Lecture 1. Relational Data Model.

Def. Database is an organized collection of interrelated data that models some aspect of the real world.

Remark. Databases are core component of most computer application. Databases are ubiquitous.

Example. Database Modelling.

i) What do we want to store?

Create a dat base that models a dipital music store.

ii) What data do we want to store for those things?

Artists, Albums and Tracks.

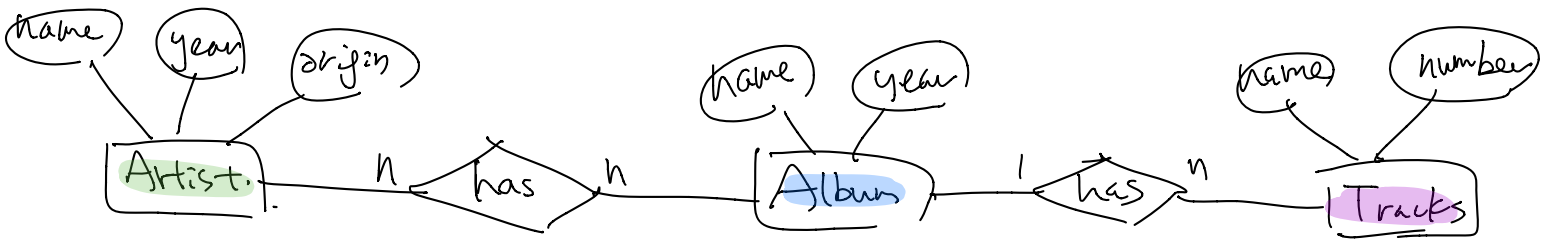
iii) How do they related together?

- Information about Artists
- What Albums those artist released?
- The Tracks on those Albums.

* Entity - Relationship Diagram

: Typical way to model a database.

What entities are inside our model? And how they are related?



Artists have names, year they started, and country of origin.

An Album has one or more Artists

Albums have release years, and names.

An Album has multiple Tracks

Tracks have a name and a number.

* Implementation: CSV files.

- Use a seperate file per entity. ex) artist.csv. album.csv. tracks.csv.
- Application has to parse the file every time read/update occurs.
- Problems:
  i) Performance issues.
  ii) Data integrity.
    ex) How do we ensure values are the same?
    What if I update the name of the artist only on artist.csv?
    ex) What if someone writes bad code? Overwriting values?
    What if someone stores not integers, but strings for track numbers?
    ex) How do we store multiple artists for a single album?
  iii) Implementation.
    ex) How do you find a particular record?
    Total search is very pricey.
    ex) What if we want to make a new app with this database?
    Implement new code per app, per language, per schema?
    ex) What if multiple threads access the database?
    locks are pricey, Shards are hard to maintain.
  iv) Durability.
    ex) What happens if the machine crashes?
    How to make rollbacks for sequential operations?
    ex) What if we want to replicate on multiple machines?
    Essential for high availability.

Remark. Database Management Systems solves all these problems!
Application programmer doesn't need to know the implementations of storing.

Def. A database management system (DBMS) is a software that allows applications to store and analyze information in a database.

A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases.

Remark. Database ≠ DBMS. ex) I'm managing MySQL ~~Database~~ DBMS.

* Types of DBMS : divided by target workloads

i) Online Transaction Processing (OLTP) ↝ Most common.
Fast operations which only read/update small amounts per operations.
ex) Front-end DBMS that ingests new data from the "outside world."

ii) Online Analytical Processing (OLAP) ↝ Typically hard to build.
Complex queries that read a lot of data to compute aggregations.
ex) Finding new information from the collected data by OLTP systems.

iii) Hybrid Transaction + Analytical Processing.
OLAP + OLTP together on a same database instance.

* Types of DBMS : divide by data models.

Def. Data Model is how database is going to represent data to store.

i) Relational : Most common (This course's focus).
ii) (Loosely) NoSQL : Key/Value, Graph, Document, Column-Family
iii) Arrays/Matrix : Common in machine learning/AI models.
iv) Hierarchical } → Obsolete. Rare.
v) Network

Remark. Relational database is the most flexible, as it can model everything else.
But it is not used because of efficiency issues.

* Relational Model.

Def. A relation is an unordered set which contains the relationship of attributes that represent entities.
↳ Implement as table.

Def. A tuple is a sequence of attribute values in the relation.
"An instance of a relationship".
↳ Implement as row or record

* Integrity Constraints (which relational model provides us)

: Ensures data are correct across multiple relations.

i) Primary keys

A relation's primary key uniquely identifies a single tuple.

ex) Artist (id, name, year, country)
↳ Inject synthetic id for uniqueness.

Nearly every DBMSs support auto-generation.
ex) AUTO-INCREMENT in MySQL. SEQUENCE in SQL.
↳ MySQL always does this. (Names of their own).

ii) Foreign keys

A foreign key specifies that an attribute from one relation has to map to a tuple in another relation.

ex) ArtistAlbum (artist_id, album_id)
↳ points to Artist ↳ points to Album.

Not all DBMSs enforce this, but SQL standart follows this.

Prevents invalid/nonexistent artist_id or album_id to be inserted in ArtistAlbum.

* Queries

Remark. The relational model is independent of any query language implementation, as it is a mathematical concept.

But SQL is the de facto standard.

i) Imperative programming language.
Have to tell exactly how to complete the answer.
ex) for line in file :
    if parsed(line) is ....

ii) Declarative programming language.
Only tell the answer. DBMSs have to figure out how to compute it.