

# 資料庫 SQL Server

編者 : Steve Dang

Q&A 社團 iCoding :

<https://www.facebook.com/groups/216955676502460>

Email : jungan0914@gmail.com



# 資料庫系統簡介

## 資料庫系統(Database System)

電腦化的資料儲存系統，使用者則透過各種應用程式來存取其中的資料

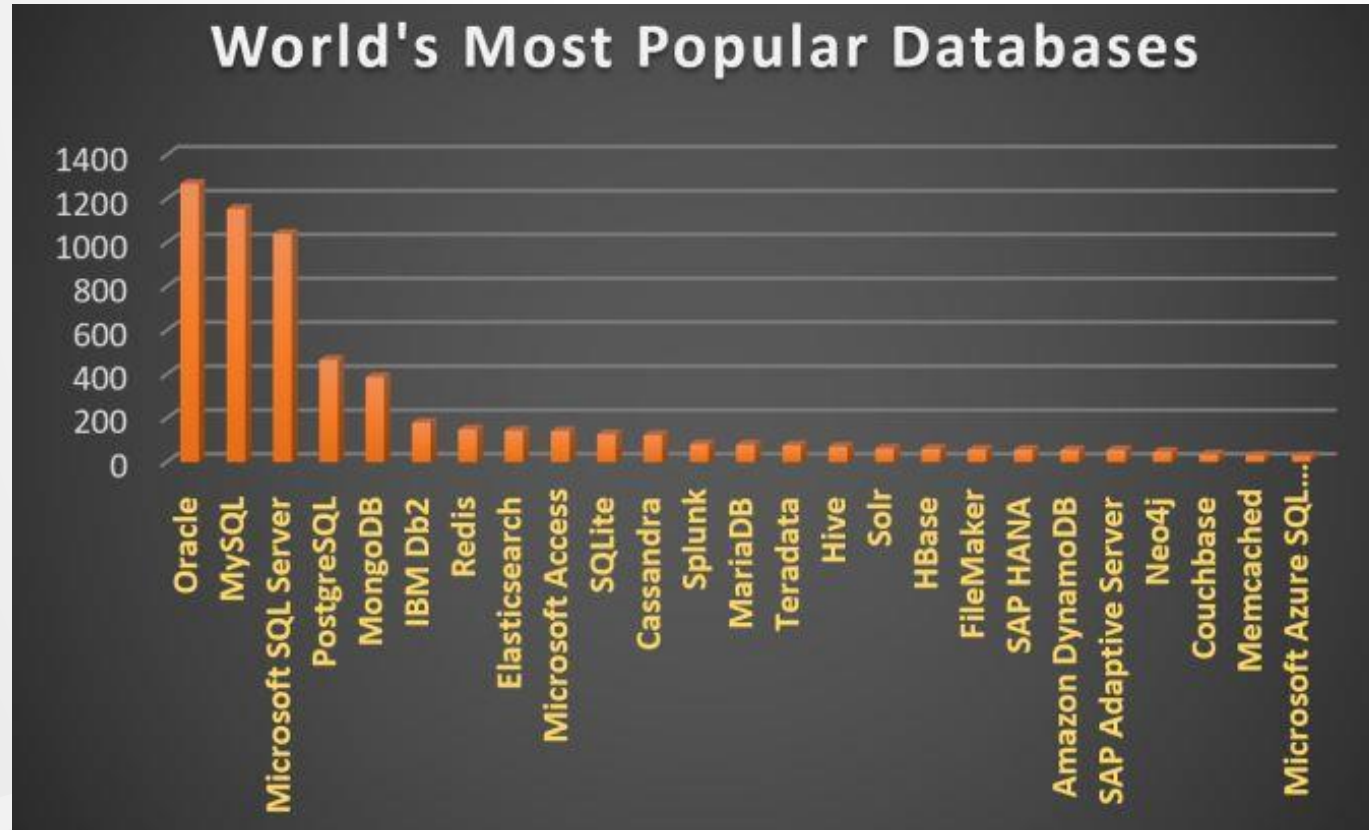
※資料庫系統分為兩部分

- 資料庫(Database) - 儲存資料的地方
- 資料庫管理系統(DataBase Management System, DBMS) - 管理資料庫的軟體

# SQL 結構化查詢語言

結構化查詢語言（英語：Structural Query Language，縮寫：SQL），是一種特殊目的之程式語言，用於資料庫中的標準資料查詢語言，IBM公司最早使用在其開發的資料庫系統中。1986年10月，美國國家標準學會對SQL進行規範後，以此作為關係式資料庫管理系統的標準語言（ANSI X3.135-1986），1987年得到國際標準組織的支援下成為國際標準。不過各種通行的資料庫系統在其實踐過程中都對SQL規範作了某些編改和擴充。所以，實際上不同資料庫系統之間的SQL不能完全相互通用。

# Most Popular Databases In The World 2019



The DB Engines score is calculated based on the following factors,

- Number of results in Google, Bing, and Yandex search engines
- Frequency of searches in Google Trends
- Frequency of technical discussions on the well-known IT-related Q&A sites Stack Overflow and DBA Stack Exchange
- The number of job offers on Indeed and Simply Hired.
- Number of profiles in professional networks including LinkedIn and Upwork.
- Mentions in Twitter tweets.

# Microsoft SQL Server

**Microsoft SQL Server**(微軟結構化查詢語言伺服器)是由美國微軟公司所推出的關聯式資料庫解決方案，最新的版本是 SQL Server 2019，已在美國時間 2019年11月3日發布。

資料庫的內建語言原本是採用美國標準局和國際標準組織所定義的SQL語言，但是微軟公司對它進行了部分擴充而成為作業用SQL。

幾個初始版本適用於中小企業的資料庫管理，但是近年來它的應用範圍有所擴充，已經觸及到大型、跨國企業的資料庫管理。

[https://zh.wikipedia.org/zh-tw/Microsoft\\_SQL\\_Server](https://zh.wikipedia.org/zh-tw/Microsoft_SQL_Server)

# 資料庫的類型和架構

## 資料庫的類型

階層式資料庫(Hierarchical Database)

採用樹狀結構，將資料分門別類儲存在不同的階層之下

網狀式資料庫(Network Database)

將每筆記錄當成一個節點，節點與節點之間可以建立關聯，形成一個複雜的網狀架構

關聯式資料庫(Relational Database)

以二維的矩陣來儲存資料，而儲存在欄跟列裡的資料必會有所關聯

物件導向式資料庫(Object-Oriented Database)

以物件導向的方式來設計資料庫，其中包含物件的屬性.方法.類別.繼承等特性

## 資料庫系統的網路架構

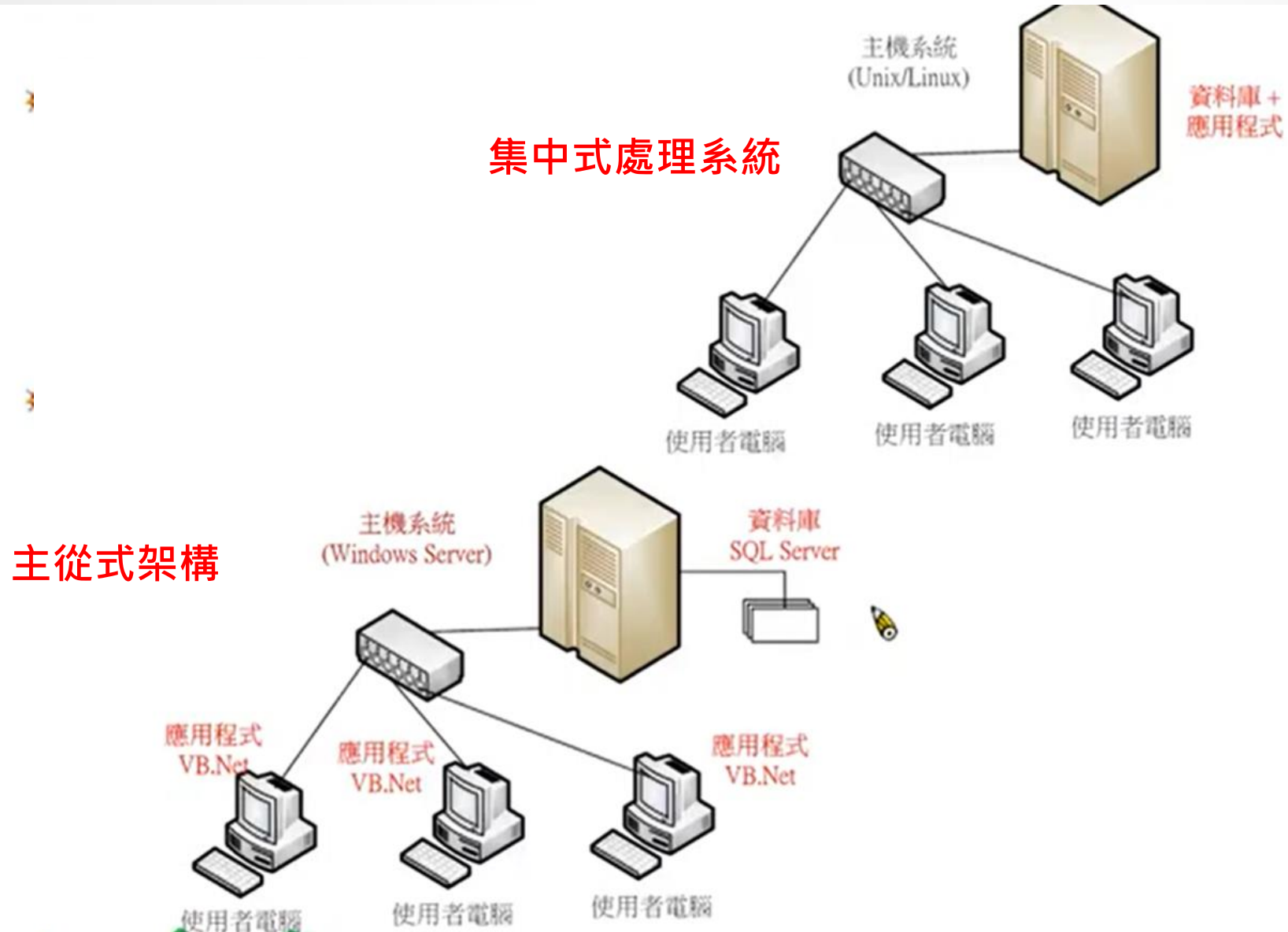
單機架構

大型主機/終端機架構

主從式架構

分散式架構

# 資料庫的系統架構



# 關聯式運算

關聯式資料模式上的資料操作部份有：

關聯式代數(Relational Algebra)與

關聯式計算(Relational Calculus)兩種

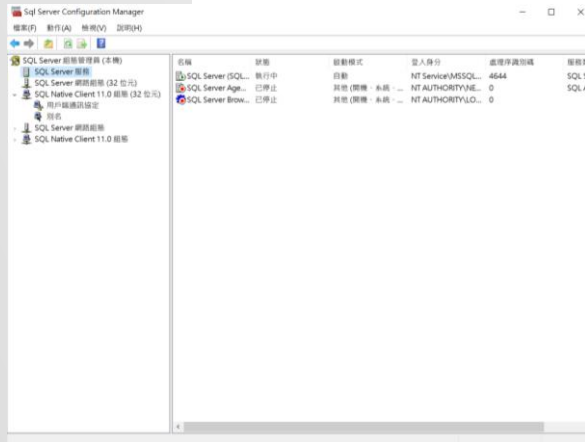
Arithmetic Operators	Relational Operators	Logical Operators	Special Operators
+	=	And	In ,Not In
-	!= or <>	Or	Between,Not Between
*	>	Not	Like ,Not Like
/	<		Is null , Is Not Null
	>=		
	<=		



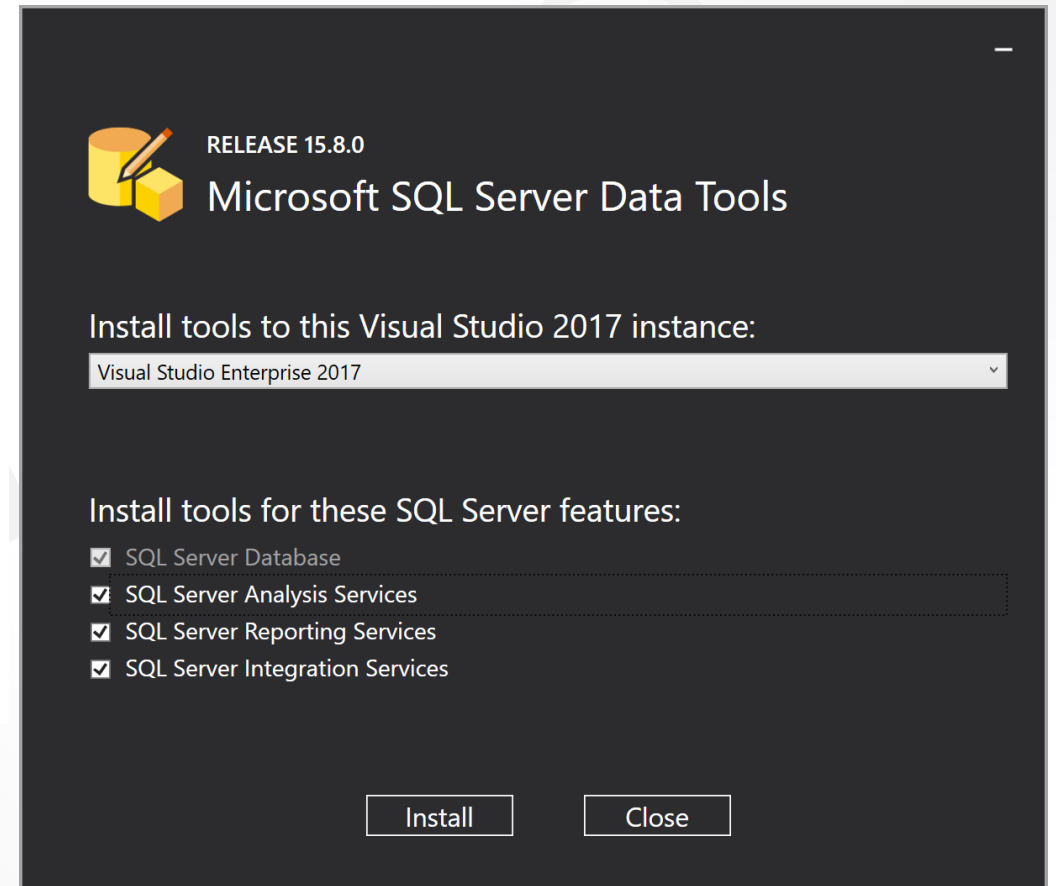
# 關聯式代數

- (1) 聯集運算(Union) :
- (2) 交集運算(Intersection) :
- (3) 差集運算(Difference) :
- (4) 乘積運算(Cartesian Product) :
- (5) 選擇運算(Select) :
- (6) 投影運算(Project) :
- (7) 關聯運算(Join) :
- (8) 除法運算(Divide) :

# SQL Server 工具



Server



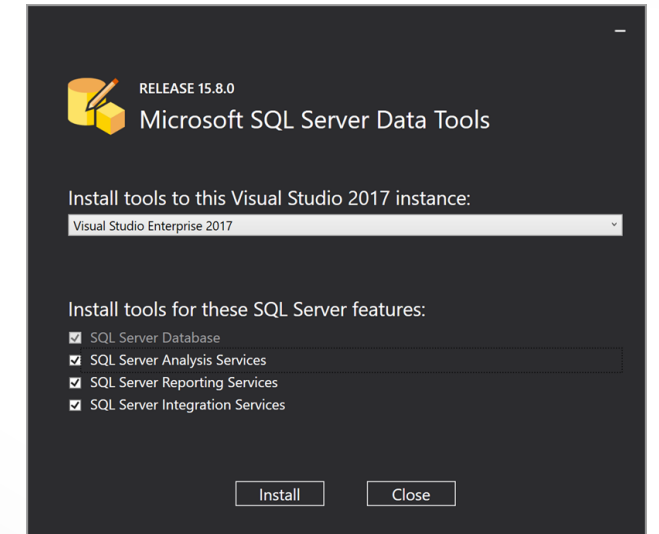
IDE

# SQL Server管理套件

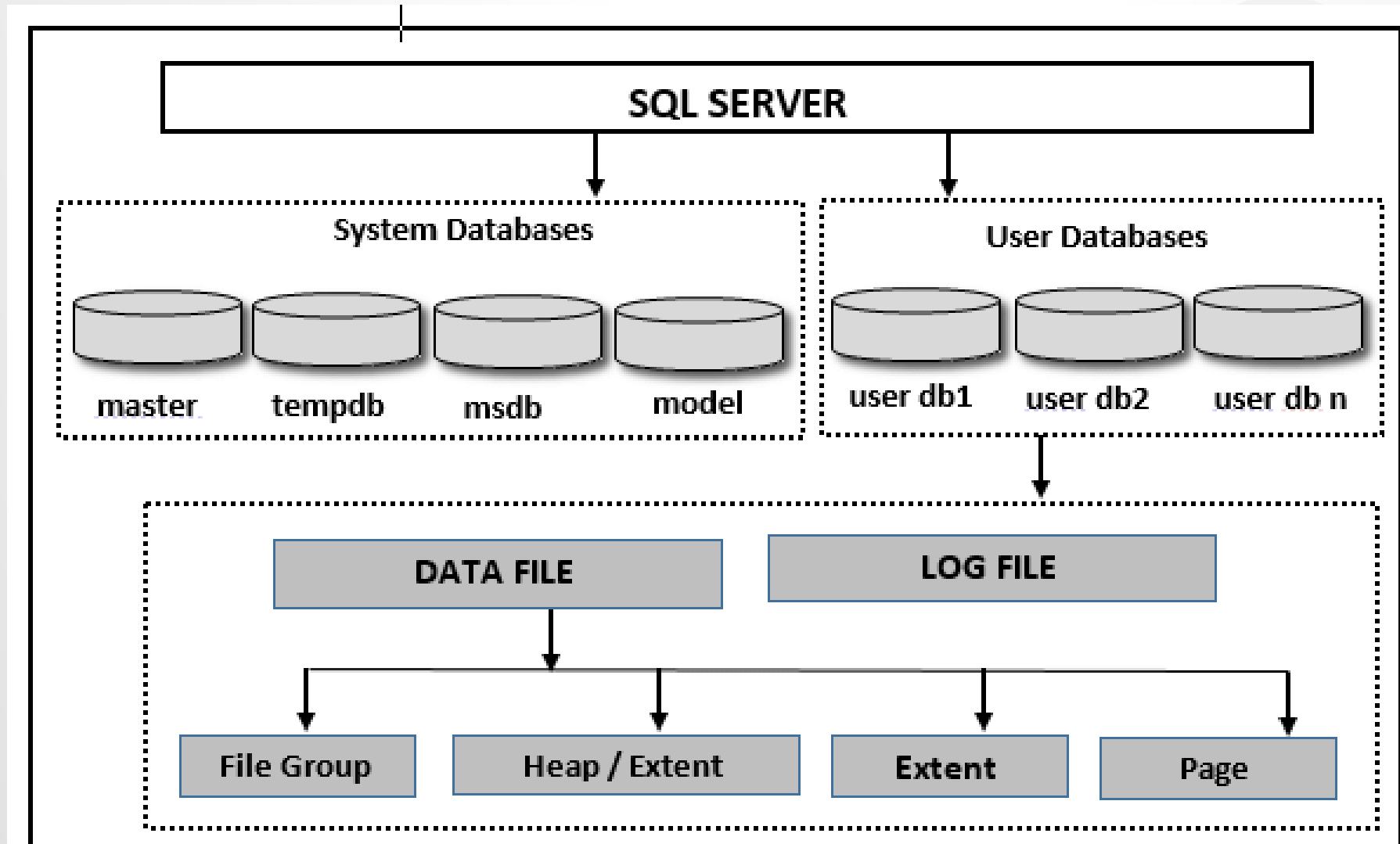
SQL Server管理套件(SSMS)是SQL Server的主管理控制台。

SSMS，可以創建數據庫對象(如數據庫，表，存儲過程，視圖等)，查看該數據在數據庫中，配置用戶帳戶，執行備份，複製，數據庫之間的數據傳輸，等等。

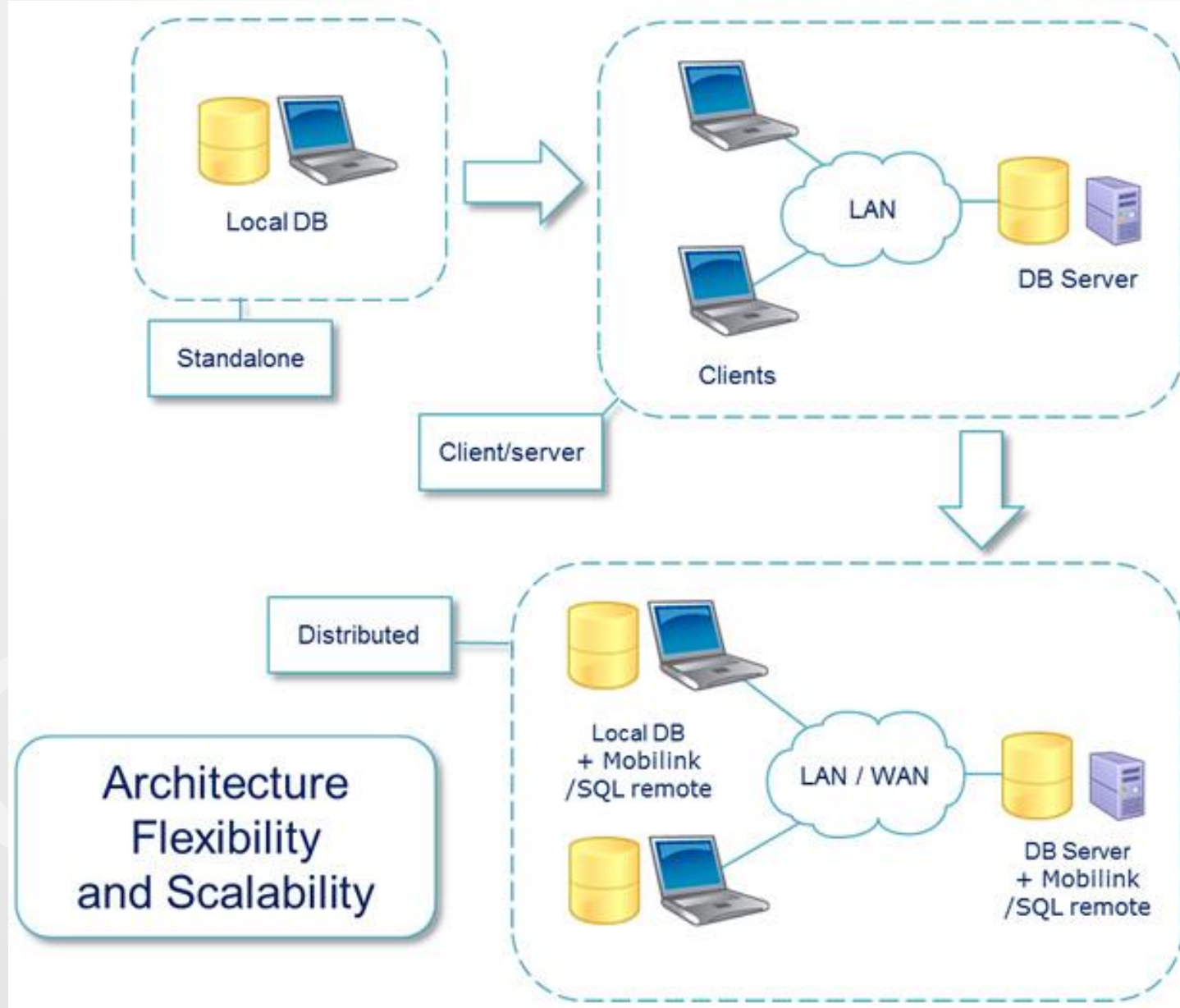
SQL Server管理套件的圖形用戶界面，最多任務是“點擊”。它也能夠運行SQL腳本接口，因此也有需要編程/腳本任務。然而，許多任務可以通過GUI或SQL腳本來執行，所以你自己的選擇使用哪一個。例如，您可以創建一個使用GUI或通過運行SQL腳本的數據庫。雖然如此，但你仍然需要GUI，以運行腳本。



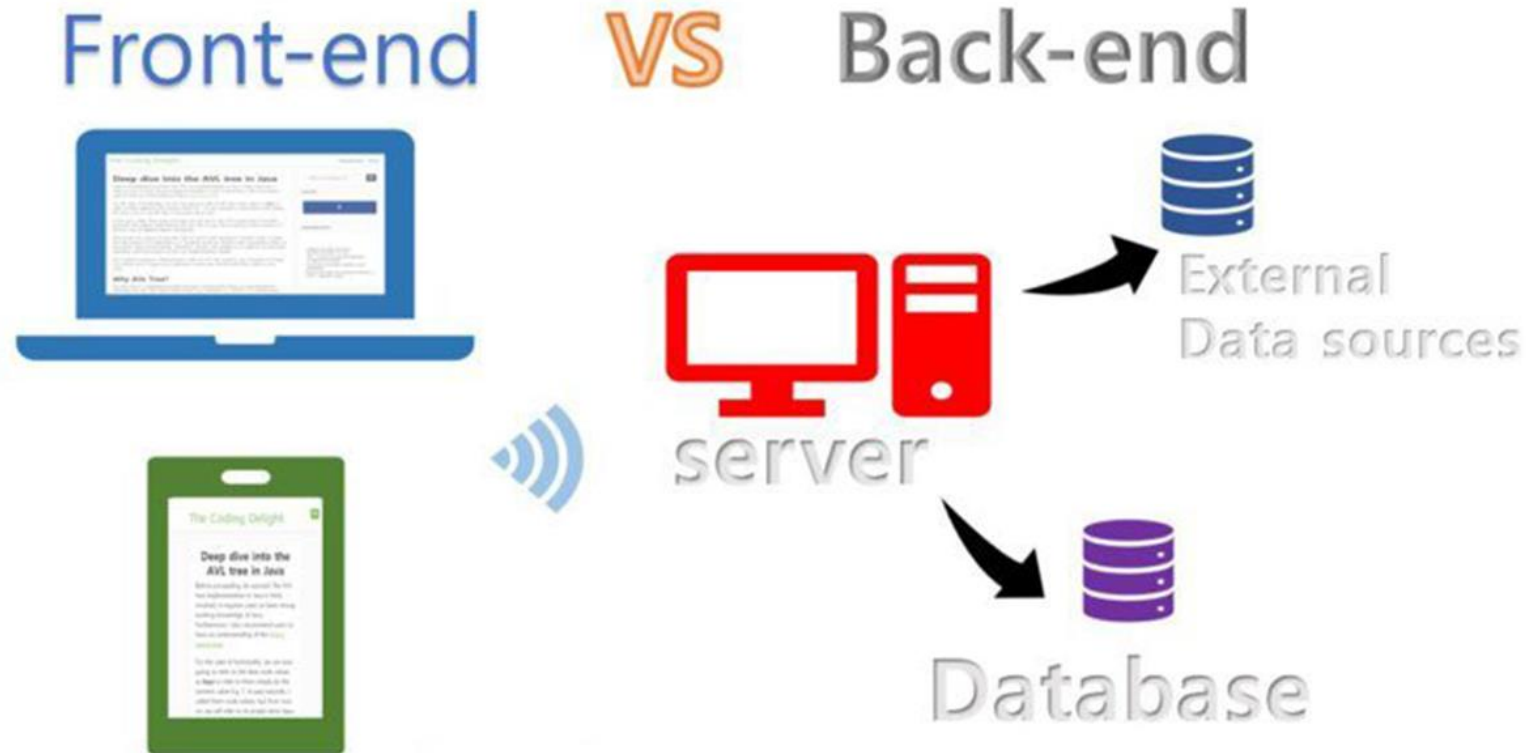
# SQL Server Database Architecture



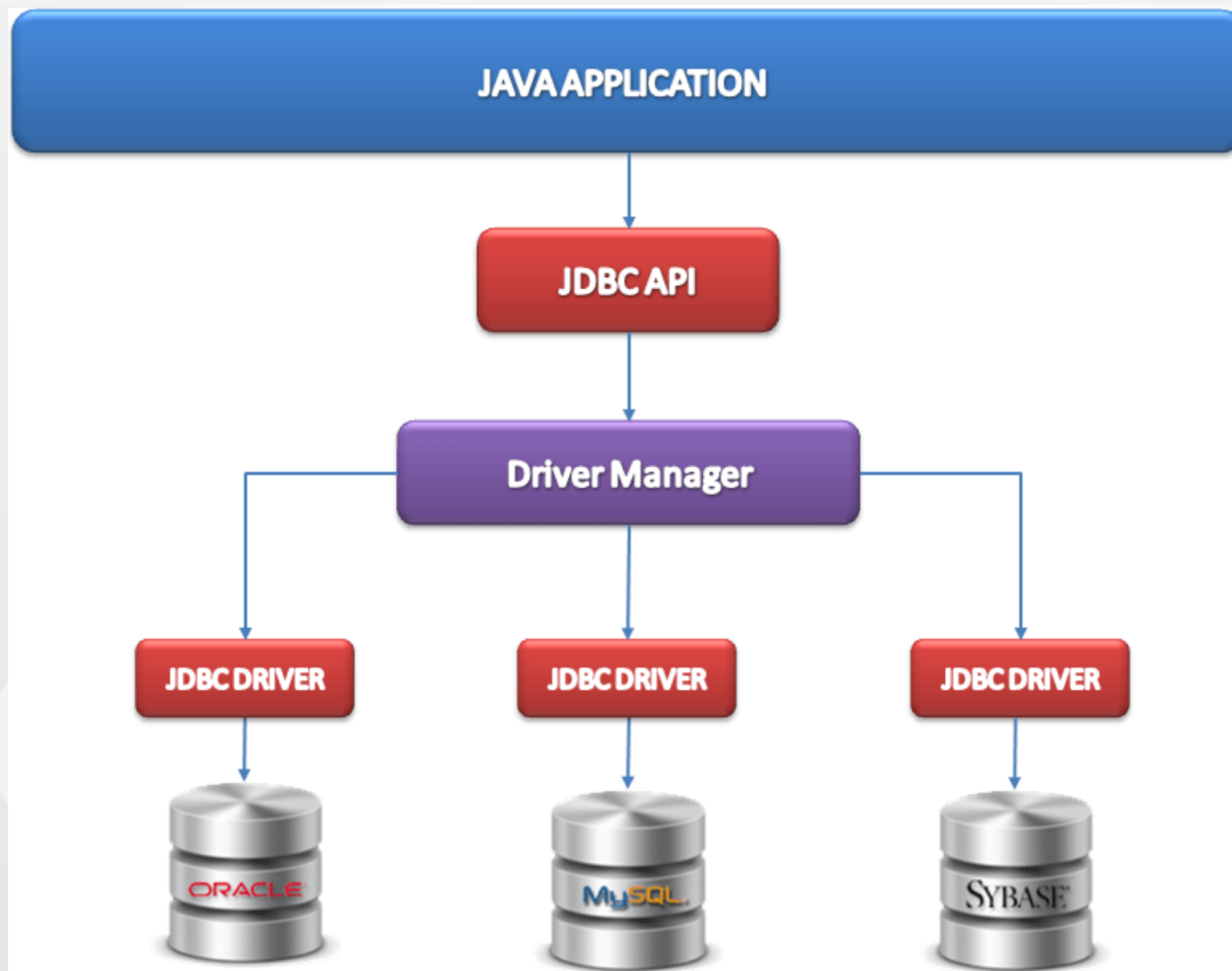
# 資料庫系統的基礎



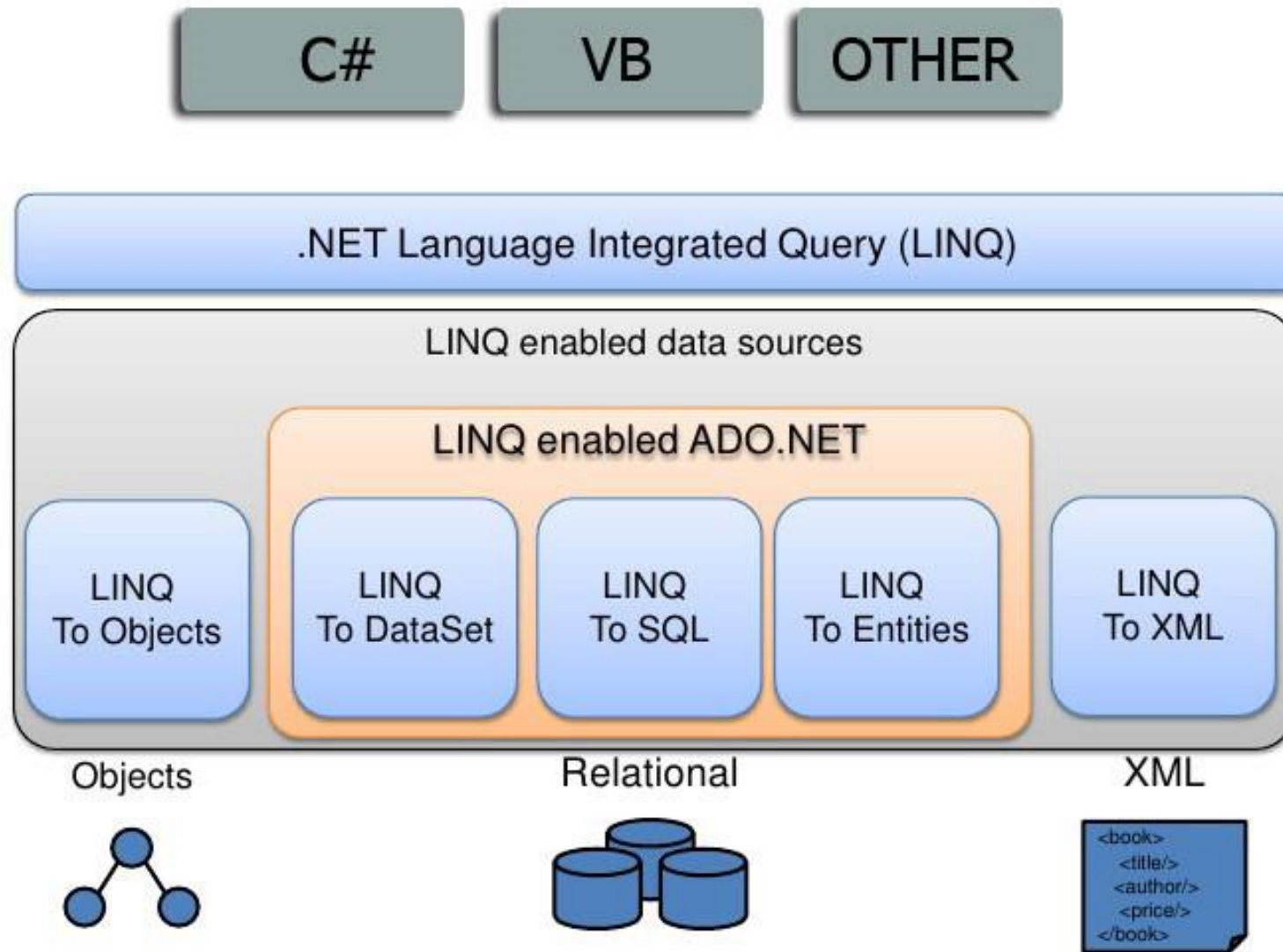
# 資料庫系統的基礎



# JAVA 基本架構的簡化觀點



# C# 基本架構的簡化觀點

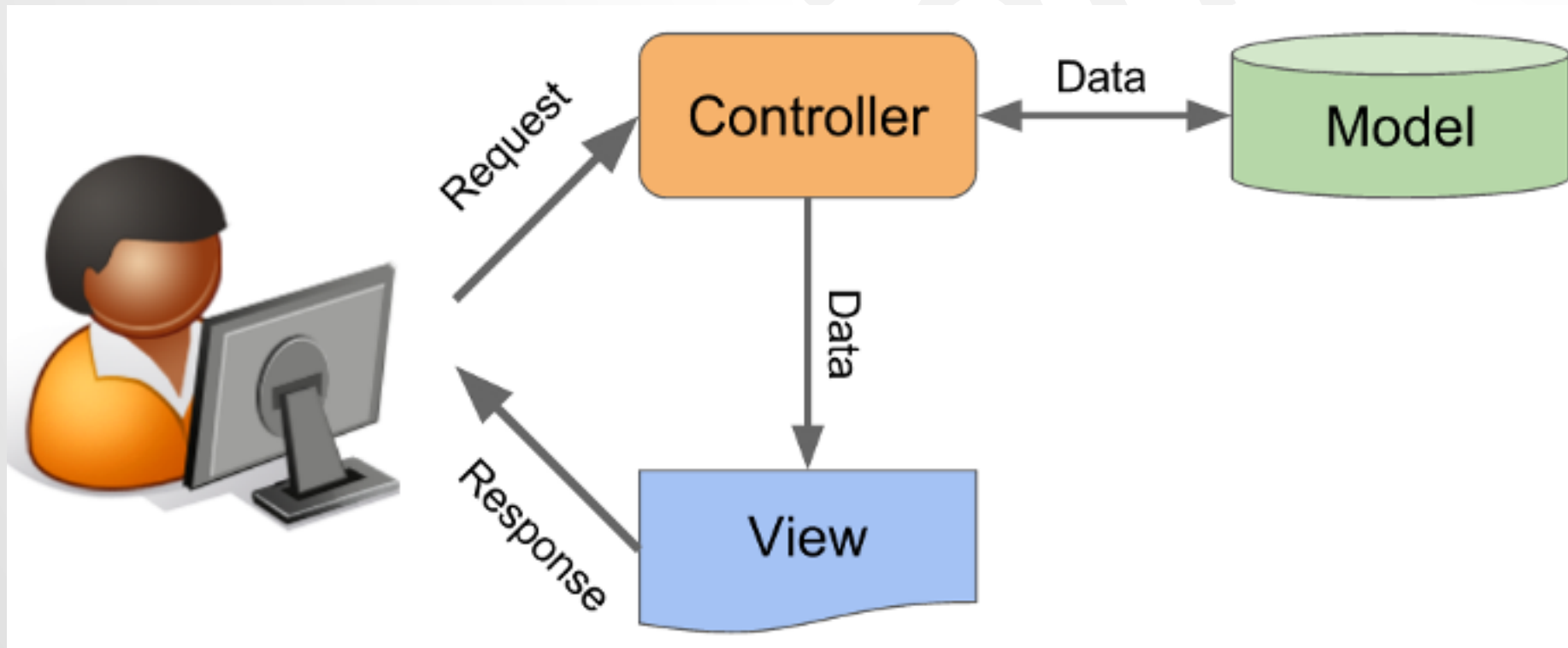




# MVC

MVC模式 ( Model-View-Controller ) 是軟體工程中的一種軟體架構模式，把軟體系統分為三個基本部分：模型 ( Model ) 、視圖 ( View ) 和控制器 ( Controller ) 。

- 模型 ( Model )
  - 程式設計師編寫程式應有的功能 ( 實現演算法等等 ) 、資料庫專家進行資料管理和資料庫設計(可以實現具體的功能)。
- 視圖 ( View )
  - 介面設計人員進行圖形介面設計。
- 控制器 ( Controller )
  - 負責轉發請求，對請求進行處理。



# 軟體的 5 層

## FIVE LAYERS OF SOFTWARES



**UI:** How Data is Presented.



**API:** How Data is Fetched.



**DATABASE:** How Data is Stored.



**LOGIC:** How Data is Processed.



**HOSTING:** Where Data is Stored.

# SQL Server 資料庫有三種檔案類型

## 資料庫檔案

SQL Server 資料庫有三種檔案類型，如下表所示。

檔案	描述
Primary	包含資料庫的啟動資訊，並指向資料庫中的其他檔案。每個資料庫都有一個主要資料檔案。建議您將主要資料檔的副檔名設為 .mdf。
次要	選擇性的使用者定義資料檔案。您可將每個檔案放在不同的磁碟機上，以將資料分散在多個磁碟上。建議您將次要資料檔的副檔名設為 .ndf。
交易記錄	記錄包含用來復原資料庫的資訊。每個資料庫至少要有一個記錄檔。建議的交易記錄檔的副檔名為 .ldf。

例如，名為 **Sales** 的簡單資料庫具有一個主要檔案，其包含所有資料和物件，且具有一個包含交易記錄資訊的記錄檔。也可以建立名為 **Orders** 的複雜資料庫，其包含一個主要檔案和五個次要檔案。在資料庫中的資料和物件平均分散於總共六個檔案中，而且有四個記錄檔包含交易記錄資訊。

<https://docs.microsoft.com/zh-tw/sql/relational-databases/databases/database-files-and-filegroups?view=sql-server-ver15>

# SQL語言四個類別的指令

## DDL (Data Definition Language) 資料定義語言

用作開新資料表、設定欄位、刪除資料表、刪除欄位，管理所有有關資料庫結構的東西，常見的指令有

- **Create**：建立資料庫的物件。
- **Alter**：變更資料庫的物件。
- **Drop**：刪除資料庫的物件。

## DML (Data Manipulation Language) 資料操作語言

用作新增一筆資料，刪除、更新等工作，常見的指令有

- **Insert**：新增資料到 Table 中。
- **Update**：更改 Table 中的資料。
- **Delete**：刪除 Table 中的資料。

## DQL (Data Query Language) 資料查詢語言

只能取回查詢結果，指令只有1個

- **Select**：選取資料庫中的資料。

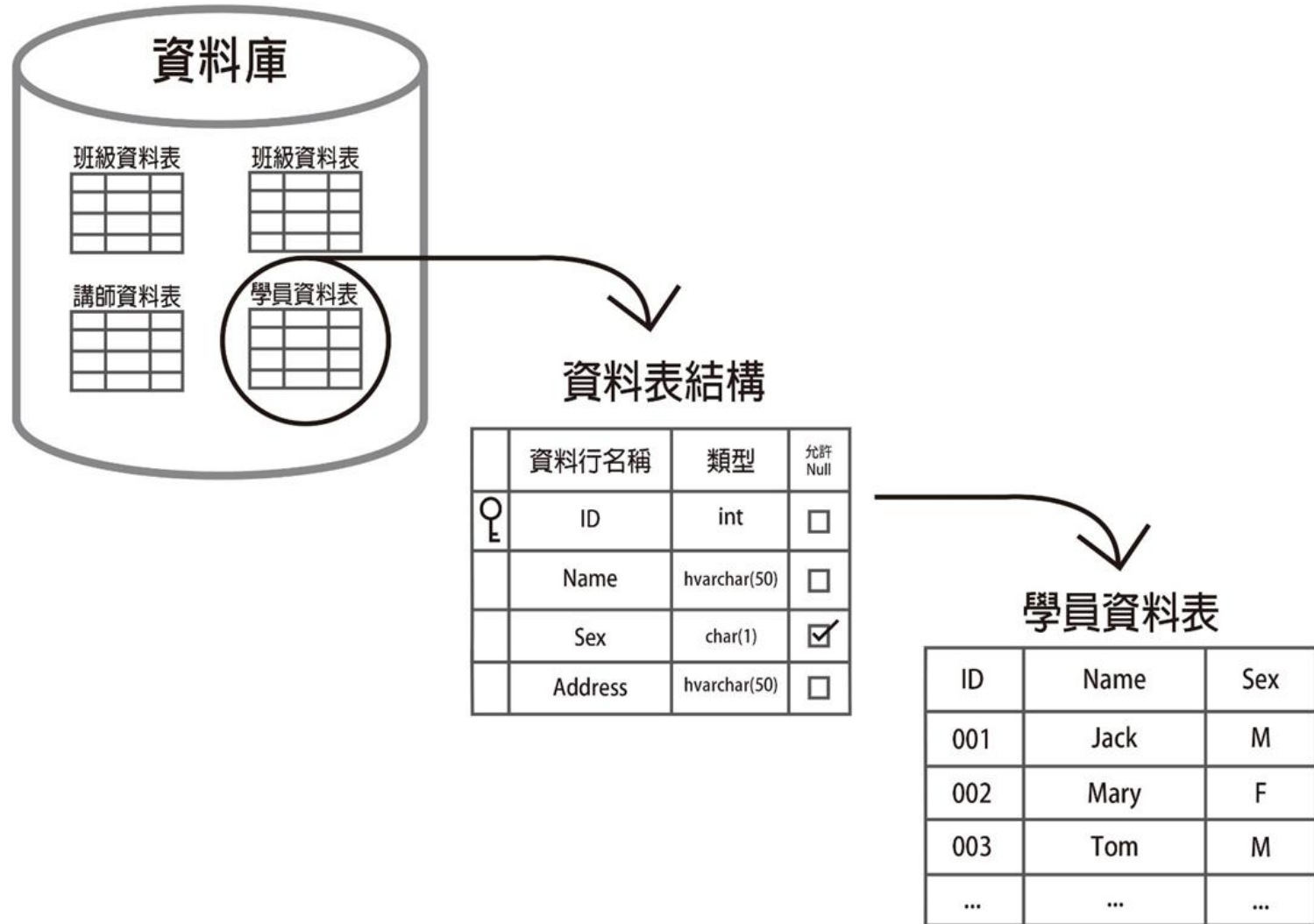
## D/TCL (Data Control Language) 資料控制語言

用作處理資料庫權限及安全設定，常見的指令有

- **Grant**：賦予使用者使用物件的權限。
- **Revoke**：取消使用者使用物件的權限。
- **Commit**：Transaction 正常作業完成。
- **Rollback**：Transaction 作業異常，異動的資料回復到 Transaction 開始的狀態

Types of SQL Commands			
DDL	DML	DCL	TCL
CREATE ALTER DROP TRUNCATE RENAME	SELECT INSERT UPDATE DELETE MERGE	GRANT REVOKE	COMMIT ROLLBACK SAVEPOINT

# SQL 資料庫結構



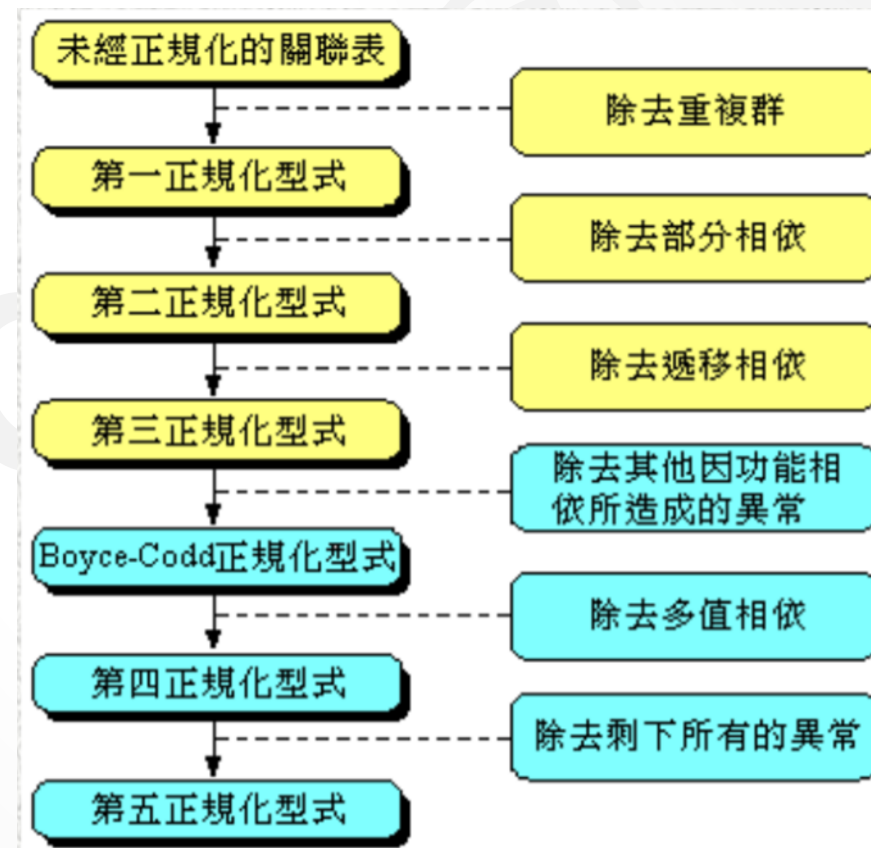
# SQL Server 資料庫有三種檔案類型

## 正規化的概念 何謂正規化(Normalization)？

就是結構化分析與設計中，建構「資料模式」所運用的一個技術，其目的是為了降低資料的「重覆性」與避免「更新異常」的情況發生。因此，就必須將整個資料表中重覆性的資料剔除，否則在關聯表中會造成新增異常、刪除異常、修改異常的狀況發生。一般而言，正規化的精神就是讓資料庫中重覆的欄位資料減到最少，並且能快速的找到資料，以提高關聯性資料庫的效能。

### 【目的】

1. 降低資料重覆性(Data Redundancy) (Data Redundancy)。
2. 避免資料更新異常(Anomalies)



[https://hackmd.io/@TSMI\\_E7ORNeP8YBbWm-IFA/rykcj8kmM?type=view](https://hackmd.io/@TSMI_E7ORNeP8YBbWm-IFA/rykcj8kmM?type=view)



# 關聯式資料庫模型

可避免：  
 Insertion Anomalies 輸入異常  
 Deletion Anomalies 刪除異常  
 Update Anomalies 更新異常

takeClass

StudentID	score1	score2	homeWork	courseID
19001	100	75	88	M001
19001	90	88	100	P001
19006	85	75	70	J001
19005	95	96	68	M001

Foreign Key 19007

Foreign Key

student

...	address	email	birth	name	StudentID
...	...	<a href="mailto:aaaa@gmail.com">aaaa@gmail.com</a>	2000/1/1	王小明	19001
...	...	<a href="mailto:bbbb@hotmail.com">bbbb@hotmail.com</a>	1999/2/2	李大明	19005
...	...	<a href="mailto:ccc@gmail.com">ccc@gmail.com</a>	2001/3/3	林中明	19006

Primary Key

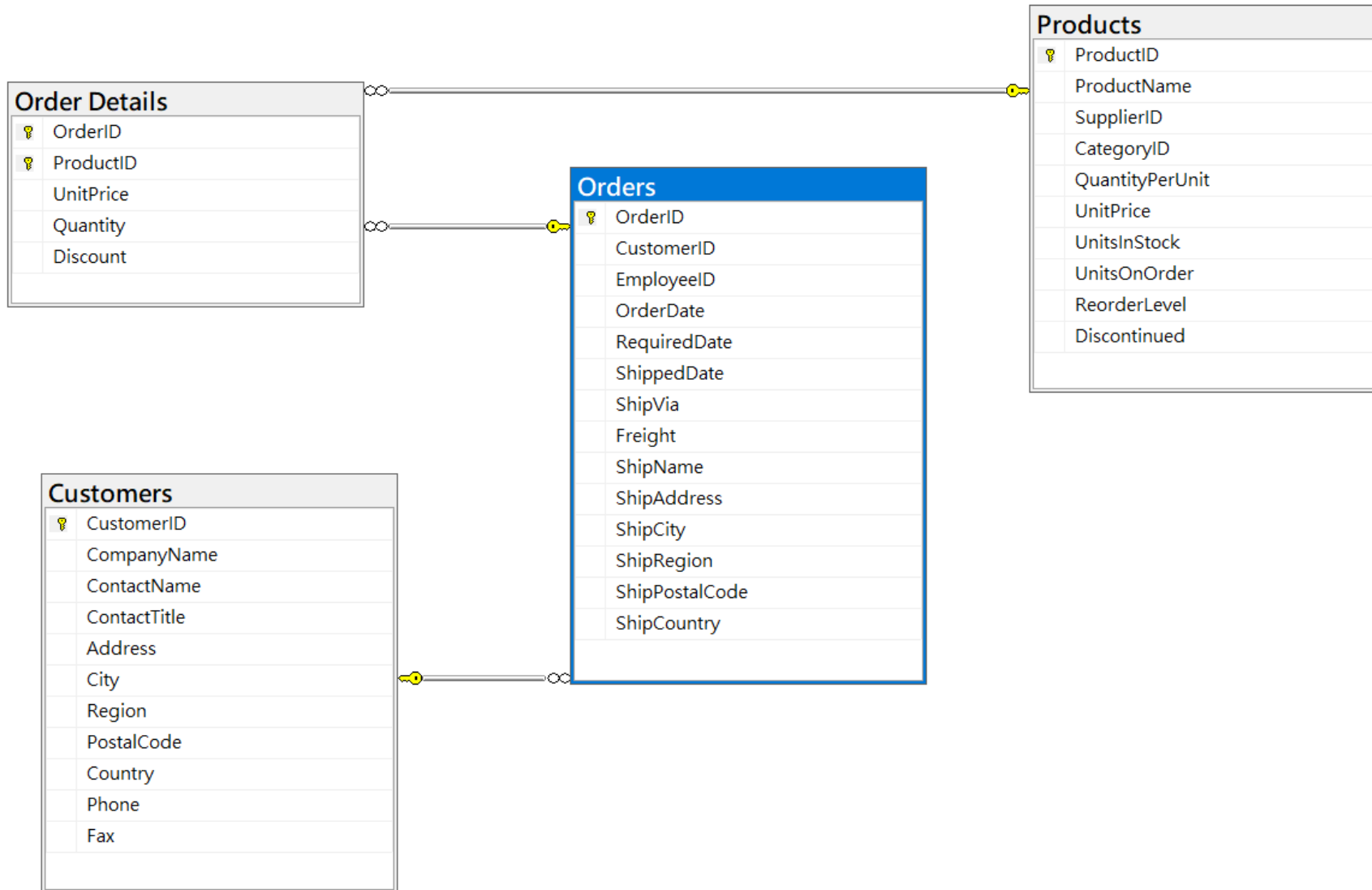
course

courseID	name	teacher	classRoom	textBook
P001	Python	王老師	...	...
M001	MySQL	李老師	...	...
J001	Java	張老師	...	...

Primary Key

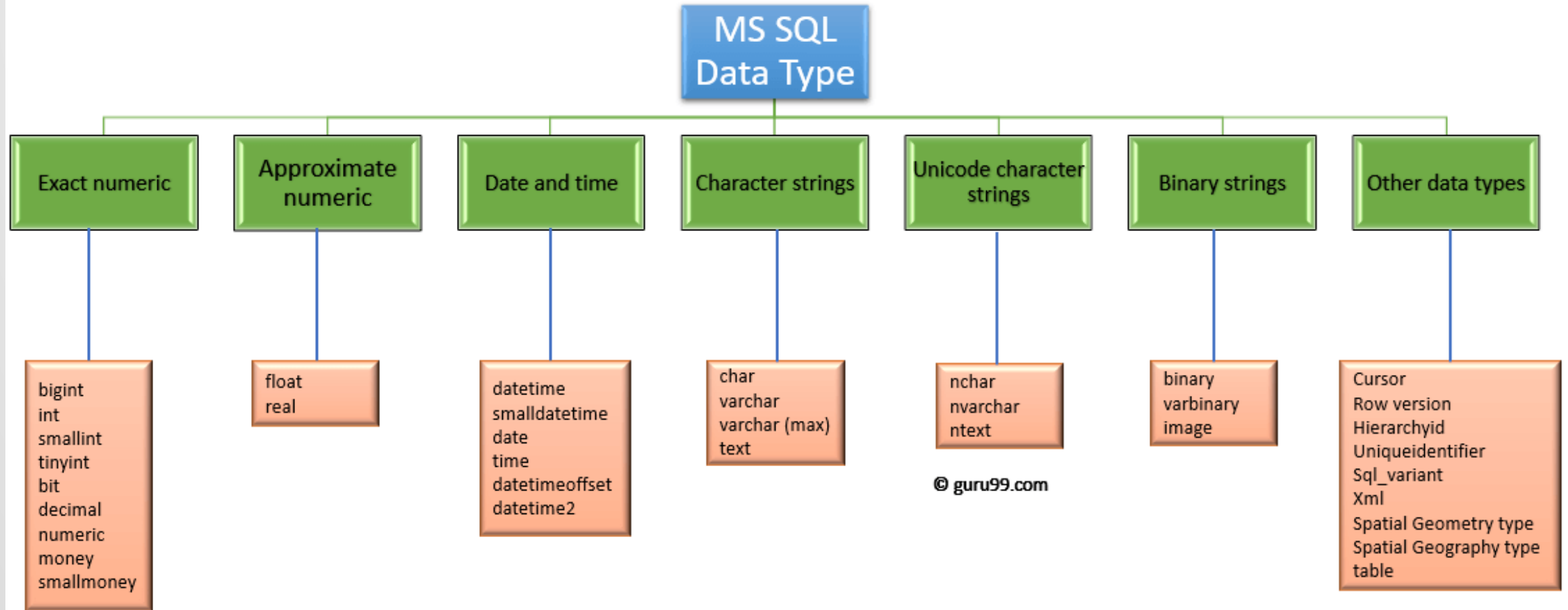
...	address	email	birth	name	StudentID	score1	score2	homeWork	courseID	name	teacher	classRoom	textBook
...	...	<a href="mailto:aaaa@gmail.com">aaaa@gmail.com</a>	2000/1/1	王小明	19001	100	75	88	M001	MySQL	李老師	...	...
					19001	90	88	100	P001	Python	王老師	...	...
					19001	90	88	100	P002	Python	王老師	...	...
					19001	90	88	100	P003	Python	王老師	...	...
					19001	90	88	100	P004	Python	王老師	...	...
					19001	90	88	100	P005	Python	王老師	...	...

# 關聯式資料庫模型





# SQL Server 資料型別



# SQL Server 資料型別

## 整數資料

資料型別	位元組	資料範圍
Tinyint	1 位元組 (Byte)	0 ~ 255
Smallint	2 位元組 (Byte)	-2 <sup>15</sup> (-32,768) 到 2 <sup>15</sup> -1 (32,767)
Int	4 位元組 (Byte)	-2 <sup>31</sup> (-2,147,483,648) 到 2 <sup>31</sup> -1 (2,147,483,647)
Bigint	8 位元組 (Byte)	-2 <sup>63</sup> (-9,223,372,036,854,775,808) 到 2 <sup>63</sup> -1 (9,223,372,036,854,775,807) 大於 2,147,483,647 的整數常數會轉換為 "decimal" 資料類型，而不是 "bigint" 資料類型。
Bit	1 位元組 (Byte)	0, 1, Null 字串值 True 和 False 可以轉換成 bit 值 True 轉換成 1 False 轉換成 0。

# SQL Server 資料型別

## 精確位數與浮點數

資料型別	位元組	資料範圍
float	8 位元組 (Byte)	-1.79E+308 到 -2.23E-308、0 2.23E-308 到 1.79E+308
real	4 位元組 (Byte)	-3.40E+38 到 -1.18E-38、0 1.18E-38 到 3.40E+38
numeric(有效位數, 小數)	視精確度而定	$-10^{38} + 1 \sim 10^{38} - 1$
decimal(有效位數, 小數)	視精確度而定	$-10^{38} + 1 \sim 10^{38} - 1$

numeric, decimal (具有固定有效位數和小數位數的數值資料類型)	有效位數	儲存體
	1 ~ 9	5 位元組 (Byte)
	10 ~ 19	9 位元組 (Byte)
	20 ~ 28	13 位元組 (Byte)
	29 ~ 38	17 位元組 (Byte)

# SQL Server 資料型別

## 字元、字串 - Unicode 字串

資料型別	位元組	資料範圍
char( <i>n</i> )	1字元 1 位元組 (Byte)	1 ~ 8000 字元
varchar( <i>n</i> )	1字元 1 位元組 (Byte)	1 ~ 8000 字元
varchar( <i>max</i> )	變動長度 max = 2GB	1 ~ 2 <sup>31</sup> -1 字元
text	變動長度 max = 2GB	1 ~ 2 <sup>31</sup> -1 字元

## Unicode 字串

nchar( <i>n</i> )	1字元 2 位元組 (Byte)	1 ~ 4000 字元 “固定長度”
nvarchar( <i>n</i> )	1字元 2Byte	1 ~ 4000 字元 “可變長度”
nvarchar( <i>max</i> )	1字元 2 位元組 (Byte) 變動長度 max = 2GB	1 ~ 2 <sup>31</sup> -1 字元 (可變長度大型文字資料)
ntext	1字元 2 位元組 (Byte) 變動長度 max = 2GB	1 ~ 2 <sup>30</sup> -1 字元

# SQL Server 資料型別

## Binary 二進位字串

資料型別	位元組	資料範圍
<code>binary(n)</code>	固定長度 8000 Bytes	1 ~ 8000 Bytes (儲存體大小是 n 位元組)
<code>varbinary(n)</code>	變動長度	1 ~ 8000 Bytes
<code>varbinary(max)</code>	變動長度 max = 2GB	1 ~ $2^{31}-1$ Bytes (儲存體大小是輸入資料的實際長度再加上 2 位元組)
<code>image</code>	變動長度 max = 2GB	0 到 $2^{31}-1$ (2,147,483,647) 位元組的可變長度二進位資料

# SQL Server 資料型別

## 日期 - 時間 - 貨幣 - Timestamp

資料型別	位元組	資料範圍
<b>datetime</b>	8 位元組(字元長度最小 19 個位置，最大 23 個位置)	1753 年 1 月 1 日到 9999 年 12 月 31 日 時間範圍 00:00:00 到 23:59:59.997 "2012-06-30 06:30:02.612"
<b>datetime2(n)</b>	6 個位元組代表有效位數小於 3，而 7 個位元組則代表有效位數是 3 和 4。所有其他有效位數則需要 8 個位元組	西元 1 年 1 月 1 日到西元 9999 年 12 月 31 日 時間範圍 00:00:00 到 23:59:59.9999999 "2012-06-30 06:30:02.612"
<b>smalldatetime</b>	4 位元組 (Byte)	1900:01:01 到 2079:06:06 1900 年 1 月 1 日到 2079 年 6 月 6 日 "2012-09-23 17:23"
<b>datetimeoffset(n)*</b>	10 Bytes(固定)預設值 最小 26 位數 (YYYY-MM-DD hh:mm:ss {+ -}hh:mm) 最大 34 位數 (YYYY-MM-DD hh:mm:ss. nnnnnnn {+ -}hh:mm)	0001-01-01 到 9999-12-31 西元 1 年 1 月 1 日到西元 9999 年 12 月 31 日

# SQL Server 資料型別

## Money 金融貨幣/精確數值

money	8 位元組 (Byte)	$-2^{63} \sim 2^{63} - 1$ “小數4位” -922,337,203,685,477.5808 到 922,337,203,685,477.5807
smallmoney	4 位元組 (Byte)	$-2^{31} \sim 2^{31} - 1$ “小數4位” -214,748.3648 到 214,748.3647

## Timestamp 時間戳記

timestamp	8 位元組 (Byte)	8 Bytes 的 16 進位值
uniqueidentifier	16 位元組 (Byte)	16 Bytes 的 16 進位值

# SQL 資料庫 Join

## Inner Join / Join

tb01								tb2			
			orderID	[....]							



Column Join  
one → one  
one → many

## Left Join

tb01								tb2			
			orderID	[....]	NULL						



Column Join  
one → one  
one → many

## Righth Join

tb01								tb2			
			orderID	[....]							
NULL											



Column Join  
one → one  
one → many

## Full Join

tb01								tb2			
			orderID	[....]	NULL						
NULL											



Column Join  
one → one  
one → many



# 真質表

*NOT*

$x$	$x'$
0	1
1	0

*AND*

$x$	$y$	$xy$
0	0	0
0	1	0
1	0	0
1	1	1

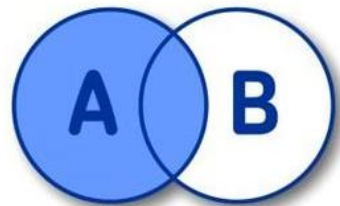
*OR*

$x$	$y$	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

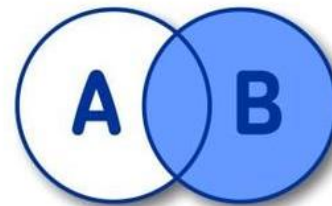
*XOR*

$x$	$y$	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

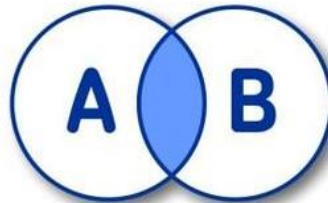
# SQL 資料表 Join



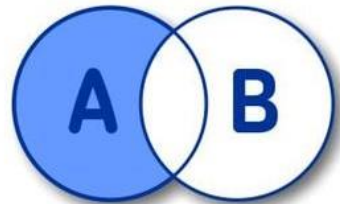
SELECT \* FROM  
A **LEFT** JOIN B  
ON A.KEY = B.KEY



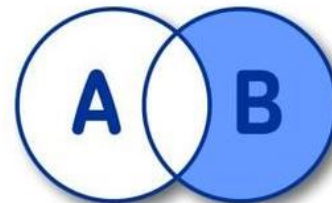
SELECT \* FROM  
A **RIGHT** JOIN B  
ON A.KEY = B.KEY



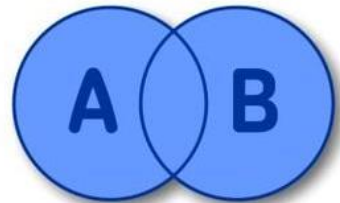
SELECT \* FROM  
A **INNER** JOIN B  
ON A.KEY = B.KEY



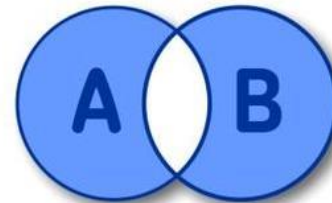
SELECT \* FROM A  
**LEFT** JOIN B  
ON A.KEY = B.KEY  
WHERE B.KEY IS NULL



SELECT \* FROM A  
**RIGHT** JOIN B  
ON A.KEY = B.KEY  
WHERE A.KEY IS NULL



SELECT \* FROM A  
**FULL OUTER** JOIN B  
ON A.KEY = B.KEY



SELECT \* FROM A **FULL  
OUTER** JOIN B ON A.KEY =  
B.KEY WHERE A.KEY IS  
NULL OR B.KEY IS NULL

# SQL Select 語法 where vs. having

## ■ 一般式如下：

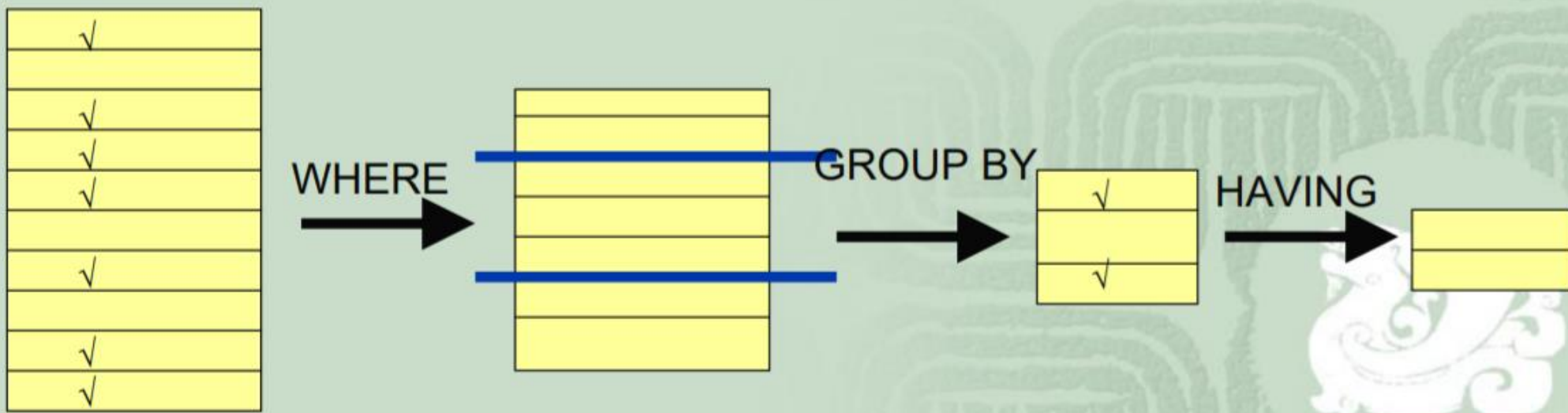
**SELECT** <分群屬性>,<彙總函數>

**FROM** <資料表>

**WHERE** <記錄選取條件>

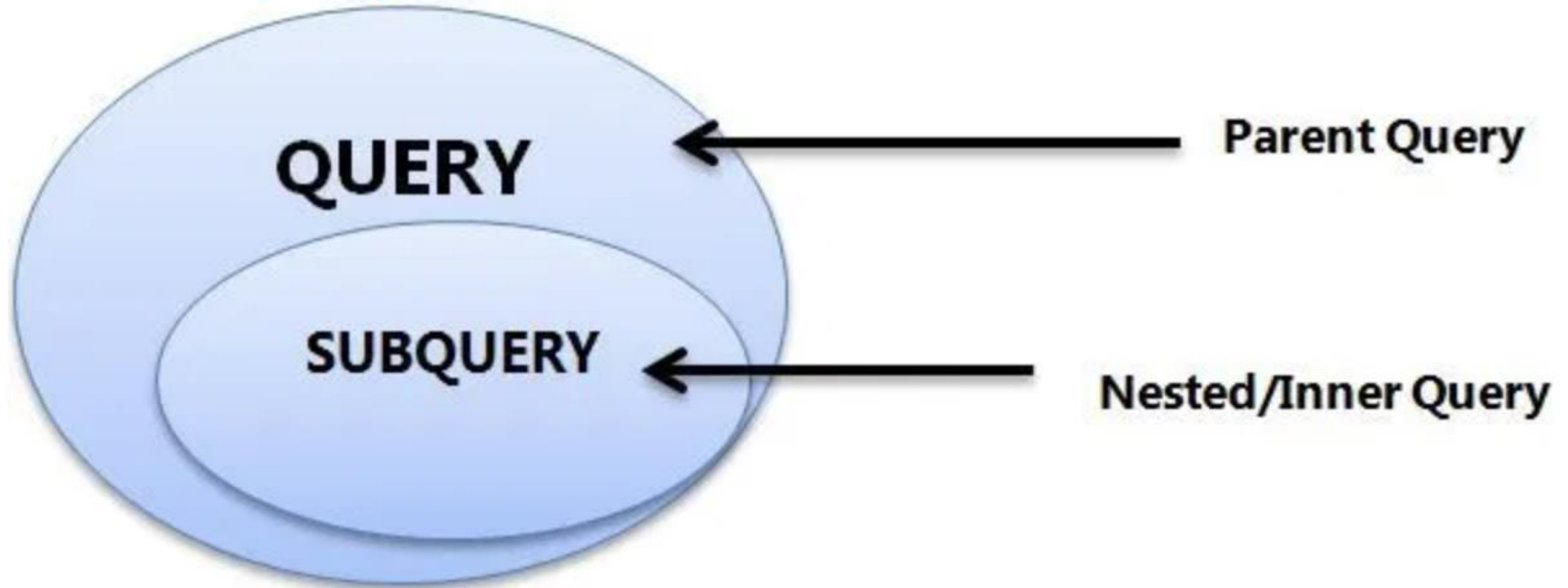
**GROUP BY** <分群屬性>

**HAVING** <記錄群選取條件>



# 子查詢 SubQuery

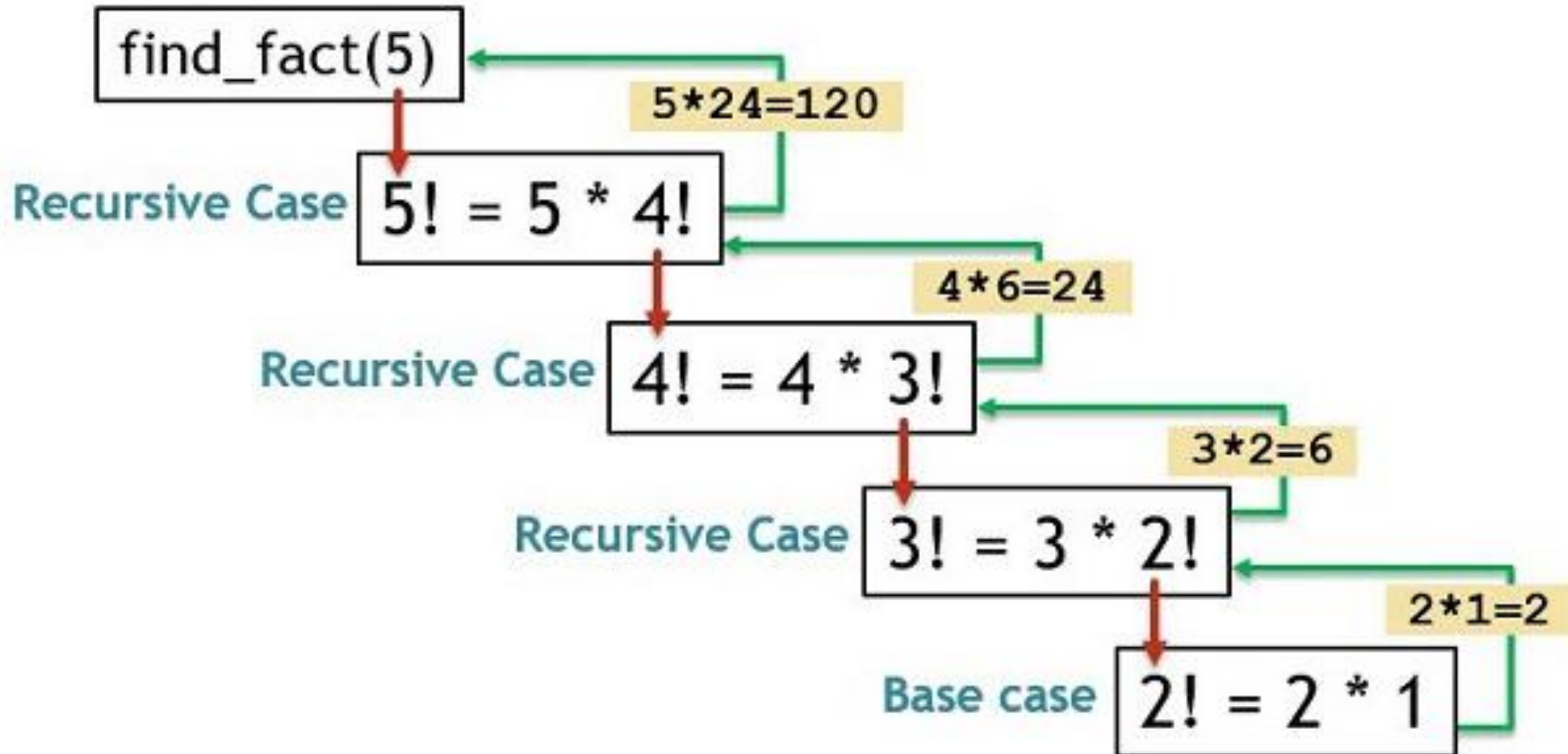
## SQL Subquery



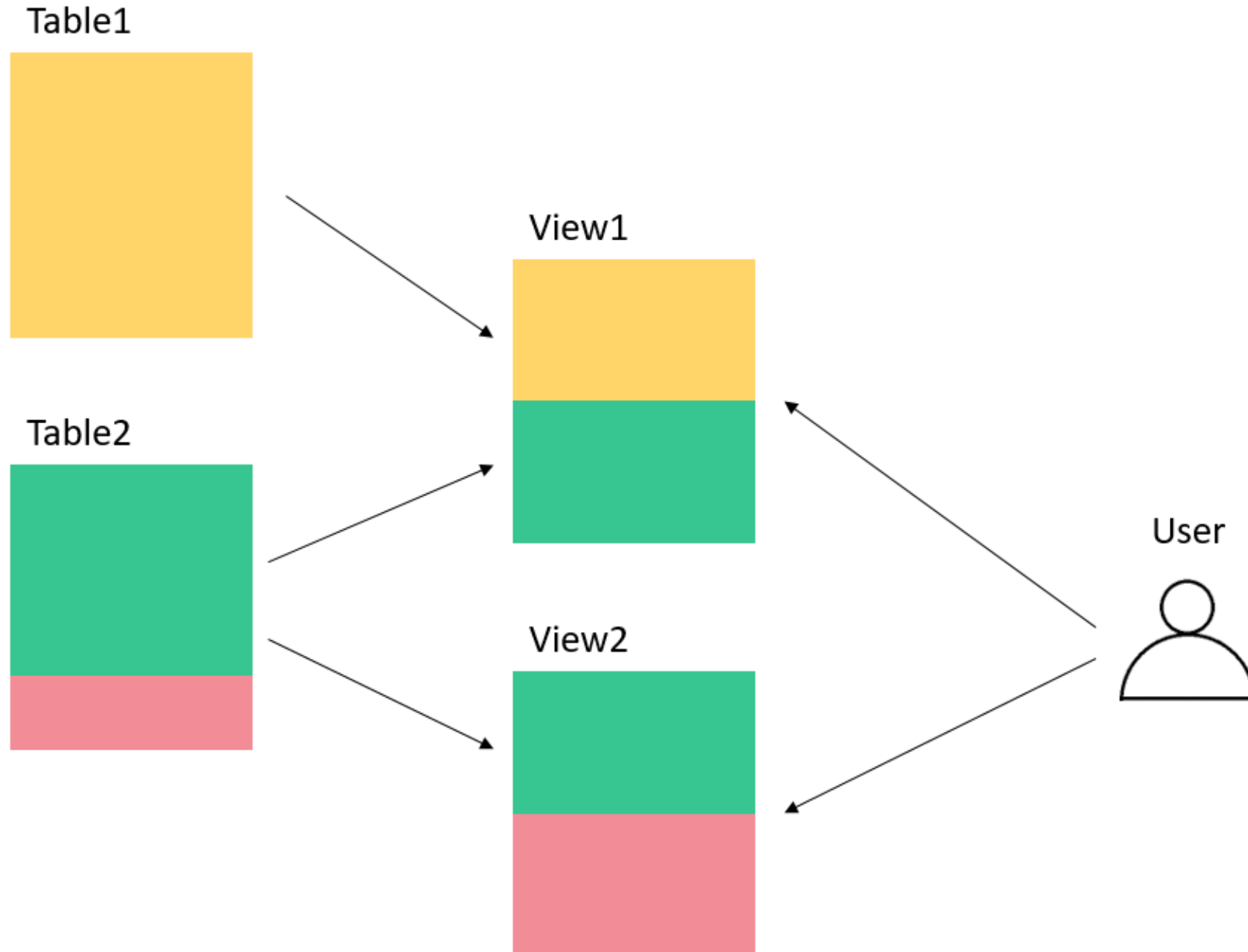
# Temporary Table

Table Variable	@
Temporary Table	#
Global Temporary Table	##

# 遞迴recursive



# SQL Server View



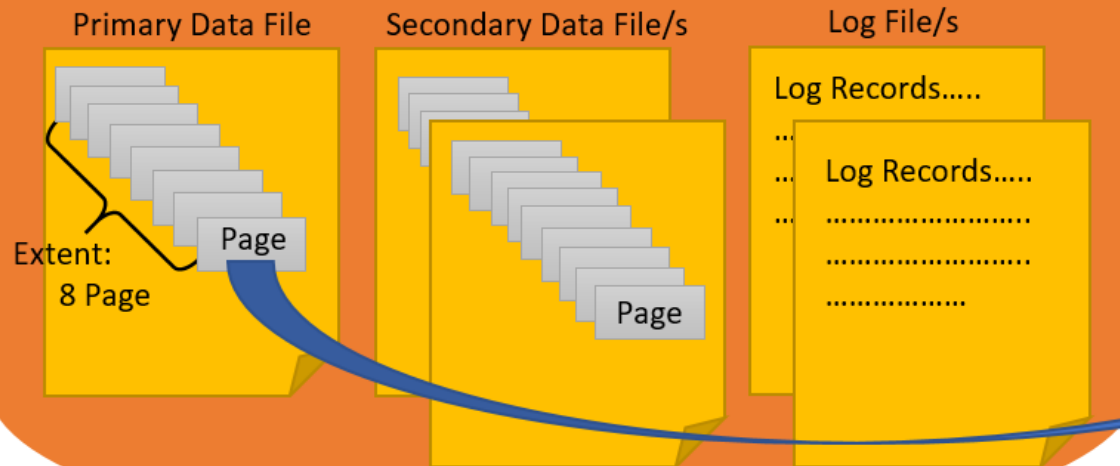


# SQL 資料庫結構

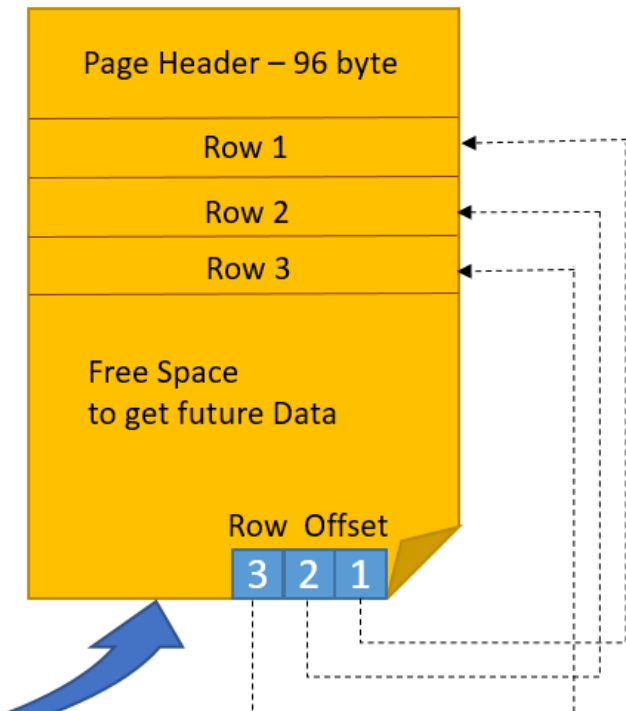
## SQL Server Database

**DB External View:** Tables, Views, SPs, Indexs, Triggers...etc.

**DB Internal View:** Data & Log Files

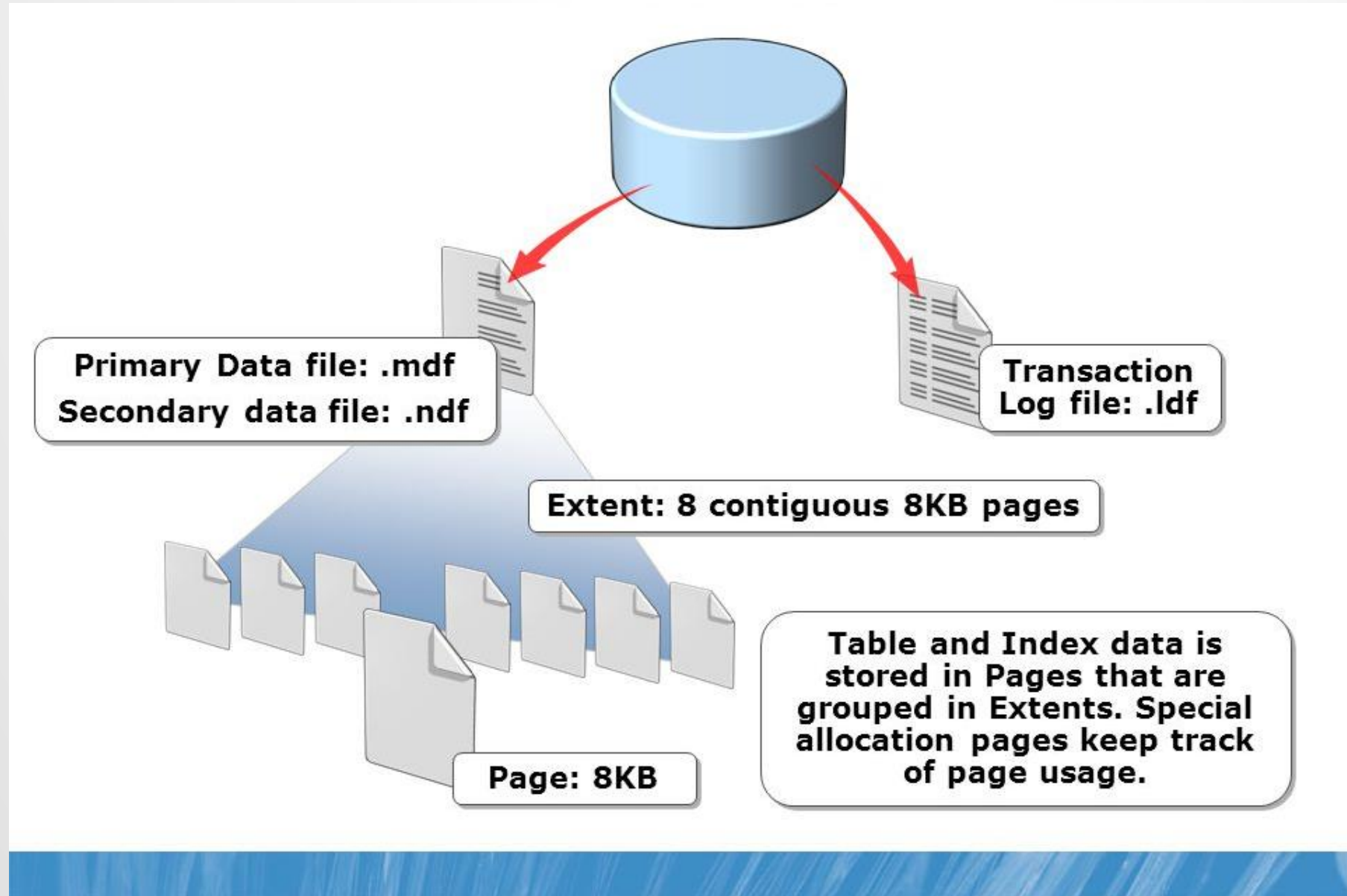


## SQL Server Data Page – 8K

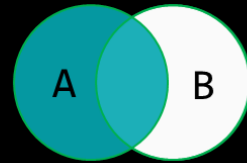




# SQL 資料庫結構



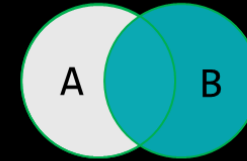
# SQL 資料庫 Union



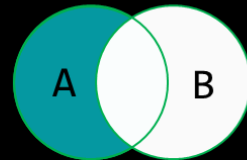
```
SELECT * FROM A LEFT  
JOIN B ON A.KEY = B.KEY
```



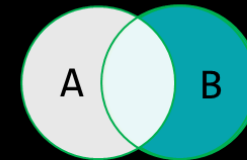
## UNIONES SQL



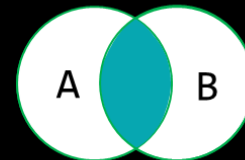
```
SELECT * FROM A RIGHT  
JOIN B ON A.KEY = B.KEY
```



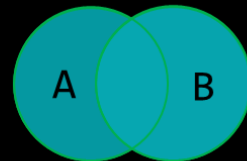
```
SELECT * FROM A LEFT JOIN  
B ON A.KEY = B.KEY WHERE  
B.KEY IS NULL
```



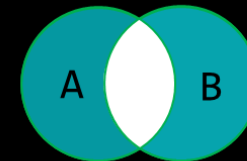
```
SELECT * FROM A RIGHT  
JOIN B ON A.KEY = B.KEY  
WHERE B.KEY IS NULL
```



```
SELECT * FROM A INNER JOIN B  
ON A.KEY = B.KEY
```



```
SELECT * FROM A FULL  
OUTER JOIN B ON A.KEY =  
B.KEY
```



```
SELECT * FROM A FULL  
OUTER JOIN B ON A.KEY =  
B.KEY WHERE A.KEY IS NULL  
OR B.KEY IS NULL
```



SerInge

# SQL 資料庫 Union

table : A

table : name		
id		
	a	
	b	
	c	

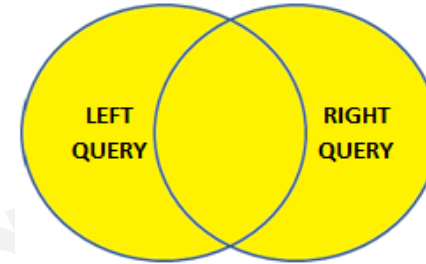
table : B

A union B = a+b+c

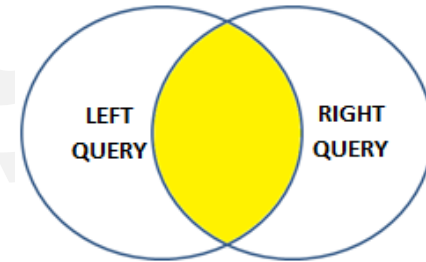
A union all B = a+b+b+c

A except B = a

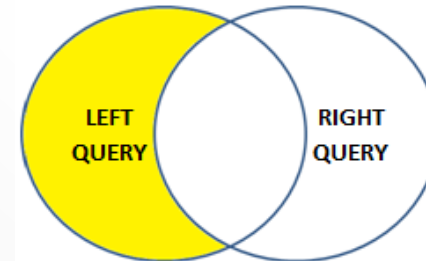
B except A = c



UNION operator returns all the unique rows from both the left and the right query. UNION ALL includes the duplicates as well

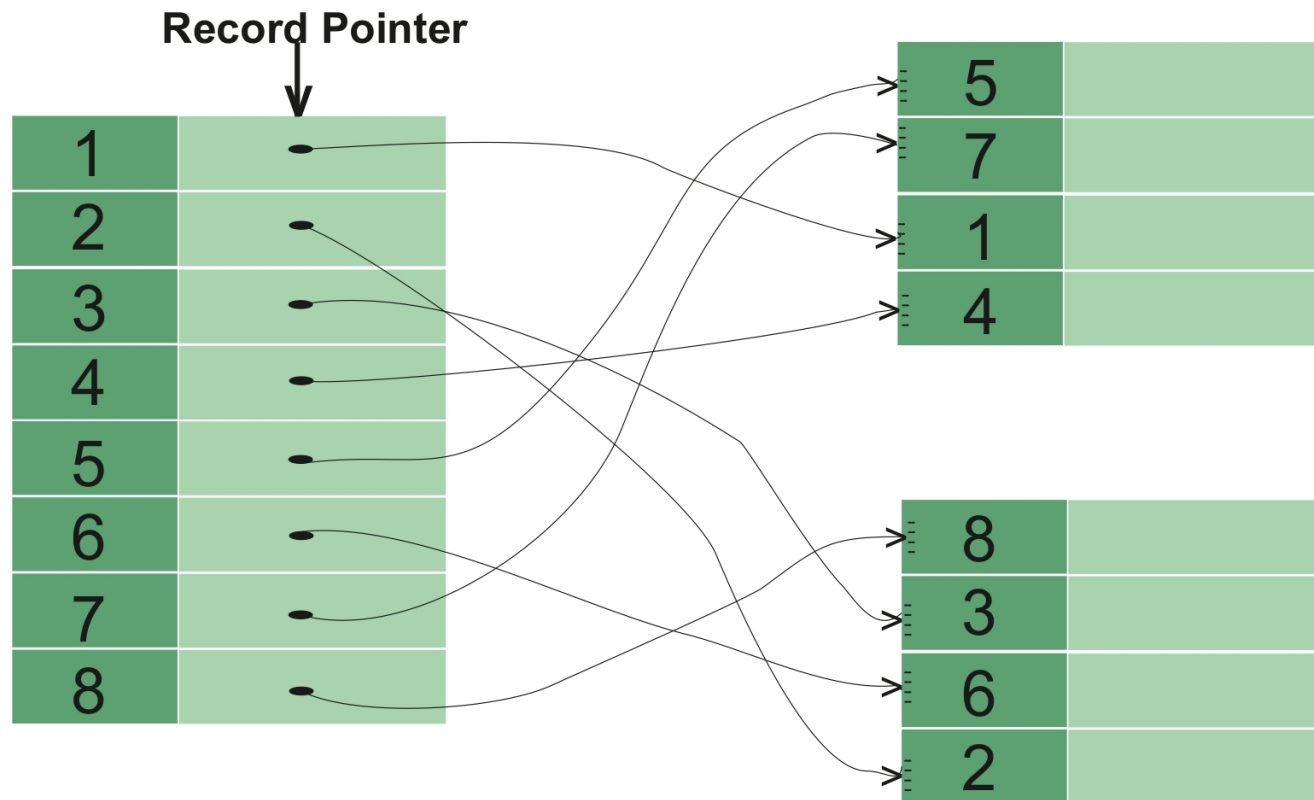


INTERSECT operator retrieves the common unique rows from both the left and the right query



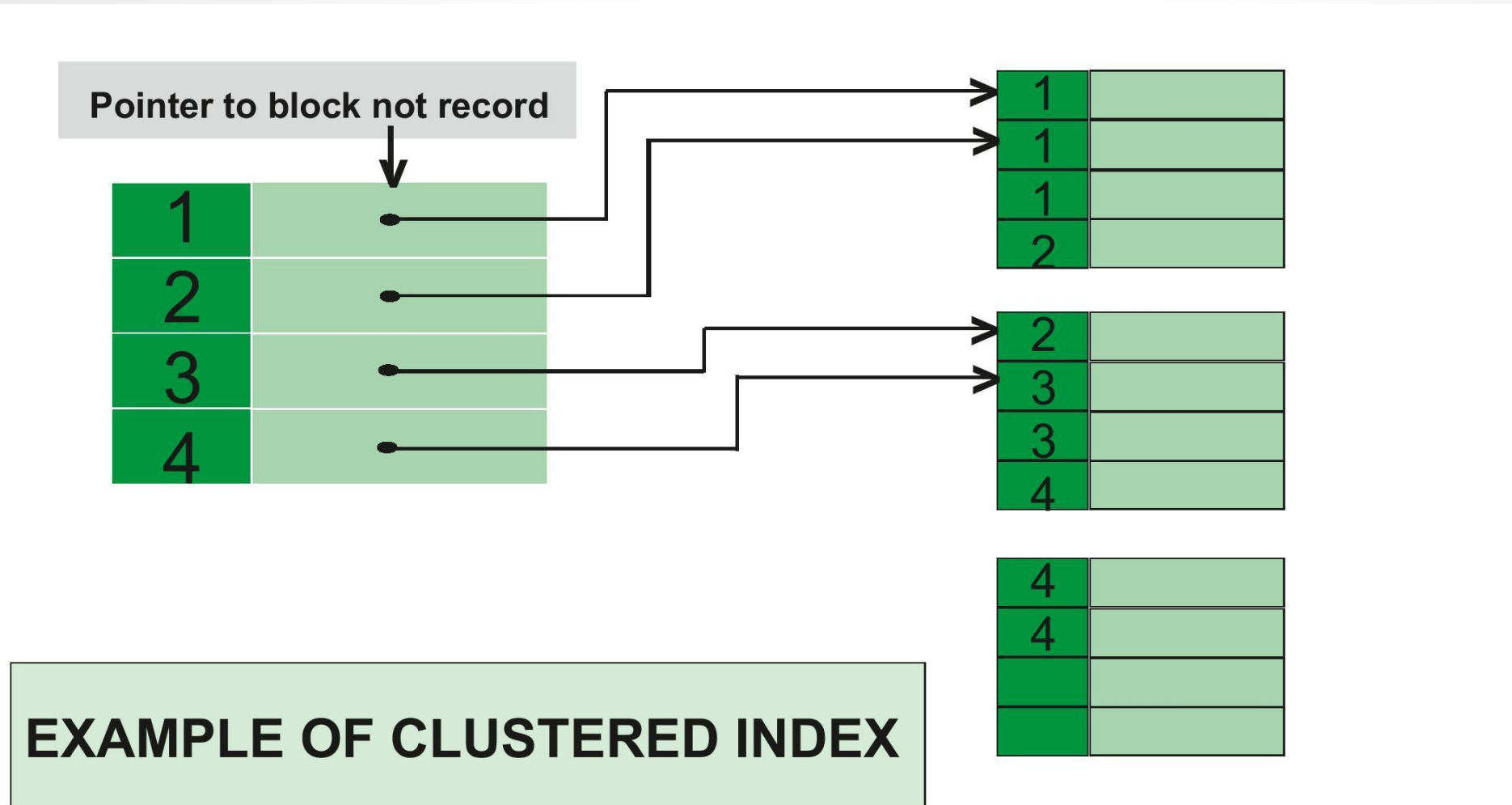
EXCEPT operator returns unique rows from the left query that aren't in the right query's results

# Non-clustered index & heap



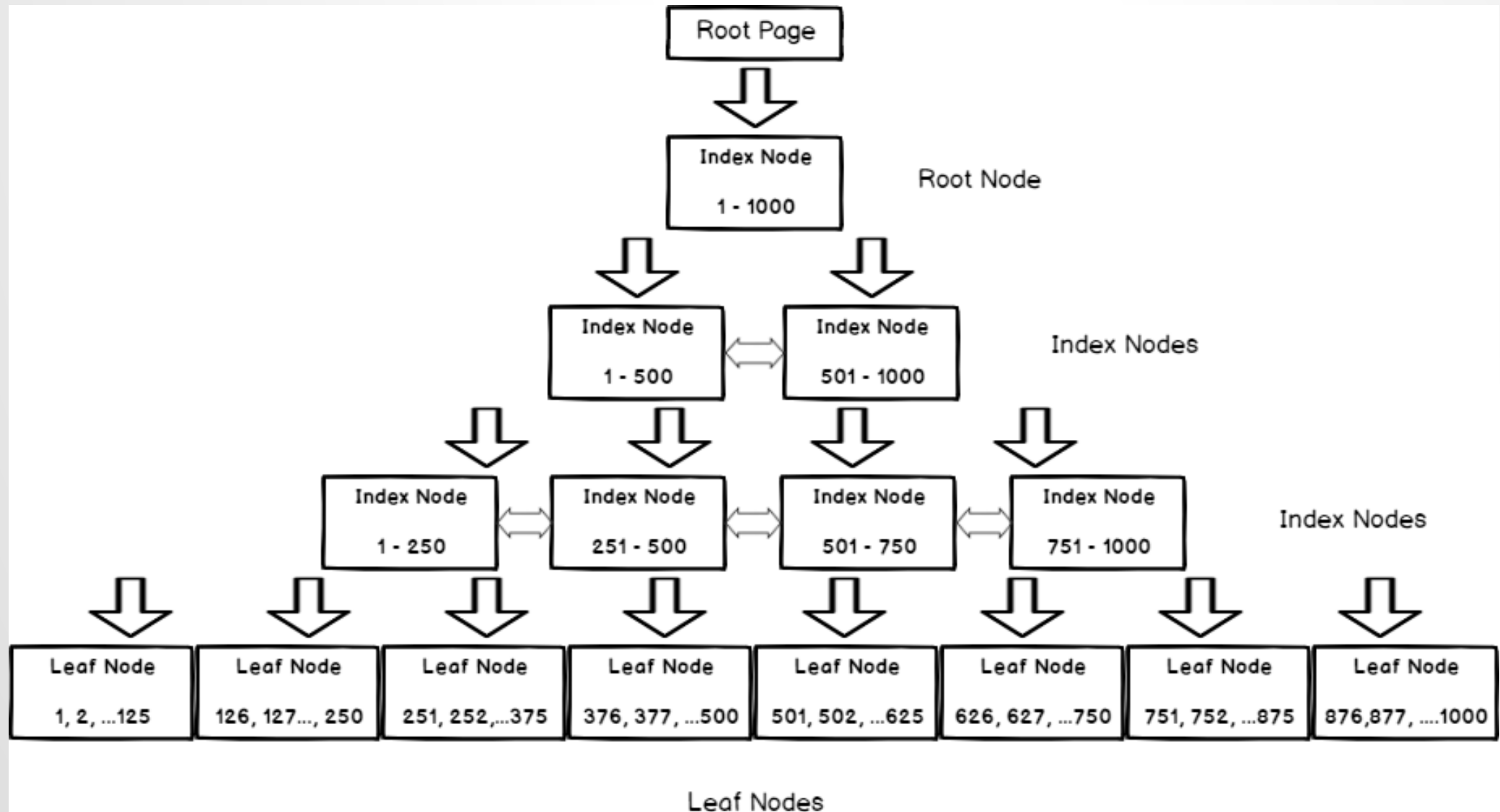
**EXAMPLE OF NON-CLUSTERED INDEX**

# Clustered index & clustered

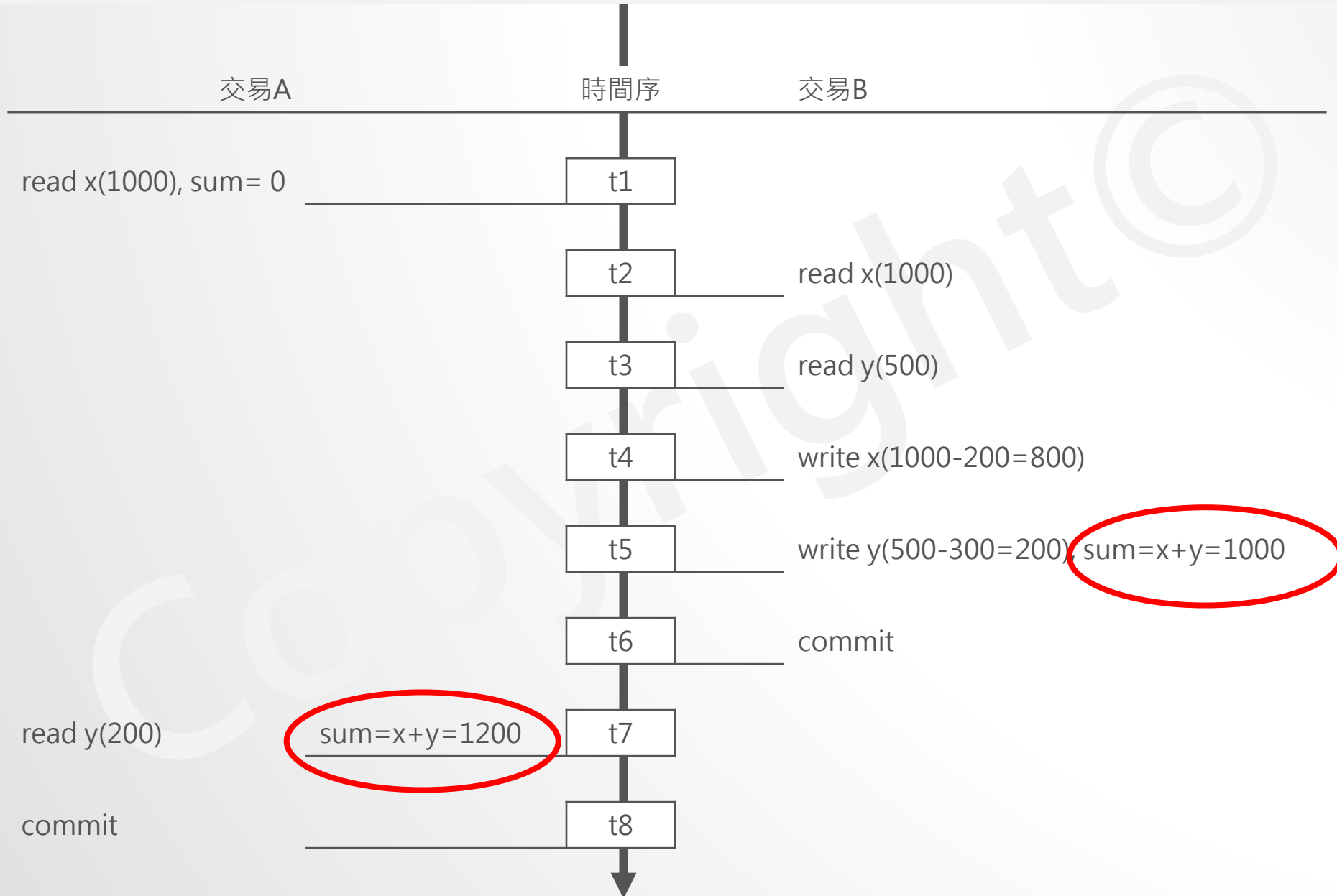


**Most Primary Key makes DB clustered**

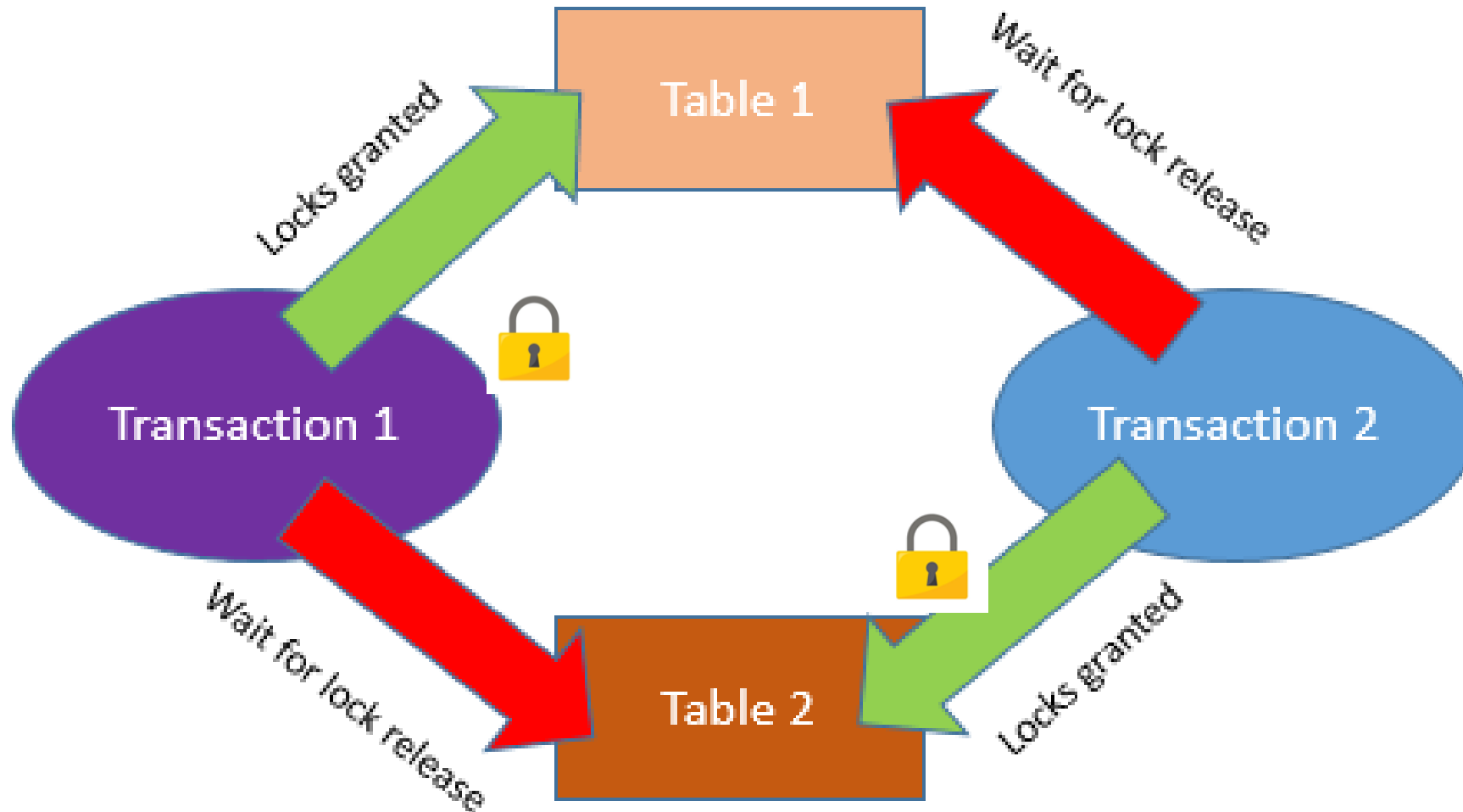
# Cluster Index (balance-tree index)



# 交易不一致

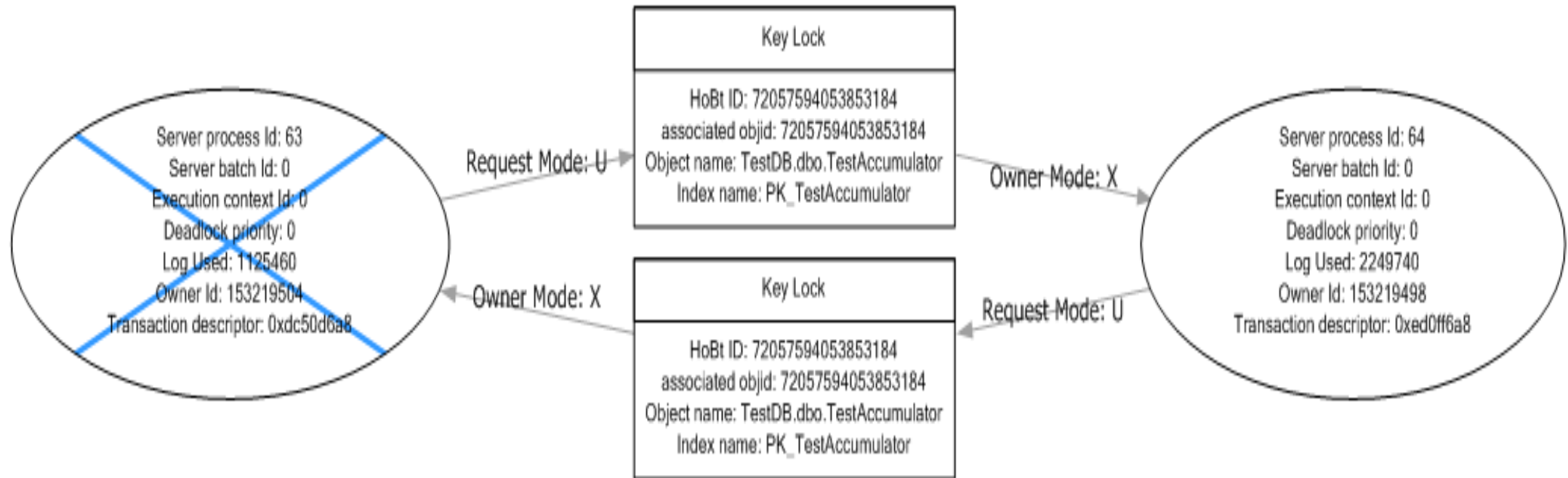


# DeadLock

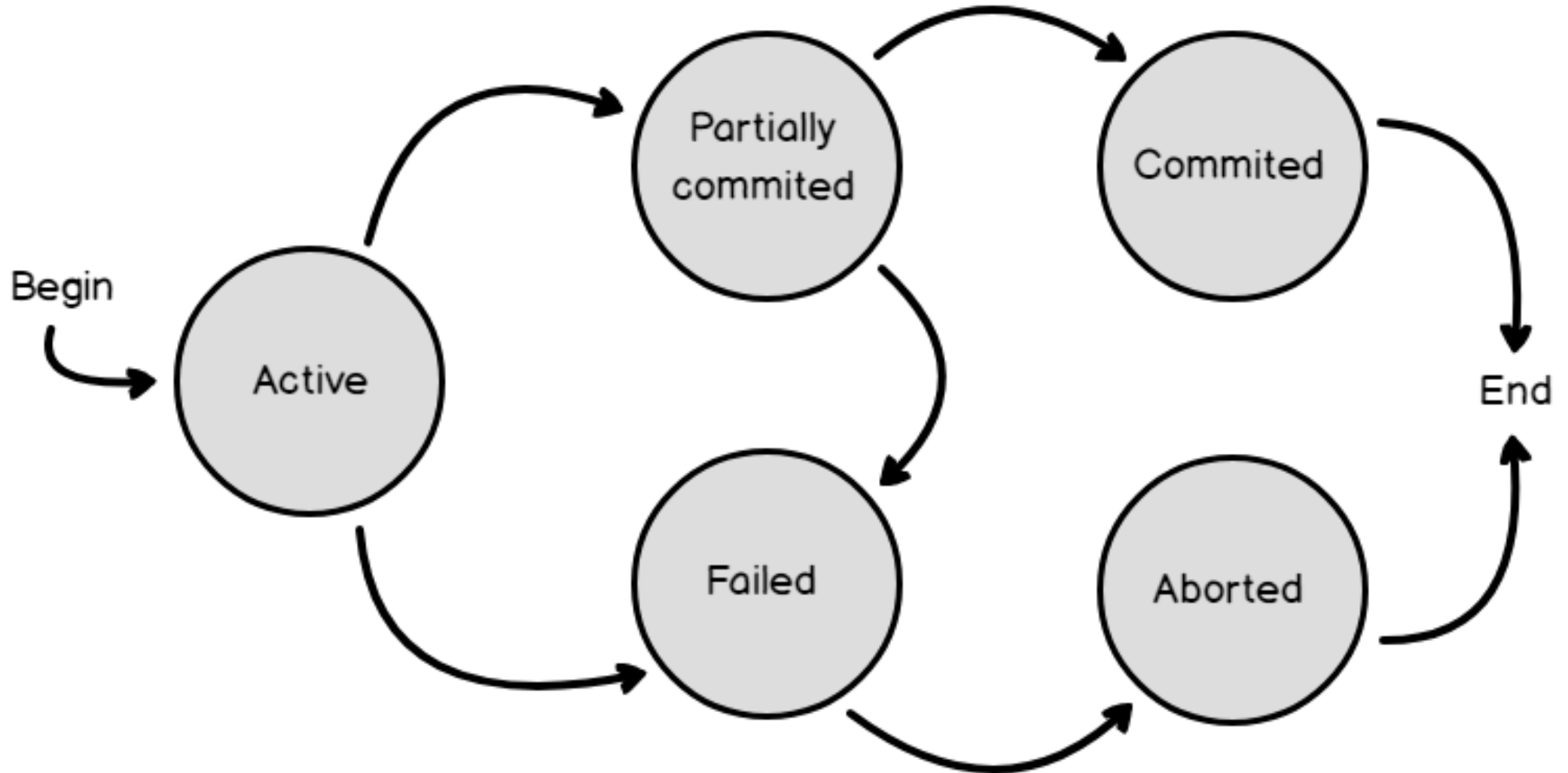




# DeadLock victim

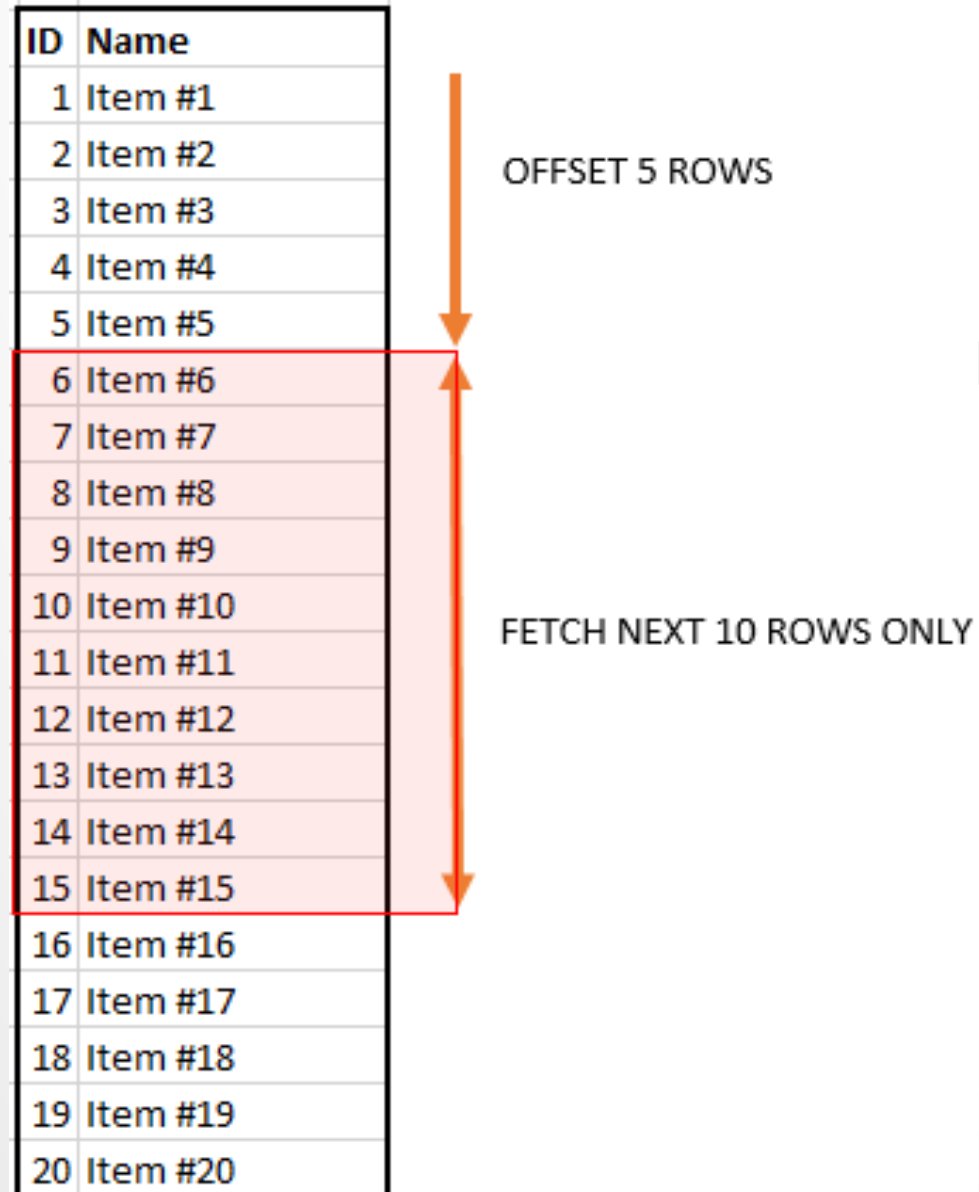


# Auto commit



# Offset function

ID	Name
1	Item #1
2	Item #2
3	Item #3
4	Item #4
5	Item #5
6	Item #6
7	Item #7
8	Item #8
9	Item #9
10	Item #10
11	Item #11
12	Item #12
13	Item #13
14	Item #14
15	Item #15
16	Item #16
17	Item #17
18	Item #18
19	Item #19
20	Item #20



OFFSET 5 ROWS

FETCH NEXT 10 ROWS ONLY

# Lag, lead functions

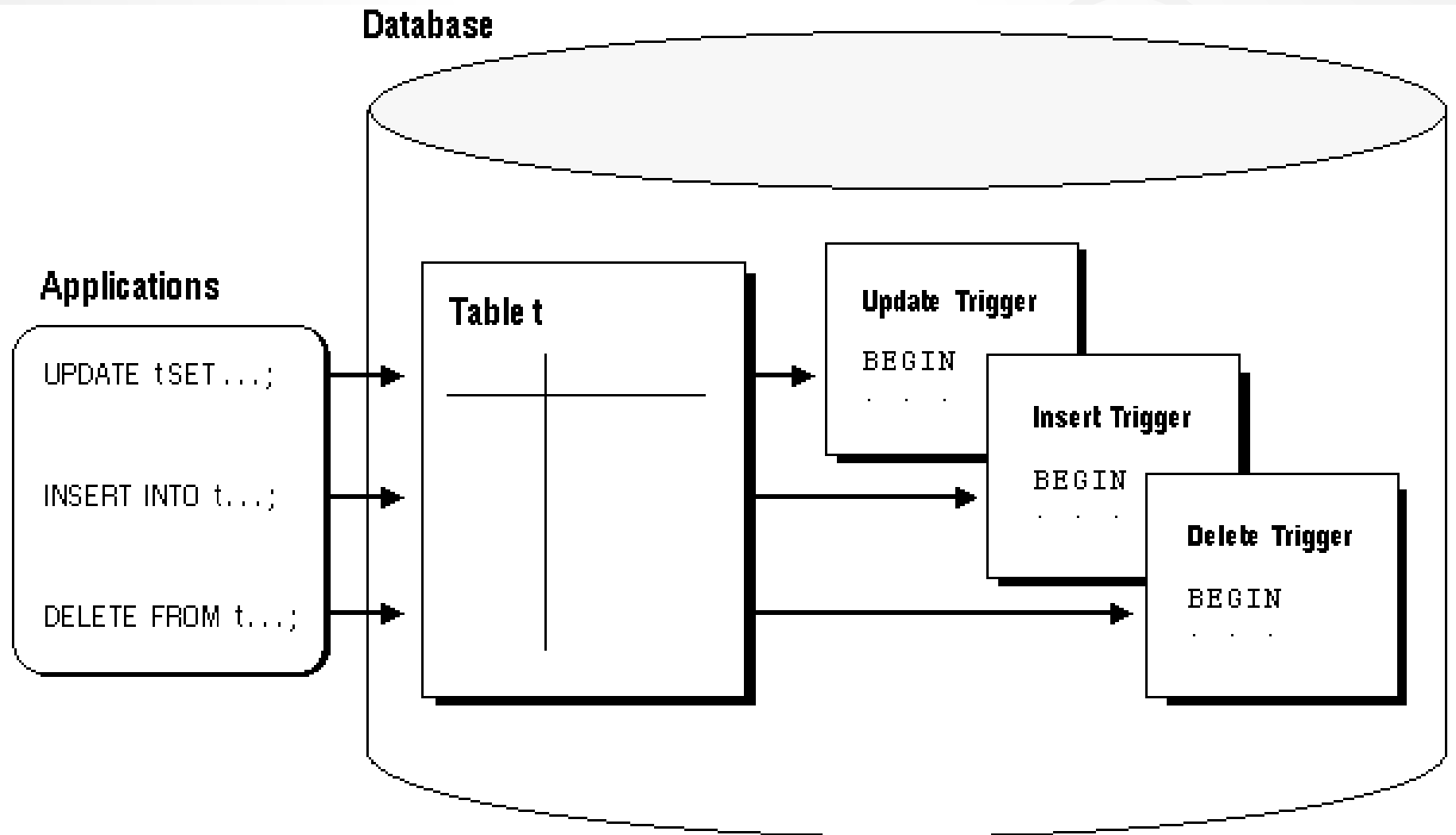
Lag

month	net_sales	previous_month_sales
1	381430	NULL
2	200657	381430
3	363990	200657
4	817920	363990
6	189	817920
7	11338	189
8	8378	11338
9	8964	8378
10	3781	8964
11	11362	3781
12	6517	11362

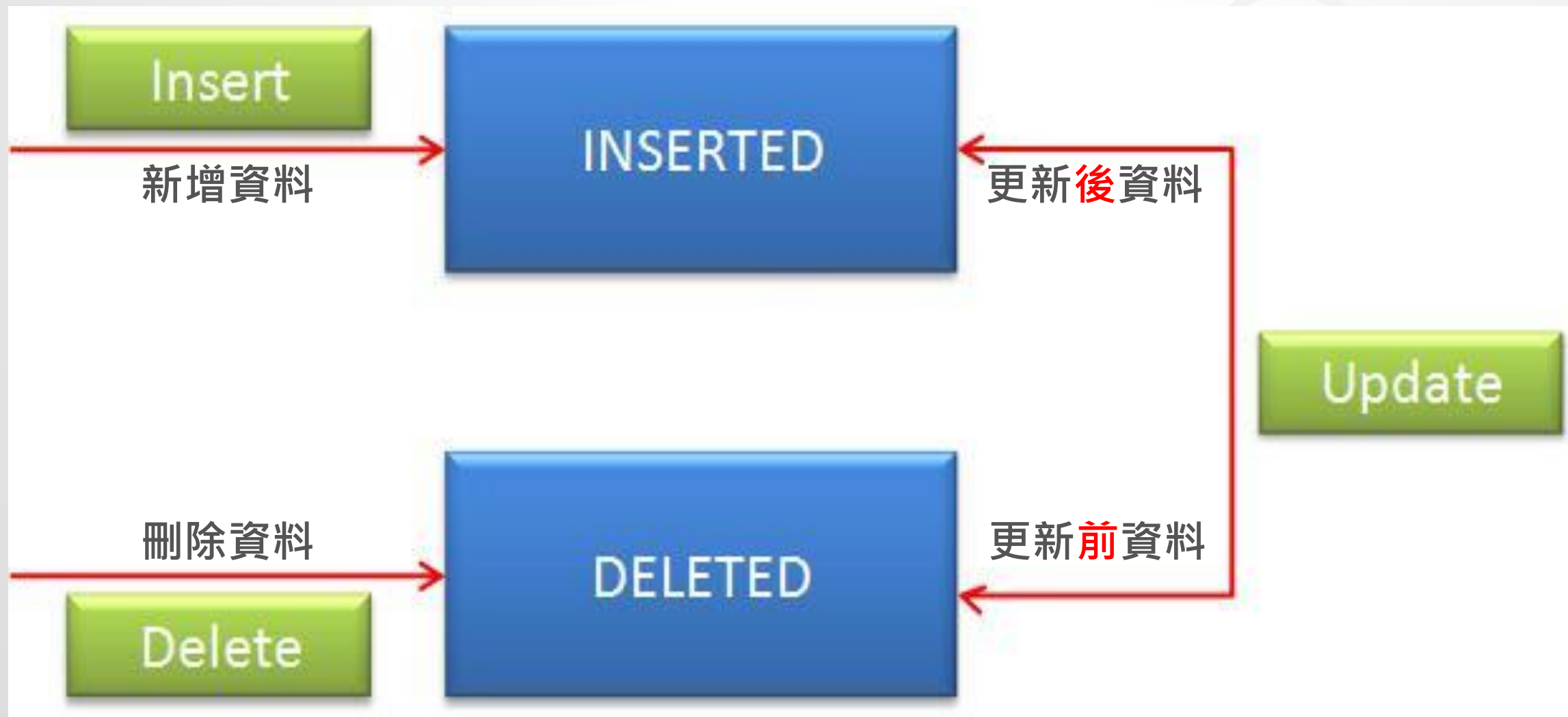
Lead

month	net_sales	next_month_sales
1	285617	312924
2	312924	308912
3	308912	227290
4	227290	268233
5	268233	378865
6	378865	229996
7	229996	290553
8	290553	293406
9	293406	310328
10	310328	281578
11	281578	259507
12	259507	NULL

# Trigger



# Inserted & deleted



**The end**