>>

Data Models

- Data Modelling
- Some Design Ideas
- Exercise: GMail Data Model
- Quality of Designs

∧ >>

❖ Data Modelling

Aims of data modelling:

- describe what information is contained in the database i.e. nouns
  (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) i.e. verbs
- 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

You are just designing how your data is going to be formatted, not actually implementing anything.

- converts requirements into a data model

requirements -> data model

Note: your model is the way you are choosing to store your data i.e. how it is represented/modelled. This is similar to how we make a model level in some programming patterns (a architecture level that is just responsible for holding data).

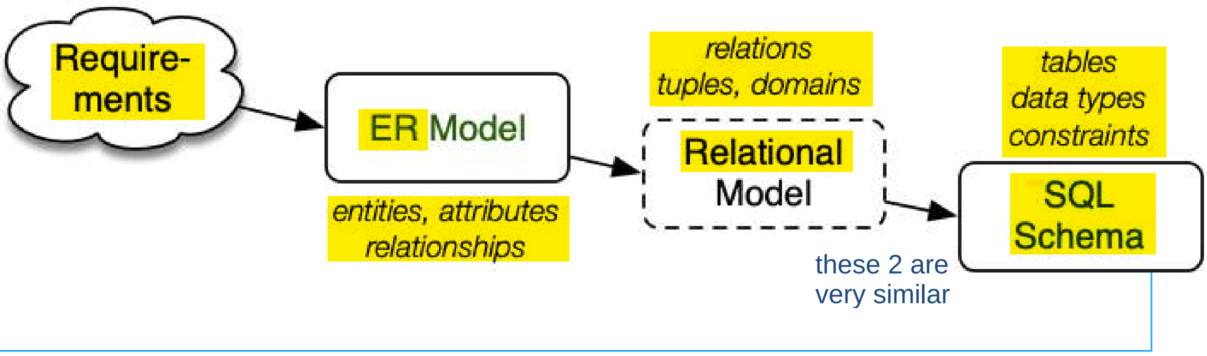
<< ∧ >>

❖ Data Modelling (cont)

Kinds of data models:

- logical: abstract, for conceptual design, e.g., ER, ODL, UML
- physical: record-based, for implementation, e.g., relational, SQL

Strategy: design using abstract model; map to physical model


Require-ments
ER Model
entities, attributes relationships
relations tuples, domains
Relational Model
tables data types constraints
SQL Schema
these 2 are very similar

<< ∧ >>

❖ Some Design Ideas

Consider the following while working through exercises:

- start simple ... evolve design as problem better understood

  evolve when there is a need to evolve e.g. you may start with something as an attribute, then realise it needs to be shared so you evolve it to its own entity.

- identify objects (and their properties), then relationships

  it is easier if you have all the entities first. Follows the above rule of simple -> detailed. If something couldn't be covered by an entity or attribute, then it is probably going to be a relationship!

- most designs involve kinds (classes) of people
- keywords in requirements suggest data/relationships

  (rule-of-thumb: nouns → data, verbs → relationships)

- don't confuse operations with relationships

  relationships store the end results / side effects of operations.

  (operation: he **buys** a book; relationship: the book **is owned** by him)

- consider all possible data, not just what is available

<< ∧ >>

❖ Exercise: GMail Data Model

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

<< ^ >>

❖ Exercise: GMail Data Model (cont)

Objects in GMail data model:

- users
  - gmail-address, name, password, ...
- messages
  - timestamp, sender*, title, content, ...
- tags
  - owner, name, colour parent*
- settings
  - name, value, user*

Relationships in GMail data model:

- recipients
  - user - message
- sent
  - user - message
- tag-hierarchy
  - child-tag - parent-tag
- settings
  - user - setting

Constraints in GMail data model:

- gmail-address values are unique
- users must have a password (strong?)

```
every message has a sender

every message has a non-empty title and content

values for each setting are valid for that setting
```

<< ∧

❖ Quality of Designs

you are going to have to do trade-offs

There is no single "best" design for a given application.

Most important aspects of a design (data model):

- correctness (satisfies requirements accurately)
- completeness (all reqs covered, all assumptions explicit)
- consistency (no contradictory statements)

Potential inadequacies in a design:

- omits information that needs to be included breaks completeness
- contains redundant information (⇒ inconsistency) breaks consistency
- leads to an inefficient implementation could have been designed better
- violates syntactic or semantic rules of data model breaks modelling language

Produced: 12 Sep 2020