# Overview
# Relational Database
# Hands-On SQL

BMI701 Introduction of Biomedical Informatics
Lab Session 1

Wei-Hung Weng

September 7, 2016

HMS DBMI — MGH LCS

# Logistics

- Wei-Hung Weng, MD
- Research fellow in MGH Lab of Comp Sci
- NLP, medical ontology, machine learning, databases
- Every Thursday 12:30pm, about 1 hour
- Flexible, not mandatory
- Website
  - Github repository (codes and slides)
  - ckbjimmy@gmail.com

# Schedule

| Lab | Date | Topic |
|---|---|---|
| 1 | Sep 08 | Overview / relational database / SQL |
| 2 | Sep 15 | Database design /<br>Database normalization / R basic |
| 3 | Sep 22 | Ontology / database using R |
| 4 | Sep 26 | **(Monday before & after Adam's class)**<br>NoSQL / Project discussion |
| | Oct 06 | **No lab (Presentation week)** |
| | Oct 13 | **No lab (Columbus day)** |
| 5 | Oct 20 | NLP using R or py / regular expression |

# Schedule

| Lab | Date | Topic |
|---|---|---|
| 6 | Oct 27 | MetaMap / cTAKES demonstration |
| 7 | Nov 03 | Data visualization using ggplot2 |
| 8 | Nov 10 | Bioinformatics tools / GWAS |
| | Nov 17 | **No lab (AMIA)** |
| 9 | Nov 24 | ML using Weka, R or python<br>(or unsupervised learning / feature engineering) |
| 10 | Dec 01 | Deep learning on image classification (py) |
| | Dec 08 | **No lab (Final week)** |

# Survey

- Background? (clinicians / scientists and engineers)
- CS courses? (introduction / data structure / algorithm)
- Math courses?
- Programming language?
  - C/C++, Java, Python, R, Matlab/Octave, Perl, ...

# Today

- Database
- Relational database
- Simple SQL syntax
- MySQL

# Why Database?

- I believe that you've learned a lot about database from Adam's class
- RAM is expensive and small → HD is cheaper and large
- You can't put all data into RAM (unless you are $$$$)
- But it's also not easy to manage data on your HD
- Therefore we need database solution
  - $\mathbf{O(1)} < \mathbf{O(log(n))} < \mathbf{O(n)} < O(nlog(n)) < O(n^m) < O(2^n)$

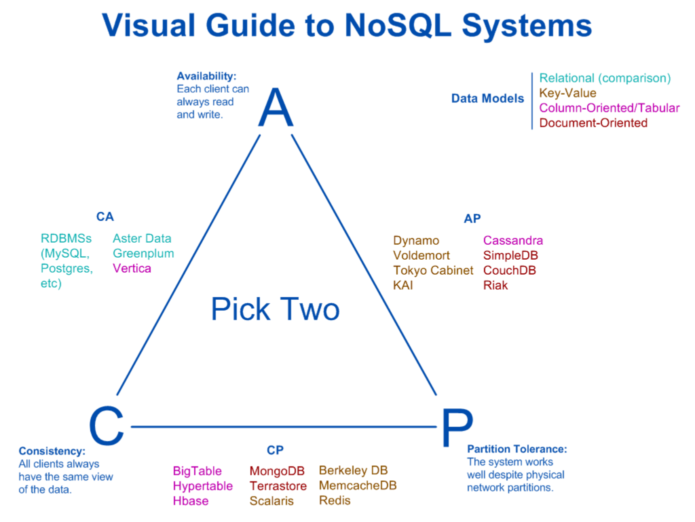
# Database CAP

- Consistency
  - All nodes should see the same data at the same time (transaction)
- Availability/Speed
  - Every request receives a response about whether the transaction is passed or failed
- Partition tolerance
  - Able to continue operation even if there are partially loss/failure of the system
- Eric Brewer 2000
- You may only pick up two of three

# SQL

- Structured Query Language (higher C and A)
    - RDB
    - Can't deal with extremely large data
- NoSQL for big data (higher P)
- Big data?
    - Google/facebook/twitter-scale (10TB to 10PB)
    - AWS: No need to use NoSQL if your user $<$ 10M
- So we will talk about NoSQL later

# Database Selection



## Visual Guide to NoSQL Systems

**Availability:**
Each client can
always read
and write.

**Data Models**

- Relational (comparison)
- Key-Value
- Column-Oriented/Tabular
- Document-Oriented

**CA**

RDBMSs (MySQL, Postgres, etc)

Aster Data
Greenplum
Vertica

**AP**

Dynamo
Voldemort
Tokyo Cabinet
KAI

Cassandra
SimpleDB
CouchDB
Riak

**CP**

BigTable
Hypertable
Hbase

MongoDB
Terrastore
Scalaris

Berkeley DB
MemcacheDB
Redis

**Consistency:**
All clients always
have the same view
of the data.

**Partition Tolerance:**
The system works
well despite physical
network partitions.

Courtesy by digbigdata.com

# Popular RDBMS

- MS SQL Server: you will probably use this in your lab/company
- Oracle: awesome if you are $$$$
- **MySQL** / MariaDB
  - Open-sourced, fast, easy to scale-up, community, BUT no JSON
- **PostgreSQL**
  - Open-sourced, fast enough, can scale-up, can use JSON, array, ...

# Relational Database

- ACID
  - Atomicity (all or none / commit or rollback)
  - Consistency (transaction between two accounts)
  - Isolation (locking)
  - Durability (won't rollback once the transaction is done)
- PK, FK, index
- ER diagram
- Normalization/denormalization (next lab)

# Relational Database

- A table has fixed columns
- A column has a datatype
  - Numeric, character, datetime, boolean, geometric, text, JSON, array (postgresql), ...
- Rows can grow (python tuple → list)

# Primary Key (PK)

- Unique identifier in the table
- MUST be unique
- CAN'T be NULL
- Index is created

# Foreign Key (FK)

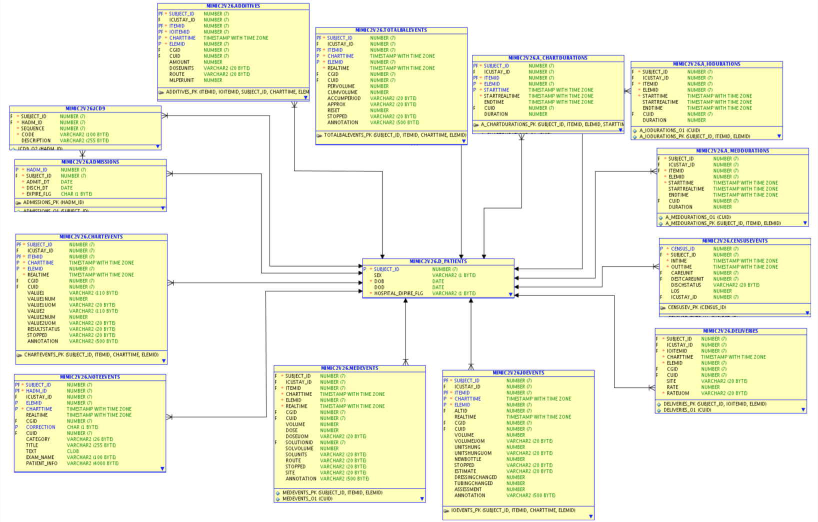
- For referential integrity
- The value MUST exist in other tables
- May be NULL or duplicated
- No index is created

# Index

- Fast but takes space
- Slow insert/update but fast read/delete
- B+ tree

| Task | No index | Index |
|---|---|---|
| Create | O(n) | O(n) |
| Read | O(n) | O(log(n)) |
| Update | O(1) | O(log(n)) |
| Delete | O(n) | O(log(n)) |

# Entity-Relationship Diagram (ER Diagram)



MIMIC2V26.ADDITIVES

| | | |
|---|---|---|
| PF * | SUBJECT_ID | NUMBER (7) |
| F | ICUSTAY_ID | NUMBER (7) |
| PF * | ITEMID | NUMBER (7) |
| PF * | IOITEMID | NUMBER (7) |
| P * | CHARTTIME | TIMESTAMP WITH TIME ZONE |
| P * | ELEMID | NUMBER (7) |
| F | CGID | NUMBER (7) |
| F | CUID | NUMBER (7) |
| | AMOUNT | NUMBER |
| | DOSEUNITS | VARCHAR2 (20 BYTE) |
| | ROUTE | VARCHAR2 (20 BYTE) |
| | MLPERUNIT | NUMBER |

ADDITIVES_PK (ITEMID, IOITEMID, SUBJECT_ID, CHARTTIME, ELEM

MIMIC2V26.TOTALBALEVENTS

| | | |
|---|---|---|
| PF * | SUBJECT_ID | NUMBER (7) |
| F | ICUSTAY_ID | NUMBER (7) |
| PF * | ITEMID | NUMBER (7) |
| P * | CHARTTIME | TIMESTAMP WITH TIME ZONE |
| P * | ELEMID | NUMBER (7) |
| * | REALTIME | TIMESTAMP WITH TIME ZONE |
| F | CGID | NUMBER (7) |
| F | CUID | NUMBER (7) |
| | PERVOLUME | NUMBER |
| | CUMVOLUME | NUMBER |
| | ACCUMPERIOD | VARCHAR2 (20 BYTE) |
| | APPROX | VARCHAR2 (20 BYTE) |
| | RESET | NUMBER |
| | STOPPED | VARCHAR2 (20 BYTE) |
| | ANNOTATION | VARCHAR2 (500 BYTE) |

TOTALBALEVENTS_PK (SUBJECT_ID, ITEMID, CHARTTIME, ELEMID)

MIMIC2V26.A_CHARTDURATIONS

| | | |
|---|---|---|
| PF * | SUBJECT_ID | NUMBER (7) |
| F | ICUSTAY_ID | NUMBER (7) |
| PF * | ITEMID | NUMBER (7) |
| P * | ELEMID | NUMBER (7) |
| P * | STARTTIME | TIMESTAMP WITH TIME ZONE |
| * | STARTREALTIME | TIMESTAMP WITH TIME ZONE |
| | ENDTIME | TIMESTAMP WITH TIME ZONE |
| F | CUID | NUMBER (7) |
| | DURATION | NUMBER |

A_CHARTDURATIONS_PK (SUBJECT_ID, ITEMID, ELEMID, STARTTIM

MIMIC2V26.A_IODURATIONS

| | | |
|---|---|---|
| F * | SUBJECT_ID | NUMBER (7) |
| F | ICUSTAY_ID | NUMBER (7) |
| F * | ITEMID | NUMBER (7) |
| * | ELEMID | NUMBER (7) |
| * | STARTTIME | TIMESTAMP WITH TIME ZONE |
| | STARTREALTIME | TIMESTAMP WITH TIME ZONE |
| | ENDTIME | TIMESTAMP WITH TIME ZONE |
| F | CUID | NUMBER (7) |
| | DURATION | NUMBER |

A_IODURATIONS_O1 (CUID)
A_IODURATIONS_PK (SUBJECT_ID, ITEMID, ELEMID)

MIMIC2V26.ICD9

| | | |
|---|---|---|
| F * | SUBJECT_ID | NUMBER (7) |
| F * | HADM_ID | NUMBER (7) |
| * | SEQUENCE | NUMBER (7) |
| * | CODE | VARCHAR2 (100 BYTE) |
| | DESCRIPTION | VARCHAR2 (255 BYTE) |

ICD9_O2 (HADM_ID)

MIMIC2V26.ADMISSIONS

| | | |
|---|---|---|
| P * | HADM_ID | NUMBER (7) |
| F * | SUBJECT_ID | NUMBER (7) |
| * | ADMIT_DT | DATE |
| * | DISCH_DT | DATE |
| * | EXPIRE_FLG | CHAR (1 BYTE) |

ADMISSIONS_PK (HADM_ID)
ADMISSIONS_O1 (SUBJECT_ID)

MIMIC2V26.A_MEDDURATIONS

| | | |
|---|---|---|
| F * | SUBJECT_ID | NUMBER (7) |
| F | ICUSTAY_ID | NUMBER (7) |
| F * | ITEMID | NUMBER (7) |
| * | ELEMID | NUMBER (7) |
| * | STARTTIME | TIMESTAMP WITH TIME ZONE |
| | STARTREALTIME | TIMESTAMP WITH TIME ZONE |
| | ENDTIME | TIMESTAMP WITH TIME ZONE |
| F | CUID | NUMBER (7) |
| | DURATION | NUMBER |

A_MEDDURATIONS_O1 (CUID)
A_MEDDURATIONS_PK (SUBJECT_ID, ITEMID, ELEMID)

MIMIC2V26.CHARTEVENTS

| | | |
|---|---|---|
| PF * | SUBJECT_ID | NUMBER (7) |
| F | ICUSTAY_ID | NUMBER (7) |
| PF * | ITEMID | NUMBER (7) |
| P * | CHARTTIME | TIMESTAMP WITH TIME ZONE |
| P * | ELEMID | NUMBER (7) |
| * | REALTIME | TIMESTAMP WITH TIME ZONE |
| F | CGID | NUMBER (7) |
| F | CUID | NUMBER (7) |
| | VALUE1 | VARCHAR2 (110 BYTE) |
| | VALUE1NUM | NUMBER |
| | VALUE1UOM | VARCHAR2 (20 BYTE) |
| | VALUE2 | VARCHAR2 (110 BYTE) |
| | VALUE2NUM | NUMBER |
| | VALUE2UOM | VARCHAR2 (20 BYTE) |
| | RESULTSTATUS | VARCHAR2 (20 BYTE) |
| | STOPPED | VARCHAR2 (20 BYTE) |
| | ANNOTATION | VARCHAR2 (500 BYTE) |

CHARTEVENTS_PK (SUBJECT_ID, ITEMID, CHARTTIME, ELEMID)

MIMIC2V26.D_PATIENTS

| | | |
|---|---|---|
| P * | SUBJECT_ID | NUMBER (7) |
| | SEX | VARCHAR2 (1 BYTE) |
| * | DOB | DATE |
| | DOD | DATE |
| * | HOSPITAL_EXPIRE_FLG | VARCHAR2 (1 BYTE) |

MIMIC2V26.CENSUSEVENTS

| | | |
|---|---|---|
| P * | CENSUS_ID | NUMBER (7) |
| F * | SUBJECT_ID | NUMBER (7) |
| * | INTIME | TIMESTAMP WITH TIME ZONE |
| * | OUTTIME | TIMESTAMP WITH TIME ZONE |
| F | CAREUNIT | NUMBER (7) |
| F | DESTCAREUNIT | NUMBER (7) |
| | DISCHSTATUS | VARCHAR2 (20 BYTE) |
| | LOS | NUMBER |
| F | ICUSTAY_ID | NUMBER (7) |

CENSUSEV_PK (CENSUS_ID)

MIMIC2V26.DELIVERIES

| | | |
|---|---|---|
| F * | SUBJECT_ID | NUMBER (7) |
| F | ICUSTAY_ID | NUMBER (7) |
| F * | IOITEMID | NUMBER (7) |
| * | CHARTTIME | TIMESTAMP WITH TIME ZONE |
| * | ELEMID | NUMBER (7) |
| F | CGID | NUMBER (7) |
| F | CUID | NUMBER (7) |
| | SITE | VARCHAR2 (20 BYTE) |
| | RATE | NUMBER |
| * | RATEUOM | VARCHAR2 (20 BYTE) |

DELIVERIES_PK (SUBJECT_ID, IOITEMID, ELEMID)
DELIVERIES_O1 (CUID)

MIMIC2V26.NOTEEVENTS

| | | |
|---|---|---|
| PF * | SUBJECT_ID | NUMBER (7) |
| PF * | HADM_ID | NUMBER (7) |
| F | ICUSTAY_ID | NUMBER (7) |
| P | ELEMID | NUMBER (7) |
| P * | CHARTTIME | TIMESTAMP WITH TIME ZONE |
| | REALTIME | TIMESTAMP WITH TIME ZONE |
| F | CGID | NUMBER (7) |
| P | CORRECTION | CHAR (1 BYTE) |
| F | CUID | NUMBER (7) |
| | CATEGORY | VARCHAR2 (26 BYTE) |
| | TITLE | VARCHAR2 (255 BYTE) |
| | TEXT | CLOB |
| | EXAM_NAME | VARCHAR2 (100 BYTE) |
| | PATIENT_INFO | VARCHAR2 (4000 BYTE) |

MIMIC2V26.MEDEVENTS

| | | |
|---|---|---|
| F * | SUBJECT_ID | NUMBER (7) |
| F | ICUSTAY_ID | NUMBER (7) |
| F * | ITEMID | NUMBER (7) |
| * | CHARTTIME | TIMESTAMP WITH TIME ZONE |
| * | ELEMID | NUMBER (7) |
| * | REALTIME | TIMESTAMP WITH TIME ZONE |
| F | CGID | NUMBER (7) |
| F | CUID | NUMBER (7) |
| | VOLUME | NUMBER |
| | DOSE | NUMBER |
| | DOSEUOM | VARCHAR2 (20 BYTE) |
| F | SOLUTIONID | NUMBER (7) |
| | SOLVOLUME | NUMBER |
| | SOLUNITS | VARCHAR2 (20 BYTE) |
| | ROUTE | VARCHAR2 (20 BYTE) |
| | STOPPED | VARCHAR2 (20 BYTE) |
| | SITE | VARCHAR2 (20 BYTE) |
| | ANNOTATION | VARCHAR2 (500 BYTE) |

MEDEVENTS_PK (SUBJECT_ID, ITEMID, ELEMID)
MEDEVENTS_O1 (CUID)

MIMIC2V26.IOEVENTS

| | | |
|---|---|---|
| PF * | SUBJECT_ID | NUMBER (7) |
| F | ICUSTAY_ID | NUMBER (7) |
| PF * | ITEMID | NUMBER (7) |
| P * | CHARTTIME | TIMESTAMP WITH TIME ZONE |
| P * | ELEMID | NUMBER (7) |
| F | ALTID | NUMBER (7) |
| | REALTIME | TIMESTAMP WITH TIME ZONE |
| F | CGID | NUMBER (7) |
| F | CUID | NUMBER (7) |
| | VOLUME | NUMBER |
| | VOLUMEUOM | VARCHAR2 (20 BYTE) |
| | UNITSHUNG | NUMBER |
| | UNITSHUNGUOM | VARCHAR2 (20 BYTE) |
| | NEWBOTTLE | NUMBER |
| | STOPPED | VARCHAR2 (20 BYTE) |
| | ESTIMATE | VARCHAR2 (20 BYTE) |
| | DRESSINGCHANGED | NUMBER |
| | TUBINGCHANGED | NUMBER |
| | ASSESSMENT | NUMBER |
| | ANNOTATION | VARCHAR2 (500 BYTE) |

IOEVENTS_PK (SUBJECT_ID, ITEMID, CHARTTIME, ELEMID)

MIMIC-II Database (Clifford, 2011)

# SQL's CRUD

- CRUD: create, read, update, delete
- Insert
- Select
  - **Filter**: where, distinct
  - **Aggregate**: group by
  - **Join**
  - **Subquery**
  - Partition (splitting the data into manageable size)
- Update
- Delete

# Hands-on SQL

- Install MySQL Server
- Install MySQL Workbench
- Small databases
- github.com/ckbjimmy/bmi701lab/blob/master/lab01.sql

# Filter

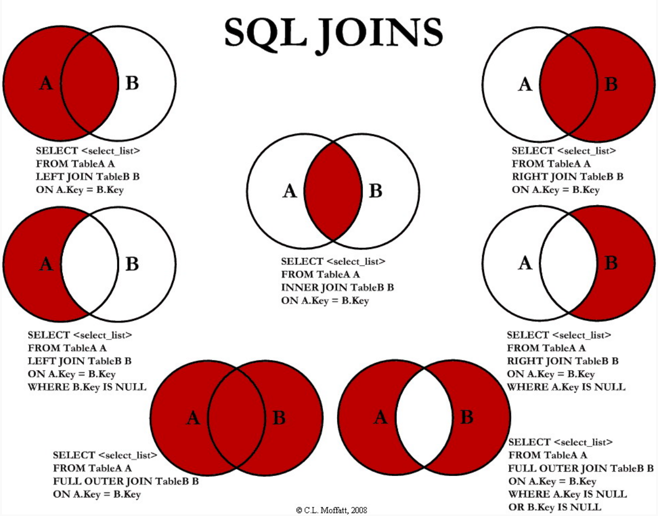
- SELECT * FROM gwas LIMIT 5
- SELECT gene FROM gwas WHERE chr_id = 22
- SELECT distinct gene, trait, p_value FROM gwas WHERE allele like ’A’ ORDER BY p_value
- SELECT gene, trait FROM gwas WHERE trait like ’%diabetes_’
- SELECT gene, trait FROM gwas WHERE trait like ’%diabetes%’
- SELECT gene, trait FROM gwas WHERE trait like ’diabetes’

- SELECT GROUP_CONCAT(DISTINCT gene SEPARATOR ’,’) FROM gwas WHERE chr_id = 3

# Aggregation

- `SELECT count(*) FROM sitka`
- `SELECT avg(size) FROM sitka`
- `SELECT max(size) FROM sitka`
- `SELECT min(size) FROM sitka`
- `SELECT stddev(size) FROM sitka`
- `SELECT tree, avg(size) FROM sitka GROUP BY tree`

# Join



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

Courtesy by codeproject.com

# Join

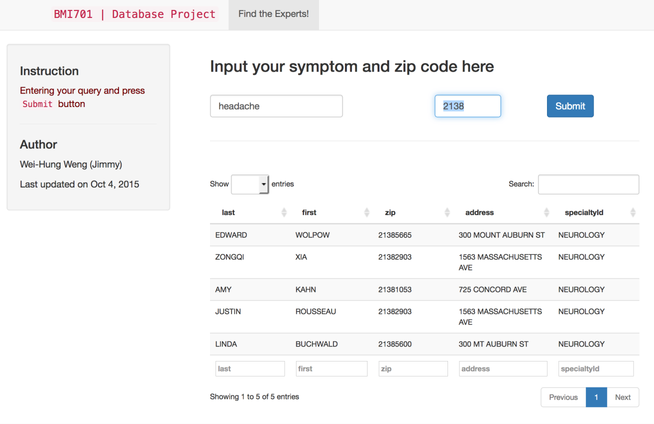
- ON col1 = col2
- Inner (default), left, right, full, cross (inner join on TRUE)
- Using PK
-
  ```
  SELECT curr_description_f.conceptid,
  curr_description_f.term,
  curr_textdefinition_f.term FROM curr_description_f
  LEFT JOIN curr_textdefinition_f ON
  curr_description_f.conceptid LIKE
  curr_textdefinition_f.conceptid WHERE
  curr_description_f.conceptid LIKE ’%777%’ LIMIT 30
  ```

# Subquery

- `SELECT * FROM sitka WHERE treat LIKE ’%ozo%’`
- `SELECT * FROM (SELECT * FROM sitka WHERE treat LIKE ’%ozo%’) oz WHERE size > 5.0`

# Demonstration

- My database project (MySQL, R, shiny, AWS)



### Instruction

Entering your query and press `Submit` button

### Author

Wei-Hung Weng (Jimmy)

Last updated on Oct 4, 2015

## Input your symptom and zip code here

headache

2138

| last | first | zip | address | specialtyId |
|---|---|---|---|---|
| EDWARD | WOLPOW | 21385665 | 300 MOUNT AUBURN ST | NEUROLOGY |
| ZONGQI | XIA | 21382903 | 1563 MASSACHUSETTS AVE | NEUROLOGY |
| AMY | KAHN | 21381053 | 725 CONCORD AVE | NEUROLOGY |
| JUSTIN | ROUSSEAU | 21382903 | 1563 MASSACHUSETTS AVE | NEUROLOGY |
| LINDA | BUCHWALD | 21385600 | 300 MT AUBURN ST | NEUROLOGY |
| last | first | zip | address | specialtyId |

Showing 1 to 5 of 5 entries

# Preparation for Next Next Week

- Install R
- Install RStudio
- Register UTS Service
- Download SNOMED (need to wait for UTS permission)

# Take Home Message

- Every Thursday 12:30pm
- `SELECT (DISTINCT) ... FROM ... JOIN ... ON ... WHERE ... GROUP BY ... ORDER BY ... LIMIT ...`
- Install R/RStudio, register UTS, download SNOMED
- Contact
  - Github repository
  - ckbjimmy@gmail.com
  - Linkedin: Wei-Hung Weng