COMP3311 24T1 Database Systems

week 1 - 1

Outline

- **People**
- How COMP3311 will run in 24T1
- Overview of DBMS
- Data Modelling

People - Lecturer

Name: Yuekang Li（悦康 李）
Role: Course Convenor - COMP3311
Personal Website: https://thepatrickstar.github.io/

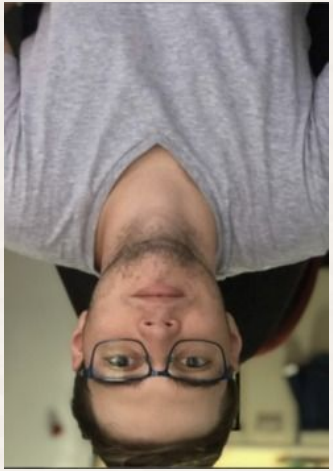
Experience: Ph.D. (2020), Bachelor (2016) in Nanyang Technological University, Singapore

Research: Software Engineering/Security, Software Testing, LLM related stuff …

What to call me? Yuekang, Dr. Li, YK …

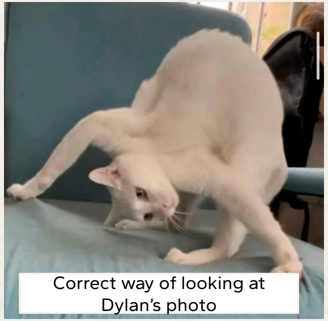
Email: cs3311@cse.unsw.edu.au

People - Course Admin



Name: Dylan Brotherston
Role: Course Admin - COMP3311

Email: cs3311@cse.unsw.edu.au



Correct way of looking at Dylan's photo

People - Tutors

- Kenneth Li
- Yun Li
- Evan Krul
- Max Xue
- Calley Chai
- Manhua Lu
- Xinzhang Chen
- Dylan Brotherston
- Ayda Valinezhad Orang
- Sarah Chen
- Edward Qian
- Justin Liu
- Abbie Worswick

https://webcms3.cse.unsw.edu.au/COMP3311/24T1/timetable

Outline

- People
- **How COMP3311 will run in 24T1**
- Overview of DBMS
- Data Modelling

How COMP3311 will run in 24T1

Course Website: https://webcms3.cse.unsw.edu.au/COMP3311/24T1/

slides, timetable, etc.

Forums: https://edstem.org/au/courses/15447/discussion/

Recordings & MyExperience: https://moodle.telt.unsw.edu.au/course/view.php?id=81670 (login required)

How COMP3311 will run in 24T1

Sources of information:

lecture notes: primary (detailed) content + summaries + examples

textbooks ... most detailed version of topics

Activities:

lectures ... primary content, work though examples, ask questions

tutorials ... work through exercises, ask questions

prac exercises ... learn the systems and skills

assignments ... practice your skills

exam ... demonstrate your knowledge/skills

How COMP3311 will run in 24T1

How are the slides organized:

- Color Codings / Icons

K Summary of Knowledge

⚠️ Attention is required

E Exercise

Make your hands dirty!

How COMP3311 will run in 24T1

Make your hands dirty!

You run you own PostgreSQL server on the host `nw-syd-vxdb2`

How to access the `vxdb2` server
from Vlab: `ssh YourZid@nw-syd-vxdb2`
from Home: `ssh YourZid@d2.cse.unsw.edu.au`
On the vxdb2 server you have your standard CSE directories
a special directory `/localstorage/YourZid/`

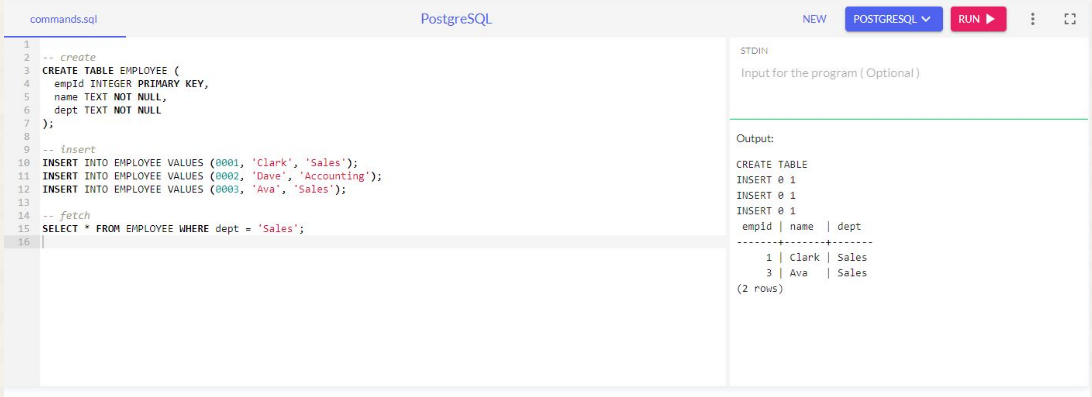
Try to get this done before Tut-1 next week.
More details can be found in Practice 1 here: ⚠
https://webcms3.cse.unsw.edu.au/COMP3311/24T1/resources/96267

How COMP3311 will run in 24T1

Make your hands dirty!

Use some online services:
https://onecompiler.com/postgresql

Some slides may contain examples that you can run on `onecompiler`



```sql

-- create
CREATE TABLE EMPLOYEE (
  empId INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  dept TEXT NOT NULL
);

-- insert
INSERT INTO EMPLOYEE VALUES (0001, 'Clark', 'Sales');
INSERT INTO EMPLOYEE VALUES (0002, 'Dave', 'Accounting');
INSERT INTO EMPLOYEE VALUES (0003, 'Ava', 'Sales');

-- fetch
SELECT * FROM EMPLOYEE WHERE dept = 'Sales';
```

Output:

```
CREATE TABLE
INSERT 0 1
INSERT 0 1
INSERT 0 1
 empid | name  | dept
-------+-------+-------
     1 | Clark | Sales
     3 | Ava   | Sales
(2 rows)
```

Outline

- People
- How COMP3311 will run in 24T1
- **Overview of DBMS**
- Data Modelling

Database Management Systems

Definition:

A Database Management System (DBMS) is a **software system** that is designed to **manage** and **organize** data in a structured manner. It allows users to create, modify, and query a database, as well as manage the security and access controls for that database.

software ...

data ...

Are you talking about Excel ?

Is Excel a DBMS? Yes? No? Why?

No. Excel is only a spreadsheet software that cannot be considered as a database because it lacks **data integrity, proper structure, table relationships, and database keys**.

Types of DBMS

- Relational Database Management System (RDBMS):
  Data Model: Organizes data into two-dimensional tables (rows and columns).
  Key Features:
  Uses SQL (Structured Query Language) for querying and managing data.
  Each table has a primary key that uniquely identifies records.
  Example: Postgresql, Mysql

- Object-Oriented Database:
  Combines relational database concepts with object-oriented principles.
  Represents data as objects (used in object-oriented programming).
  Requires less code and is easy to maintain.
  Example: MongoDB.

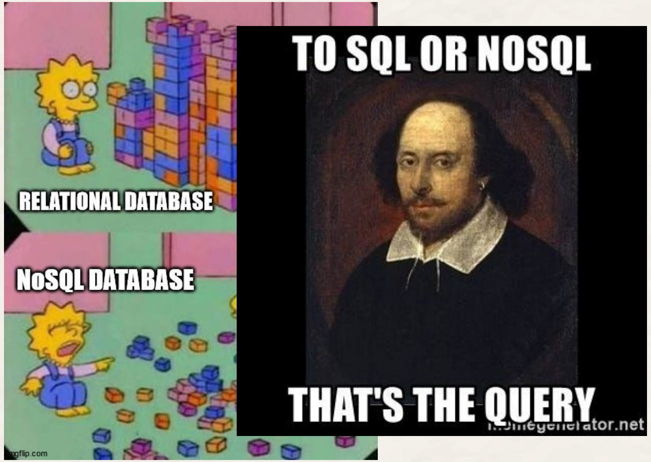
Types of DBMS

- Graph (Network) Database:
  Data elements are organized like a graph, allowing more than one parent for a child record.
  Example: Neo4J

- Others
  - KV pairs (e.g., Redis)
  - Vector Database

In this course, we will focus on Relational Database Management Systems (RDBMS) ⚠️

Why learn SQL in 2024?



- The concept of NOSQL was quite popular

- 8 Reasons for using SQL: https://blog.sqlizer.io/posts/sql-43/

- Personally:
  - SQL is normally much faster when scale grows up
  - Well-rounded
  - A unique way of describing the world

Outline

- People
- How COMP3311 will run in 24T1
- Overview of DBMS
- **Data Modelling**

Data Modelling

Aims of data modelling:

- describe what **information** is contained in the database
  (e.g., entities: students, courses, accounts, branches, patients, ...)
- ⚠️ describe **relationships** between data items
  (e.g., John is enrolled in COMP3311, Tom's account is held at Coogee)
- describe **constraints** on data
  (e.g., 7-digit IDs, students can enrol in no more than 3 courses per term)

Data modelling is a design process

- converts requirements into a data model

Data Modelling

- **Inputs** to data modelling:
  enterprise to be modelled, user requirements

- **Outputs** from data modelling:
  (semi) formal description of the database structure

- Many languages/methodologies have been developed to assist, e.g.
  - entity-relationship (ER) diagrams
  - ODL = object design language
  - UML = unified modelling language

Data Modelling

Data models are either:

- logical models ... deal with conceptual modelling of information
- physical models ... deal with physical layout of data in storage

Two main groups of logical data models:

- object-based models
  e.g. object-oriented, semantic, ...
- record based models
  e.g. relational, network, hierachical, ...

Object vs Record

Object-based data models

- treat database as a collection of entities of various kinds
- provide very flexible/natural data structuring facilities
- may also allow description of code for actions on objects

Record-based data models

- treat database as a collection of fixed-size records
- less flexible data structures than with object-based models
- closer to physical level so easier to implement efficiently

Practical Design Idea

Formal mappings exist between the different kinds of models.

A useful strategy:

- design in an object-based model (for clarity)
- then convert to a record-based model (for efficiency)

We adopt this (very common) strategy in this course.

Practical Design Idea

Consider the following while working through exercises:

- start simple ... evolve design as problem better understood
- identify objects (and their properties), then relationships
    - also don’t forget the constraints
- most designs involve kinds (classes) of people/actors
- keywords in requirements suggest data/relationships
    - (rule-of-thumb: nouns → data, verbs → relationships)
- consider all possible data, not just what is available

Exercise 1:

Imagine that we wanted a database of course outlines.

Work out requirements by looking at real course outlines.

Develop an informal data model for it by identifying:

- the data items involved (objects and their attributes)
- relationships between these data items
- constraints on the data and relationships

E

Exercise 1 (Thinking Process):

Remember, let's start from objects first, and then think about their relationships.

What's the most important object for a course outline?

Of course, course!

What are the attributes of a course?

Course: code, name, description

What about the lecturers and students of the course?

Better not, when it involves some kind of "relationship"

E

Exercise 1 (Thinking Process):

Let's make users a separate class of objects.

What are the attributes of a user?

User: zid, name, email, types

What else objects do we have?

Class: types, day, starting, ending, where

Assessment: name, description, weight, due_date

E

Exercise 1 (Thinking Process):

Let's make users a separate class of objects.

What are the attributes of a user?

User: zid, name, email, types

What else objects do we have?

Class: types, day, starting, ending, where

Assessment: name, description, weight, due_date

E

Exercise 1 (Thinking Process):

Now we have the objects:

Course: code, name, description
User: zid, name, email, types
Class: types, day, starting, ending, where
Assessment: name, description, weight, due_date

What about their relationships?

	course	user	class	assessment
course	prerequisite, equivalent	take, teach	belong	part_of
user		-	enrolled, in_charge_of	-
class			-	-
assessment				-

E

Exercise 1 (Thinking Process):

Now we have the objects and their relationships:

Course: code, name, description
User: zid, name, email, types
Class: types, day, starting, ending, where
Assessment: name, description, weight, due_date

Prerequisite: course, course
Equivalent: course, course,
Take: course, user
Teach: course, user
Belong: class, course
Part_Of: course, assessment
Enrolled: class, user
In_Charge_Of: class, user

Can relationships have attributes?

Yes. Teach: course, user, role

E

Exercise 1 (Thinking Process):

Now, this is what we have:

Course: code, name, description
User: zid, name, email, types
Class: types, day, starting, ending, where
Assessment: name, description, weight, due_date

Prerequisite: course, course
Equivalent: course, course,
Take: course, user
Teach: course, user, role
Belong: class, course
Part_Of: course, assessment
Enrolled: class, user
In_Charge_Of: class, user

Can optimize (trim) some relationships?

Yes. We need to think about the nature of the relationships.

Exercise 1 (Thinking Process):

Let’s go through the relationships one by one:

Prerequisite: course, course
Equivalent: course, course,
Take: course, user
Teach: course, user, role
Belong: class, course
Part_Of: course, assessment
Enrolled: class, user
In_Charge_Of: class, user

These two relationships are one-to-many and more importantly, they rely on the existence of a course:
Belong: class, course
Part_Of: course, assessment

We can turn them into attributes to save space!

E

Exercise 1 (Thinking Process):

We can make these changes (as optimizations):

Course: code, name, description
User: zid, name, email, types
Class: types, day, starting, ending, where, course*
Assessment: name, description, weight, due_date, course*

Prerequisite: course, course
Equivalent: course, course,
Take: course, user
Teach: course, user, role
~~Belong: class, course~~
~~Part_Of: course, assessment~~
Enrolled: class, user
In_Charge_Of: class, user

Exercise 1 (Thinking Process):

This is what we have now:

Course: code, name, description
User: zid, name, email, types
Class: types, day, starting, ending, where, course*
Assessment: name, description, weight, due_date, course*

Prerequisite: course, course
Equivalent: course, course,
Take: course, user
Teach: course, user, role
Enrolled: class, user
In_Charge_Of: class, user

Since we have the objects/entities and their relationships. Now we need to think about the constraints.

Exercise 1 (Thinking Process):

This is what we have now:

Course: code (10 chars), name (20 words), description (100 words)
User: zid (10 chars), name (80 chars), email (valid format/100 chars), types (fixed list of options)
Class: types (fixed list of options), day (date), starting (time), ending (time), where (fixed list of options), course* (a code/unique ID for a valid course in the database)
Assessment: name, description, weight (digit, >0, <100), due_date, course*

Prerequisite: course, course
Equivalent: course, course,
Take: course, user
Teach: course, user, role
Enrolled: class, user
In_Charge_Of: class, user

A user cannot take a course twice!

Exercise 2:

Consider the GMail system (or any other modern mail client)

Develop an informal data model for it by identifying:

- the data items involved (objects and their attributes)
- relationships between these data items
- constraints on the data and relationships

E

Exercise 2 (Thinking Process):

The objects:

Messages: title, date, content, ...

Users: address, name, ...

Labels: name

...

The relationships:

Sent-by: message, sender (user)

Received-by: message, receiver (user)

Belongs-to: label, message

...

E

Exercise 2 (Thinking Process):

The objects:

Messages: title (50 words), date (datetime), content (1000 words), ...

Users: address, name, ...

Labels: name

...

The relationships:

Sent-by: message, sender (user)

Received-by: message, receiver (user)

Belongs-to: label, message

...

A message must have a sender (creator)

E

Exercise 2 (Answer):

The objects:

Messages: title (50 words), date (datetime), content (1000 words), creator* (user) ...

Users: address, name, ...

Labels: name

...

The relationships:

Received-by: message, receiver (user)

Belongs-to: label, message

...

Remember, data modelling is a design process, and you can have alternative answers! ⚠

Thank you!