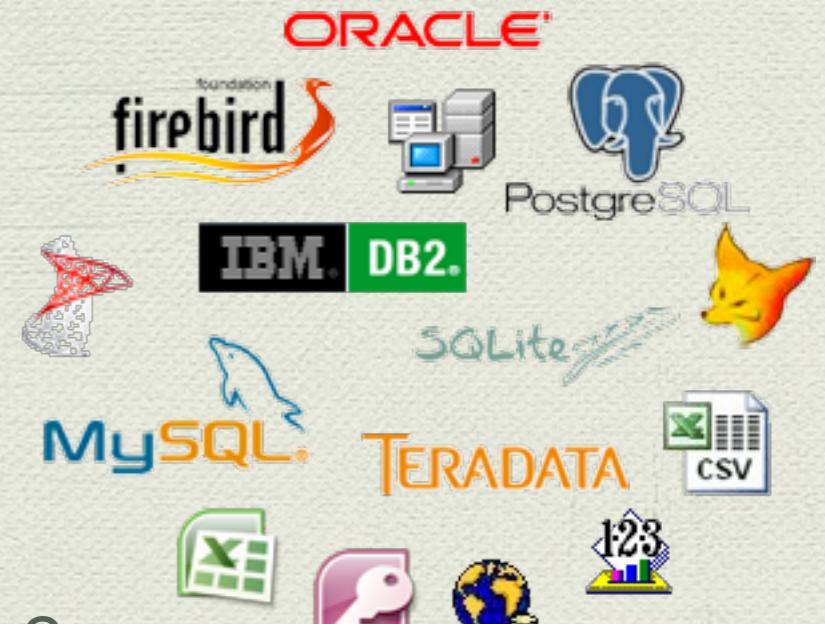


Web Programming 05/30

- * Database
- * SQL / NoSQL

What is a “database”?

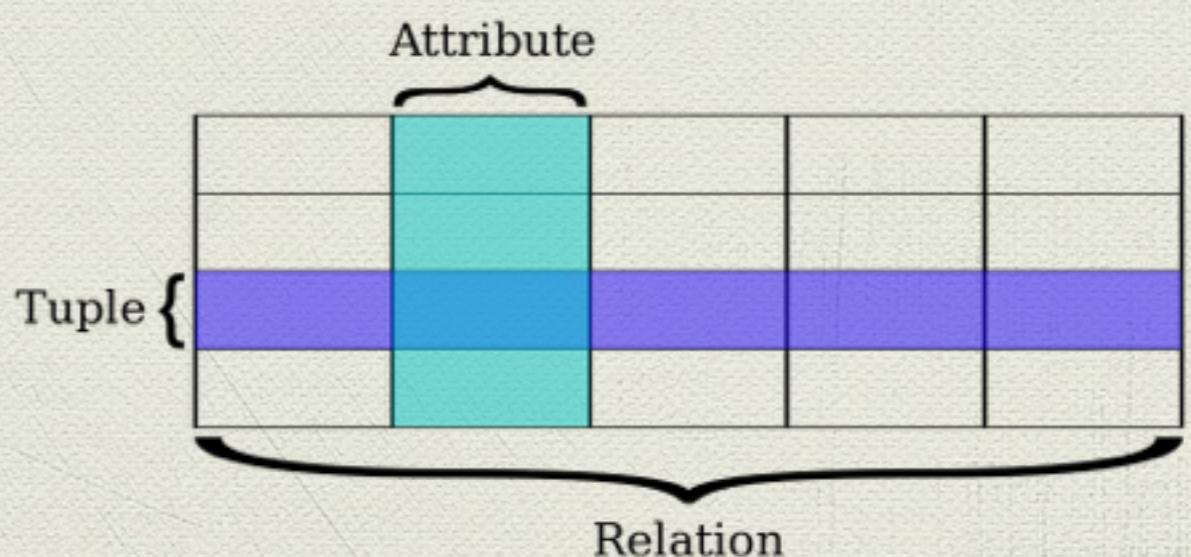
- ◆ Database? Data Structure?
- ◆ What is database? What is a database management system (DBMS)?
- ◆ Think...
 - ◆ 資料如何儲存在電腦裡面？檔案？記憶體？
 - ◆ 如何找到特定一筆資料？如何找到符合特定條件的資料？如何統計符合特定條件的資料？
 - ◆ 如何讓多人的資料可以多工的(被)操作？如何保證不會有讀寫覆蓋、資料不一致、或是資料遺失等問題？
 - ◆ 如何保證資料不會被竊取、竄改？



Database 的發展已經有超過 50 年以上
比較詳細的可以看 NoSQL 大腸花

Relational Database Management System (RDBMS)

- ◆ Relational? 關聯式資料庫?
- ◆ 比方說，一個 Excel Table
- ◆ RDBMS 架構
 - ◆ Server (伺服器)
 - ◆ Database (資料庫)
 - ◆ Table (資料表)
 - ◆ Column (欄位) // Attribute
- ◆ 每個 row (tuple) 代表一筆資料
 - ◆ Query 符合某個欄位的某些條件



我的 mysql 跑不起來...

- ◆ 有沒有 sudo 啊...
- ◆ mysqld 根本沒有跑起來
 - ◆ ps -ef | grep mysqld
 - ◆ 試試看 mysqld_safe
- ◆ 再不行，只好把錯誤訊息貼去問辜狗大神

In-Class Practice

- ◆ 方便起見，請安裝一個 SQL Client Tool：
 - ◆ SequelPro (Mac Only)：
<https://sequelpro.com/download>
 - ◆ MySQL Workbench:
<https://dev.mysql.com/downloads/workbench>
- ◆ 然後從 Ceiba 下載 “final_exam.csv” 以及 “final_exam 2.csv”

SAMPLE TABLE (FINAL_EXAM)

Name	Subject	Score
Amy	Math	30
Ben	Math	90
Calvin	Math	80
David	Math	100
Amy	English	90
David	English	100
Calvin	English	40
Ben	English	80
Amy	Chinese	80
David	Chinese	70
Calvin	Chinese	100
Ben	Chinese	50
Amy	Science	100
David	Science	60
Calvin	Science	90
Ben	Science	80
Amy	Social_Science	70
David	Social_Science	20
Calvin	Social_Science	60
Ben	Social_Science	100

SQL: Structured Query Language

- ◆ Database
 - ◆ CREATE/USE DATABASE
- ◆ Table
 - ◆ CREATE TABLE
 - ◆ INSERT/UPDATE/DELETE/LOAD data to table
- ◆ Query
 - ◆ SELECT data from table
- ◆ Data types: numerical, date / time, string

把 final_exam.csv 讀進 DB

1. 首先，連接到 localhost 的 DB server

- ◆ Default port: 3306

2. 建立一個 database，並指定使用它

- ◆ `CREATE DATABASE testDB;`
- ◆ `USE testDB;`

3. 建立一個 table

- ◆ `CREATE TABLE final_exam (Name VARCHAR(255), Subject VARCHAR(255), Score INT);`

4. 把資料讀進去

- ◆ `INSERT INTO final_exam (Name, Subject, Score) VALUES ('Amy', 'Math', 30);`
- ◆ `INSERT INTO final_exam (Name, Subject) VALUES ('Ben', 'Math');`

RDBMS 的特性

- ◆ Key Words: Case Insensitive
- ◆ Column Name & Value: Case Sensitive
- ◆ Constraints
 - ◆ Primary key (PK): uniquely specifies a tuple within a table
 - ◆ Foreign key (FK): a field in a relational table that matches the primary key column of another table
 - ◆ Index(ing): to speed up the access of data
- ◆ What are the PKs and FKs in the tables of the studied case?

SQL Data Types

- ◆ Int
- ◆ Float
- ◆ Varchar
- ◆ Text
- ◆ Date
- ◆ 不同 DB 有些微的不同

Using SQL Client Program

- ◆ 不過，如果使用 MySQL Client-side Program, 通常你不會使用到上述指令

[練習] Import data from file (e.g. csv)

- ◆ You may need to:
 - Create connection to your localhost database server
 - Create a new database
 - Create a new table
 - Select “import data”

UPDATE table

- ◆ 修改符合條件的資料：
 - ◆ `UPDATE final_exam SET Score=90 WHERE Name='Ben' AND Subject='Math';`
- ◆ 修改所有資料：
 - ◆ `UPDATE final_exam SET Score=0;`
 - ◆ `UPDATE final_exam SET Score=0 WHERE 1=1;`

DELETE data from table

- ◆ 刪除符合條件的資料：
 - ◆ `DELETE FROM final_exam WHERE Name='Amy' AND Subject='Math';`
 - ◆ 刪除所有資料
 - ◆ `DELETE FROM final_exam;`
 - ◆ `DELETE FROM final_exam WHERE 1=1;`

LOAD DATA from file

- ◆ **LOAD DATA LOCAL INFILE
‘[your_file_path]’
INTO TABLE final_exam
FIELDS TERMINATED BY ‘,’
ENCLOSED BY “”
LINES TERMINATED BY ‘\n’
IGNORE 1 ROWS;**

SQL 最主要的功能當然是讓大家利用它來“query” database 的內容啦！

```
SELECT  
    [ALL | DISTINCT | DISTINCTROW ]  
    [HIGH_PRIORITY]  
    [STRAIGHT_JOIN]  
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]  
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]  
    select_expr [, select_expr ...]  
    [FROM table_references]  
    [WHERE where_condition]  
    [GROUP BY {col_name | expr | position}  
        [ASC | DESC], ... [WITH ROLLUP]]  
    [HAVING where_condition]  
    [ORDER BY {col_name | expr | position}  
        [ASC | DESC], ...]  
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]  
    [PROCEDURE procedure_name(argument_list)]  
    [INTO OUTFILE 'file_name' export_options  
        | INTO DUMPFILE 'file_name'  
        | INTO var_name [, var_name]]  
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

Select 指令(重點)

SELECT select_expr

FROM table

WHERE constraints

GROUP BY grouping method (e.g. col_name)

ORDER BY ordering

LIMIT row_count

;

SELECT

- ◆ 選出所有資料：
 - ◆ `SELECT * FROM final_exam;`
- ◆ 選出特定欄位資料：
 - ◆ `SELECT Name, Score FROM final_exam;`
- ◆ 選出前5筆資料：
 - ◆ `SELECT * FROM final_exam LIMIT 5; //MySQL`
 - ◆ `SELECT TOP 5 * FROM final_exam; //MSSQL`

SELECT

- ◆ 計算考卷總數：
 - ◆ `SELECT COUNT(*) FROM final_exam;`
- ◆ 計算考生總數：
 - ◆ `SELECT COUNT(DISTINCT Name) FROM final_exam;`
- ◆ 計算所有考卷平均分數：
 - ◆ `SELECT AVG(Score) AS avg_score FROM final_exam;`
- ◆ 計算個人總分：
 - ◆ `SELECT Name, SUM(Score) AS sum_score FROM final_exam GROUP BY Name;`

SELECT (Challenges!)

- ◆ 計算科目平均分數並由平均分數低到高排序：
 - ◆ `SELECT Subject, AVG(Score) AS avg_score
FROM final_exam GROUP BY Subject ORDER
BY avg_score;`
- ◆ 選出滿分或不及格的考卷並由分數高到低排序：
 - ◆ `SELECT * FROM final_exam WHERE Score =
100 OR Score < 60 ORDER BY Score DESC;`

另外 SQL 很重要的一個功能是增加新的欄位

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

UPDATE (challenges!)

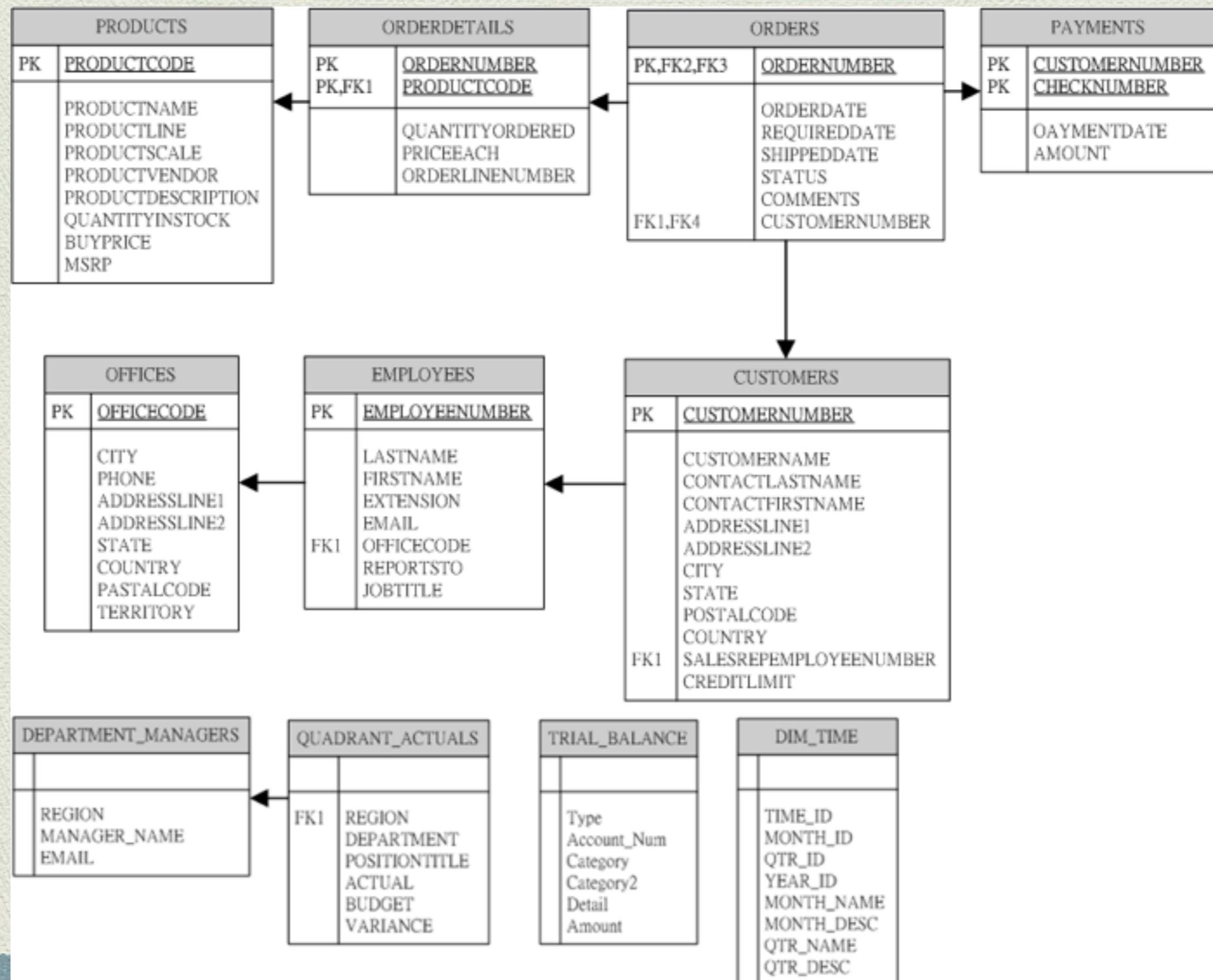
- ◆ 新增一個欄位，記錄考試成績是否及格(≥ 60)，如果及格，填入“及格”，否則填入“不及格”
- ◆ `UPDATE final_exam SET 'pass_or_fail' = IF ('Score' >= 60, '及格', '不及格');`

RDBMS 的 ACID 原則

- ◆ 原子性 (atomicity)：transaction 中的所有操作，要麼全部完成，要麼全部不完成。發生錯誤，會被 rollback
- ◆ 一致性 (consistency)：寫入的資料必須完全符合所有的預設規則，包含資料的精確度等等
- ◆ 隔離性 (isolation)：資料庫允許多個並發事務同時對其數據進行讀寫和修改的能力，隔離性可以防止多個事務並發執行時由於交叉執行而導致數據的不一致
- ◆ 持久性 (durability)：對資料庫所作的改變持久地保存在資料庫之中

設計一個 RDB 最重要的是要先定義好它的 Data Schema

◆ Example:



RDBMS 的正規化

- ◆ 緣由：當相同的資訊出現在不同的 tables, 會有資料不一致的問題
 - ◆ 在課程表的資料庫裡頭，「課程名稱」同時出現在「教師」以及「教室」兩個 tables 裡面
 - ◆ 如果有一天在「教師」table 更改課名，在「教室」table 上的資料是否會被更新？
- ◆ What are the PKs and FKs in the tables of the studied case?

正規化(Normalization)

- ◆ To remove structural redundancies and inconsistencies
- ◆ 1st normal form
 - ◆ Repeating attribute groups
- ◆ 2nd normal form
 - ◆ Functional dependencies
- ◆ 3rd normal form (3NF)
 - ◆ Transitive dependencies

Given a SQL table, 如何進行正規化？

- ◆ 刪除某些欄位
 - ◆ `ALTER TABLE table_name DROP COLUMN col_name;`
- ◆ 將不同面向的資料拆開，建立在不同的表單上
 - ◆ `CREATE TABLE new_table (
 SELECT expression AS col_name,...
 FROM source_table
 WHERE constraints
 GROUP BY col_name,...
)`
 - ◆ 誰是 primary key? 誰是 foreign key?

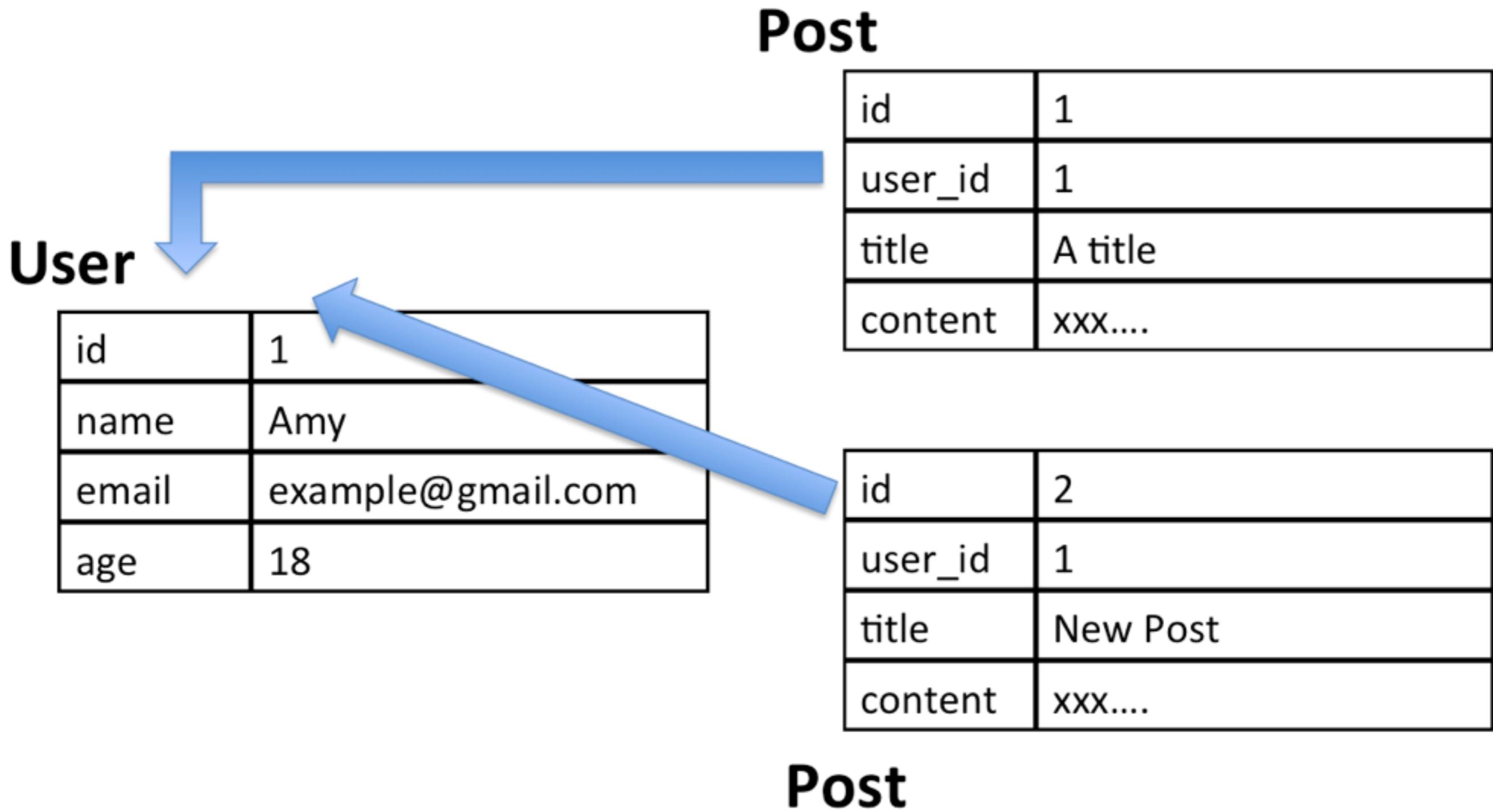
Recall: PRIMARY KEY

- ◆ 中文：主鍵
- ◆ 每張資料表只能有一個Primary Key
- ◆ 但Primary Key可以由多個欄位組成
- ◆ 同一張資料表內，Primary Key的值不能重複亦不能有NULL

PRIMARY KEY

- ◆ 設定Primary Key：
 - ◆ `ALTER TABLE final_exam ADD PRIMARY KEY(Name);` //會失敗，因為Name有重複值。
 - ◆ `ALTER TABLE final_exam ADD CONSTRAINT pk_final_exam PRIMARY KEY(Name, Subject);`
- ◆ 刪除Primary Key
 - ◆ `ALTER TABLE final_exam DROP PRIMARY KEY;` //MySQL
 - ◆ `ALTER TABLE final_exam DROP CONSTRAINT pk_final_exam;` //MSSQL

FOREIGN KEY



INDEX

- ◆ 中文：索引
- ◆ 每張資料表可以有多個INDEX
- ◆ INDEX可以由多個欄位組成
- ◆ INDEX的值可以重複也可以有NULL

INDEX

- ◆ 創造INDEX：
 - ◆ **CREATE INDEX name_index ON final_exam (Name);**
 - ◆ **CREATE INDEX name_and_subject_index ON final_exam (Name, Subject);**
- ◆ 刪除INDEX：
 - ◆ **DROP INDEX final_exam.name_and_subject_index;**
 //MSSQL
 - ◆ **ALTER TABLE final_exam DROP INDEX name_and_subject_index;** //MySQL

UNIQUE

- ◆ 每張表可以有多個UNIQUE
- ◆ UNIQUE可以由多個欄位組成
- ◆ 同一張資料表內，UNIQUE的值不能重複但可以有NULL

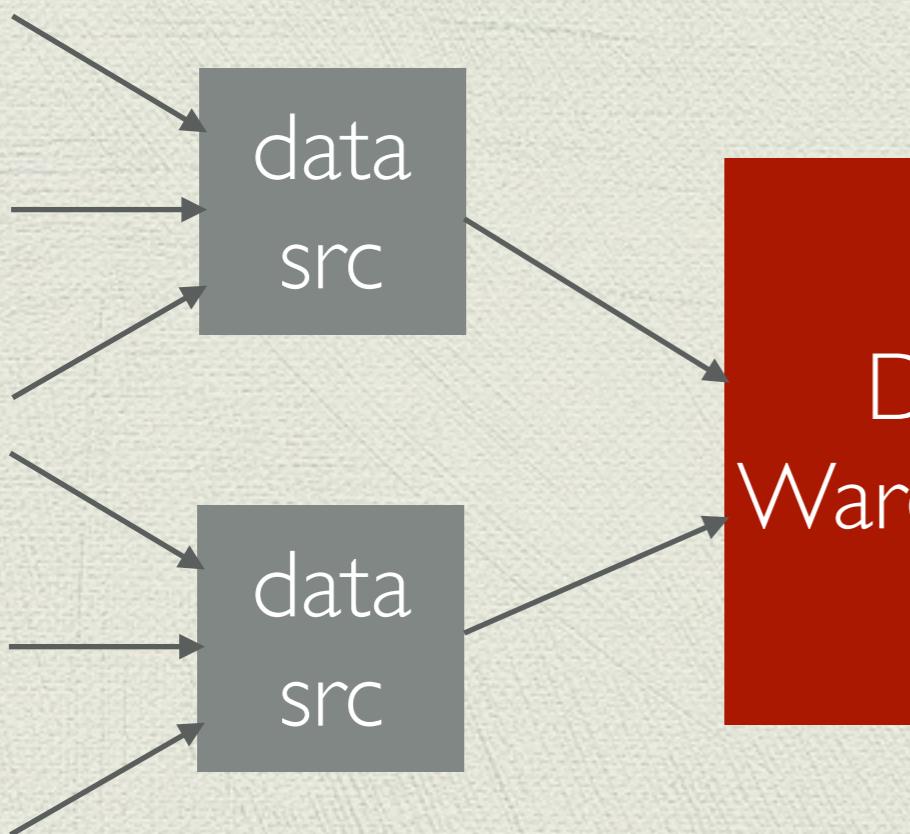
INDEX的優缺點

- ◆ 優點：
 - ◆ 加快SELECT速度 ($O(n) \rightarrow O(\log n)$)
- ◆ 缺點：
 - ◆ 創造INDEX費時、減慢INSERT及DELETE速度
- ◆ 原理：
 - ◆ 每個INDEX都代表一個Binary Tree

Data Processing Flow

資料收集(OLTP)

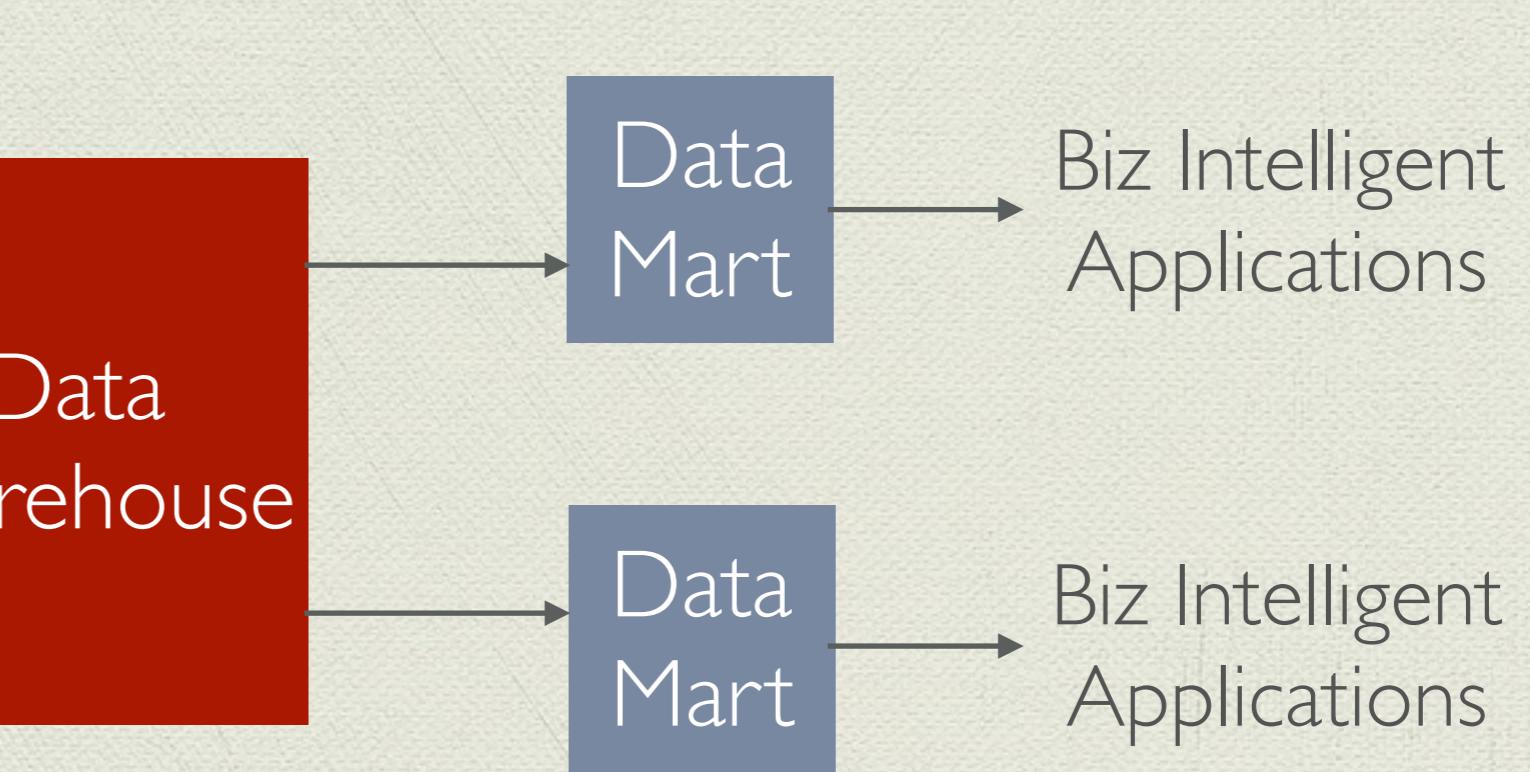
Efficient data insert, update
Multi-client access



data normalization

資料分析(OLAP)

Selected interested data
Efficient computation



dimensional model

JOIN : 合併數張資料表

- ◆ Denormalizatation (去正規劃)

- ◆ SELECT

```
m.Name, m.Subject, m.Score AS Midterm_Score,  
f.Score AS Final_Score  
FROM midterm_exam AS m  
INNER JOIN final_exam AS f  
ON m.Name=f.Name AND  
m.Subject=f.Subject;
```

Different types of join

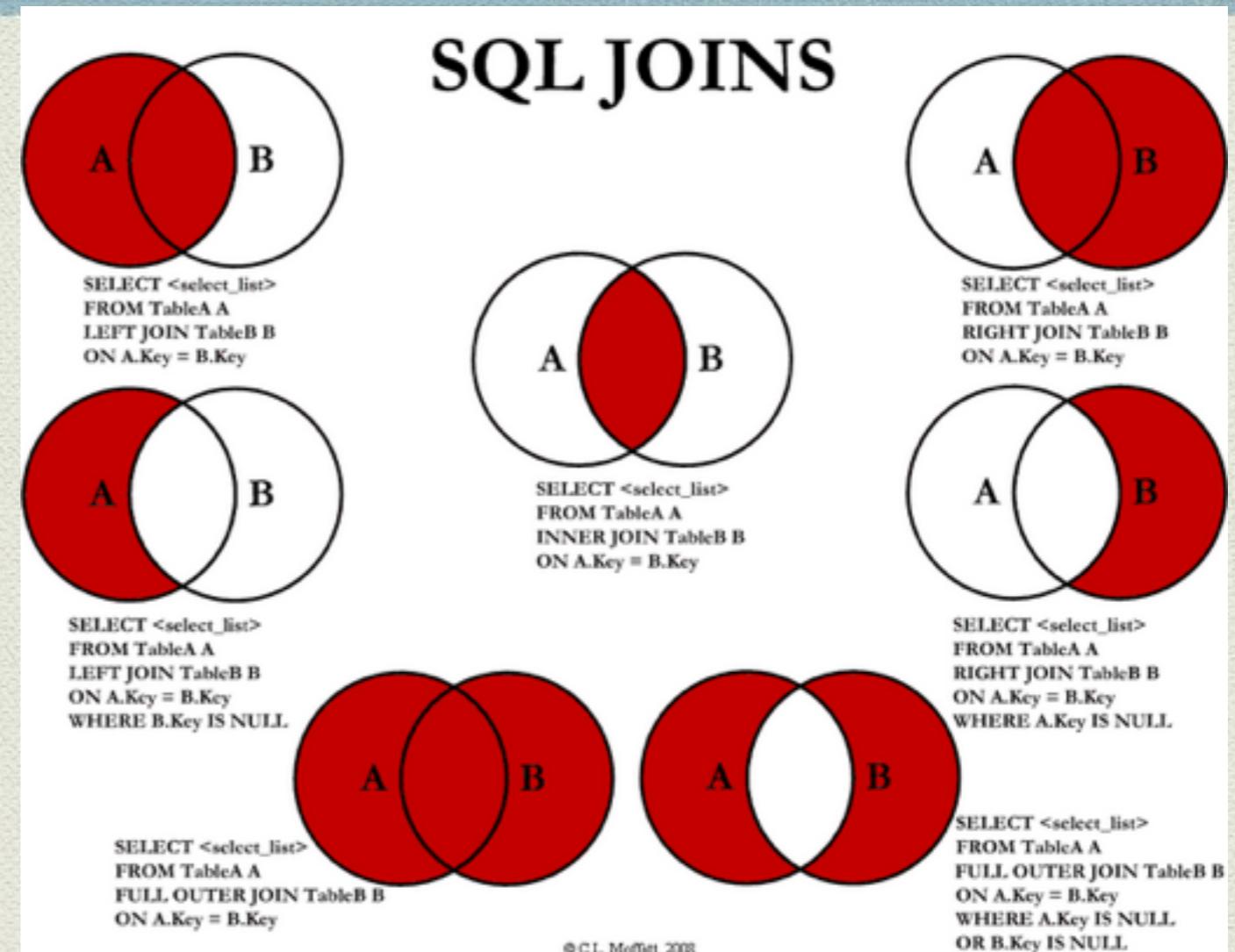
- ◆ INNER JOIN

- ◆ LEFT JOIN

- ◆ RIGHT JOIN

- ◆ OUTER JOIN

- ◆ 其他 : CROSS JOIN, STRAIGHT_JOIN, NATURAL XX JOIN



一些資料分析案例

1. 某品牌服飾商想要知道他們的新增會員數為何減少？如何增加線上電商的交易量？
2. 某手機公司想要分析其在各論壇與粉絲頁的社群資料，找出社群中之意見領袖，希望能從他們來帶給其他客戶正面的影響
3. 某知名汽車廠牌推出一套汽車保養套餐，不知為何客戶青睞度很低，希望找出原因
4. 某人壽保險公司希望能利用雲端高速的數據分析，有效地抓出可能的金融詐騙行為
5. 某熱門論壇每日有上千上萬篇文章，活躍的讀者有數十萬人，想要實現一個個人化的文章推薦系統
6. 某醫美診所想要根據過去正顎銷骨手術的圖片資料，去預估新的客戶術後的長相
7. 某國家級的醫療中心收集了許多癌症病人組織切片的蛋白體質譜儀的資料，希望能透過定序、定量的分析，找出早發性病人的發病原因

Case study: 一個 DVD 出租店的分析實例



案例大綱

- ◆ Problem statement
- ◆ Data outline and database (schema) design
- ◆ Data analytics for mass market strategy
- ◆ Data analytics for customer-centric strategy
- ◆ (Not done yet) Predictive model

討論問題(一)

- ◆ 從案例的 part I, 他們在面對一個商業問題的時候，想要用資料分析的方法來解決，首先，要從哪邊開始下手？
 - ◆ 定義分析的需求與目標
 - ◆ 確認資料來源是否充足與可靠？
 - ◆ 設計資料庫欄位格式 (schema)
 - ◆ 確認此問題是否需要進行資料分析？

討論問題 (二)

- ◆ 從案例 part I 的 database design, 他們是怎麼樣去設計與規劃 database 的 schema 呢？
- ◆ Data base 的設計好之後，考量實際收集資料的情境，還需要注意什麼問題？為什麼要分成這麼多個 tables 呢？
 - ◆ 安全性
 - ◆ 正確性 (data integrity)
 - ◆ (效率) 存取時間 與 儲存空間
- ◆ Recall: OLTP vs. OLAP

討論問題 (三)

- In Part II, 作者用了以下的統計表格來檢視出租店第三個月的營運狀況：

Total customers	300
Total CDs bought	1000
CDs rented	450
Total Revenue generated (INR)	6750
Fixed Cost (INR)	30000
CDs cost (INR)	2000
Net Profit	-25250

- 從原始的 database schema, 如何利用 SQL 的 queries 來獲取以上的資訊？

其他的統計表格

Language	Genre	#movies available	# movies rented
Hindi	Action	150	5
	Romantic	150	220
	Sci-fi	150	0
	Comedy	150	5
English	Action	100	210
	Romantic	100	0
	Sci-fi	100	5
	Comedy	100	5

	First 2 months	Next 2 months
New acquisitions	300	450
Total CDs bought	1,000	500
CDs rented	450	1,500
Total Revenue generated (INR)	6,750	22,500
Fixed Cost	30,000	30,000
CDs cost	2,000	1,000
Net Profit	(25,250)	(8,500)

Name of society	Standard	Customer Acquired	Total marketing cost	CDs rented/customer	Revenue generated	Profit/Loss
HEWO	Medium	100	1000	1.1	1430	430
Rail residency	Low	95	1000	1	1235	235
Water works	Low	95	1000	1	1235	235
Olive	High	25	1000	2	650	-350
Suntech	High	25	1000	2	650	-350

結論：MRDBMS/SQL 的好處

- ◆ 資料處理很穩健
- ◆ 會使用的人多，資源多易學
- ◆ 歷史悠久，長年效能優化
- ◆ 可使用 join 等功能，功能性強

結論：MRDBMS/SQL 的缺點

- ◆ 即便長年下來 SQL 衍伸出很多垂直分割、讀寫分離等等技術，但遇到某些狀況還是不適用 SQL 處理..
 - ◆ e.g. 社群網路中的留言、按讚等資料

TO SQL OR NOSQL,
THAT IS THE QUESTION

NoSQL

(Not only SQL)

非關係型、分散式、不提供 ACID 的
資料庫設計模式

Types of NoSQL

key-value

Amazon
DynamoDB (Beta)

ORACLE
BERKELEY DB 11g



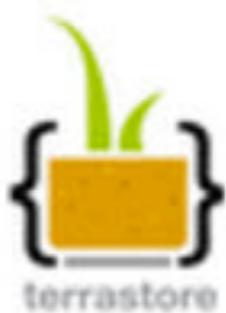
graph



column



document

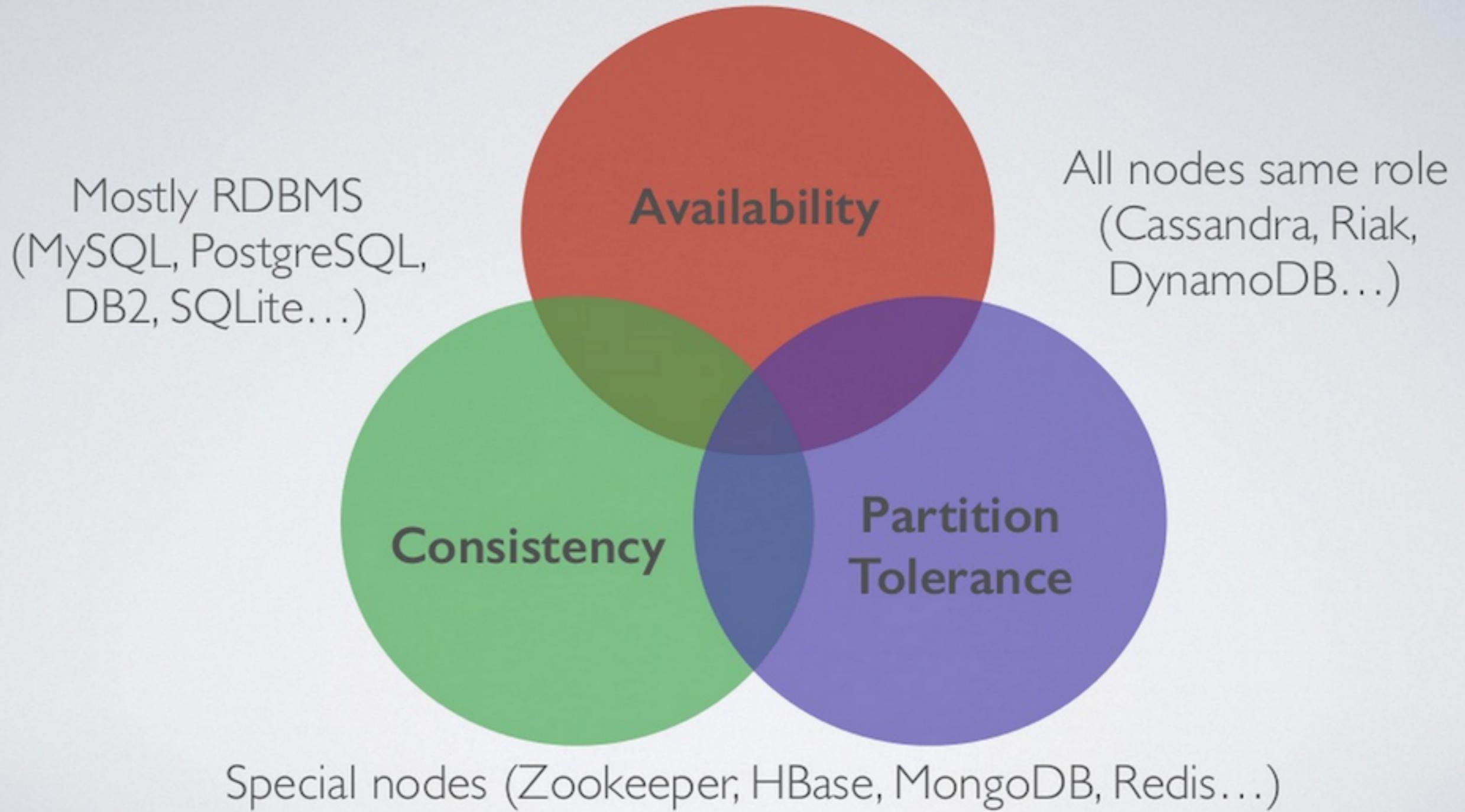


NoSQL那麼多種，要用哪一個？

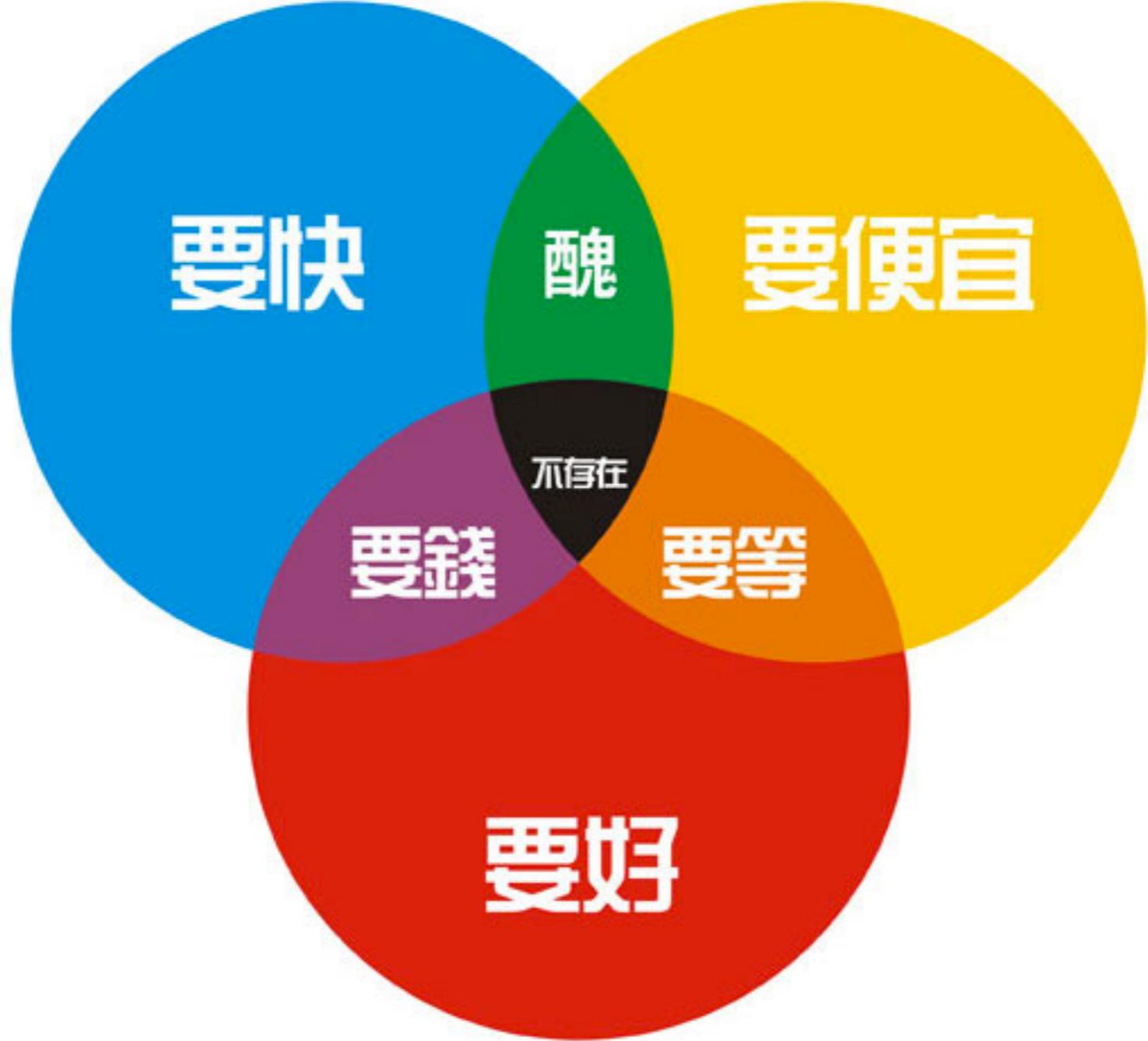
- ◆ Brewer's CAP Theorem
 - ◆ Consistency 一致性
 - ◆ Availability 可用性
 - ◆ Partition tolerance 分區容忍
 - ◆ 根據定理，分布式系統只能滿足三項中的兩項而不可能滿足全部三項。理解CAP理論的最簡單方式是想像兩個節點分處分區兩側。允許至少一個節點更新狀態會導致數據不一致，即喪失了C性質。如果為了保證數據一致性，將分區一側的節點設置為不可用，那麼又喪失了A性質。除非兩個節點可以互通信，才能既保證C又保證A，這又會導致喪失P性質。

Visual Guide to NoSQL Systems

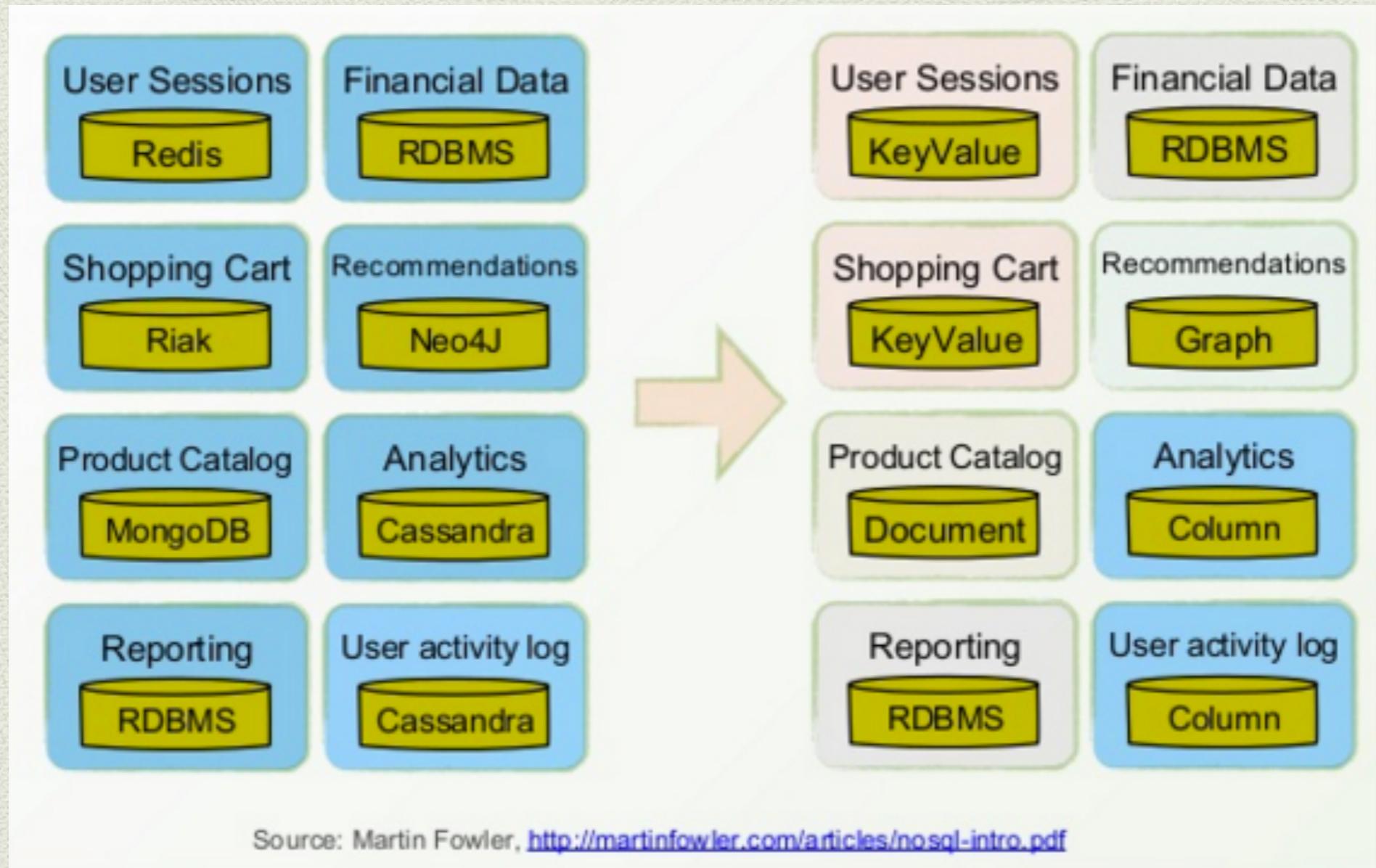




讓人想到這張經典圖...

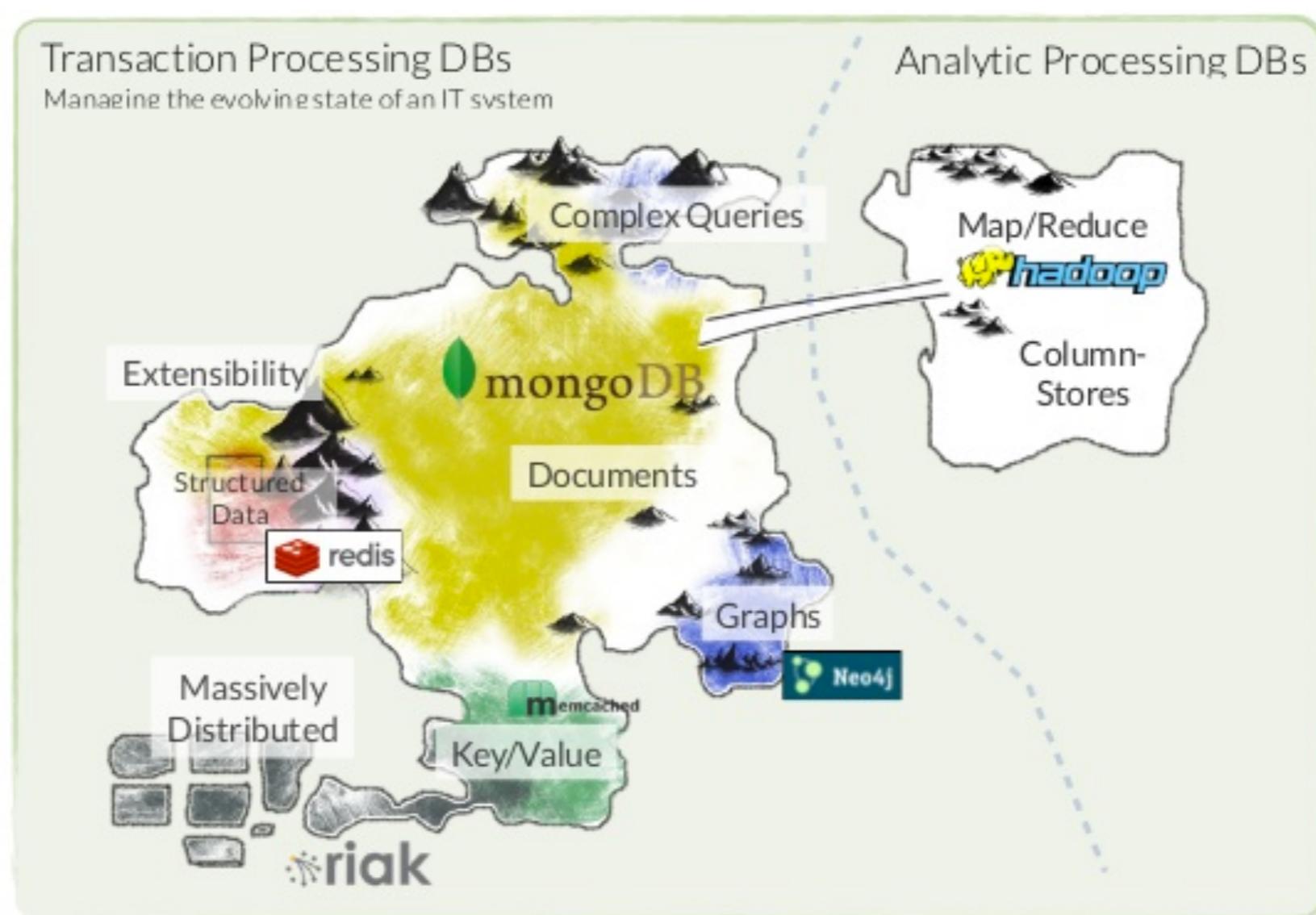


不同型態的 DB 適用於不同的情境



NoSQL 世界地圖

NoSQL Map



MongoDB

- ◆ MongoDB 是 10gen 這家公司開發的一個 NoSQL Database，屬於 Document-Oriented Database 這一類型，希望能夠結合 Relational Database 與 Key / Value Database 雙方的優點，很適合用在 Web 應用程式、Internet 架構的環境底下。



MongoDB Terminologies

RDBMS	MongoDB
Database	Database
Table	Collection
Record/Row	Document
Column	Field
Primary Key	<code>_id</code>

MongoDB Document Format

- 寫法看起來就跟 JSON 一樣。_id 欄位就是 Primary Key，其他都是一般的欄位。

```
{  
  "_id": "10280",  
  "city": "NEW YORK",  
  "state": "NY",  
  "pop": 5574,  
  "loc": [-74.016323, 40.710537]  
}
```

MongoDB Query Examples

- ◆ 查詢 OA 出版社出版的所有書籍資料 db.books.find({pubId: "OA"});
- ◆ 查詢所有定價超過 50 元美金的書籍資料 db.books.find({listPrice: {\$gte: 50}});
- ◆ 查詢 OA 出版社出版的所有書籍中，書名出現過 Java 的書籍資料
db.books.find({pubId: "OA", title: /.*\Java.* / g});
- ◆ 查詢 OA 或 PH 出版社出版的所有書籍資料
db.books.find({\$or: [{pubId: "OA"}, {pubId: "PH"}]}); 或
db.books.find({pubId: {\$in: ["OA", "PH"]}});
- ◆ Query 預設會傳回所有欄位，如果要查詢 OA 出版社出版的所有書籍，只需要書名與定價資料
db.books.find({pubId: "OA"}, {title:1, listPrice: 1, _id: 0});

MongoDB Query Examples

- ◆ 如果要查詢所有書籍，不想要出版日期 db.books.find(null, {releaseDate: 0});
- ◆ 搭配 sort 與 limit、skip 等方法，可以控制輸出的順序 — 如果要查詢所有書籍，只需要書名與定價資料，根據定價由小排到大或由大排到小 db.books.find(null, {title:1, listPrice: 1, _id: 0}).sort({listPrice: 1});
- ◆ 如果要查詢所有書籍，根據定價由小排到大，只要前兩筆 db.books.find().sort({listPrice: 1}).limit(2);
- ◆ 透過 count 之類的方法，可以輕易達到彙總的功能 — 如果要查詢 OA 出版社出版的所有書籍總數
db.books.count({pubId: "OA"});

其他應該要知道的 NoSQL

- ◆ **Redis:** in-memory 的 key-value database，因此常常被用在需要快取 (Cache) 一些資料的場合，加快資料存取速度。
- ◆ **Riak:** Basho公司基於Amazon的Dynamo理論推廣開發的 key-value distributed database. Based on Erlang, 一開始就支持分布式、容錯應用程序。沒有主節點的概念，因此在處理故障方面有更好的彈性。
- ◆ **Cassandra:** 一套開源分散式NoSQL資料庫系統。它最初由Facebook開發，用於儲存收件箱等簡單格式資料，集Google BigTable的資料模型與Amazon Dynamo的完全分散式架構於一身。Facebook於2008將Cassandra 開源，此後，由於Cassandra良好的可延伸性和效能，被許多知名網站所採用，成為了一種流行的分散式結構化資料儲存方案。

如何在 Web Program 之中使用 DB?

◆ ORM (Object Relational Mapping)

- ◆ 理想上，在寫 web program 的時候，我們希望 focus on web 操作的邏輯與資料流，不想要去管 DB 如何儲存、以及如何使用等語法細節。
- ◆ ORM 概念上像是創建了一個可在編程語言裡使用的「虛擬物件資料庫」，讓DB的存取與查詢都可以用比較高階、物件化的方式進行
- ◆ 只不過，現實上，DB 的種類太多，在 Web Programming 技術不斷推陳出新的情況下，似乎沒有人有那個力氣 / 時間打造出一個通用、大家都認可、願意使用的 high-level ORM

如何在 Web Program 之中使用 DB?

- ◆ 最低階 / 直接的做法 — e.g. npm install mysql

```
var mysql      = require('mysql');
var connection = mysql.createConnection({
  host      : 'localhost',
  user      : 'me',
  password  : 'secret',
  database  : 'my_db'
});

connection.connect();

connection.query('SELECT 1 + 1 AS solution', function (error, results, fields) {
  if (error) throw error;
  console.log('The solution is: ', results[0].solution);
});

connection.end();
```

如何在 Web Program 之中使用 DB?

- ◆ 或者是使用一些 query libraries (e.g. Knex, Sequelize)
- ◆ npm install knex

```
var knex = require('knex')({  
  client: 'mysql',  
  connection: {  
    host : '127.0.0.1',  
    user : 'your_database_user',  
    password : 'your_database_password',  
    database : 'myapp_test'  
  }  
});
```

knex.js Examples

```
var pg = require('knex')({client: 'pg'});
knex('table').insert({a: 'b'}).returning('*').toString();
// "insert into "table" ("a") values ('b')"
```

```
pg('table').insert({a: 'b'}).returning('*').toString();
// "insert into "table" ("a") values ('b') returning *"
```

```
knex.avg('sum_column1').from(function() {
  this.sum('column1 as sum_column1').from('t1').groupBy('column1').as('t1')
}).as('ignored_alias')
```

```
knex('users')
.where((builder) =>
  builderwhereIn('id', [1, 11, 15]).whereNotIn('id', [17, 19])
)
.andWhere(function() {
  this.where('id', '>', 10)
})
```

In-Class Hackathon 06/06

- ◆ Individual Project
 - ◆ i.e. 評分是 by individual, 但可以跟其他人(e.g. 期末 project 的 partners) project 有互動、關聯，但自己還是要能 demonstrate 你做的是什麼
 - ◆ 不限定一定要包含前端、後端、資料庫
 - ◆ 可以使用任何套件
 - ◆ 會有業師助教來課堂給大家問問題
- ◆ 一定要寫 Readme.md, 提供 project 功能/目的說明、如何使用、你貢獻的部分... 等
- ◆ As usual, 請在 06/12 (Tue) 11:59pm 以前將 Github Repo POST 在 FB/Ric's Web Programming 課程版，範例請見：<https://www.facebook.com/groups/NTURicWebProg/permalink/430937773920815/>