後的觀念
- 如何刪除表格的資料？
- 如何更改或修改表格裡的資料？

5-1 處理資料概觀

資料處理語言(DML)是 SQL 的一部份，用來允許資料庫使用者傳送對關連式資料庫資料的變更，使用 DML 時使用者可以公布已更新的表格資訊如最新的資料、更新現有表格裡的資料和刪除來表格的資料，DML 指令可以執行簡單的資料庫查詢，SQL 有三個基本 DML 指令。

INSERT 插入
UPDATE 更新
DELETE 刪除

Select 指令也可以與 DML 指令一起使用，在第 7 小時“介紹資料庫查詢”會更詳細地討論。

5-2 載入新資料到表格

載入新資料到表格(Populating a Table)簡單的說就是輸入新資料進入表格內，不論是藉由個人指令程序、使用批次程式或其他相關軟體。

NEW TERM **Populating a table** 是載入新資料的程序。

有許多因素可能影響什麼資料與多少資料可以被放在居住的表格，一些主要的因素包含，已存在的表格條件限制、實際的表格大小、欄位的資料形式、欄位的長度和其他完整性條件，例如主引索與外部引索，以下章節將幫助您學習插入新資料進入表之內的基本方法。

註解

別忘記 SQL 可以為大寫或小寫字母，資料是依賴資料庫如何儲存來決定是否有關大小寫，這些例子都使用小寫與大寫字母，來表示不因為大小寫上的不同而結果會有不同差異。

5-2-1 插入資料進入表格內

使用 INSERT 指令來插入新的資料到表格內，有一些 INSERT 指令選擇項，從下列的基本語法來開始：

```
insert into schema.table_name
VALUES ('value1', 'value2', [ NULL ] );
```

使用 INSERT 指令語法，您必須包含在表格裡每個欄位所敘述的 VALUES 清單，請注意清單裡的每個數值用被逗點來分開，被插入進入表格之內的數值，字元與日期資料格式必須使用雙引號包圍起來，雙引號標誌不被數值或 NULL 的資料格式所要求，因爲表格裡的每個欄位，數值應該是被展現。

在下列各項實例，您插入新的記錄在 PRODUCTS_TBL 表格之內。

Table structure:

```
products_tbl

COLUMN Name                    Null?    DATA Type
------------------------------ -------- ------------
PROD_ID                        NOT NULL VARCHAR2(10)
PROD_DESC                      NOT NULL VARCHAR2(25)
COST                           NOT NULL NUMBER(6,2)
```

Sample INSERT statement:

TYPE
```
insert into products_tbl
values ('7725', 'LEATHER GLOVES',24.99);
```

Output
```
1 row created.
```

在這個例子中，您插入三個值在表格的三個欄位之內，被插入值的順序與欄位一般列會於表格內，最初被插入二個數值使用雙引號包圍起來，因爲欄位所對應的資料格式爲字元，第三個值所關連的欄位—COST 是數字所以不需要雙引號，即使它們可以被選擇。


註解

資料庫結構名稱或表格所有者，沒有指定表格名稱的部份，如同它在語法所顯示的一樣，如果您連接到資料庫是資料庫所有者，資料庫結構名稱就不需要了。

5-2-2 插入資料到有限的表格欄位

您可以把資料插入到有限的表格欄位內，舉例來說您只有需要插入全部員工的資料，除了呼叫器號碼外，在這您必須指定欄位列表與在 INSERT 指令的 VALUES 清單。插入表格裡的欄位是有限制，其語法是如下：

```
insert into schema.table_name ('column1', 'column2')
values ('value1', 'value2');
```

在下列實例，您使用 ORDERS_TBL 和插入值進入指定的欄位之內。

Table structure:

```
ORDERS_TBL

COLUMN NAME                    Null?    DATA TYPE
------------------------------ -------- ------------
ORD_NUM                        NOT NULL VARCHAR2(10)
CUST_ID                        NOT NULL VARCHAR2(10)
PROD_ID                        NOT NULL VARCHAR2(10)
QTY                            NOT NULL NUMBER(4)
ORD_DATE                                DATE
```

Sample INSERT statement:

TYPE
```
insert into orders_tbl (ord_num,cust_id,prod_id,qty)
values ('23A16','109','7725',2);
```

Output
```
1 row created.
```

在表格名稱之後的 INSERT 指令，您已經指定一個在刮號內的欄位清單，列出您希望插入資料所有的欄位，ORD_DATE 是唯一排除在外

的欄位，您可以發現就表格的定義來說，在表格內的每筆記錄 ORD_DATE 並不需要輸入資料，因爲 NOT NULL 在表格定義中沒有特別指定，所以 ORD_DATE 不需要輸入資料，NOT NULL 告訴我們，NULL 是不被欄位所允許，而且數值清單必須依您要插入他們到欄位清單的順序出現。


註解

在欄位清單的 INSERT 指令裡，相關表格定義不必反應和欄位相同的順序，但是數值清單必須與欄位清單裡所關連的順序相同。

5-2-3 插入另一個表格的資料

利用 INSERT 與 SELECT 指令的組合，您可以插入查詢自另外表格結果的資料到現有的表格中，參考第 7 小時“介紹資料庫查詢”可以瞭解更多有關資料庫查詢，查詢是使用者向資料庫提出詢問的問題，被回復的資料是答案 [回答]，在結合 INSERT 與 SELECT 指令，您可以插入來自查詢結果的資料到表格中。

插入來自另外一個表格資料的語法是：

```
insert into schema.table_name [('column1', 'column2')]
select [*|('column1', 'column2')]
from table_name
[where condition(s)];
```


您可以看見這個語法裡有三個新的關鍵字，這些關鍵字是 SELECT、FROM、WHERE。SELECT 是 SQL 裡查詢的主要指令，FROM 是敘述目標資料所應該找的表格名稱，WHERE 也是查詢的一部份，被用於設定條件查詢本身，實際條件陳述如下：WHERE NAME = SMITH，這三個關鍵字在第 7、8 小時會有更多的說明。

NEW TERM 條件 condition 是被 SQL 指令所使用，是影響資料標準的方法。

下列範例使用簡單的查詢來檢視 PRODUCTS_TBL 表格裡的所有資料，SELECT * 告訴資料庫，您要檢視表格內所有欄位的訊息，因爲沒有 WHERE 的子句，就可以看見表格裡所有記錄。

TYPE
```
select * from products_tbl;
```

Output
```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
11235      WITCHES COSTUME                29.99
222        PLASTIC PUMPKIN 18 INCH         7.75
13         FALSE PARAFFIN TEETH            1.1
90         LIGHTED LANTERNS               14.5
15         ASSORTED COSTUMES              10
9          CANDY CORN                      1.35
6          PUMPKIN CANDY                   1.45
87         PLASTIC SPIDERS                 1.05
119        ASSORTED MASKS                  4.95
1234       KEY CHAIN                       5.95
2345       OAK BOOKSHELF                  59.99

11 rows selected.
```

以上述查詢爲基礎，現在插入數值在 PRODUCTS_TMP 表格之內，您可以看見 11 個列建立在暫時表格內。

TYPE
```
insert into products_tmp
select * from products_tbl;
```

Output
```
11 rows created.
```

下列的查詢表示 PRODUCTS_TMP 表格裡所有的資料，PRODUCTS_TMP 表格就是您剛剛插入的：

TYPE
```
select * from products_tmp;
```

Output
```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
11235      WITCHES COSTUME                29.99
222        PLASTIC PUMPKIN 18 INCH         7.75
13         FALSE PARAFFIN TEETH            1.1
```

```
90          LIGHTED LANTERNS              14.5
15          ASSORTED COSTUMES             10
9           CANDY CORN                     1.35
6           PUMPKIN CANDY                  1.45
87          PLASTIC SPIDERS                1.05
119         ASSORTED MASKS                 4.95
1234        KEY CHAIN                      5.95
2345        OAK BOOKSHELF                 59.99

11 rows selected.
```

5-2-4 插入 NULL 值

插入 NULL 到表格的欄位中很簡單，如果查詢欄位的數值是未知的，您可能希望插入 NULL 值在欄位之內，舉例來說，不是每個人都有呼叫器，所以規定要輸入呼叫器號碼是錯誤的，您也不用預算呼叫器資料的空間，NULL 值可以利用關鍵字 NULL 插入到表格的一個欄位之內。

插入 NULL 語法爲：

```
insert into schema.table_name values
('column1', NULL, 'column3');
```

NULL 關鍵字應該要和表格存在關連的欄位一起使用，如果您輸入 NULL，那個欄位將不會有資料在裡面，在語法中 NULL 值應從 COLUMN2 的地方開始輸入。

練習以下範例：

TYPE
```
insert into orders_tbl (ord_num,cust_id,prod_id,qty,ORD_DATE)
values ('23A16','109','7725',2,NULL);
```

OUTPUT
```
1 row created.
```

在第一個例子中，列出所有插入 ORDERS_TBL 表格欄位的每個值，在 ORD_DATE 欄位插入的值爲 NULL，意謂您也不知道定貨日期，或在這個時間沒有定貨日期。

TYPE
```
insert into orders_tbl
values ('23A16','109','7725',2, '');
```

OUTPUT
```
1 row created.
```

第二個例子裡的第一個指令有二種差異，但是結果是相同的，第一，沒有欄位清單，請記得，如果您是插入資料到表格內所有的欄位之內，那麼就不需要欄位的清單，第二，''（二個單刮號放在一起）可以代代替插入 NULL 值到 ORD_DATE 欄位之內，這個記號也表示 NULL 值（因爲在它們之間什麼也沒有）。

5-3 更新已存在的資料

先前存在表格裡的資料可以利用 UPDATE 指令修改，UPDATE 指令不是把新的資料加入表格，它也不是刪除資料，只是更新已存在的資料，更新通常被用更新資料庫的一個表格，但是也能同時被用在更新表格內多重欄位，表格內每一個單一列可以被更新或因需要而用一個指令更改許多列。

5-3-1 更新單一欄位的值

最簡單的更新指令形式，是更新表格裡的單一欄位，當要更新表格內單一列的時候，不論單一或多個列的紀錄都可以被更新。

更新單一欄位語法如下：

```
update table_name
set column_name = 'value'
[where condition];
```

以下範例是更新 ORDERS 表格的 QTY 欄位，把 ORD_NUM 23A16 的新數值更新爲 1，在這您已經指定使用 WHERE 的句子。

TYPE
```
update orders_tbl
set qty = 1
where ord_num = '23A16';
```

Output
```
1 row updated.
```

以下範例和之前的例子是相同，除了沒有出現 WHERE 子句：

TYPE
```
update orders_tbl
set qty = 1;
```

Output
```
11 rows updated.
```

請注意，在這個範例中，11 列的資料被更新了，因爲您設定 QTY 爲 1，同時也更新 ORDERS_TBL 表格裡所有列的資料，這是您實際上想做的嗎？也許在少許情況下是真的，但是您很少執行 UPDATE 指令而沒有 WHERE 的句子。


注意

當使用 UPDATE 指令的時候，請盡量小心使用 WHERE 子句，，如果條件沒有指定使用 WHERE，目標表格裡的欄位資料會全部被更新。

5-3-2 在一筆或多筆記錄中更新多重欄位

在這裡將學會如何使用單一的 UPDATE 指令來更新多重欄位。

學習下列語法：

```
update table_name
set column1 = 'value',
   [column2 = 'value',]
   [column3 = 'value']
[where condition];
```

請注意在這語法裡使用唯一的 SET，是用在多重欄位，每個欄位被逗點分開，一般來說 SQL 裡逗點通常被用在分開 SQL 指令與不同類型函數中的獨立變數。

TYPE
```
update orders_tbl
set qty = 1,
    cust_id = '221'
where ord_num = '23A16';
```

Output
```
1 row updated.
```

逗點用來分開被更新的二個欄位，WHERE 是可選擇項，但是通常是必需的。


註解

每個更新指令，關鍵字 SET 只有使用一次，如果超過一個欄位將被更新，逗點就會用來分開將被更新的欄位。

5-4 刪除表格的資料

DELETE 指令被用在刪除表格列的資料，DELETE 指令不是用在將數值從特定的欄位移開；如完整的記錄，全部的欄位會被移動，DELETE 指令必須小心使用，因爲它使用效果非常得好，下二個章節，用來討論刪除表格資料的方法。

爲了要刪除單一記錄或選擇來自表格的記錄，DELETE 指令必須與下列各項語法一起使用：

```
delete from schema.table_name
[where condition];
```

TYPE
```
delete from orders_tbl
where ord_num = '23A16';
```

Output
```
1 row deleted.
```

請留心使用 WHERE，WHERE 是 DELETE 指令一個必要的部份，如果您正在嘗試刪除選擇來自表格的資料列，您很少會執行 DELETE 指令而沒有使用 WHERE，如果您這麼做，結果會類似下列各個範例：

```
delete from orders_tbl;
```

```
11 rows deleted.
```


注意

如果 WHERE 從 DELETE 指令中省略，那麼所有來自表格的資料列將全部被刪除！如同一般的規則，總是一起使用 WHERE 與 DELETE 指令。


註解

暫時表格是這個較早從原始表格載入的，在執行與原始表格不同的命令之前，它在測試 DELETE 與 UPDATE 指令是非常有用的。

5-5 摘要

您已經學習資料處理語言（DML）裡三個基本的指令：INSERT、UPDATE 與 DELETE 指令，正如您已經看到的，資料處理是 SQL 非常有力的部份，它允許資料庫使用者載入新的資料、更新存在的資料和刪除來自表格的資料，非常重要的是，當更新或刪除來自資料庫表格資料，有時往往忘記使用 WHERE，請記得 SQL 放置 WHERE 條件的情況，特別在 UDPATE 與 DELETE 的運算，當指定特定列的時候，資料將會在異動期間受影響，所有目標表格的資料列會被影響，如果不使用 WHERE，那對資料庫是損失慘重的，應謹慎小心的操作處理資料，要確保您的資料。

5-5-1 Q&A

Q 由於有關 DELETE 與 UPDATE 的所有警告，使用他們時會害怕，如果因為不使用 WHERE 而意外地更新表格裡的所有記錄，變更可以被復原嗎？

A 您沒有理由去害怕，因爲對資料庫所做的處理很少不能被改正的，雖然得投入相當的時間和工作，下個小時將討論異動控制的觀念，允許在資料處理後可以用異動控制來結束或恢復原狀。

5-6 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

5-6-1 隨堂測驗

1. 使用下列 employee_tbl 的結構：

column	datatype	(not)null
last_name	varchar2(20)	not null
first_name	varchar2(20)	not null
ssn	char(9)	not null
phone	number(10)	null

LAST_NAME	FIRST_NAME	SSN	PHONE
SMITH	JOHN	312456788	3174549923
ROBERTS	LISA	232118857	3175452321
SMITH	SUE	443221989	3178398712
PIERCE	BILLY	310239856	3176763990

如果執行下列各項指令，將會發生甚麼結果？：

a. insert into employee_tbl
('JACKSON', 'STEVE', '313546078', '3178523443');

b. insert into employee_tbl values
('JACKSON', 'STEVE', '313546078', '3178523443');

c. insert into employee_tbl values
('MILLER', 'DANIEL', '230980012', NULL);

d. insert into employee_tbl values
('TAYLOR', 'NULL', '445761212', '3179221331');

```
e. delete from employee_tbl;

f. delete from employee_tbl
where last_name = 'SMITH';

g. delete from employee_tbl
where last_name = 'SMITH'
and first_name = 'JOHN';

h. update employee_tbl
set last_name = 'CONRAD';

i. update employee_tbl
set last_name = 'CONRAD'
where last_name = 'SMITH';

j. update employee_tbl
set last_name = 'CONRAD',
first_name = 'LARRY';

k. update employee_tbl
set last_name = 'CONRAD'
first_name = 'LARRY'
where ssn = '313546078';
```


5-6-2 練習題

1. 請參閱本書附錄 E，執行 INSERT 指命載入第 3 小時的練習題 1 所建立的表格，當您完成的時，應該對於書中的範例和練習題有更多的瞭解。

2. 使用下列 employee_tbl 的結構：

column	*datatype*	*(not)null*
last_name	varchar2(20)	not null
first_name	varchar2(20)	not null

ssn	char(9)	not null
phone	number(10)	null

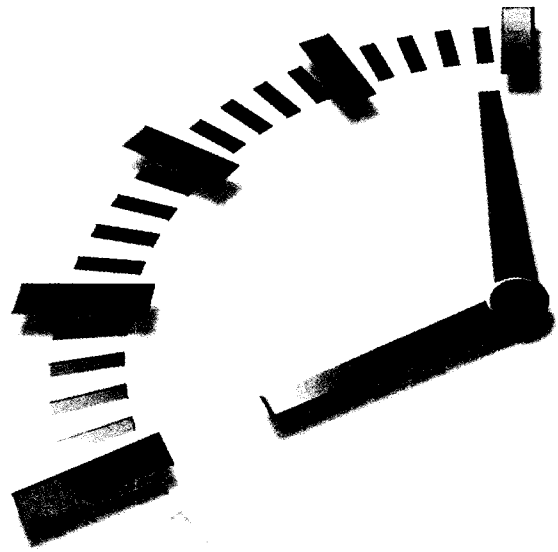
LAST_NAME	FIRST_NAME	SSN	PHONE
SMITH	JOHN	312456788	3174549923
ROBERTS	LISA	232118857	3175452321
SMITH	SUE	443221989	3178398712
PIERCE	BILLY	310239856	3176763990

用 DML 來完成下列敘述：

a. Billy Pierce's 正確的 ssn 為 310239857.

b. 增加 Ben Moore，phone number 是 317-5649880，ssn 是 313456789。

c. John Smith 辭職，刪除他的記錄。



第 6 小時

管理資料庫異動

在這個小時，您將學習資料庫異動的管理與概念。

本小時的重點包括：

- 異動的定義
- 那些指令用來控制異動
- 異動指令的語法和例子
- 何時使用異動指令
- 不正當的異動控制的結果是什麼？

6-1 何為異動？

異動是完成合乎邏輯順序的工作單位，在使用 SQL 的關連式資料庫，藉著使用者或藉著資料庫程式來完成。異動是利用在第 5 小時“處理資料”所討論的 DML 指令（INSERT, UPDATE, and DELETE）來完

成的，異動是增加資料庫的變更，舉例來說，如果執行 UPDATE 指令來改變表格的名稱就表示您正在執行異動，。

NEW TERM 異動(transaction)是指一個資料庫被執行的工作單位。

異動可以是一個 DML 指令或一個組群的指令，當管理異動群組的時候，每個指定異動群組必須成功，否則他們將會全部失敗。

下列描述異動的性質：

- 所有的異動都必須有開始和結束
- 異動能存檔復原
- 如果異動在中間失敗，沒有任何部份可以存到資料庫


註解

啟動或執行異動是依各加廠商特性而定，您必須檢查您的廠商該如何開始異動，在 ANSI 標準裡，沒有特別外在的開始或啟動異動。

6-2 異動控制是什麼？

異動控制是管理不同異動的能力，也可能在關連式資料庫管理系統裡面發生發生，當您提到異動時，就表示也正在提及 INSERT，UPDATE 與 DELETE 指令。

當異動執行成功後，表格並不會立刻改變，雖然它可能表現在輸出的時候才看得出來。當異動成功完成時，用異動控制指令來結束異動，儲存對資料庫所做的異動，或復原被異動所做的變化。

有三個指令使用在異動控制：

- COMMIT
- ROLLBACK

- SAVEPOINT

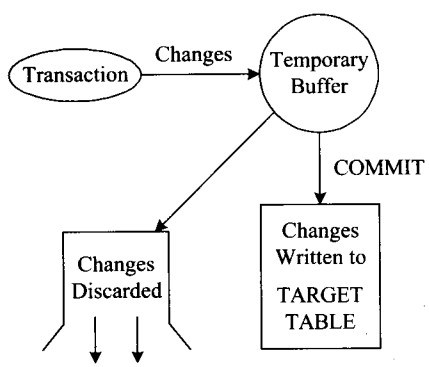
每一個將分別在下列章節詳細討論。

註解

異動控制指令只能與 DML 指令（INSERT、UPDATE、DELETE）一起使用，舉例來說，您不能在建立表格之後，執行 COMMIT 指令，當表格被建立的時候，它自動地被交付到資料庫，而且您不能在修正表格後再執行 ROLLBACK。

當異動已經完成的時候，異動訊息會儲存在資料庫裡的分派區域或暫時的 ROLLBACK，所有的變化會儲存在這個暫時的 ROLLBACK 區域，直到異動控制指令執行。當異動控制指令被執行的時候，變更不是在資料庫產生就是被丟棄；然後暫時的 ROLLBACK 區域被清空。圖 6.1 舉例說明變更如何運用於關連式資料庫。

圖 6.1. Rollback



6-2-1 COMMIT 指令

COMMIT 指令是異動指令儲存對資料庫異動的變化，COMMIT 指令儲存自最後一次 COMMIT 或 ROLLBACK 指令對資料庫所做的異動。

這個指令的語法是：

```
commit [ work ];
```

關鍵字 COMMIT 是語法中唯一命令的部份，可連同字元或指令用來終止指令；WORK 是可選擇的關鍵字，主要是使指令更有使用者親和性。

在下列範例，先選擇自 PRODUCT_TMP 表格所有資料：

```
select * from products_tmp;
PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
11235      WITCHES COSTUME                29.99
222        PLASTIC PUMPKIN 18 INCH         7.75
13         FALSE PARAFFIN TEETH            1.1
90         LIGHTED LANTERNS               14.5
15         ASSORTED COSTUMES              10
9          CANDY CORN                      1.35
6          PUMPKIN CANDY                   1.45
87         PLASTIC SPIDERS                 1.05
119        ASSORTED MASKS                  4.95
1234       KEY CHAIN                       5.95
2345       OAK BOOKSHELF                  59.99

11 rows selected.
```

然後，刪除來自產品表格內價值小於 $14.00 的所有的記錄。

```
delete from products_tmp
where cost < 14;

8 rows deleted.
```

COMMIT 指令被執行來儲存對資料庫的變化而完成異動。

```
commit;

Commit complete.
```


注意

太多的 commits 得花上許多額外的時間來完成工作，首先所有的變化被送回暫時 ROLLBACK 區域，如果暫時 ROLLBACK 區域的執行空間不夠，又不能儲存資料庫有關的變化，資料庫會可能停機，不許更進一步的異動活動。


註解

在一些廠商，沒有執行 commit 指令，異動就被委託執行，僅僅由資料庫簽出而產生委託。

6-2-2 ROLLBACK 指令

Rollback 指令是異動控制指令，用來復原未被資料庫儲存的異動，ROLLBACK 指令只能復原最後被 COMMIT 或 ROLLBACK 所執行的異動。

ROLLBACK 指令的語法是：

```
rollback [ work ];
```

再一次在 COMMIT 指令，關鍵字 WORK 是 ROLLBACK 語法中可以選擇的一部份。

在下列實例中，藉由選擇早先刪除了 14 個記錄的表格 PRODUCTS_TMP 的所有記錄開始：

TYPE
```
select * from products_tmp;

PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
11235      WITCHES COSTUME                29.99
90         LIGHTED LANTERNS                14.5
2345       OAK BOOKSHELF                  59.99

3 rows selected.
```

然後更新表格，變更產品編號 11235 的產品價值爲 $39.99。

TYPE
```
update products_tmp
```

```
set cost = 39.99
where prod_id = '11235';

1 row updated.
```

如果您執行表格上的快速查詢，可以看見變化已經產生：

TYPE
```
select * from products_tmp;

PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
11235      WITCHES COSTUME                39.99
90         LIGHTED LANTERNS                14.5
2345       OAK BOOKSHELF                  59.99

3 rows selected.
```

現在復原上個更改執行 ROLLBACK 指令：

TYPE
```
rollback;

Rollback complete.
```

最後，確認變化不被委託到資料庫：

TYPE
```
select * from products_tmp;

PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
11235      WITCHES COSTUME                29.99
90         LIGHTED LANTERNS                14.5
2345       OAK BOOKSHELF                  59.99

3 rows selected.
```

6-2-3 SAVEPOINT 指令

Savepoint 是指異動裡的點，讓您可以還原異動而不用還原全體所做的異動。

Savepoint 指令的語法是：

```
savepoint savepoint_name
```

這個指令只能在建立 SAVEPOINT 與異動指令之上執行，ROLLBACK 指令是用來復原一個群組的異動，SAVEPOINT 是藉由打破大量的異動管理方法進入較小、易管理的群組之內，


註解

> Savepoint 名稱在異動的關連群組內必須是唯一的，然而，SAVEPOINT 可以有和表格或其他物件一樣的名稱。明細請參照各家廠商文件所提的命名協定。

ROLLBACK 到 savepoint 的語法：

```
ROLLBACK TO SAVEPOINT_NAME;
```

在這個例子中，您計劃刪除來自 PRODUCTS_TMP 表格那些剩餘的三個記錄，您要在做每一刪除前建立個 SAVEPOINT，所以您可以到任何的 SAVEPOINT 並在任何時間回到它資料最初的狀態：

```
savepoint sp1;

savepoint created.
delete from products_tmp where prod_id = '11235';

1 row deleted.
savepoint sp2;

Savepoint created.
delete from products_tmp where prod_id = '90';

1 row deleted.
savepoint sp3;

Savepoint created.
delete from products_tmp where prod_id = '2345';

1 row deleted.
```

現在那三個刪除動作已經發生，比方說您已經改變主意並且決定*復原到如同 SP2 的資料，因爲 SP2 在第一個刪除動作之後被建立*，所以最後二個刪除動作被恢復原狀：

TYPE
```
rollback to sp2;
```

Output
```
Rollback complete.
```

請注意由於我們復原到 SP2，所以只有第一個刪除動作發生：

TYPE
```
select * from products_tmp;
```

Output
```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
90         LIGHTED LANTERNS                14.5
2345       OAK BOOKSHELF                  59.99

2 rows selected.
```

請記得 ROLLBACK 指令會獨自回復到上一個 COMMIT 或 ROLLBACK，由於尚未執行 COMMIT，所以所有的刪除動作會被恢復原狀，如在以下範例：

```
rollback;

Rollback complete.
select * from products_tmp;

PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
11235      WITCHES COSTUME                29.99
90         LIGHTED LANTERNS                14.5
2345       OAK BOOKSHELF                  59.99

3 rows selected.
```

6-3 異動控制和資料庫執行效率

再三地強調，不當的異動控制會影響資料庫執行效率甚至導致資料庫當機，因爲在 COMMIT 或 ROLLBACK 命令執行完成前，由於大量執行 inserts、updates 或 deletes 的指令，缺乏有效異動控制而導致不只是在大量批次過程中需要中央處理器與記憶體來執行，同時也佔去暫時給 rollback 儲存訊息的空間。當 COMMIT 被執行，rollback 異動訊息

被寫到目標表格，ROLLBACK 在暫儲區的訊息會被清除；當 ROLLBACK 被執行的時候，對資料庫沒有進行更改，ROLLBACK 在暫儲區的訊息也會被清除。如果 COMMIT 或 ROLLBACK 都沒被執行，ROLLBACK 在暫儲區的訊息會繼續增加，直到沒有剩下空間，因此會壓迫資料庫停止所有的處理，直到空間被釋放。

6-4 摘要

在這個小時，您藉著如何使用三個異動控制指令學習異動管理的初步觀念：COMMIT、ROLLBACK 和 SAVEPOINT。COMMIT 是用來儲存資料庫的異動、ROLLBACK 是用來復原所執行的異動、SAVEPOINT 是用來切割異動進入群組內，允許您還原到特定且合乎邏輯的異動程序。記得應該要時常使用 COMMIT 與 ROLLBACK 指令，當執行大量的異動工作時，也要保存資料庫適當的空間，也請記住這些異動指令只能與 DML 的三個指令（INSERT, UPDATE, and DELETE）一起使用。

6-4-1 Q&A

Q 是否必需在每一個的 insert 指令之後都要執行 COMMIT？

A 不一定，如果正在插入千、百筆資料進入表格內，每 5,000-10,000 筆資料會要求執行 COMMIT 一次，因視暫存區域容量大小而定，請記得，當 ROLLBACK 區域塞滿的時候，資料庫會停止作業。

Q ROLLBACK 指令是如何復原異動？

A ROLLBACK 指令清除來自 ROLLBACK 區域所有的變化。

Q 如果執行異動且 99% 的異動已經完成，但是 1% 出錯誤，我能否只再做錯誤部份嗎？

A 不可以，全部的異動必須一次成功；否則資料會缺乏完整性。

Q 執行 COMMIT 後，異動成為固定時，我可以改變更新資料嗎？

A 固定在這表示的意思是，因爲它現在已經是資料庫的一部份，UPDATE 指令可以使用作爲資料庫的校正。

6-5 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

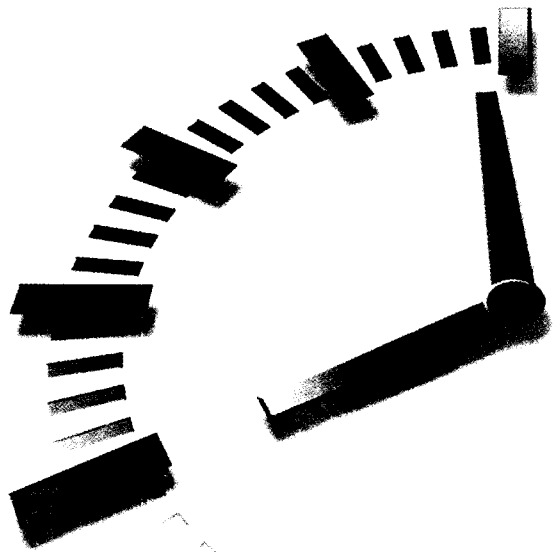
6-5-1 隨堂測驗

1. 是非題：如果您已經 commit 幾個異動，但仍有許多異動尚未完成，那麼在執行 ROLLBACK 指令的同時您所做的異動並未完成。
2. 是非題：在一定的異動已經執行完成之後，savepoint 實際上已經儲存異動了。
3. 簡短地描述下列各項指令的目的：COMMIT、ROLLBACK 和 SAVEPOINT。

6-5-2 練習題

1. 利用下列各項異動，在每三個異動之後建立 savepoints，然後 COMMIT 那些異動。

```
transaction1;
transaction2;
transaction3;
transaction4;
transaction5;
transaction6;
transaction7;
transaction8;
transaction9;
transaction10;
transaction11;
transaction12;
```



第 III 單元

從查詢得到有效的結果

第 7 小時　介紹資料庫查詢

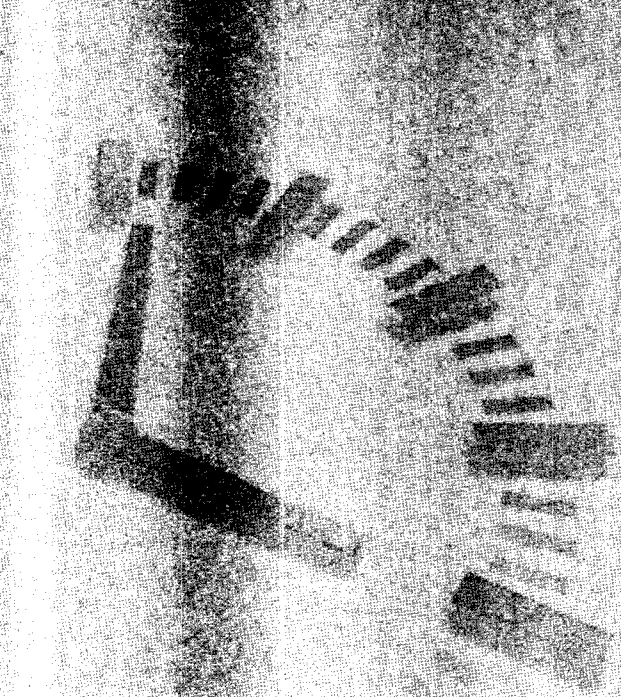
第 8 小時　使用運算子來類別資料

第 9 小時　資料查詢結果總結

第 10 小時　資料的分類和排序

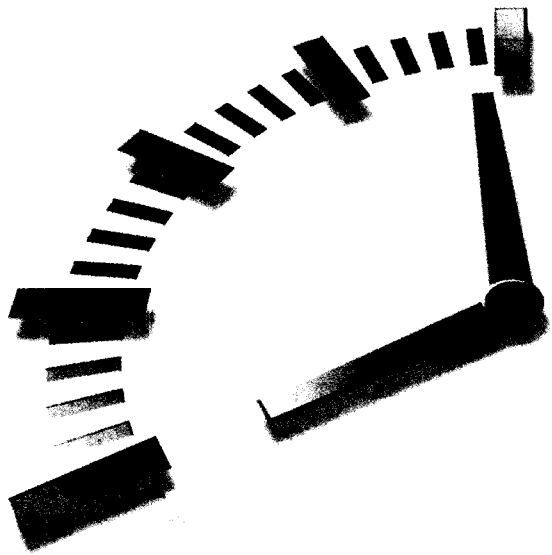
第 11 小時　更改資料展現結構

第 12 小時　理解日期與時間



第三單元

果蔬的效育運輸儲查給



第 7 小時

介紹資料庫查詢

在這第 7 小時，您將學到資料庫查詢包括 SELECT 指令使用，SELECT 指令大概是資料庫建立後，最常使用的 SQL 指令，。

本小時的重點包括：

- 資料庫查詢是什麼？
- 該如何使用 SELECT 指令
- 使用 WHERE，把條件加入查詢
- 使用欄位別名
- 選擇來自另外使用者表格的資料

7-1 查詢是什麼？

查詢是使用 SELECT 指令到資料庫內所做的調查，查詢是到可讀取的資料庫格式裡，依照使用者的請求抽取資料，舉例來說，如果您

有員工資料表格，您可能執行 SQL 指令來傳回那位員工薪資最高，這個要求對可用的員工資料庫內是很典型的，可以在關連式資料庫中執行。

7-2 介紹 SELECT 指令

SELECT 指令，代表 SQL 資料查詢語言（DQL）的指令，是用來建立資料庫查詢，SELECT 指令不一定是單一指令，意謂需要其他子句，除了必需的子句外，另有選擇性子句來增加 SELECT 指令的全體功能，SELECT 指令是目前爲止 SQL 裡最有用的指令之一，FROM 子句是必須連同 SELECT 一起使用的子句。

有四個關鍵字或子句，在 SELECT 指令是有非常重要的部份，這些關鍵字如下：

- SELECT
- FROM
- WHERE
- ORDER BY

每一個關鍵字在下列章節會詳細討論。

7-2-1 SELECT 指令

SELECT 指令是與 FROM 一起使用，用來從資料庫抽取資料，組織成可讀的資料格式，SELECT 是查詢的一部份，依照欄位在表格儲存的方式，選取您要的資料，

簡單的 SELECT 指令語法如下：

```
SELECT [ * | all | distinct column1, column2 ]
from table1 [ , table2 ];
```

SELECT 在查詢裡的關鍵字，是跟隨著您希望顯示爲查詢輸出的一部份欄位清單。FROM 關鍵字，是跟隨著一個或多個表格清單，您希望從那些表格選取，星號（*）習慣表示爲表格裡所有的欄位且應該被顯示爲輸出的一部份，檢查您的 SQL 廠商有關其他特定使用方法，包括所有被用來顯示欄位所有的數值的選項與複製。不同的選項用來排除被複製的列，ALL 是 DISTINCT 與 ALL 的預設值不需要特別說明。請注意，SELECT 之後被逗點分開的欄位，是如同跟隨 FROM 之後的表格清單。


註解

逗點是用來分開 SQL 指令裡的獨立引數，一些通常查詢的欄位清單包括，要查詢的表格清單，被插入表格之內的數值和聚集數值當做查詢裡 WHERE 子句的條件。

NEW TERM 引數（Argument）在 SQL 指令或命令的語法，是不一定需要的或是可以選擇的數值。

藉由學習下列各例子，來探究 SELECT 指令的基本效能，首先使用 PRODUCTS_TBL 表格來執行個簡單的查詢。

TYPE
```
SELECT * FROM PRODUCTS_TBL;
```

Output
```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
11235      WITCHES COSTUME               29.99
222        PLASTIC PUMPKIN 18 INCH        7.75
13         FALSE PARAFFIN TEETH            1.1
90         LIGHTED LANTERNS               14.5
15         ASSORTED COSTUMES              10
9          CANDY CORN                     1.35
6          PUMPKIN CANDY                  1.45
87         PLASTIC SPIDERS                1.05
119        ASSORTED MASKS                 4.95

9 rows selected.
```

星號表示您能看見在格式內 PROD_ID、PROD_DESC 和 COST 表格裡的所有欄位，輸出裡的每個欄位是顯示它在表格的順序，這個表格裡有 9 個記錄，以回應 9 rows selected 來識別，這個回應，會因廠商不同而異；舉例來說，相同查詢的另外回應，爲 9 rows affected。

現在 SELECT 來自 CANDY_TBL 表格的資料，建立表格 PRODUCTS_TBL，以下例子的基本表格，列出在 SELECT 關鍵字之後欄位的名稱，顯示表格裡唯一的欄位。

TYPE
```
SELECT PROD_DESC FROM CANDY_TBL;
```

Output
```
PROD_DESC
------------------------
CANDY CORN
CANDY CORN
HERSHEYS KISS
SMARTIES

4 rows selected.
```

四筆資料存在表格 CINDY_TBL 中，您已經在下一個指令裡用 ALL，來顯示 ALL 是個多餘的選項，不需要去指定 ALL；這是預設選項。

TYPE
```
SELECT ALL PROD_DESC
FROM CANDY_TBL;
```

Output
```
PROD_DESC
------------------------
CANDY CORN
CANDY CORN
HERSHEYS KISS
SMARTIES

4 rows selected.
```

DISTINCT 選項在下列各指令被用在禁止顯示複製的記錄，請注意，CANDY CORN 值在這個例子只有被顯示一次。

TYPE
```
SELECT DISTINCT PROD_DESC
FROM CANDY_TBL;
```

Output
```
PROD_DESC
------------------------
CANDY CORN
HERSHEYS KISS
SMARTIES

3 rows selected.
```

DISTINCT 和 ALL 也可以與弧號內關連的欄位一起使用，在 SQL 中時常使用刮弧與其他電腦語言，來增進 SQL 的可讀性。

TYPE
```
SELECT DISTINCT(PROD_DESC)
FROM CANDY_TBL;
```

Output
```
PROD_DESC
------------------------
CANDY CORN
HERSHEYS KISS
SMARTIES

3 rows selected.
```

7-2-2 FROM 子句

FROM 子句總是與 SELECT 指令一起使用，它是任何查詢所必需的元件，FROM 子句目的是在查詢資料表格時告訴資料庫表格存取的順序，FROM 子句可以包含一個或較多個的表格。

FROM 的語法爲下：

```
from table1 [ , table2 ]
```

7-2-3 使用條件來區別資料

條件是查詢的一部份，用來顯示使用者所指定的選擇性訊息，條件的數值不是對就是錯，藉此限制查詢所收到的資料，WHERE 子句是用來放置查詢的條件，來減少因查詢時未放置條件所回應的結果。

在 WHERE 子句中可以有超過一個條以上的件，如果有超過一個以上的條件，它們會被 AND 與 OR 運算子連接，在第 8 小時“使用運算子分類資料”，在下個小時您也會學習到一些條件運算子，可以在查詢裡指定條件，這個小時只有提到在每個查詢使用單一條件。

NEW TERM 運算子（operator）是 SQL 裡的字元或關鍵字，用來聯合 SQL 指令裡的元件。

WHERE 語法如下：

```
select [ all | * | distinct column1, column2 ]
from table1 [ , table2 ]
where [ condition1 | expression1 ]
[ and condition2 | expression2 ]
```

下列是沒有指定條件簡單的 SELECT 子句：

TYPE
```
SELECT *
FROM PRODUCTS_TBL;
```

Output
```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ -----------
11235      WITCHES COSTUME              29.99
222        PLASTIC PUMPKIN 18 INCH       7.75
13         FALSE PARAFFIN TEETH           1.1
90         LIGHTED LANTERNS              14.5
15         ASSORTED COSTUMES             10
9          CANDY CORN                     1.35
6          PUMPKIN CANDY                  1.45
87         PLASTIC SPIDERS                1.05
119        ASSORTED MASKS                 4.95

9 rows selected.
```

現在加入條件到相同的查詢。

TYPE
```
SELECT *FROM PRODUCTS_TBL
WHERE COST < 5;
```

Output

```
PROD_ID    PROD_DESC                 COST
---------- ------------------------- ----------
13         FALSE PARAFFIN TEETH      1.1
9          CANDY CORN                1.35
6          PUMPKIN CANDY             1.45
87         PLASTIC SPIDERS           1.05
119        ASSORTED MASKS            4.95

5 rows selected.
```

唯一顯示的記錄，是那些 COST 小於 $5。

在下列的查詢，您希望顯示 product description 與 cost，和產品編號 119 相同。

TYPE

```
SELECT PROD_DESC, COST
FROM PRODUCTS_TBL
WHERE PROD_ID = '119';
```

Output

```
PROD_DESC                 COST
------------------------- ----------
ASSORTED MASKS            4.95

1 row selected.
```

7-2-4 輸出排序

您通常希望您的輸出經過排序，資料可以藉由使用 ORDER BY 子句來排序，ORDER BY 子句安排您所查詢的結果排列在您指定的清單內。如果輸出的名稱是字母的話，ORDER BY 所預設排序順序是照按英文字母順序 A 到 Z，遞減排序會依字母輸出將會從 Z 顯示到 A。數字的遞增排序，數字 1 和 9 會顯示爲 1 到 9；遞減排序則顯示從 9 到 1。

ORDER BY 的語法如下：

```
select [ all | * | distinct column1, column2 ]
from table1 [ , table2 ]
where [ condition1 | expression1 ]
[ and condition2 | expression2 ]
ORDER BY column1|integer [ ASC|DESC ]
```

藉著前面的指令來學習 ORDER BY 子句，**product description** 爲字母遞增排序，請注意使用 ASC 選項，ASC 可以用在 ORDER BY 子句指定的任何欄位之後。

TYPE
```
SELECT PROD_DESC, PROD_ID, COST
FROM PRODUCTS_TBL
WHERE COST < 20
ORDER BY PROD_DESC ASC;
```

Output
```
PROD_DESC                 PROD_ID    COST
------------------------- ---------- ----------
ASSORTED COSTUMES         15         10
ASSORTED MASKS            119         4.95
CANDY CORN                9           1.35
FALSE PARAFFIN TEETH      13          1.1
LIGHTED LANTERNS          90         14.5
PLASTIC PUMPKIN 18 INCH   222         7.75
PLASTIC SPIDERS           87          1.05
PUMPKIN CANDY             6           1.45

8 rows selected.
```


註解

因為遞增排序是預設值，ASC 不需要另外被指定。

如果您希望輸出顛倒的字母順序可以用 DESC，如同下列各項指令，。

TYPE
```
SELECT PROD_DESC, PROD_ID, COST
FROM PRODUCTS_TBL
WHERE COST < 20
ORDER BY PROD_DESC DESC;
```

Output
```
PROD_DESC                 PROD_ID    COST
------------------------- ---------- ----------
PUMPKIN CANDY             6           1.45
PLASTIC SPIDERS           87          1.05
PLASTIC PUMPKIN 18 INCH   222         7.75
LIGHTED LANTERNS          90         14.5
FALSE PARAFFIN TEETH      13          1.1
CANDY CORN                9           1.35
ASSORTED MASKS            119         4.95
```

```
ASSORTED COSTUMES          15              10

8 rows selected.
```

在 SQL 裡有捷徑，欄位列在 ORDER BY 子句旁可以與整數一起縮寫，INTEGER 爲實際的欄位替換名稱，用於識別在關鍵字 SELECT 之後欄位的位置。

下列爲使用整數的範例，跟隨識別 ORDER BY 子句。

TYPE

```
SELECT PROD_DESC, PROD_ID, COST
FROM PRODUCTS_TBL
WHERE COST < 20
ORDER BY 1;
```

Output

```
PROD_DESC                  PROD_ID    COST
-------------------------- ---------- ----------
ASSORTED COSTUMES          15         10
ASSORTED MASKS             119         4.95
CANDY CORN                 9           1.35
FALSE PARAFFIN TEETH       13          1.1
LIGHTED LANTERNS           90         14.5
PLASTIC PUMPKIN 18 INCH    222         7.75
PLASTIC SPIDERS            87          1.05
PUMPKIN CANDY              6           1.45

8 rows selected.
```

在這個查詢中，整數 1 表示欄位 PROD_DESC，整數 2 表示 PROD_ID 欄位，3 表示 COST 欄位等等。

您可以藉著多重欄位查詢來排序，使用 SELECT 裡的欄位名稱或連結欄位數值。

```
ORDER BY 1,2,3
```

ORDER BY 的欄位是不需要隨著 SELECT 出現在關連欄位的相同排序中，如下列實例：

```
ORDER BY 1,3,2
```

7-2-5 大小寫不同

大小寫在 SQL 編碼時是非常重要的觀念，一般 SQL 的指令與關鍵字是不需要注重大小寫的，可依您的喜好而定，可以隨意輸入，大小寫可以混合使用（在單一的字或指令），請參閱第 5 小時“處理資料”。

然而，在處理 SQL 裡資料的時候，大小寫不同是一個主要的因素，在大多數的情形，資料唯一被儲存在關連式資料庫的方式是用大寫字母來保持資料的一致性。

舉例來說，如果任意地輸入大小寫到您的資料庫，則您的資料不會有一致性，：

```
JOHN SMITH
John Smith
john smith
```

如果名稱被儲存爲 john smith，依照下列執行查詢時，沒有列會回復。

```
SELECT *
FROM EMPLOYEE_TBL
WHERE EMPLOYEE_NAME = 'JOHN SMITH';
```

註解

當參考資料庫裡資料的時候，查詢裡的大小寫和被儲存的資料必須相同，當輸入資料的時候，查閱公司的規定，在哪些時候應用大寫或小寫。

7-3 查詢的簡單範例

這個章節討論用查詢觀念建立的一些範例，這章開始執行最簡單的查詢，和建立累進查詢，您使用表格 EMPLOYEE_TBL。

選取來自表格所有記錄並顯示所有的欄位：

```
SELECT * from employee_tbl;
```

選取來自表格所有記錄並顯示指定的欄位：

```
SELECT employee_id
from employee_tbl;
```

選取來自表格所有記錄並顯示指定的欄位，如果需要的話，您可以輸入一行碼或使用停止回傳：

```
SELECT employee_id from employee_tbl;
```

選取來自表格所有記錄並顯示用逗點分開的所有多重欄位：

```
SELECT EMPLOYEE_ID, EMPLOYEE_NAME
FROM EMPLOYEE_TBL;
```

顯示給設定資料的條件：

```
SELECT employee_id, employee_name
from employee_tbl
where employee_id = '333333333';
```

顯示給設定資料的條件並排序輸出：

```
SELECT employee_id, employee_name
from employee_tbl
where CITY = 'INDIANAPOLIS'
ORDER BY EMPLOYEE_ID;
```

顯示給資料設定的條件並排序輸出多重欄位，一個欄位顛倒排序的順序：

```
SELECT employee_id, employee_name
from employee_tbl
where CITY = 'INDIANAPOLIS'
ORDER BY EMPLOYEE_ID, EMPLOYEE_NAME DESC;
```

顯示給資料設定的條件並在欄位的地方使用整數 spelled-out 輸出名稱排序：

```
SELECT employee_id, employee_name
from employee_tbl
where CITY = 'INDIANAPOLIS'
```

```
ORDER BY 1;
```

顯示給資料設定的條件並藉著整數排序輸出多重欄位，欄位排序方式與用 SELECT 關鍵字之後排序方式是不同的：

```
SELECT employee_id, employee_name
from employee_tbl
where CITY = 'INDIANAPOLIS'
ORDER BY 2, 1;
```


註解

當從較大的表格來選取所有列的資料時候，通常結果能夠傳回實質上足夠的資料。

7-3-1 計算表格資料筆數

簡單的查詢可以在表格上被執行，迅速得到表格裡的記錄筆數，或在表格裡的欄位數值的筆數，筆數藉著函數 COUNT 來完成的，雖然函數在這本書較遲的章節才會被討論，但函數在這裡要先介紹，因爲它是您可以建立的最簡單的查詢之一。

```
SELECT COUNT(*)
FROM TABLE_NAME;
```

COUNT 函數與括號一起使用，括號圍繞著筆數的目標欄位或用星號來計算表格裡所有資料的筆數。

計算 PRODUCTS_TBL 表格裡記錄筆數：

TYPE
```
SELECT COUNT(*) FROM PRODUCTS_TBL;
```

Output
```
COUNT(*)
----------
         9

1 row selected.
```

計算 PRODUCTS_TBL 表格裡的 PROD_ID 數值的數目。

TYPE
```
SELECT COUNT(PROD_ID) FROM PRODUCTS_TBL;
```

Output
```
COUNT(PROD_ID)
--------------
             9

1 row selected.
```


註解

如果計算的欄位是 NOT NULL（必需的欄位），計算欄位的數值數目是相同於計算表格裡的記錄筆數。

7-3-2 從其他使用者表格選取資料

權限必須被授與給表格使用者來存取另外使用者的表格，如果沒有授與權限，不是表格擁有者的使用者將不被允許存取，在被授權於使用另外使用者表格後（GRAND 指令將在第 20 小時“建立、使用檢視與同義字”中討論），您可以選擇來自另外一個使用者表格的資料，SELECT 指令裡，爲了存取另外一個使用者的表格，您必須用資料庫結構名稱與表格名稱或表格擁有者名稱，來做下列各項實例：

```
SELECT EMP_ID
FROM SCHEMA.EMPLOYEE_TBL;
```


註解

如果同義字存在於您想要存取的資料庫表格，您不必須指定表格資料庫結構名稱，同義字是表格輪流的名稱，在第 21 小時“與系統目錄工作“會討論，。

7-3-3 欄位別名

欄位別名是用來更新表格欄位的名稱，特別是應用在查詢，employee_tbl 舉例說明欄位別名的使用。

```
SELECT COLUMN_NAME ALIAS_NAME
FROM TABLE_NAME;
```

下列各項實例，顯示兩次產品描述，給第二欄位命名爲 PRODUCT，請注意輸出的欄位標題：

TYPE

```
select prod_desc,
       prod_desc product
from products_tbl;
```

Output

```
PROD_DESC                 PRODUCT
------------------------- -------------------------
WITCHES COSTUME           WITCHES COSTUME
PLASTIC PUMPKIN 18 INCH   PLASTIC PUMPKIN 18 INCH
FALSE PARAFFIN TEETH      FALSE PARAFFIN TEETH
LIGHTED LANTERNS          LIGHTED LANTERNS
ASSORTED COSTUMES         ASSORTED COSTUMES
CANDY CORN                CANDY CORN
PUMPKIN CANDY             PUMPKIN CANDY
PLASTIC SPIDERS           PLASTIC SPIDERS
ASSORTED MASKS            ASSORTED MASKS

9 rows selected.
```

欄位別名可以針對客戶的名稱需求設計，在一些 SQL 廠商，也可以爲欄位參考成較短的名稱。

當欄位在 SELECT 指令被重新命名的時候，名稱不再是固定不變的，變化的是特別的 SELECT 指令。

7-4 摘要

您已經瞭解資料庫查詢，這表示如何從關連式資料庫獲得有用的訊息。SELECT 指令是資料庫查詢語言（DQL）的指令，用來建立 SQL 查詢。FROM 子句必須和每個 SELECT 指令包含在一起使用。您已經學習如何放置條件在查詢上並使用 WHERE 子句，和如何使用 ORDER BY 來排序資料。除了已經學習寫查詢的基本概念外，在做了一些練習之後，您應該準備在下個小時學習更多有關如何使用查詢的方法。

7-4-1 Q&A

Q 為什麼 SELECT 一定要和 FROM 一起使用？

A SELECT 子句只告訴資料庫您希望看見什麼資料，FROM 子句告訴資料庫該在哪裡得到這些資料。

Q 當我藉著子句 ORDER BY 選取遞減排序的時候，實際對資料做了什麼？

A 當您藉著子句 ORDER BY 並且已經選取來自 employee_tbl 的 last_name，如果使用遞減排序選項，排序將會文字以字母 Z 開始並結束到 A。

Q 變更欄位名稱有什麼優點？

A 新的欄位名稱能夠更接近傳回的資料的描述。

7-5 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

7-5-1 隨堂測驗

1. 說明任何 SELECT 指令所需要部分。
2. 在 WHERE 子句，是否所有資料需要單一括號？
3. 在 SQL 語言的部份，SELECT 指令（資料庫查詢）所扮演什麼角色？
4. 多重條件能被用在 WHERE 子句嗎？

7-5-2 練習題

1. 檢視下列的 SELECT 指令，檢查語法是否正確，如果語法是不正確的，如何修正它？應用表格 employee_tbl。

a.

```
SELECT employee_id, last_name, first_name,
FROM EMPLOYEE_TBL;
```

b.

```
SELECT EMPLOYEE_ID, LAST_NAME
ORDER BY EMPLOYEE TBL
FROM EMPLOYEE_TBL;
```

c.

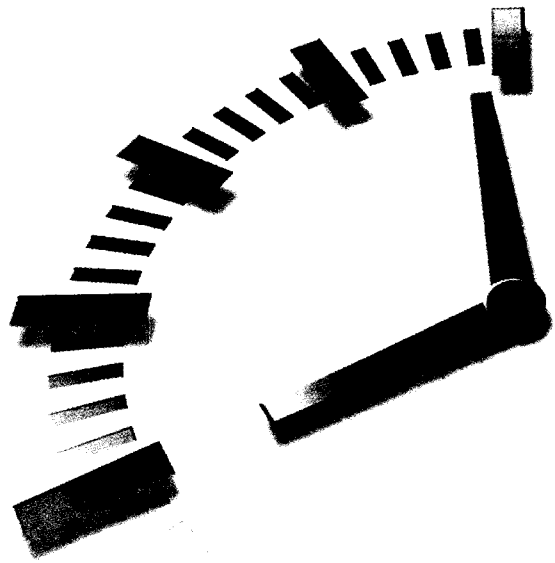
```
SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME
FROM EMPLOYEE_TBL
WHERE EMPLOYEE_ID = '333333333'
ORDER BY EMPLOYEE_ID;
```

d.

```
SELECT EMPLOYEE_ID SSN, LAST_NAME
FROM EMPLOYEE_TBL
WHERE EMPLOYEE ID = '333333333'
ORDER BY 1;
```

e.

```
SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME
FROM EMPLOYEE_TBL
WHERE EMPLOYEE_ID = '333333333'
ORDER BY 3, 1, 2;
```



第 8 小時

使用運算子來類別資料

本小時的重點包括：

- 運算子是什麼？
- SQL 裡運算子的概觀
- 使用少見地運算子好嗎？
- 運算子如何組合使用？

8-1 SQL 裡運算子是什麼？

運算子是保留字，爲 SQL 指令的主要字元 WHERE 子句來執行運算，像是比較運算子和算數運算子。運算子說明 SQL 指令裡的條件和爲指令裡多重條件的連接詞。

在這個小時討論的運算子是：

- Comparison operators（比較運算子）
- Logical operators（邏輯運算子）
- Operators used to negate conditions（用運算子否定條件）
- Arithmetic operators（算術運算子）

8-2 比較運算子

比較運算子用來測試 SQL 指令裡單一的數值，在這所討論的運算子由 =、<>、< 和 > 組成。

這些運算子用來測試：

- Equality 相等
- Non-equality 不相等
- Less-than values 較小數值
- Greater-than values 較大數值

使用方法的實例和比較運算子意義，在下列章節會討論。

8-2-1 Equality 相等

相等運算子是比較 SQL 指令裡數值的單一性，相等符號爲（=），當測試相等的時候，比較的數值必須正好相同，否則不會傳回任何資料，如果在相等比較期間，二個數值是相等的，傳回的數值爲正確的；如果沒有找到相等資料，傳回的數值爲錯誤的，Boolean 值（正確的／錯誤的）用來決定資料是否依照條件傳回，= 運算子與其他的運算子可以獨自使用或者聯合。

使用與意義相等運算子的實例：

Usage	Meaning
`WHERE SALARY = '20000'`	薪資等於 20000

下列查詢傳回所有列的 PROD_ID 爲 2345 的資料。

TYPE

```
SELECT *
FROM PRODUCTS_TBL
WHERE PROD_ID = '2345';
```

Output

```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
2345       OAK BOOKSHELF                 59.99

1 row selected.
```

8-2-2 Non-equality 不相等

在 SQL，用來衡量不相等的運算子是 <>（較小符號結合較大符號），如果條件發現不相等條件，會傳回正確訊息，如果發現條件相等，會傳回錯誤訊息。

註解

<> 另外不同的選擇是 !=，主要的廠商已經採用 != 來表示不相等，檢查您的廠商特別的用法。

Usage	Meaning
WHERE SALARY <> `20000'	薪資不等於 20000

TYPE

```
SELECT *
FROM PRODUCTS_TBL
WHERE PROD_ID <> '2345';
```

Output

```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
11235      WITCHES COSTUME               29.99
222        PLASTIC PUMPKIN 18 INCH        7.75
13         FALSE PARAFFIN TEETH           1.1
90         LIGHTED LANTERNS              14.5
15         ASSORTED COSTUMES             10
9          CANDY CORN                     1.35
6          PUMPKIN CANDY                  1.45
87         PLASTIC SPIDERS                1.05
```

```
119         ASSORTED MASKS                  4.95
1234        KEY CHAIN                       5.95
7725        LEATHER GLOVES                 24.99

11 rows selected.
```

8-2-3 Less then，Greater then 較小於，較大於

符號 <（較小）和 >（較大）可以獨自使用，或聯合其他的運算子一起使用。

Usage	Meaning
WHERE SALARY < `20000'	薪資少於 20000
WHERE SALARY > `20000'	薪資大於 20000

在第一個例子，任何少於 20000 而且不等於的值，將傳回正確的訊息，任何 20000 或較大的值將傳回錯誤訊息，較大於和較小於的運作方式剛好相反。

TYPE

```
SELECT *
FROM PRODUCTS_TBL
WHERE COST > 20;
```

Output

```
PROD_ID    PROD_DESC                       COST
---------- ------------------------- ----------
11235      WITCHES COSTUME                29.99
2345       OAK BOOKSHELF                  59.99
7725       LEATHER GLOVES                 24.99

3 rows selected.
```

在下一個例子，請注意 24.99 不被包含在查詢結果，較小於運算子不被包含在內。

TYPE
```
SELECT *
FROM PRODUCTS_TBL
WHERE COST < 24.99;
```

Output
```
PROD_ID    PROD_DESC                    COST
---------- ---------------------------- ----------
222        PLASTIC PUMPKIN 18 INCH        7.75
13         FALSE PARAFFIN TEETH           1.1
90         LIGHTED LANTERNS              14.5
15         ASSORTED COSTUMES             10
9          CANDY CORN                     1.35
6          PUMPKIN CANDY                  1.45
87         PLASTIC SPIDERS                1.05
119        ASSORTED MASKS                 4.95
1234       KEY CHAIN                      5.95

9 rows selected.
```

8-2-4 比較運算子組合的例子

如下列各例子；相等的運算子可以與較小於和較大於的運算子一起使用：

Usage	Meaning
WHERE SALARY <= `20000'	薪資少於或等於 20000
WHERE SALARY >= `20000'	薪資大於或等於 20000

較小於或等於 20000，包括 20000 和所有小於 20000 的值，在這範圍裡的任何數值都會傳回正確訊息；任何的數值大於 20000，會傳回錯誤訊息，相同的，較大於或相等於也包括 20000，與較小於或相等所表現的方式相同。

TYPE
```
SELECT *
FROM PRODUCTS_TBL
WHERE COST <= 24.99;
```

Output

```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ -----------
222        PLASTIC PUMPKIN 18 INCH         7.75
13         FALSE PARAFFIN TEETH            1.1
90         LIGHTED LANTERNS               14.5
15         ASSORTED COSTUMES              10
9          CANDY CORN                      1.35
6          PUMPKIN CANDY                   1.45
87         PLASTIC SPIDERS                 1.05
119        ASSORTED MASKS                  4.95
1234       KEY CHAIN                       5.95
7725       LEATHER GLOVES                 24.99

10 rows selected.
```

請注意 24.99 被包含在這個輸出結果中。

8-3 Logical Operators 邏輯運算子

邏輯運算子是當 SQL 關鍵字用來做比較時，替代符號的運算子。邏輯運算子將分成下列各部份討論。

- IS NULL
- BETWEEN
- IN
- LIKE
- EXISTS
- UNIQUE
- ALL and ANY

8-3-1 IS NULL

NULL 運算子用來把數值與 NULL 數值作比較。舉例來說，您可能藉由 EMPLOYEE_TBL 表格呼叫器欄裡的 NULL 數值來尋找沒有呼叫器的員工。

下列各實例顯示比較數值和 NULL 數值的使用方法：

Usage	Meaning
WHERE SALARY IS NULL	Salary has no value

下列各實例未發現 NULL 數值：

Usage	Meaning
WHERE SALARY = NULL	薪資值爲字母 N-U-L-L

TYPE

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME, PAGER
FROM EMPLOYEE_TBL
WHERE PAGER IS NULL;
```

Output

```
EMP_ID    LAST_NAM FIRST_NA PAGER
--------- -------- -------- ----------
311549902 STEPHENS TINA
442346889 PLEW     LINDA
220984332 WALLACE  MARIAH
443679012 SPURGEON TIFFANY

4 rows selected.
```

務必了解字面上的零（null）與 NULL 數值是不同的，請查閱下列各實例：

TYPE

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME, PAGER
FROM EMPLOYEE_TBL
WHERE PAGER = NULL;
```

Output

```
no rows selected.
```

8-3-2 BETWEEN

運算子之間用來查詢 set 的數值，必須設定最小值和最大值，最小值和最大值是 set 條件的一部份。

Usage	Meaning
WHERE salary between `20000'and`30000'	薪資必須介於 20000 與 30000 之間，且包含 20000 與 30000

TYPE
```
SELECT *
FROM PRODUCTS_TBL
WHERE COST BETWEEN 5.95 AND 14.5;
```

Output
```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ -----------
222        PLASTIC PUMPKIN 18 INCH         7.75
90         LIGHTED LANTERNS               14.5
15         ASSORTED COSTUMES              10
1234       KEY CHAIN                       5.95

4 rows selected.
```

請注意數值 5.95 和 14.5 包含在輸出結果中。


註解

包含是指在查詢結果裡包括最小值和最大值之間在內。

8-3-3 IN

IN 運算子用來比較數值和一連串字面上所指定的數值，若要傳回正確的訊息，比較的數值必須至少和清單其中的一個數值相同。

Usage	Meaning
WHERE SALARY IN (`20000', `30000', `40000')	薪資必須爲 20000，30000 或 40000 其中一個值。

TYPE
```
SELECT *
FROM PRODUCTS_TBL
WHERE PROD_ID IN (`13,9',`87',`119');
```

Output

```
PROD_ID    PROD_DESC                 COST
---------- ------------------------- ----------
119        ASSORTED MASKS            4.95
87         PLASTIC SPIDERS           1.05
9          CANDY CORN                1.35
13         FALSE PARAFFIN TEETH      1.1

4 rows selected.
```

使用 IN 運算子能達成和 OR 運算子相同結果，而且傳回的速度更快。

8-3-4 LIKE

LIKE 運算子被萬用字元用來比較相同數值，有二個萬用字元一起與 LIKE 運算子使用，

- percent sign（%）百分比符號
- underscore（_）底線

百分比符號代表零、一、或多重字元；底線表示單一數目或字元，符號可以互相組合使用。

使用例子是：

`WHERE SALARY LIKE '200%'`	找尋任何數值，從 200 開始後的任何數值，包括數值 200。
`WHERE SALARY LIKE '%200%'`	找尋任何位置中有 200 的數值。
`WHERE SALARY LIKE '_00%'`	找尋在第二與第三位置有 00 的任何數值。
`WHERE SALARY LIKE '2_%_%'`	找尋從 2 開始的數值。
`WHERE SALARY LIKE '%2'`	找尋任何數值結束值爲 2。
`WHERE SALARY LIKE _2%3`	找尋任何在第二位置值爲 2 結束值爲 3 的數值。

```
WHERE SALARY LIKE `2__3'
```

找尋任何五位數值開始值爲 2 結束值爲 3。

下列各實例，顯示所有產品描述結束字母爲 S：

Output

```
SELECT PROD_DESC
FROM PRODUCTS_TBL
WHERE PROD_DESC LIKE `%S';
```

Output

```
PROD_DESC
-------------------------
LIGHTED LANTERNS
ASSORTED COSTUMES
PLASTIC SPIDERS
ASSORTED MASKS
LEATHER GLOVES

5 rows selected.
```

下列各實例，顯示所有產品描述第二字母爲 S：

TYPE

```
SELECT PROD_DESC
FROM PRODUCTS_TBL
WHERE PROD_DESC LIKE `_S%';
```

Output

```
PROD_DESC
-------------------------
ASSORTED COSTUMES
ASSORTED MASKS

2 rows selected.
```

8-3-5 EXISTS

EXISTS 運算子用來尋找與指定表格符合的列。

Usage	Meaning
WHERE EXISTS (SELECT EMPLOYEE_ID FROM EMPLOYEE_TBL WHERE EMPLOYEE_ID='333333333')	從 employee_tbl 找尋 employee_id 爲 3333333333。

這個實例是 subquery 的一種，在第 14 小時會更進一步的討論：

TYPE
```
SELECT COST
FROM PRODUCTS_TBL
WHERE EXISTS ( SELECT COST
               FROM PRODUCTS_TBL
               WHERE COST > 100 );No rows selected.
----------
```

沒有列被選取，因爲儲存的記錄中，沒有大於 100 的 COST 存在。

看看下一個例子：

TYPE
```
SELECT COST
FROM PRODUCTS_TBL
WHERE EXISTS ( SELECT COST
               FROM PRODUCTS_TBL
               WHERE COST < 100 );
```

Output
```
COST
----------
     29.99
      7.75
       1.1
      14.5
        10
      1.35
      1.45
      1.05
      4.95

9 rows selected.
```

因爲有小於 100 的記錄存在，所以表格裡 COST 的記錄顯示出來。

8-3-6 UNIQUE

UNIQUE 運算子是搜尋表格內唯一的列（沒有複製的）。

Usage	Meaning
WHERE UNIQUE (SELECT SALARY FROM EMPLOYEE_TBL WHERE EMPLOYEE_ID =`333333333')	測試薪資有無重複。

8-3-7 ALL 與 ANY 運算子

ALL 運算子用來比較數值和另外不同 SET 裡的所有數值。

Usage	Meaning
WHERE SALARY > ALL (SELECT SALARY FROM EMPLOYEE_TBL WHERE CITY =`INDIANAPOLIS')	檢視薪資是否大於住在 Indianapol 的所有員工。

TYPE

```
SELECT *
FROM PRODUCTS_TBL
WHERE COST > ALL (SELECT COST
                  FROM PRODUCTS_TBL
                  WHERE COST < 10);
```

Output

```
PROD_ID    PROD_DESC                 COST
---------- ------------------------- ----------
11235      WITCHES COSTUME           29.99
90         LIGHTED LANTERNS          14.5
15         ASSORTED COSTUMES         10
2345       OAK BOOKSHELF             59.99
7725       LEATHER GLOVES            24.99

5 rows selected.
```

在這個輸出訊息，有五個記錄的 COST 大於所有紀錄的 COST 小於 10。

ANY 運算子用來比較清單裡的任何可適用條件的數值。

Usage	Meaning
WHERE SALARY > ANY (SELECT SALARY FROM EMPLOYEE_TBL WHERE CITY= `INDIANAPOLIS')	檢視薪資是否大於住在 Indianapol 的任何員工。

TYPE

```
SELECT *
FROM PRODUCTS_TBL
WHERE COST > ANY (SELECT COST
                  FROM PRODUCTS_TBL
                  WHERE COST < 10);
```

Output

```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ ----------
11235      WITCHES COSTUME                29.99
222        PLASTIC PUMPKIN 18 INCH         7.75
13         FALSE PARAFFIN TEETH             1.1
90         LIGHTED LANTERNS                14.5
15         ASSORTED COSTUMES               10
9          CANDY CORN                       1.35
6          PUMPKIN CANDY                    1.45
119        ASSORTED MASKS                   4.95
1234       KEY CHAIN                        5.95
2345       OAK BOOKSHELF                   59.99
7725       LEATHER GLOVES                  24.99

11 rows selected.
```

上列輸出的資訊，相較於 ALL 運算子，傳回更多的記錄，因爲 COST 必須比任何一個小於 10 的 COST 大，有一個 COST 1.05 的記錄不會顯示，因爲 1.05 不大於任何小於 10 的數值（事實上是 1.05）。

8-4 Conjunctive Operators

如果您要使用 SQL 指令，要如何設多重的條件限制資料？能夠結合條件稱爲 **conjunctive operators** 連結運算子。這些運算子是：

- AND
- OR

在 SQL 相同的指令，這些運算子用不同的運算子來製造複雜的比較，下列部分是描述每個運算子的行爲。

8-4-1 AND

AND 運算子允許 SQL 指令存在多重條件的子句，爲了執行 SQL 指令，不論是異動或查詢，所有的條件都必須被 AND 分開，而且必須是正確的。

Usage	Meaning
WHERE EMPLOYEE_ID =333333333 AND SALARY = `20000'	employee_id 必須符合 333333333，薪資必須爲 20000。

TYPE

```
SELECT *
FROM PRODUCTS_TBL
WHERE COST > 10
  AND COST < 30;
```

Output

```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ -----------
11235      WITCHES COSTUME              29.99
90         LIGHTED LANTERNS              14.5
7725       LEATHER GLOVES               24.99

3 rows selected.
```

輸出的結果顯示，傳回的 COST 值必須是同時大於 10 和小於 30。

TYPE
```
SELECT *
FROM PRODUCTS_TBL
WHERE PROD_ID = `7725'
  AND PROD_ID = `2345';
```

Output
```
no rows selected
```

輸出的結果顯示，資料沒有被傳回，因爲每筆資料只有一個產品編號。

8-4-2 OR

OR 運算子用來與 SQL 指令 WHERE 結合來產生多重條件，爲了執行 SQL 指令，不論是異動或查詢，至少一個條件必須用 OR 分開，而且必須是正確的。

Usage	Meaning
WHERE SALARY = `20000' OR SALARY = `30000'	薪資必須符合 20000 或 30000。

註解

在運算或比較時，邏輯運算子可以單一使用或一起使用。

TYPE
```
SELECT *
FROM PRODUCTS_TBL
WHERE PROD_ID = '7725'
   OR PROD_ID = '2345'
```

Output
```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ -----------
2345       OAK BOOKSHELF                59.99
7725       LEATHER GLOVES               24.99

2 rows selected.
```

在這輸出結果，其中一個條件必須是正確的才能傳回結果，找到二筆資料，且符合其中一個條件。


提示

當在 SQL 指令使用多重條件與運算子的時候，如果使用括號來隔開指令為合乎邏輯的群組時，您可能會發現它提升了全部的閱讀性，然而請知道誤用括號，會影響您輸出的結果。

在下一個例子，請注意使用 AND 和 OR 運算子，除此之外，請注意合理的放置括弧號，使指令的閱讀性更高。

TYPE
```
SELECT *
FROM PRODUCTS_TBL
WHERE COST > 10
  AND (PROD_ID = `222'
   OR  PROD_ID = `90'
   OR  PROD_ID = `11235' );
```

Output
```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ -----------
11235      WITCHES COSTUME                29.99
90         LIGHTED LANTERNS               14.5

2 rows selected.
```

這個輸出結果裡，COST 必須是大於 10 而且產品編號必須是在三個清單的任何一個，列沒有傳回 PROD_ID 222，因爲它的 COST 沒有大於 10。

8-5 用 NOT 運算子為否定條件

這裡都討論所有邏輯運算子測試的條件，爲了要改變條件觀點，有方法可以否定這些運算子的結果。

NOT 運算子顛倒它與邏輯運算子一起使用來的意義，NOT 可以與下列各方法的運算子一起使用：

- NOT IN
- NOT EXISTS
- NOT BETWEEN

- NOT LIKE
- IS NOT NULL

每個方法在下列的章節討論。

8-5-1 Not Equal

您已經學習如何使用 <> 運算子測試不相等，因爲爲了要測試，您實際上正在否定相等運算子，所以在這個章節提及一些 SQL 廠商所提供的不相等值。在這裡顯示第二個方法測試不相等：

Usage	Meaning
WHERE SALARY <> 20000	薪資不等於 20000。
WHERE SALARY != 20000	薪資不等於 20000。

在第二例子，您可以看見驚嘆號用來否定相等的比較，使用驚嘆號，與一些廠商使用 <> 同爲不相等運算子，視各廠商不同標準而定。

註解

檢查您的 SQL 廠商有關使用驚歎號與不相等運算子。

8-5-2 NOT BETWEEN

BETWEEN 運算子在下列例子是否定：

Usage	Meaning
WHERE Salary not between`20000' and `30000'	薪資必須不在 20000 與 30000 之間，且不包含 20000 與 30000。

TYPE
```
SELECT *
FROM PRODUCTS_TBL
WHERE COST NOT BETWEEN 5.95 AND 14.5;
```

Output
```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ -----------
11235      WITCHES COSTUME                29.99
13         FALSE PARAFFIN TEETH            1.1
9          CANDY CORN                      1.35
6          PUMPKIN CANDY                   1.45
87         PLASTIC SPIDERS                 1.05
119        ASSORTED MASKS                  4.95
2345       OAK BOOKSHELF                  59.99
7725       LEATHER GLOVES                 24.99

8 rows selected.
```


註解

請記得 BETWEEN 是指包含在內的；因此，在早先的例子，任何等於 5.95 或 14.50 的列不包含在查詢結果之中。

8-5-3 NOT IN

IN 運算子爲否定 NOT IN 的運算子，在下列例子所有薪資，會回傳任何沒有在清單的值：

Usage	Meaning
WHERE SALARY NOT IN (`20000',`30000',`40000')	要執行的話，薪資不能等於其中的值。

TYPE
```
SELECT *
FROM PRODUCTS_TBL
WHERE PROD_ID NOT IN (`13',`9',`87',`119');
```

Output
```
PROD_ID    PROD_DESC                      COST
---------- ------------------------ -----------
11235      WITCHES COSTUME                29.99
222        PLASTIC PUMPKIN 18 INCH         7.75
90         LIGHTED LANTERNS                14.5
```

```
15          ASSORTED COSTUMES          10
6           PUMPKIN CANDY               1.45
1234        KEY CHAIN                   5.95
2345        OAK BOOKSHELF              59.99
7725        LEATHER GLOVES             24.99

8 rows selected.
```

在這個輸出結果，不列出在 NOT IN 運算子之後的清單編號。

8-5-4 NOT LIKE

LIKE 或萬用字元，爲否定 NOT LIKE 運算子，當 NOT LIKE 使用時，只傳回不相似的數值。

使用的例子包括：

Usage	Meaning
WHERE SALARY NOT LIKE `200%'	找尋任何值，開始值不爲 200。
WHERE SALARY NOT LIKE `%200%'	找尋任何值沒有 200 在任何位置。
WHERE SALARY NOT LIKE `_00%'	找尋任何值第二個位置的開始值爲 00。
WHERE SALARY NOT LIKE `2_%_%'	不要找尋任何值，開始值爲 2。

TYPE

```
SELECT PROD_DESC
FROM PRODUCTS_TBL
WHERE PROD_DESC NOT LIKE 'L%';
```

Output

```
PROD_DESC
-------------------------
WITCHES COSTUME
PLASTIC PUMPKIN 18 INCH
FALSE PARAFFIN TEETH
ASSORTED COSTUMES
CANDY CORN
PUMPKIN CANDY
PLASTIC SPIDERS
ASSORTED MASKS
```

```
KEY CHAIN
OAK BOOKSHELF

10 rows selected.
```

在這個輸出結果，產品描述爲字母 L 的不被顯示。

8-5-5 IS NOT NULL

IS NULL 運算子爲否定 IS NOT NULL 運算子，用來測試值不是 NULL。

Usage	Meaning
WHERE SALARY IS NOT NULL	只有 NOT NULL 列被傳回。

TYPE

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME, PAGER
FROM EMPLOYEE_TBL
WHERE PAGER IS NOT NULL;
```

Output

```
EMP_ID    LAST_NAM FIRST_NA PAGER
--------- -------- -------- ----------
213764555 GLASS    BRANDON  3175709980
313782439 GLASS    JACOB    8887345678

2 rows selected.
```

8-5-6 NOT EXISTS

EXISTS 爲否定 NOT EXISTS 的運算子。

Usage	Meaning
WHERE NOT EXISTS (SELECT EMPLOYEE_ID WHERE EMPLOYEE_ID =`333333333'	找尋 employee_id 3333333333 不在 employee_tbl。

TYPE

```
SELECT MAX(COST)
FROM PRODUCTS_TBL
WHERE NOT EXISTS ( SELECT COST
                   FROM PRODUCTS_TBL
                   WHERE COST > 100 );
```

Output

```
MAX(COST)
----------
     59.99
```

在這個輸出結果顯示表格內最大的 COST，因爲沒有任何大於 100 的記錄存在。

8-5-7 NOT UNIQUE

UNIQUE 運算子爲否定 NOT UNIQUE 運算子。

Usage	Meaning
WHERE NOT UNIQUE (SELECT SALARY FROM EMPLOYEE_TBL)	測試表格內薪資不是單一的。

8-6 Arithmetic operators 算術運算子

算術運算子用來執行 SQL 裡的數學函數，與其他電腦語言相同，有四個傳統的數學函數運算子。

+（addition）加法

-（subtraction）減法

*（multiplication）乘法

/（division）除法

8-6-1 Addition 加法

加法用（+）符號來執行。

Usage	Meaning
SELECT SALARY + BONUS FROM EMPLOYEE_PAY_TBL	salary 加上 bonus 欄位來產生每列的總合。

Usage	Meaning
SELECT SALARY FROM EMPLOYEE_PAY_TBL WHERE SALARY + BONUS >`40000′	傳回所有 salary 加上 bonus 的值大於 40000。

8-6-2 Subtraction 減法

減法用（-）符號來執行。

Usage	Meaning
SELECT SALARY - BONUS FROM EMPLOYEE_PAY_TBL	salary 減去 bouns。
SELECT SALARY FROM EMPLOYEE_PAY_TBL WHERE SALARY *10 > `40000′	傳回所有 salary 減去 bonus 的值大於 40000。

8-6-3 Multiplication 乘法

乘法用星號（*）符號來執行。

Usage	Meaning
SELECT SALARY * 10 FROM EMPLOYEE_PAY_TBL	salary 乘 10。
SELECT SALARY FROM EMPLOYEE_PAY_TBL WHERE SALARY * 10 > `40000′	傳回所有 salary 乘 10 的值大於 40000。

下列例子裡的薪資是乘以 1.1 倍，比目前增加 10 %：

TYPE

```
SELECT EMP_ID, PAY_RATE, PAY_RATE * 1.1
FROM EMPLOYEE_PAY TBL
WHERE PAY_RATE IS NOT NULL;
```

Output

```
EMP_ID      PAY_RATE PAY_RATE*1.1
--------- ---------- ------------
442346889       14.75       16.225
220984332          11         12.1
443679015          15         16.2

3 rows selected
```

8-6-4 Division 除法

除法用（/）符號來執行。

Usage	Meaning
SELECT SALARY / 10 FROM EMPLOYEE_PAY_TBL	salary 除 10。
SELECT SALARY FROM EMPLOYEE_PAY_TBL WHERE SALARY / 10 > `40000'	傳回所有 salary 除 10 的值大於 40000。

8-6-5 算術運算子組合

算術運算子可以彼此組合使用，請記得數學裡基本的優先規則，乘法和除法運算優先執行，然後才是加法和減法運算，使用者控制數學運算規則的唯一的方法，是藉著括弧號的使用。

NEW TERM **Precedence** 優先，是數學運算式來解決數學算式或與植入 SQL 裡的函數。

Expression	Result
1 + 1 * 5	6
(1 + 1) * 5	10
10 - 4 / 2 + 1	9
(10 - 4) / (2 + 1)	2

在下列各例子，請注意刮號放置運算式的地方，如果只有使用乘法和除法，不會影響結果，優先用在這些情況不是重要的因素。

Expression	Result
4 * 6 / 2	12
(4 * 6) / 2	12
4 * (6 / 3)	12

更多的例子：

```
SELECT SALARY * 10 + 1000
FROM EMPLOYEE_PAY_TBL
WHERE SALARY > 20000

SELECT SALARY / 52 + BONUS
FROM EMPLOYEE_PAY_TBL

SELECT (SALARY - 1000 + BONUS) / 52 * 1.1
FROM EMPLOYEE_PAY_TBL

SELECT SALARY
FROM EMPLOYEE_PAY_TBL
WHERE SALARY < BONUS * 3 + 10 / 2 - 50
```

因爲不使用刮號，所以數學的優先順序會影響結果，條件對 Bonus 會產生巨大地改變。


注意

當組合數學運算子時，請考慮到優先順序，缺少刮號會產生不正確的結果。

8-7 摘要

已經介紹您 SQL 各種不同的運算子，您應該已經學習運算子的 how 和 why，也已經看到運算子單獨使用與各種不同互相組合使用的例子，如使用 AND 與 OR。您已經學習那些基本的算術函數：加法，減法，乘法和除法；比較運算子被用來測試相等、不相等、小於和大

於數值；邏輯運算子包括 BETWEEN、IN、LIKE、EXIST、ANY 與 ALL，另外也瞭解元件如何加入到 SQL 指令來敘述條件，提供 SQL 較好的控制與處理。

8-7-1 Q&A

Q 我可以有多於一個的 AND 在 WHERE 子句嗎？

A 是的，事實上，所有的運算子能重複使用。例子如下：

```
FROM EMPLOYEE_PAY_TBL
WHERE SALARY > 20000
AND BONUS BETWEEN 1000 AND 3000
AND POSITION = VICE PRESIDENT
```

Q 如果我用單引號放在 WHERE 子句的數字資料格式，會怎樣呢？

A 您的查詢仍然會處理，數字欄位不一定需要單引號。

8-8 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

8-8-1 隨堂測驗

1. 是非題：使用 OR 運算子時，兩者的條件必須都是正確的。
2. 是非題：使用 IN 運算子時，所有的指定數值必須相配。
3. 是非題：AND 運算子可以用在 SELECT 和 WHERE 子句。

4. 下列哪一個 SELECT 指令是錯誤的？

a.

```
SELECT SALARY
FROM EMPLOYEE_PAY_TBL
WHERE SALARY BETWEEN 20000, 30000
```

b.

```
SELECT SALARY + DATE_HIRE
FROM EMPLOYEE_PAY_TBL
```

c.

```
SELECT SALARY, BONUS
FROM EMPLOYEE_PAY_TBL
WHERE DATE_HIRE BETWEEN '22-SEP-97'
AND '23-NOV-97'
AND POSITION = 'SALES'
OR POSITION = 'MARKETING'
AND EMPLOYEE_ID LIKE '%55%'
```

8-8-2 練習題

1. 使用 CUSTOMER_TBL 做描述：

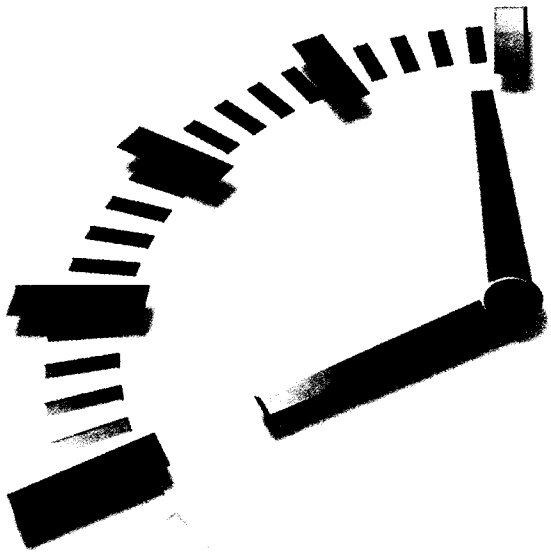
```
describe customer_tbl

Name                           Null?    Type
------------------------------ -------- ----
CUST_ID                        NOT NULL VARCHAR2(10)
CUST_NAME                      NOT NULL VARCHAR2(30)
CUST_ADDRESS                   NOT NULL VARCHAR2(20)
CUST_CITY                      NOT NULL VARCHAR2(12)
CUST_STATE                     NOT NULL CHAR(2)
CUST_ZIP                       NOT NULL CHAR(5)
CUST_PHONE                              NUMBER(10)
CUST_FAX                                NUMBER(10)
```

寫出 SELECT 指令來傳回客戶 ID 和居住在 Indiana，Ohio，Michigan，Illinois 的客戶和客戶名稱開始值為 A 或 B 名稱的客戶（用字母排序）。

2. 使用 PRODUCTS_TBL 描述：

```
describe products_tbl
Name                            Null?    Type
------------------------------- -------- ----
PROD_ID                         NOT NULL VARCHAR2(10)
PROD_DESC                       NOT NULL VARCHAR2(25)
COST                            NOT NULL NUMBER(6,2)
```

寫出 SELECT 指令來傳回產品 ID、產品敘述與產品價值，限制產品價值在 $1.00 和 $12.50 之間。



第 9 小時

資料查詢結果總結

在這個小時，您將學習有關 SQL 的聚合函數，您可以使用聚合函數執行多種有用的函數。

本小時的重點包括：

- 函數是什麼？
- 如何使用函數？
- 何時使用函數？
- 使用聚合函數
- 用聚合函數來做摘要資料的運用
- 使用函數產生的結果

9-1 聚合函數是什麼？

函數是 SQL 欄裡面爲了產生輸出，而使用的關鍵字，函數是連同欄名稱或運算句一起使用的指令，SQL 裡有許多類型的函數，這個小時將討論聚合函數。

聚合函數是包含筆數、摘要和表示資料限度的函數，如 COUNT、SUM、MAX、MIN 和 AVG 是聚合函數。

NEW TERM 聚合函數：聚合函數提供訊息摘要給 SQL 指令，例如筆數，總計和平均數。

這個小時討論的聚合函數是：

- COUNT（筆數）
- SUM（總計）
- MAX（最大值）
- MIN（最小值）
- AVG（平均數）

下列查詢大多數爲這個小時所用的資料：

TYPE
```
select *
from products_tbl;
```

Output
```
PROD_ID    PROD_DESC                      COST
---------- ------------------------- ----------
11235      WITCHES COSTUME                29.99
222        PLASTIC PUMPKIN 18 INCH         7.75
13         FALSE PARAFFIN TEETH            1.1
90         LIGHTED LANTERNS               14.5
15         ASSORTED COSTUMES              10
9          CANDY CORN                      1.35
6          PUMPKIN CANDY                   1.45
87         PLASTIC SPIDERS                 1.05
119        ASSORTED MASKS                  4.95
1234       KEY CHAIN                       5.95
```

```
2345        OAK BOOKSHELF                    59.99
7725        LEATHER GLOVES                   24.99

12 rows selected.
```

下列的查詢結果列出一些沒有呼叫器的員工：

TYPE
```
select emp_id, last_name, first_name, pager
from employee_tbl;
```

Output
```
EMP_ID    LAST_NAM FIRST_NA PAGER
--------- -------- -------- ----------
311549902 STEPHENS TINA
442346889 PLEW     LINDA
213764555 GLASS    BRANDON  3175709980
313782439 GLASS    JACOB    8887345678
220984332 WALLACE  MARIAH
443679012 SPURGEON TIFFANY

6 rows selected.
```


9-1-1 COUNT 函數

COUNT 函數主要用在資料統計列的筆數或一些不包含數值的欄，當使用查詢的時候，COUNT 函數傳回數字值，當 COUNT 函數與 DISTINCT 指令一起用時，只有統計不同的資料列，ALL（DISTINCT 的反義字）是預設值；因爲使用 ALL 在語法是不必需的，所以如果沒有事先指定，會統計複製的資料列，COUNT 函數其他的選項是和星號一起使用，COUNT 和星號一起使用的時候，會統計表格內所有列，包括複本、不論 NULL 數值有無包含在欄內。COUNT 函數的語法如下列：

```
COUNT [ (*) | (DISTINCT | ALL ] COLUMN NAME)
```


註解

DISTINCT 指令不能夠與 COUNT (*) 一起使用，但是可以與 COUNT(column_name) 一起用。

Usage	Meaning
SELECT COUNT(EMPLOYEE_ID) FROM EMPLOYEE_PAY_ID	統計員工 ID
SELECT COUNT(DISTINCT SALARY) FROM EMPLOYEE_PAY_TBL	統計單一列
SELECT COUNT(ALL SALARY) FROM EMPLOYEE_PAY_TBL	統計所有薪資列
SELECT COUNT(*) FROM EMPLOYEE_TBL	統計 employee 表格的所有列

COUNT(*) 使用在下列例子來統計 EMPLOYEE_TBL 表格內所有紀錄。有六個員工。

TYPE
```
SELECT COUNT(*)
FROM EMPLOYEE_TBL;
```

Output
```
COUNT(*)
----------
         6
```

COUNT(EMP_ID) 在下一個例子用在統計所有存在表格內的員工編號，因爲所有員工都有編號，所以傳回結果和上一個查詢一樣。

TYPE
```
SELECT COUNT(EMP_ID)
FROM EMPLOYEE_TBL;
```

Output
```
COUNT(EMP_ID)
-------------
            6
```

COUNT(PAGER) 在這個例子被用來統計所有有呼叫器員工的記錄，只有二位員工有呼叫器。

TYPE
```
SELECT COUNT(PAGER)
FROM EMPLOYEE_TBL;
```

Output
```
COUNT(PAGER)
-------------
            2
```

在這裡所顯示的 ORDERS_TBL 表格，在下一個 COUNT 的例子使用：

TYPE
```
SELECT *
FROM ORDERS_TBL;
```

Output
```
ORD_NUM    CUST_ID    PROD_ID          QTY ORD_DATE
---------- ---------- ---------- ---------- --------
56A901     232        11235               1 22-OCT-97
56A917     12         907               100 30-SEP-97
32A132     43         222                25 10-OCT-97
16C17      090        222                 2 17-OCT-97
18D778     287        90                 10 17-OCT-97
23E934     432        13                 20 15-OCT-97
90C461     560        1234                2

7 rows selected.
```

最後這個例子得到 ORDERS_TBL 表格裡的所有不同產品的編號。

TYPE
```
SELECT COUNT(DISTINCT(PROD_ID))
FROM ORDERS_TBL;
```

Output
```
COUNT(DISTINCT(PROD_ID))
------------------------
                       5
```

PROD_ID 222 在表格裡有二個項目，減少從 6 到 5 不同的數值。


註解

因為 COUNT 函數統計列數，資料格式不是統計的一部份，列可以包含任何資料格式的欄位。

9-1-2 SUM 函數

SUM 函數用來傳回一群資料列在欄位數值的總數，SUM 函數也可以連同 DISTINCT 一起使用，當 SUM 與 DISTINCT 一起使用時，只有統計不同的資料列，這樣做沒有多大的意義，因爲這樣，您總計全體資料列的值就不正確了，因爲有的資料列被遺漏了。SUM 函數的語法如下：

```
SUM ([ DISTINCT ] COLUMN NAME)
```


註解

函數中獨立變數的數值，是為了要使用 SUM 函數。要使用 SUM 函數，欄所使用的資料格式只能為數字，其他，例如字元或日期都不可以。

Usage	Meaning
SELECT SUM(SALARY) FROM EMPLOYEE_PAY_TBL	薪資總和
SELECT SUM(DISTINCT SALARY) FROM EMPLOYEE_PAY_TBL	不同的薪資總和

從 PRODUCTS_TBL 表格傳回，總計所有價值的總數。

TYPE
```
SELECT SUM(COST)
FROM PRODUCTS_TBL;
```

Output
```
SUM(COST)
----------
    163.07
```

9-1-3 AVG 函數

AVG 函數使用找尋一群資料列的平均值，當使用 DISTINCT 指令的時候，AVG 函數傳回不同資料列的平均值，AVG 函數的語法如下：

```
AVG ([ DISTINCT ] COLUMN NAME)
```


註解

函數中的獨立變數的值必須為數值的，才能使用 AVG 函數。

Usage	Meaning
SELECT AVG(SALARY) FROM EMPLOYEE_PAY_TBL	傳回薪資平均
SELECT AVG(DISTINCT SALARY)	傳回薪資不同的平均

```
FROM EMPLOYEE_PAY_TBL
```

在第一個例子，傳回 PRODUCTS_TBL 表格欄裡的所有數值的平均值。

TYPE
```
SELECT AVG(COST)
FROM PRODUCTS_TBL;
```

Output
```
 AVG(COST)
----------
13.5891667
```

這個例子中，相同的查詢使用二個聚合函數，因爲一些員工是每小時支付薪資而其他的人是支付薪水，您希望傳回 PAY_RATE 和 SALARY 的平均值。

TYPE
```
SELECT AVG(PAY_RATE), AVG(SALARY)
FROM EMPLOYEE_PAY_TBL;
```

Output
```
AVG(PAY_RATE) AVG(SALARY)
------------- -----------
13.5833333       30000
```

9-1-4 MAX 函數

MAX 函數用來傳回一群資料列裡欄的最大值，當使用 MAX 函數的時候，NULL 數值會被忽略。DISTINCT 指令是選擇項，雖然所有資料列的最大值和不同資料列的最大值是相同的，但它是無用的。

```
MAX([ DISTINCT ] COLUMN NAME)
```

Usage	Meaning
SELECT MAX(SALARY) FROM EMPLOYEE_PAY_TBL	傳回最大的薪資值
SELECT MAX(DISTINCT SALARY) FROM EMPLOYEE_PAY_TBL	傳回最大的薪資值

下列例子傳回 PRODUCTS_TBL 表格裡 COST 欄的最大值：

TYPE
```
SELECT MAX(COST)
FROM PRODUCTS_TBL;
```

Output
```
MAX(COST)
----------
     59.99
```

9-1-5 MIN 函數

MIN 函數傳回一群資料列的最小值，當使用 MIN 函數的時候，NULL 會被忽略。DISTINCT 指令是選擇項，然而，所有資料列的最小值和不同資料列的最小值是相同的，但它是無用的。

```
MIN([ DISTINCT ] COLUMN NAME)
```

Usage	Meaning
SELECT MIN(SALARY) FROM EMPLOYEE_PAY_TBL	傳回最小的薪資值
SELECT MIN(DISTINCT SALARY) FROM EMPLOYEE_PAY_TBL	傳回最小的薪資值

下列例子傳回 PRODUCTS_TBL 表格裡 COST 欄的最小值：

TYPE
```
SELECT MIN(COST)
FROM PRODUCTS_TBL;
```

Output
```
MIN(COST)
----------
      1.05
```


注意

當使用聚合函數的時候，請特別注意使用 DISTINCT，不正當的使用，可能會傳回不同的結果，聚合函數的用意是傳回表格裡所有資料列的摘要資料。

最後的例子，結合聚合函數與算術運算子：

TYPE
```
SELECT COUNT(ORD_NUM), SUM(QTY),
       SUM(QTY) / COUNT(ORD_NUM) AVG_QTY
FROM ORDERS_TBL;
```

Output
```
COUNT(ORD_NUM)    SUM(QTY)     AVG_QTY
-------------- ---------- ----------
160 22.8571429
```

您已經執行所有的數字排序的統計，演算所有的總計，二個數字互除，也知道如何試算每個項目的平均值，您也建立欄位別名用來做電腦運算：AVG_QTY。

9-2 摘要

聚合函數是非常有用的，而且使用相當簡單，您已經學習如何統計欄的數值、表格資料列的筆數、取得欄的最大值和最小值、演算欄裡數值的總計和演算欄裡數值的平均值。請記得，使用 NULL 的時候，聚合函數不考慮 NULL 數值，除了當格式使用 COUNT(*) 的時候以外。在 SQL 裡，聚合函數是您所學習的第一個函數，以後會談到更多。聚合函數也可以爲群組所使用，在下個小時會討論。當您學習其他的函數時，您會看見大多數的函數語法類似，他們的使用觀念也是彼此有關也是容易瞭解。

9-2-1 Q&A

Q 當使用 MAX 或 MIN 時，NULL 數值為什麼忽略？

A NULL 數值表示在那裡不關緊要。

Q 當使用 COUNT 函數時，資料格式為什麼有關係？

A COUNT 函數只有統計資料列。

9-3 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

9-3-1 隨堂測驗

1. 是非題：AVG 函數傳回所有選定列的平均值包括任何的 NULL 數值。
2. 是非題：SUM 函數被使用在增加欄的總計 。
3. 是非題：COUNT(*) 函數統計表格裡的所有列。
4. 下列的 SELECT 指令會正常工作？如果不可以，哪些指令需要修改？

a.

```
SELECT COUNT *
FROM EMPLOYEE_PAY_TBL;
```

b.

```
SELECT COUNT(EMPLOYEE_ID), SALARY
FROM EMPLOYEE_PAY_TBL;
```

c.

```
SELECT MIN(BONUS), MAX(SALARY)
FROM EMPLOYEE_PAY_TBL
WHERE SALARY > 20000;
```

9-3-2 練習題

1. 使用 EMPLOYEE_PAY_TBL：

```
EMP_ID     POSITION       DATE_HIRE PAY_RATE DATE_LAST   SALARY     BONUS
---------- -------------- --------- -------- --------- --------- ---------
311549902 MARKETING      23-MAY-89           01-MAY-97     30000      2000
442346889 TEAM LEADER    17-JUN-90     14.75 01-JUN-97
213764555 SALES MANAGER  14-AUG-94           01-AUG-97     40000      3000
313782439 SALESMAN       28-JUN-97                         20000      1000
220984332 SHIPPER        22-JUL-96        11 01-JUL-97
443679012 SHIPPER        14-JAN-91        15 01-JAN-97

6 rows selected.
```

建立 SQL 指令來尋找：

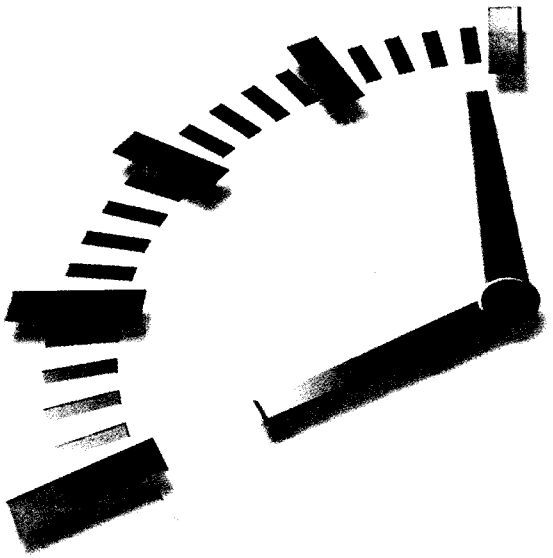
a. The average salary

b. The maximum bonus

c. The total salaries

d. The minimum pay rate

e. The total rows in the table



第 10 小時

資料的分類和排序

您已經學習到該如何查詢資料庫，而且有組織的傳回資料，也學習如何排序查詢的資料，在這個小時中，您將學習如何分類從查詢傳回的資料到群組裡，並增加其可讀性。

本小時的重點包括：

- 為什麼要群組資料？
- GROUP BY 子句
- 群組數值函數
- 如何使用群組函數
- 藉著欄位來分群組
- GROUP BY 與 ORDER BY 的比較
- HAVING 子句

10-1 為什麼要群組資料？

群組資料是一種用合乎邏輯的方式，結合欄位裡重複數值的程序，舉例來說，資料庫可能包含有關客戶的資料；許多客戶居住在不同的城市，當然有一些客戶居住在相同的城市，您可能希望查詢住在每座城市裡客戶資料，之前您經由城市來群組員工資料，而且會建立摘要報表。

假如您希望依照每座城市找出一般員工的薪資，您將會藉由使用聚合函數 AVG 來取 SALARY 欄位的平均值，如同您上個小時所學習的一樣，與藉由使用 GROUP BY 子句，用城市去做資料群組聚集的輸出。

群組資料是藉著使用 GROUP BY 子句與 SELECT 子句（查詢）所完成的。上個小時，您學習如何使用聚合函數，在這小時您會看見聚合函數如何連同 GROUP BY 子句，對資料庫產生更有效的結果。

10-2 GROUP BY 子句

GROUP BY 子句與 SELECT 指令一起使用，安排完全相同的資料到群組之內，GROUP BY 子句與 SELECT 指令的順序是在 WHERE 指令之後，但是在 ORDER BY 子句之前。GROUP BY 查詢子句的排序如下：

```
SELECT
FROM
WHERE
GROUP BY
ORDER BY
```

GROUP BY 子句必須隨著 WHERE 的條件，而且如果有使用 ORDER BY 子句，也必須一起執行，包括 GROUP BY 子句的 SELECT 指令語法如下：

```
SELECT COLUMN1, COLUMN2
FROM TABLE1, TABLE2
WHERE CONDITIONS
GROUP BY COLUMN1, COLUMN2
ORDER BY COLUMN1, COLUMN2
```

下列的章節所舉的例子，說明如何在多種情形下使用 GROUP BY 子句。

10-2-1 群組選擇資料

群組資料是簡單的程序，選擇的欄位可能被 GROUP BY 子句引用爲參考的欄位，如果不能讓 SELECT 指令找到欄位，GROUP BY 子句可能無法使用。

其實您想一下是合乎邏輯的，如果資料不顯示，您如何能群組報表上的資料？如果欄位名稱已經選取，那麼選取的名稱必須放到 GROUP BY 子句。欄位名稱也可以用數目字表示，如何使用數目字來表示在稍後這個小時會討論。當群組資料的時候，欄位的次序不必須與 SELECT 指令的次序相同。

10-2-2 群組函數

典型的群組函數與 GROUP BY 子句一起使用，安排群組裡的資料包括 AVG、MAX、MIN、SUM 與 COUNT，這些是在第 9 小時 “資料查詢結果總結” 所學習的聚合函數。請記得聚合函數在第 9 小時都是使用單一數值；現在您可以使用複數聚合函數數值。

10-2-3 建立群組和使用聚合函數

當使用 GROUP BY 的時候，SELECT 指令必須符合某些條件，無論選擇甚麼欄位，都必須在 GROUP BY 子句旁顯示，當他們顯示在 GROUP BY 子句時，欄位的排列規則不必與 SELECT 子句相同，如果選取 SELECT 子句裡的欄位，欄位所指定的名稱必須在 GROUP BY 子句中使用，這裡有 GROUP BY 子句語法的範例：

Usage

```
SELECT EMP_ID, CITY
FROM EMPLOYEE_TBL
GROUP BY CITY, EMP_ID;
```

ANALYSIS SQL 指令從 EMPLOYEE_TBL 選擇 emp_id 和 CITY，讓傳回的資料先用 CITY 聚集，然後再用 emp_id 聚集。


註解

選擇的欄位順序，必須在 GROUP BY 子句裡和欄位的順序互相呼應。

Usage

```
SELECT EMP_ID, SUM(SALARY)
FROM EMPLOYEE_PAY_TBL
GROUP BY SALARY, EMP_ID;
```

ANALYSIS 這個 SQL 指令傳回 EMP_ID 和總計全體群組的薪資，並群組薪資和員工 ID。

Usage

```
SELECT SUM(SALARY)
FROM EMPLOYEE_PAY_TBL;
```

ANALYSIS 這個 SQL 指令傳回 EMPLOYEE_PAY_TBL 的全體薪資總計。

Usage

```
SELECT SUM(SALARY)
FROM EMPLOYEE_PAY_TBL
GROUP BY SALARY;
```

ANALYSIS 這個 SQL 指令傳回不同群組的薪資總計。

在這個例子中，您可以看見在 EMPLOYEE_TBL 表格內有三個不同的城市，使用實際的資料所做的例子如下：

TYPE
```
SELECT CITY
FROM EMPLOYEE_TBL;
```

Output
```
CITY
------------
GREENWOOD
INDIANAPOLIS
WHITELAND
INDIANAPOLIS
INDIANAPOLIS
INDIANAPOLIS

6 rows selected.
```


在下列各例子裡，您先選擇城市和記錄每個城市，之後藉著 GROUP BY 子句使用群組的方式，可以統合每座不同城市的資料。

TYPE
```
SELECT CITY, COUNT(*)
FROM EMPLOYEE_TBL
GROUP BY CITY;
```

Output
```
CITY            COUNT(*)
------------ ----------
GREENWOOD              1
INDIANAPOLIS           4
WHITELAND              1

3 rows selected.
```

下列各暫時表格是用 EMPLOYEE_TBL 和 EMPLOYEE_PAY_TBL 查詢出來的，您很快會學習如何利用一個查詢結合二個表格。

TYPE
```
select *
from emp_pay_tmp;
```

Output

```
CITY         LAST_NAM FIRST_NA   PAY_RATE      SALARY
------------ -------- -------- ---------- ----------
GREENWOOD    STEPHENS TINA                      30000
INDIANAPOLIS PLEW     LINDA         14.75
WHITELAND    GLASS    BRANDON                   40000
INDIANAPOLIS GLASS    JACOB                     20000
INDIANAPOLIS WALLACE  MARIAH            11
INDIANAPOLIS SPURGEON TIFFANY           15

6 rows selected.
```

現在您使用聚合函數 AVG，平均每座城市的薪資與稅率，在 GREENWOOD 或 WHITELAND 是沒有平均稅率，因爲沒有以小時支付薪資的員工住那些城市裡。

TYPE

```
SELECT CITY, AVG(PAY_RATE), AVG(SALARY)
FROM EMP_PAY_TMP
GROUP BY CITY;
```

Output

```
CITY         AVG(PAY_RATE) AVG(SALARY)
------------ ------------- -----------
GREENWOOD                        30000
INDIANAPOLIS    13.5833333       20000
WHITELAND                        40000

3 rows selected.
```

在下一個例子，您使用查詢裡的多重元件傳回群組的資料，您仍然希望只看見 INDIANAPOLIS 和 WHITELAND 的平均薪資與稅率，既然您在其他欄位正在使用聚合函數並用城市來群組資料，所以您沒有其他的選擇，最後您想要報表順序先 2 再 3，即先一般的稅率再算平均薪資，練習下列的細節和輸出結果。

TYPE

```
SELECT CITY, AVG(PAY_RATE), AVG(SALARY)
FROM EMP_PAY_TMP
WHERE CITY IN ('INDIANAPOLIS','WHITELAND')
GROUP BY CITY
ORDER BY 2,3;
```

Output

```
CITY           AVG(PAY_RATE) AVG(SALARY)
-------------- ------------- -----------
INDIANAPOLIS     13.5833333       20000
WHITELAND                         40000
```

有數值的欄位排在 NULL 之前，所以 INDIANAPOLIS 優先顯示，而沒有選擇 GREENWOOD 是因爲如果顯示的話，將在 WHITELAND 之前，因爲 GREENWOOD 平均薪資是 $30,000（第二種在 ORDER BY 子句分類是在平均薪資上）。

下一個例子顯示 MAX 和 MIN 聚合函數與 GROUP BY 子句的使用。

TYPE

```
SELECT CITY, MAX(PAY_RATE), MIN(SALARY)
FROM EMP_PAY_TMP
GROUP BY CITY;
```

Output

```
CITY           MAX(PAY_RATE) MIN(SALARY)
-------------- ------------- -----------
GREENWOOD                          30000
INDIANAPOLIS              15       20000
WHITELAND                          40000

3 rows selected.
```

10-2-4 用數目表現欄位名稱

不像 ORDER BY 子句，除了使用 UNION 與欄位名稱不同外，GROUP BY 子句不能夠藉由整數來表示欄位名稱的排序，這裡是用數目表現欄位名稱排序的例子：

```
SELECT EMP_ID, SUM(SALARY)
FROM EMPLOYEE_PAY_TBL
UNION
SELECT EMP_ID, SUM(PAY_RATE)
FROM EMPLOYEE_PAY_TBL
GROUP BY 2, 1;
```

ANALYSIS 這個 SQL 指令傳回員工 ID，而且爲薪資群組做了總計，2 是表示用薪資群組排序，而 1 是表示 emp_id。

10-3 GROUP BY 與 ORDER BY

您應該了解 GROUP BY 子句與 ORDER BY 子句，兩者都是用來排序資料，ORDER BY 子句特別用在資料分類的查詢；爲了要適當群組資料，GROUP BY 子句必須排序來自查詢的資料，因此，GROUP BY 子句能和 ORDER BY 一樣，用在資料的排序。

使用 GROUP BY 來排序時，有一些缺點存在：

- 若選擇為非聚合欄位，則必須在 GROUP BY 子句旁邊列出。
- 如同 ORDER BY 一樣，在 SELECT 關鍵字後，GROUP BY 不會使用整數表示欄位。
- 除非使用聚合函數，否則 GROUP BY 子句通常不是必需的。

下列的例子，是執行 GROUP BY 子句來替代 ORDER BY 子句：

TYPE
```
SELECT LAST_NAME, FIRST_NAME, CITY
FROM EMPLOYEE_TBL
GROUP BY LAST_NAME;
```

Output
```
SELECT LAST_NAME, FIRST_NAME, CITY
                  *
ERROR at line 1:
ORA-00979: not a GROUP BY expression
```

在這個例子，收到來自資料庫伺服器的錯誤訊息，說明 FIRST_NAME 不在 GROUP BY 的運算式內，請記得除了聚合欄位以外（那些被聚合函數選中的欄位），所有欄位和 SELECT 裡的運算式，必須列在 GROUP BY 子句旁，。

在下一個例子中可以看到之前的問題藉由 SELECT 裡的所有運算式加入 GROUP BY 子句而解決：

TYPE
```
SELECT LAST_NAME, FIRST_NAME, CITY
FROM EMPLOYEE_TBL
GROUP BY LAST_NAME, FIRST_NAME, CITY;
```

Output

```
LAST_NAM FIRST_NA CITY
-------- -------- ------------
GLASS    BRANDON  WHITELAND
GLASS    JACOB    INDIANAPOLIS
PLEW     LINDA    INDIANAPOLIS
SPURGEON TIFFANY  INDIANAPOLIS
STEPHENS TINA     GREENWOOD
WALLACE  MARIAH   INDIANAPOLIS

6 rows selected.
```

在這個例子中，相同的欄位從相同的表格選擇出來，所有的欄位在 GROUP BY 子句旁邊列出，正如他們出現在 SELECT 關鍵字之後，結果排列的順序是 LAST_NAME 第一、FIRST_NAME 第二，然後 CITY 第三。這些結果完成了之前的 ORDER BY 子句，然而，它可能幫助您了解 GROUP BY 如何使用，爲了要群組資料，資料一定要先經過排序。

下列各例子表示，從 EMPLOYEE_TBL 的選擇並且使用 GROUP BY 子句用 CITY 來排序，進而引導進入下一個例子。

TYPE

```
SELECT CITY, LAST_NAME
FROM EMPLOYEE_TBL
GROUP BY CITY, LAST_NAME;
```

Output

```
CITY         LAST_NAM
------------ --------
GREENWOOD    STEPHENS
INDIANAPOLIS GLASS
INDIANAPOLIS PLEW
INDIANAPOLIS SPURGEON
INDIANAPOLIS WALLACE
WHITELAND    GLASS

6 rows selected.
```

請注意，早先結果裡的資料排序是依每座城市人的 LAST_NAME，統計所有在 EMPLOYEE_TBL 表格裡的員工記錄，結果是用 CITY 群組，但是排序是以統計每座城市爲優先。

TYPE
```
SELECT CITY, COUNT(*)
FROM EMPLOYEE_TBL
GROUP BY CITY
ORDER BY 2,1;
```

OUTPUT
```
CITY            COUNT(*)
------------ ----------
GREENWOOD              1
WHITELAND              1
INDIANAPOLIS           4
```

請注意排序的結果，是每座統計城市（1-4）的優先排序，前二座城市的統計數是都是 1，既然統計數是相同的，所以在 ORDER BY 子中，第一個運算式用城市來排序，因此 GREENWOOD 被放置在 WHITELAND 之前。

雖然 GROUP BY 和 ORDER BY 執行類似的功能，還是有一主要的差異，GROUP BY 設計爲聚集相同的資料，而 ORDER BY 只設計讓資料用特定的方式排序，GROUP BY 和 ORDER BY 可以用在相同的 SELECT 指令，但是必須特定規則，在 SELECT 指令中 GROUP BY 總是放置在 ORDER BY 之前。


提示

GROUP BY 可以用在 CREATE VIEW 指令分類資料，但 ORDER BY 則不可以使用，CREATE VIEW 指令在第 20 小時 “建立、使用檢視與同義字” 會深入的討論。

10-4 HAVING 子句

HAVING 子句是在 SELECT 指令連 GROUP BY 一起使用，告訴 GROUP BY 哪些群組應該包含在輸出結果內，HAVING 與 GROUP BY 的關係有如 WHERE 與 SELECT，換句話說，WHERE 子句是放置條件在選擇欄位上，然而 HAVING 子句條件是放在 GROUP BY 子句所建立的群組上。

這裡介紹查詢放置 HAVING 子句的位置：

```
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
```

在查詢時，HAVING 子句必須跟著 GROUP BY 子句，如果有使用 ORDER BY 子句，則也必須放置在 ORDER BY 子句之前。

包含 HAVING 子句與 SELECT 指令的語法如下：

```
SELECT COLUMN1, COLUMN2
FROM TABLE1, TABLE2
WHERE CONDITIONS
GROUP BY COLUMN1, COLUMN2
HAVING CONDITIONS
ORDER BY COLUMN1, COLUMN2
```


在下列各項例子，除了 GREENWOOD 外，選擇所有城市平均薪資與稅率，用城市群組輸出結果，但是只顯示平均薪資大於 $20,000 的群組（城市），藉由每座城市的平均薪資爲排序輸出結果的依據。

TYPE

```
SELECT CITY, AVG(PAY_RATE), AVG(SALARY)
FROM EMP_PAY_TMP
WHERE CITY <> 'GREENWOOD'
GROUP BY CITY
HAVING AVG(SALARY) > 20000
ORDER BY 3;
```

Output

```
CITY          AVG(PAY_RATE) AVG(SALARY)
------------ ------------- -----------
WHITELAND                         40000

1 row selected.
```

這個查詢，爲什麼只有傳回一個列？

- GREENWOOD 城市從 WHERE 子句除去。
- 因為平均薪資是 20000，INDIANAPOLIS 是 20000 而不是大於 20000，所以刪除。

10-5 摘要

您已經學習在查詢結果裡使用 GROUP BY 子句、GROUP BY 子句與 SQL 主要聚合函數像 SUM、AVG、MAX、MIN 和 COUNT 的使用方式，GROUP BY 性質與 ORDER BY 類似，兩者都是分類查詢的結果。GROUP BY 子句必須合乎邏輯排序群組資料結果，但是也可以用在分類排序資料，相對的 ORDER BY 則簡單許多。HAVING 子句，是 GROUP BY 子句的擴充，用來放置建立群組查詢條件，在 SELECT 指令 WHERE 子句用於放置查詢的條件。在下個小時您將學習一個新的重要函數，允許您進一步處理查詢的結果。

10-5-1 Q&A

Q 在使用 SELECT 指令時，ORDER BY 子句是否一定要一起使用 GROUP BY 子句？

A 不，使用 GROUP BY 子句絕對是有選擇性的，但是與 ORDER BY 子句一起使用是非常有用的。

Q 群組值是什麼？

A 如來自 EMPLOYEE_TBL 的 CITY 欄位，如果您選擇員工名稱和城市，輸出結果以城市群組，所有相同的城市會排在一起。

10-6 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

10-6-1 隨堂測驗

1. 下列各 SQL 指令可以工作嗎？

a.

```
SELECT SUM(SALARY), EMP_ID
FROM EMPLOYEE PAY TBL
GROUP BY 1 and 2;
```

b.

```
SELECT EMP_ID, MAX(SALARY)
FROM EMPLOYEE_PAY_TBL
GROUP BY SALARY, EMP_ID;
```

c.

```
SELECT EMP_ID, COUNT(SALARY)
FROM EMPLOYEE_PAY_TBL
ORDER BY EMP ID
GROUP BY SALARY;
```


2. 是非題：當使用 HAVING 子句的時候，您也必須使用 GROUP BY 子句？

3. 是非題：這 SQL 指令可以傳回薪資總數群組。

```
SELECT SUM(SALARY)
FROM EMPLOYEE_PAY_TBL;
```

4. 是非題：被選擇欄位的順序，必須與 GROUP BY 子句排序相同。

5. 是非題：HAVING 子句告訴 GROUP BY 子句必須包含那一個群組 。

10-6-2 練習題

1. 寫個能從 EMPLOYEE_TBL 傳回員工 ID、員工名稱和城市的 SQL 指令，並藉由城市欄位先聚集。

2. 寫個能從 EMPLOYEE_TBL 傳回城市和統計每座城市員工的 SQL 指令，增加 HAVING 子句，只有顯示那些有超過二位員工所被統計的城市。

第 11 小時

更改資料展現結構

在這個小時這段時期，您學習該如何廣泛的使用函數，重新更改與排列資料輸出的結果，如一些主要的 SQL 廠商所使用的基本變數與 ANSI 標準函數與其他函數。

本小時的重點包括：

- 介紹字元函數（Character functions）
- 如何與何時該使用字元函數
- ANSI SQL 函數例子
- 許多廠商所指定使用的轉換函數（Conversion functions）概觀與例子
- 如何與何時該使用轉換函數

11-1 ANSI 字元函數的觀念

字元函數是用來表現 SQL 裡字串的格式，與他們如何儲存在表格裡的函數，這章的第一個部份將討論 ANSI 所涵蓋的字元函數觀念，這個小時的第二個部份將展示有關使用各種不同 SQL 廠商特有函數的實際例子，在這小時討論的 ANSI 函數包括串接 concatenation，子字串 substring，翻譯 TRANSLATE，轉換 CONVERT 和位置 POSITION。

11-1-1 串接 Concatenation

串接(Concatenation)是聯合二分開的字串成爲一個字串的程序，舉例來說，您可能要串接每個人的姓和名成爲一個完整的名字。

JOHN 和 SMITH 串接 = JOHN SMITH。

11-1-2 子字串 Substring

子字串（substring）的觀念是擷取一個字串的部份，或 "sub" 字串，舉例來說，下列值是 JOHNSON 的 substrings：

J JOHN JO ON SON -

11-1-3 翻譯 TRANSLATE

翻譯函數（TRANSLATE function）是用在翻譯字串、字元成爲另外的字串，通常有三個引數與翻譯函數有關：轉換字串，轉換一連串的字元，和取代一連串字元。常用的例子在這個小時的下一個部份會被顯示。

11-1-4 轉換 CONVERT

轉換函數（CONVERT function）用來轉換某格式的全體字串，成爲另外相同長度的格式，要轉換的格式是由程式設計師提供，轉換的功能與翻譯函數是有許多相似之處，事實上，轉換通常透過使用翻譯函數，或由其他特定 SQL 廠商所提供相似的函數。

11-1-5 位置 POSITION

位置函數（POSITION function）用來傳回字串在另外的一個字串裡面的確實位置，舉例來說，您可能要傳回字元 "M" 在字串 SMITH 裡的位置，位置函數會傳回數值 2，數值 2 是被搜尋字串在目標字元（或字串）的位置。

11-2 各種不同共用字元函數

字元函數主要地用在比較（compare）、加入（join）、搜尋（search）和抽選（extract）一個段落裡的字串或欄的數值，有幾個字元函數可供 SQL 程式設計師使用的。

下列的段落，舉例說明一些 SQL 領導廠商使用 ANSI 的觀念，例如在 Oracle、Sybase、SQLBase、Informix 和 SQL Server。

註解

在這本書所討論的 ANSI 觀念，只是單純的概念而已，由 ANSI 所提供的標準，只是作為在 SQL 關聯式資料庫裡的該如何使用與實現的指導方針，相對於，請記住在這章所討論那些特定的函數，在您所使用特定的 SQL 廠商，不是都可以準確使用的，觀念和函數工作方式通常是相同的，但是函數名稱與確實的語法可能不一致。

11-2-1 串接

串接與其他函數，在各種不同的 SQL 廠商之中表現些微地不同，下列各項例展示在 Oracle 和 SQL Server 裡串接的使用。

在 Oracle

SELECT 'JOHN' || 'SON'	returns	JOHNSON

在 SQL Server

SELECT 'JOHN' + 'SON'	returns	JOHNSON

Oracle 的語法爲：

```
column_name || ['' || ] column_name [ column_name ]
```

SQL Server 的語法爲：

```
column_name + [ '' + ] column_name [ column_name ]
```

Usage	Meaning
SELECT CITY + STATE FROM EMPLOYEE_TBL;	在 SQL Server 語法裡串接城市和洲名稱成爲一個值。
SELECT CITY ||', '||STATEFROM EMPLOYEE_TBL;	這是 Oracle 的語法，串接城市名稱和洲名稱成爲一個值，放置個逗號在城市與洲名稱之間。
SELECT CITY +' ' + STATE FROM EMPLOYEE_TBL;	在 SQL Server 語法裡串接城市和洲名稱成爲一個值，在二個原始值之間放置個空格。

例子：

TYPE

```
SELECT LAST_NAME || ,  || FIRST_NAME NAME
FROM EMPLOYEE_TBL;
```

Output
```
NAME
------------------
STEPHENS, TINA
PLEW, LINDA
GLASS, BRANDON
GLASS, JACOB
WALLACE, MARIAH
SPURGEON, TIFFANY

6 rows selected.
```


註解

請注意在 SQL 裡指令裡，逗點與單引號的使用方式，如果使用單引號，就可允許使用大多的字元和符號，一些廠商可能使用雙引號來標示文字的字串。

11-2-2 翻譯 TRANSLATE

翻譯函數搜尋一連串字元和檢查特定的字元並記下位置，它搜尋相同的位置，然後以新的數值代替那個字元。語法是：

```
TRANSLATE(CHARACTER SET, VALUE1, VALUE2)
```

Usage	Meaning
SELECT TRANSLATE (CITY,'IND','ABC' FROM EMPLOYEE_TBL;	這是 SQL 語法，字串裡有 I 的全部取代成 A，字串裡有 N 的全部取代成 B，字串裡有 D 的全部取代成 C。

下列例子，使用真實的資料舉例說明：

TYPE
```
SELECT CITY, TRANSLATE(CITY,'IND','ABC')
FROM EMPLOYEE_TBL;
```

Output
```
CITY         TRANSLATE(CI
------------ ------------
GREENWOOD    GREEBWOOC
INDIANAPOLIS ABCAABAPOLAS
WHITELAND    WHATELABC
INDIANAPOLIS ABCAABAPOLAS
```

```
INDIANAPOLIS ABCAABAPOLAS
INDIANAPOLIS ABCAABAPOLAS

6 rows selected.
```

請注意，在這個例子一起取代字串爲，把全部的 I 取代成 A、N 爲 B、D 爲 C，城市 INDIANAPOLIS 裡，IND 與 ABC 一起取代，但是在 GREENWOOD，D 也換成 C，請注意數值 WHITELAND 如何翻譯。

11-2-3 取代 REPLACE

取代函數（REPLACE function）用來以敘述每當發生字元代替字元的狀況，使用這函數的方法與翻譯函數非常類似，只有一個特定的字元或字串，將在另外的一個字串裡面取代。語法是：

```
REPLACE(、VALUE', 'VALUE', [ NULL ] 、VALUE、)
```

Usage	Meaning
SELECT REPLACEFIRST_ NAME, 'T', 'B') FROM EMPLOYEE_TBL	這指令傳回所有的名，而且更改字串裡有 T 的全部取代成 B。

TYPE

```
SELECT CITY, REPLACE(CITY,'I','Z')
FROM EMPLOYEE_TBL;
```

Output

```
CITY        REPLACE(CITY
----------- ------------
GREENWOOD   GREENWOOD
INDIANAPOLIS ZNDZANAPOLZS
WHITELAND   WHZTELAND
INDIANAPOLIS ZNDZANAPOLZS
INDIANAPOLIS ZNDZANAPOLZS
INDIANAPOLIS ZNDZANAPOLZS

6 rows selected.
```

11-2-4 大寫 UPPER

大多數的廠商藉由使用函數的方法來控制資料，大寫函數（UPPER function）用來轉換特定的小寫字母字串成爲大寫字母，語法如下：

```
UPPER(character string)
```

Usage	Meaning
SELECT UPPER(LAST_NAME) FROM EMPLOYEE_TBL;	轉換所有在 LAST_NAME 欄位的字元成爲大寫。

TYPE

```
SELECT UPPER(CITY)
FROM EMPLOYEE_TBL;
```

Output

```
UPPER(CITY)
------------
GREENWOOD
INDIANAPOLIS
WHITELAND
INDIANAPOLIS
INDIANAPOLIS
INDIANAPOLIS

6 rows selected.
```

11

11-2-5 小寫 LOWER

相反於大寫函數，小寫函數用來轉換特定的字串大寫字母成爲小寫字母爲。語法如下：

```
LOWER(character string)
```

Usage	Meaning
SELECT LOWER(LAST_NAME) FROM EMPLOYEE_TBL;	轉換所有在 LAST_NAME 欄位的字元成爲小寫。

TYPE

```
SELECT LOWER(CITY)
FROM EMPLOYEE_TBL;
```

Output

```
LOWER(CITY)
------------
greenwood
indianapolis
whiteland
indianapolis
indianapolis
indianapolis

6 rows selected.
```

11-2-6 SUBSTR

在運算式中 substring 通常是大多 SQL 廠商最常用的，但是函數名稱可能不一致，如下列各 Oracle 和 SQL Server 例子所示。

Oracle 的語法爲：

```
SUBSTR(COLUMN NAME, STARTING POSITION, LENGTH)
```

SQL Server 的語法爲：

```
SUBSTRING(COLUMN NAME, STARTING POSITION, LENGTH)
```

唯一不一樣的是函數名稱。

Usage	Meaning
SELECT SUBSTRING (EMP_ID,1,3) FROM EMPLOYEE_TBL	這是 SQL 指令，傳回 EMP_ID 前三個字元。
SELECT SUBSTRING (EMP_ID,4,2) EMPLOYEE_TBL	這是 SQL 指令，傳回 EMP_ID 第四、第五個字元。
SELECT SUBSTRING (EMP_ID,6,4) FROM EMPLOYEE_TBL	這是 SQL 指令，傳回 EMP_ID 第六到第十個字元。

使用 Microsoft SQL Server 的範例：

TYPE
```
SELECT EMP_ID, SUBSTRING(EMP_ID,1,3)
FROM EMPLOYEE_TBL
```

Output
```
EMP_ID    SUB
--------- ---
311549902 311
442346889 442
213764555 213
313782439 313
220984332 220
443679012 443

6 rows affected.
```

使用 Oracle7 的範例：

TYPE
```
SELECT EMP_ID, SUBSTRING(EMP_ID,1,3)
FROM EMPLOYEE_TBL
```

Output
```
EMP_ID    SUB
--------- ---
311549902 311
442346889 442
213764555 213
313782439 313
220984332 220
443679012 443

6 rows selected.
```


註解

請注意在二個查詢的回應不同，第一個例子傳回被影響(affected)的有六排和第二個傳回的被選擇(selected)有六排，您看見在不同廠商之間的差異。

11-2-7 INSTR

INSTR 函數是位置函數（POSITION function）的一種變化，它用來搜尋特定組的字元，而且報告那些字元的位置。

```
INSTR(COLUMN NAME, 'SET',
[ START POSITION [ , OCCURRENCE ] ]);
```

Usage	Meaning
SELECT INSTR(STATE ,'I',1,1) FROM EMPLOYEE_TBL;	這是 SQL 指令，傳回在 EMPLOYEE_TBL 第一個出現字母 I 的位置。

TYPE

```
SELECT PROD_DESC,
       INSTR(PROD_DESC,'A',1,1)
FROM PRODUCTS_TBL;
```

Output

```
PROD_DESC                 INSTR(PROD_DESC,A,1,1)
------------------------- ----------------------
WITCHES COSTUME                                0
PLASTIC PUMPKIN 18 INCH                        3
FALSE PARAFFIN TEETH                           2
LIGHTED LANTERNS                              10
ASSORTED COSTUMES                              1
CANDY CORN                                     2
PUMPKIN CANDY                                 10
PLASTIC SPIDERS                                3
ASSORTED MASKS                                 1

9 rows selected.
```

請注意，如果被搜尋的字元不在字串裡，將會傳回數值 0。

11-2-8 LTRIM

LTRIM 函數是截割部份字串的另外一個方法，這函數與 SUBSTRING 是屬於相同的家族，LTRIM 用來整理來自字串左邊的字元。語法是：

```
LTRIM(CHARACTER STRING [ ,'set' ])
```

Usage	Meaning
SELECT LTRIM(FIRST_ NAME,'LES') FROM CUSTOMER_TBL WHERE FIRST_NAME = 'LESLIE';	這是 SQL 指令，從所有 LESLIE 的名稱，截割 LES。

TYPE
```
SELECT POSITION, LTRIM(POSITION, 'SALES')
FROM EMPLOYEE_PAY_TBL;
```

Output
```
POSITION        LTRIM(POSITION,
--------------- ---------------
MARKETING       MARKETING
TEAM LEADER     TEAM LEADER
SALES MANAGER   MANAGER
SALESMAN        MAN
SHIPPER         HIPPER
SHIPPER         HIPPER

6 rows selected.
```

SHIPPER 裡的 S 被截割，即使 SHIPPER 不包含字串 SALES，忽略了 SALES 的前面的四個字元。搜尋的字元必須要與搜尋字串的次序相同，並且必須是在字串的最左邊。

11-2-9 RTRIM

像 LTRIM，RTRIM 函數用來從字串的右邊截割字元。語法是：

```
RTRIM(CHARACTER STRING [ , 'set' ])
```

Usage	Meaning
SELECT RTRIM(FIRST_ NAME, 'ON' FROM EMPLOYEE_TBL *WHERE FIRST_NAME* = 'BRANDON';	在 SQL 指令，傳回 BRANDON 這名，開啓截割，留下 BRAND 爲輸出結果。

TYPE
```
SELECT POSITION, RTRIM(POSITION, 'ER')
FROM EMPLOYEE_PAY_TBL;
```

Output
```
POSITION        RTRIM(POSITION,
--------------  --------------
MARKETING       MARKETING
TEAM LEADER     TEAM LEAD
SALES MANAGER   SALES MANAG
SALESMAN        SALESMAN
SHIPPER         SHIPP
SHIPPER         SHIPP

6 rows selected.
```

字串 ER 從所有適用的字串右邊被截割。

11-2-10 DECODE

譯碼函數（DECODE function），不屬於 ANSI，起碼在寫這本書的時候還不是，在這討論是因爲他非常有影響力的，這函數在 SQLBase、Oracle 和其他的廠商使用，譯碼用來搜尋數值或字串的字串，而且如果發現字串，將顯示交互的字串，成爲查詢輸出的部份。語法是：

```
DECODE(COLUMN NAME, 'SEARCH1', 'RETURN1' ,[ 'SEARCH2' , 'RETURN2' , 、
DEFAULT VALUE'])
```

Usage	Meaning
SELECT DECODE(LAST_NAME, 'SMITH', 'JONES', 'OTHER') FROM EMPLOYEE_ TBL;	這查詢找尋在 EMPLOYEE_TBL 所有的姓，如果找到 SMITH 就展示 JONES，所有其他名展示爲 OTHER，爲預設值。

在下列例子，使用譯碼在 EMPLOYEE_TBL 裡的城市值上：

TYPE
```
SELECT CITY,
       DECODE(CITY,'INDIANAPOLIS','INDY',
                   'GREENWOOD','GREEN', 'OTHER')
  FROM EMPLOYEE_TBL;
```

Output

```
CITY          DECOD
------------- ----
GREENWOOD     GREEN
INDIANAPOLIS  INDY
WHITELAND     OTHER
INDIANAPOLIS  INDY
INDIANAPOLIS  INDY
INDIANAPOLIS  INDY

6 rows selected.
```

輸出表示數值 INDIANAPOLIS 顯示爲 INDY、GREENWOOD 顯示爲 GREEN、而且所有的其他城市顯示爲 OTHER。

11-3 各種的字元函數

下列章節展示值提到的字元函數，再一次強調，這些是在主要的廠商中常見的函數。

11-3-1 找尋數值的長度

長度函數（LENGTH function）是用來找尋位元組裡字串、數字、日期或運算式長度的函數。語法是：

```
LENGTH(CHARACTER STRING)
```

Usage	Meaning
SELECT LENGTH (LAST_NAME) FROM EMPLOYEE_TBL;	這是 SQL 指令傳回每位員工名字的長度。

TYPE

```
select prod_desc, length(prod_desc)
from products_tbl;
```

Output

```
PROD_DESC                          LENGTH(PROD_DESC)
---------------------------------------
WITCHES COSTUME                                15
PLASTIC PUMPKIN 18 INCH                        23
FALSE PARAFFIN TEETH                           19
LIGHTED LANTERNS                               16
ASSORTED COSTUMES                              17
CANDY CORN                                     10
PUMPKIN CANDY                                  13
PLASTIC SPIDERS                                15
ASSORTED MASKS                                 14

9 rows selected.
```

11-3-2 NVL (NULL Value)

NVL 函數用來傳回如果運算式的資料是 NULL，許多資料格式能使用 NVL；然而，數值和替代值必須是相同的資料格式。語法是：

```
NVL(VALUE, SUBSTITUTION)
```

Usage	Meaning
SELECT NVL(SALARY, '00000') FROM EMPLOYEE_PAY_TBL;	這是 SQL 指令用 0000 替代 NULL 值。

TYPE

```
SELECT PAGER, NVL(PAGER,9999999999)
FROM EMPLOYEE_TBL;
```

Output

```
PAGER      NVL(PAGER,
---------- ----------
           9999999999
           9999999999
3175709980 3175709980
8887345678 8887345678
           9999999999
           9999999999

6 rows selected.
```

唯一 NULL 值是被表示爲 9999999999。

11-3-3 LPAD

LPAD（左邊填補）用來增加字元或空格到字串的左邊。語法是：

```
LPAD(CHARACTER SET)
```

下列例子填補到每個產品描述的左邊，在真實的數值和填補期之間總共有 30 個字元。

TYPE
```
SELECT LPAD(PROD_DESC,30, '.') PRODUCT
FROM PRODUCTS_TBL;
```

Output
```
PRODUCT
------------------------------
...............WITCHES COSTUME
.......PLASTIC PUMPKIN 18 INCH
..........FALSE PARAFFIN TEETH
..............LIGHTED LANTERNS
.............ASSORTED COSTUMES
.....................CANDY CORN
.................PUMPKIN CANDY
..............PLASTIC SPIDERS
................ASSORTED MASKS

9 rows selected.
```

11-3-4 RPAD

RPAD（右邊填補）用來增加字元或空格到那字串的右邊。語法是：

```
RPAD(CHARACTER SET)
```

下列例子填補到每個產品描述的右邊，在真實的數值和填補期之間總共有 30 個字元。

TYPE
```
SELECT RPAD(PROD_DESC,30, '.') PRODUCT
FROM PRODUCTS_TBL;
```

Output
```
PRODUCT
------------------------------
WITCHES COSTUME...............
PLASTIC PUMPKIN 18 INCH.......
```

```
FALSE PARAFFIN TEETH.....
LIGHTED LANTERNS.....
ASSORTED COSTUMES.....
CANDY CORN.....
PUMPKIN CANDY.....
PLASTIC SPIDERS.....
ASSORTED MASKS.....

9 rows selected.
```

11-3-5 聽起來起來像什麼？

SOUNDEX 函數用來傳回運算式像另外的一個運算式的聲音，這函數被用在當搜尋的時候，當您不知道精確的數值或類似的數值，是否被一致地輸入時。語法是：

TYPE

```
WHERE SOUNDEX(EXPRESSION) = SOUNDEX(EXPRESSION)
SELECT LAST_NAME
FROM EMPLOYEE_TBL
WHERE SOUNDEX(LAST_NAME) = SOUNDEX('STEVENS')
```

Output

```
LAST_NAM
--------
STEPHENS

1 row selected.
```

11-3-6 ASCII

ASCII 函數用來傳回 ASCII 字串的最左邊字元的表示方法。語法是：

```
ASCII (CHARACTER SET)
```

例子：

```
ASCII('A') returns 65
ASCII('B') returns 66
ASCII('C') returns 67
```

11-4 數學函數

數學函數是在 SQL 廠商裡也是差不多相同，這些函數是依照數學規則，處理資料庫裡的數字數值。最常用的函數包括：

絕對值	(ABS)
捨取	(ROUND)
更號	(SQRT)
正副號值	(SIGN)
次方	(POWER)
最高與最低值	(CEIL, FLOOR)
指數值	(EXP)
SIN, COS, TAN	

一般數學函數的語法是：

```
FUNCTION(EXPRESSION)
```


11-5 轉換函數（Conversion Functions）

轉換函數用來把資料格式轉換成另外的一個 資料格式，舉例來說，有時您要把字元資料轉換成數值資料，您可能正常地以字元格式儲存資料，但是有時候您爲了作計算而要轉換成數值，數學函數與計算不允許資料以字元格式表現出來。

一般類型的資料轉換：

- 字元到數值
- 數值到字元
- 字元到日期
- 日期到字元

最初的二個轉換類型在這個小時討論，其餘的轉換類型在第 12 小時會討論“理解日期和時間”，日期和時間的儲存方式會更詳細地討論。


註解

當必需轉換資料格的時候，一些廠商可能會提示。

11-5-1 轉換字元、字串到數值

關於在數值的資料格式和字元、字串資料格式之間，應注意二件不同事物。

1. 算術運算式和函數可使用在數值上。
2. 在輸出結果時，數值是向右邊對齊，然而字元資料格式是向左邊對齊。

當字元、字串被轉換到數值的時候，被提到的二個屬性正好適用在數值上。

一些廠商可能沒有函數轉換字元字串到數值，但有些轉換函數是產生類似這結果，在任一情況，請接洽您廠商文件所規定爲轉換而定的語法。


註解

轉換到數字的字元、字串裡，典型地是從 0 到 9；附加符號如減號和逗點，也能用來表示正數、負數字和小數點，舉例來說，字串 STEVE 不能夠被轉換到數字，然而個人社會福利號碼可以用字元的方式儲存，但是能夠使用轉換函數，容易地轉換到數值。

使用 ORACLE 轉換函數，數值轉換的例子：

TYPE
```
SELECT EMP_ID, TO_NUMBER(EMP_ID)
FROM EMPLOYEE_TBL;
```

Output

```
EMP_ID              TO_NUMBER(EMP_ID)
--------            ----------------
311549902           311549902
442346889           442346889
213764555           213764555
313782439           313782439
220984332           220984332
443679012           443679012

6 rows selected.
```

在轉換後，職員編號是右邊對齊。

提示

資料的對齊方式是識別資料格式最簡單的方法。

11-5-2 轉換數字到字串

對字元而言，數值轉換到字串是與字串轉換到數值剛好相反。

使用 Microsoft SQL Server 的函數，用 TransactSQL 轉換數值到字元的例子：

TYPE

```
SELECT PAY = PAY_RATE, NEW_PAY = STR(PAY_RATE)
FROM EMPLOYEE_PAY_TBL
WHERE PAY_RATE IS NOT NULL;
```

Output

```
       PAY NEW_PAY
---------- ----------------------------------------
      17.5 17.5
     14.75 14.75
     18.25 18.25
      12.8 12.8
        11 11
        15 15

6 rows affected.
```

相同的例子使用 ORACLE 轉換函數：

TYPE
```
SELECT PAY_RATE, TO_CHAR(PAY_RATE)
FROM EMPLOYEE_PAY_TBL
WHERE PAY_RATE IS NOT NULL;
```

Output
```
  PAY_RATE TO_CHAR(PAY_RATE)
---------- ----------------------------------------
      17.5 17.5
     14.75 14.75
     18.25 18.25
      12.8 12.8
        11 11
        15 15

6 rows selected.
```

11-6 組合字元函數的觀念

大多數的函數能在單一 SQL 指令中組合，如果不允許函數組合，那麼 SQL 將變得太有限制，下列例子表示在查詢裡，可以彼此組合使用的一些函數，：

TYPE
```
SELECT LAST_NAME || ',' || FIRST_NAME NAME,
       SUBSTR(EMP_ID,1,3) || '-' ||
       SUBSTR(EMP_ID,4,2) || '-' ||
       SUBSTR(EMP_ID,6,4) ID
FROM EMPLOYEE_TBL;
```

Output
```
NAME                 ID
-------------------- -----------
STEPHENS, TINA       311-54-9902
PLEW, LINDA          442-34-6889
GLASS, BRANDON       213-76-4555
GLASS, JACOB         313-78-2439
WALLACE, MARIAH      220-98-4332
SPURGEON, TIFFANY    443-67-9012

6 rows selected.
```

這個例子組合查詢（有 substring 的串接）裡的二個函數，分別地把 EMP_ID 分入三小段，您用串接的方式連接那些段落，而得到社會福利編號。

TYPE
```
SELECT SUM(LENGTH(LAST_NAME) + LENGTH(FIRST_NAME)) TOTAL
FROM EMPLOYEE_TBL;
```

Output
```
     TOTAL
----------
        71

1 row selected.
```

這個例子使用長度函數和數學運算子（+），把名的長度加入每個欄姓的長度，sun 函數然後計算所有姓名總長度。


註解

當箝入函數在 SQL 指令內部函數時，請記得先處理第一個函數，然後接下來的每個函數，從內到外一一處理。

11

11-7 摘要

您已經介紹在 SQL 指令常用的各種不同函數，用作修正或加強輸出結果表現的方式，函數包括字元、數學和轉換函數，了解它是以 ANSI 標準爲基準是非常重要的，了解 SQL 廠商如何實現與應用，但是廠商在有限度的改變下，是不用精確的語法，大多數廠商有標準函數，而且使用 ANSI 一致的觀念，但是每個廠商有他自己可用特定的函數清單，函數名字可能不一致，而且語法可能不一致，但是有所有函數觀念是相同的。

11-7-1 Q&A

Q ANSI 標準是適用所有函數嗎？

A 不，不是所有的函數是 ANSI SQL 標準，函數，像資料格式時常是依廠商而不同，好比在這所選擇的廠商與所舉的幾個函數例子在內，許多廠商使用相似的函數（雖然他們可能有些微地不一致），請接洽您的廠商，與瞭解特定函數的使用方法。。

Q 當使用函數的時候，在資料庫的資料會實際上被改變嗎？

A 不，當使用函數的時候，在資料庫的資料不會被改變，函數典型地使用在查詢，爲了要製照輸出結果。

11-8 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

11-8-1 隨堂測驗

描述與可能相配的函數。

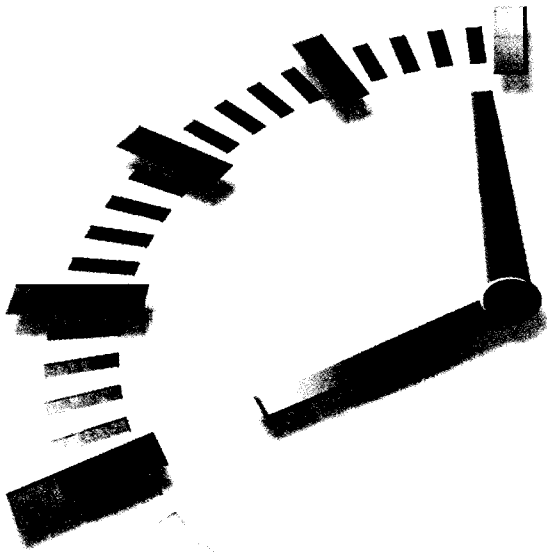
	DESCRIPTIONS	FUNCTIONS
a.	使用選擇部分字串	||
		RPAD
		LPAD
b.	從字元左邊或右邊截割字元	LENGTH
		UPPER
		SOUNDEX
c.	更改所有字元爲小寫	LTRIM
		RTRIM
		LOWER

	DESCRIPTIONS	FUNCTIONS
d.	找尋字元長度	SUBSTR
e.	組合字元	

1. 是非題：SOUNDEX 函數用來比較可能聽起來一樣的字串。
2. 是非題：當在查詢裡的函數箝入其他函數裡面的時候，最外圍的函數總是首先處理。

11-8-2 練習題

1. 使用適當的函數轉換字串 “hello” 為大寫字母。
2. 使用適當的函數，只列印字串 Johnson 的最初四個字元。
3. 使用函數去串接般字串 JOHN 和 SON。



第 12 小時

理解日期與時間

在這個小時內，您會 SQL 學習日期與時間的性質，這小時不只詳細討論日期與時間的資料格式；您也會瞭解一些廠商如何使用日期，一些常用的規則，與在需要時格式化日期與時間。

本小時的重點包括：

- 理解日期和時間
- 日期和時間如何被儲存
- 典型的日期與時間格式
- 如何使用日期函數
- 如何使用日期轉換


註解

您現在應該知道，有許多不同的 SQL 廠商，這本書顯示 ANSI 標準和最常用的非標準的函數，命令和運算子，Oracle 用在那些例子裡，甚至在 Oracle，日期能儲存成不同的格式，您必須檢查特定的廠商有關日期儲存的格式，無論它如何儲存，您的廠商應該有函數來轉換日期格式。

12-1 日期如何被儲存？

每個廠商都有預設日期和時間的格式，這預設值與其他資料格式一樣，在不同的廠商之中可能有所不同，先藉由溫習下列區段的日期與時間資料格式和它的元件標準格式開始，然後在一些 SQL 的主要廠商，看見日期與時間的資料格式，包括 Oracle，Sybase 和 Microsoft SQL Server。

12-1-1 標準日期與時間資料格式

有三種標準 SQL 日期與時間（DATETIME）的資料格式（DATETIME）儲存方式：

資料格式	用法
DATE	儲存日期
TIME	儲存時間
TIMESTAMP	儲存日期和時間

每資料格式的有效格式範圍：

日期
Format: YYYY-MM-DD
Range: 0001-01-01 to 9999-12-31

時間

Format: HH:MI:SS.nn...

Range: 00:00:00 to 23:59:61.999

時間戳記

Format: YYYY-MM-DD HH:MI:SS.nn...

Range: 0001-01-01 00:00:00 to 9999-12-31 23:59:61.999...

12-1-2 DATETIME 元件

DATETIME 元件是包含日期與時間，屬於 DATETIME 定義的部份，y94 在下列，我們列出每種元件對 DATETIME 元件的有效範圍值。

YEAR	0001 to 9999
MONTH	01 to 12
DAY	01 to 31
HOUR	00 to 23
MINUTE	00 to 59
SECOND	00.000 to 61.999

秒能用小數來表示，允許十分之一秒、百分之一秒等等，每種元件是說明，每日我們所處理的時間元件。

12-1-3 特定廠商的資料格式

與其他的資料格式一樣，每個廠商提供它自己的表示方法和語法，這區段展示三種產品（Oracle，Sybase，SQLBase）如何顯示日期與時間。

產品	資料格式	使用方式
Oracle	DATE	儲存日期和時間兩者訊息。
Sybase	DATETIME	儲存日期和時間兩者訊息。
	SMALLDATETIME	儲存日期和時間兩者訊息，但是較包括範圍小於 DATETIME。
SQLBase	DATETIME	儲存日期和時間兩者訊息。
	TIMESTAMP	儲存日期和時間兩者訊息。
	DATE	儲存日期值。
	TIME	儲存時間值。


註解

每廠商有它自己的特定日期與時間的資料格式，然而，大多數的廠商遵從 ANSI 標準的日期和時間，並包含在他們聯合的資料格式中，內部日期儲存的方法，是依照廠商而定。

12-2 日期函數 Date Functions

日期函數在選項方面，是依照每個特定的 SQL 廠商而定，日期函數，與字元、字串函數類似，用來處理日期與時間的表示方法，可用的日期函數時常用於日期與時間的輸出格式，比較彼此日期與時間值，計算在日期之間的間隔等等。

12-2-1 現在日期

您可能會有問題：我要由資料庫如何得到現在的日期？要從資料庫取回現在的日期，可從幾種情形開始說明，要得到現在的日期，通常地是傳回與儲存日期比較的日期，或傳回現在的日期值，如時間戳記（timestamp）。

最後在主機上資料庫儲存的現在日期，稱爲系統日期，資料庫與作業系統有適當的界面，能因自己的需要向系統傳回日期或解決資料庫所提出的請求，例如查詢。

看看二個不同的廠商，爲了要傳回系統日期，而下的一些指令。

Sybase 使用 GETDATE() 函數呼叫系統日期，這函數在下列查詢使用，輸出結果將會是傳回今天的日期是否爲 1998 新年前夕。

```
SELECT GETDATE()

Dec 31, 1997
```


註解

這本書討論的選項大多數可以應用在兩廠商 Sybase 和 Microsoft 二家廠商，因為兩者使用的資料庫 SERVER 為 SQL SERVER，兩家廠商也使用擴充標準的 SQL 語言－Transact-SQL 聞名。

Oracle 使用稱爲虛擬 SYSDATE 傳回現在日期，SYSDATE 當作表格裡其他的欄使用，而且能從資料庫裡選擇任何表格，即使它不是表格實際上定義的部份。

Oracle 爲了要傳回系統日期，下列指令傳回今天是否爲 1998 之前的新年前夕：

TYPE
```
SELECT SYSDATE FROM TABLE_NAME
```

Output
```
31-DEC-97
```


12-2-2 時區 Time Zones

時區的使用是因爲處理日期與時間的資訊，舉例來說，美國中央的下午 6:00 與澳洲的時間是不相等的，雖然確實的時間點是相同的，那些居住在日光節約的時區的人，一年必須調整兩次時鐘，如果時區是您在維護資料時候的考量，您必需考慮 SQL 廠商有關時區的問題並且執行時間轉換。

一些常用的時區和他們的縮寫：

縮寫	定義
AST, ADT	大西洋標準，日光節約時區
BST, BDT	Bering 標準，日光節約時區
CST, CDT	CDT 中央標準，日光節約時區
EST, EDT	東方標準，日光節約時區
GMT	格林威治標準時間
HST, HDT	阿拉斯加州/夏威夷標準，日光節約時區
MST, MDT	Mountain 標準，日光節約時區
NST	Newfoundland 標準，日光節約時區
PST, PDT	太平洋標準，日光節約時區
YST, YDT	Yukon 標準，日光節約時區


註解

一些廠商允許您處理不同時區的函數；然而，不可能所有廠商都支援使用時區函數，一定要查證您特別廠商，依資料庫需要的情況使用時區函數。

12-2-3 增加時間到日期

日、月與時間的其他部份都能加入日期，藉此比較彼此的日期，或在查詢 WHERE 的子句時，提供特定的條件。

間隔能使用在 DATETIME，把一段時間加入日期。如標準定義，在那些下列各例子，間隔用來製造 DATETIME 的值：

```
DATE '1997-12-31' + INTERVAL '1' DAY

'1998-01-01'

DATE '1997-12-31' + INTERVAL '1' MONTH

'1998-01-31'
```

使用 SQL Server 函數例子 DATEADD：

TYPE
```
SELECT DATEADD(MONTH, 1, DATE_HIRE)
FROM EMPLOYEE_PAY_TBL
```

Output
```
DATE_HIRE ADD_MONTH
--------- ----------------  --------
23-MAY-89 23-JUN-89
17-JUN-90 17-JUL-90
14-AUG-94 14-SEP-94
28-JUN-97 28-JUL-97
22-JUL-96 22-AUG-96
14-JAN-91 14-FEB-91

6 rows affected.
```

使用 Oracle 函數 ADD_MONTHS 例子：

TYPE
```
SELECT DATE_HIRE, ADD_MONTHS(DATE_HIRE,1)
FROM EMPLOYEE_PAY_TBL;
```

Output
```
DATE_HIRE ADD_MONTH
--------- ----------------  --------
23-MAY-89 23-JUN-89
17-JUN-90 17-JUL-90
14-AUG-94 14-SEP-94
28-JUN-97 28-JUL-97
22-JUL-96 22-AUG-96
14-JAN-91 14-FEB-91

6 rows selected.
```

增加一天到 Oracle 裡的：

TYPE
```
SELECT DATE_HIRE, DATE_HIRE + 1
FROM EMPLOYEE_PAY_TBL
WHERE EMP_ID = 311549902;
```

Output
```
DATE_HIRE DATE_HIRE
--------- ----------------  --------
23-MAY-89 24-MAY-89

1 row selected.
```

請注意在這些例子雖然 SQL Server 和 Oracle 與 ANSI 的例子語法結構不同，但都是基於 SQL 標準相同觀念轉換而來。

12-2-4 比較日期和時間

重疊（OVERLAPS）是對 DATETIME 非常有用的標準 SQL 條件運算子，重疊運算子用來比較二個時間點並且傳回 Boolean 值，是正確或錯誤的，全因二個時間點是否重疊而定，下列比較傳回正確值：

```
(TIME '01:00:00', TIME '05:59:00')
(TIME '05:00:00', TIME '07:00:00')
```

下列比較傳回錯誤值：

```
(TIME '01:00:00', TIME '05:59:00')
(TIME '06:00:00', TIME '07:00:00')
```

12-2-5 各種日期函數

下列目錄表示在 SQL Server 和 Oracle 二個廠商裡，所存在的一些有影響力的日期函數。

SQL Server	
DATEPART	傳回實數給 datepart 當作日期
DATENAME	傳回文字給 datepart 當作日期
GETDATE()	傳回系統日期
DATEDIFF	傳回二個日期之間敘述不同的部分日期，例如日、分和秒

Oracle	
NEXT_DAY	傳回星期內所敘述日子的下個日子（舉例來說，星期五）
MONTHS_BETWEEN	傳回二個時間點之間的月數

12-3 轉換日期

日期需要轉換是因爲有許多不同的理由，轉換主要用來改變資料格式所定義的值，如 DATETIME，或特別廠商的任何有效的資料格式。

傳統轉換日期的理由：

- 比較不同資料格式的日期值
- 格式化日期
- 轉換字串成為日期格式

ANSI CAST 運算子用來把資料格式轉換成其他的資料格式。

基本的語法如下：

```
CAST ( EXPRESSION AS NEW_DATA_TYPE )
```

下列依照一些廠商特定的語法舉例說明：

- DATETIME 值的部份表示法
- 轉換日期為字元
- 轉換字元為日期

12-3-1 日期圖片

日期圖片是用來格式化從資料庫所摘錄下的資訊，並加以格式化，成爲想要的格式，日期圖片不可能被所有的 SQL 廠商所使用。

沒有日期圖片和一些典型的轉換函數，日期和時間的資訊是從資料庫傳回並使用預定的格式，例如：

```
1997-12-31
31-DEC-9731
1997-12-31 23:59:01.11
...
```

如果您想要日期顯示爲下列的方式，您必須轉換日期從 DATETIME 格式到字元格式。

```
December 31, 1997
```

這是藉由廠商特定函數來完成的，較詳細的舉例說明如下。

Sybase date pictures:

yy	年
qq	季
mm	月
dy	年日
wk	星期
dw	星期的日子
hh	小時
mi	分鐘
ss	秒
ms	百萬分之秒

Oracle date pictures:

AD	紀元後
AM	上午
BC	紀元前
CC	世紀
D	在星期內的天數
DD	月內的天數
DDD	在一年內的天數
DAY	拼出大寫日子，如星期一 MONDAY
Day	拼出日子，如星期一 Monday
day	拼出小寫日子，如星期一 monday
DY	星期大寫縮寫如 MON

Oracle date pictures:

Dy	星期縮寫如 Mon
dy	星期小寫縮寫如 mon
HH	每天的小時
HH12	每天的小時，12 小時制
HH24	每天的小時，24 小時制
J	自從西元前 12-31-4713 年 Julius Caesar 以後的日子
MI	每小時的分鐘
MM	月份的數字
MON	月份大寫縮寫，如一月 JAN
Mon	月份縮寫，如一月 Jan
mon	月份小寫縮寫，如一月 jan
MONTH	拼出大寫月份，如一月 JANUARY
Month	拼出月份，如一月 January
month	拼出小寫月份，如一月 january
PM	下午
Q	第幾季
RM	羅馬人用數字所代表的月份
RR	二數字所帶表的年
SS	一分鐘的秒數
SSSSS	自從午夜以後的秒數
SYYYY	符號化年份，如果西元前 500，那麼爲 -500
W	一個月內幾個星期
WW	一年內幾個星期
Y	年的最後一位數字
YY	年的最後二位數字
YYY	年的最後三位數字
YYYY	年的四位數字

Oracle date pictures:	
YEAR	拼出大寫年份，如十九九七 NINETEEN-NINETY-SEVEN
Year	拼出年份，如十九九七 Nineteen-Ninety-Seven
year	拼出小寫年份，如十九九七 nineteen-ninety-seven

12-3-2 轉換日期成字串

特別地，DATETIME 值轉換到字串，是用來改變來查詢所輸出的外表，應用轉換函數用來達成這需求，下列二個查詢例子，轉換指定日期和時間成爲想要的字串：

TYPE
```
SELECT DATE_HIRE = DATENAME(MONTH, DATE_HIRE)
FROM EMPLOYEE_PAY_TBL
```

Output
```
DATE_HIRE
-----------------
May
June
August
June
July
Jan

6 rows affected.
```

Oracle 使用 TO_CHAR 函數轉換日期：

TYPE
```
SELECT DATE_HIRE, TO_CHAR(DATE_HIRE,'Month dd,yyyy')
FROM EMPLOYEE_PAY_TBL;
```

Output
```
DATE_HIRE HIRE
--------- ----------------------------  --------------------
23-MAY-89 May        23, 1989
17-JUN-90 June       17, 1990
14-AUG-94 August     14, 1994
28-JUN-97 June       28, 1997
```

```
22-JUL-96 July       22，1996
14-JAN-91 January    14，1991

6 rows selected.
```

12-3-3 字串轉換為日期

下列例子舉例說明，廠商轉換字串進入日期格式的方法，當完全轉換的時候，資料能儲存在特別欄，定義成 DATETIME 資料格式。

TYPE

```
SELECT TO_DATE('JANUARY 01 1998','MONTH DD YYYY')
FROM EMPLOYEE_PAY_TBL;
```

Output

```
TO_DATE('
----------------
01-JAN-98
01-JAN-98
01-JAN-98
01-JAN-98
01-JAN-98
01-JAN-98

6 rows selected.
```

您可能是覺得奇怪，當只有提供一個日期值的時候，爲什麼會選擇這六個排，因爲文字從 EMPLOYEE_PAY_TBL 選擇轉換，而 EMPLOYEE_PAY_TBL 有六排的資料，因此，在表格裡文字的轉換，每筆記錄都會選擇。

12

12-4 摘要

基於 ANSI 標準所提供的事實，您已經瞭解 DATETIME 值的意義，然而，如同與許多 SQL 元件，大多數的廠商已經脫離標準的 SQL 函數命令和語法，但是在處理日期與時間的觀念上，總是保持相同。上個小時，您看見函數如何依每家廠商上的不同而異，而這個小時，您已經看到在日期和時間的一些不同資料格式，函數和運算子。請記住

不是所有在這章討論的例子，都能在您的廠商上運作，但是日期和時間觀念是相同的，並且應該可以應用在任何的廠商上。

12-4-1 Q&A

Q 為什麼廠商選擇脫離資料格式，組合單一標準的函數？

A 廠商資料格式和函數表示方法不一樣，主要地因爲每個廠商在內部所儲存的資料能提供最有效率取回資料的方法，然而，基於 ANSI 所規定，所有的廠商應該提供相同的儲存方法給日期與時間的必需元件，例如那年、月、分鐘、小時、日、秒等等。

Q 如果我要以，同於我的廠商儲存日期資料，[illegible]

A 如果您選擇日期定義欄爲可變長度字串，日期幾乎能被儲存在任何類型格式。請記得，主要比較彼此日期事物的時候，它通常被需要先轉換字串到您的廠商所接受的 DATETIME 格式與表示方法；如果有適當的轉換函數可以使用。

12-5 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

12-5-1 隨堂測驗

1. 通常系統日期與時間在哪裡可以得到？
2. 列出 DATETIME 內在元件標準值。
3. 如果您的公司是國際組織，關於日期與時間的表示方法和比較，什麼可能是主要的因素？
4. 在有效的 DATETIME 資料格式，字串日期值能與日期值做比較嗎？

12-5-2 練習題

1. 藉由下列提供的各項訊息練習編寫 SQL 碼：

   使用 SYSDATE 表是現在的日期而且計時。

   使用表格稱為日期。

   使用 TO_CHAR 函數轉換日期成字串與語法：

   TO_CHAR(EXPRESSION,DATE_PICTURE)

   使用 TO_DATE 函數轉換字串為日期，與語法：

   TO_DATE(EXPRESSION,DATE_PICTURE)

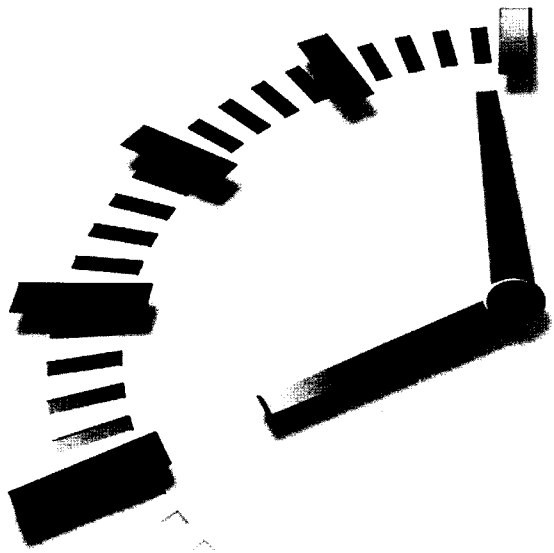
   日期圖片資訊：

日期圖片	意義
MONTH	拼出大寫月份
DAY	拼出大寫日期
DD	月內的天數
MM	年內的天數
YY	年的最後二位數字
YYYY	年的四位數字
MI	每小時的分鐘
SS	一分鐘的秒數

1. 假設今天是 1997-12-31，轉換現在日期到格式 December 31 1997。
2. 轉換下列各項字串到日期格式：

   'DECEMBER 31 1997'

3. 編寫碼傳回 1998 新年前夕的那週，哪的天在那天之上。假定那個日期以 31-DEC-97 的格式被儲存是有效的 DATETIME 資料格式。



第 IV 單元

建立複雜的資料庫查詢

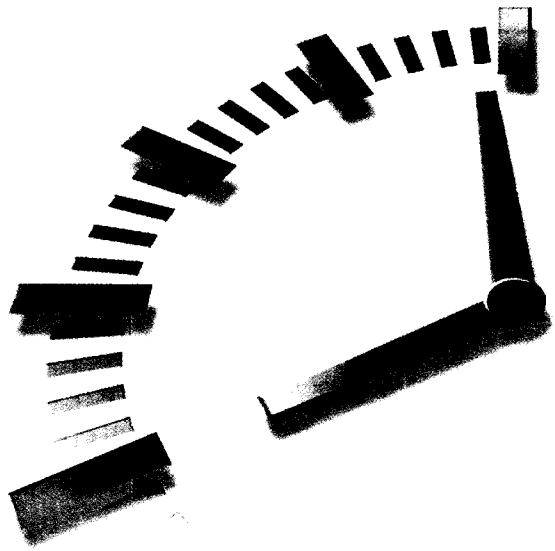
第 13 小時　在查詢裡加入表格

第 14 小時　使用子查詢定義未知的資料

第 15 小時　組合多重查詢

第VI單元

獨立樣本的資料檢定



第 13 小時

在查詢裡加入表格

到這裡爲止，所有關資料庫的查詢，您都在單一表格的資料庫裡執行，在這個小時裡，您將學習該如何在查詢裡加入表格，所以資料可從多個表格被取回。

本小時的重點包括：

- 介紹加入表格
- 加入不同的類型
- 如何與何時使用加入
- 加入表格的實際例子
- 不適當加入表格的影響
- 在查詢裡使用別名的表格

13-1 選擇來自多重表格的資料

可以選擇來自多重表格的資料，是 SQL 最重要的功能之一，沒有這個功能，關連式資料庫的觀念將無法實行，在單一表格查詢，有時是含有相當有限的資訊，但是在真實的世界，最實際的查詢是從資料庫的多重表格取得想要的資料。

在如您在正常化的那個小時所見， 關連式資料庫打碎成爲單純較小的、較容易辨辨識的表格內，而且全部較容易管理，當表格區分爲較小的表格時候，有關係的表格，通常建立一個主索引，這些索引用來建立彼此有關係的表格。

NEW TERM 加入（join）二個或較多的表格是爲了要取回來自多重表格的資料。

您可能會問，到底爲什麼要正常化表格？爲了要傳回您要的資料，最後又再組回那些表格。您很少地選擇表格裡所有的資料，所以選取依照每個人需要做選擇查詢是比較好的方式，雖然在正常化資料庫時，執行效率可能遭受些微地影響，但在全部的編碼和維護確實是比較簡單得多。

13-2 加入類型

當然不同廠商有許多加入表格的方式，在這您將專注於最常用的加入方式，您將學習加入的類型是：

EQUIJOINS 同等加入
NATURAL JOINS 自然加入
NON-EQUIJOINS 非同等加入
OUTER JOINS 外部加入
SELF JOINS 自我加入

13-2-1 元件位置的加入條件

從上個小時，您已經知道 SELECT 與 FROM 子句是主要 SQL 指令元件；當加入表格的時候，WHERE 子句是 SQL 指令的一個必需元件，要加入的表格列在 FROM 子句，加入的表格放在 WHERE 子句，幾個運算子能用來加入表格，例如 =、<、>、<>、<=、>=、!=、between、like 和 not；他們全部能用來加入表格，然而，最常用的運算子是相等符號 =。

13-2-2 Joins of Equality 同等加入

最常使用和最重要的加入方式是同等加入 EQUIJOIN，也是稱爲內部加入，同等加入 EQUIJOIN，加入二個表格常用的欄，通常爲主索引，EQUIJOIN 的語法是：

```
SELECT TABLE1.COLUMN1, TABLE2.COLUMN2
FROM TABLE1, TABLE2 [, TABLE3 ]
WHERE TABLE1.COLUMN_NAME = TABLE2.COLUMN_NAME
[ AND TABLE1.COLUMN_NAME = TABLE3.COLUMN_NAME ]
```

註解

請看 SQL 指令所舉的例子，SQL 指令所用的空格是用做增加可讀性，空格不是必須的，但是建議使用。

Usage

```
SELECT EMPLOYEE_TBL.EMP_ID,
       EMPLOYEE_PAY_TBL.DATE_HIRE
FROM EMPLOYEE_TBL,
     EMPLOYEE_PAY_TBL
WHERE EMPLOYEE_TBL.EMP_ID = EMPLOYEE_PAY_TBL.EMP_ID;
```

ANALYSIS 指令傳回職員編號和職員開始上班日期，職員編號從 EMPLOYEE_TBL 選擇（雖然它存在在兩個表格，但是您必須指定一個表格），當開始上班日期從 EMPLOYEE_PAY_TBL 選擇，因爲職員編號在兩個表格都存在，所以表格裡兩個欄都必須進行調整，藉由表格名稱調整那些欄，您告訴資料庫該在哪裡可以得到資料。

下列例子裡的資料，自 EMPLOYEE_TBL 和 EMPLOYEE_PAY_TBL 表格選擇出來，因爲需要資料存在每個表格裡，所以需要使用同等加入。

TYPE
```
SELECT EMPLOYEE_TBL.EMP_ID, EMPLOYEE_TBL.LAST_NAME,
       EMPLOYEE_PAY_TBL.POSITION
FROM EMPLOYEE_TBL, EMPLOYEE_PAY_TBL
WHERE EMPLOYEE_TBL.EMP_ID = EMPLOYEE_PAY_TBL.EMP_ID;
```

Output
```
EMP_ID    LAST_NAM POSITION
--------- -------- ---------
311549902 STEPHENS MARKETING
442346889 PLEW     TEAM LEADER
213764555 GLASS    SALES MANAGER
313782439 GLASS    SALESMAN
220984332 WALLACE  SHIPPER
443679012 SPURGEON SHIPPER

6 rows selected.
```

請注意在 SELECT 子句裡，爲了要識別每個欄，必須聯合表格名稱，這樣取得查詢資格的欄，藉著查詢，讓存在於超過一個表格的欄成爲資格欄，您通常必須讓所有欄取得資格，避免在除錯或修改 SQL 編碼的時，產生任何問題。

13-2-3 自然加入

自然加入（NATURAL JOIN）幾乎相同於 EQUIJOIN；然而，不同於 EQUIJOIN 的，自然加入除去，由加入的時所產生複製的欄，加入（JOIN）條件是相同的，但是所選擇的欄不一致。

語法是依下列各項：

```
SELECT TABLE1.*, TABLE2.COLUMN_NAME
       [ TABLE3.COLUMN_NAME ]
FROM TABLE1, TABLE2 [ TABLE3 ]
WHERE TABLE1.COLUMN_NAME = TABLE2.COLUMN_NAME
[ AND TABLE1.COLUMN_NAME = TABLE3.COLUMN ]
```

Usage

```
SELECT E.*, EP.SALARY
FROM EMPLOYEE_TBL E,
     EMPLOYEE_PAY_TBL EP
WHERE E.EMP_ID = EP.EMP_ID
```

之前的 SQL 指令傳回來自 EMPLOYEE_TBL 表格的所有欄與自 EMPLOYEE_PAY_TBL 的薪水資料，EMP_ID 是在兩個表格，但是只有從 EMPLOYEE_TBL 取回，因爲兩者包含相同的訊息，所以不需要選擇。

下列各例子選擇來自 EMPLOYEE_TBL 表格所有欄和只選擇來自 EMPLOYEE_PAY_TBL 表格的一個欄，請記得星號（*）表示表格的所有欄。

TYPE

```
SELECT EMPLOYEE_TBL.*, EMPLOYEE_PAY_TBL.POSITION
FROM EMPLOYEE_TBL, EMPLOYEE_PAY_TBL
WHERE EMPLOYEE_TBL.EMP_ID = EMPLOYEE_PAY_TBL.EMP_ID;
```

Output

```
EMP ID    LAST NAM FIRST NA M ADDRESS       CITY         ST ZIP    PHONE
--------- -------- -------- - -------------- ------------ -- ----- ----------
PAGER      POSITION
---------- --------------
311549902 STEPHENS TINA     D RR 3 BOX 17A  GREENWOOD    IN 47890 3178784465
           MARKETING

442346889 PLEW     LINDA    C 3301 BEACON   INDIANAPOLIS IN 46224 3172978990
           TEAM LEADER

213764555 GLASS    BRANDON  S 1710 MAIN ST  WHITELAND    IN 47885 3178984321
3175709980 SALES MANAGER
```

```
313782439 GLASS    JACOB    3789 RIVER BLVD INDIANAPOLIS IN 45734 3175457676
8887345678 SALESMAN

220984332 WALLACE  MARIAH   7889 KEYSTONE   INDIANAPOLIS IN 46741 3173325986
          SHIPPER

443679012 SPURGEON TIFFANY  5 GEORGE COURT  INDIANAPOLIS IN 46234 3175679007
          SHIPPER

6 rows selected.
```

13-2-4 使用表格別名

使用表格別名，表示使用特別的 SQL 指令重新命名表格，重新命名是暫時的變化，真實的表格名稱在資料庫中不會被改變，正如您所學習自己加入（SELF JOIN），給表格別名是必須的，因爲給表格別名時常用來減少使用按鍵，使 SQL 指令變得比較短和容易讀，除此之外，使用較少的按鍵表示，犯較少的錯誤。給表格別名也表示，選擇的欄必須與表格一起取得資格別名。在這裡有一些表格別名的例子和對應的欄：

Output

```
SELECT E.EMP_ID, EP.SALARY, EP.DATE_HIRE, E.LAST_NAME
FROM EMPLOYEE_TBL E,
     EMPLOYEE_PAY_TBL EP
WHERE E.EMP_ID = EP.EMP_ID
AND EP.SALARY > 20000
```

ANALYSIS

先前的 SQL 指令，表格已經給予別名，EMPLOYEE_TBL 已經重新命名 E. EMPLOYEE_PAY_TBL已經重新命名 EP。選擇該重新命名什麼表格是任意。EMPLOYEE_TBL 開始於字母 E，所以字母 E 選擇，EMPLOYEE_PAY_TBL 開始也是字母 E，所以您不能再一次使用 E，替代的方式爲，名字（PAY）裡第二個字的第一個字母和第一個字母（E）當作別名使用，選擇的欄會與對應表格別名互相證明，請注意，SALARY 被用在 WHERE 子句和必須與表格別名互相證明。

13-2-5 非同等加入

非同等加入（NON-EQUIJOIN）是基於二個或較多表格，敘述的欄數值不等於在另外的一個表格所敘述的欄數值，非同等加入的語法是：

```
FROM TABLE1, TABLE2 [, TABLE3 ]
WHERE TABLE1.COLUMN_NAME != TABLE2.COLUMN_NAME
[ AND TABLE1.COLUMN_NAME != TABLE2.COLUMN_NAME ]
```

Output

```
SELECT EMPLOYEE_TBL.EMP_ID, EMPLOYEE_PAY_TBL.DATE_HIRE
FROM EMPLOYEE_TBL,
     EMPLOYEE_PAY_TBL
WHERE EMPLOYEE_TBL.EMP_ID != EMPLOYEE_PAY_TBL.EMP_ID
```

ANALYSIS 之前的 SQL 指令沒有表格對應記錄，所以只傳回職員編號和開始上班日，下列是非同等加入的例子：

TYPE

```
SELECT E.EMP_ID, E.LAST_NAME, P.POSITION
FROM EMPLOYEE_TBL E,
     EMPLOYEE_PAY_TBL P
WHERE E.EMP_ID <> P.EMP_ID;
```

Output

```
EMP_ID    LAST_NAM POSITION
--------- -------- --------------
442346889 PLEW     MARKETING
213764555 GLASS    MARKETING
313782439 GLASS    MARKETING
220984332 WALLACE  MARKETING
443679012 SPURGEON MARKETING
311549902 STEPHENS TEAM LEADER
213764555 GLASS    TEAM LEADER
313782439 GLASS    TEAM LEADER
220984332 WALLACE  TEAM LEADER
443679012 SPURGEON TEAM LEADER
311549902 STEPHENS SALES MANAGER
442346889 PLEW     SALES MANAGER
313782439 GLASS    SALES MANAGER
220984332 WALLACE  SALES MANAGER
443679012 SPURGEON SALES MANAGER
311549902 STEPHENS SALESMAN
442346889 PLEW     SALESMAN
```

```
213764555 GLASS     SALESMAN
220984332 WALLACE   SALESMAN
443679012 SPURGEON  SALESMAN
311549902 STEPHENS  SHIPPER
442346889 PLEW      SHIPPER
213764555 GLASS     SHIPPER
313782439 GLASS     SHIPPER
443679012 SPURGEON  SHIPPER
311549902 STEPHENS  SHIPPER
442346889 PLEW      SHIPPER
213764555 GLASS     SHIPPER
313782439 GLASS     SHIPPER
220984332 WALLACE   SHIPPER

30 rows selected.
```

您可能覺得好奇，爲什麼當表格只存在 6 個列的時候，確傳回 30 個列，在 EMPLOYEE_TBL 的每一筆記錄，都有 EMPLOYEE_PAY_TBL 裡的記錄對應，因爲非同等測試了二個加入的表格，除了與自己對應列以外，在每個列第一個表格與來自第二個表格的所有列配成一對；這表示，每六個列與第二個表格的五個不相關列配成一對；六乘以五等於 30 列。

在早先的測試同等的例子中，第一個表格裡的每六個列只和第二個表格唯一的列配成一對（每個列對應的列）；六個列乘以一總數爲六列。


注意

當使用非同等加入時，您可能收到幾個列的資料是對您沒用的，小心地檢查您的結果。

13-2-6 外部加入

外部加入（OUTER JOIN）用來傳回存在表格的所有列，即使對應的列不存在所加入的表格，符號（+）用來指示在查詢使用外部加入，符號（+）放置在 WHERE 子句的表格名稱後，表格有符號（+）應該是沒有對應列的表格，在許多廠商，外部加入被分成左外部加入（LEFT

OUTER JOIN）、右邊外部（RIGHT OUTER JOIN）加入和全外部加入（FULL OUTER JOIN），外部加入在這些廠商正常地可選擇。


註解

您必須檢查您的廠商有關外部加入正確的用法和語法，符號（+）被一些主要的廠商所使用，但是非標準化。

一般的語法是：

```
FROM TABLE1
{RIGHT | LEFT | FULL} [OUTER] JOIN
ON TABLE2
```

Oracle 的語法是：

```
FROM TABLE1, TABLE2 [, TABLE3 ]
WHERE TABLE1.COLUMN_NAME[(+)] = TABLE2.COLUMN_NAME[(+)]
[ AND TABLE1.COLUMN_NAME[(+)] = TABLE3.COLUMN_NAME[(+)]
```


註解

外部加入只能用在加入條件的一邊；然而，在相同表格中，您能使用超過一個外部加入在一個欄的加入條件中。

外部加入的觀念在下二個例子中解釋；在第一個例子，選擇產品描述和所定的量； 兩數值從二個分開的表格中摘錄，請注意一個重要的因數，不是爲每種產品在 ORDERS_TBL 表格裡都有對應記錄，一般的同等加入執行：

TYPE

```
SELECT P.PROD_DESC, O.QTY
FROM PRODUCTS_TBL P,
     ORDERS_TBL O
WHERE P.PROD_ID = O.PROD_ID;
```

Output

```
PROD_DESC                      QTY
-----------------------  ----------
WITCHES COSTUME                 1
PLASTIC PUMPKIN 18 INCH        25
PLASTIC PUMPKIN 18 INCH         2
LIGHTED LANTERNS               10
FALSE PARAFFIN TEETH           20
```

```
5 rows selected.
```

只選擇 5 個列，但是有 10 種不同的產品，您想要顯示所有的產品，不論那些產品已經下了訂單。

下一個例子藉著使用外部加入完成需要的輸出結果，Oracles 語法用來做外部加入。

TYPE

```
SELECT P.PROD_DESC, O.QTY
FROM PRODUCTS_TBL P,
     ORDERS_TBL O
WHERE P.PROD_ID = O.PROD_ID(+);
```

Output

```
PROD_DESC                      QTY
------------------------- -----------
WITCHES COSTUME                  1
ASSORTED MASKS
FALSE PARAFFIN TEETH            20
ASSORTED COSTUMES
PLASTIC PUMPKIN 18 INCH         25
PLASTIC PUMPKIN 18 INCH          2
PUMPKIN CANDY
PLASTIC SPIDERS
CANDY CORN
LIGHTED LANTERNS                10

10 rows selected.
```

查詢傳回所有的產品，即使他們可能沒有任何的訂單，外部加入是包含 PRODUCTS_TBL 表格裡所有列的資料在內，不論對應的列在 ORDERS_TBL 表格是否存在。

13-2-7 自我加入

自我加入（SELF JOIN）用本身來加入表格，在 SQL 指令裡，好像暫時更名的表格是代表二個表格。

Output

```
SELECT A.COLUMN_NAME, B.COLUMN_NAME, [ C.COLUMN_NAME ]
FROM TABLE1 A, TABLE2 B [, TABLE3 C ]
WHERE A.COLUMN_NAME = B.COLUMN_NAME
[ AND A.COLUMN_NAME = C.COLUMN_NAME ]
SELECT A.LAST_NAME, B.LAST_NAME, A.FIRST_NAME
FROM EMPLOYEE_TBL A,
     EMPLOYEE_TBL B
WHERE A.LAST_NAME = B.LAST_NAME
```

ANALYSIS 之前的 SQL 指令傳回來在 EMPLOYEE_TBL 有相同名的所有職員。

13-3 加入的考量

幾件事物應該在使用加入之前考慮，一些考量包括應加入什麼欄，如果沒有常用的欄可以加入，執行效率會被質疑，執行效率在第18小時被討論“管理資料庫使用者”。

13-3-1 使用基礎表格

您是否需要取回，來自二個沒有常用欄的表格資料，您必須使用另一個表格或兩個表格有常用欄來加入，那個表格變成基礎表格(BASE TABLE)，基礎表格用來加入一個或多個有常用欄的表格或加入沒有常用欄的表格。使用下列三個表格作爲基礎表格的例子：

CUSTOMER_TBL

cust_id	varchar2(10)	not null	primary key
cust_name	varchar2(30)	not null	
cust_address	varchar2(20)	not null	
cust_city	varchar2(15)	not null	
cust_state	char(2)	not null	
cust_zip	number(5)	not null	
cust_phone	number(10)		
cust_fax	number(10)		

ORDERS_TBL

ord_num	varchar2(10)	not null	primary key
cust_id	varchar2(10)	not null	
prod_id	varchar2(10)	not null	
qty	number(6)	not null	
ord_date	date		

PRODUCTS_TBL

prod_id	varchar2(10)	not null	primary key
prod_desc	varchar2(40)	not null	
cost	number(6,2)	not null	

您需要使用 CUSTOMERS_TBL 和 PRODUCTS_TBL，因爲沒有常用欄可以加入表格，現在看 ORDERS_TBL，ORDERS_TBL 與 CUSTOMERS_TBL 都有 cust_id 加入，CUSTOMERS_TBL 也有 CUST_ID，PRODUCTS_TBL 有 PROD_ID，PROD_ID 也在 ORDERS_TBL。加入條件與產生的結果：

TYPE

```
SELECT C.CUST_NAME, P.PROD_DESC
FROM CUSTOMER_TBL C,
     PRODUCTS_TBL P,
     ORDERS_TBL O
WHERE C.CUST_ID = O.CUST_ID
  AND P.PROD_ID = O.PROD_ID;
```

Output

```
CUST_NAME                      PROD_DESC
------------------------------ ------------------------
LESLIE GLEASON                 WITCHES COSTUME
SCHYLERS NOVELTIES             PLASTIC PUMPKIN 18 INCH
WENDY WOLF                     PLASTIC PUMPKIN 18 INCH
GAVINS PLACE                   LIGHTED LANTERNS
SCOTTYS MARKET                 FALSE PARAFFIN TEETH

5 rows selected.
```

註解

請注意在 WHERE 子句所使用表格別名與欄。

13-3-2 笛卡爾產品

笛卡爾產品（Cartesian Product）是加入或不加入的結果，如果您從二個或較多表格選擇不加入那些表格，您的輸出結果可能是選擇來自所有表格的所有列，如果您的表格是很大的，結果可能是數十萬計的列或百萬計的資料列，在 SQL 指令從二個或較多的表格取回資料，WHERE 子句是大大地被推薦，笛卡爾產品也被稱爲是交叉加入。語法是：

```
FROM TABLE1, TABLE2 [, TABLE3 ]
WHERE TABLE1, TABLE2 [, TABLE3 ]
```

交叉加入的例子或恐懼笛卡爾產品，被顯示在這裡：

TYPE

```
SELECT E.EMP_ID, E.LAST_NAME, P.POSITION
FROM EMPLOYEE_TBL E,
     EMPLOYEE_PAY_TBL P;
```

Output

```
EMP_ID    LAST_NAM POSITION
--------- -------- --------------
311549902 STEPHENS MARKETING
442346889 PLEW     MARKETING
213764555 GLASS    MARKETING
313782439 GLASS    MARKETING
220984332 WALLACE  MARKETING
443679012 SPURGEON MARKETING
311549902 STEPHENS TEAM LEADER
442346889 PLEW     TEAM LEADER
213764555 GLASS    TEAM LEADER
313782439 GLASS    TEAM LEADER
220984332 WALLACE  TEAM LEADER
443679012 SPURGEON TEAM LEADER
311549902 STEPHENS SALES MANAGER
442346889 PLEW     SALES MANAGER
213764555 GLASS    SALES MANAGER
313782439 GLASS    SALES MANAGER
220984332 WALLACE  SALES MANAGER
443679012 SPURGEON SALES MANAGER
311549902 STEPHENS SALESMAN
442346889 PLEW     SALESMAN
213764555 GLASS    SALESMAN
313782439 GLASS    SALESMAN
```

```
220984332 WALLACE   SALESMAN
443679012 SPURGEON  SALESMAN
311549902 STEPHENS  SHIPPER
442346889 PLEW      SHIPPER
213764555 GLASS     SHIPPER
313782439 GLASS     SHIPPER
220984332 WALLACE   SHIPPER
443679012 SPURGEON  SHIPPER
311549902 STEPHENS  SHIPPER
442346889 PLEW      SHIPPER
213764555 GLASS     SHIPPER
313782439 GLASS     SHIPPER
220984332 WALLACE   SHIPPER
443679012 SPURGEON  SHIPPER

36 rows selected.
```

資料從二個分開表格被選擇，然而沒有執行加入運算，因爲您還沒有敘述該如何加入第一個表格裡的列到第二個表格裡列，資料庫用第二個表格裡每個列與第一個表格裡的每個列來配對，因爲每個表格有 6 列的資料，所以選擇 36 個產品，是從 6 列乘以個 6 列所達成的。

請小心地在查詢裡加入所有表格，如果在查詢二個表格不是已加入的和每個表格包含 1,000 列的資料，笛卡爾產品由 1,000 列乘以 1,000 列所組成，傳回總數為 1,000,000 列的資料。

13-4 摘要

您已經介紹過 SQL 表格中最著名的特徵-加入，假想如果您不能夠從單一查詢裡，摘錄一或多個表格的資料是多不方便的事。在這顯示幾種加入的類型，每個依它的自己的特性，放置在查詢條件之中，加入是連接來自同等和非同等表格的資料。外部加入是非常有用的，允許從一個表格取回的資料，即使相關的資料不在加入的表格中發現。自我加入對用來加入表格本身。交叉加入，又以笛卡爾產品（Cartesian Product）聞名，笛卡爾產品是多重表格之間沒有加入的結果，時常產生不必要的輸出結果，當選擇資料從超過一個表格的時

候，一定要適當地依照有關係的欄（通常是主索引）加入那些表格，失敗地適當加入表格會夠造成不完全或錯誤的輸出。

13-4-1 Q&A

Q 當加入表格，他們出現的順序必須與 FROM 子句相同嗎？

A 不，他們出現的順序不必相同；然而，執行效率可能因表格在 FROM 子句的順序和表格被加入順序而定。

Q 當使用基礎表格加入不關連表格的時候，我必須選擇來自基礎表格的任何欄嗎？

A 不，使用基礎表格加入不關連表格不需要選擇來自基礎表格的欄。

Q 在表格之間，我能加入超過一個欄嗎？

A 是的，一些查詢可能需要您加入超過一個欄在每個表格中，來提供那些被加入的表格裡，在列之間資料的完全關係。

13-5 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

13-5-1 隨堂測驗

1. 什麼類型的加入，您將會使用傳回來自一個表格的記錄，不管有聯合關係的記錄存在的表格裡？
2. 加入條件位於 SQL 指令的什麼部份？
3. 什麼類型的加入，您會使用評估有關係表格列之中的同等性？

4. 如果您從二個不同的表格選擇，但是那些表格加入失敗，會發生什麼？

5. 使用那些下列各項表格：

ORDERS_TBL

ord_num	varchar2(10)	not null	primary key
cust_id	varchar2(10)	not null	
prod_id	varchar2(10)	not null	
qty	number(6)	not null	
ord_date	date		

PRODUCTS_TBL

prod_id	varchar2(10)	not null	primary key
prod_desc	varchar2(40)	not null	
cost	number(6,2)	not null	

請檢查使用外部加入的語法？

```
SELECT C.CUST_ID, C.CUST_NAME, O.ORD_NUM
FROM CUSTOMER_TBL C, ORDERS_TBL O
WHERE C.CUST_ID(+) = O.CUST_ID(+)
```

13-5-2 練習題

使用下列表格來做練習：

EMPLOYEE_TBL

emp_id	varchar2(9)	not null	PRIMARY KEY
last_name	varchar2(15)	not null	
first_name	varchar2(15)	not null	
middle_name	varchar2(15)		
address	varchar2(30)	not null	
city	varchar2(15)	not null	

EMPLOYEE_TBL

state	char(2)	not null	
zip	number(5)	not null	
phone	char(10)		
pager	char(10)		

EMPLOYEE_PAY_TBL

emp_id	varchar2(9)	not null	primary key
position	varchar2(15)	not null	
date_hire	date		
pay_rate	number(4,2)	not null	
date_last_raise	date		
salary	number(6,2)		
bonus	number(4,2)		
constraint emp_fk foreign key (emp_id) references employee_tbl (emp_id)			

CUSTOMER_TBL

cust_id	varchar2(10)	not null	primary key
cust_name	varchar2(30)	not null	
cust_address	varchar2(20)	not null	
cust_city	varchar2(15)	not null	
cust_state	char(2)	not null	
cust_zip	number(5)	not null	
cust_phone	number(10)		
cust_fax	number(10)		

ORDERS_TBL

ord_num	varchar2(10)	not null	primary key
cust_id	varchar2(10)	not null	

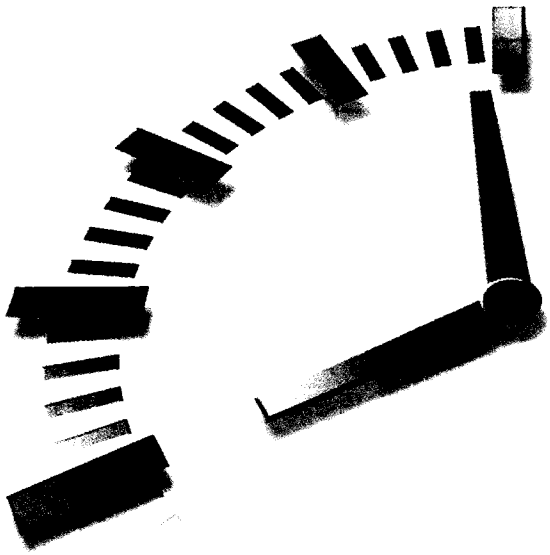
ORDERS_TBL

prod_id	varchar2(10)	not null	
qty	number(6)	not null	
ord_date	date		

PRODUCTS_TBL

prod_id	varchar2(10)	not null	primary key
prod_desc	varchar2(40)	not null	
cost	number(6,2)	not null	

1. 寫 SQL 指令來傳回來自 EMPLOYEE_TBL 的 emp_id，last_name 和 first_name 與來自 EMPLOYEE_PAY_TBL 的 BOUNS 和 salary。
2. 選擇來自 CUSTOMERS_TBL 的 CUST_ID、CUST_NAME，選擇從 PRODUCTS_TBL 的 PROD_ID、COST，選擇從 ORDERS_TBL 的 ORD_NUM 和 QTY，使用一個 SQL 指令加入所有三個表格。



第 14 小時

使用子查詢定義未知的資料

在這個小時，將顯示子查詢（Subquery）的觀念，使得資料庫查詢傳回的結果能更有效。

本小時的重點包含：

- 什麼是子查詢？
- 為何使用子查詢
- 一般資料庫裡子查詢的例子
- 和資料操作指令一起使用子查詢
- 箝入子查詢

14-1 子查詢是什麼？

子查詢是查詢裡面的另一個查詢，這也是著名的築巢查詢，子查詢用在主查詢，再進一步對傳回的資料做限制，而得到想要的資料，子查詢與 SELECT、INSERT、UPDATE 與 DELETE 一起使用。

NEW TERM 子查詢(**subquery**) 是箝入在 WHERE 的子句所做的另一個查詢，用在主查詢，再進一步對傳回的資料做限制。

在某些情況下，基於表格之間一個或較多的條件，子查詢能間接地參加運算與資料聯接，當子查詢用在查詢的時候，子查詢是首先處理，然後主查詢依照子查詢的條件做其他處理，子查詢的結果用來處理在主查詢的 WEHRE 運算句，子查詢能用在主查詢的 **WHERE** 子句或 HAVING 子句，邏輯和關係運算子例如 =、>、<、<>、IN、NOT IN、AND、OR 等等，能在子查詢裡面用在 WHERE 與 HAVING 子句評估子查詢。


註解

適用於查詢的標準相同也適用於子查詢，加入運算、函數、轉換和其他選擇項都可能在子查詢裡面使用。

子查詢有些規則必須遵從：

- 子查詢必須定義在刮弧號裡面。
- 在 SELECT 子句中子查詢可能只有一個欄，除非是在主查詢有多重欄與子查詢所選擇欄做比較。
- ORDER BY 不能夠在子查詢使用，雖然主查詢能使用 ORDER BY，但在子查詢 GROUP BY 被用來執行與 ORDER BY 相同的函數。
- 子查詢與多重數值運算子一起使用傳回更多比較列，例如 IN 運算子。
- BETWEEN 運算子不能夠被子查詢使用；然而 BETWEEN 能在子查詢裡面使用。

子查詢的基本語法：

```
SELECT COLUMN_NAME
FROM TABLE
WHERE COLUMN_NAME = (SELECT COLUMN_NAME
                     FROM TABLE
                     WHERE CONDITIONS);
```

下列各例子表示 BETWEEN 運算子可能和不可能被子查詢使用：

在子查詢正確使用 BETWEEN：

```
SELECT COLUMN_NAME
FROM TABLE
WHERE COLUMN_NAME OPERATOR (SELECT COLUMN_NAME
                            FROM TABLE)
                            WHERE VALUE BETWEEN VALUE)
```

在子查詢不正確使用 BETWEEN：

```
SELECT COLUMN_NAME
FROM TABLE
WHERE COLUMN_NAME BETWEEN VALUE AND (SELECT COLUMN_NAME
                                     FROM TABLE)
```

14-1-1 子查詢與 SELECT 指令

子查詢最時常與 SELECT 指令一起使用，雖然他們可能在處理資料指令裡面使用，子查詢與 SELECT 指令一起使用時，取回主查詢所要的資料，以解決主要的查詢。

基本語法如下：

```
SELECT COLUMN_NAME [, COLUMN_NAME ]
FROM TABLE1 [, TABLE2 ]
WHERE COLUMN_NAME OPERATOR
                  (SELECT COLUMN_NAME [, COLUMN_NAME ]
                   FROM TABLE1 [, TABLE2 ]
                  [ WHERE ]
SELECT E.EMP_ID, E._LAST_NAME, E.FIRST_NAME, EP.PAY_RATE
FROM EMPLOYEE_TBL E, EMPLOYEE_PAY_TBL EP
WHERE E.EMP_ID = EP.EMP_ID
AND EP.PAY_RATE > (SELECT PAY_RATE
                   FROM EMPLOYEE_PAY_TBL
```

```
WHERE EMP_ID = '313782439')
```

ANALYSIS 先前的 SQL 指令傳回所有員工中，員工編號大於 313782439 的員工編號、姓、名和 pay rate，在這情況，您不一定得知道這位特定員工正確的 pay rate；您只需擔心爲了在子查詢得到一連串員工的 pay rate。

第一個查詢選擇薪資爲特定員工的 pay rate，在查詢下一個例子時，當作子查詢使用。

TYPE
```
SELECT PAY_RATE
FROM EMPLOYEE_PAY_TBL
WHERE EMP_ID = '313782439';
```

Output
```
  PAY_RATE
----------
      12.8

1 row selected.
```

下列 WHERE 子句，先前的查詢當做子查詢使用。

Output
```
SELECT E.EMP_ID, E.LAST_NAME, E.FIRST_NAME, EP.PAY_RATE
FROM EMPLOYEE_TBL E, EMPLOYEE_PAY_TBL EP
WHERE E.EMP_ID = EP.EMP_ID
  AND EP.PAY_RATE > (SELECT PAY_RATE
                     FROM EMPLOYEE_PAY_TBL
                     WHERE EMP_ID = '313782439');
```

Output
```
EMP_ID    LAST_NAM FIRST_NA   PAY_RATE
--------- -------- -------- ----------
311549902 STEPHENS TINA           17.5
442346889 PLEW     LINDA         14.75
213764555 GLASS    BRANDON       18.25
443679012 SPURGEON TIFFANY         15

4 rows selected.
```

子查詢的結果是 12.8（如最後一個例子所顯示），同樣地評估，最後一個在 WHERE 子句的條件：

```
AND EP.PAY_RATE > 12.8
```

當您執行查詢的時候，您不知道特定人的 pay rate 爲多少，然而，主查詢能夠對子查詢做比較每個人 pay rate。


註解

當那些精確，子查詢常用來放在查詢的條件是未知的時候，薪水為 313782439 是未知的，但是子查詢設計為您做跑腿的工作。

14-1-2 子查詢與 INSERT 指令

子查詢也可以連同資料處理語言（DML）指令一起使用，INSERT 指令是您檢查的第一個。INSERT 指令使用在從子查詢傳回另外的一個表格內的資料，子查詢裡選擇的資料能與任何字元、日期或函數一起修改。

基本語法如下：

TYPE
```
INSERT INTO TABLE_NAME [ (COLUMN1 [ COLUMN2 ]) ]
SELECT [ *|COLUMN1 [ COLUMN2 ])
FROM TABLE1 [, TABLE2 ]
[ WHERE VALUE OPERATOR ]
```

子查詢有 INSERT 指令的例子：

```
INSERT INTO RICH_EMPLOYEES
SELECT E.EMP_ID, E.LAST_NAME, E.FIRST_NAME, EP.PAY_RATE
FROM EMPLOYEE_TBL E, EMPLOYEE_PAY_TBL EP
WHERE E.EMP_ID = EP.EMP_ID
  AND EP.PAY_RATE > (SELECT PAY_RATE
                     FROM EMPLOYEE_PAY_TBL
                     WHERE EMP_ID = '313782439');
```

Output
```
4 rows created.
```

這個 INSERT 指令插入 EMP_ID、LAST_NAME、FIRST_NAME 和 PAY_RATE 到稱爲 RICH_EMPLOYEES 的表格之內，因爲有大於員工編號 313782439 的 pay rate 的員工。


註解

請記得去使用 COMMIT 和 ROLLBACK 指令當使用 DML 指令的時候，例如 INSERT 指令。

14-1-3 子查詢與 UPDATE 指令

子查詢能連同 UPDATE 指令一起使用，在子查詢和 UPDATE 指令一起使用的時候，表格裡的單一或多重欄可以更新。

基本語法如下：

```
UPDATE TABLE
SET [(}COLUMN_NAME [, COLUMN_NAME) ] =
    (SELECT [(]COLUMN_NAME [, COLUMN_NAME) ]
    FROM TABLE
    [ WHERE ]
```

例子表示在 UPDATE 指令之後使用查詢，第一個查詢傳回在 Indianapolis 住的所有員工的員工編號，您能看見有四個符合這標準的人。

TYPE
```
SELECT EMP_ID
FROM EMPLOYEE_TBL
WHERE CITY = 'INDIANAPOLIS';
```

Output
```
EMP_ID
--------
442346889
313782439
220984332
443679012

4 rows selected.
```

第一個查詢 UPDATE 指令後面接著子查詢，子查詢傳回在第一個查詢裡，有多少員工編號，現在使用 UPDATE 子查詢：

TYPE
```
UPDATE EMPLOYEE_PAY_TBL
SET PAY_RATE = PAY_RATE * 1.1
WHERE EMP_ID IN (SELECT EMP_ID
```

```
FROM EMPLOYEE_TBL
WHERE CITY = 'INDIANAPOLIS');
```

Output
```
4 rows updated.
```

如預期的，更新四個列，一件非常重要的事需要注意的是，不像第一個區段裡的例子，這子查詢傳回多重列資料，因爲您預期傳回多重列，您已經用 IN 運算子代替相等的符號，請記得 IN 在列表是用來比較運算句的值，如果使用相等的符號，將會傳回錯誤訊息。

當評估子查詢的時候，一定要使用正確的運算子，舉例來說，運算子用來比較運算句和一個數值，例如相等的符號，是不能夠用來評估傳回超過一個列的資料子查詢。

14-1-4 子查詢與 DELETE 指令

子查詢也可與 DELETE 指令一起使用。

基本語法如下：

```
DELETE FROM TABLE_NAME
[ WHERE OPERATOR [ VALUE ]
                 [ (SELECT COLUMN_NAME
                 FROM TABLE_NAME) ]
                 [ WHERE) ]
```

在這個例子，您從 EMPLOYEE_PAY_TBL 格表刪除 BRANDON GLASS 的記錄，您不知道什麼 **Brandons** 員工編號，但是您使用子查詢由 EMPLOYEE_TBL 表格得到他的編號，EMPLOYEE_TBL 表格包含 FIRST_NAME 和 LAST_NAME 欄。

Output
```
DELETE FROM EMPLOYEE_PAY_TBL
WHERE EMP_ID = (SELECT EMP_ID
                FROM EMPLOYEE_TBL
                WHERE LAST_NAME = 'GLASS'
                  AND FIRST_NAME = 'BRANDON');
```

Output
```
1 row deleted.
```


注意

請別忘記使用在 WHERE 子句使用 UPDAT 與 DELETE 指令，所有列會從目標表格更新或刪除，如果不使用 WHERE 子句。

14-2 在子查詢裡面箝入子查詢

子查詢可能正箝入在另一個子查詢裡面，正如您能在一般查詢裡面箝入子查詢，當使用子查詢的時候，子查詢在主要的查詢之前會處理，同樣地，最內的子查詢是首先箝入的或築巢、由子查詢一直處理到主查詢，完成工作。


註解

您必須檢查您的廠商在子查詢數量上的限制，如果任何的話，可能會使用單一指令，它在廠商之間可能不一致。

子查詢箝入基本語法如下：

```
SELECT COLUMN_NAME [, COLUMN_NAME ]
FROM TABLE1 [, TABLE2 ]
WHERE COLUMN_NAME OPERATOR (SELECT COLUMN_NAME
                            FROM TABLE
                            WHERE COLUMN_NAME OPERATOR
                                    (SELECT COLUMN_NAME
                                    FROM TABLE
                                    [ WHERE ]))
```

下列各例子使用二個子查詢，一個箝入另一個裡面，您要發現客戶已經下訂單，所訂的量乘以產品的單價是大於所有產品單價的總和。

TYPE
```
select cust_id, cust_name
from customer_tbl
where cust_id in (select o.cust_id
                  from orders_tbl o, products_tbl p
                  where o.prod_id = p.prod_id
                    and o.qty * p.cost > (select sum(cost)
                                          from products_tbl));
```

Output
```
CUST_ID    CUST_NAME
---------- ------------------------------
287        GAVINS PLACE
43         SCHYLERS NOVELTIES

2 rows selected.
```

二個列符合兩個子查詢的被選擇。

下列二個例子表示每一子查詢的結果，幫助您了解主查詢如何處理。

TYPE
```
select sum(cost) from products_tbl;
```

Output
```
 SUM(COST)
----------
     72.14

1 row selected.
```

TYPE
```
select o.cust_id
from orders_tbl o, products_tbl p
where o.prod_id = p.prod_id
  and o.qty * p.cost > 72.14;
```

Output
```
CUST_ID
----------
43
287

2 rows selected.
```

在基本方面，主查詢（在子查詢處理之後）如下列各例子所顯示，第二個子查詢的替換：

TYPE
```
select cust_id, cust_name
from customer_tbl
where cust_id in (select o.cust_id
                  from orders_tbl o, products_tbl p
                  where o.prod_id = p.prod_id
                    and o.qty * p.cost > 72.14;
```

第一個子查詢的替換：

TYPE
```
select cust_id, cust_name
from customer_tbl
where cust_id in (287,43);
```

最後的結果：

Output
```
CUST_ID    CUST_NAME
---------- ------------------------------
287        GAVINS PLACE
43         SCHYLERS NOVELTIES

2 rows selected.
```


注意

多重子查詢使用的結果，造成反應時間比較慢，和因指令寫錯而降低準確度。

14-2-1 相互關係的子查詢（Correlated Subqueries）

在子查詢裡，有相互關係在許多 SQL 廠商是很通常的，有相互關係的子查詢的觀念，如討論過的 ANSI 標準的 SQL 主題一樣，在

這個小時會簡短地討論，有相互關係的子查詢是依賴子查詢裡主查詢的訊息。

在下列各例子，在加入 CUSTOMER_TBL 表格和子查詢裡的 ORDERS_TBL 之間，是依賴主查詢裡 CUSTOMER_TBL(C) 的別名，這個查詢傳回所有客戶已經訂超過 10 或更多單位項目。

TYPE
```
select c.cust_name
from customer_tbl c
where 10 < (select sum(o.qty)
            from orders_tbl o
            where o.cust_id = c.cust_id);
```

Output
```
CUST_NAME
------------------------------
MARYS GIFT SHOP
SCOTTYS MARKET
SCHYLERS NOVELTIES
```


註解

在有相互關係的子查詢情況下，在子查詢能處理之前，主查詢裡的是必須先完成參考表格。

在下一個指令子查詢些微地修改，展示給您看每個客戶所訂單位的總量，再確認早先的結果。

TYPE
```
select c.cust_name, sum(o.qty)
from customer_tbl c,
     orders_tbl o
where c.cust_id = o.cust_id
group by c.cust_name;
```

Output
```
CUST_NAME                      SUM(O.QTY)
------------------------------ ----------
ANDYS CANDIES                           2
GAVINS PLACE                           10
LESLIE GLEASON                          1
MARYS GIFT SHOP                       100
SCHYLERS NOVELTIES                     25
```

```
SCOTTYS MARKET                          20
WENDY WOLF                               2

7 rows selected.
```

因爲另外的一個欄選擇 SUM 函數，所以這個例子裡需要 GROUP BY 子句，給您每個客戶的總數，所以在最初的子查詢，不需要 GROUP BY 子句，因爲使用 SUM 來達到總合全體查詢，這是指全體查詢對個別客戶的記錄。

14-3 摘要

藉著簡單的定義和一般的觀念，子查詢是在另外的查詢裡面，以條件來進一步執行查詢，子查詢可能可以在 SQL 指令的 WHERE 或 HAVING 子句使用，查詢典型地在其他查詢（資料查詢語言，DQL）裡面使用，但是也可能在資料處理語言（DML）指令分解使用，例如 INSERT、UPDATE 和 DELETE，在子查詢裡，所有和 DML 指令使用的基本都應用到，子查詢語法事實上是相同於單查詢的，但有些較小的限制，這些限制的其中一個是 ORDER BY 子句不能在子查詢裡面使用：然而，GROUP BY 子句能使用，達到實際上相同的效果，子查詢用來放置條件，主查詢不必知道，以提供給 SQL 較多的能力和彈性。

14-3-1 Q&A

Q 在子查詢的例子裡，我相當注意第一行開頭的空格，在子查詢的語法裡，這元件是必需的嗎？

A 第一行開頭的空格只被用在分割指令爲分開的零件，使指令更容易讀和更容易了解。

Q 在單一查詢所用的箝入子查詢的數目有上限嗎？

A 例如箝入的子查詢數目上限，與在查詢時可加入的表格數目，是因每個廠商而特定的，有些廠商可能沒有限度，雖然使用太多箝

入的子查詢，會降低 SQL 執行效果，大多數的限制是因爲硬體的設備如，中央處理器速度和系統記憶體影響與許多其他的考慮因素。

Q 在查詢與子查詢除錯是非常困惑，特別是箝入子查詢時，最好是用什麼方法做子查詢與查詢除錯？

A 在子查詢裡除錯查詢的最好方式爲評估區段裡的查詢，首先，評估最低層的子查詢，然後到主查詢（相同的方法評估查詢資料庫）檢查您的工作，當您評估每個子查詢的時候，爲了要檢查您的查詢邏輯，您能以數值代替每次傳回的值查詢，有子查詢的錯誤，時常是運算子的使用，例如 (=)、IN、>、< 等等。

14-4 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

14-4-1 隨堂測驗

1. 當與 SELECT 指令用的時候，子查詢的函數是什麼?
2. 當使用子查詢 UPDATE 指令的時候，您能更新超過一個欄嗎？
3. 下列各項語法是正確的嗎?如果不是，正確的語法是什麼？

a.

```
SELECT CUST_ID, CUST_NAME
        FROM CUSTOMER_TBL
        WHERE CUST_ID =
                        (SELECT CUST_ID
                                FROM ORDERS_TBL
                                WHERE ORD_NUM = '16C17')
```

b.

```
SELECT EMP_ID, SALARY
       FROM EMPLOYEE_PAY_TBL
       WHERE SALARY BETWEEN 20000
                    AND (SELECT SALARY
                         FROM EMPLOYEE_ID
                         WHERE SALARY = '40000')
```

c.

```
UPDATE PRODUCTS_TBL
   SET COST = 1.15
   WHERE CUST_ID =
                  (SELECT CUST_ID
                   FROM ORDERS TBL
                   WHERE ORD_NUM = '32A132')
```

4. 如果下列各項指令執行後將會發生什麼事？

```
DELETE FROM EMPLOYEE_TBL
WHERE EMP_ID IN
              (SELECT EMP_ID
              FROM EMPLOYEE_PAY_TBL)
```

14-4-2 練習題

使用下列各表格完成練習：

EMPLOYEE_TBL

emp_id	varchar2(9)	not null	primary key
last_name	varchar2(15)	not null	
first_name	varchar2(15)	not null	
middle_name	varchar2(15)		
address	varchar2(30)	not null	
city	varchar2(15)	not null	
state	char(2)	not null	
zip	number(5)	not null	
phone	char(10)		
pager	char(10)		

EMPLOYEE_PAY_TBL

emp_id	varchar2(9)	not null	primary key
position	varchar2(15)	not null	
date_hire	date		
pay_rate	number(4,2)	not null	
date_last_raise	date		

constraint emp_fk foreign key (emp_id) references employee_tbl (emp_id)

CUSTOMER_TBL

cust_id	varchar2(10)	not null	primary key
cust_name	varchar2(30)	not null	
cust_address	varchar2(20)	not null	
cust_city	varchar2(15)	not null	
cust_state	char(2)	not null	
cust_zip	number(5)	not null	
cust_phone	number(10)		
cust_fax	number(10)		

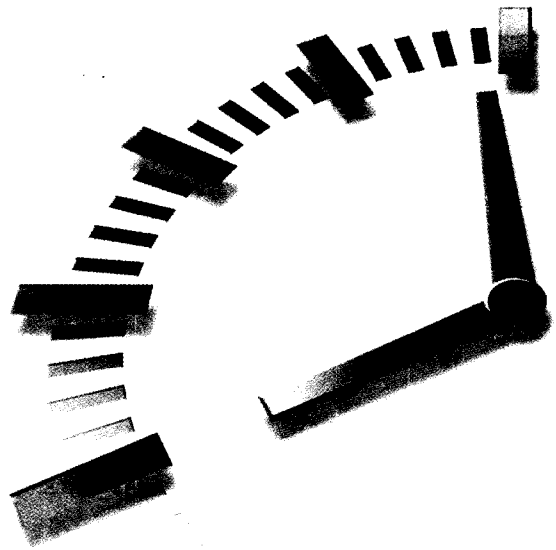
ORDERS_TBL

ord_num	varchar2(10)	not null	primary key
cust_id	varchar2(10)	not null	
prod_id	varchar2(10)	not null	
qty	number(6)	not null	
ord_date	date		

PRODUCTS_TBL

prod_id	varchar2(10)	not null	primary key
prod_desc	varchar2(40)	not null	
cost	number(6,2)	not null	

1. 使用子查詢，寫 SQL 指令更新 CUSTOMER_TBL 表格，變更客戶名字為 DAVID MARKET，訂單號碼為 23E934。
2. 使用子查詢，傳回所有員工的 pay rate 比 JOHN DOE 多與誰的員工編號是 343559876 的查詢。
3. 使用子查詢，列出 COST 超過所有產品平均值的所有產品的查詢。



第 15 小時

組合多重查詢

在這個小時，您將學習該如何使用 UNION、UNION ALL、INTERSECT 與 EXCEPT 等運算子，組合建立 SQL 查詢，再一次，您必須檢查您的廠商使用 UNION、UNION ALL、INTERSECT 與 EXCEPT 等運算子的方法與相關變數。

本小時的重點包括：

- 組合查詢使用運算子的概觀
- 何時該使用哪些指令組合查詢
- 使用 GROUP BY 與合成運算子
- 使用 ORDER BY 與合成運算子
- 該如何取回正確的資料

15-1 單一查詢 vs. 合成查詢

單一查詢是一個 SELECT 指令，而合成查詢包括二或多個 SELECT 指令。

合成查詢藉由使用一些運算子而形成，用來加入二個查詢，下列各例子裡的 UNION 運算子用來加入二個查詢。

單一 SQL 指令能寫成：

```
SELECT EMP_ID, SALARY, PAY_RATE
FROM EMPLOYEE_PAY_TBL
WHERE SALARY IS NOT NULL OR
PAY_RATE IS NOT NULL
```

這是使用 UNION 運算子的相同指令：

```
SELECT EMP_ID, SALARY
FROM EMPLOYEE_PAY_TBL
WHERE SALARY IS NOT NULL
UNION
SELECT EMP_ID, PAY_RATE
FROM EMPLOYEE_PAY_TBL
WHERE PAY_RATE IS NOT NULL
```

之前的指令傳回薪資是以每小時支付或是固定支付薪資的所有員工。

註解

如果您執行第二個查詢，輸出二個欄名：EMP_ID 和 SALARY，在 SALARY 欄底下，列出每個人薪資的 pay rate，當使用 UNION 運算子的時候，在第一個 SELECT 所使用的欄名稱或欄別名，來決定欄名。

15-2 為什麼我要使用合成查詢？

合成運算子用來組合，而且限制二個 SELECT 指令結果，這些運算子能用來傳回或限制重複記錄的輸出，合成運算子能帶來儲存在不同欄位的相似資料。

合成查詢允許您組合更多查詢，傳回單一組的資料，合成查詢時常較有複雜條件，所以單一查詢比較簡單，合成查詢也允許資料檢索提供更多的彈性。

15-3 合成查詢運算子

合成查詢運算子在資料庫廠商之中有相當的不同，ANSI 標準包括下列各項運算子：UNION、UNION ALL、EXCEPT 與 INTERSECT，在下列的區段會討論。

15-3-1 UNION 運算子

UNION 運算子用來組合二個或更多 SELECT 指令的結果，傳回任何沒有重複的列，換句話說，如果輸出的列在查詢的結果存在，相同的列將不會傳回，即使它在第二個用 UNION 運算子組合查詢也存在。爲了要使用 UNION，每個 SELECT 必須選擇相同數目的欄，相同數目的欄運算句，相同的資料格式，而且與他們相同的排序，但是他們長度不一定要相同。語法如下列：

```
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
UNION
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
```

Usage：

```
SELECT EMP_ID FROM EMPLOYEE_TBL
UNION
SELECT EMP_ID FROM EMPLOYEE_PAY_TBL
```

ANALYSIS 兩個表格的員工 ID 在結果只出現一次。

這個小時的例子，從選擇二個簡單表格開始：

TYPE
```
Select prod_desc from products_tbl;
```

Output
```
PROD_DESC
------------------------
WITCHES COSTUME
PLASTIC PUMPKIN 18 INCH
FALSE PARAFFIN TEETH
LIGHTED LANTERNS
ASSORTED COSTUMES
CANDY CORN
PUMPKIN CANDY
PLASTIC SPIDERS
ASSORTED MASKS

9 rows selected.
```

TYPE
```
select prod_desc from products_tmp;
```

Output
```
PROD_DESC
------------------------
ASSORTED COSTUMES
ASSORTED MASKS
WITCHES COSTUME
FALSE PARAFFIN TEETH
LIGHTED LANTERNS
CANDY CORN

6 rows selected.
```

現在您使用 UNION 運算子結合二個相同的查詢，製造合成查詢。

TYPE
```
select prod_desc from products_tbl
UNION
select prod_desc from products_tmp;
```

Output
```
PROD_DESC
------------------------
ASSORTED COSTUMES
ASSORTED MASKS
CANDY CORN
FALSE PARAFFIN TEETH
LIGHTED LANTERNS
PLASTIC PUMPKIN 18 INCH
PLASTIC SPIDERS
PUMPKIN CANDY
WITCHES COSTUME

9 rows selected.
```

在第一個查詢，傳回九列的資料，而且六列的資料從第二個查詢傳回，當 UNION 運算子組合二個查詢的時候，傳回九列的資料，因爲當使用 UNION 運算子的時候，重複的資料列不會傳回，所以傳回九個列。

下個例子表示使用 UNION 運算子，組合二個不相關的查詢：

TYPE
```
select prod_desc from products_tbl
union
select last_name from employee_tbl;
```

Output
```
PROD_DESC
------------------------
ASSORTED COSTUMES
ASSORTED MASKS
CANDY CORN
FALSE PARAFFIN TEETH
GLASS
LIGHTED LANTERNS
PLASTIC PUMPKIN 18 INCH
PLASTIC SPIDERS
PLEW
PUMPKIN CANDY
SPURGEON
```

```
STEPHENS
WALLACE
WITCHES COSTUME

14 rows selected.
```

一起列出 PROD_DESC 和 LAST_NAME 數值，而且欄名在第一個查詢的欄名稱時拿走。

15-3-2 UNION ALL 運算子

UNION ALL 運算子用來組合二個 SELECT 指令結果（包括所重複列的資料），適用於 UNION 的規則全部適用在 UNION ALL，UNION 和 UNIONALL 運算子，雖然一個傳回重複列的資料，而另一個則不是。語法如下：

```
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
UNION ALL
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
```

Usage：

```
SELECT EMP_ID FROM EMPLOYEE_TBL
UNION ALL
SELECT EMP_ID FROM EMPLOYEE_PAY_TBL
```

ANALYSIS 之前的 SQL 指令傳回來自兩個表格所有的員工 ID 與重複列。

這裡有和之前區段裡相同的合成查詢，但是使用 UNION ALL 運算子：

TYPE
```
select prod_desc from products_tbl
union all
select prod_desc from products_tmp;
```

Output

```
PROD_DESC
------------------------
WITCHES COSTUME
PLASTIC PUMPKIN 18 INCH
FALSE PARAFFIN TEETH
LIGHTED LANTERNS
ASSORTED COSTUMES
CANDY CORN
PUMPKIN CANDY
PLASTIC SPIDERS
ASSORTED MASKS
ASSORTED COSTUMES
ASSORTED MASKS
WITCHES COSTUME
FALSE PARAFFIN TEETH
LIGHTED LANTERNS
CANDY CORN

15 rows selected.
```

請注意在這個查詢是傳回 15 列 (9+6)，因爲 UNION ALL 運算子一起傳回重複記錄。

15-3-3 INTERSECT 運算子

INTERSECT 運算子用來組合二個 SELECT 指令，但是只傳回第一個 SELECT 指令和第二個 SELECT 指令相同的列，正如與 UNION 使用的時候，相同的規則應用在 INTERSECT 運算子。語法如下：

```
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
INTERSECT
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
```

Usage：

```
SELECT CUST_ID FROM CUSTOMER_TBL
INTERSECT
SELECT CUST_ID FROM ORDERS_TBL;
```

ANALYSIS 之前的 SQL 指令傳回已經下訂單的客戶編號。

下列例子說明 INTERSECT 在這個小時使用在二個最初的查詢：

TYPE
```
select prod_desc from products_tbl
intersect
select prod_desc from products_tmp;
```

Output
```
PROD_DESC
-------------------------
ASSORTED COSTUMES
ASSORTED MASKS
CANDY CORN
FALSE PARAFFIN TEETH
LIGHTED LANTERNS
WITCHES COSTUME

6 rows selected.
```

因爲在二個單一查詢輸出結果中只有六個列是相同的，所以只有六個列被傳回。

15-3-4 EXCEPT 運算子

EXCEPT 運算子組合二個 SELECT 指令，和傳回第一個 SELECT 指令所選的列，而第二個 SELECT 指令沒有選到，再一次，適用於 UNION 運算子的規則也適用於 EXCEPT 運算子，語法如下：

```
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
EXCEPT
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
```

學習下列例子：

TYPE
```
select prod_desc from products_tbl
EXCEPT
select prod_desc from products_tmp;
```

Output
```
PROD_DESC
-------------------------
PLASTIC PUMPKIN 18 INCH
PLASTIC SPIDERS
PUMPKIN CANDY

3 rows selected.
```

依照結果，第一個查詢傳回三個列，但在第二個查詢傳回卻沒有。


註解

EXCEPT 運算子在一些廠商稱為 MINUS 運算子，檢查您的廠商執行類似 EXCEPT 運算子的運算子名稱與相關函數。

TYPE
```
select prod_desc from products_tbl
minus
select prod_desc from products_tmp;
```

Output
```
PROD_DESC
-------------------------
PLASTIC PUMPKIN 18 INCH
PLASTIC SPIDERS
PUMPKIN CANDY

3 rows selected.
```

15-4 使用 ORDER BY 與合成查詢

ORDER BY 子句合成查詢一起使用，然而，ORDER BY 子句只能用來排序兩個查詢的結果，因此，只能有一個 ORDER BY 子句在合成查詢裡，即使合成查詢可能由多重查詢或 SELECT 指令所組成，ORDER BY 必須參考排序的別名或欄排序的數目。語法如下：

```
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
OPERATOR{UNION | EXCEPT | INTERSECT | UNION ALL}
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
[ ORDER BY ]
```

Usage：

```
SELECT EMP_ID FROM EMPLOYEE_TBL
UNION
SELECT EMP_ID FROM EMPLOYEE_PAY_TBL
ORDER BY 1;
```

ANALYSIS 合成查詢的結果用每個查詢的第一個欄排序，重複的記錄能容易地藉由分類找出合成查詢。

註解

在 ORDER BY 子句旁的參考數字 1 是代替真實的欄名稱。

ANALYSIS 之前的 SQL 指令傳回來自 EMPLOYEE_TBL 和 EMPLOYEE_PAY_TBL 的員工 ID，但是不顯示重複的紀錄與用員工 ID 來排序。

下列各例子顯示 ORDER BY 子句應用在合成查詢，欄名稱能用在 ORDER BY 子句旁邊，如果在所有個別查詢裡，欄用相同名稱來排序。

TYPE

```
select prod_desc from products_tbl
union
select prod_desc from products_tbl
order by prod_desc;
```

Output

```
PROD_DESC
------------------------
ASSORTED COSTUMES
ASSORTED MASKS
CANDY CORN
FALSE PARAFFIN TEETH
LIGHTED LANTERNS
PLASTIC PUMPKIN 18 INCH
PLASTIC SPIDERS
```

```
PUMPKIN CANDY
WITCHES COSTUME

9 rows selected.
```

下列的查詢用數字代替真實的欄名稱在 ORDER BY 子句旁邊：

TYPE

```
select prod_desc from products_tbl
union
select prod_desc from products_tbl
order by 1;
```

Output

```
PROD_DESC
------------------------
ASSORTED COSTUMES
ASSORTED MASKS
CANDY CORN
FALSE PARAFFIN TEETH
LIGHTED LANTERNS
PLASTIC PUMPKIN 18 INCH
PLASTIC SPIDERS
PUMPKIN CANDY
WITCHES COSTUME

9 rows selected.
```

15-5 使用 GROUP BY 與合成查詢

不像 ORDER BY，GROUP BY 能在合成查詢的每個 SELECT 指令中應用，也可能在下列各項查詢中使用，除此之外，HAVING 子句（一起和 GROUP BY 子句使用）可能在合成指令使用 SELECT 指令。語法如下：

```
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
[ GROUP BY ]
[ HAVING ]
OPERATOR {UNION | EXCEPT | INTERSECT | UNION ALL}
SELECT COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE ]
[ GROUP BY ]
```

```
[ HAVING ]
[ ORDER BY ]
```

在下列各例子，您選擇用文字的字串表現客戶記錄，員工記錄和產品記錄，每個個別的查詢只是簡單統計適當的表格裡所有的記錄，GROUP BY 子句用數字 1，來聚集所有報表結果，數值 1 代表每個查詢表格裡的第一個欄。

TYPE

```
select 'CUSTOMERS' TYPE, COUNT(*)
FROM CUSTOMER_TBL
UNION
SELECT 'EMPLOYEES' TYPE, COUNT(*)
FROM EMPLOYEE_TBL
UNION
SELECT 'PRODUCTS' TYPE, COUNT(*)
FROM PRODUCTS_TBL
GROUP BY 1;
```

Output

```
TYPE         COUNT(*)
-------- ----------
CUSTOMERS          15
EMPLOYEES           6
PRODUCTS            9

3 rows selected.
```

下列查詢和之前的查詢相同，除了使用 ORDER BY 子句外：

TYPE

```
select CUSTOMERS TYPE, COUNT(*)
FROM CUSTOMER_TBL
UNION
SELECT EMPLOYEES TYPE, COUNT(*)
FROM EMPLOYEE_TBL
UNION
SELECT PRODUCTS TYPE, COUNT(*)
FROM PRODUCTS_TBL
GROUP BY 1
ORDER BY 2;
```

Output

```
TYPE         COUNT(*)
-------- ----------
EMPLOYEES           6
PRODUCTS            9
CUSTOMERS          15

3 rows selected.
```

這用欄 2 分類，欄 2 是每個表格的統計，因此，最後輸出結果的排序是從最少的到最大的。

15-6 取回正確的資料

當使用合成運算子的時候要謹慎，舉例來說，在第一個單獨查詢，如果您使用 INTERSECT 運算子與不當的 SELECT 指令，傳回資料的可能不正確或不完全，除此之外，考慮在重複記錄時，是否該使用 UNION 和 UNION ALL 運算子，如果使用 EXCEPT 會怎麼樣？您是否不需要任何一個列，在第二個查詢裡傳回？正如您所見，錯誤合成查詢運算子與排序錯誤，會容易誤導將傳回的資料。

註解

查詢傳回不完全的資料資格，如同不正確的資料。

15-7 摘要

您學習過合成查詢，這個小時所提到的 SQL 指令由單一查詢所組成，合成查詢允許多重查詢一起使用單一查詢來產生，產生所需要輸出的資料，合成查詢運算子討論過的運算子，包括 UNION、UNION ALL、INTERSECT 與 EXCEPT (MINUS)。UNION 傳回二個單一查詢沒有顯示重複列的資料輸出結果，UNION ALL 只是顯示所有單一查詢的輸出結果，不管重複列是否已存在。INTERSECT 用來傳回在二個查詢之間的相同列。EXCEPT（相同於 MINUS）用來傳回資料存在一個查詢而不在另外一個查詢的結果，合成查詢提供較多彈性，當試著使滿足各種不同查詢的需求，沒有使用合成的運算子，會使查詢變成非常複雜的。

15-7-1 Q&A

Q 在使用的時候 GROUP BY 子句合成查詢時，如何參考在 GROUP BY 子句的欄？

A 欄可能由真實欄名稱或在查詢放置欄的數目所參考，如果二個查詢的欄名稱不是相同的。

Q 我了解 EXCEPT 運算子能做什麼，但是如果我顛倒 SELECT 指令，結果將會如何改變？

A 是的。個別查詢的排序是非常重要，特別是使用 EXCEPT 或 MINUS 運算子的時候。請記得所有的列被第一個查詢傳回而不被第二個查詢傳回，變更合成查詢裡二個個別查詢的排序，會明確地影響結果。

Q 合成查詢裡的資料格式和欄的長度在兩個查詢必須是相同的嗎？

A 不。只有資料格式必須相同，但長度可以不一致。

15-8 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

15-8-1 隨堂測驗

1. 下列合成查詢語法是否正確？如果不，如何改正語法？使用下列的 EMPLOYEE_TBL 和 EMPLOYEE_PAY_TBL：

```
EMPLOYEE_TBL
-------------------------------------------------
emp_id              varchar2(9)         not null,
last_name           varchar2(15)        not null,
```

EMPLOYEE_TBL

```
first_name          varchar2(15)        not null,
middle_name         varchar2(15),
address             varchar2(30)        not null,
city                varchar2(15)        not null,
state               char(2)             not null,
zip                 number(5)           not null,
phone               char(10),
pager               char(10),
constraint emp_pk primary key (emp_id)
```

EMPLOYEE_PAY_TBL

```
emp_id              varchar2(9)    not null,   primary key,
position            varchar2(15)   not null,
date_hire           date,
pay_rate            number(4,2)    not null,
date_last_raise     date,
salary              number(8,2),
bonus               number(8,2),
constraint emp_fk foreign key (emp_id)
references employee_tbl (emp_id)
```

a.

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME
FROM EMPLOYEE_TBL
UNION
SELECT EMP_ID, POSITION, DATE_HIRE
FROM EMPLOYEE_PAY_TBL
```

b.

```
SELECT EMP_ID FROM EMPLOYEE_TBL
UNION ALL
SELECT EMP_ID FROM EMPLOYEE_PAY_TBL
ORDER BY EMP_ID
```

c.

```
SELECT EMP_ID FROM EMPLOYEE_PAY_TBL
INTERSECT
SELECT EMP_ID FROM EMPLOYEE_TBL
ORDER BY 1
```

2. 對應下列各指令到正確的運算子。

	指令	運算子
a.	展示重複	UNION INTERSECT
b.	傳回唯一與第一個查詢和在第二個查詢對應的列	UNION ALL EXCEPT
c.	傳回沒有重複	
d.	傳回唯一被第一個查詢選擇而不被第二個查詢選擇的列	

15-8-2 練習題

1. 使用 CUSTOMER_TBL 和 ORDERS_TBL 列出：

CUSTOMER_TBL

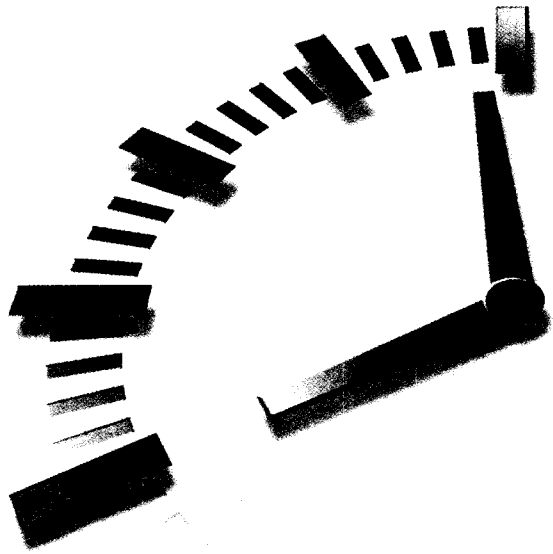
cust_id	varchar2(10)	not null	primary key,
cust_name	varchar2(30)	not null,	
cust_address	varchar2(20)	not null,	
cust_city	varchar2(15)	not null,	
cust_state	char(2)	not null,	
cust_zip	number(5)	not null,	
cust_phone	number(10),		
cust_fax	number(10)		

ORDERS_TBL

ord_num	varchar2(10)	not null	primary key,
cust_id	varchar2(10)	not null,	
prod_id	varchar2(10)	not null,	
qty	number(6)	not null,	
ord_date	date		

a. 寫個合成查詢，找尋已經下訂單的客戶。

b. 寫個合成查詢，找尋那些還沒有下訂單的客戶。



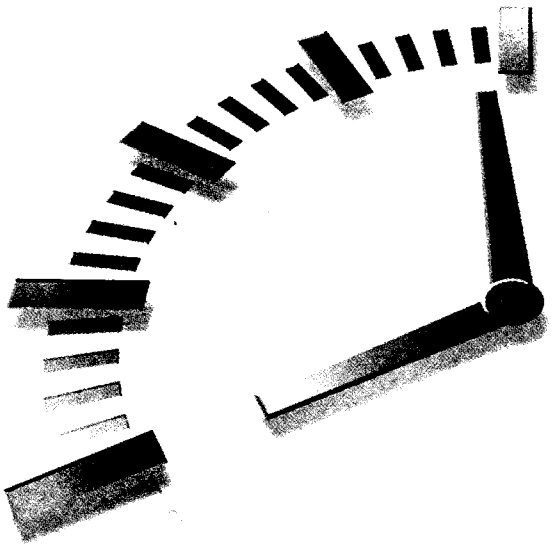
第 V 單元

協調 SQL 執行效率

第 16 小時　使用索引增加執行效果

第 17 小時　增進資料庫執行效率

第Ⅴ単元



第16小時

使用索引增加執行效果

在這個小時期間，您將學習該如何藉由建立與使用索引，增加SQL指令的執行效果。用 CREATE INDEX 指令與學習該如何在已經建立索引的表格使用索引。

本小時的重點包括：

- 如何建立索引？
- 索引如何運作？
- 不同類型的索引
- 何時使用索引？
- 何時不用索引？

16-1 索引是什麼？

簡單來說，索引只是放個指標到表格的資料裡，資料庫裡的索引與書背面的索引是非常相似。舉例來說，如果您要參考書裡所有討論某主題的頁，您首先參照索引，索引所有的主題是按字母排序地，然後指定到一頁或較特定的頁數，在資料庫裡的索引操作方式也是相同，查詢指到表格裡資料實際位置，您實際上是指資料到資料庫裡的一個檔案位置裡，正如您所考量的，您正在參考一個表格。

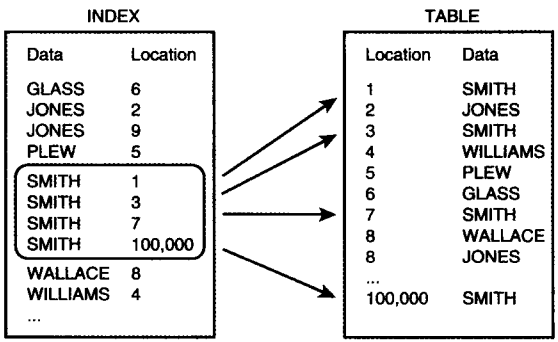
那一個將會是比較快的，一頁接著一頁，爲了找尋一些訊息而看過整本書，或是尋找書所編的索引，而得到頁數？當然，如果那本書是很厚的，使用書索引是最有效率的方法，能節省許多時間。如果說您有一本很少頁的書，在這情況，可能一頁一頁的檢查訊息是與在書索引和頁之間來回地的翻快得許多。當資料庫不使用索引的時候，它是搜尋的是典型地方式，被稱爲完整表格掃描，相同於一頁一頁的檢查訊息。完整表格掃描將於稍後討論。

索引建立是與表格分開地儲存，索引主要目的是增加資料檢索的執行效果，索引的建立或取消對資料沒有任何的影響，然而，取消索引，資料檢索的執行效果可能減慢，索引確實佔實際空間，而且有時常大過表格本身。

16-2 索引如何運作？

當索引建立時，它記錄表格裡與索引欄有關的數值位置，當新資料加入到表格的時候，項目就加到索引，當查詢對資料庫執行時，在 WHERE 子句的欄，所敘述的條件是索引的，在 WHERE 子句，將首先搜尋索引的數值，如果發現數值在索引裡，索引會傳回表格裡所搜尋到資料的精確位置。圖 16.1 舉例說明索引如何運作。

圖 16.1.
使用索引存取表格



假如執行下列的查詢：

```
SELECT *
FROM TABLE_NAME
WHERE NAME = SMITH;
```

如圖 16.1 所示，名字索引是參考所有名字的位置，找出與 SMITH 相同的資料，一旦決定位置，資料很快地能從表格裡取回，在這情況下的資料，依索引用字母順序排列。

如果表格上沒有索引，將會發生完整表格掃描，而且相同的查詢被執行，表示爲了要取回所有的名字爲 SMITH 的，表格裡每個列的資料將會被讀取一次。

16-3 CREATE INDEX 指令

CREATE INDEX 指令，與 SQL 裡的許多其他的指令一樣，在不同的關連式資料庫廠商中而不同，最大多數的關連式資料庫廠商使用 CREATE INDEX 指令。

```
CREATE INDEX INDEX_NAME ON TABLE_NAME
```

語法是廠商開始產生不同的地方，因爲 CREATE INDEX 指令有著不同的選擇項，一些廠商允許儲存子句的規格(如 CREATETABLE 指令)，排序（DESC||ASC），和叢集的使用，您必須檢查您的廠商有關語法正確的使用方式。

16-4 索引的類型

有不同類型的索引能在資料庫裡的表格上建立，全部爲了相同的目標：藉由加快資料檢索來增加資料庫執行效果，這個小時討論單一欄索引（Single-column index）、複合索引（Composite index）和唯一索引（Unique index）。


註解

一些廠商在表格建立期間，也能建立索引，大多數的廠商使用指令，在 CREATE TABLE 指令旁建立索引，您必須檢查您特別廠商有關指令的正確語法。

16-4-1 單一欄索引

表格單一欄上的索引是最簡單和最常用來顯示索引的方式，明顯地，單一欄索引是建立在表格欄上，基本語法如下：

```
CREATE INDEX INDEX_NAME
ON TABLE_NAME (COLUMN_NAME)
```

舉例來說，如果您要在 EMPLOYEE_TBL 表格對員工姓建立索引，建立索引的指令將會看起來像：

```
CREATE INDEX NAME_IDX
ON EMPLOYEE_TBL (LAST_NAME);
```


註解

您應該計劃您的表格與索引，別假定索引已經建立了，且所有的執行效果已經解決，索引可能一點也沒幫助（它可能實際上阻礙執行效果），且佔磁碟空間。


提示

當時常單獨在欄上使用 WHERE 子句為查詢條件時，單一欄索引是最有效的，單一欄最好的索引是個別的數目、序號或系統所分配值。

16-4-2 唯一索引

唯一索引使用不單爲了執行效果，且爲了資料完整性，唯一索引不允許任何的複製數值插入表格之內，否則，唯一索引會執行與一般索引執行的方法相同，語法如下：

```
CREATE UNIQUE INDEX INDEX_NAME
ON TABLE_NAME (COLUMN_NAME)
```


如果您要在 EMPLOYEE_TBL 表格上對員工姓建立唯一索引，建立索引的指令將會看起來像：

```
CREATE UNIQUE INDEX NAME_IDX
ON EMPLOYEE_TBL (LAST_NAME);
```

這個索引的唯一問題是 EMPLOYEE_TBL 表格裡的每個姓一定要是唯一的，所以建立這樣的索引是不實用的，然而，唯一索引應該被欄位建立，例如社會福利編號，因爲這數值對每個人是唯一的。

您可能是覺得奇怪，爲什麼員工社會福利編號是表格的主索引？當您定義表格主索引的時候，索引通常暗地裡建立，然而，公司可能使用員工身份證爲虛構數目，維持每位員工社會福利編號，您或許要索引這個欄，且確定所有的項目進入這個欄是唯一的數值。

註解

唯一索引能只能在數值是唯一的表格欄上建立，換句話說，您不能在已存在資料索引的表格建立唯一索引。

16-4-3 複合索引

複合索引是表格裡有二個或較多欄的索引，當建立複合索引時候，您應該考慮執行效果，如同索引裡的欄排序會影響資料檢索的的速度，通常，爲了最佳執行效果，最限制的數值應該首先放置。然而，敘述欄總是應該被先放置，語法如下：

```
CREATE INDEX INDEX_NAME
ON TABLE_NAME (COLUMN1, COLUMN2)
```

複合索引的例子：

```
CREATE INDEX ORD_IDX
ON ORDERS_TBL (CUST_ID, PROD_ID);
```

在這個例子，您基於 ORDERS_TBL 表格，建立二個欄的複合索引：CUST_ID 和 PROD_ID，您假定這二個欄是時常一起使用，且在查詢裡，爲 WHERE 子句的條件。

提示

> 複合索引最有效的使用方法是，是把表格常用的欄，當做在查詢裡 WHERE 子句的條件一起使用。

16-4-4 複合索引 vs. 單一欄索引

在決定方面是否建立單一欄索引或複合索引的時候，考慮在查詢可能時，經常使用在 WHERE 子句當做過濾條件的欄，如果只有使用一個欄，是應該選擇單一欄索引的。若需要二個或較多個常用欄在 WHERE 子句當做過濾，則複合索引將會是最好的選擇。

16-5 何時該考慮使用索引？

爲了要使主索引運作，唯一索引暗地連同主索引一起運作，外部索引也是很好的索引選擇之一，如同他們時常用來加入上層表格一樣，大部分，如果不是所有的，用來做表格的欄應該被編入索引。

ORDER BY 和 GROUP BY 子句參考的欄，應該考慮爲索引，舉例來說，如果您正分類名字，欄上的任何索引，將會是相當有用的，它依每個名字上的字母自動排序，而單一化實際的分類運算且加快輸出結果。

此外，索引應該在唯一較多的數值欄上建立，或在 WHERE 子句當做過濾條件，傳回來自表格裡較少的資料，而這些是可能產生錯誤的地方，正如產生程式碼與資料庫結構應該是在進入製造前讓廠商測

試，然後再設定索引。這些測試應該是試驗不同組合索引，如沒有索引，單一欄索引和複合索引等，並測試不同索引的規則。有效的使用索引，需要完全瞭解表格關係、查詢、異動需求和資料本身。

16-6 何時避免使用索引？

雖然索引的目的是提高資料庫執行效果，但也有應該避免使用的時候，下列各項指引是考慮是否該使用索引：

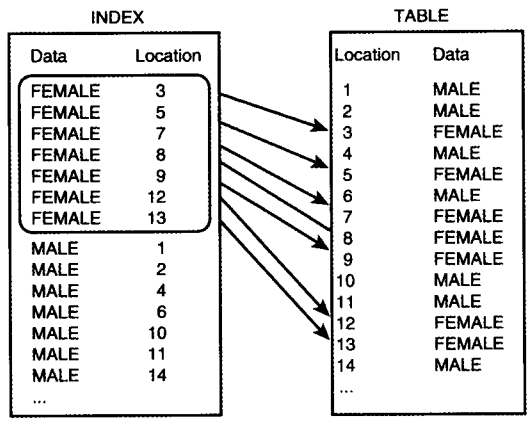
- 索引不應該在小的表格上被使用。
- 索引不應該用在 WHERE 子句為查詢條件的時候，想傳回較多的資料，舉例來說，您將不會為“這”或“和”這類的字設索引。
- 表格時常性和批次性資料更新的運作能索引，然而，整批運作執行效果會因索引減慢執行速度，有表格上的索引的衝突時，常被入或藉由批次檔處理，在取消索引前被修正，然後在運作已經完成後，從新建立索引。
- 索引不應該在包含很多 NULL 數值的欄上使用。
- 時常處理的欄不應該編入索引。

當建立索引的時候應小心表格上極長的鍵，如同在執行時不可避免 I/O 所花費的時間。

您可以看見在圖 16.2，性別欄上的索引證明是沒有幫助的，舉例來說，下列查詢提給資料庫：

```
SELECT *
FROM TABLE_NAME
WHERE GENDER = FEMALE;
```

圖 16.2.
何時避免使用索引



藉由圖 16.2，基於早先的查詢，您能看見表格和它的索引之間的固定活動，由於許多資料由 WHERE GENDER = 'FEMALE'(or MALE) 傳回，資料庫必須不斷地讀索引、然後表格、然後索引、然後表格等等，在這情況，可能執行完整表格掃瞄比較有效率，因爲表格內大多數的資料都必須讀取過。

一般而言，您不會在查詢裡需要傳回很多資料的欄上使用索引，換句話說，不建立索引在如性別的欄上或包含很少確定數值的任何欄。


提示

索引對執行效果可能是非常好的，但是在某些情況可能實際上傷害執行效果，減少建立索引在包含很少唯一數值的欄上，例如性別、住宅狀態等等。

16-6-1 取消索引

索引能被簡單的取消，檢查您特別廠商的相關語法，但是最主要的是使用 DROP 指令，當取消索引的時候應小心，如同執行效果可能激烈地減慢（或增加！）一樣。語法如下：

```
DROP INDEX INDEX_NAME
```

取消索引最通常的理由，是嘗試增加執行效果，請記得如果您取消索引，您也能恢復它，索引有時可能需要從建來減少片段的產生，通常必需以資料庫裡的索引使用作實驗，來決定到最好的執行效果，最好的執行效果可能包括建立索引、取消它和最後從新建立它，與或沒有任何修改。

16-7 摘要

您已經了解索引能用來增加在資料庫裡面執行的查詢、異動處理的性能，資料庫索引，就像書一樣，很快地到表格特定的參考資料，建立索引最通常的方法是使用 CREATEINDEX 指令。在各種不同的 SQL 廠商中，有不同類型的索引，唯一索引、單一欄索引與複合索引是不同類型索引。有許多因數考慮在何時使用那種索引類型，又符合您資料庫的需要。索引的有效使用時常需要一些知識，如了解表格之間的關係和資料的完整性，和一點耐性，耐性能省下現在的幾分鐘、小時或稍後每天運作的時間。

16-7-1 Q&A

Q 索引實際上和表格一樣佔空間嗎？

A 是的，索引在資料庫裡佔實際的空間，事實上，有時索引能變得比被建立的表格還大。

Q 如果您為了使批次運作較快而取消索引，要再恢復索引需要多久？

A 許多因數被考慮，例如被取消索引的大小、中央處理器的使用和機器的能力。

Q 所有的索引應該是唯一索引嗎？

A 不，唯一索引不允許複製值，但可能有的表格裡，需要複製的數值，所以容複製值產生。

16-8 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

16-8-1 隨堂測驗

1. 甚麼是使用索引的主要的缺點？
2. 為什麼在複合索引欄的排序是重要的？
3. 有許多 NULL 的欄可以被編入索引嗎？
4. 表格裡索引主要的目的是停止複製值。
5. 是非題：複合索引主要的理由是為了在索引使用聚合函數。

16-8-2 練習題

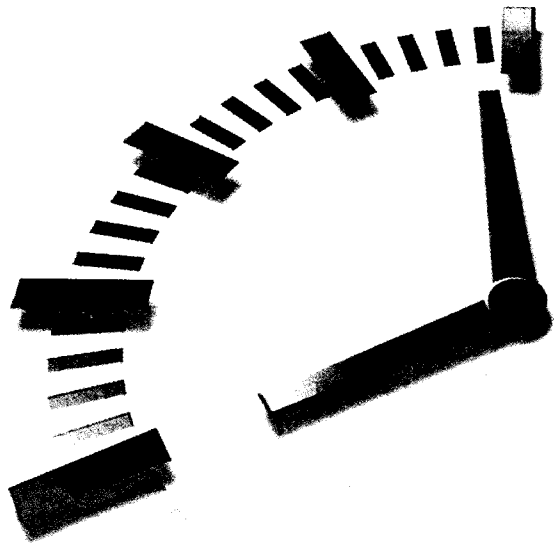
1. 下列各項情形，決定是否該使用索引，如果該使用，什麼類型的索引應該被使用。

   a. 幾個欄，但是較小的表格。

   b. 中型表格，複製不被允許。

   c. 幾個欄，非常大的表格，幾個欄在 WHERE 子句，當做過濾條件。

   d. 大的表格，許多欄與許多資料處理。



第 17 小時

增進資料庫執行效率

在這個小時內，您學習如何使用一些非常簡單的方法協調 SQL 指令，而得到最大的執行效率。

本小時的重點包括：

- 什麼是協調 SQL 指令？
- 協調 SQL 指令 vs. 協調資料庫
- 格式化 SQL 指令
- 適當加入表格
- 最多數的條件限制
- 完整表格掃瞄
- 呼叫使用索引
- 避免使用 OR 和 HAVING
- 避免大量分類運算

17-1 什麼是協調 SQL 指令？

協調 SQL 指令，是把 SQL 指令達到最佳化與最有效率的方式，協調 SQL 開始於在查詢元件裡的基本安排順序與方式，簡單的格式化在最佳化裡扮演更重要的角色。

協調 SQL 指令主要在討論 FROM 和 WHERE 子句的用法，就是在這二個子句裡，資料庫伺服器決定該如何執行查詢與得到資料，到此您已經知道 FROM 和 WHERE 子句基本用法，現在是比較好的時間來開始學習如何協調它們得到較好的結果。

17-2 協調 SQL 指令 vs. 協調資料庫

在繼續討論協調 SQL 指令前，了解協調 SQL 指令與協調資料庫的不同是非常重要的。

協調資料庫實際上是協調真實資料庫的程序，包含配置記憶體、磁碟用法、中央處理器、輸入／輸出和真實資料庫的處理，協調資料庫也包括管理和處理資料庫本身結構，如表格的設計、規劃與索引設定，在協調資料庫的時候，有許多其他的因素要考慮，但是這些主要藉由資料庫管理員完成，協調資料庫的目的是確定在預期資料庫的處理方式內，資料庫已經設計到最好的處理程度。

協調 SQL 是協調 SQL 指令存取資料庫的程序，這些 SQL 指令包括插入、更新和刪除資料庫查詢與異動運算，協調 SQL 指令的目的是利用資料庫和系統資源使用索引的優點，用最有效的公式表示目前存取資料庫的狀態。

註解

協調資料庫和協調 SQL 指令，都是在存取資料庫的時候，為了要達成最佳結果所做的程序，不適當的協調資料庫，可能會浪費在協調 SQL 裡的努力而反之亦然。

17-3 格式化 SQL 指令

格式化 SQL 指令聽起來好像沒什麼，但是值得一提的是建立 SQL 指令的時候，對 SQL 的新手或許會沒考慮到。下列的討論列出一般考量，：

- 格式化 SQL 指令增加可讀性
- FROM 子句在表格的排序
- 在 WHERE 子句限制條件所放置的位置
- 在 WHERE 子句放置加入條件的位置

註解

大多數的關連式資料庫廠商有所謂的 SQL optimizer 評估 SQL 指令，而且基於 SQL 指令所寫的方式和資料庫裡所用的索引，來決定最好執行指令的方法，並不是所有的 optimizers 都相同。請檢查您的廠商或與資料庫管理員討論，optimizer 如何讀 SQL 編碼，您應該了解 optimizer 如何工作，並有效地協調 SQL 指令。

17-3-1 格式化 SQL 指令增加可讀性

爲了增加可讀性而格式化，在寫 SQL 指令是相當平常的，但是許多 SQL 指令並沒有簡潔的寫出，雖然指令的簡潔與否是不影響實際的執行效率，但是小心的格式化是協調的第一個步驟，當您看著 SQL 指令企圖協調的時候，使指令可讀是必須做的第一件事，如何決定指令寫得好不好，是非常困難的！

一些基本規則讓指令增加可讀性：

- 每個子句開始總是用新的一行。舉例來說，放置 FROM 子句在與 SELECT 子句不同行，放置 WHERE 子句與 FROM 子句在不同一行等等。
- 當指令裡子句函數中的獨立變數超過一個行的時候，使用定位點或空格來隔開。

- 一致使用定位點與空格。
- 指令使用多重表格的時候，使用表格別名，因為取得資格的欄位上，使用完整的表格名稱會造成閱讀困難。
- 在 SQL 指令裡使用特有節省空間的指令，如果它是可用在您特定廠指令。
- 如果選擇許多欄，在 SELECT 子句，開始新的一行，使用每個欄的名稱。
- 如果選擇許多表格，在 FROM 子句，開始新的一行，使用每個表格的名稱。
- 在 WHERE 子句，開始新的一行，能容易看見他們使用的指令和所有條件的順序。

無法讀取指令的例子：

TYPE
```
SELECT CUSTOMER_TBL.CUST_ID, CUSTOMER_TBL.CUST_NAME,
CUSTOMER_TBL.CUST_PHONE, ORDERS_TBL.ORD_NUM, ORDERS_TBL.QTY
FROM CUSTOMER_TBL, ORDERS_TBL
WHERE CUSTOMER_TBL.CUST_ID = ORDERS_TBL.CUST_ID
AND ORDERS_TBL.QTY > 1 AND CUSTOMER_TBL.CUST_NAME LIKE 'G%'
ORDER BY CUSTOMER_TBL.CUST_NAME;
```

Output
```
CUST_ID    CUST_NAME                      CUST_PHONE ORD_NUM
QTY
---------- ------------------------------ ---------- ----------
----------
287        GAVINS PLACE                   3172719991 18D778
10

1 row selected.
```

增進可讀性的重新格式化指令的例子：

TYPE
```
SELECT C.CUST_ID,
       C.CUST_NAME,
       C.CUST_PHONE,
       O.ORD_NUM,
       O.QTY
```

```
FROM ORDERS_TBL O,
     CUSTOMER_TBL C
WHERE O.CUST_ID = C.CUST_ID
  AND O.QTY > 1
  AND C.CUST_NAME LIKE 'G%'
ORDER BY 2;
```

Output

```
CUST_ID    CUST_NAME                    CUST_PHONE ORD_NUM
QTY
---------- ---------------------------- ---------- ----------
----------
287        GAVINS PLACE                 3172719991 18D778
10

1 row selected.
```


兩個指令正好相同，但是第二個指令增加了可讀性，第二個指令藉由在查詢裡經由 FROM 子句所定義的表格別名，使指令簡化，間距已經用在排列每個子句的元件，使每個子句突出。

再一次強調，增加指令的可讀性不會直接增進它的執行效率，但是可幫助您在修改和除錯所費冗長的時間，和避免面對複雜的指令，現在您能容易識別選擇與使用表格欄位，及加入正在執行的表格和放置查詢的條件。

17-3-2 在 FROM 子句中安排適當的表格

在 FROM 子句表格的安排或排序，可能因不同 optimizer 而對 SQL 指令有所差異，舉例來說，可能先列出較小的表格然後再列出那些較大的表格是比較有幫助的，許多有經驗的使用者已經發現，在 FROM 子句列出較大的表格證明是更有效率的。

FROM 子句的例子：

```
FROM SMALLEST TABLE,
     LARGEST TABLE
```


註解

檢查您的廠商為執行效率所做的提示，在 FROM 子句如何列出多重表格。

17-3-3 適當的依序加入條件

第 13 小時“在查詢裡加入表格”所學習到，最常使用加入的是基礎表格聯接表格，利用一個或較多常用的欄，瞭解該加入那個表格。基礎表格是主要的表格，大部分或所有的表格都在查詢裡加入。來自基礎表格的欄，通常放置在 WHERE 子句加入運算的右邊，加入到基礎表格的表格是依照順序從最小的到最大排序，正如表格列出在 FROM 子句般。

不應該有基礎表格（BASE TABLE），因爲表格應該從最小列到最大列，最大的表格通常放置在 WHERE 子句加入運算的右邊，加入條件應該是在 WHERE 子句的第一個位置然後是過濾子句。例子如下：

```
FROM TABLE1,                          從最小的表格
     TABLE2,                             到
     TABLE3                          最大的表格，也是（BASE TABLE）
WHERE TABLE1.COLUMN = TABLE3.COLUMN          加入條件
  AND TABLE2.COLUMN = TABLE3.COLUMN          加入條件
[ AND CONDITION1 ]              過濾子句
[ AND CONDITION2 ]              過濾子句
```

在這個例子中，TABLE3 被當作基礎表格使用，TABLE1 和 TABLE2 被加入到 TABLE3 證明簡化的效率。


提示

因為加入傳回來自表格較多的資料列，加入條件應該是在限制條件後傳回較符合的資料。

17-3-4 最限制的條件

最限制的條件是在 SQL 查詢裡爲了達成最佳執行效率所控制的因素。何謂最限制的條件？條件在 WHERE 子句，所傳回的資料列最少，

相反地，最沒有限制的條件會傳回最多的資料列，這一小時是考慮最限制的條件，因爲條件將傳回大部分查詢過濾後的資料。

SQL optimizer 的目的首先評估最限制的條件，因爲較少的資料因條件而傳回，如此減少查詢的執行，查詢裡最限制的條件，需要知道 optimizer 如何操作的知識，optimizers 在某些情況的運作像是從底部讀 WHERE 子句。因此，您要放置最限制的條件在 WHERE 子句後面，因爲它是 optimizer 首先讀的條件。

```
FROM TABLE1,                          從最小的表格
     TABLE2,                                到
     TABLE3                          最大的表格，也是（BASE TABLE）
WHERE TABLE1.COLUMN = TABLE3.COLUMN          加入條件
  AND TABLE2.COLUMN = TABLE3.COLUMN          加入條件
[ AND CONDITION1 ]           最少限制的
[ AND CONDITION2 ]           最多限制的
```


提示

如果您不知道廠商的 SQL optimizer 如何工作，而 DBA 也不知道，或您沒有充份的相關文件，先執行一個較大的查詢，然後在 WHERE 子句重新排列條件，當完成需要的查詢時，一定要記錄每一次所作變化，應該只須執行一些測試，就能演算出 optimizer 是否讀 WHERE 子句從頂端到底部，或底部到頂端。

使用假表格的例子如下：

Table:	TEST
Row count:	95,867
Conditions:	WHERE LAST_NAME = 'SMITH'
	returns 2,000 rows
	WHERE CITY = 'INDIANAPOLIS'
	returns 30,000 rows
Most restrictive condition is:	WHERE LAST_NAME = 'SMITH'

```
QUERY1:
```

```
SELECT COUNT(*)
FROM TEST
WHERE LAST_NAME = 'SMITH'
  AND CITY = 'INDIANAPOLIS';

  COUNT(*)
----------
     1,024

QUERY2:

SELECT COUNT(*)
FROM TEST
WHERE CITY = 'INDIANAPOLIS'
  AND LAST_NAME = 'SMITH';

  COUNT(*)
----------
     1,024
```

假如 QUERY1 花 20 秒完成而 QUERY2 花 10 秒完成時，因爲 QUERY2 傳回較快的結果且最限制的條件在 WHERE 子句被列出，可以假定 optimizer 在 WHERE 子句是從下讀到上的。


註解

試著使用索引欄查詢最限制的條件是個很好的練習，索引通常可增進查詢執行效率。

17-4 完整表格掃瞄（Full Table Scans）

當不使用索引或是在 SQL 指令所用表格裡沒有索引的時候，完整表格掃描就會發生，完整表格掃描傳回資料的速度通常比有用索引慢很多，當表格愈大，執行完整表格掃描時，資料傳回的速度也愈慢，當查詢 optimizer 決定是否使用索引來執行 SQL 指令的時候，在大多數的情況，索引是存在的。

廠商能巧妙的決定在查詢 optimizers 是否該用索引，這基於在資料庫上統計被聚集的物件，例如物件的大小和條件傳回有索引欄的估計數目，有關 optimizer 的決策能力，請參照您的廠商所編寫的文件。

17-4-1 如何避免完整表格掃瞄

當讀大表格的時候，應該避免執行完整表格掃描，舉例來說，當沒有索引的表格時，執行完整表格掃描是適當的，但是通常花較長的時間傳回資料，索引應該用在大的表格，小的表格 optimizer 可能選擇完整表格掃瞄而不使用索引，如果小的表格有索引的情況下，考慮取消索引而保留在資料庫裡其他需要索引物件空間的。

提示

除了確認索引的存在外，最容易避免完整表格掃瞄的方法就是在查詢裡的 WHERE 子句設條件，將過濾的資料傳回。

應該有索引的資料：

- 欄當做主索引
- 欄當做外部索引
- 欄時常用在加入表格
- 欄時常作為查詢使用的條件
- 唯一較多資料的欄

註解

完整掃描表格是很好的，完整表格掃描應該在查詢時，較小的表格執行或查詢的條件傳回較多的資料，最容易迫使完整表格掃描的方法是，避免建立表格索引。

17-5 其他執行效率的考量

在協調 SQL 指令時，其他因執行效率而考慮的因素應記下，下列章節將討論以下觀念：

- 使用 LIKE 運算子和萬用字元
- 避免 OR 運算子
- 避免 HAVING 子句
- 避免大量分類運算
- 使用儲存程序（stored procedures）

17-5-1 使用 LIKE 運算子和萬用字元

LIKE 運算子是用在查詢裡放置條件的有用工具，查詢裡的萬用字元的放置和使用，能除去傳回多餘資料的可能性，萬用字元尋找相似的資料是非常有彈性的(數值不一定是相等的資料)。

假如您寫查詢使用 EMPLOYEE_TBL，選擇 EMP_ID、LAST_NAME、FIRST_NAME 和 STATE 欄，您需要知道所有姓 Stevens 的員工 ID、名字和洲名。以不同放置萬用字元的三個 SQL 指令爲例：

```
QUERY1:

SELECT EMP_ID, LAST_NAME, FIRST_NAME, STATE
FROM EMPLOYEE_TBL
WHERE LAST_NAME LIKE '%E%'

QUERY2:

SELECT EMP_ID, LAST_NAME, FIRST_NAME, STATE
FROM EMPLOYEE_TBL
WHERE LAST_NAME LIKE '%EVENS%'

QUERY3:

SELECT EMP_ID, LAST_NAME, FIRST_NAME, STATE
FROM EMPLOYEE_TBL
```

```
WHERE LAST_NAME LIKE '%ST%'
```

SQL 指令不一定傳回相同的結果，比較有可能的是，QUERY1 將會傳回較多的列，QUERY2 和 QUERY3 是爲了要傳回資料而設定的，如此可以除去其他資料符合 QUERY1 的可能性，而且加速資料檢索的時間，此外，QUERY3 可能比 QUERY2 更快，因爲字串的第一個文字做了特別的搜尋（而且欄 LAST_NAME 有可能是索引），QUERY3 利用了索引的好處。


註解

使用第一個指令，可能傳回所有姓 Stevens 的人，但是 Stevens 不是有不同的拼法嗎？第二個 SQL 指令用姓 Stevens 和所有各種不同拼音的人，第三個 SQL 指令也用以姓為 St 做出發，這是唯一保證您收到所有 Stevens(Stephens) 的方法。

17-5-2 避免 OR 運算子

重寫 SQL 指令，一致使用 IN 運算子代替 OR 運算子，實際上增進資料檢索速度，您的廠商將會告訴您關於 IN 運算子與 OR 運算子，是可以使用計時或檢查執行效率的工具，下列的例子讓您瞭解該如何重寫 SQL 指令，更換 IN 運算子與 OR 運算子：


註解

在第 8 小時“使用運算子來類別資料”可以參考使用 IN 運算子與 OR 運算子的方式。

使用 OR 運算子的查詢：

TYPE
```
SELECT EMP_ID, LAST_NAME, FIRST_NAME
FROM EMPLOYEE_TBL
WHERE CITY = 'INDIANAPOLIS'
   OR CITY = 'BROWNSBURG'
   OR CITY = 'GREENFIELD'
```

相同的查詢使用 IN 運算子：

TYPE
```
SELECT EMP_ID, LAST_NAME, FIRST_NAME
```

```
FROM EMPLOYEE_TBL
WHERE CITY IN ('INDIANAPOLIS', 'BROWNSBURG',
               'GREENFIELD')
```

SQL 指令傳回相同的資料；然而經過測試，您會發現資料檢索的速度可藉由在 IN 與 OR 的更換來加快，如第二個查詢所顯示。

17-5-3 避免 HAVING 子句

HAVING 子句是有用的子句；然而，使用它必須付出代價，使用 HAVING 子句導致 SQL optimizer 做額外的工作，而用去額外的時間，如果可能的話，SQL 指令應該不需要使用 HAVING 子句。

17-5-4 避免大量分類運算

大的分類運算意謂 ORDER BY、GROUP BY 與 HAVING 子句的使用。每當執行分類運算資料的子集必須儲存在記憶體或磁碟（如果有分配的記憶體空間不足），您必須時常分類資料，主要是因爲這些分類運算影響 SQL 指令反應時間。

17-5-5 使用儲存程序（Stored Procedures）

儲存程序應該建立在執行一般基本的 SQL 指令，特別是大的異動或查詢，儲存程序只是簡單的 SQL 指令，可以被編譯並且永久儲存爲在資料庫中可執行格式。

一般而言，當資料庫執行 SQL 指令的時候，資料庫必須在資料庫裡面檢查語法，把指令轉換成可執行的格式（稱爲剖析），指令一旦剖析後，會儲存在記憶體；然而它不是永久儲存的，這表示當其他的運算需要記憶體的時候，指令可以釋放記憶體，對儲存程序的情況而言，SQL 指令的格式總是可執行而且保持在資料庫中，像任何其他的資料庫一樣，直到取消爲止，儲存程序在第 22 小時“進階 SQL 主題”中會有更詳細地討論。

17-6 摘要

您已經學習關連式資料庫裡協調 SQL 指令的意義，您已經了解有二個基本類型的協調：協調資料庫和協調 SQL 指令，都是對資料庫和 SQL 指令有效率且重要的運算，每個是相等地重要，而且沒有協調另一個不能達到最佳化，協調資料庫的責任落在 DBA 的手下，然而協調 SQL 指令則落在每個寫指令的人身上，這本書對後者比較關心。

您已經讀過有關協調 SQL 指令的方法，開始爲指令的可讀性，增加指令的可讀性，不直接地增進執行效率，但是幫助程式設計者對指令的發展和管理。SQL 指令執行效率裡的其中一個主要重點是索引的使用，有時可以使用索引，有時則避免使用他們，當表格被讀取時，執行完整表格掃描，而且不用索引，在完整表格掃瞄裡，表格裡每個列的資料會完全地被讀。協調指令的其他的考慮，例如元件在查詢的安排已討論。在指令裡最重要是，最限制條件放置在 WHERE 子句的位置。爲了增進 SQL 指令的執行效率，了解資料本身非常重要的，資料庫的設計與關係及使用者存取資料庫都是一樣重要的。

像建立表格上的索引，協調 SQL 指令時常包括深入的測試，能找出錯誤與決定方法，沒有固定協調在資料庫裡面的資料或 SQL 指令，所有的資料庫是不同的，如同每家公司所需要的生意是不同的，這些不同，影響了資料在資料庫被取回的方法，您的工作是設計最有效率的 SQL 指令並在資料庫上增加執行效率。

17-6-1 Q&A

Q 藉由我已經學習有關執行效率的觀念，在實際執行效率會得到什麼，如資料檢索的時間，我能看見像預期般的結果嗎？

A 實際上您能從一秒到分鐘、小時或每一天看見執行效率的增進。

Q 我如何能測試我 SQL 指令執行的效率？

A 每個廠商應該有工具或系統檢查執行效率，Oracle7 用這本書裡的 SQL 指令來測試。Oracle 在檢查執行效率方面有使用幾個工具，一些工具被稱爲 Explain Plan、TKPROF 與 Set Timing。檢查您特別的廠商是否有類似 Oracles 的工具。

17-7 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

17-7-1 隨堂測驗

1. 在小表格上使用唯一索引有何益處？
2. 當查詢已經被執行的時候 optimizer 選擇不要使用表格上的索引時會發生什麼？
3. 在 WHERE 子句，最限制的子句是被放置在加入條件之前或之後？

17-7-2 練習題

重寫下列各 SQL 指令，增進他們的執行效率。使用在這所描述的 EMPLOYEE_TBL 和 EMPLOYEE_PAY_TBL：

```
EMPLOYEE_TBL
-------------------------------------------------------
emp_id          varchar2(9)         not null,
last_name       varchar2(15)        not null,
first_name      varchar2(15)        not null,
middle_name     varchar2(15),
address         varchar2(30)        not null,
```

```
city                varchar2(15)        not null,
state               char(2)             not null,
zip                 number(5)           not null,
phone               char(10),
pager               char(10),
constraint emp_pk primary key (emp_id)
```

EMPLOYEE_PAY_TBL

```
emp_id              varchar2(9)         not null,
position            varchar2(15)        not null,
date_hire           date,
pay_rate            number(4,2)         not null,
date_last_raise     date,
salary              number(8,2),
bonus               number(8,2),
  constraint emp_fk foreign key (emp_id)
  references employee_tbl (emp_id)
```


a.

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME,
          PHONE
   FROM EMPLOYEE_TBL
   WHERE SUBSTR(PHONE, 1, 3) = '317' OR
         SUBSTR(PHONE, 1, 3) = '812' OR
         SUBSTR(PHONE, 1, 3) = '765'
```

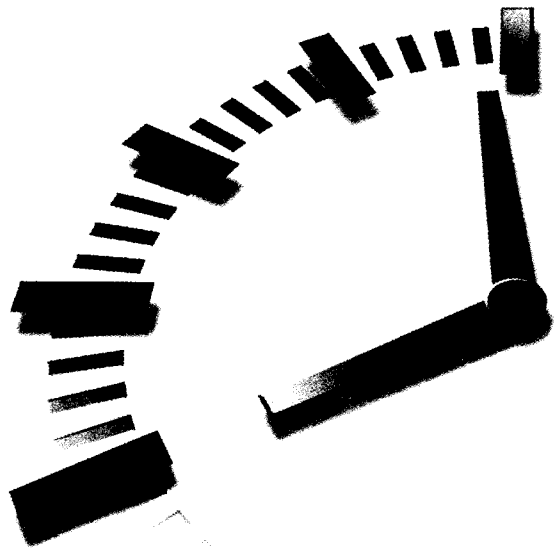
b.

```
SELECT LAST_NAME, FIRST_NAME
   FROM EMPLOYEE_TBL
   WHERE LAST_NAME LIKE '%ALL%
```

c.

```
SELECT E.EMP_ID, E.LAST_NAME, E.FIRST_NAME,
          EP.SALARY
   FROM EMPLOYEE_TBL E,
   EMPLOYEE_PAY_TBL EP
```

```
WHERE LAST_NAME LIKE 'S%'
AND E.EMP_ID = EP.EMP_ID
```



第 VI 單元

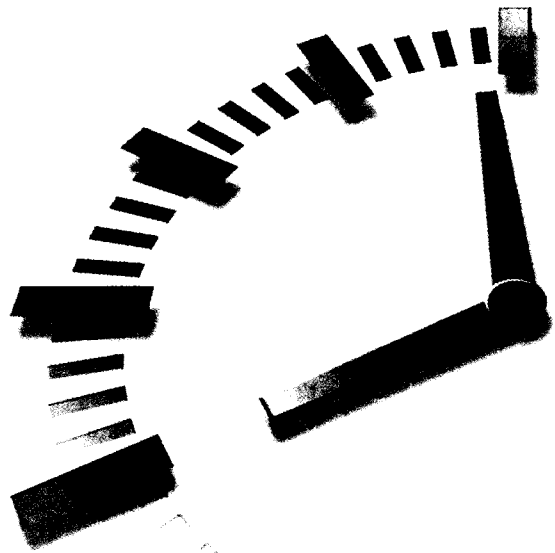
使用 SQL 管理使用者與安全

第 18 小時　管理資料庫使用者

第 19 小時　資料庫安全管理

第 11 單元

使用 SQL 管理與操作資安



第 18 小時

管理資料庫使用者

在這個小時，您將學習關連式資料庫最基本目的：管理資料庫使用者，您將會學習在 SQL 建立使用者、使用者安全性、使用者與資料庫結構、使用者敘述、使用者屬性和使用者使用工具等觀念。

本小時的重點包括：

- 使用者的類型
- 使用者的管理
- 在資料庫的使用者
- 使用者與資料庫結構
- 使用者期間
- 改變使用者屬性
- 使用者敘述

- 從資料庫取消使用者
- 使用者所利用的工具


註解

SQL 標準所提及的資料庫使用者識別為授權識別號（authID）。在大多數的主要廠商，authID 只是被視為使用者，在這本書提及的授權識別號如同使用者，資料庫使用者或資料庫使用者帳戶，SQL 標準陳述授權識別號是讓系統知道資料庫使用者的名字。

18-1 使用者的理由

使用者是爲了設計、建立、完成和維持資料庫的理由而使用資料庫，當設計資料庫時，使用者的需求必須考慮進去，因爲資料庫的最後目標是讓使用者可以使用資料庫，因此使用者才會使用資料庫而且在許多地方能自行開發。

通常使用者認爲，如果沒有使用資料庫，資料庫就不會損壞，雖然這個說法與事實相差不遠，然而資料庫的建立是保存資料，讓使用者在他們每日的工作都能順利的使用到。

雖然管理使用者通常是資料庫管理員的任務，但是其他的人在使用者的管理程序裡也扮演部份角色，使用者的管理在關連式資料庫是很重要的，雖然不同的廠商，管理方式不同，但透過 SQL 的觀念與指令的使用能有大概的瞭解。

18-1-1 使用者的類型

有幾種類型的資料庫使用者：

- 資料登錄員
- 程式設計者

- 系統工程師
- 資料庫管理員
- 系統分析員
- 開發者
- 測試者
- 管理者
- 一般使用者

每個類型的使用者有它的工作功能（和問題），所有的功能（和問題）與他們每日的工作有很大的關係，此外，每個類型的使用者在資料庫裡有不同層次的權利。

18-1-2 誰管理使用者？

公司裡的管理員應負責管理使用者每日的使用，然而，資料庫管理員或被分派到的人，就應該負責管理資料庫裡面的使用者。

資料庫管理員通常管理來自資料庫所建立的使用者帳戶、角色、權限、描述和取消使用者帳戶，因爲它對整個環境有很大的影響，所以一些公司有專門的使用者協助處理管理資料庫的安全。

安全管理員，如果一旦被分配，通常只是處理一些紙上作業，而資料庫管理員必須瞭解使用者的工作需求，讓資料庫管理員知道何時使用者不再需要對資料庫執行存取的工作。

系統分析員或系統管理員，通常對作業系統安全必須付責，要建立使用者而且適當的分配權限，同樣地安全管理員也可能協助系統分析員做一些權限控管的事。

18-1-3 那些使用者該放在資料庫內

使用者應該給予必需的角色，而且給與必需的權限完成他的工作，使用者不應該有超過自己本身工作範圍的權限，而存取資料庫時所有設定使用者帳戶和權限的理由是爲了保護資料。如果使用者錯誤的存取資料，即使是無心的也可能損害或遺失資料，當使用者不再需要存取資料庫的時候，該帳戶應該從資料庫移除，或是讓使用者失效，無法存取資料庫。

所有使用者在資料庫裡有他們應該使用的範圍；在某些方面可能比其他人要付較多的責任，資料庫使用者像人身體的每一部份（至少那是它應該的方式），一起工作完成一些目標。

18-1-4 使用者和資料庫結構有何不同？

與資料庫使用者帳戶有關的稱爲資料庫結構，資料庫結構是資料庫使用者擁有的一組資料庫物件，這個資料庫使用者叫做資料庫結構擁有者，一般的資料庫使用者和資料庫結構擁有者之間的不同，是資料庫結構擁有者擁有在資料庫裡面的物件，然而大多數的使用者不擁有物件，大多數的使用者是因爲要存取資料庫，而包含在其他的資料庫結構的資料，所以才有資料庫帳戶。

18-2 管理程序

穩定的使用者管理系統對任何資料庫系統裡的資料安全性是很重要的，使用者管理系統開始於新的使用者啓動，就立即被監督，如誰應該開始請求存取，然後經過公司主管當局簽核，這是需要時間的，如果被管理人員接受，就傳到安全管理員或資料庫管理員，安全管理員或資料庫管理員就採取行動，好的通知程序是必需的；監督者和使用者必須被通知使用者帳戶已經被建立，而且已經允許對資料庫存取，使用者帳戶密碼應該是只給使用者，使用者在開始的登錄後，應該立刻改變資料庫的密碼。


註解

您必須檢查廠商有關使用者的建立，當建立和管理使用者的時候，也必須參照公司所規定的政策和程序，下列章節將比較在 Oracle、Sybase 和 Microsoft SQL Server 之間如何建立使用者。

18-2-1 建立使用者

資料庫使用者的建立包括在資料庫裡面使用 SQL 類型的指令，沒有爲建立 SQL 裡的資料庫使用者而定的固定標準指令；每個廠商有不同的方法，一些廠商有相似的指令，當其他的語法可能改變的時候，不管廠商如何基本的觀念是相同的。

當資料庫管理員或被分配爲安全管理員收到使用者帳戶請求的時候，應該分析所請求者的資料，包括因在公司職務的差異有特別的需求，這些訊息是用來建立使用者身份證。

應該包括在內的項目是社會福利編號、全名、位址、電話號碼、辦公室或部門名字、分配資料庫及有時被提議的使用者身份證。

下列的段落，顯示二個不同廠商之間比較的建立使用者例子。

在 Oracle 裡建立使用者

建立 Oracle 資料庫使用者帳戶的步驟：

1. 建立資料庫使用者帳戶與相關預設值。
2. 給予使用者帳戶適當的權限。

建立使用者：

```
CREATE USER USER_ID
IDENTIFIED BY PASSWORD
[ DEFAULT TABLESPACE TABLESPACE_NAME ]
[ TEMPORARY TABLESPACE TABLESPACE_NAME ]
```

```
[ QUOTA (INTEGER (K | M) | UNLIMITED) ON TABLESPACE_NAME ]
[ PROFILE PROFILE_TYPE ]
```


註解

早先的語法為建立使用者，能用來把使用者加入 Oracle 資料庫，和一些其他主要的關連式資料庫廠商相同。

如果您不用 Oracle，不必瞭解語法裡的一些選項，表格空間是合乎邏輯的區域也包含了資料庫物件，例如表格與索引中，DEFAULT TABLESPACE 是使用者特別建立在表格空間的物件，TEMPORARY TABLESPACE 是用來做表格空間分類，讓使用者執行查詢運算（表格加入、ORDER BY、GROUP BY）。QUOTA 是空間的限度，放置使用者存取特別表格的空間上限，PROFILE 是特別資料庫的敘述，已經指定給使用者。

得到使用者帳戶權限：

```
GRANT PRIV1 [ , PRIV2, ... ] TO USERNAME | ROLE [, USERNAME ]
```

GRANT 指令能取得一個或較多的權限給相同指令裡同一個或其他的使用者。

在 Sybase 和 Microsoft SQL Server 裡建立使用者

下列步驟爲 Sybase 和 Microsoft SQL Server 建立資料庫使用者的帳戶指引：

1. 建立 SQL SERVER 資料庫使用者帳戶，而且為使用者分配密碼和預設資料庫。
2. 增加使用者到適當的資料庫。
3. 給予使用者帳戶適當的權限。

建立使用者帳戶：

```
SP_ADDLOGIN USER_ID ,PASSWORD, DEFAULT_DATABASE ]
```

增加使用者到資料庫：

```
SP_ADDUSER USER_ID [, NAME_IN_DB [, GRPNAME ] ]
```

給予使用者帳戶的權限：

```
GRANT PRIV1 [ , PRIV2, ... ] TO USER_ID
```


註解

在關連式資料庫裡面，權限的討論在第 19 小時“資料庫安全管理”有更進一步地說明。

建立資料庫結構

資料庫結構經由 CREATE SCHEMA 建立。

語法是：

```
CREATE SCHEMA [ SCHEMA_NAME ] [ USER_ID ]
              [ DEFAULT CHARACTER SET CHARACTER_SET ]
              [ SCHEMA_ELEMENT_LIST ]
```


例子如下：

```
CREATE SCHEMA USER1
CREATE TABLE TBL1
  (COLUMN1     DATATYPE     [NOT NULL],
   COLUMN2     DATATYPE     [NOT NULL]...)
CREATE TABLE TBL2
  (COLUMN1     DATATYPE     [NOT NULL],
   COLUMN2     DATATYPE     [NOT NULL]...)
GRANT SELECT ON TBL1 TO USER2
GRANT SELECT ON TBL2 TO USER2
[ OTHER DDL COMMANDS ... ]
```

以下是廠商的資料庫使用 CREATE SCHEMA 指令的應用程式：

TYPE

```
CREATE SCHEMA AUTHORIZATION USER1
CREATE TABLE EMP
  (ID       NUMBER          NOT NULL,
   NAME     VARCHAR2(10)    NOT NULL)
CREATE TABLE CUST
```

```
  (ID       NUMBER          NOT NULL,
   NAME     VARCHAR2(10)    NOT NULL)
GRANT SELECT ON TBL1 TO USER2
GRANT SELECT ON TBL2 TO USER2
/
```

Output
```
Schema created.
```

AUTHORIZATION 關鍵字被加到 CREATE SCHEMA 指令。這個例子在 Oracle 資料庫被執行。如同您也已經看到早先的例子，廠商因語法指令的不同，使用方法也不同。


註解

一些廠商可能不支援 CREATE SCHEMA 指令，然而，當使用者建立物件的時候，資料庫結構可能暗地裡建立，CREATE SCHEMA 指令是完成這個目標的單一步驟。物件已經被使用者建立，使用者可給予其他使用者對存取物件的權限。

18-2-2 取消資料庫結構

資料庫結構可使用 DROP SCHEMA 從資料庫移除，當取消資料庫結構的時候，有二個必須考慮的選擇項，首先，RESTRICT 選擇項，指定使用 RESTRICT 時，如果物件存在資料庫結構，會發生錯誤。第二個選擇項是 CASCADE，如果任何物件存在現有的資料庫結構時，必須使用 CASCADE 選擇項。請記得，當您取消資料庫結構的時候，您也取消與那個資料庫結構所有關連的資料庫物件。

```
DROP SCHEMA SCHEMA_NAME { RESTRICT | CASCADE }
```


註解

因為物件如表格，可能使用 DROP TABLE 指令而取消，所以資料庫結構裡的物件可能消失，一些廠商可能用程序或指令取消使用者，使用者也可以用來取消資料庫結構。如果在您的廠商 DROP SCHEMA 指令是不可以用，您能藉由除去擁有資料庫結構的使用者，來除去資料庫結構物件。

18-2-3 變更使用者

管理使用者非常重要的部份，是在建立使用者後改變使用者的屬性。如果使用者帳戶從不升職，不離開公司或如果新職員增加數最少，對資料庫管理員來說將會是比較簡單的工作；但在真實的世界，人事和使用者責任的異動是存在的，使用者管理的真實面和重要因素就是，幾乎每個人都會改變工作或工作責任，因此，使用者在資料庫必須調整適合使用者的權限。

廠商改變使用者目前狀態的例子：

對於 Oracle：

```
ALTER USER USER_ID [ IDENTIFIED BY PASSWORD | EXTERNALLY ]
[ DEFAULT TABLESPACE TABLESPACE_NAME ]
[ TEMPORARY TABLESPACE TABLESPACE_NAME ]
[ QUOTA  INTEGER K|M |UNLIMITED ON TABLESPACE_NAME ]
[ PROFILE PROFILE_NAME ]
[ DEFAULT ROLE ROLE1 [, ROLE2 ] | ALL
[ EXCEPT ROLE1 [, ROLE2 | NONE ] ]
```


多數使用者的屬性可用這個語法進行改變，不幸地，並不是所有的廠商提供簡單的處理指令，方便變更資料庫使用者，一些廠商也提供使用者 GUI 工具來建立、更改與移除。


註解

您必須確定廠商改變使用者檢查的正確語法，這裡顯示 Oracle 改變使用者語法，在大多數的主要廠商，有用來改變或變更使用者角色、權限、屬性和密碼的工具。


註解

使用者能改變已建立的密碼，您必須檢查您的廠商有關正確的語法或使用其他工具重新設定密碼。Oracle 使用典型的 ALTER USER 指令。

18-2-4 使用者交談期

使用者資料庫交談期，是從使用者開始登錄資料庫的時間到登出的時間，當使用者登入資料庫（使用者交談期），使用者可能執行各種不同的動作，例如查詢和異動。

當使用者從用戶端連到伺服器使用 CONNECT 指令的時候，SQL 交談期就開始，直到切斷連接，在建立連接和交談期開始，能處理的任何數目的異動和啓動執行，直到資料庫使用者結束交談期。

使用者可以明確地從資料庫連接和終止，使用下列各項的指令，開始與結束 SQL 交談期：

```
CONNECT TO DEFAULT | STRING1 [ AS STRING2 ] [ USER STRING3 ]

DISCONNECT DEFAULT | CURRENT | ALL | STRING

SET CONNECTION DEFAULT | STRING
```

註解

請記得不同語法存在不同廠商之間，除此之外，大多數的資料庫使用者不用手動的方式連上資料庫，大多數的使用者藉著廠商或協力廠商所提供的工具，來提供使用者帳號和密碼藉以存取資料庫，開始讓資料庫與使用者建立交談期。

使用者交談期可以被資料庫管理員或其他的人，對使用者活動有興趣的人監視，當監視使用者的時候，有關使用者交談期與使用者帳戶也被監視。直到使用者最後結束與資料庫的連接，如同主機作業系統上的程序。

18-2-5 移除使用者存取

將使用者從資料庫移開或不許使用者存取，是能藉著一些簡單指令來完成。再一次，在不同的廠商之中的變化是很多的，因此您必須

檢查您的廠商或使用相關工具的語法，確定完成移除使用者或讓使用者停止存取資料庫。

移除使用者存取資料庫的方法：

- 改變使用者密碼。
- 取消來自資料庫的使用者帳戶。
- 撤銷早先允許使用者的權限。

一些廠商使用 DROP 指令，以取消資料庫的使用者。

```
DROP USER USER_ID [ CASCADE ]
```

在許多廠商 REVOKE 指令與 GRANT 指令的用法相似，取消對使用者已經給予的權限，一些廠商對這指令所用的語法如下：

```
REVOKE PRIV1 [ ,PRIV2, ... ] FROM USERNAME
```


18-3 資料庫使用者所利用的工具

一些人說您不需要知道 SQL 如何執行資料庫查詢，在某方面，他們可能是正確的；然而，知道 SQL 如何執行資料庫查詢，可以幫助資料庫查詢，即使使用圖型的使用者介面（GUI）爲工具也一樣。使用 GUI 工具是很好的，而且應該在需要的時候使用，對於了解資料如何處理有幫助，所以您能利用這些使用者的工具達到最大效益。

有許多 GUI 工具能藉由瀏覽視窗、產生報表和選擇選項幫助資料庫使用者自動產生 SQL 編碼。產生表格讓使用者能建立、更新、插入或刪除來自資料庫的資料，有工具轉換資料進入圖表、有用來檢測資料庫執行效果的資料庫執行工具，和允許到遠端的資料庫連接的工具，資料庫廠商提供一些工具，同時協力廠商也提供一些工具給廠商。

18-4 摘要

所有的資料庫都有使用者，不論是一個或數千個，資料庫是爲使用者而定的，有三個基本管理使用者的步驟，首先，必須建立資料庫使用者帳戶，其次，必須取得權限給使用者，讓使用者能在資料庫裡面執行必須的任務。最後，使用者帳戶必須從資料庫移除或取消在資料庫裡面使用者的權限。

另外提到管理使用者的概念與目的，因爲大多數廠商在資料庫管理使用者不一致，所以在這裡避免提到太細的地方，然而，由於它和 SQL 的關係，所以討論使用者管理是很重要的。多數指令尚未定義管理使用者，但在 ANSI 標準有明細的討論，因爲基本觀念是相同的。

18-4-1 Q&A

Q 有增加使用者到資料庫的標準 SQL 語法嗎？

A 一些指令和觀念是由 ANSI 提供，但是每個廠商和每家公司，用來建立或增加到資料庫的使用者都有它自己的指令與工具和規則。

Q 能暫時將使用者取消，而身份完全不用從資料庫移開嗎？

A 是的，使用者存取能暫時被中止，只是變更那些使用者密碼或藉由撤銷使用者連接到資料庫的權限。使用者帳戶的功能藉由變更和發行新的使用者的密碼，讓他復原及取得使用者權限。

Q 使用者能改變自己的密碼嗎？

A 是的，並在大多數的主要廠商都有提供，在資料庫使用者建立或增加時，密碼通常是給使用者而且必須儘快更改密碼成爲他所選擇的密碼，一旦完成，甚至資料庫管理員也不知道使用者密碼。

18-5 綜合練習

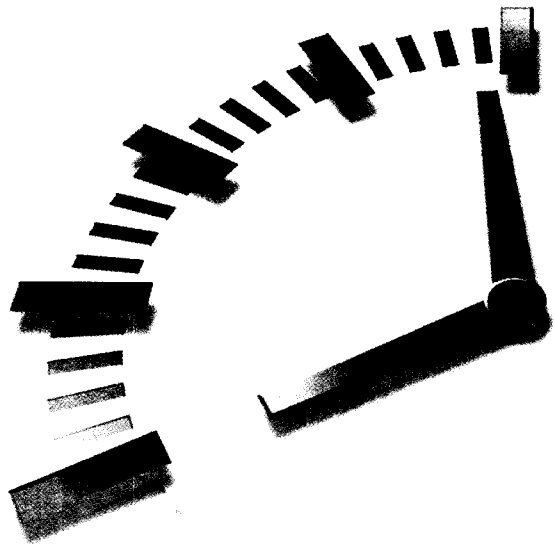
下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

18-5-1 隨堂測驗

1. 什麼指令用來建立交談期？
2. 那一個選擇項必須用來取消資料庫結構，而 仍然包含資料庫的資料庫結構物件？
3. 什麼指令用來除去資料庫權限?
4. 什麼指令建立表格、檢視和權限？

18-5-2 練習題

描述或列出允許新職員資料庫存取的步驟。



第 19 小時

資料庫安全管理

在這個小時，您將學習基本關連式資料庫的安全管理，與學習使用相關的 SQL 指令，每個主要廠商在安全命令語法上也不一致，但是所有與關連式資料庫安全有關的，都在 ANSI 標準中討論，您必須檢查您的廠商有關安全指令的語法，與特別在安全方面的說明。

本小時的重點包括：

- 資料庫安全管理
- 安全管理 vs. 使用者管理
- 資料庫系統的特權
- 資料庫物件的特權
- 取得特權給使用者
- 撤銷使用者的特權
- 資料庫的安全特色

19-1 資料庫安全管理是什麼？

資料庫安全只是保護資料，免於未經認可使用的程序。未經認可的使用，包括只有對資料庫部分的讀取權限，而不是所有的資料庫，這也保護包括對未經認可的連接和散佈的監視行爲。在資料庫有許多不同的使用者等級，從資料庫建立者、對維持資料庫有責任的人（例如 DBA）、資料庫程式設計者和使用者，使用者雖然大多數是有限制的存取，卻是資料庫存在的使用者，每個使用者對資料庫的存取有不同的層次，而且應該限制到針對他們的工作需要，得到有限的特權，執行最少的工作。

19-2 安全管理和使用者管理有何不同？

您可能覺得很奇怪，在使用者管理和資料庫安全之間有什麼不同，畢竟，上一個小時所討論的管理使用者，也涵蓋安全管理，雖然管理使用者和資料庫安全是有明確地關係，每個有他們自己的目的，而且一起工作達成確保資料庫安全的存取。

一個計畫與維護好的使用者管理程式，是必須跟隨完整的資料庫安全管理一起，使用者分配到使用者帳戶和密碼，讓他們能對資料庫存取資料，在資料庫裡面的使用者帳戶應該儲存著使用者的資料，如他們真實的名字、辦公室和部門、使用者的工作、電話號碼與分機和使用者存取資料庫的名稱，資料庫使用者帳戶的密碼由 DBA 或安全管理員分配後，應該立刻變更。

安全管理必需考量更多；舉例來說，如果使用者不再需要他們所取得的某種特權，那些特權應該被取消，如果使用者不再需要存取資料庫，那麼使用者帳戶應該從資料庫被取消。

通常，管理使用者是建立使用者帳戶、移除使用者帳戶和追蹤使用者在資料裡面的動作程序，資料庫安全則更進一步，藉由取得對特

定資料庫的特權，撤銷對使用者的特權和爲了要保護資料庫的其他部份，例如資料庫底下的檔案等。


註解

因為這是有關 SQL 的書，不是使用資料庫的書，比較著重資料庫特權上，然而，您必須知道資料庫安全管理是有其他重點，例如保護資料庫下面的檔案，它含有相當重要的訊息，如資料庫所散佈的特權，在高階的資料庫安全管理變成更複雜，在關連式資料庫廠商之間都不一致。

19-3 特權是什麼？

特權它本身是用來存取在資料庫裡面物件的授權層次，處理資料庫裡的資料與執行在資料庫裡面的各種不同的管理功能，特權經由 GRANT 指令所發行與用 REVOKE 指令所取消。

只是因爲使用者能連接到資料庫，並不意謂使用者能存取在資料庫裡面的資料，要在資料庫裡面，對資料存取必須有這些特權。有二種類型的特權：

1. 系統特權（System Privileges）
2. 物件特權（Object Privileges）

19-3-1 系統特權

系統特權是允許資料庫使用者，在資料庫裡面執行管理的動作，例如下列各項：建立資料庫、取消資料庫、建立使用者帳戶、取消使用者、取消與改變資料庫物件、改變物件的狀態、改變資料庫的狀態和如果不小心使用，其他的動作對資料庫造成的嚴重傷害。

系統特權在不同的關連式資料庫廠商之中有很大的不同，所以必須檢查您的廠商，所有可用的系統特權與他們的正確用法。

在 Sybase，一些常用的系統特權：

CREATE DATABASE	建立資料庫
CREATE DEFAULT	建立預設值
CREATE PROCEDURE	建立程序
CREATE RULE	建立規則
CREATE VIEW	建立檢視
DUMP DATABASE	取消資料庫
DUMP TRANSACTION	取消處理
EXECUTE	執行

在 Oracle，一些常用的系統特權：

CREATE TABLE	建立表格
CREATE ANY TABLE	建立任何表格
ALTER ANY TABLE	改變任何表格
DROP TABLE	取消表格
CREATE USER	建立使用者
DROP USER	取消使用者
ALTER USER	改變使用者
ALTER DATABASE	改變資料庫
ALTER SYSTEM	改變系統
BACKUP ANY TABLE	備份任何表格
SELECT ANY TABLE	選擇任何表格

19-3-2 物件特權

物件特權在物件上的授權，意謂您爲了要執行資料庫物件上的某運算，必須取得適當的特權。舉例來說，爲了要選擇來自其他使用者表格的資料，首先使用者必須授與這項特權，使用者取得物件特權，是藉由資料庫裡的物件擁有者，請記得這個擁有者，也稱爲資料庫結構擁有者。

ANSI 標準所謂的特權包括下列的物件特權：

USAGE　授予使用特定的範圍

SELECT　允許存取特定的表格

INSERT(column_name)　允許資料插入到表格所敘述的特定欄

INSERT　允許資料插入到特定表格的所有欄

UPDATE(column_name)　允許特定表格所敘述的特定欄的資料被更新

UPDATE　允許特定表格的所有欄被更新

REFERENCES(column_name)　允許在完整性的限制裡，參考特定表格的特定欄；這項特權需要所有的完整性限制

REFERENCES　允許所參考特別表格裡所有的欄


註解

物件的擁有者自動取得與物件所有相關的特權，這些特權也已經授與 GRAND OPTION。GRAND OPTION 在稍後這個小時會被討論，在一些 SQL 廠商是一個很好的特點。

這些物件層次特權，應該是爲了要在資料庫結構，被限制取得與存取的特權，這些特權能保護來自相同資料庫裡的使用者，存取另外的資料庫結構裡的物件。

有許多不同的物件特權在不同的廠商之中，而在這沒被列出，能刪除另外使用者的物件，是在許多廠商所常用的物件特權之一，因此一定要檢查您廠商所提供的文件，所有相關物件層次的物件特權。

19-3-3 誰可以給予與取消特權？

資料庫管理者（DBA）通常是執行 GRANT 與 REVOKE 指令的一個人，如果安全管理者存在，也可有特權如此做，在什麼層次上該用 GRANT 或 REVOKE，將由管理者以書面方式說明。

物件的擁有者對物件必須能授與特權給資料庫裡的其他使用者，甚至 DBA 不能夠取得不屬於 DBA 物件的特權，雖然有方法可以做到。

19-4 控制使用者存取

使用者存取資料庫，主要是用使用者帳戶和密碼控制，但是在大多數的主要廠商，對於那樣的存取資料庫是不夠的，建立使用者帳戶只是得到對資料庫的存取和控制存取的第一個步驟。

一旦建立使用者帳戶，爲了讓使用者能在資料庫執行一些工作，如建立表格或選擇表格，資料庫管理者、安全管理員或被指定負責的人，必須能夠對使用者散佈適當的特權，如果您不能夠做任何事，連接資料庫有什麼好處？此外，資料庫結構擁有者在資料庫結構裡，通常需要資料庫使用者存取物件，所以使用者才能做他的工作。

在 SQL 裡有二個指令允許取得資料庫的控制權，包括分配特權和取消特權。

在關連式資料庫裡，二個用在散佈系統與物件特權的指令是：

GRANT 授與
REVOKE 取消

19-4-1 GRANT 指令

GRANT 指令用來取得現存資料庫使用者帳戶的系統層次與物件層次特權。

語法如下：

```
GRANT PRIVILEGE1 [, PRIVILEGE2 ][ ON OBJECT ]
TO USERNAME [ WITH GRANT OPTION ]
```

取得使用者的一項特權如下：

```
GRANT SELECT ON EMPLOYEE_TBL TO USER1;

Grant succeeded.
```

取得使用者多重特權：

```
GRANT SELECT, INSERT ON EMPLOYEE_TBL TO USER1;

Grant succeeded.
```

注意指令裡，取得多重特權給使用者的時候，每個特權必須用逗點分開。

取得多重使用者特權如下：

```
GRANT SELECT, INSERT ON EMPLOYEE_TBL TO USER1, USER2;

Grant succeeded.
```


註解

注意授與成功（Grant succeeded.），表示每個授與指令已完成，這是您在本書的例子中 Oracle 所傳回的訊息。大多數的廠商都會回傳某些訊息，告知動作已經完成，但是訊息可能因廠商的不同而異。

GRANT OPTION

GRANT OPTION 是 GRANT 指令非常有力的選項，當物件擁有者使用 GRANT OPTION 取得物件特權給另外的使用者，新的使用者也給予其他的使用者那個物件的特權，即使使用者實際上沒有擁有那個物件。

例子如下：

```
GRANT SELECT ON EMPLOYEE_TBL TO USER1 WITH GRANT OPTION;

Grant succeeded.
```

19-4-2 REVOKE 指令

REVOKE 指令是對資料庫使用者取消已得到的特權，REVOKE 指令有二個選擇項，RESTRICT 與 CASCADE，當使用 RESTRICT 選項時，如果明確地敘述 REVOKE 指令，且沒有其他拋棄特權（Abandoned

privileges）的使用者，REVOKE 才會成功的執行；CASCADE 選項將會取消使用者留給其他使用者的特權。換句話說，如果物件擁有者用 GRANT OPTION 取得 user1 特權，user1 用 GRANT OPTION 取得 user2 特權，擁有者取消給 user1 的特權，CASCADE 也除去 user2 的特權。

NEW TERM 被拋棄特權（Abandoned privileges）是用 GRANT OPTION 取得特權的使用者，所留下的特權。

語法如下：

```
REVOKE PRIVILEGE1 [, PRIVILEGE2 ] [ GRANT OPTION FOR ] ON OBJECT
FROM USER { RESTRICT | CASCADE }
```

例子如下：

```
REVOKE INSERT ON EMPLOYEE_TBL FROM USER1;

Revoke succeeded.
```

19-4-3 PUBLIC 資料庫帳戶

PUBLIC 資料庫使用者帳戶是表示資料庫裡所有使用者的資料庫帳戶，所有使用者是公用帳戶的一部份，如果特權被公用帳戶得到，那麼所有資料庫使用者就有那特權，同樣地，如果取消公用帳戶的特權，那麼所有資料庫使用者的特權也被取消，除非那項特權被特定的使用者取得。

例子如下：

```
GRANT SELECT ON EMPLOYEE_TBL TO PUBLIC;

Grant succeeded.
```

當取得特權給公用（PUBLIC）的時候要非常小心，因為所有的資料庫使用者將會獲得那些特權。

19-4-4 群組特權（Group Previleges）

一些廠商在資料庫有設群組特權，這些群組權限被不同的名稱所參考，有群組特權對一群使用者設定 GRANT 與 REVOKE 特權的方式變得更簡單，舉例來說，如果群組有十項特權，那麼群組能替使用者取得所有十項特權。

SQLBase有設群組特權稱爲授權層次(authority levels)，在Oracle裡的群組特權叫做角色（Roles），SQLBase 和 Oracle 兩者都用有設群組特權在他們的產品裡：

```
CONNECT
RESOURCE
DBA
```

CONNECT 群組允許使用者連接到資料庫，使用者可執行存取任何資料庫裡的物件運算。

RESOURCE 群組允許使用者建立物件、取消他們自己的物件、授與物件等等。

DBA 群組允許使用者執行在資料庫裡面的任何的功能，使用者能存取任何資料庫物件與用這個群組執行任何的運算。

取得群組特權的例子如下：

```
GRANT DBA TO USER1;

Grant succeeded.
```


註解

每個廠商在資料庫使用對每個群組給的特權都不一致，如果可用的話，這個優點應該被用來管理資料庫安全。

19-5 摘要

您已經瞭解 SQL 資料庫或關連式資料庫的基本安全管理，您將學習資料庫使用者基本的安全管理，第一個步驟，是建立資料庫使用者，一旦使用者被建立，使用者必須取得對資料庫的特定部份，有某種存取特權。有二種類型的特權；系統特權和物件特權。

系統特權是取得使用者在資料庫裡面執行各種不同的工作，例如連接到資料庫、建立表格、建立使用者與改變資料庫狀態等等，物件特權是取得在資料庫裡面的使用者對特定的物件存取，例如選擇資料或處理特定表格裡的資料。

在 SQL 裡有二個指令：GRANT 指令和 REVOKE 指令，允許使用者來回取得與取消資料庫裡其他使用者的特權，這二個指令用來控制資料庫裡特權的管理，雖然仍有許多關連式資料庫安全管理的考量，但是關於 SQL 語言的基本觀念在這小時都已討論過。

19-5-1 Q&A

Q 如果使用者忘記他的密碼，使用者應該再做什麼，才能對資料庫存取？

A 使用者應該立即找他們的 help desk，help desk 應該能夠重新設定使用者密碼，如果沒有的話，DBA 或安全管理員也能重新設定密碼，一旦密碼被重新設定，使用者應該換成得到的密碼。

Q 如果我要取得 CONNECT 給使用者，我應該做什麼，而且使用者不需要所有 CONNECT 角色所指定的特權？

A 您只是不授與 CONNECT，但是只有那特權是必需的，您應該曾經授與 CONNECT 和使用者不再需要所有的特權，就簡單的從使用者取消 CONNECT 與取得特定需要的特權。

Q 為什麼讓新的使用者改變密碼很重要？

A 密碼是在使用者身份證被建立後所分配的，沒有人、甚至 DBA 或管理人員知道使用者密碼，密碼應該好好保存，避免另外一個使用者用另外使用者的帳戶登錄到資料庫內。

19-6 綜合練習

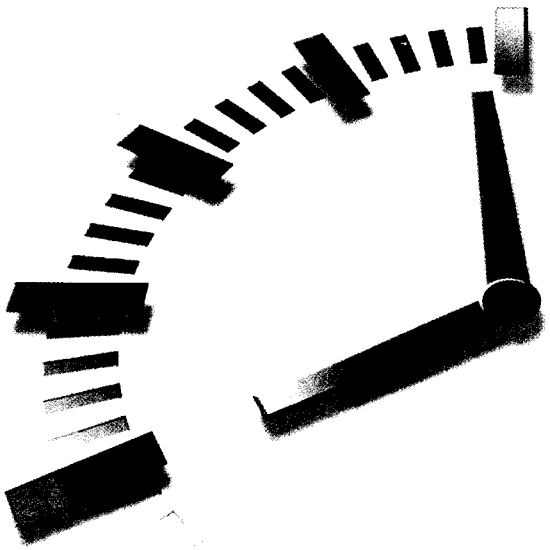
下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

19-6-1 隨堂測驗

1. 什麼選項使用者必須取得，才能授予另外一個使用者不屬於使用者所擁有的物件特權？
2. 當特權被 PUBLIC 取到，所有資料庫的使用者可獲得那些特權嗎？或只是所敘述的特定使用者而已？
3. 什麼特權需要檢視特定的表格資料？
4. 什麼樣的類型有 SELECT 的權限？

19-6-2 練習題

1. 寫個程式讓使用者 rplew 取得您擁有表格 employee_tbl 的存取權限，它應該允許 rplew 有特權給相同表格上的另外一個使用者。
2. 寫個程式取消練習題 1 裡，兩個使用者對資料庫的連接。
3. 寫個程式允許 rplew 選擇、插入與更新 employee_tbl 表格。



第 VII 單元

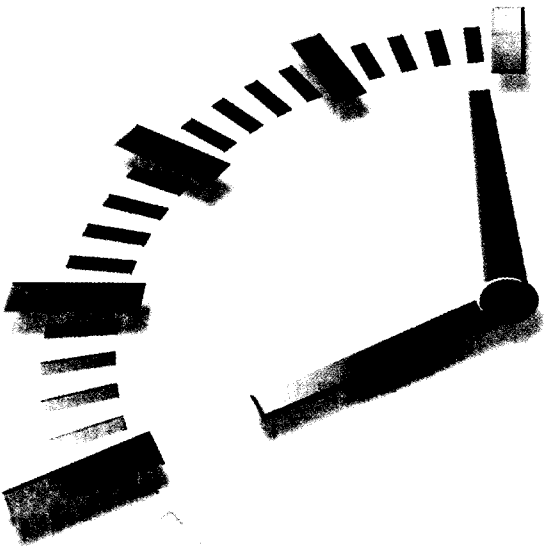
資料庫結構摘要

第 20 小時　建立、使用檢視與同義字

第 21 小時　系統型錄運作

第Ⅲ單元

資料庫結構設計



第 20 小時

建立、使用檢視與同義字

在這個小時內，您將學習有關執行效率，和如何建立與取消檢視，如何使用安全的檢視和如何提供單純的資料檢索與報告給使用者，您也將會看到有關同義字的討論。

本小時的重點包括：

- 檢視是什麼？
- 檢視如何使用？
- 檢視和安全
- 檢視的儲存
- 建立檢視
- 加入檢視
- 檢視裡的資料處理
- 同義字是什麼？

- 管理同義字
- 同義字的建立
- 取消同義字

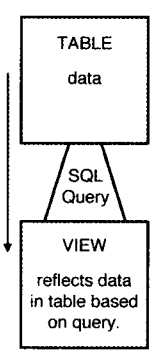
20-1 檢視是什麼？

檢視是虛擬的表格，檢視就像在看個表格，而且對使用者而言就像表格一樣，檢視實際上是由預設的查詢形式所組成的表格，舉例來說，檢視能從 EMPLOYEE_TBL 表格被建立，只包含員工名稱與定址，代替 EMPLOYEE_TBL 表格裡的所有欄位，檢視包含表格的所有列或選擇列，檢視能從一個或許多個表格中建立。

NEW TERM 檢視是預先設定儲存在資料庫的查詢，而且外表像平常的表格，並且像表格一樣存取，但是不需要實際的儲存空間。

當建立檢視的時候，SELECT 指令實際上跟著資料庫跑，定義資料庫檢視，SELECT 指令用來定義檢視，可能只是從表格選個欄名，或是明確地使用各種不同的功能和計算處理使用者想看的資料。看圖 20.1 裡的檢視說明。

■ 20.1. 檢視



檢視考慮到資料庫物件，雖然檢視沒有暫儲存空間。檢視和表格之間的主要不同是表格裡的資料實際儲存在硬碟裡，而因爲實際是讀取來自表格的資料，所以檢視不需要實際的儲存空間。

檢視在使用方式與表格在資料庫使用方式是相同的，意謂著資料能從檢視選擇，雖然有些限制，但正如從表格選擇一樣。資料在檢視也能被處理，下列的章節討論一些常用的檢視使用，與檢視如何儲存在資料庫。


注意

如果取消用來建立檢視的表格時，檢視會變成無法讀取，當對檢視試著去讀的時候，您會收到錯誤訊息。

20-1-1 檢視也能為成一種安全格式

檢視能使用爲資料庫裡的一種安全格式，比方說您有一個稱爲 EMPLOYEE_TBL 的表格。EMPLOYEE_TBL 包括員工名稱、位址、電話號碼、緊急聯絡電話、部門、位置和薪水或每小時的薪水率，您有一些臨時工幫您寫一些報告，需要一個員工名稱、位址和電話號碼的報告，如果您給臨時工存取 EMPLOYEE_TBL，他們能看見您的每一位員工收到多少津貼，這是您不想發生的，爲了避免這些，您只能建立一個檢視，包含必需的訊息：員工名稱、位址、電話號碼、和緊急聯絡電話，然後給臨時工存取檢視的權限來寫報告，而不用存取表格裡一些不必要的欄位。


提示

檢視能用來限制使用者存取表格裡的特別欄位，或符合在 WHERE 子句所定義的特別條件的資料。

20-1-2 檢視能保存資料摘要

檢視能維持資料摘要，如果在表格裡，您有資料報告的摘要，或是表格裡的資料時常更新，而且報告時常被建立，那麼檢視資料的摘要，可是非常好的選擇。

舉例來說，您有個表格，包含每個人的訊息，例如他們所居住的城市、性別，他們的薪水和他們的年齡，您能夠基於表格建立檢視，

顯示每座城市人口的摘要，例如平均年齡，平均薪水、男性總數和女性總數。建立檢視來傳回自基礎表格的訊息，您能用簡單的查詢檢視代替複雜的 SELECT 指令。

建立檢視摘要資料的語法與建立來自單一或多重表格的檢視語法唯一不同在聚合函數的使用，請參考第 9 小時 “資料查詢結果總結 “有關聚合函數的使用。

20-1-3 檢視如何儲存？

檢視只儲存在記憶體，檢視沒有使用到儲存空間與其他的資料庫物件，僅需要儲存空間來儲存檢視本身定義，檢視是檢視建立者或資料庫結構擁有者所擁有的，檢視擁有者自動有那檢視上所有適用的特權，而且能允許其他的使用者取得檢視的特權，就與表格一樣，GRANT 指令與 GRANT OPTION 特權的運作方式與在表格上相同，請參考第 19 小時 “資料庫安全管理 “取的更多資訊。

20-2 建立檢視

建立檢視是使用 CREATE VIEW 指令，檢視能由單一表格、多重表格或另外的檢視上建立，爲了建立檢視，必須依照特定的廠商規定，使用者必須有適當的系統特權。

基本 CREATE VIEW 語法如下：

```
CREATE VIEW VIEW_NAME AS
{SELECT STATEMENT}
```

下列各小部份探討建立使用檢視 CREATE VIEW 指令的不同方法。


註解

在 ANSI SQL 裡沒有 ALTER VIEW 的指令。

20-2-1 從單一表格建立檢視

檢視能從單一表格中建立，WITH CHECK OPTION 在稍後將被討論。

語法如下：

```
CREATE VIEW VIEW_NAME AS
SELECT * | COLUMN1 [, COLUMN2 ]
FROM TABLE_NAME
[ WHERE EXPRESSION1 [, EXPRESSION2 ]]
[ WITH CHECK OPTION ]
[ GROUP BY ]
```

建立檢視最簡單的形式是建在單一表格的全體內容，如下列各項例子；

TYPE
```
create view customers as
select *
from customer_tbl;
```

Output
```
View created.
```

下個例子減少內容，只檢視基礎表格某些特定的欄位：

TYPE
```
create view emp_view as
select last_name, first_name, mid_init
from employee_tbl;
```

Output
```
View Created.
```

下列的例子是爲了要形成檢視裡的欄位，能結合或處理來自基礎表格的欄位，檢視欄在 SELECT 指令是使用別名 - NAME。

TYPE
```
create view names as
select last_name || ',' || first_name ||'' || mid_init name
from employee_tbl;
```

Output
```
View created.
```

現在選擇您建立檢視的所有資料，稱爲 NAME。

TYPE
```
select *
from names;
NAME
```

Output
```
------------------
STEPHENS, TINA D
PLEW, LINDA C
GLASS, BRANDON S
GLASS, JACOB
WALLACE, MARIAH
SPURGEON, TIFFANY
6 rows selected.
```

下列各項例子，表示如何用以下的表格，建立資料摘要的檢視：

TYPE
```
create view city_pay as
select e.city, avg(p.pay_rate) avg_pay
from employee_tbl e,
     employee_pay_tbl p
where e.emp_id = p.emp_id
group by e.city;
```

Output
```
View created.
```

現在，選擇您的摘要檢視:

TYPE
```
select *
from city_pay;
```

Output
```
CITY            AVG_PAY
------------ ----------
GREENWOOD          17.5
INDIANAPOLIS    13.3875
WHITELAND         18.25
3 rows selected.
```

藉由檢視摘要看到，選擇未來可能簡化在檢視下的表格。

20-2-2 從多重表格建立檢視

檢視能藉由在 SELECT 指令中使用 JOIN，從多重表格中建立，WITH CHECK OPTION 在稍後會被討論。語法如下：

```
CREATE VIEW VIEW_NAME AS
SELECT * | COLUMN1 [, COLUMN2 ]
FROM TABLE_NAME1, TABLE_NAME2 [, TABLE_NAME3 ]
WHERE TABLE_NAME1 = TABLE_NAME2
[ AND TABLE_NAME1 = TABLE_NAME 3 ]
[ EXPRESSION1 ][, EXPRESSION2 ]
[ WITH CHECK OPTION ]
[ GROUP BY ]
```

建立來自多重表格檢視的例子：

TYPE
```
create view employee_summary as
select e.emp_id, e.last_name, p.position, p.date_hire,
p.pay_rate
from employee_tbl e,
     employee_pay_tbl p
where e.emp_id = p.emp_id;
```

Output
```
View created.
```

請記得，當選擇來自多重表格資料的時候，那些表格必須在 WHERE 子句中被加入，檢視它本身就是 SELECT 指令的應用；因此，表格在檢視定義被加入，和在一般 SELECT 指令時的應用時相同，有關表格別名的使用，也是爲了要增加多重表格查詢的可讀性。

20-2-3 從檢視建立檢視

檢視能從使用下列的格式從另外檢視中建立：

```
CREATE VIEW2 AS
SELECT * FROM VIEW1
```

別太深入地建立檢視

您可能建立許多層的檢視（檢視內的檢視等等），能建立多深是依廠商而定，唯一的問題是，如何管理建立於其他檢視的檢視，舉例來說，您想建立 view2 基於 view1，然後建立 view3 基於 view2，如果 view1 被取消，對 view2 和 view3 是不好的。下面的訊息是說明這些檢視不再存在，因此最好維持資料庫檢視的易瞭解性，和知道那些檢視依賴其他的物件。見圖 20.2 檢視關連性。


註解

如果建立另外的檢視，就像從基礎表格建立檢視一樣的容易和有效率，應傾向從基礎表格建立檢視。

圖 20.2.
檢視關連性

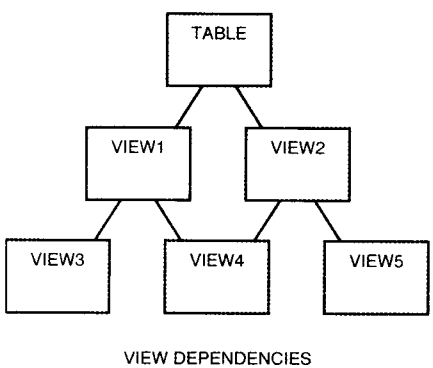


圖 20.2 表示檢視的關係，不只在表格上，也在其他的檢視上，VIEW1 和 VIEW2 依賴 TABLE，VIEW3 依賴 VIEW1，VIEW4 依賴 VIEW1 和 VIEW2，VIEW5 依賴 VIEW2。在這些關係上，能的到下列結果：

- 如果 VIEW1 被取消，那麼 VIEW3 和 VIEW4 就無效。
- 如果 VIEW2 被取消，那麼 VIEW4 和 VIEW5 就無效。
- 如果 TABLE 被取消，那麼沒有檢視是有效的。

20-2-4 WITH CHECK OPTION

WITH CHECK OPTION 是 CREATE VIEW 指令選項，WITH CHECK OPTION 的目的是確定所有的 INSERT 與 UPDATE 符合檢視定義裡的條件，如果他們不符合條件，UPDATE 或 INSERT 會傳回錯誤訊息，WITH CHECK OPTION 有二個選項，CASCADED 和 LOCAL。WITH CHECK OPTION 實際上藉由檢查檢視定義知道參考完整性是無效的。

下列是用 WITH CHECK OPTION 檢查建立檢視的例子：

TYPE
```
create view employee_pagers as
select last_name, first_name, pager
from employee_tbl
where pager is not null
with check option;
```

Output
```
View created.
```

在這狀況，WITH CHECK OPTION 拒絕在檢視欄－PAGER 裡的資料值是 NULL，因爲在檢視裡，PAGER 欄的資料被定義不得有 NULL 值。

試著插入 NULL 值到 PAGER 欄裡：

TYPE
```
insert into employee_pagers
values (SMITH,JOHN,NULL);
```

Output
```
insert into employee_pagers
            *
ERROR at line 1:
ORA-01400: mandatory (NOT NULL) column is missing or NULL during
insert
```

WITH CHECK OPTION 工作。

CASCADED Vs. LOCAL

用檢視建立其他的檢視時，會使用到二個選擇項：CASCADED 和 LOCAL，假定沒有特別敘述的話，CASCADED 是預設值，CASCADED 選項

檢查在基礎表格的更新期間，所有的檢視，所有完整性條件和對第二檢視條件的定義；LOCAL 選項只用來檢查和第二個檢視定義條件的完整性，而不是下面的基礎表格裡，用 CASCADED 選項建立檢視是比較安全的，因爲基礎表格的參考完整性是被保護。

20-2-5 更新檢視

檢視在某些情況可以更新：

- 檢視不能包含加入。
- 檢視不能包含 GROUP BY。
- 檢視不能夠包含 pseudocolumn ROWNUM 的任何參考。
- 檢視不能夠包含任何的群組功能。
- 不能用 DISTINCT 子句。
- WHERE 子句不能包含巢狀表格運算式，那包括在與 FROM 子句所參考到相同表格的表格運算式。

溫習第 14 小時，有關使用 UPDATE 的語法。

20-2-6 插入列進入檢視

資料列可以插入檢視之內，同樣的規定適用於 UPDATE 指令，同時也適用於 INSERT 指令請溫習第 14 小時，有關使用 INSERT 的語法。

20-2-7 刪除從檢視來的列

資料列可能從檢視被刪除，同樣的規定適用於 UPDATE、INSERT 指令，同時也適用於 DELETE 指令，請溫習第 14 小時，有關使用 INSERT 的語法。

20-2-8 在表格和其他的檢視加入檢視

檢視能和表格與其他的檢視一起加入，適用於加入表格到其他表格裡，也應用在檢視裡加入表格與其他的檢視，請溫習第 13 小時，有關表格的加入。

20-2-9 從表格來建立檢視

表格可能從檢視中被建立，正如表格能從另外一個表格（或從另外一個檢視來檢視）被建立。

語法如下列：

```
CREATE TABLE TABLE_NAME AS
SELECT {* | COLUMN1 [, COLUMN2 ]
FROM VIEW_NAME
[ WHERE CONDITION1 [, CONDITION2 ]
[ ORDER BY ]
```

首先，建立基於二個表格的檢視：

TYPE
```
create view active_customers as
select c.*
from customer_tbl c,
     orders_tbl o
where c.cust_id = o.cust_id;
```

Output
```
View created.
```

下一步，建立基於早先建立檢視的表格：

TYPE
```
create table customer_roster_tbl as
select cust_id, cust_name
from active_customers;
```

Output
```
Table created.
```

最後，選擇來自表格的資料，和任何其他的表格相同：

TYPE
```
select *
from customer_roster_tbl;
```

Output
```
CUST_ID    CUST_NAME
---------- ------------------------------
232        LESLIE GLEASON
12         MARYS GIFT SHOP
43         SCHYLERS NOVELTIES
090        WENDY WOLF
287        GAVINS PLACE
432        SCOTTYS MARKET
6 rows selected.
```


註解

請記得在表格和檢視之間，主要的不同是表格包含真實的資料而且佔去儲存空間，然而檢視沒有包含資料而且不需要儲存空間，除了儲存檢視定義（查詢）外。

20-2-10 檢視與 ORDER BY 子句

ORDER BY 子句不能夠用在 CREATE VIEW 指令；然而在用 CREATE VIEW 指令的時候，GROUP BY 子句有相同的效果。


註解

查詢檢視時，在 SELECT 指令裡使用 ORDER BY 子句是比在 CREATE VIEW 指令裡使用 GROUP BY 子句好和簡單。

GROUP BY 子句建立在 CREATE VIEW 令的例子：

TYPE
```
create view names2 as
select last_name || ',' || first_name ||'' || mid_init name
from employee_tbl
group by last_name || ',' || first_name ||'' || mid_init;
```

Output
```
View created.
```

如果您從檢視來選擇所有的資料，資料是依字母順序的排列（因爲用 NAME 群組）。

TYPE
```
select *
from names2;
NAME
```

Output
```
--------------------
GLASS, BRANDON S
GLASS, JACOB
PLEW, LINDA C
SPURGEON, TIFFANY
STEPHENS, TINA D
WALLACE, MARIAH
6 rows selected.
```

20-3 取消檢視

DROP VIEW 指令用來取消來自資料庫的檢視，DROP VIEW 指令有二個選：RESTRICT 與 CASCADE，如果在任何其他的檢視是參考限制條件的時候，用 RESTRICT 選項取消檢視時，DROP VIEW 會傳回錯誤訊息；如果使用 CASCADE 選項，而且參考其他的檢視或是限制條件的時候，DROP VIEW 就會成功，而且下面的檢視或限制條件也會取消。例子如下：

TYPE
```
DROP VIEW NAMES2;
```

Output
```
View dropped.
```


20-4 同義字是什麼?

同義字是表格或檢視另外一個名稱，一般都會建立同義字，所以使用者能避免因爲另外使用者取得其他資料表格或存取表格、檢視。同義字能被建立爲 PUBLIC 或 PRIVATE，PUBLIC 同義字能和任何資料庫

使用者使用：PRIVATE 同義字只能被擁有者和已經被取得特權的使用者使用。


註解

同義字被幾個主要的廠商所使用，同義字不是 ANSI SQL 標準；然而，因為同義字被主要的廠商使用，所以我們簡短地討論，如果您的廠商有提供的話，您必須檢查同義字的使用方法。

20-4-1 管理同義字

同義字不是被資料庫管理員（或另外的指定人）就是被個別使用者處理，因爲有二類型的同義字，PUBLIC 和 PRIVATE（公共和私人），建立不同的同義字，需要不同的系統特權，所有的使用者通常都能建立私人同義字，典型地，只有 DBA 或有特權的資料庫使用者能建立公用同義字，請參照您的特定廠商的文件，有關建立同義字需的特權。

建立同義字

一般建立同義字的語法如下：

```
CREATE [PUBLIC|PRIVATE] SYNONYM SYNONYM_NAME FOR TABLE|VIEW
```

在下列的例子中，您建立的同義字被稱爲 CUST，爲 CUSTOMER_TBL 的簡稱，這從有助於您減輕拼出完整表格名稱的負擔。

TYPE
```
create synonym cust for customer_tbl;
```

Output
```
Synonym created.
```

TYPE
```
select cust_name
from cust;
```

Output
```
CUST_NAME
------------------------------
LESLIE GLEASON
NANCY BUNKER
ANGELA DOBKO
WENDY WOLF
MARYS GIFT SHOP
SCOTTYS MARKET
JASONS AND DALLAS GOODIES
MORGANS CANDIES AND TREATS
SCHYLERS NOVELTIES
GAVINS PLACE
HOLLYS GAMEARAMA
HEATHERS FEATHERS AND THINGS
RAGANS HOBBIES INC
ANDYS CANDIES
RYANS STUFF
15 rows selected.
```

在已經允許您存取的表格去建立同義字，您不必用表格擁有者的名稱得到表格名稱。

TYPE
```
CREATE SYNONYM PRODUCTS_TBL FOR USER1.PRODUCTS_TBL;
```

Output
```
Synonym created.
```

取消同義字

取消同義字像取消其他任何的資料庫物件，一般取消同義字的語法如下：

```
DROP [PUBLIC|PRIVATE] SYNONYM SYNONYM_NAME
```

例子如下：

TYPE
```
DROP SYNONYM CUST;
```

Output
```
Synonym dropped.
```

20-5 摘要

檢視與同義字，在這個小時裡討論 SQL 二個重要的特徵，在許多情況，當關連式資料庫的使用者能使用全部的功能時，這些功能就不被使用。檢視定義爲虛擬表格，像表格一樣可以看和執行的物件；但是不像表格，暫去實際的空間。檢視實際上是定義對表格和資料庫裡其他的檢視做查詢，檢視用來限制使用者所看見的資料，而且簡化並摘要資料，檢視也能從檢視中建立，但是必須小心的使用，不要互相使用檢視太深入，以便遺失管理上的控制，因一些廠商而定，當建立檢視的時候，有各種不同的選項。

本小時也討論到同義字，代表可以表示其他物件的資料庫物件，同義字用來簡化資料庫裡的另一個物件名稱，或藉由您可以存取的物件中，建立同義字縮短物件的名稱，或建立另外個使用者擁有物件的同義字，有二類型的同義字：PUBLIC 和 PRIVATE，PUBLIC 同義字是對所有資料庫的使用者，然而 PRIVATE 同義字對單一使用者而定的，DBA 通常建立 PUBLIC 同義字，而個別的使用者通常建立的 PRIVATE 同義字。

20-5-1 Q&A

Q 檢視怎麼能包含資料但是卻沒有佔去儲存空間?

A 檢視不包含資料，檢視是虛擬的表格或儲存的查詢，在檢視唯一需要的空間，是建立檢視指令，呼叫檢視定義。

Q 如果建立的表格取消，檢視會發生什麼事？

A 檢視會變成無效，因爲檢視下面的資料不再存在。

Q 當建立同義字的時候，同義字是否有限度？

A 這是依廠商而不同，然而大多數的主要廠商的同義字命名，大都遵從適用於資料庫表格和其他物件的相同規則。

20-6 綜合練習

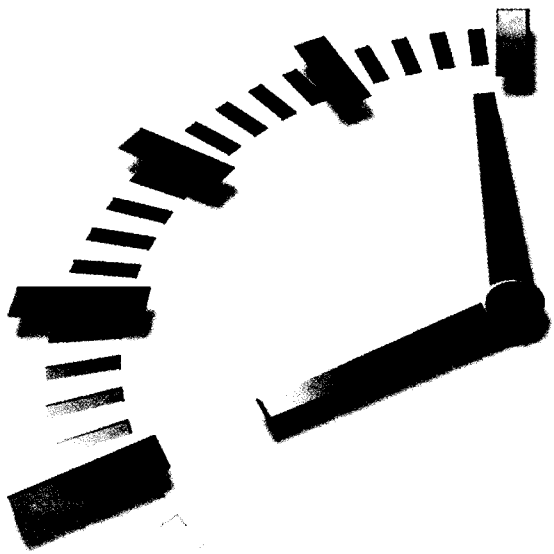
下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

20-6-1 隨堂測驗

1. 一列的資料能從多重表格所建立的檢視中被刪除嗎？
2. 當建立表格時，擁有者自動取得那個表格上的適當特權，當建立檢視的時候，也是一樣嗎？
3. 當建立檢視的時候，哪個子句用來排序資料?
4. 當從檢視建立檢視的時候，檢查限制條件的完整性時，哪個選項能被使用？
5. 您試著取消檢視，但收到錯誤訊息，是因為下面有一個或更多的檢視，您必須做什麼取消檢視？

20-6-2 練習題

1. 編寫建立基於 EMPLOYEE_TBL 表格總內容的檢視指令。
2. 編寫建立檢視摘要，包含 EMPLOYEE_TBL 表格裡的每個城市，平均 Pay Rate 和平均薪水的指令。
3. 編寫取消您在練習 1 和 2 所建立的二個檢視的指令。



第 21 小時

系統型錄運作

在這個小時您將學到系統型錄，在某些廠商裡被認爲是資料庫共用的資料字典，在這個小時結束前，您將會了解系統型錄內容與目的、和查詢系統型錄。找尋您在早先小時所建立資料庫的訊息，每個主要的廠商都有一些形式的系統型錄來儲存資料庫本身的訊息，這章顯示包含在一些不同的系統型錄元件的例子。

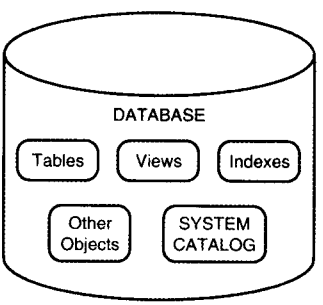
本小時的重點包括：

- 系統型錄是什麼？
- 系統型錄如何建立的？
- 怎樣的資料包含在系統型錄？
- 系統型錄表格的例子
- 查詢系統型錄
- 更新系統型錄

21-1 系統型錄是什麼？

系統型錄是一個收集表格與檢視有關資料庫的重要訊息，系統型錄是讓每個資料庫可以使用的，系統型錄裡的訊息定義資料庫的結構，舉例來說資料庫裡所有表格的 DDL（資料字典語言）儲存在系統型錄，見在圖 21.1，顯示資料庫裡的系統型錄。

圖 21.1.
系統型錄



在如圖 21.1 所視，資料庫的系統型錄實際上是資料庫的一部份，在資料庫裡面是表格、索引和檢視等物件，系統型錄基本上是一群物件包含定義資料庫裡其他物件的訊息、資料庫本身結構和各種其他不同的重要訊息。

您的廠商系統型錄可能被區分爲合乎邏輯的群組物件，讓不只是資料庫管理員可以存取表格，而是任何其他的資料庫使用者都可以存取。舉例來說，使用者可能需要檢視它已經有特權的資料庫，但是並不需要知道有關資料庫的內在結構或程序等，使用者通常查詢系統型錄來獲得使用者所擁有的物件和特權，然而 DBA 需要能夠查詢在資料庫裡任何的結構或事件，在一些廠商中，只有資料庫管理員能存取系統型錄物件。

系統型錄對資料庫管理員或任何其他想知道資料庫結構的資料庫使用者是很重要的，系統型錄允許被保留，不只對資料庫管理員和使用者，但是也對同時資料庫伺服器本身。


註解

每個廠商對系統型錄表格和檢視有它自己的命名系統，命名不是很重要，學習查詢系統型錄才是重要的，正如它包含的是如何與該在哪裡取回訊息。

21-2 系統型錄如何建立？

系統型錄是當資料庫建立時就自動地建立，就是當資料庫建立後，資料庫管理員立刻跟著建立。舉例來說，執行一組預先提供 Oracle 定義的 SQL 腳本，建立所有資料庫表格與檢視，供資料庫使用者存取的系統型錄，系統型錄表格和檢視是系統所擁有的，而不是屬於任何的一個資料庫結構，在 Oracle 舉例來說，系統型錄擁有者是被稱爲 SYS 的使用者帳戶，稱爲 SYS 的使用者帳戶有資料庫裡的完整權限，在 Sybase 中，SQL Server 的系統型錄是位於主資料庫（master database）。

21-3 什麼包含在系統型錄內？

系統型錄包含對許多使用者存取的各種不同訊息，有時被不同的使用者用在不同方面，系統型錄包含下列各項訊息：

- 使用者帳戶和預定的設定
- 特權和其他安全訊息
- 執行效率統計
- 物件大小
- 物件成長
- 表格結構和儲存
- 索引結構和儲存
- 資料庫其他物件的訊息例如檢視、同義字、觸發與儲存程序
- 表格限制和參考完整性訊息

- 使用者期間
- 稽核訊息
- 內在資料庫設定
- 資料庫檔案的位置

系統型錄由資料庫伺服器維護，舉例來說，當建立表格的時候，資料庫伺服器把資料插在適當的系統型錄表格或檢視之內，當表格結構被修改的時候，資料字典裡適當的物件也一起更新。下列的章節描述類別包含在系統型錄的資料類型。

21-3-1 使用者資料

所有個別使用者的訊息儲存在系統型錄；系統和物件特權已給與使用者，使用者擁有的物件和那些不被使用者擁有，但可被使用者存取的物件。使用者表格或檢視可讓每個人存取與做資料查詢，檢視您的廠商有關系統型錄物件的文件。

21-3-2 安全訊息

系統型錄也儲存安全訊息，例如使用者 ID、密碼加密與各種不同的特權和資料庫使用者存取資料的群組特權。稽核表格存在一些廠商裡，用來追蹤在資料庫裡面發生的動作，包括人物時間等等，資料庫使用期間，廠商也能透過系統型錄緊密的監視。

21-3-3 資料庫設計訊息

系統型錄包含真實資料庫的訊息，訊息包括資料庫建立的日期、名稱、物件大小、資料檔案存放的位置、參考完整性的訊息、存在資料庫的索引與特定欄的訊息和在資料庫裡欄對每個表格的屬性。

21-3-4 執行效率統計

執行效率統計主要用來維護系統型錄，SQL 指令執行效率的訊息，包括經過的時間和 optimizer 取得的 SQL 指令執行方法、其他執行效率的訊息如記憶體分配和使用，資料庫裡的自由空間，和允許表格與資料庫裡面控制的有關索引段落的訊息。這類的執行效率訊息能適當地協調資料庫、排列 SQL 查詢或對資料的存取重新設計，而達成較好的性能和 SQL 查詢反應時間。

21-4 廠商系統型錄表格的實例

每個廠商有幾個表格與檢視組成系統型錄，由使用者層次、系統層次與 DBA 層次分類。對於您的廠商，應該查詢這些表格，與閱讀所編寫的文件，取得較多的系統型錄表格訊息，參考表 21.1 檢視五個主要廠商的實例。

表 21.1 主要廠商的系統型錄物件

Microsoft SQL Server

表格名稱	描述
SYSUSERS	資料庫使用者訊息
SYSSEGMENTS	所有資料庫訊息段落
SYSINDEXES	所有索引上的訊息
SYSCONSTRAINTS	所有限制上的訊息

dBase

表格名稱	描述
SYSVIEWS	所有檢視訊息
SYSTABLS	所有表格訊息
SYSIDXS	所有索引訊息
SYSCOLS	表格欄上的訊息

Microsoft Access

表格名稱	描述
MSysColumns	有關表格欄裡的訊息
MSysIndexes	表格索引訊息
MSysMacros	有關句集訊息的建立
MSysObjects	有關所有資料庫物件訊息
MSysQueries	建立查詢的訊息
MSysRelationships	有關表格關係訊息

Sybase

表格名稱	描述
SYSMESSAGES	列出所有伺服器錯誤訊息
SYSKEYS	主索引和外部索引的主要訊息
SYSTABLES	所有表格和檢視的訊息
SYSVIEWS	所有檢視文字
SYSCOLUMNS	表格欄的訊息
SYSINDEXES	索引上的訊息
SYSOBJECTS	表格，觸發、檢視和相似的上的訊息
SYSDATABASES	伺服器上的所有資料庫的訊息
SYSPROCEDURES	檢視、觸發、和儲存程序的訊息,

Oracle

表格名稱	描述
ALL_TABLES	使用者存取表格上的訊息
USER_TABLES	使用者擁有的表格上訊息
DBA_TABLES	資料庫裡所有表格的訊息
DBA_SEGMENTS	段落儲存訊息
DBA_INDEXES	所有索引的訊息
DBA_USERS	資料庫的所有使用者的訊息

表格名稱	描述
DBA_ROLE_PRIVS	有關允許角色的訊息
DBA_ROLES	有關資料庫裡角色的訊息
DBA_SYS_PRIVS	有關得到系統特權的訊息
DBA_FREE_SPACE	有關資料庫自由空間的訊息
V$DATABASE	有關建立資料庫的訊息
V$SESSION	在目前期間的訊息


註解

這些只是一些不同的關連式資料庫廠商，取得的一些系統型錄，許多系統型錄物件，在廠商之間是相似的，整體而言，每個廠商對系統型錄內容的組織是特定的。

21-5 查詢系統型錄

系統型錄表格或檢視被當做任何其他的表格或檢視一樣，在 SQL 資料庫查詢中，使用者通常能查詢與使用者相關的表格，如對有特權的資料庫使用者帳戶，能存取各種不同的系統表格，例如資料庫管理員。

您建立 SQL 查詢，正如您爲了要取回來自系統型錄的資料，建立查詢來存取資料庫裡的表格，。

舉例來說，下列查詢傳回的所有列是來自 Sybase的表格 SYSTABLES 的資料：

```
SELECT * FROM SYSTABLES
```

下列的章節顯示您可能在查詢系統型錄表格所產生問題和一些訊息的實例。

21-5-1 查詢系統型錄的實例

下列各例子使用 Oracle 系統型錄，選擇 Oracle 沒有特別的理由，因爲是筆者最熟悉的廠商。

下列的查詢列出資料庫裡所有使用者帳戶：

TYPE
```
SELECT USERNAME
FROM ALL_USERS;
```

Output
```
USERNAME
------------------------------
SYS
SYSTEM
RYAN
SCOTT
DEMO
RON
USER1
USER2
8 rows selected.
```

下列查詢列出使用者擁有的所有表格：

TYPE
```
SELECT TABLE_NAME
FROM USER_TABLES;
```

Output
```
TABLE_NAME
------------------------------
CANDY_TBL
CUSTOMER_TBL
EMPLOYEE_PAY_TBL
EMPLOYEE_TBL
PRODUCTS_TBL
ORDERS_TBL
6 rows selected.
```

查詢返回所有系統特權，已授權給資料庫使用者 BRANDON。

TYPE
```
SELECT GRANTEE, PRIVILEGE
FROM SYS.DBA_SYS_PRIVS
WHERE GRANTEE = 'BRANDON';
```

Output

```
GRANTEE                        PRIVILEGE
------------------------------ ------------------------------
BRANDON                        ALTER ANY TABLE
BRANDON                        ALTER USER
BRANDON                        CREATE USER
BRANDON                        DROP ANY TABLE
BRANDON                        SELECT ANY TABLE
BRANDON                        UNLIMITED TABLESPACE
6 rows selected.
```

MS Access 的例子：

TYPE

```
SELECT NAME
FROM MSYSOBJECTS
WHERE NAME = 'MSYSOBJECTS'
```

Output

```
NAME
------------------
MSYSOBJECTS
```

註解

在這個章節顯示的例子，對於您從任何的系統型錄能取回的訊息是微不足道，請參照您的廠商所編寫文件，有關可用的表格裡面特定的系統型錄表格和欄位。

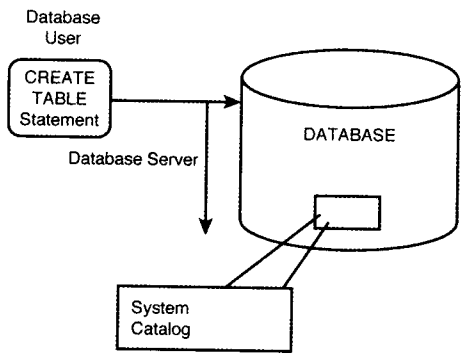
21-6 更新系統型錄物件

系統型錄只有用在查詢的時候，甚至被資料庫管理員使用的時候，系統型錄的更新是藉著資料庫伺服器自動地完成，舉例來說，在資料庫建立一個表格的時候，資料庫使用者發出 CREATE TABLE 指令，資料庫伺服器然後放置 DLL，放在適當的系統型錄表格之下，建立系統型錄裡的表格，從不需要手動更新系統型錄裡的任何表格，每個廠商的資料庫伺服器會依照在資料庫裡面發生的動作執行更新，如圖 21.2 所示。


注意

從不直接以任何方式處理系統型錄裡的表格，這樣做可能無法妥協處理資料庫的完整性，請記得在系統型錄維護，有關資料庫結構和資料庫裡所有物件的訊息，系統型錄從資料庫裡的所有其他資料被隔離出來。

圖 21.2.
更新系統型錄



21-7 摘要

您已經學會關連式資料庫的系統型錄，系統型錄感覺上是在資料庫裡面的資料庫，系統型錄本質上是包含所有常駐資料庫訊息的資料庫，它是維持資料庫全部的結構、追蹤事件和在資料庫裡面發生的變化，及對全部資料庫提供管理的訊息，系統型錄只被用來做查詢運算，資料庫使用者應該不曾直接到系統表格作任何變化，然而，每一次對資料庫結構進行變化，都是默默進行，例如表格的建立。系統型錄裡這些項目都是自動被資料庫伺服器變更。

21-7-1 Q&A

Q 做為資料庫使用者，了解我能發現物件的訊息，要如何能發現其他的使用者物件的訊息？

A 有一組表格和檢視，使用者能使用查詢最常用系統型錄，這些表格和檢視包括您想存取的物件的訊息。

Q 如果使用者忘記他的密碼，資料庫管理員能從表格查詢得到密碼嗎？

A 不可以，密碼是存放在系統型錄，是加密過的，所以就算是使用者也無法讀取密碼，如果使用者忘記密碼，密碼將會必須被重新設定，資料庫管理員能夠容易完成這工作。

Q 我如何能告訴欄的所在是在系統型錄表格？

A 系統型錄表格能被查詢，如同任何其他的表格，做個簡單的查詢，查詢表格的特別訊息。

21-8 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

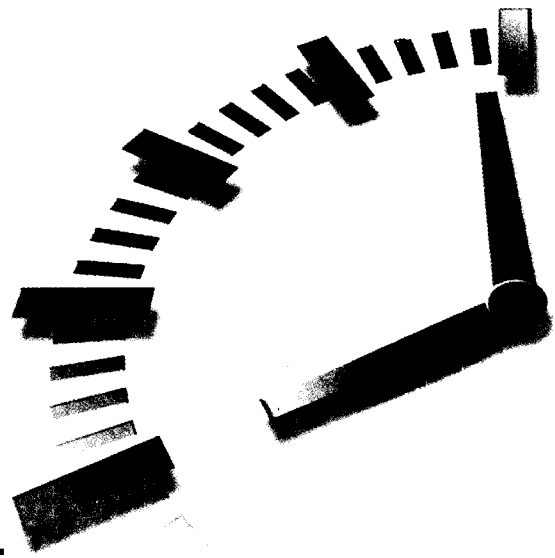
21-8-1 隨堂測驗

1. 系統型錄在一些廠商中也稱為什麼？
2. 一般的使用者可以更新系統型錄嗎？
3. 在 Sybase 什麼樣的系統表格可以用來取回檢視存在資料庫的訊息？
4. 誰擁有系統型錄？
5. 在 Oracle 系統物件 ALL_TABLES 和 DBA_TABLES 之間，有什麼不同？
6. 誰可以修改系統表格？

21-8-2 練習題

1. 查詢您的廠商系統型錄表格，先從保存使用者資料庫帳戶訊息的表格開始。

2. 查詢系統型錄，列出所有您可以存取的表格。
3. 查詢您已經得到系統與物件特權的表格。
4. 如果您有 DBA 或 SELECT 資料庫管理員表格上的特權，然後查詢這些表格，如果您沒有這些特權，檢視您的廠商所編寫的文件。



第 VIII 單元

應用 SQL 主要原理到今日世界

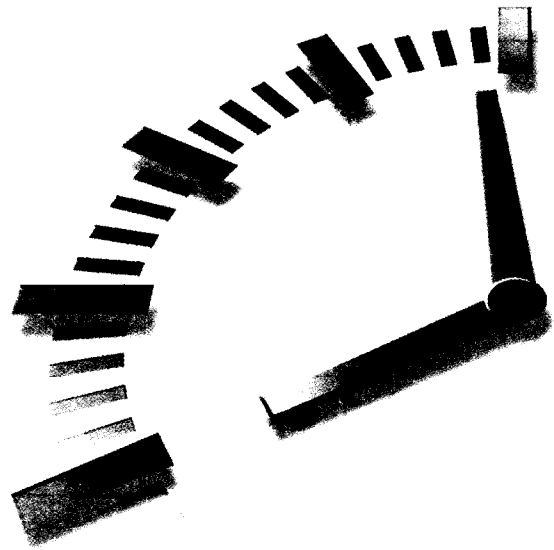
第 22 小時　進階 SQL 主題

第 23 小時　擴充 SQL 到企業、網際網路和企業內部網路

第 24 小時　標準 SQL 的擴充

式单 IIW 焊

焊出白令海峡到新主 108 用通



第 22 小時

進階 SQL 主題

在這個小時期間，您將瞭解一些進階 SQL 的主題，在這小時的結束後，您對游標、儲存程序、觸發、動態 SQL、直接箝入 SQL 和從 SQL 產生指令的觀念有進一步的了解。

本小時的重點包括：

- 游標是什麼？
- 使用儲存程序
- 觸發是什麼？
- 基本的動態 SQL
- 使用 SQL 產生 SQL
- 直接 SQL vs. 箝入 SQL
- 呼叫層次介面

22-1 進階主題

這個小時討論的進階 SQL 主題是，延伸您到現在爲止所學的基本運算，例如從資料庫查詢資料、建立資料庫結構和處理在資料庫裡的資料，在許多廠商提供進階主題，在這部份所討論的全部是爲了加強 SQL 資料處理。

並不是所有的主題是 ANSI SQL，所以您必須檢查您的廠商語法與規定，這裡顯示一些主要的廠商語法。

22-2 游標（Cursor）

對大多數的人而言，游標只是普通一閃一閃的點或正方形，出現在監視器上，並且指示檔案或應用程式正在那裡執行；然而，那不是這裡討論的游標類型，SQL 游標是在資料庫記憶體的區域裡，最後一個讓 SQL 指令儲存的，如果目前 SQL 指令是資料庫查詢，那麼來自查詢的列也被存在記憶體，這個列是游標目前的數值或目前的列，記憶體裡的區域被命名而且對程式來說是有用的。

游標主要用來取回來自資料庫的資料位置，因此，每個游標裡的列藉著程式可以一次一列的計算，當被程序類型箝入程式時，游標正常使用在 SQL 中。資料庫伺服器也建立一些游標，然而因爲其他 SQL 程式設計者定義，所以每個 SQL 廠商可能對游標的使用定義有所不同。

這一小節顯示來自二個廠商語法的實例：Microsoft SQL SERVER 和 Oracle。

對 Microsoft SQL SERVER 宣告游標的語法如下：

```
DECLARE CURSOR_NAME CURSOR
FOR SELECT STATEMENT
[ FOR [READ ONLY | UPDATE [ COLUMN_LIST ]}]
```

Oracle 宣告游標的語法如下：

```
DECLARE CURSOR CURSOR_NAME
IS {SELECT_STATEMENT}
```

下列各項游標包含來自 EMPLOYEE_TBL 表所有記錄的結果子集合：

```
DECLARE CURSOR EMP_CURSOR IS
SELECT * FROM EMPLOYEE_TBL
{ OTHER PROGRAM STATEMENTS }
```

依照 ANSI 標準，下列各項運算用來存取游標已經被定義：

OPEN	開啓游標定義
FETCH	從游標到程式變數取得列
CLOSE	當對游標的運算完成的時候閉合游標

22-2-1 開啟游標

當開啓游標的時候，特定的游標在 SELECT 指令執行，而且查詢結果儲存在記憶體裡的區域。

在 dBase 裡開啓游標的語法如下：

```
OPEN CURSOR_NAME
```

在 Oracle 裡開啓游標的語法如下：

```
OPEN CURSOR [ PARAMETER1 [, PARAMETER2 ]]
```

開啓 EMP_CURSOR：

```
OPEN EMP_CURSOR
```

22-2-2 從游標取得資料

一旦開啓游標，游標內容（從查詢得到的結果）能透過使用 FETCH 指令傳回來。

Microsoft SQL SERVER 裡 FETCH 指令的語法如下：

```
FETCH CURSOR_NAME [ INTO FETCH_LIST ]
```

Oracle 裡 FETCH 指令的語法如下：

```
FETCH CURSOR_NAME {INTO : HOST_VARIABLE
[[ INDICATOR ] : INDICATOR_VARIABLE ]
[, : HOST_VARIABLE
[[ INDICATOR ] : INDICATOR_VARIABLE ]]
| USING DESCRIPTOR DESCRIPTOR }
```

dBase 的語法如下：

```
FETCH CURSOR_NAME into MEMORY_VARIABLES
```

取得 EMP_CURSOR 的內容進入變數稱爲 EMP_RECORD，FETCH 指令如下：

```
FETCH EMP_CURSOR INTO EMP_RECORD
```

22-2-3 關閉游標

如果您能開啓游標，很顯然的您也能關閉游標，關閉游標相當簡單，一旦關閉它，對使用者程式不再是有用的。

> 關閉游標不一定會釋放與游標有關的記憶體，在一些廠商，游標使用的記憶體，一定要藉由使用 deallocate 指令來釋放記憶體空間，當游標 deallocated 的時候，釋放相關的記憶體空間，然後游標的名稱能重複使用，其他的廠商，當關閉游標的時候，記憶體暗自 dallocated，也以讓其他的運算使用，例如，一旦游標使用的空間釋放就可以開啟另一個游標。

Microsoft SQL Server 關閉游標和游標的 deallocation 語法如下：

```
CLOSE CURSOR_NAME
DEALLOCATE CURSOR CURSOR_NAME
```

在 Oracle 中當關閉游標時，釋放資源和名稱不用 deallocate 指令，Oracle 的語法如下：

```
CLOSE CURSOR_NAME
```

爲了釋放 dBase 裡的資源，表格必須關閉，而且在資源被釋放之前重開，這樣名稱才能重複使用，dBase 的語法如下：

```
CLOSE CURSOR_NAME
```


註解

正如您從先前的例子看到，對 SQL 的進階特徵和擴充，在那些廠商之中的差異更廣，第 24 小時將討論“擴充標準的 SQL”，您必須檢查您的廠商，有關游標正確使用方法。

22-3 儲存程序（Stored Procedures）

儲存程序是群組相關的 SQL 指令，通常是關於程式設計者所參照的函數與子程式，讓使用者使用更容易和有彈性。會這麼容易的原因是因爲，儲存程序通常比執行許多個別 SQL 指令較容易，儲存程序可以建立在其他儲存程序內部，所以儲存程序可以呼叫另外的儲存程序，另外的儲存程序也可能呼叫另外的儲存程序等等。

NEW TERM 儲存程序是由一個群體或較多的 SQL 指令或函數所組成的，它儲存在資料庫裡編譯，而且準備讓資料庫使用者執行。

當建立儲存程序的時候，那些各種不同的子程式實際上是由儲存程序及函數（使用 SQL）所組成，儲存在資料庫，這些儲存程序是被預先剖析，而且立刻準備好，準備讓使用者執行使用。

在 Microsoft SQL SERVER 建立儲存程序語法如下：

```
CREATE PROCEDURE PROCEDURE_NAME
[ [(] @PARAMETER_NAME
DATATYPE [(LENGTH) | (PRECISION [, SCALE ])
[ = DEFAULT ][ OUTPUT ]
[, @PARAMETER_NAME
```

```
DATATYPE [(LENGTH) | (PRECISION [, SCALE ])
[ = DEFAULT ][ OUTPUT ]] [()]]
[ WITH RECOMPLIE ]
AS SQL_STATEMENTS
```

Oracle 的語法如下：

```
CREATE [ OR REPLACE ] PROCEDURE PROCEDURE_NAME
[ (ARGUMENT [{IN | OUT | IN OUT} ] TYPE,
ARGUMENT [{IN | OUT | IN OUT} ] TYPE) ] {IS | AS}
PROCEDURE_BODY
```

非常簡單的儲存程序，例子如下：

TYPE

```
CREATE PROCEDURE NEW_PRODUCT
(PROD_ID IN VARCHAR2, PROD_DESC IN VARCHAR2, COST IN NUMBER)
AS
BEGIN
  INSERT INTO PRODUCTS_TBL
  VALUES (PROD_ID, PROD_DESC, COST);
  COMMIT;
END;
```

Output

```
Procedure created.
```

這個程序用來插入新的列到 PRODUCTS_TBL 表格之內。

執行 Microsoft SQL SERVER 裡的儲存的程序語法如下：

```
EXECUTE [ @RETURN_STATUS = ]
PROCEDURE_NAME
[[@PAREMETER_NAME = ] VALUE |
[@PARAMETER_NAME = ] @VARIABLE [ OUTPUT ]]
[WITH RECOMPLIE]
```

Oracle 的語法如下：

```
EXECUTE [ @RETURN STATUS =] PROCEDURE NAME
[[ @PARAMETER NAME = ] VALUE | [ @PARAMETER NAME = ] @VARIABLE
[ OUTPUT ]]]
[ WITH RECOMPLIE ]
```

現在執行您已經建立的程序：

TYPE
```
EXECUTE NEW_PRODUCT (9999,INDIAN CORN,1.99);
```

Output
```
PL/SQL procedure successfully completed.
```

22-3-1 儲存程序的優點

儲存程序清楚的顯示幾個個別在資料庫執行 SQL 指令的優點，包括下列各項：

- 指令已經儲存在資料庫。
- 指令已經剖析和是可執行的格式。
- 儲存程序支援模組化程式。
- 儲存程序呼叫其他的程序及函數。
- 儲存程序能讓其他的類型程式呼叫。
- 儲存程序典型地對整體的反應時間較好。
- 使用更容易。

22-4 觸發

當敘述資料處理語言的動作，在表格上執行的時候，觸發是儲存程序所執行的一種格式，觸發可以在 INSERT、DELETE 或 UPDATE 之後或之前執行，用來檢查資料完整性。觸發能取消異動，他們可以在一個表格修改資料，而從另外一個資料庫裡的另外一個表格讀取。

NEW TERM 觸發是用來執行資料庫裡面發生的其他動作，由資料庫裡編譯的 SQL 程序。

大體而言，觸發是非常好使用的功能，然而，卻造成 I/O 較多的負擔，當儲存程序或程式能用造成較少的負擔完成相同結果的時候，觸發就不應該被使用，Microsoft SQL SERVER 建立的觸發語法如下：

```
CREATE TRIGGER TRIGGER_NAME
ON TABLE_NAME
FOR { INSERT | UPDATE | DELETE [, ..]}
AS
SQL_STATEMENTS
[ RETURN ]
```

Oracle 的語法如下：

```
CREATE [ OR REPLACE ] TRIGGER TRIGGER_NAME
[ BEFORE | AFTER ]
[ DELETE | INSERT | UPDATE [ OF COLUMN_NAME ]
ON [ USER.TABLE_NAME ]
[ FOR EACH ROW ] [ WHEN CONDITION ]
[ PL/SQL BLOCK ]
```

觸發的例子：

TYPE

```
CREATE TRIGGER EMP_PAY_TRIG
AFTER UPDATE ON EMPLOYEE_PAY_TBL
FOR EACH ROW
BEGIN
  INSERT INTO EMPLOYEE_PAY_HISTORY
  (EMP_ID, PREV_PAY_RATE, PAY_RATE, DATE_LAST_RAISE,
   TRANSACTION_TYPE)
  VALUES
  (:NEW.EMP_ID, :OLD.PAY_RATE, :NEW.PAY_RATE,
   :NEW.DATE_LAST_RAISE, 'PAY CHANGE');
END;
/
```

OUTPUT

```
Trigger created.
```

這個例子表示建立稱爲 EMP_PAY_TRIG 的觸發這個觸發插入一個列到 EMPLOYEE_PAY_HISTORY 表格內，反映每一次對資料列在 EMPLOYEE_PAY_TBL 表格所做的更新。


註解

觸發的本身可能不可以改變，您必須替換或重新建立觸發，一些廠商允許觸發替換（如果有相同名稱的觸發已經存在），當做 CREATE TRIGGER 指令的一部份。

取消觸發的語法如下：

```
DROP TRIGGER TRIGGER_NAME
```

22-5 動態 SQL

動態 SQL 允許程式設計者或使用者在 runtime 建立特定的 SQL 指令，而且通過指令到資料庫，資料庫傳回資料到程式變數之內，程式變數是在約束 SQL runtime。

爲了要了解動態的 SQL，看一下靜態的 SQL，靜態的 SQL 是這本書一直討論的，靜態的 SQL 指令被編寫而且不能被改變，雖然靜態的 SQL 指令可以儲存爲檔案稍後執行，或當做資料庫裡的儲存程序，但是靜態的 SQL 不像動態的 SQL 一樣的有彈性。

靜態 SQL 的問題是，即使有很多的查詢提供給使用者使用，但是也有可能這些查詢將不能滿足使用者每次的需要，動態的 SQL 時常被 ad hoc 查詢工具所使用，允許 SQL 指令能讓使用者馬上建立，滿足特別的查詢需求，一旦指令根據使用者需要修改，指令就會送到資料庫，檢查語法是否有錯誤，與需要執行指令的相關特權，然後在資料庫編譯，指令在那裡被資料庫伺服器執行，動態的 SQL 能藉由使用呼叫層次介面被建立，在下一個章節將有詳細的解釋。


註解

雖然動態的 SQL 為使用者提供較有彈性的查詢需要，執行效率不可以與儲存程序比較，因為儲存程序的碼有被 SQL optimizer 分析過。

22-6 呼叫層次介面（Call-Level Interface）

應用程式的設計者應該對呼叫層次介面的觀念感到非常熟悉，它是其中一個程式語言，允許程式設計者插入不同的程序到 SQL，當使用呼叫層次介面（CLI）的時候，您只是通過 SQL 本文的指令，進入一個變數之中，使用主機程式語言的規則，透過 SQL 本文的變數的使用，您能在主程式裡執行 SQL 指令。

NEW TERM 呼叫層次介面用來插入 SQL 碼到主程式，例如 ANSI C。

EXEC SQL 是常用的主程式語言，允許您呼叫來自在 SQL 指令（CLI）裡面程式的指令。

22-6-1 EXEC SQL

支持 CLI 的程式語言的例子:

- COBOL
- ANSI C
- Pascal
- Fortran
- Ada


註解

參照您正在使用呼叫層次介面，選擇項有關的主程式語言的語法。

22-7 使用 SQL 產生 SQL

使用 SQL 產生 SQL 是非常節省時間，特別是在編寫 SQL 指令的時候，假定您已經有 100 個使用者在資料庫裡，新的角色—ENABLE（使用者定義物件，而且已取得特權）已經建立並且必須允許那 100 個使用者可以使用，下列各項 SQL 指令幫您產生那些指令，替代人工建立 100 個 GRANT 指令的方法：

```
SELECT 'GRANT ENABLE TO '|| USERNAME||';'
FROM SYS.DBA_USERS
```

這個例子使用 Oracle 系統型錄來檢視（爲包含使用者訊息）。

請注意在 GRANT ENABLE TO 所使用的單刮號，使用單刮號是允許任何在刮號之間的文字，記得文字值可能從表格被選擇，相同於從表格選擇欄一樣，USERNAME 是系統目錄表格 SYS.DBA_USERS 裡的欄，雙管符號（||）用來串接那些欄，跟隨著雙管符號使用的是 ";" 串接在分號後面的 username，如此表示完成指令。

SQL 指令的結果看起來像這樣：

```
GRANT ENABLE TO RRPLEW;
GRANT ENABLE TO RKSTEP;
```

這結果應該轉存到檔案，檔案能送到資料庫，資料庫依次執行檔案裡每個 SQL 的指令，節省您許多按鍵的時間，GRANT ENABLE TO USERNAME 能夠對資料庫裡每個使用者重複一次的指令。

下次您在寫 SQL 指令，重複幾次相同的指令，並且想像讓 SQL 爲您工作。

22-8 直接 SQL vs. 箝入 SQL

直接 SQL 是 SQL 指令，也是終端機執行格式交換的地方，SQL 的結果直接傳回到執行那個指令的終端機，這本書大多數的重點著重在直接 SQL 上，直接 SQL 也稱爲交談式執行或直接執行。

箝入 SQL 是將 SQL 碼用在其他程式裡，例如 Pascal、Fortran、COBOL 和 C. SQL 碼實際上箝入在主程式語言，如先前在呼叫層次介面所討論過，箝入主程式語言裡的 SQL 指令碼普遍是藉著 EXEC SQL 開始和在許多情況使用分號結束，其他終端字元包括 END-EXEC 和右括弧號。

這是主程式裡箝入 SQL 的一個例子，像 ANSI C 語言：

```
{host programming commands}
EXEC SQL {SQL statement};
{more host programming commands}
```

22-9 摘要

在這個小時討論一些進階的 SQL 觀念，雖然未談到許多細節，但是它確實提供您如何快速應用那些您未學習過的觀念，讓您瞭解一些基本的觀念，從游標開始，用游標查詢，傳送選擇的資料進入記憶體裡，一旦在程式裡宣告游標，便方便存取。首先它必須先開啓，然後游標的內容要進入變數之內，那時候資料能用做程式處理，所設定的結果會包含在記憶體裡，直到關閉游標和記憶體 deallocated。

然後討論儲存程序和觸發，基本上儲存程序是在資料庫一起儲存的 SQL 指令，這些指令連同其他廠商特定的指令，在資料庫編譯並且將任何特定的資料庫供使用者執行。觸發也是儲存程序的一種，基於在資料庫執行其他動作，所以允許儲存程序自動執行，儲存程序比個別的 SQL 指令提供更好的執行效率。

最後是討論動態的 SQL，使用 SQL 產生其他的 SQL 指令，與直接的 SQL 和箝入的 SQL 之間的不同，動態 SQL 是在 runtime 期間，使用者動態地立 SQL 碼，不像靜態 SQL，使用 SQL 編碼產生其他的 SQL 指令，也是最省時間的方法，應用您的廠商所提供的方法，可以自動建立許多沈悶的 SQL 指令，例如串接和文字值的選擇，最後，直接的 SQL 和箝入的 SQL 之間，最主要的不同是讓使用者直接從終端機發出 SQL 指令，然而箝入的 SQL 實際上箝入到主程式裡，爲了幫助程序處理資料。

在這個小時也討論一些進階的主題觀念，用來舉例說明企業 SQL 的應用程式，將在第 23 小時討論“擴充 SQL 到企業、網際網路和企業內部網路”。

22-9-1 Q&A

Q 儲存程序能呼叫另外的儲存程序嗎？

A 是的。正在被呼叫的儲存程序被稱爲 nested。

Q 我如何執行游標？

A 簡單的使用 OPEN CURSOR 指令，把游標的結果送到上層區域。

22-10 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

22-10-1 隨堂測驗

1. 觸發可以改變嗎？
2. 當游標被關閉的時候，您能重複使用名稱嗎？
3. 當游標已經開啟後，用什麼指令來取回結果？
4. 觸發要在 INSERT、DELETE 或 UPDATE 之後或之前執行？

22-10-2 練習題

1. 使用您的廠商系統目錄表格，建立下列各項 SQL 指令，以真實物件名稱代替物件名稱。

   a.

   ```
   GRANT SELECT ON TABLE_NAME TO USERNAME;
   ```

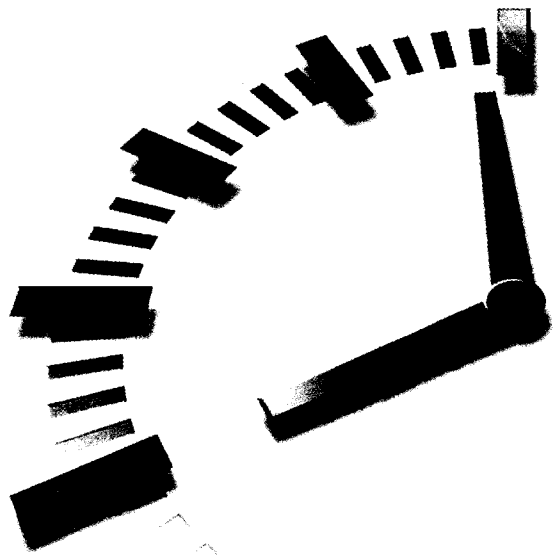
   b.

   ```
   GRANT, CONNECT, RESOURCE TO USERNAME;
   ```

   c.

   ```
   SELECT COUNT(*) FROM TABLE_NAME;
   ```

2. 寫個指令，建立儲存程序，用來刪除 PRODUCTS_TBL 表格的項目；它應該類似在這個小時所用的，插入新產品的例子。
3. 寫個指令，可以執行您在練習 2 所建立的儲存程序，刪除 PROD_ID 為 9999 的列。



第 23 小時

擴充 SQL 到企業、網際網路和企業內部網路

在這個小時中，您將學習 SQL 如何實際在企業與公司 intranet 上使用，及它如何應用到網際網路。

本小時的重點包括：

- SQL 和企業
- 前端和後端應用程式
- 存取遠端資料庫
- SQL 和網際網路
- SQL 和 intranet

23-1 SQL 和企業

早先討論一些高階的 SQL 主題，這些主題在書裡之前的內容有提到，從已經學習的 SQL 給您看實際的應用程式，這個小時著重在觀念上，把 SQL 擴充到企業中，包括 SQL 應用程式和建立所有員工每日使用的資料。

NEW TERM 企業（enterprise）是商業組織或公司。

許多商業指定可用的資料讓其他的企業、客戶和廠商使用，舉例來說，企業可能已經有產品詳細的說明，可供客戶存取，以便獲得較多的訂單，企業員工也需要包括在內，舉例來說，特定的員工資料也可以使用，如工作時間表、安排假期、訓練時間表、公司政策等等，建立相關資料庫。客戶和員工能經由 SQL 和網際網路存取企業重要的資料。

23-1-1 後端

任何應用程式的核心是後端應用程式，後端應用程式包括資料庫伺服器、資料來源和適當的使用 middleware 連結應用程式到區域網路上的 Web 或遠端的資料庫。

一些主要的資料庫伺服器包括 Oracle、Informix、Sybase、Microsoft SQL Server 和 Borland InterBase，這是企業藉著區域網路（LAN）、企業的 intranet 或是網際網路，移植任何應用程式的第一個步驟，資料庫伺服器應該由瞭解了解公司和應用程式需求的內部資料庫管理員建立。

NEW TERM 移植（Porting）描述對使用者提供實施應用程式環境的程序。

應用程式的 middleware 包括網路伺服器和能夠連接 Web 伺服器到資料庫伺服器的工具，主要目的是擁有能夠和公司資料庫的網路溝通的應用程式。

23-1-2 前端應用程式

前端應用程式是與使用者互動的應用程式，前端應用程式是公司購買的商用軟體產品，或是公司內部使用其他的協力廠商的應用程式，或用工具自行開發的軟體，協力廠商的工具敘述如下。

今日，許多有新的前端工具可以使用，使用者必須知道該在 C++、HTML 等程式語言規劃發展以 Web 爲基礎的應用程式，其他的語言、如 ANSI C、COBOL、Fortran 和 Pascal，已經早用在開發前端工具，公司內部的應用程式，主要是以字元爲基礎，今天最嶄新的前端開發應用程式是 GUI，有圖解式的使用者介面。

今天可用的工具，是使用者親和力高的物件導向，經由圖像、精靈和用滑鼠拖曳然後放下一些實用的工具，把應用程式轉成 Web，包括 C++Builder 和 Borland IntraBuilder 和微軟公司 Visual J++、C++，其他的應用程式，以公司區域網路爲基礎來開發的應用程式，如 Powersoft 的 PowerBuilder、Oracle 的 Developer/2000、微軟公司的 Visual Basic 和 Borland 的 Delphi。


註解

前端應用程式是簡化資料庫使用者的使用方式，在底下的資料庫，碼和在資料庫裡面發生的事件，對使用者是看不見的，前端應用程式的開發用來減輕使用者的困擾，用直覺的方式使用系統本身，新的技術更直覺與直接的使用應用程式，使端末使用者能夠把重心集中在他們的工作，藉此增加全部的生產力。

*圖 23.1 舉例說明資料庫應用程式的後端和前端元件，後端常駐*在主機伺服器上，也是資料庫存放的地方，後端使用者包括開發者、程式設計者、資料庫管理員、系統管理員和系統分析員；前端應用程

式常駐在用戶端機器上，用戶機器是使用者個人電腦，末端使用者是前端元件應用程式的廣大聽眾，可能包括資料登錄辦事員和會計員，使用者能夠藉著區域網路（LAN）或廣域區域網路（WAN）存取後端資料庫，一些類型的 middleware（例如 ODBC 驅動器）用來提供連接前面和後面之間的網路。

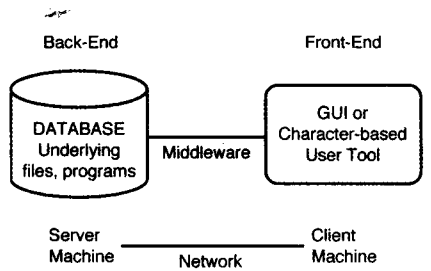


圖 23.1. 資料庫應用程式

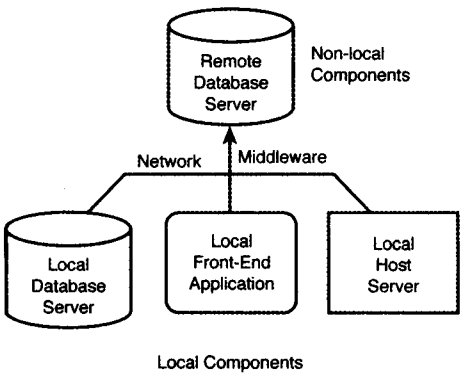
23-2 存取遠端資料庫

有時您正在存取的資料庫是區域性的資料庫，您可以直接連接到資料庫，大體而言，您或許會存取一些遠端資料庫的資料，遠端資料庫是非區域性的，意謂您必須利用網路和一些網路協定，才能對資料庫進行存取。

NEW TERM 遠端資料庫（remote database）是位於您現在連接到的伺服器之外伺服器上的資料庫。

有幾個方法可以存取遠端資料庫，從寬意的角度來說遠端資料庫經由使用 middleware 產品（ODBC 標準的 middleware 在下一個章節討論）的網路或網際網路連接存取。圖 23.2 表示存取遠端資料庫的三個情況。

圖 23.2.
存取遠端資料庫



這個圖顯示來自另外一個區域性的資料庫伺服器，對遠端伺服器的存取，區域性的前端應用程式和區域性的主機伺服器，因爲資料庫正常地常駐在區域性的主機伺服器上，所以區域性的資料庫伺服器和區域性的主機伺服器時常是相同的，然而，您通常能從區域性的伺服器連接到遠端資料庫，那個區域是目前沒有連接到的資料庫。對於一般使用者來說，前端應用程式是最典型存取遠端資料庫的方法，所有的方法必須依定他們的資料庫所設定的網路路線。

23-2-1 ODBC

開放資料庫連接（ODBC），藉著驅動程式允許連接到遠端資料庫，爲了連接到遠端資料庫也可能需要網路驅動程式，應用程式稱說有 ODBC 功能，是驅動程式管理員載入 ODBC 驅動程式。ODBC 驅動程式處理呼叫，傳送 SQL 請求與傳回來自資料庫的結果，ODBC 現在是一種標準而且有幾種產品使用，例如 Sybases PowerBuilder、FoxPro、Visual C++、Visual Basic、Borlands Delphi、Microsoft Access 等。

NEW TERM ODBC 驅動程式（ODBC driver）作爲前端應用程式與後端資料庫的介面。

所有的 RDBMS 廠商，有資料庫應用程式介面（Application Programmatic Interface, API）當做 ODBC 的一部份，在 Oracle 稱爲

開啓呼叫介面（Open Call Interface, OCI）與 Centuras SQLGateway 和 SQLRouter 也是一些可用的產品。

23-2-2 廠商連接產品

除了 ODBC 驅動程式之外，許多廠商有他們自己的產品，允許使用者連接到遠端資料庫，這些廠商的產品是針對特別的廠商而定，而且可能無法對其他類型的資料庫伺服器產生作用。

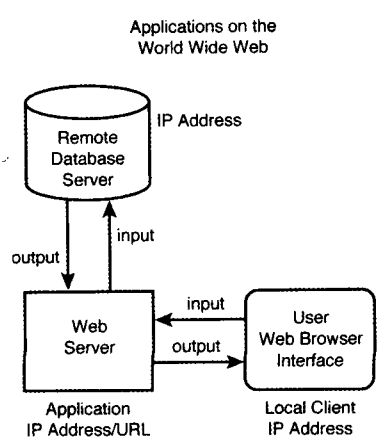
Oracle 公司有稱爲 SQL*NET 的產品，允許和遠端資料庫連接，SQL*Net 能被 TCP/IP、OSI、SPX/IPX 等的幾乎所有主要網路產品使用，除此之外，SQL*Net 能在大部分主要作業系統上執行。

Sybase 公司有稱爲 Open Client/C Developers Kit 的產品，支援例如 Oracle SQL*Net 廠商的產品。

23-3 藉著 Web 介面存取遠端資料庫

藉著區域性的網路與藉著 Web 介面存取遠端資料庫是非常類似的，主要不同的是，所有來自使用者對資料庫的請求必須透過 Web 伺服器（見圖 23.3）。

圖 23.3.
遠端資料庫 WEB 介面



您在圖 23.3 看見使用者藉著 Web 介面存取資料庫，但首先必須先啓動 Web 瀏覽器，Web 瀏覽器用來連接到特別的 URL 或網際網路 IP 位址，由 Web 伺服器決定位置，Web 伺服器爲了證明使用者存取權，傳送使用者請求，也許是查詢到遠端資料庫，遠端資料庫就確認使用者的權限，然後資料庫伺服器傳回結果給 Web 伺服器，Web 伺服器在 Web 瀏覽器對使用者顯示結果。未經認證的伺服器存取能藉由使用防火牆來控制。

NEW TERM 防火牆（firewall）是對未經認可連接到伺服器所設定的安全機器，一個或多重防火牆，能監控對資料庫或伺服器的存取。


注意

放在 Web 上使用訊息要小心，記得確定在所有的層次，適當地引用安全設定；包括 Web 伺服器，主機伺服器和遠端資料庫，個人隱私資料，例如個人社會福利編號，應該是保護好的，而不應該被流傳。

23-4 SQL 和網際網路

SQL 能被箝入或使用連結程式語言，例如 C 或 COBOL 等，舉例來說，SQL 也能箝入到網際網路程式語言，例如 JAVA，來自 HTML 的本文，另外網際網路語言，能翻譯進入 SQL 之內，從前端 Web 把查詢送來遠端資料庫，一旦資料庫處理查詢，輸出翻譯成 HTML，而且在個人 Web 瀏覽器上顯示查詢結果，下列的章節討論在網際網路上 SQL 的使用。

23-4-1 使全世界的客戶能使用資料

由於網際網路的出現，資料變得可以讓全世界的客戶和廠商使用，資料可以正常地透過前端工具，以唯讀存取的型態出現。

對客戶是可用的資料，可能包含客戶訊息、產品訊息、發票訊息、目前的訂單，後續的訂單和其他的相關訊息，而私人的訊息，例如公司策略和員工訊息，不應該公布。

網際網路上的 Home page 已經幾乎變成公司所必須的，以保持他們競爭力，網頁是非常有力的工具，能告訴來上網的人有關公司的產品、服務和其他的訊息，而且開支非常的少。

23-4-2 員工和有特權的客戶可用的資料

資料庫可能藉著網際網路或公司 intranet，讓員工或它的客戶存取，使用網際網路技術是有非常有價值的通信資產，可以保存有關公司政策的員工訊息、權益、訓練等等。

23-4-3 使用 SQL 為前端 Web 工具

有幾個存取資料庫的工具，多數有圖型使用者介面，使用者不必了解 SQL 資料庫查詢，這些前端工具允許使用者用滑鼠點一下，選擇物件代表表格，處理資料內部物件，將特定資料傳回等等，這些工具時常被開發，而且根據客戶需要或爲了要符合公司資料庫需要而修改。

23-5 SQL 和企業內部網路

IBM 本來建立 SQL，是爲了使用在大型電腦主機的資料庫和使用者機器之間，使用者經由區域網路連接到那些主機，SQL 被選爲在資料庫和使用者之間溝通的標準語言，intranet 基本上是小型的網際網路，主要的不同是 intranet 是單一組織使用，然而網際網路是讓一般的公眾存取的。在 intranet 裡，使用者(客戶)的介面與 client/server 環境相同。


註解

資料庫安全比網際網路上的安全更穩定，瞭解您資料庫伺服器安全的特性，並使用那些安全特性保護您的資料。

23-6 摘要

SQL 和資料庫應用程式延展到網際網路的觀念在本書的最後一個小時討論，它是非常重要的。在今日的時代，公司保持競爭性，爲了完成這個目的，必須開發應用程式，特別是全球資訊網的出現已經證明它的重要性，而且甚至從主從式系統移植到 Web 伺服器上，可以在網際網路移動，其中當出版這類的資訊或公司資料的時候，安全是最重要的考量也必須嚴格地厲行。

在區域性網路存取遠端資料庫和在網際網路上存取也討論到，爲了存取任何類型的遠端資料庫，每個方法都需要使用網路與協定通訊將需求編譯到資料庫，這是 SQL 廣義的程式應用觀念，如在區域性網路、公司 intranets 和網際網路上，您應該準備勇敢地進入 SQL 最後一個小時的旅程，並嘗試一些隨堂測驗和練習題。

23-6-1 Q&A

Q 在網際網路和 intranet 之間的不同是什麼？

A 網際網路提供連接，藉由使用 Web 介面，對公眾提供訊息，intranet 也使用 Web 介面，但是只允許內部存取，如公司員工和有特權的客戶。

Q 資料庫後端為 Web 應用程式與資料庫後端為主從式系統有任何不同嗎？

A 資料庫後端本身爲 Web 應用程式，與主從式系統不一定有任何不同，然而要以 Web 爲基礎的應用程式必須達到某些需求，舉例來

說，Web 伺服器用 Web 應用程式存取資料庫，使用 Web 應用程式，一般使用者不能直接連接到資料庫。

23-7 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

23-7-1 隨堂測驗

1. 伺服器上的資料庫能從另外一個伺服器存取嗎？
2. 什麼能傳播公司員工的訊息？
3. 允許到資料庫連接的產品叫做什麼？
4. SQL 能箝入到網際網路程式語言之內嗎？
5. 遠端資料庫如何透過 Web 應用程式存取？

23-7-2 練習題

1. 連到網際網路，看一看各種不同的公司首頁，如果您的公司有首頁，比較它與那些競爭廠商的首頁，試著回答以下問題：

   a. 網頁是否很快地傳過來，或是被太多圖形而傳遞變慢？

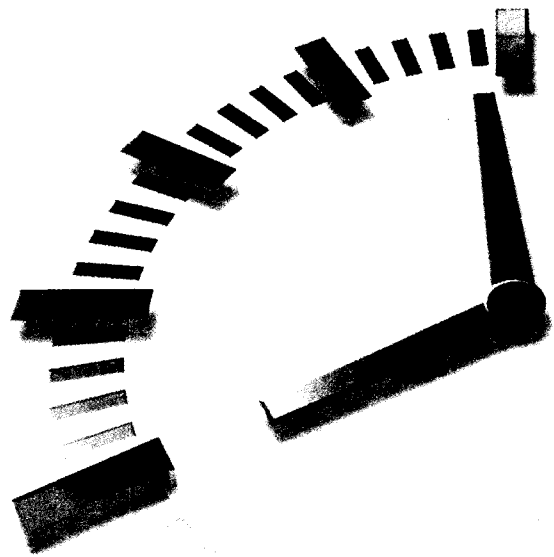
   b. 網頁有趣嗎？

   c. 您在讀過之後，知道關於公司的服務或產品的訊息嗎？

   d. 如果適用，對資料庫的存取是否容易？

   e. 網頁看起來，是否有任何的安全限制嗎？

2. 如果您的公司有 intranet，看一看有什麼訊息是有關公司的，有資料庫可以用嗎？如果有廠商是誰？是什麼類型的前端工具可供使用？



第 24 小時

標準 SQL 的擴充

這個小時涵蓋到 ANSI-standard SQL 的擴充，大多數的廠商大體而言使用一致的標準，許多廠商已經藉著各種不同的擴充，增強標準 SQL。

本小時的重點包括：

- 各種不同的廠商
- 廠商之間的差異
- 符合 ANSI SQL
- 交談式 SQL 指令
- 使用變數
- 使用參數

24-1 各種不同的廠商

有各種不同的 SQL 廠商釋放不同的產品，所有關連式資料庫的廠商，不能可能都提到，然而，一些領導廠商還是會討論，討論的廠商包括 Sybase、dBase、Microsoft SQL Server 和 Oracle，其他提供的資料庫產品廠商，除了上面被提到的外，還包括 Borland、IBM、Informix、Progress、CA-Ingres 等。

24-1-1 廠商之間的差異

雖然關連式資料庫產品的廠商，每個之間有特定的不同，這些不同起源於產品的設計和資料庫引擎處理資料的方式；然而，這本書專注在不同的 SQL 外觀，正如 ANSI 所指示，所有廠商使用 SQL 語言，與資料庫溝通。


註解

不同的廠商採用不同的 SQL 標準，為了要提高對 ANSI SQL 的使用執行效率和使用更容易，廠商也努力改善，增強他們本身產品的優點，使他們製造的產品更吸引客戶。

既然您知道 SQL，您應該對 SQL 有這樣的疑問，就是各種不同廠商的語法必須做調整，換句話說，如果您能寫 Sybase 裡的 SQL，您應該能夠寫 Oracle 裡的 SQL，此外，瞭解各種不同廠商的 SQL，除了讓您的履歷表比較好看之外，並不代表什麼。

下列的章節，是一些主要的廠商與 ANSI 標準，互相比較 SELECT 指令的語法。

這是 ANSI Standard：

```
SELECT [DISTINCT ] {* | COLUMN1 [, COLUMN2 ]
[ INTO HOST_VARIABLE ]
FROM TABLE1 [, TABLE2 ]
[ WHERE SEARCH_ CONDITION ]
GROUP BY { TABLE_ALIAS | COLUMN1 [, COLUMN2 ]
```

```
[ HAVING SEARCH_CONDITION ]]
[{UNION | INTERSECT | EXCEPT}[ ALL ]
[ CORRESPONDING [ BY (COLUMN1 [, COLUMN2 ]) ]
QUERY_SPEC | SELECT * FROM TABLE | TABLE_CONSTRUCTOR ]
[ORDER BY SORT_LIST ]
```

這是爲 **SQLBase** 的語法：

```
SELECT  [ ALL | DISTINCT ] COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE SEARCH_CONDITION ]
[ GROUP BY COLUMN1 [, COLUMN2 ]
[ HAVING SEARCH_CONDITION ]]
[ UNION [ ALL ]]
[ ORDER BY SORT_LIST ]
[ FOR UPDATE OF COLUMN1 [, COLUMN2 ]
```

這是 **Oracle** 的語法：

```
SELECT [ ALL | DISTINCT ] COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE SEARCH_CONDITION ]
[[ START WITH SEARCH_CONDITION ]
CONNECT BY SEARCH_CONDITION ]
[ GROUP BY COLUMN1 [, COLUMN2 ]
[ HAVING SEARCH_CONDITION ]]
[{UNION [ ALL ] | INTERSECT | MINUS} QUERY_SPEC ]
[ ORDER BY COLUMN1 [, COLUMN2 ]
[ NOWAIT ]
```

這是爲 **Informix** 的語法：

```
SELECT [ ALL ] | DISTINCT | UNIQUE ] COLUMN1 [, COLUMN2 ]
FROM TABLE1 [, TABLE2 ]
[ WHERE SEARCH_CONDITION ]
[ GROUP BY {COLUMN1 [, COLUMN2 ] | INTEGER}
[ HAVING SEARCH_CONDITION ]]
[ UNION QUERY_SPEC ]
[ ORDER BY COLUMN1 [, COLUMN2 ]
[ INTO TEMP TABLE [ WITH NO LOG ]]
```

當您比較語法實例時，您可以看見基本的語法就是 *SELECT*、*FROM*、WHERE、GROUP BY、HAVING、UNION 和 ORDER BY 子句，每個子句用相同

地概念執行，但是一些附加選擇項，不會在其他廠商中發現，這些選擇項稱爲增強選項。

24-1-2 符合 ANSI SQL

廠商確實努力遵從 ANSI SQL；然而，不可能百分之百達到 ANSI SQL 的標準，一些廠商已經增加指令與函數到 ANSI SQL，而大部份新的指令或函數，ANSI SQL 也已經採用。廠商所遵從的標準，有許多有正面的利益，明顯的好處是學習那些廠商的產品變得容易的，因爲 SQL 碼的可攜性，所以 SQL 碼到其他廠商也可以使用，也因此資料庫可以轉移到另外一個廠商的系統。爲什麼公司會花無數的費用，轉換到一個不是符合標準的廠商上？可能是應用程式執行有太多變化，他們或許不會新的產品，也或許學習新的廠商產品很困難。因此，產品符合 ANSI SQL 在大部份的情形下不會產生上述問題。

24-1-3 擴充 SQL 的功能

實際上所有主要的廠商都有擴充 SQL 的功能，每個廠商所擴充的 SQL 都是不一樣的，而且只能在自己的產品上使用，然而，ANSI 瞭解大眾擴充的標準後，有時會成爲新的標準，PL/SQL 是 Oracle 公司的一種產品，Transact-SQL 被 Sybase 和 Microsoft SQL Server 使用，這二個例子是標準 SQL 擴充，兩個擴充與有關係的細節，在這個小時會用一些實例做討論。

24-2 一些廠商的擴充實例

PL/SQL 和 Transact-SQL 都考慮爲第四代程式語言，兩者都是程序語言而 SQL 是非程序語言。

非程序語言 SQL 包括如下列的各項指令：

- INSERT 插入
- UPDATE 更新
- DELETE 刪除
- SELECT 選擇
- COMMIT 委託
- ROLLBACK

SQL 擴充考慮的程序語言需包括所有之前指令與標準的 SQL 函數，除此之外，擴充的指令包括：

- Variable declarations 變數的宣告
- Cursor declarations 游標宣告
- Conditional statements 有條件的指令
- Loops 迴圈
- Error handling 錯誤處理
- Variable incrementing 變數增量
- Date conversions 日期變換
- Wildcard operators 萬用字元運算子
- Triggers 觸發
- Stored procedures 儲存程序

這些指令允許程式設計者對處理程序語言有較多的控制。

註解

標準的 SQL 主要是非程序語言，非程序語言表示您執行指令到資料庫伺服器，資料庫伺服器決定如何最佳的執行指令，程序語言除了允許程式設計者請求資料傳回或處理，而且正好告訴資料庫伺服器該如何執行請求。

24-2-1 Transact-SQL

Transact-SQL 是程序語言，程序語言是您告訴資料庫如何及在哪裡找要處理的資料；SQL 是非程序的，而且資料庫決定如何和選擇在哪找尋處理的資料，Transact-SQL 的執行效能很強，包括宣告位置和全域變數、游標、錯誤處理、觸發、儲存程序，迴圈，萬用字元運算子、日期變換與摘要報表。

Transact-SQL 指令如下：

```
IF (SELECT AVG(COST) FROM PRODUCTS_TBL) > 50
BEGIN
  PRINT "LOWER ALL COSTS BY 10 PERCENT."
END
ELSE
  PRINT "COSTS ARE REASONABLE."
END
```

ANALYSIS 這是非常簡單的 Transact-SQL，它陳述如果在表格 PRODUCTS_TBL 的平均值大於 50，那麼本文 LOWER ALL COSTS BY 10 PERCENT 將會被印出，如果在表格 PRODUCTS_TBL 的平均值小於或等於 50，那麼本文 COSTS ARE REASONABLE 被印出。

請注意 IFELSE 指令是使用評估資料數值的條件，PRINT 指令也是新的指令，這些附加的選項只是 Transact-SQL 的一小部分而已。

24-2-2 PL/SQL

PL/SQL 是 Oracle 對 SQL 的擴充，像 Transact-SQL，PL/SQL 是程序語言，PL/SQL 用合乎邏輯的區塊所構成，PL/SQL 區塊有三個區段，其中二個是選擇性的，第一個區段是 DECLARE 區段是選擇性的，DECLARE 區段包含變數、游標和常數。第二個區段叫做 PROCEDURE 區段，PROCEDURE 區段包含條件的指令和 SQL 指令，這個區段是被區塊控制的地方，PROCEDURE 區段是必須的。第三個區段叫做 EXCEPTION 區段，EXCEPTION 區段定義程式如何處理錯誤和使用者定義例外狀況，

EXCEPTION 區段是選擇性的。PL/SQL 的重點包括變數、常數、游標、屬性、迴圈，處理例外使用、顯示輸出到程式設計者、異動控制的輸出、儲存程序、觸發和 packages。

PL/SQL 指令的例子如下：

```
DECLARE
  CURSOR EMP_CURSOR IS SELECT EMP_ID, LAST_NAME, FIRST_NAME, MID_INIT
                       FROM EMPLOYEE_TBL;
  EMP_REC EMP_CURSOR%ROWTYPE;
BEGIN
  OPEN EMP_CURSOR;
  LOOP
    FETCH EMP_CURSOR INTO EMP_REC;
    EXIT WHEN EMP_CURSOR%NOTFOUND;
    IF (EMP_REC.MID_INIT IS NULL) THEN
      UPDATE EMPLOYEE_TBL
      SET MID_INIT = 'X'
      WHERE EMP_ID = EMP_REC.EMP_ID;
      COMMIT;
    END IF;
  END LOOP;
  CLOSE EMP_CURSOR;
END;
```


ANALYSIS 在這個例子三個區段被使用了二個：DECLARE 區段和 PROCEDURE 區段，首先，稱爲 EMP_CURSOR 的游標查詢定義，其次，稱爲 EMP_REC 的宣告變數，數值有相同 的資料格式（%ROWTYPE），如同每個欄位在定義的游標裡一樣。PROCEDURE 區段裡的第一個步驟（在 BEGIN 語句）是開啓游標，一旦開啓游標，您使用 LOOP 指令捲動過游標的每個記錄，最後 END LOOP 結束迴圈，如果員工的中間縮寫是 NULL，EMPLOYEE_TBL 表格裡在游標裡的所有列應該更新，更新將中間縮寫設定爲 X，一直委託改變直到游標關閉。

24-3 交談式 SQL 指令

交談式 SQL 指令是完全執行前問您變數、參數或一些資料的形式的 SQL 指令，假設您已經有交談式的 SQL 指令。指令用在資料庫

之內建立使用者，SQL 指令會提示您像是使用者身份證、使用者的名字和電話號碼等資訊，並能夠針對一個或多個使用者且只會執行一次。否則，每個使用者將會用 CREATE USER 指令，個別輸入個人資料，SQL 指令也會問您相關的特權。並不是所有廠商有交談式 SQL 指令，必須檢查看看您的廠商，下列的章節表示使用 Oracle 交談式 SQL 的一些例子。

24-3-1 使用參數

參數是寫在 SQL 並且在應用程式常駐的變數，在 runtime 的時候，參數能傳到 SQL 指令之內，允許使用者執行較有彈性的指令，多數主要的廠商允許使用參數，下列的章節表示 Oracle 和 Sybase 使用參數的例子。

Oracle

在 Oracle 裡，參數能傳到靜態的 SQL 指令之內。

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME
FROM EMPLOYEE_TBL
WHERE EMP_ID = '&EMP_ID'
```

之前的 SQL 指令傳回 EMP_ID、LAST_NAME 和 FIRST_NAME，無論您在 emp_id 輸入什麼。

```
SELECT *
FROM EMPLOYEE_TBL
WHERE CITY = '&CITY'
AND STATE = '&STATE'
```

之前的指令，問您城市和洲的資料，查詢傳回您輸入住在城市裡職員的所有資料。

Sybase

Sybase 裡的參數能傳到儲存程序之內。

```
CREATE PROC EMP_SEARCH
(@EMP_ID)
AS
SELECT LAST_NAME, FIRST_NAME
FROM EMPLOYEE_TBL
WHERE EMP_ID = @EMP_ID
```

鍵入下列指令然後執行儲存程序與傳送參數：

```
SP_EMP_SEARCH 443679012
```


24-4 摘要

這個小時討論廠商和他們對符合 ANSI 標準所提供標準 SQL 的擴充，一旦您學會 SQL 就能輕易應用您的知識與編碼到 SQL 的其他廠商，SQL 在廠商之間是有相容性的，大多數的 SQL 編碼能做一些小幅度的修改，且能應用於大多數的廠商中。

這個小時的最後一個部份，顯示三個廠商特定的二個擴充功能，Transact-SQL 被 Microsoft SQL Server 和 Sybase 使用，而 PL/SQL 被 Oracle 採用，您應該會看到在 Transact-SQL 和 PL/SQL 之間一些類似的地方，值得注意的是，這二個廠商已經符合標準，然後才增強廠商自己的產品，並增加更好的功能和效率。這個小時是想讓您知道許多 SQL 擴充確實存在，和教導廠商瞭解 ANSI SQL 標準的重要。

如果您帶著在這本書所學到的—建立編碼、測試和建造在您的知識上，在未來管理應用 SQL 上，您會無往不利。公司可以有資料，但沒有資料庫則不能運作，既然 SQL 是標準的語言，所以關連式資料庫是無所不在，您要溝通與管理關連式資料庫，必須下定決心好好學習 SQL。祝您好運！

24-4-1 Q&A

Q 為什麼 SQL 裡有差異存在？

A SQL 裡的差異在那些各種不同的廠商之間，因爲資料被儲存的方式不同， 各種不同的廠商試著在競爭得到優勢，所以新的主意不斷產生，所以各家產品也跟著不同。

Q 在學習基本的 SQL 之後，我將會能夠使用不同廠商的 SQL 嗎？

A 是的。然而，記得廠商之間的不同和差異，而且 SQL 的基本結構在大多數的廠商之中是相同的。

24-5 綜合練習

下列的綜合練習是由一系列的隨堂測驗與實做練習題所組成。隨堂測驗是用來測試您對本小時的整體瞭解度。實做練習題是讓您有機會應用本小時所提到的概念同時複習上一小時的課程。請花一些時間來完成這些隨堂測驗與實做練習題，您可以參考在附錄 C 的解答。

24-5-1 隨堂測驗

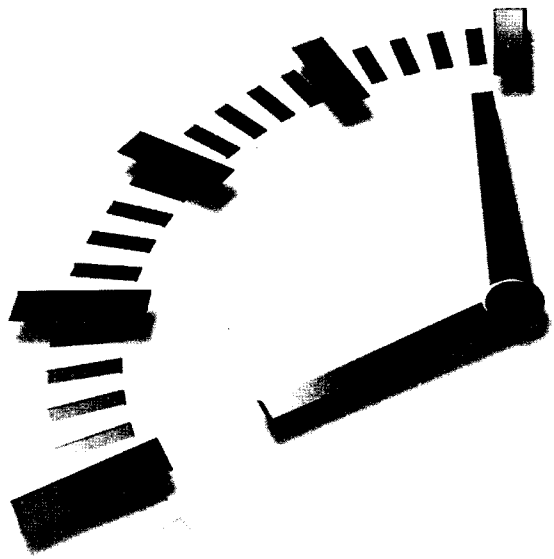
1. SQL 是程序或非程序語言？
2. 為什麼 SQL 有理由可以存在不同的差異？
3. 什麼是在游標、宣告游標外面的三個基本運算？
4. 程序或非程序中，哪一個是讓資料庫自己決定該如何評估並且執行 SQL 指令？

24-5-2 練習題

1. 試著研究有關在各種不同廠商之中的 SQL，到圖書館或書店尋找不同 SQL 廠商的書，比較各種不同的 SQL 指令，例如資料處理語言（DML），比較插入、刪除和更新的不同，您也可以尋找 ANSI SQL 的書與比較。

2. 使用 EMPLOYEE_TBL 表格（見附錄 D），編寫交談式 SQL 指令，傳回郵遞區號為 46234 的所有職員名稱。

3. 交出您的履歷表。



第 IX 單元

附錄

附錄 A　SQL 指令

附錄 B　專有名詞

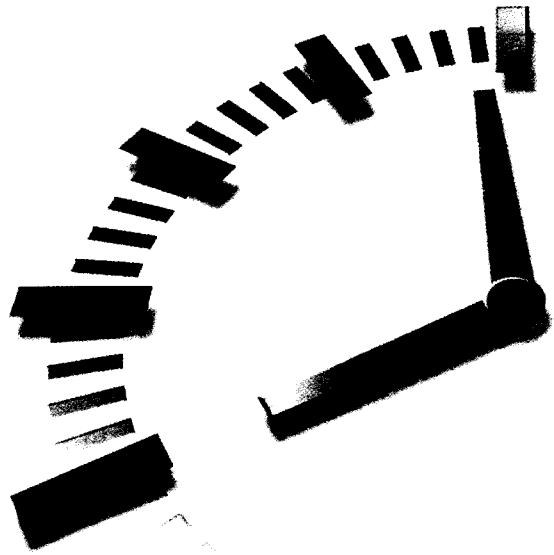
附錄 C　隨堂測驗與練習題答案

附錄 D　本書用 CREATE TABLE 指令建立的指令

附錄 E　本書用 INSERT 指令建立的範例資料

示範 XD 說，

設計



附錄 A

SQL 指令

A-1 SQL 指令

A-1-1 ALTER TABLE

```
alter table table_name
[modify | add | drop]
  [column column_name][datatype|null not null] [restrict|cascade]
[add | drop]  [constraint constraint_name]
```

敘述：更改表格欄

A-1-2 COMMIT

```
COMMIT [ TRANSACTION ]
```

敘述：儲存資料庫異動

A-1-3 CREATE INDEX

```
CREATE INDEX INDEX_NAME
ON TABLE_NAME (COLUMN_NAME)
```

敘述：建立表格索引

A-1-4 CREATE TABLE

```
CREATE TABLE TABLE_NAME
( COLUMN1     DATA_TYPE     [NULL|NOT NULL],
  COLUMN2     DATA_TYPE     [NULL|NOT NULL])
```

敘述：建立資料庫表格

A-1-5 CREATE TABLE AS

```
CREATE TABLE TABLE_NAME AS
SELECT COLUMN1, COLUMN2,...
FROM TABLE_NAME
[ WHERE CONDITIONS ]
[ GROUP BY COLUMN1, COLUMN2,...]
[ HAVING CONDITIONS ]
```

敘述：用資料庫表格建立其他表格

A-1-6 CREATE VIEW

```
CREATE VIEW AS
SELECT COLUMN1, COLUMN2,...
FROM TABLE_NAME
[ WHERE CONDITIONS ]
[ GROUP BY COLUMN1, COLUMN2,... ]
[ HAVING CONDITIONS ]
```

敘述：建立資料庫檢視

A-1-7 DELETE

```
DELETE
FROM TABLE_NAME
[ WHERE CONDITIONS ]
```

敘述：從資料庫刪除一列

A-1-8 DROP INDEX

```
DROP INDEX INDEX_NAME
```

敘述：取消表格索引

A-1-9 DROP TABLE

```
DROP TABLE TABLE_NAME
```

敘述：從資料庫取消表格

A-1-10 DROP VIEW

```
DROP VIEW VIEW_NAME
```

敘述：取消表格檢視

A-1-11 GRANT

```
GRANT PRIVILEGE1, PRIVILEGE2, ... TO USER_NAME
```

敘述：讓使用者得到特權

A-1-12 INSERT

```
INSERT INTO TABLE_NAME [ (COLUMN1, COLUMN2,...]
VALUES ('VALUE1','VALUE2',...)
```

敘述：插入新資料列到表格

A-1-13 INSERT...SELECT

```
INSERT INTO TABLE_NAME
SELECT COLUMN1, COLUMN2
FROM TABLE_NAME
[ WHERE CONDITIONS ]
```

敘述：用其他表格中的資料插入新資料列到表格中

A-1-14 REVOKE

```
REVOKE PRIVILEGE1, PRIVILEGE2, ... FROM USER_NAME
```

敘述：取消使用者特權

A-1-15 ROLLBACK

```
ROLLBACK [ TO SAVEPOINT_NAME ]
```

敘述：回覆資料庫異動

A-1-16 SAVEPOINT

```
SAVEPOINT SAVEPOINT_NAME
```

敘述：建立異動 SAVEPOINT

A-1-17 SELECT

```
SELECT [ DISTINCT ] COLUMN1, COLUMN2,...
FROM TABLE1, TABLE2,...
[ WHERE CONDITIONS ]
[ GROUP BY COLUMN1, COLUMN2,...]
[ HAVING CONDITIONS ]
[ ORDER BY COLUMN1, COLUMN2,...]
```

敘述：建立查詢，從資料庫傳回一筆或多筆資料

A-1-18 UPDATE

```
UPDATE TABLE_NAME
SET COLUMN1 = 'VALUE1',
    COLUMN2 = 'VALUE2',...
[ WHERE CONDITIONS ]
```

敘述：更新表格現有資料

A-2 SQL 子句

A-2-1 SELECT

```
SELECT *
SELECT COLUMN1, COLUMN2,...
SELECT DISTINCT (COLUMN1)
SELECT COUNT(*)
```

敘述：定義欄顯示查詢結果

A-2-2 FROM

```
FROM TABLE1, TABLE2, TABLE3,...
```

敘述：從傳回資料定義表格

A-2-3 WHERE

```
WHERE COLUMN1 = 'VALUE1'
  AND COLUMN2 = 'VALUE2'
...
WHERE COLUMN1 = 'VALUE1'
   OR COLUMN2 = 'VALUE2'
...
```

敘述：在查詢中定義限制的條件

A-2-4 GROUP BY

```
GROUP BY GROUP_COLUMN1, GROUP_COLUMN2,...
```

敘述：一種排序運算，將群組分到不同的組裡

A-2-5 HAVING

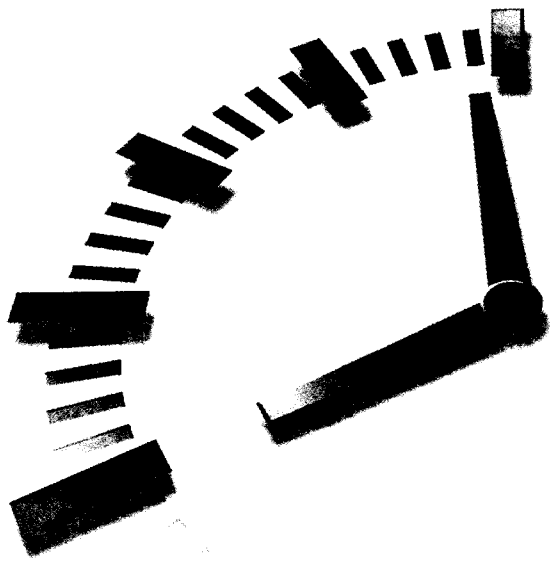
```
HAVING GROUP_COLUMN1 = 'VALUE1'
   AND GROUP_COLUMN2 = 'VALUE2'
...
```

敘述：類似 WHERE 子句，在 GROUP BY 放置條件

A-2-6 ORDER BY

```
ORDER BY COLUMN1, COLUMN2,...
ORDER BY 1,2,...
```

敘述：排序查詢結果



附錄 B

專有名詞

Alias（別名） 表格或欄的另外名稱。

ANSI American National Standards Institute.美國國家標準協會。

Application（應用程式） 能使用資料庫來執行有關商業功能的畫面、表格和報表。

Buffer（緩衝區） SQL 編輯或執行時所佔的記憶體區域。

Cartesian product（笛卡爾產品） SQL 指令中表格不加入所產生結果的 WHERE 子句，因爲在查詢時，若表格不加入結果的時候，表格裡的每個列與所有其他表格裡的每個列成爲一對。

Client（用戶端） 用戶端是典型的個人電腦。

Column（欄） 表格的一部份，有名字和特定的資料格式。

COMMIT（委託） 對資料做永久的變化。

composite index（複合索引） 二個或以上的欄位所組成的索引。

Condition（條件） 搜尋查詢裡的標準，用 WHERE 子句評估對或錯。

Constant（常數） 不改變的數值。

Constraint（限制） 在資料層次上對資料的強制限制。

Cursor（游標） 記憶體裡的工作區域，儲存目前的 SQL 指令。

data dictionary（資料字典） 爲系統目錄另外的名稱

Database（資料庫） 資料的收集。

Datatype（資料格式） 定義資料格式例如數目，日期或字元。

DBA Database Administrator，資料庫管理員，管理資料庫的人。

DDL Data Definition Language，資料定義語言。

Default（預設值） 當規格沒有被設定時所使用的值。

Distinct（不同的） 唯一的；SELECT 子句傳回唯一的數值。

DML Data Manipulation Language，資料處理語言。

DQL Data Query Language，資料查詢語言。

End Users 工作上需要查詢或處理資料庫裡資料的使用者，端末使用者。

Field（欄位） 爲表格裡另外的欄位名稱。

foreign key（外部索引） 一個或以上的欄位，基於其他表格主索引的數值。

full table scan（完整表格掃瞄） 沒有索引時，對表格搜尋所使用的查詢方法。

Function（函數） 被預先定義的運算，可在 SQL 指令中處理資料。

GUI Graphical User Interface，圖解式使用者介面。

Host（主機） 資料庫位在的電腦。

Index（索引） 指到表格資料中，是使存取表格更有效率的指標。

Join（加入） 藉由連接欄，聯合來自不同表格的資料，使用在 SQL 指令的 WHERE 子句。

Key（鍵） 識別表格列的欄。

Normalization（正常化） 藉由打破較大的表格進入較小的表格，減少資料重複，使資料庫變小，容易管理。

NULL value（NULL 值） 未知的數值。

Object（物件） 資料庫裡的元件，例如表格、檢視、程序。

Operator（運算子） 保留字或符號，用在執行運算例如加法、減法。

Optimizer 資料庫的一部份，決定該如何執行 SQL 指令而且傳回答案。

Parameter（參數） 一個範圍的數值，用來解決 SQL 指令的一部份程式。

Primary key（主索引） 特定的表格欄位，是表格中唯一的識別列。

Privilege（特權） 允許使用者執行資料庫裡特定運算的特權。

Procedure（程序） 一組指令，用來節省重複執行與呼叫。

Public（公用） 表現所有資料庫使用者的使用者帳戶。

Query（查詢） SQL 指令用來傳回來自資料庫的資料。

Record（記錄） 參考“列”。

Referential integrity（參考完整性） 保證欄的數值，是因另外欄位的數值而定。

Relational database（關連式資料庫） 資料庫是由列所組成的表格，列包含相同項目的資料，表格在資料庫裡藉著常用的欄位，建立表格與表格之間的關係。

ROLLBACK 一個指令能自最後的 COMMIT 或 SAVEPOINT 指令所執行後，復原所有的異動。

Row（列） 表格裡的記錄組。

Savepoint 在異動中特定的點，讓您開始 ROLLBACK 或復原變更。

Schema（資料庫結構） 擁有一組資料庫物件。

Security（安全） 確定資料庫裡的資料被完全保護的程序。

SQL Structured Query Language，結構化查詢語言。

Stored procedure（儲存程序） SQL 碼儲存在資料庫和即將執行。

Subquery（子查詢） 另外 SQL 指令箝入的 SELECT 指令。

Synonym（同義字） 是給表格或檢視的另外一個名字。

Syntax for SQL（SQL 的語法） 一組規則，表示 SQL 指令構造，包含強制性和可選擇的部份。

System catalog（系統目錄） 收集包含資料庫訊息的表格或檢視。

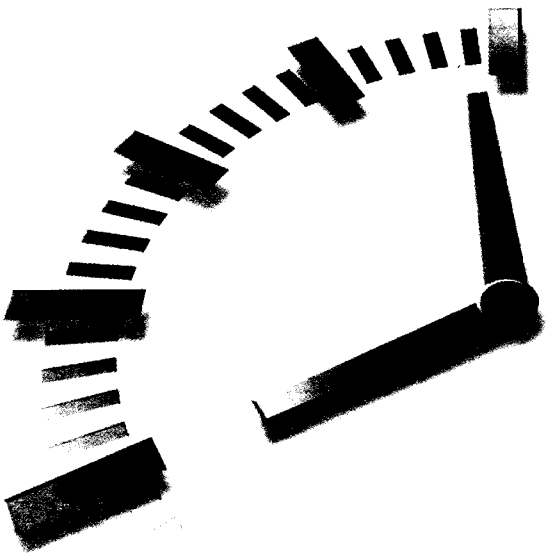
Table（表格） 在關連式資料庫裡，基本邏輯的儲存單位。

Transaction（異動） 一個或多個 SQL 指令執行爲一個單位。

Trigger（觸發） 儲存程序，執行在資料庫裡特定的事件，像是在表格更新之後或之前。

Variable（變數） 不保持固定的數值。

View（檢視） 從一個或較多的表格所建立的資料庫物件，而且能像表格一樣被使用，檢視是不需要儲存空間，因爲是虛擬的表格。



附錄 C

隨堂測驗與練習題答案

C-1 第 1 小時：歡迎來到 SQL 的世界

C-1-1 隨堂測驗答案

1. SQL 的縮寫代表什麼？

   a. SQL 代表結構查詢語言。

2. SQL 指令的六個主要類別為何？

   a. DDL，Data Definition Language（資料定義語言）

      DML，Data Manipulation Language（資料製造語言）

      DQL，Data Query Language（資料查詢語言）

      DCL，Data Control Language（資料控制語言）

      Data administration commands （資料管理指令）

      Transactional control commands（異動控制指令）

3. 四種異動控制指令是指那四種？

   a. Commit

      Rollback

      Savepoint

      Set Transactions

4. 主從式系統與大型主機的主要差異為何？

   a. 大型主機是使用者終端機集中聯接到的電腦，在環境，使用者經由網路聯接到伺服器，而且使用者典型地用個人電腦，與在主從式所用的終端機不同。

5. 如果欄位定義為 null，是否代表這欄位一定要輸入資料？

   a. 不，如果欄被定義為 NULL，欄無須輸入任何值。如果欄被定義不是 NULL，則某些值必須被輸入。

C-1-2 練習題答案

1. 請說明下列 SQL 指令屬於那一種指令類別？

```
create table
delete
select
insert
alter table
update
```

   a. Create table-DDL，Data Definition Language

      Delete-DML，Data Manipulation Language

      Select-DQL，Data Query Language

      Insert-DML，Data Manipulation Language

      alter table-DDL，Data Definition Language

      update-DML，Data Manipulation Language

C-2 第 2 小時：資料結構的定義

C-2-1 隨堂測驗答案

1. 是非題：個人社會福利號碼可屬於下列任何一種資料格式：固定長度字元、變化長度字元與數值。

   a. 對的，只要長度正確。

2. 是非題：數值內的比例值是代表整個數值的長度。

   a. 錯的，精確值是總長度，比例值表示向右保留數目到小數點。

3. 全部廠商的 SQL 都使用同樣的資料格式嗎？

   a. 不，大多數的廠商他們的資料格式的使用不一致，被 ANSI 規定的資料格式是重要的，但是可能因每個廠商儲存的方式不同，所以在廠商之間有所不同。

4. 下列的精準值與比例值為何？

   a. DECIMAL (4,2) 精準值 4，比例值 2

   b. DECIMAL (10,2) 精準值 10，比例值 2

   c. DECIMAL (14,1) 精準值 14，比例值 1

5. 5. 下列哪一個數字可以被用在欄位定義為 DECIMAL (4,1)？

   a. 16.2

   b. 116.2

   c. 16.21

   d. 1116.2

   e. 1116.21

前三個合適，雖然 16.21 是被四捨五入的，1116.2 和 1116.21 超過最大的精確值，最大的精確值被設定在 4。

C-2-2 練習題答案

1. 使用下列欄位名稱，定義他們的資料格式與決定長度：

   a. ssn constant-length character

   b. city constant-length character

   c. state varying-length character

   d. zip constant-length character

   e. phone_number constant-length character

   f. last_name varying-length character

   g. first_name varying-length character

   h. middle_name varying-length character

   i. salary numeric-datatype

   j. hourly_pay_rate decimal

   k. date_hired date

2. 使用同樣的欄位名稱並定義他們為 null 或 NOT NULL，有些欄位通常是 NOT NULL，有些欄位通常是由應用程式決定為 NULL 或兩者皆可。

   a. ssn NOT NULL

   b. state NOT NULL

   c. city NOT NULL

   d. phone_number NULL

   e. zip NOT NULL

   f. last_name NOT NULL

   g. first_name NOT NULL

   h. middle_name NULL

   i. salary NULL

   j. hourly_pay_rate NULL

   k. date_hired NOT NULL

並非每個都有電話（雖然機會很小），而且不是每個人有中間的名字，所以這些欄應該允許 NULL 價，除此之外，並不是所有的職員的薪資是用每小時時薪來支付。

C-3 第 3 小時：管理資料庫物件

C-3-1 隨堂測驗答案

1. 下列 CREATE TABLE 指令能否正常執行？如果不行，請修正該修正的地方。

```
a.   create table employee_table as
       ( SSN          NUMBER(9)     NOT NULL,
LAST_NAME     VARCHAR2(20)     NOT NULL
    FIRST_NAME     VARCHAR2(20)     NOT NULL,
    MIDDLE_NAME     VARCHAR2(20)     NOT NULL,
    ST ADDRESS     VARCHAR2(30)     NOT NULL,
    CITY          CHAR(20)     NOT NULL,
    STATE          CHAR2)          NOT NULL,
    ZIP          NUMBER(4)     NOT NULL,
    DATE HIRED     DATE)
    STORAGE
        (INITIAL          3K,
         next                1k);
```

a. 建立表格指令將不會工作，因為在語法裡有幾個錯誤，被改正的指令如下，先列出不正確的指令再跟著是應該改正的指令。

```
create table employee_table
    ( ssn          number(9)     not null,
 last_name     varchar2(20)     not null,
 first_name     varchar2(20)     not null,
 middle_name     varchar2(20),
 st_address     varchar2(30)     not null,
 city          varchar2(20)     not null,
 state          char(2)          not null,
 zip          number(5)     not null,
 date_hired     date )
 storage
     (initial          3k
      next                 1k);
```

需要做下列的更改：

1. as 不應該出現在這建立表格指令。
2. last_name 欄之後 not null 缺少逗點。
3. 不一定每個人都有中間的名字，所以 middle_name 欄應該是 NULL。
4. 欄 st address 應該是 st_address，成為二個字，資料庫認為 st 為欄名，將會使資料庫尋找有效的資料格式，所以會發現 ADDRESS 是指資料格式。
5. city 欄會運作，但是使用 varchar2 的資料格式是比較好的，如果所有的城市名字是固定的長度，char 也可以。
6. 欄位 state 缺少左括弧。
7. 郵遞區號欄長度應該是 (5)，不是 (4)。
8. 開始上班日期欄應該是 date_hired 有底線，為了要使欄名稱成為連續字串。
9. 儲存子句裡 3k 後面的逗點，不應該在那裡。

2. 您可以從表格取消一個 column 嗎？
   a. 對的。然而，即使是 ANSI 標準，您必須檢查您的廠商，看看是否它接受這樣的指令。
3. 如果您沒有放入 storage clause 在 create table 指令內，會發生什麼？
   a. CREATE TABLE 指令應該會處理，當然禁止任何語法上的錯誤；然而，大多數的廠商都有預設值，檢查您特別的廠商的預設值大小。

C-4 第 4 小時：正常化程序

C-4-1 隨堂測驗答案

1. 是非題：正常化是將組群資料分成為合乎邏輯與有關聯組群的程序。

   a. 對的。

2. 是非題：資料庫裡沒有重複或多餘的資料，資料庫裡每件事物被正常化是最好方法。

   a. 錯的。正常化會減慢執行效率，因為越多加入表格，就增加 CPU 和 I/O 處理的時間。

3. 是非題：如果資料是在第三正常形式，它自動地是在第一與第二正常形式。

   a. 對的。

4. 什麼是正常化的資料庫主要大於 denormalized 資料庫的優點？

   a. 增加執行效率。

5. Denormalization 的主要缺點是什麼？

   a. 有多餘和複製的資料，會暫去有價值的空間；很難去編碼，而且較多的資料需要被維護。

C-4-2 練習題答案

1. 員工：

   Angela Smith, secretary, 317-545-6789, RR 1 Box 73, Greensburg, Indiana, 47890, $9.50 hour, date started January 22, 1996, SSN is 323149669.

   Jack Lee Nelson, salesman, 3334 N Main St, Brownsburg, IN, 45687, 317-852-9901, salary of $35,000.00 year, SSN is 312567342, date started 10/28/95.

客戶：

Roberts Games and Things, 5612 Lafayette Rd, Indianapolis, IN, 46224, 317-291-7888, customer ID is 432A.

Reeds Dairy Bar, 4556 W 10th St, Indianapolis, IN, 46245, 317-271-9823, customer ID is 117A.

客戶訂單：

Customer ID is 117A, date of last order is February 20, 1997, product ordered was napkins and the product ID is 661.

a.

Employees	Customers	Orders
ssn	customer ID	customer ID
name	name	product ID
street address	street address	product
city	city	date ordered
state	state	
zip	zip	
phone number	phone number	
salary		
hourly pay		
start date		
position		

C-5 第 5 小時：處理資料

C-5-1 隨堂測驗答案

1. 使用下列 employee_tbl 的結構：

column	datatype	(not) null
last_name	varchar2(20)	not null
first_name	varchar2(20)	not null
ssn	char(9)	not null
phone	number(10)	null

LAST_NAME	FIRST_NAME	SSN	PHONE
SMITH	JOHN	312456788	3174549923
ROBERTS	LISA	232118857	3175452321
SMITH	SUE	443221989	3178398712
PIERCE	BILLY	310239856	3176763990

如果執行下列各項指令，將會發生甚麼結果？

a. insert into employee_tbl
('JACKSON', 'STEVE', '313546078', '3178523443');

a. insert 指令將不會執行，因為缺少了關鍵字 values。

b. insert into employee_tbl values
('JACKSON', 'STEVE', '313546078', '3178523443');

a. 一個列會被插入到 employee_tbl。

c. insert into employee_tbl values
('MILLER', 'DANIEL', '230980012', NULL);

a. 一個列會被插入到 employee_tbl，而界電話欄會是 NULL。

d. insert into employee_tbl values
('AYLOR', NULL, '445761212', '3179221331');

a. insert 指令將不會執行，因為 first_name 為 NOT NULL。

e. delete from employee_tbl;

a. employee_tbl 的所有列會被刪除。

f. delete from employee_tbl
where last_name = 'SMITH';

a. 在 employee_tbl，所有姓 SMITH 的列會被刪除。

g. delete from employee_tbl
where last_name = 'SMITH'
and first_name = 'JOHN';

a. 在 employee_tbl，叫 JOHN SMITH 會被刪除。

h. update employee_tbl
set last_name = 'CONRAD';

a. 所有的姓會被改成 CONRAD。

i. update employee_tbl
set last_name = 'CONRAD'
where last_name = 'SMITH';

a. JOHN SMITH 和 SUE SMITH 會變成 JOHN CONRAD 和 SUE CONRAD。

j. update employee_tbl
set last_name = 'CONRAD',
first_name = 'LARRY';

a. 所有員工都叫 LARRY CONRAD。

k. update employee_tbl
set last_name = 'CONRAD',
first_name = 'LARRY'
where ssn = '313546078';

a. JOHN SMITH 現在被稱為 LARRY CONRAD。

C-5-2 練習題答案

2. 使用下列 employee_tbl 的結構：

column	datatype	(not) null
last_name	varchar2(20)	not null
first_name	varchar2(20)	not null
ssn	char(9)	not null
phone	number(10)	null

LAST_NAME	FIRST_NAME	SSN	PHONE
SMITH	JOHN	312456788	3174549923
ROBERTS	LISA	232118857	3175452321
SMITH	SUE	443221989	3178398712
PIERCE	BILLY	310239856	3176763990

寫 DML 來確實完成下列各項：

a. Billy Pierces 正確的 ssn 為 310239857。

   a.
```
update employee_tbl
set ssn = '310239857'
where ssn = '310239856';
```

b. 增加 Ben Moore，phone number 是 317-5649880，ssn 是 313456789。

   a.
```
insert into employee_tbl values
('MOORE', 'BEN', '313456789', '3175649880');
```

c. John Smith 辭職；刪除他的記錄。

   a.
```
delete from employee_tbl
where ssn = '312456788';
```

C-6 第 6 小時：管理資料庫異動

C-6-1 隨堂測驗答案

1. 是非題：如果您已經委託幾個異動，但仍有許多異動尚未完成，那麼在執行 ROLLBACK 指令的同時您所做的異動並未完成。

   a. 錯的。當處理被委託時， 處理不能夠被 ROLL BACK。

2. 是非題：在一定的異動已經執行完成之後，savepoint 實際上已經儲存異動。

   a. 錯的。savepoint 只被當做個點，以便使用 savepoint。

3. 簡短地描述下列各項指令的目的：COMMIT、ROLLBACK 和 SAVEPOINT。

   a. COMMIT 儲存被處理所做的變化。ROLLBACK 復原被處理所做的變化。SAVEPOINT 建立處理合乎邏輯點，知道異動從那開始。

C-6-2 練習題答案

1. 利用下列各項異動，在每三個異動之後帶建立 savepoints，然後 COMMIT 那些異動。

```
savepoint savepoint1
transaction1;
transaction2;
transaction3;
savepoint savepoint2
transaction4;
transaction5;
transaction6;
savepoint savepoint3
transaction7;
transaction8;
transaction9;
savepoint savepoint4
transaction10;
transaction11;
transaction12;
commit;
```

C-7 第 7 小時：介紹資料庫查詢

C-7-1 隨堂測驗答案

1. 說明任何 SELECT 指令所需要的部分。

   a. SELECT 和 FROM 關鍵字，也稱子句，為所有 SELECT 指令所需要。

2. 在 WHERE 子句，是否所有資料需要單一括號？

   a. 當選擇文字數值為資料格式的時候，單括弧號是被需要的，數值資料格式，不需要單括弧號。

3. 在 SQL 語言部份，SELECT 指令（資料庫查詢）所扮演什麼角色？

   a. SELECT 指令是資料查詢語言。

4. 多重條件能被用在 WHERE 子句嗎？

   a. 是的。多重條件能在 SELECT、WHERE、INSERT、UPDATE 與 DELETE 指令被指定，多重條件與運算子 AND 和 OR 一起使用。

C-7-2 練習題答案

1. 檢視下列的 SELECT 指令，檢查語法是否正確的，如果語法是不正確的，如何修正它？應用表格 employee_tbl。

   a.

```
SELECT employee_id, last_name, first_name,
FROM EMPLOYEE_TBL;
```

   b.

```
SELECT EMPLOYEE_ID, LAST_NAME
ORDER BY EMPLOYEE TBL
FROM EMPLOYEE_TBL;
```

   c.

```
SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME
FROM EMPLOYEE_TBL
WHERE EMPLOYEE ID = '333333333'
ORDER BY EMPLOYEE_ID;
```

d.

```
SELECT EMPLOYEE_ID SSN, LAST_NAME
FROM EMPLOYEE_TBL
WHERE EMPLOYEE ID = '333333333'
ORDER BY 1;
```

e.

```
SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME
FROM EMPLOYEE_TBL
WHERE EMPLOYEE ID = '333333333'
ORDER BY 3, 1, 2;
```

a. 因為在 first_name 欄之後有逗點，所以這個 SELECT 指令不會工作，正確的語法為：

```
SELECT employee_id, last_name, first_name
     FROM EMPLOYEE_TBL;
```

b. 這個 SELECT 指令不會工作，因為 FROM 與 ORDER BY 子句是不正確的排序。正確的語法為：

```
SELECT EMPLOYEE_ID, LAST_NAME
FROM EMPLOYEE_TBL
 ORDER BY EMPLOYEE_ID;
```

c. 這個 SELECT 指令的語法是正確的。

d. 這個 SELECT 指令的語法是正確的，注意 employee_id 欄被重新命名 SSN。

e. 對的。對 SELECT 指令語法是正確的，請注意 ORDER BY 裡欄的排序，這個 SELECT 指令傳回 first_name、employee_id，然後 last_name。

C-8 第 8 小時：使用運算子來類別資料

C-8-1 隨堂測驗答案

1. 是非題：在使用 OR 運算子，兩者的條件必須都是正確的。

   a. 錯的。只有其中一個條件必須是真實的。

2. 是非題：使用 IN 運算子時，所有的指定數值必須相配。

   a. 錯的。只有其中一個數值必須相配。

3. 是非題：AND 運算子可以用在 SELECT 和 WHERE 子句。

   a. 錯的。AND 只能被用在 WHERE 子句。

4. 下列哪一個 SELECT 指令是錯誤的？

   a.

```
SELECT SALARY
FROM EMPLOYEE_PAY_TBL
WHERE SALARY BETWEEN 20000, 30000
```

   a. 在 20000、30000 之間缺少 AND，正確語法如下：

```
SELECT SALARY
FROM EMPLOYEE_PAY_TBL
WHERE SALARY BETWEEN 20000 AND 30000
```

   b.

```
SELECT SALARY + DATE_HIRE
FROM EMPLOYEE_PAY_TBL
```

   b. date_hire 欄是日期資料格式與不正確的運算函數。

   c.

```
SELECT SALARY, BONUS
FROM EMPLOYEE_PAY_TBL
WHERE DATE_HIRE BETWEEN 22-SEP-97
AND 23-NOV-97
AND POSITION = 'SALES'
OR POSITION = 'MARKETING'
AND EMPLOYEE_ID LIKE %55%
```

   c. 語法是正確的。

C-8-2 練習題答案

1. 使用 CUSTOMER_TBL 做描述：

```
describe customer_tbl
```

```
Name                           Null?    Type
------------------------------ -------- ----
CUST_ID                        NOT NULL VARCHAR2(10)
CUST_NAME                      NOT NULL VARCHAR2(30)
CUST_ADDRESS                   NOT NULL VARCHAR2(20)
CUST_CITY                      NOT NULL VARCHAR2(12)
CUST_STATE                     NOT NULL CHAR(2)
CUST_ZIP                       NOT NULL CHAR(5)
CUST_PHONE                              NUMBER(10)
CUST_FAX                                NUMBER(10)
```

寫出 SELECT 指令來傳回客戶 ID 和居住在 Indiana，Ohio，Michigan 與 Illinois 客戶和客戶名稱開始值為 A 或 B 的客戶名稱（用字母排序）。

a.

```
SELECT CUST_ID, CUST_NAME, CUST_STATE
FROM CUSTOMER_TBL
WHERE CUST_STATE IN ('IN', 'OH', 'MI', 'IL')
AND CUST_NAME LIKE 'A%'
OR CUST_NAME LIKE 'B%'
ORDER BY CUST_NAME
```

2. 使用 PRODUCTS_TBL 為描述：

```
describe products_tbl
```

```
Name                           Null?    Type
------------------------------ -------- ----
PROD_ID                        NOT NULL VARCHAR2(10)
PROD_DESC                      NOT NULL VARCHAR2(25)
COST                           NOT NULL NUMBER(6,2)
```

寫出 SELECT 指令來傳回產品 ID、產品敘述與產品價值，限制產品價值在 $1.00 和 $12.50 之間。

a.

```
SELECT *
FROM PRODUCTS_TBL
WHERE COST BETWEEN 1.00 AND 12.50
```

C-9 第 9 小時：資料查詢結果總結

C-9-1 隨堂測驗答案

1. 是非題：AVG 函數傳回所有選定列的平均值包括任何的 NULL 數值。

   a. 錯的。那些 NULL 是不被考慮。

2. 是非題：SUM 函數被使用在增加欄的總計。

   a. 錯的。SUM 函數用來傳回一群列的總數。

3. 是非題：COUNT(*) 函數統計表格裡的所有列。

   a. 對的。

4. 下列的 SELECT 指令會正常工作？如果不可以，哪些指令需要修改？

   a.

```
SELECT COUNT *
FROM EMPLOYEE_PAY_TBL;
```

   a. 這指令將不會工作，因為在 * 左右缺少刮號，正確的語法為：

```
SELECT COUNT(*)
FROM EMPLOYEE_PAY_TBL;
```

   b.

```
SELECT COUNT(EMPLOYEE_ID), SALARY
FROM EMPLOYEE_PAY_TBL;
```

   a. 這個語法正確。.

   c.

```
SELECT MIN(BONUS), MAX(SALARY)
FROM EMPLOYEE_PAY_TBL
WHERE SALARY > 20000;
```

   a. 這個語法正確。

C-9-2 練習題答案

1. 使用 EMPLOYEE_PAY_TBL：

```
EMP_ID    POSITION       DATE_HIRE PAY_RATE DATE_LAST SALARY BONUS
--------- -------------- --------- -------- --------- ------ -----
311549902 MARKETING      23-MAY-89          01-MAY-97  30000  2000
442346889 TEAM LEADER    17-JUN-90    14.75 01-JUN-97
213764555 SALES MANAGER  14-AUG-94          01-AUG-97  40000  3000
313782439 SALESMAN       28-JUN-97                     20000  1000
220984332 SHIPPER        22-JUL-96       11 01-JUL-97
443679012 SHIPPER        14-JAN-91       15 01-JAN-97

6 rows selected.
```

建立 SQL 指令來找尋：

a. The average salary.

a. 平均 salary 為 $30,000，用下列 SQL 指令可以找出。

```
SELECT AVG(SALARY)
FROM EMPLOYEE_PAY_TBL;
```

b. The maximum bonus.

a. 最大 bonus 為 $3000.00，用下列 SQL 指令可以找出。

```
SELECT MAX(BONUS)
FROM EMPLOYEE_PAY_TBL;
```

c. The total salaries.

a. salaries 總和為 $60,000.00 用下列 SQL 指令可以找出。

```
SELECT SUM(SALARY)
FROM EMPLOYEE_PAY_TBL;
```

d. The minimum pay rate.

a. 最小 pay rate 是每小時 $11.00，用下列 SQL 指令可以找出。

```
SELECT MIN(PAY_RATE)
FROM EMPLOYEE_PAY_TBL;
```

e. The total rows in the table.

a. 總列數為 6，用下列 SQL 指令可以找出。

```
SELECT COUNT(*)
FROM EMPLOYEE_PAY_TBL;
```

C-10 第 10 小時：資料的分類和排序

C-10-1 隨堂測驗答案

1. 下列各 SQL 指令可以工作嗎？

a.

```
SELECT SUM(SALARY), EMP_ID
FROM EMPLOYEE_PAY_TBL
GROUP BY 1 and 2;
```

a. 不，這個指令不會工作，GROUP BY 子句不屬於那裡，而且在 GROUP BY 子句旁邊不能夠使用整數，正確的語法是：

```
SELECT SUM(SALARY),
FROM EMPLOYEE_PAY_TBL
GROUP BY SALARY, EMP_ID;
```

b.

```
SELECT EMP_ID, MAX(SALARY)
FROM EMPLOYEE_PAY_TBL
GROUP BY SALARY, EMP_ID;
```

a. 是的，這個指令將會工作。

c.

```
SELECT EMP_ID, COUNT(SALARY)
FROM EMPLOYEE_PAY_TBL
ORDER BY EMP_ID
GROUP BY SALARY;
```

a. 不，這個指令將不會工作，ORDER BY 子句和 GROUP BY 子句排序不正確，正確的語法是：

```
SELECT EMP_ID, COUNT(SALARY)
FROM EMPLOYEE_PAY_TBL
GROUP BY SALARY
ORDER BY EMP_ID;
```

2. 是非題：當使用 HAVING 子句的時候，您也必須使用 GROUP BY 子句？

a. 錯的。HAVING 子句可以不和 GROUP BY 子句一起使用。

3. 是非題：這 SQL 指令傳回薪資總數群組。

```
SELECT SUM(SALARY)
FROM EMPLOYEE_PAY_TBL;
```

a. 錯的。指令不能夠傳回薪水總數，因為沒有 GROUP BY 子句。

4. 是非題：被選擇欄的順序，必須與 GROUP BY 子句排序相同。

a. 錯的。SELECT 子句裡欄的排序可以和 GROUP BY 子句不同。

5. 是非題：HAVING 子句告訴 GROUP BY 子句必須包含那一個群組 。

a. 對的。

C-10-2 練習題答案

1. 寫個能從 EMPLOYEE_TBL 傳回員工 ID、員工名稱和城市的 SQL 指令，並藉由城市欄先聚集。

a.

```
select emp_id, last_name, first_name, city
from employee_tbl
group by city, emp_id, last_name, first_name
```

2. 寫個能從 EMPLOYEE_TBL 傳回城市和統計每座城市的員工的 SQL 指令，增加 HAVING 子句，只有顯示那些有超過二位員工所被統計的城市。

a.

```
select city, count(emp_id)
from employee_tbl
group by city
having count(emp_id) > 2
```

C-11 第 11 小時：更改資料展現結構

C-11-1 隨堂測驗答案

1. 描述與可能相配的函數。

	DESCRIPTIONS	SOLUTION 答案
a.	使用選擇部分字串	SUBSTR
b.	從字元左邊或右邊截割字元	LTRIM/ RTRIM
c.	更改所有字元爲小寫	LOWER
d.	找尋字元長度	LENGHT
e.	組合字元	CONCATENTION (||)

2. 是非題：SOUNDEX 函數用來比較可能聽起來一樣的字串。

   a. 真的。

3. 是非題：當在查詢裡的函數箝入其他函數裡面的時候，最外圍的函數總是首先處理。

   a. 錯的。當函數相互箝入時，內部函數總是先處理。

C-11-2 練習題答案

1. 使用適當的函數轉換字串 “hello” 為大寫字母。

   a.

```
SELECT UPPER('hello') FROM TABLE_NAME
```

2. 使用適當的函數，只列印字串 Johnson 的最初四個字元。

3. 使用函數去串接般字串 JOHN 和 SON。

   a. Oracle

```
SELECT 'JOHN' || 'SON' FROM TABLE_NAME
```

   或

   a. SQL Server

```
SELECT 'JOHN' + 'SON' FROM TABLE_NAME
```

C-12 第 12 小時：理解日期與時間

C-12-1 隨堂測驗答案

1. 通常系統日期與時間在哪裡可以得到？

   a. 目前日期和時間的系統日期是從主機上的作業系統得到的。

2. 列出 DATETIME 內在元件標準值。

   a. YEAR、MONTH、DAY、HOUR、MINUTE 與 SECOND。

3. 如果您的公司是國際組織，關於日期與時間的表示方法和比較，什麼可能是主要的因素？

   a. 時區的考量是重要的。。

4. 在有效的 DATETIME 資料格式，字串日期值能與日期值做比較嗎？

   a. DATETIME 資料格式不可能實際地與被定義的字元串的日期數值比較，字元串必須首先被轉換到 DATETIME 資料格式。

C-12-2 練習題答案

1. 藉由下列提供的各項訊息練習編寫 SQL 碼：

   使用 SYSDATE 表是現在的日期而且計時。

   使用表格稱為日期。

   使用 TO_CHAR 函數轉換日期成字串與語法：TO_CHAR(EXPRESSION, DATE_PICTURE)

   使用 TO_DATE 函數轉換字串為日期，與語法：TO_DATE(EXPRESSION, DATE_PICTURE)

   日期圖片資訊：

日期圖片	意義
MONTH	拼出大寫月份
DAY	拼出大寫日期

日期圖片	意義
DD	月內的天數
MM	年內的天數
YY	年的最後二位數字
YYYY	年的四位數字
MI	每小時的分鐘
SS	一分鐘的秒數

1. 假設今天是 1997-12-31，轉換現在日期到格式 December 31 1997。

   a.

```
SELECT TO_CHAR(SYSDATE,'MONTH DD YYYY')
FROM DATES;
```

2. 轉換下列各項字串到日期格式：

```
'DECEMBER 31 1997'
```

   a.

```
SELECT TO_DATE('DECEMBER 31 1997','MONTH DD YYYY')
FROM DATES;
```

3. 編寫碼傳回 1998 新年前夕的那週，哪的天在那天之上。假定那個日期以 31-DEC-97 的格式被儲存是有效的 DATETIME 資料格式。

   a.

```
SELECT TO_CHAR('31-DEC-97','DAY')
FROM DATES;
```

C

C-13 第 13 小時：在查詢裡加入表格

C-13-1 隨堂測驗答案

1. 什麼類型的加入，您將會使用傳回來自一個表格的記錄，不管有聯合關係的記錄存在的表格裡？

   a. 您將會使用 OUTER JOIN。

2. 加入條件位於 SQL 指令的什麼部份？

   a. JOIN 參加條件位於 WHERE 子句。

3. 什麼類型的加入，您會使用評估有關係表格列之中的同等性？

   a. 您將會使用 EQUIJOIN。

4. 如果您從二個不同的表格選擇，但是那些表格加入失敗，會發生什麼？

   a. 您收到 Cartesian Productu 有關不參加的表格（這也被叫做交叉參加）。

5. 使用那些下列各項表格：

ORDERS_TBL

ord_num	varchar2(10)	not null	primary key
cust_id	varchar2(10)	not null	
prod_id	varchar2(10)	not null	
qty	number(6)	not null	
ord_date	date		

PRODUCTS_TBL

prod_id	varchar2(10)	not null	primary key
prod_desc	varchar2(40)	not null	
cost	number(6,2)	not null	

請檢查使用外部加入的語法？

```
SELECT C.CUST_ID, C.CUST_NAME, O.ORD_NUM
FROM CUSTOMER_TBL C, ORDERS_TBL O
WHERE C.CUST_ID(+) = O.CUST_ID(+)
```

a. 不，語法不正確。運算子 (+) 只應該跟隨 O.CUST_ID 欄在那哪裡子句，正確的語法是：

```
SELECT C.CUST_ID, C.CUST_NAME, O.ORD_NUM
FROM CUSTOMER_TBL C, ORDERS_TBL O
WHERE C.CUST_ID = O.CUST_ID(+);
```

C-13-2 練習題答案

1. 使用下列表格來做練習：

EMPLOYEE_TBL

emp_id	varchar2(9)	not null	PRIMARY KEY
last_name	varchar2(15)	not null	
first_name	varchar2(15)	not null	
middle_name	varchar2(15)		
address	varchar2(30)	not null	
city	varchar2(15)	not null	
state	char(2)	not null	
zip	number(5)	not null	
phone	char(10)		
pager	char(10)		

EMPLOYEE_PAY_TBL

emp_id	varchar2(9)	not null	primary key
position	varchar2(15)	not null	
date_hire	date		
pay_rate	number(4,2)	not null	
date_last_raise	date		
salary	number(6,2)		
bonus	number(4,2)		

```
constraint emp_fk foreign key  (emp_id) references
  employee_tbl (emp_id)
```

CUSTOMER_TBL

cust_id	varchar2(10)	not null	primary key
cust_name	varchar2(30)	not null	
cust_address	varchar2(20)	not null	

C

CUSTOMER_TBL

cust_city	varchar2(15)	not null	
cust_state	char(2)	not null	
cust_zip	number(5)	not null	
cust_phone	number(10)		
cust_fax	number(10)		

ORDERS_TBL

ord_num	varchar2(10)	not null	primary key
cust_id	varchar2(10)	not null	
prod_id	varchar2(10)	not null	
qty	number(6)	not null	
ord_date	date		

PRODUCTS_TBL

prod_id	varchar2(10)	not null	primary key
prod_desc	varchar2(40)	not null	
cost	number(6,2)	not null	

1. 寫 SQL 指令來傳回來自 EMPLOYEE_TBL 的 emp_id、last_name 和 first_name 與來自 EMPLOYEE_PAY_TBL 的 BOUNS 和 salary。

a.

```
SELECT E.EMP_ID, E.LAST_NAME, E.FIRST_NAME
         EP.SALARY, EP.BONUS
    FROM EMPLOYEE_TBL E,
         EMPLOYE_PAY_TBL EP
    WHERE E.EMP_ID = EP.EMP_ID
```

2. 選擇來自 CUSTOMERS_TBL 的 CUST_ID、CUST_NAME，選擇從 PRODUCTS_TBL 的 PROD_ID、COST，選擇從 ORDERS_TBL 的 ORD_NUM 和 QTY，使用一個 SQL 指令加入所有三個表格。

a.

```
SELECT C.CUST_ID, C.CUST_NAME, P.PROD_ID, P.COST,
  O.ORD_NUM, O.QTY
  FROM CUSTOMER_TBL C,
       PRODUCT_TBL P,
       ORDERS_TBL O
  WHERE C.CUST_ID = O.CUST_ID
  AND P.PROD_ID = O.PROD_ID
```

C-14 第 14 小時：使用子查詢定義未知的資料

C-14-1 隨堂測驗答案

1. 當與 SELECT 指令用的時候，子查詢的函數是什麼？

   a. subquery 的主要功能是與 SELECT 指令一起使用的時候，是返回資料給主查詢，以使用查詢。

2. 當使用子查詢 UPDATE 指令的時候，您能更新超過一個欄嗎？

   a. 是的，您能使用相同的 UPDATE 和 subquery 指令，更新多個欄位的資料。

3. 下列各項語法是正確的嗎?如果不是，正確的語法是什麼？

   a.

```
SELECT CUST_ID, CUST_NAME
        FROM CUSTOMER_TBL
        WHERE CUST_ID =
                    (SELECT CUST_ID
                            FROM ORDERS_TBL
                            WHERE ORD_NUM = '16C17')
```

   a. 是的，這個語法是正確的。

   b.

```
SELECT EMP_ID, SALARY
       FROM EMPLOYEE_PAY_TBL
       WHERE SALARY BETWEEN 20000
                    AND (SELECT SALARY
```

C

```
                    FROM EMPLOYEE_ID
                    WHERE SALARY = 40000)
```

a. 不，BETWEEN 運算子不能在這格式使用。

c.

```
UPDATE PRODUCTS_TBL
   SET COST = 1.15
   WHERE CUST_ID =
                  (SELECT CUST_ID
                   FROM ORDERS_TBL
                   WHERE ORD_NUM = '32A132')
```

a. 是的，這個語法是正確的。

4. 如果下列各項指令執行後將會發生什麼事？

```
DELETE FROM EMPLOYEE_TBL
WHERE EMP_ID IN
             (SELECT EMP_ID
             FROM EMPLOYEE_PAY_TBL)
```

a. 從 EMPLOYEE_PAY_TBL 被取回所有的列，將會被 DELETE 將他們從 EMPLOYEE_TBL 移開，subquery 裡 WHERE 子句有提出相關的建議。

C-14-2 練習題答案

1. 使用下列表格：

EMPLOYEE_TBL

emp_id	varchar2(9)	not null	primary key
last_name	varchar2(15)	not null	
first_name	varchar2(15)	not null	
middle_name	varchar2(15)		
address	varchar2(30)	not null	
city	varchar2(15)	not null	
state	char(2)	not null	
zip	number(5)	not null	

EMPLOYEE_TBL

phone	char(10)
pager	char(10)

EMPLOYEE_PAY_TBL

emp_id	varchar2(9)	not null	primary key
position	varchar2(15)	not null	
date_hire	date		
pay_rate	number(4,2)	not null	
date_last_raise	date		

```
constraint emp_fk foreign key  (emp_id) references
  employee_tbl (emp_id)
```

CUSTOMER_TBL

cust_id	varchar2(10)	not null	primary key
cust_name	varchar2(30)	not null	
cust_address	varchar2(20)	not null	
cust_city	varchar2(15)	not null	
cust_state	char(2)	not null	
cust_zip	number(5)	not null	
cust_phone	number(10)		
cust_fax	number(10)		

ORDERS_TBL

ord_num	varchar2(10)	not null	primary key
cust_id	varchar2(10)	not null	
prod_id	varchar2(10)	not null	
qty	number(6)	not null	
ord_date	date		

PRODUCTS_TBL			
prod_id	varchar2(10)	not null	primary key
prod_desc	varchar2(40)	not null	
cost	number(6,2)	not null	

2. 使用子查詢，寫 SQL 指令更新 CUSTOMER_TBL 表格，變更客戶名字為 DAVID MARKET，訂單號碼為 23E934。

a.

```
UPDATE CUSTOMER_TB
   SET CUST_NAME = 'DAVIDS MARKET'
   WHERE CUST_ID =
                  (SELECT CUST_ID
                   FROM ORDERS_TBL
                   WHERE ORD_NUM = '23E934'
```

3. 使用子查詢，寫傳回所有員工的 pay rate 比 JOHN DOE 多與誰的員工編號是 343559876 的查詢。

a.

```
SELECT E.LAST_NAME, E.FIRST_NAME, E.MIDDLE_NAM
   FROM EMPLOYEE_TBL E,
        EMPLOYEE_PAY_TBL P
   WHERE P.PAY_RATE > (SELECT PAY_RATE
                       FROM EMPLOYEE_PAY_TBL
                       WHERE EMP_ID = 343559876);;
```

4. 使用子查詢，寫列出 COST 超過所有產品平均值的所有產品的查詢。

a.

```
SELECT PROD_DESC
   FROM PRODUCTS_TBL
   WHERE COST > (SELECT AVG(COST)
                 FROM PRODUCTS_TBL);
```

C-15 第 15 小時：組合多重查詢

C-15-1 隨堂測驗答案

1. 下列合成查詢語法是否正確？如果不，如何改正語法？使用下列的 EMPLOYEE_TBL 和 EMPLOYEE_PAY_TBL：

```
EMPLOYEE_TBL
-------------------------------------------------------
emp_id          varchar2(9)    not null,
last_name       varchar2(15)   not null,
first_name      varchar2(15)   not null,
middle_name     varchar2(15)
                ,
address         varchar2(30)   not null,
city            varchar2(15)   not null,
state           char(2)        not null,
zip             number(5)      not null,
phone           char(10),
pager           char(10),
constraint emp_pk primary key (emp_id)
```

```
EMPLOYEE_PAY_TBL
-------------------------------------------------------
emp_id              varchar2(9)    not null,  primary key,
position            varchar2(15)   not null,
date_hire           date,
pay_rate            number(4,2)    not null,
date_last_raise     date,
salary              number(8,2),
bonus               number(8,2),
constraint emp_fk foreign key (emp_id)
references employee_tbl (emp_id)
```

a.

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME
FROM EMPLOYEE_TBL
UNION
SELECT EMP_ID, POSITION, DATE_HIRE
FROM EMPLOYEE_PAY_TBL
```

a. 這因為資料格式不符合，所以複合查詢不工作。EMP_ID 欄符合，但是 LAST_NAME 和 FIRST_NAME 資料格式不符合 POSITION 和 DATE_HIRE 資料格式。

b.

```
SELECT EMP_ID FROM EMPLOYEE_TBL
UNION ALL
SELECT EMP_ID FROM EMPLOYEE_PAY_TBL
ORDER BY EMP_ID
```

a. 是的，指令是正確的。

c.

```
SELECT EMP_ID FROM EMPLOYEE_PAY_TBL
INTERSECT
SELECT EMP_ID FROM EMPLOYEE_TBL
ORDER BY 1
```

a. 是的，這合成查詢會工作。

2. 對應下列各指令到正確的運算子。

指令	運算子
a.	展示重複 a. ALL
b.	傳回唯一與第一個查詢和在第二個查詢對應的列 a. INTERSECT
c.	傳回沒有重複 a. UNION
d.	傳回唯一被第一個查詢選擇而不被第二個查詢選擇的列 a. EXCEPT

C-15-2 練習題答案

1. 使用 CUSTOMER_TBL 和 ORDERS_TBL 列出：

CUSTOMER_TBL

cust_id	varchar2(10)	not null	primary key,
cust_name	varchar2(30)	not null,	
cust_address	varchar2(20)	not null,	
cust_city	varchar2(15)	not null,	
cust_state	char(2)	not null,	
cust_zip	number(5)	not null,	
cust_phone	number(10),		
cust_fax	number(10)		

ORDERS_TBL

ord_num	varchar2(10)	not null	primary key,
cust_id	varchar2(10)	not null,	
prod_id	varchar2(10)	not null,	
qty	number(6)	not null,	
ord_date	date		

a. 寫個合成查詢找尋已經下訂單的客戶。

a.

```
SELECT CUST_ID FROM CUSTOMER_TBL
INTERSECT
SELECT CUST_ID FROM ORDERS_TBL
```

b. 寫個合成查詢找尋那些還沒有下訂單的客戶。

a.

```
SELECT CUST_ID FROM CUSTOMER_TBL
EXCEPT
SELECT CUST_ID FROM ORDERS_TBL
```

C-16 第 16 小時：使用索引增加執行效果

C-16-1 隨堂測驗答案

1. 甚麼是使用索引的主要的缺點？

   a. 索引主要的缺點包括減慢 batch，磁碟儲存空間，和索引維護。

2. 為什麼在複合索引欄的排序是重要的？

   a. 因為查詢執行效率，藉由先限制表格欄的值。

3. 有許多 NULL 的欄可以被編入索引嗎？

   a. 不，因為當列的符合的百分率較高時，存取這些列的速度降低，所以符合的百分率較高的欄不應該被編入索引。

4. 表格裡索引主要的目的是停止複製值。

   a. 不，索引的主要目的是提高資料檢索速度，但是停止表格裡副本的複製也是其中的功能。

5. 5. 是非題：複合索引主要的理由是為了在索引使用聚合函數。

   a. 錯誤的。主要的理由有合成索引是因為相同表格裡有二個或較多欄被索引的。

C-16-2 練習題答案

1. 1. 下列各項情形，決定是否該使用索引，如果該使用，什麼類型的索引應該被使用。

   a. 幾個欄，但是較小的表格。

      a. 成為非常小的表格就不需要索引。

   b. 中型表格，複製不被允許。

      a. 使用唯一索引。

   c. 幾個欄，非常大的表格，幾個欄在 WHERE 子句，當做過濾條件。

      a. 欄上合成索引在 WHERE 子句使用，當做過濾。

d. 大的表格，許多欄與許多資料處理。

a. 單一欄或合成索引選擇應該被考慮，但也因過濾而定，排序和群組而定。對於大量的資料處理，索引能夠被取消與建立，一旦插入，更新或刪除等工作被執行。

C-17 第 17 小時：增進資料庫執行效率

C-17-1 隨堂測驗答案

1. 在小表格上使用唯一索引有何益處？

   a. 索引可能對執行效率沒有任何用處；但是，唯一索引將會保存參考完整性尚未被人碰過。參考完整性在第 3 小時被討論過。

2. 當查詢已經被執行的時候 optimizer 選擇不要使用表格上的索引時會發生什麼？

   a. 會執行完整表格掃描。

3. 在 WHERE 子句，最限制的子句是被放置在加入條件之前或之後？

   a. 最限制的子句應該是之前所評估的參加條件，自那時後，正常地參加條件傳回很多的列。

C-17-2 練習題答案

1. 重寫下列各 SQL 指令，增進他們的執行效率。使用在這所描述的 EMPLOYEE_TBL 和 EMPLOYEE_PAY_TBL：

```
EMPLOYEE_TBL
-------------------------------------------------
emp_id          varchar2(9)       not null,
last_name       varchar2(15)      not null,
first_name      varchar2(15)      not null,
middle_name     varchar2(15),
address         varchar2(30)      not null,
```

C

```
EMPLOYEE_TBL
city                varchar2(15)        not null,
state               char(2)             not null,
zip                 number(5)           not null,
phone               char(10),
pager               char(10),
constraint emp_pk primary key (emp_id)

EMPLOYEE_PAY_TBL
emp_id              varchar2(9)         not null,
position            varchar2(15)        not null,
date_hire           date,
pay_rate            number(4,2)         not null,
date_last_raise     date,
salary              number(8,2),
bonus               number(8,2),
  constraint emp_fk foreign key (emp_id)
  references employee_tbl (emp_id)
```

a.

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME,
         PHONE
   FROM EMPLOYEE_TBL
   WHERE SUBSTR(PHONE, 1, 3) = '317' OR
         SUBSTR(PHONE, 1, 3) = '812' OR
         SUBSTR(PHONE, 1, 3) = '765'
```

a.

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME,
   FROM EMPLOYEE_TBL
   WHERE SUBSTR(PHONE, 1, 3) IN ('317', '812', '765')
```

b.

```
SELECT LAST_NAME, FIRST_NAME
   FROM EMPLOYEE_TBL
   WHERE LAST_NAME LIKE '%ALL%
```

a.

```
SELECT LAST_NAME, FIRST_NAME
 FROM EMPLOYEE_TBL
 WHERE LAST_NAME LIKE 'WAL%'
```

c.

```
SELECT E.EMP_ID, E.LAST_NAME, E.FIRST_NAME,
          EP.SALARY
   FROM EMPLOYEE_TBL E,
   EMPLOYEE_PAY_TBL EP
   WHERE LAST_NAME LIKE 'S%'
   AND E.EMP_ID = EP.EMP_ID
```

a.

```
SELECT E.EMP_ID, E.LAST_NAME, E.FIRST_NAME,
          EP.SALARY
   FROM EMPLOYEE_PAY_TBL EP,
        EMPLOYEE_TBL E
   WHERE E.EMP_ID = EP.EMP_ID
   AND LAST_NAME LIKE 'S'
```

C-18 第 18 小時：管理資料庫使用者

C-18-1 隨堂測驗答案

1. 什麼指令用來建立交談期？

   a. CONNECT TO 指令。

2. 那一個選擇項必須用來取消資料庫結構，而 仍然包含資料庫的資料庫結構物件？

   a. CASCADE 選項允許資料庫結構被取消，如果他們依舊是資料庫結構下。

3. 什麼指令用來除去資料庫權限？

   a. REVOKE 指令用來移去資料庫特權。

4. 4. 什麼指令建立一群表格、檢視和權限？

   a. CREATE SCHEMA 指令。

C-18-2 練習題答案

1. 描述或列出允許新職員資料庫存取的步驟。

   a. 立即的監督者器應該提出請求，完成使用者身份證申請表格的處理，使用者身份證申請表格，應包含把使用者所有必需的訊息加入資料庫，然後表格應該被轉寄給安全官員，使用者請求然後被到轉到資料庫管理原或被指定的人，如用資料庫安全協助管理員，所以使用者可以被加入到資料庫，這是每家公司應有的一般程序。

C-19 第 19 小時：資料庫安全管理

C-19-1 隨堂測驗答案

1. 什麼選項使用者必須取得，才能授予另外一個使用者不屬於使用者所擁有的物件特權？

   a. GRANT OPTION。

2. 當特權被 PUBLIC 取到，所有資料庫的使用者獲得那些特權嗎？或只是所敘述的特定使用者而以？

   a. 資料庫所有使用者將會被授予那些特權。

3. 什麼特權需要檢視特定的表格資料？

   a. SELECT 特權。

4. 什麼樣的類型有 SELECT 的權限？

   a. 物件層次特權。

C-19-2 練習題答案

1. 寫個程式能讓使用者 rplew 取得您擁有的表格 employee_tbl 的存取權限，它應該允許 rplew 可以給相同表格上的特權與另外一個使用者。

   a.

   ```
   GRANT SELECT ON EMPLOYEE_TBL TO RPLEW WITH GRANT OPTION
   ```

2. 寫個程式取消來自練習題 1.2 裡，兩個使用者對資料庫的連接。

   a.

   ```
   REVOKE CONNECT ON EMPLOYEE_TBL FROM RPLEW
   ```

3. 寫個程式允許 rplew 去選擇、插入與更新 employee_tbl 表格。

   a.

   ```
   GRANT SELECT, INSERT, UPDATE ON EMPLOYEE_TBL TO RPLEW
   ```

C-20 第 20 小時：建立、使用檢視與同義字

C-20-1 隨堂測驗答案

1. 一個列的資料能從多重表格所建立的檢視中被刪除嗎？

   a. No，DELETE、INSERT 與 UPDATE 指令只能從單一表格所建立的檢視上使用。

2. 當建立表格時，擁有者自動取得那個表格上的適當特權，當建立檢視的時候，也是一樣嗎？

   a. 是的，檢視擁有者自動地被授予檢視上的適當特權。

3. 當建立檢視的時候，哪個子句用來排序資料？

   a. GROUP BY 子句（或 GROUP BY 子句）在檢視的動作與 ORDER BY 子句在一般的查詢一樣的多。

4. 當從檢視建立檢視的時候，檢查限制條件的完整性時，哪個選項能被使用？

   a. WITH CHECK OPTION。

5. 您試著取消檢視，但收到錯誤訊息，是因為在下面有一個或更多的檢視，您必須做什麼來取消檢視？

   a. 從新執行 DROP 指令與 CASCADE 選項，這允許 DROP 指令能成功被取消下面所有的檢視。

C-20-2 練習題答案

1. 編寫建立基於 EMPLOYEE_TBL 表格總內容的檢視指令。

   a.

```
CREATE VIEW EMP_VIEW AS
SELECT * FROM EMPLOYEE_TBL
```

2. 編寫建立檢視摘要，包含 EMPLOYEE_TBL 表格裡的每個城市，平均 Pay Rate 和平均薪水的指令。

   a.

```
CREATE VIEW AVG_PAY_VIEW AS
SELECT E.CITY, AVG(P.PAY_RATE), AVG(P.SALARY)
FROM EMPLOYEE_PAY_TBL P,
EMPLOYEE_TBL E
WHERE P.EMP_ID = E.EMP_ID
GROUP BY E.CITY
```

3. 編寫取消您在練習 1 和 2 所建立的二個檢視的指令。

   a.

```
DROP VIEW EMP_VIEW
DROP VIEW AVG_PAY_VIEW
```

C-21 第 21 小時：系統型錄運作

C-21-1 隨堂測驗答案

1. 系統型錄在一些廠商中也稱為什麼？

   a. 系統目錄也被稱為資料字典。

2. 一般的使用者可以更新系統型錄嗎？

   a. 並不直接地，然而，當使用者建立物件時，例如表格等，系統目錄自動地被更新。

3. Sybase 使用什麼樣的系統表格可以用來取回檢視存在資料庫的訊息？

   a. SYSVIEWS

4. 誰擁有系統型錄？

   a. 系統目錄的擁有者時，通常是有資料庫使用者帳戶特權稱為 SYS 或 SYSTEM，系統目錄也可能被資料庫擁有者所擁有，但通常不是被資料庫裡特別資料庫結構所擁有。

5. 在 Oracle 系統物件 ALL_TABLES 和 DBA_TABLES 之間，有什麼不同？

   a. ALL_TABLES 表示所有表格能被哪些特定使用者所存取，而 DBA_TABLES 表示在資料庫存在的所有的表格。

6. 誰可以修改系統表格？

   a. 資料庫伺服器本身。

C-22 第 22 小時：進階 SQL 主題

C-22-1 隨堂測驗答案

1. 觸發可以改變嗎？

   a. 不可以，觸發必須被替換或從新建立。

2. 當游標被關閉的時候，您能重複使用名稱嗎？

   a. 這是看廠商而定，在一些廠商，關閉游標將會允許您重複使用名字和甚至釋放記憶體，但其他的廠商在名字能被重複使用之前，您必須使用 deallocate 指令。

3. 當游標已經開啟後，用什麼指令來取回結果？

   a. FETCH 指令。

4. 觸發要在 INSERT、DELETE 或 UPDATE 之後或之前執行？

   a. 觸發能在 INSERT、DELETE 或 UPDATE 之後或之前執行，有許多不同類型的觸發能被建立。

C-22-2 練習題答案

1. 使用您的廠商系統目錄表格，建立下列各項 SQL 指令，以真實物件名稱代替物件名稱。

a.

```
GRANT SELECT ON TABLE_NAME TO USERNAME;
```

a.

```
SELECT 'GRANT SELECT ON '||TABLE_NAME|| ' TO ||
USERNAME||';'
FROM SYSTEM CATALOG TABLE_NAME
```

b.

```
GRANT, CONNECT, RESOURCE TO USERNAME;
```

b.

```
SELECT 'GRANT, CONNECT, RESOURCE TO '
||USERNAME||';'
FROM SYSTEM CATALOG TABLE_NAME
```

c.

```
SELECT COUNT(*) FROM TABLE_NAME;
```

c.

```
SELECT 'SELECT COUNT(*) FROM '||TABLE_NAME||';'
FROM SYSTEM CATALOG TABLE_NAME
```

2. 寫個指令，建立儲存程序，用來刪除 PRODUCTS_TBL 表格的項目；它應該類似在這個小時所用的，插入新產品的例子。

a.

```
CREATE PROCEDURE DELETE_PRODUCT
(OLD_PROD_ID IN VARCHAR2)
AS
BEGIN
    DELETE FROM PRODUCTS_TBL
    WHERE PROD_ID = OLD_PROD_ID;
    COMMIT;
END;
/
```

3. 寫個指令，可以執行您在練習 2 所建立的儲存程序，刪除 PROD_ID 為 9999 的列。

a.

```
EXECUTE DELETE_PRODUCT ('9999');
```

C-23 第 23 小時：擴充 SQL 到企業、網際網路和企業內部網路

C-23-1 隨堂測驗答案

1. 伺服器上的資料庫能從另外一個伺服器存取嗎？

   a. 是的，藉由使用 middleware 產品，這被叫做存取遠端資料庫。

2. 什麼能傳播公司員工的訊息？

   a. intranet。

3. 允許到資料庫連接的產品叫做什麼？

   a. Middleware。

4. SQL 能箝入到網際網路程式語言之內嗎？

   a. 是的，SQL 能正如爪哇一樣，能箝入網際網路程式語言裡。

5. 遠端資料庫如何透過 Web 應用程式存取？

   a. 經由 Web 伺服器。

C-24 第 24 小時：標準 SQL 的擴充

C-24-1 隨堂測驗答案

1. SQL 是程序或非程序語言？

   a. SQL 是一非程序的，意謂資料庫決定該如何執行 SQL 指令。

2. 為什麼 SQL 有理由可以存在不同的差異？

   a. SQL 存在不同的差異，是因為廠商之中的存在儲存需求、取得競爭優勢、使用者方便和執行效率等考慮。

3. 什麼是在游標、宣告游標外面的三個基本運算？

   a. OPEN、FETCH 與 CLOSE。

4. 程序或非程序的，哪一個是讓資料庫自己決定該如何評估並且執行 SQL 指令？

   a. 非程序的。

C-24-2 練習題答案

1. 沒有特定答案。

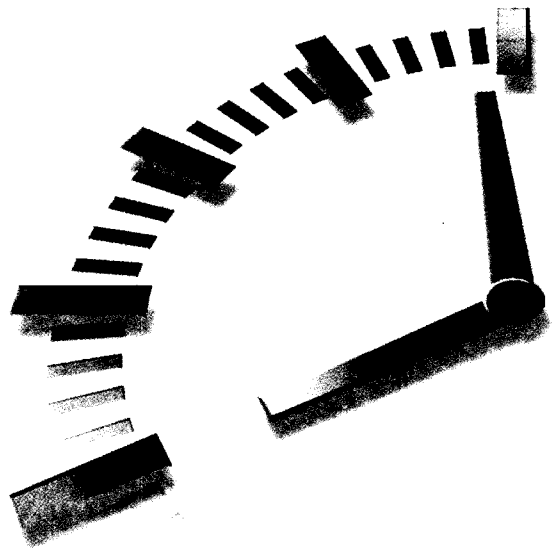
2. 使用 EMPLOYEE_TBL 表格（見附錄 D），編寫交談式 SQL 指令，傳回郵遞區號為 46234 的所有職員名稱。

Name	Null	Type
EMP_ID	NOT NULL	VARCHAR2(9)
LAST_NAME	NOT NULL	VARCHAR2(8)
FIRST_NAME	NOT NULL	VARCHAR2(8)
MID_INIT	CHAR(1)	
ADDRESS	NOT NULL	VARCHAR2(15)

Name	Null	Type
CITY	NOT NULL	VARCHAR2(12)
STATE	NOT NULL	CHAR(2)
ZIP	NOT NULL	CHAR(5)
PHONE	CHAR(10)	
PAGER	CHAR(10)	

a.

```
SELECT LAST_NAME, FIRST_NAME
FROM EMPLOYEE_TBL
WHERE ZIP = '&ZIP';

Enter value for zip: 46234
old   3: WHERE ZIP = '&ZIP'
new   3: WHERE ZIP = '46234'

Results of Query


LAST_NAM FIRST_NA
-------- ---------
SPURGEON TIFFANY

1 row selected.
```



附錄 D

本書用 CREATE TABLE 指令建立的範例

這個附錄非常有用，列出本書用 CREATE TABLE 指令建立的範例，您可以用這些指令建立自己的查詢。

D-1 EMPLOYEE_TBL

```
create table EMPLOYEE_TBL
(
emp_id              varchar2(9)        not null,
last_name           varchar2(15)       not null,
first_name          varchar2(15)       not null,
middle_name         varchar2(15),
address             varchar2(30)       not null,
city                varchar2(15)       not null,
state               char(2)            not null,
zip                 number(5)          not null,
phone               char(10),
```

```
pager               char(10),
constraint emp_pk primary key (emp_id)
)
/
```

D-2 EMPLOYEE_PAY_TBL

```
create table EMPLOYEE_PAY_TBL
(
emp_id              varchar2(9)        not null        primary key,
position            varchar2(15)       not null,
date_hire           date,
pay_rate            number(4,2),
date_last_raise     date,
salary              number(8,2),
bonus               number(6,2),
constraint emp_fk foreign key  (emp_id) references employee_tbl
(emp_id)
)
/
```

D-3 CUSTOMER_TBL

```
create table CUSTOMER_TBL
(
cust_id             varchar2(10)    not null        primary key,
cust_name           varchar2(30)    not null,
cust_address        varchar2(20)    not null,
cust_city           varchar2(15)    not null,
cust_state          char(2)         not null,
cust_zip            number(5)       not null,
cust_phone          number(10),
cust_fax            number(10)
)
/
```

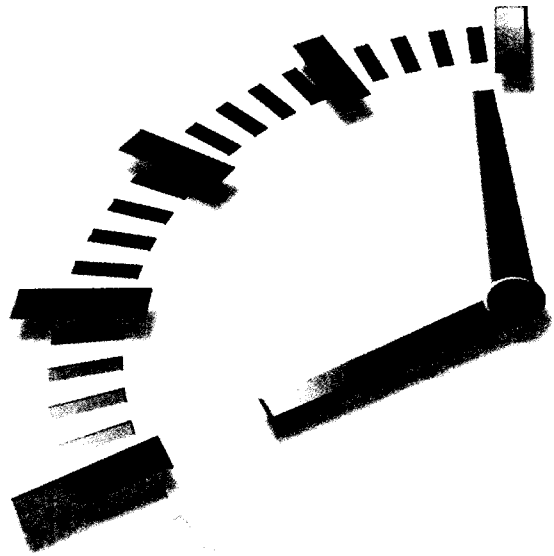
D-4 ORDERS_TBL

```
create table ORDERS_TBL
(
ord_num             varchar2(10)    not null        primary key,
cust_id             varchar2(10)    not null,
```

```
prod_id             varchar2(10)    not null,
qty                 number(6)       not null,
ord_date            date
)
/
```

D-5 PRODUCTS_TBL

```
create table PRODUCTS_TBL
(
prod_id         varchar2(10)    not null        primary key,
prod_desc       varchar2(40)    not null,
cost            number(6,2)     not null
)
```



附錄 E

本書用 INSERT 指令建立的範例資料

這附錄包含使用 INSERT 指令，插入資料到附錄 D 的表格中。在表格建立後，可以用在 INSERT 指令插入新的資料到表格裡。

E-1 INSERT Statements

E-1-1 EMPLOYEE_TBL

```
insert into employee_tbl values
('311549902', 'STEPHENS', 'TINA', 'DAWN,  RR 3 BOX 17A', 'GREENWOOD',
'IN', '47890', '3178784465',  NULL)
/

insert into employee_tbl values
('442346889', 'PLEW', 'LINDA', 'CAROL', '3301 BEACON',
'INDIANAPOLIS',
```

```
'IN', '46224', '3172978990', NULL)
/

insert into employee_tbl values
('213764555', 'GLASS', 'BRANDON', 'SCOTT', '1710 MAIN ST',
'WHITELAND', 'IN', '47885', '3178984321', '3175709980')
/

insert into employee_tbl values
('313782439', 'GLASS', 'JACOB', NULL, '3789 WHITE RIVER BLVD',
'INDIANAPOLIS', 'IN', '45734', '3175457676',  '8887345678')
/

insert into employee_tbl values
('220984332', 'WALLACE', 'MARIAH', NULL, '7889 KEYSTONE AVE',
'INDIANAPOLIS', 'IN', '46741', '3173325986', NULL)
/

insert into employee_tbl values
('443679012', 'SPURGEON', 'TIFFANY',  NULL, '5 GEORGE COURT',
'INDIANAPOLIS', 'IN', '46234', '3175679007', NULL)
/
```

E-1-2 EMPLOYEE_PAY_TBL

```
insert into employee_pay_tbl values
('311549902', 'MARKETING', '23-MAY-89', NULL, '01-MAY-97', '40000',
NULL)
/

insert into employee_pay_tbl values
('442346889', 'TEAM LEADER', '17-JUN-90', '14.75', '01-JUN-97', NULL,
NULL)
/

insert into employee_pay_tbl values
('213764555', 'SALES MANAGER', '14-AUG-94',  NULL, '01-AUG-97',
'30000', '2000')
/

insert into employee_pay_tbl values
('313782439', 'SALESMAN', '28-JUN-97',  NULL, NULL, '20000', '1000')
/
```

```
insert into employee_pay_tbl values
('220984332', 'SHIPPER', '22-JUL-96', '11.00', '01-JUL-97', NULL,
NULL)
/

insert into employee_pay_tbl values
('443679012', 'SHIPPER', '14-JAN-91', '15.00', '01-JAN-97', NULL,
NULL)
/
```

E-1-3 CUSTOMER_TBL

```
insert into customer_tbl values
('232', 'LESLIE GLEASON', '798 HARDAWAY DR', 'INDIANAPOLIS', 'IN',
'47856', '3175457690', NULL)
/

insert into customer_tbl values
('109', 'NANCY BUNKER', 'APT A 4556 WATERWAY', 'BROAD RIPPLE', 'IN',
'47950', '3174262323', NULL)
/

insert into customer_tbl values
('345', 'ANGELA DOBKO', 'RR3 BOX 76', 'LEBANON', 'IN', '49967',
'7658970090', NULL)
/
insert into customer_tbl values
('090', 'WENDY WOLF', '3345 GATEWAY DR', 'INDIANAPOLIS', 'IN',
'46224', '3172913421', NULL)
/
insert into customer_tbl values
('12', 'MARYS GIFT SHOP', '435 MAIN ST', 'DANVILLE', 'IL', '47978',
'3178567221', '3178523434')
/

insert into customer_tbl values
('432', 'SCOTTYS MARKET', 'RR2 BOX 173', 'BROWNSBURG', 'IN',
'45687', '3178529835', '3178529836')
/

insert into customer_tbl values
('333', 'JASONS AND DALLAS GOODIES', 'LAFAYETTE SQ MALL',
'INDIANAPOLIS', 'IN', '46222', '3172978886', '3172978887')
/
```

```
insert into customer_tbl values
('21', 'MORGANS CANDIES AND TREATS', '5657 W TENTH ST',
'INDIANAPOLIS', 'IN', '46234', '3172714398', NULL)
/

insert into customer_tbl values
('43', 'SCHYLERS NOVELTIES', '17 MAPLE ST', 'LEBANON', 'IN',
'48990', '3174346758', NULL)
/

insert into customer_tbl values
('287', 'GAVINS PLACE', '9880 ROCKVILLE RD', 'INDIANAPOLIS',
'IN', '46244', '3172719991', '3172719992')
/

insert into customer_tbl values
('288', 'HOLLYS GAMEARAMA', '567 US 31 SOUTH', 'WHITELAND',
'IN', '49980', '3178879023', NULL)
/

insert into customer_tbl values
('590', 'HEATHERS FEATHERS AND THINGS', '4090 N SHADELAND AVE',
'INDIANAPOLIS', 'IN', '43278', '3175456768', NULL)
/

insert into customer_tbl values
('610', 'RAGANS HOBBIES INC', '451 GREEN ST', 'PLAINFIELD', 'IN',
'46818', '3178393441', '3178399090')
/

insert into customer_tbl values
('560', 'ANDYS CANDIES', 'RR 1 BOX 34', 'NASHVILLE', 'IN',
'48756', '8123239871', NULL)
/

insert into customer_tbl values
('221', 'RYANS STUFF', '2337 S SHELBY ST', 'INDIANAPOLIS', 'IN',
'47834', '3175634402', NULL)
/
```

E-1-4 ORDERS_TBL

```
insert into orders_tbl values
('56A901', '232', '11235', '1', '22-OCT-97')
/

insert into orders_tbl values
('56A917', '12', '907', '100', '30-SEP-97')
/

insert into orders_tbl values
('32A132', '43', '222', '25', '10-OCT-97')
/

insert into orders_tbl values
('16C17', '090', '222', '2', '17-OCT-97')
/

insert into orders_tbl values
('18D778', '287', '90', '10', '17-OCT-97')
/

insert into orders_tbl values
('23E934', '432', '13', '20', '15-OCT-97')
/
```

E-1-5 PRODUCTS_TBL

```
insert into products_tbl values
('11235', 'WITCHES COSTUME', '29.99')
/

insert into products_tbl values
('222', 'PLASTIC PUMPKIN 18 INCH', '7.75')
/

insert into products_tbl values
('13', 'FALSE PARAFFIN TEETH', '1.10')
/

insert into products_tbl values
('90', 'LIGHTED LANTERNS', '14.50')
/
```

```
insert into products_tbl values
('15', 'ASSORTED COSTUMES', '10.00')
/
insert into products_tbl values
('9', 'CANDY CORN', '1.35')
/

insert into products_tbl  values
('6', 'PUMPKIN CANDY', '1.45')
/

insert into products_tbl values
('87', 'PLASTIC SPIDERS', '1.05')
/

insert into products_tbl values
('119', 'ASSORTED MASKS', '4.95')
```

廣告回信
台灣北區郵政管理局登記證
北台字第 1037 號

＜免貼郵票＞

台北市信義路五段 18 號 B1

第三波資訊股份有限公司
資 訊 圖 書 部　　收

姓名：

地址：

第三波電腦叢書　讀者回函

親愛的讀者:

首先感謝您對本公司電腦叢書的支持與愛護，為了建立本公司的服務體系及希望對於本書的市場反應能夠正確掌握，本公司非常殷切地期盼您能夠利用一點點時間，填妥以下資料，再把它裁(或"撕")下來並且封訂起來，投入郵筒。我們收到後，將會審慎考量您寶貴的意見並改進。對了，千萬記得*不用貼郵票喔*。

或者，您也可以傳真正反面～～　（02）8780－5656 ※註明：致 資訊圖書部

再者，您還可以到～～ **http://www.acertwp.com.tw**

點選客戶回函註冊卡，完成回函登記。

[redacted]

編　號：6104.06601.000

書　名：UNIX—24 小時自學手冊

購書地點(店名)：_____購買日期：19_____年_____月_____日

個 人 基 本 資 料

姓名：_____ 性別：□1.男□0.女 生日：19__年__月__日

行業： □1.公家機構 □2.教育單位 □3.資訊硬體 □4.資訊軟體 □5.電子業
□6.一般製造業 □7.金融業 □8.工商貿易 □9.學 生
□10.其他_____

工作性質： □1.MIS □2.採購 □3.銷售 □4.市場/企劃 □5.行政/財務
□6.設計/研發 □7.維修/品管/測試 □8.生產/製造
□9.教育訓練/管理 □10.其他_____

服務單位（公司/學校）：_____

部門/科系：_____

地址：_____市（縣）_____

電話：(O)(　)_____分機 _____ (H)(　)_____分機 _____

傳真：(　)_____ Email：_____

本書品質：5-非常好、4-很好、3-普通、2-不好、1-很差

・封面設計：□5 □4 □3 □2 □1 ・印刷：□5 □4 □3 □2 □1

・作者功力：□5 □4 □3 □2 □1 ・編排：□5 □4 □3 □2 □1

本書的內容，對您而言……

□太難了 □有點難 □恰到好處 □有點簡單 □太簡單

如果你要買資訊圖書，你最常到……

□一般書店 □重慶南路書店 □電腦門市 □大型量販店 □電腦展

□郵購 □Internet 線上訂購 □其他_____

你選擇購買本書的原因為……（可複選）

□品質優良 □朋友建議 □價格合理 □店頭推薦 □雜誌推薦 □作者 □出版社

□廣告吸引 □促銷優惠 □附加價值(光碟或磁片) □其他_____

您希望第三波將來為您出版哪一類型圖書？（可複選）

□電子／電腦硬體 □程式設計 □通訊／網路 □文書處理／試算表

□繪圖／影像處理 □作業系統 □兒童電腦 □行銷管理 □資料庫

□其他_____

--您是否願意收到第三波的活動或產品訊息？

□是 □否 （若您未勾選，我們將視爲您同意我們寄其他資料給您！）

--您是否願意收到第三波公司以外的活動或產品訊息？

□是 □否 （若您未勾選，我們將視爲您同意我們寄其他資料給您！）

其他建議：_____

_____

_____

_____

國家圖書館出版品預行編目資料

SQL — 24小時自學手冊 / Ryan K. Stephens, Ronald R. Plew 編著；佐登 編譯 .--初版.
--臺北市：第三波，1999[民88]
面； 公分
譯自：Teach Yourself SQL in 24 Hours
ISBN 957-23-0728-2 （平裝）

1.SQL（電腦程式語言）
312.932S77 88003310

SQL — 24小時自學手冊

譯者/佐登
發行人/王振容
企劃編輯/許雅玲、匡湘萍、李玉敏、牛楨蕙、陳慧玉、吳東燕

發行所/書局總經銷/第三波資訊股份有限公司
地址/臺北市信義路5段18號B1
電話/(02)8780-3636
客戶服務免付費專線
電話/080-083-636
服務時間/週一至週五9:00~21:00
印刷所/祥峰印刷事業有限公司

郵政劃撥帳號/19480950
帳戶/第三波資訊股份有限公司
新竹辦事處/新竹市四維路130號14F之6
電話/(03)5254436
台中分公司/台中市文心路一段218號22F
電話/(04)4711600
高雄分公司/高雄市博愛二路106號1樓
電話/(07)5588067
香港分公司/香港九龍土瓜灣道82-84號星華中心6字樓1號室
電話/(852)27643830

1999年03月 初版一刷
2001年03月 初版九刷
出版登記證/局版台業字第3010號

第三波

Teach Yourself

SQL 24小時自學手冊

開始囉！

保證24小時學會的課程

只要在短短的二十四小時內，您就可以上手並使用SQL。本書採用Step-by-Step的方法，以延續性的課程規劃，讓您溫故知新、舉一反三，快速掌握SQL的精髓。

提示 點出問題的捷徑和解決方法

註解 釐清重要的觀念和程序

注意 幫助您避免常見的錯誤

本書重點.....

- 學習管理資料庫物件
- 資料查詢的主要函數與子句
- 瞭解如何使用SQL管理使用者和安全性
- 建立表格和進階查詢
- 學習協調SQL執行效率
- 使用2D的圖形
- 從業界權威得到深入的瞭解、提示、技術與建議
- 瞭解資料的排序、群組與摘要，及更改資料展現結構
- 發掘SQL如何有效、互動的管理關聯式資料庫系統

類別：資料庫／程式設計

Ryan Stephens & Ronald R. Plew
是Oracle的資料庫管理員，也是Indianapolis州的Indiana University-Purdue University的 教授，負責教導SQL、PL/SQL與Oracle資料庫管理，專精於Oracle 與SQL，已有七年以上的管理開發經驗。

ISBN:957-23-0728-2 NT$480

9 789572 307281 00480

第三波資訊股份有限公司