COSC210 Database Mangement Sytems

University of New England

Lecture 2 - Databases System Concepts and Architecture

Dr. Mitchell Welch / Dr. Edmund Sadgrove

Reading

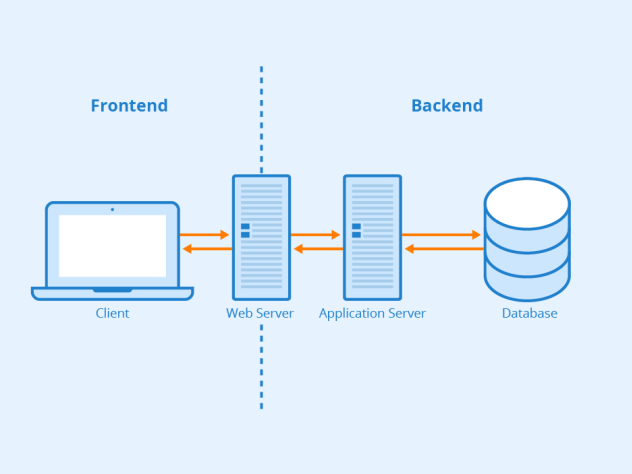
- Chapter 2 from ***Fundamentals of Database Systems***

Summary

- Data Models, Schemas and Instances
- The Three Schema Architecture
- Languages and Interfaces
- A Quick Look At Client/Server Architectures

Data Models, Schemas and Instances

- **Database-driven applications** distribute their functionality between:
  - The **Client Module**:
    - **Users workstation** - personal device.
      - User interface (PC).
      - Mobile Application.
  - The **Server Module**:
    - **remote machine** - virtualised system.
      - Access.
      - Data storage.
      - Search functions.
      - Other operations.



Data Models, Schemas and Instances

- **Data Abstraction**:
  - The suppression (hiding) of implementation details.
  - Achieved through the *Data Model*.
- **Data Model**:
  - High level.
    - Conceptual data models.
    - User understanding.
  - Low level.
    - Physical data models.
    - Data Storage.
- The most prominent model in use, is the ***Relational Data Model*** (next lecture)

Data Models, Schemas and Instances

* **Conceptual Data Models** * **Entity-Relationship model (ER)**: * Entities represent real-world objects. * Linked through relationships. * **Physical Data Models** * **DBMS developers**: * Defines interaction with: * Operating system. * Hardware.

Data Models, Schemas and Instances

- Important to distinguish between the *description* and *actual* database.
  - **Database Schema**:
    - Description of DB (unchanging).
    - Specified during design.
    - Can be depicted as a schema diagram.
  - **Database State**:
    - Snapshot of live DB (changing).
    - Can be empty or populated.

The Three Schema Architecture

- The **Database Schema** is often described using the Three-Schema Architecture.
  - **Internal Schema**:
    - Physical data model.
  - **Conceptual Schema** :
    - Representational data model.
    - Hides physical storage structure.
  - **External Schemas**:
    - Representational data model.
    - User views.
    - Hides the rest of the DB.

The Three Schema Architecture

- The three-schema architecture can be used to explain *Data Independence*.
- This occurs at two levels:
  - **Logical Data Independence**:
    - Allows **conceptual schema** changes without **client applications** changes.
    - Changes: data-types, relationships, constraints.
  - **Physical Data Independence**:
    - Allows **internal schema** changess without **conceptual schema** changes.

* Changes: data storage, operating system, hardware.

Languages and Interfaces

- Three languages are used to define and interact with the Database Schema:
  - **Data Definition Language (DDL)**:
    - Defines internal and conceptual schema.
    - SQL: `CREATE, ALTER, DROP`

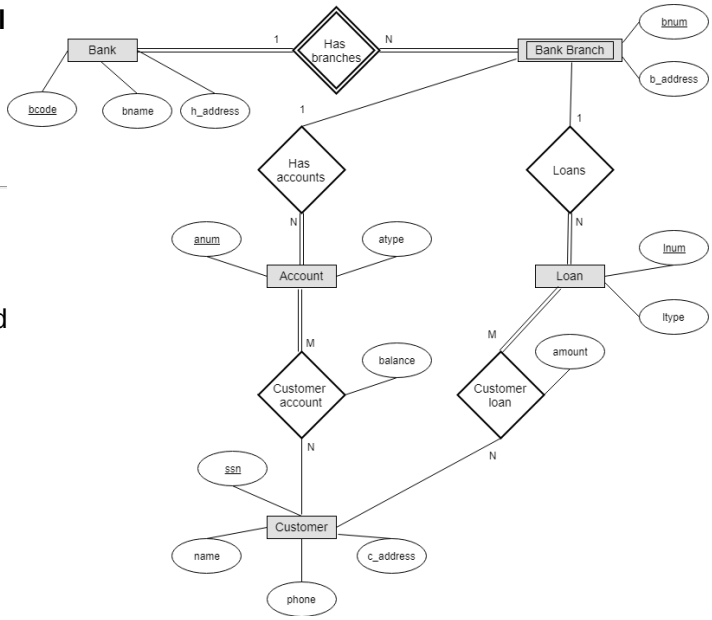


Figure 9.7
Mapping the EER categories (union types) in Figure 8.8 to relations.

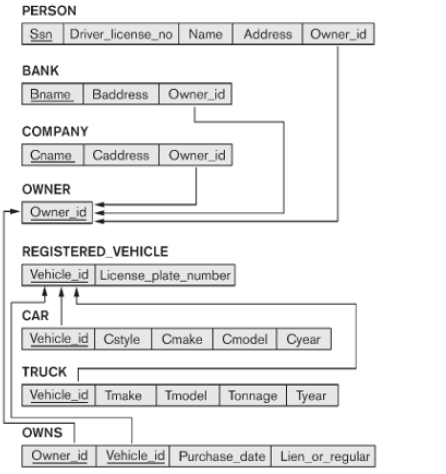
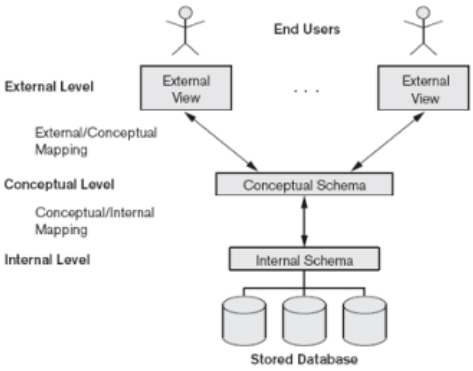
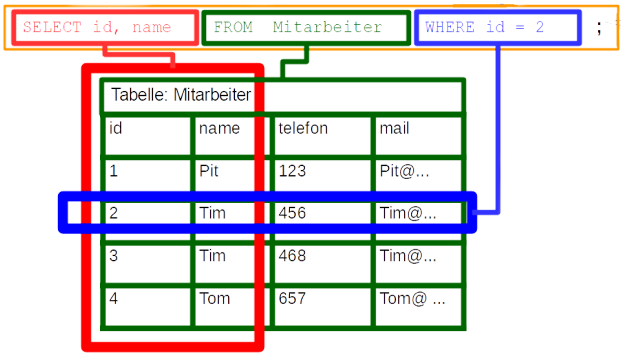


Figure 2.2
The three-schema architecture.



- **Data Manipulation Language (DML)**:
  - Performs operations on data.
  - SQL: `SELECT, INSERT, UPDATE`
- **View Definition Language (VDL)**:
  - Defines external schema.
  - SQL: `CREATE VIEW xyz as SELECT....`

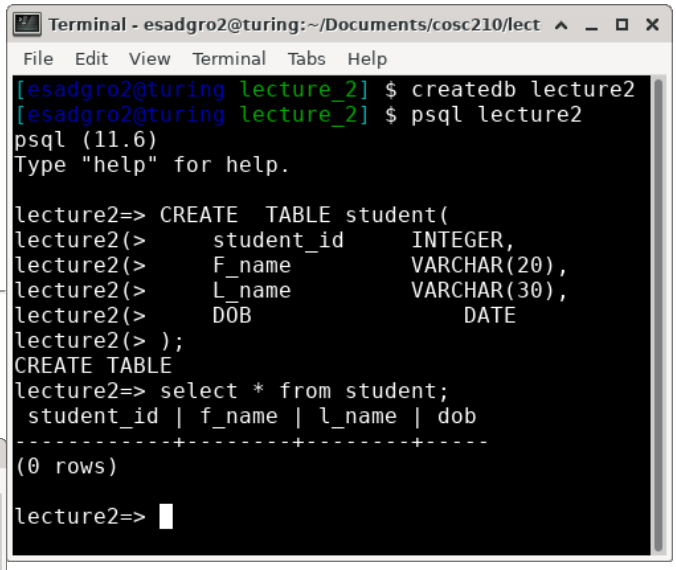


* Structured Query language (SQL) provides facilities for all three types.

Languages and Interfaces

- Table creation (DDL):

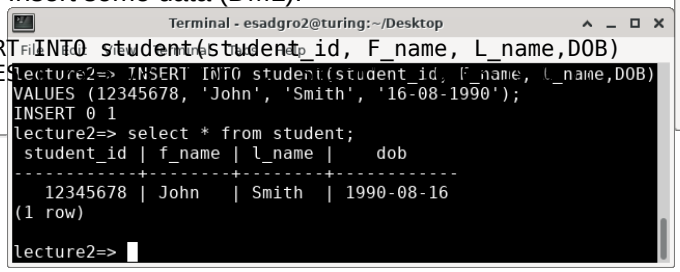
```
CREATE  TABLE student(
    student_id  INTEGER,
    F_name      VARCHAR(20),
    L_name      VARCHAR(30),
    DOB             DATE
);
```



Languages and Interfaces

- Insert some data (DML):

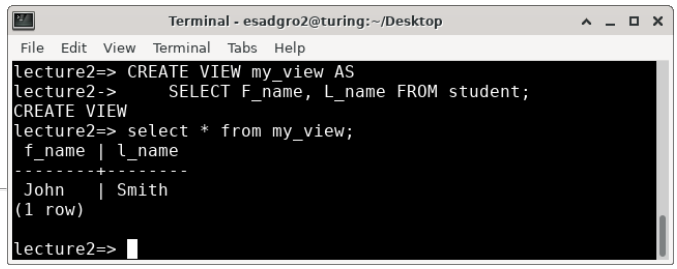
```
INSERT INTO student(student_id, F_name, L_name,DOB)
VALUES (12345678, 'John', 'Smith', '16-08-1990');
```



Languages and Interfaces

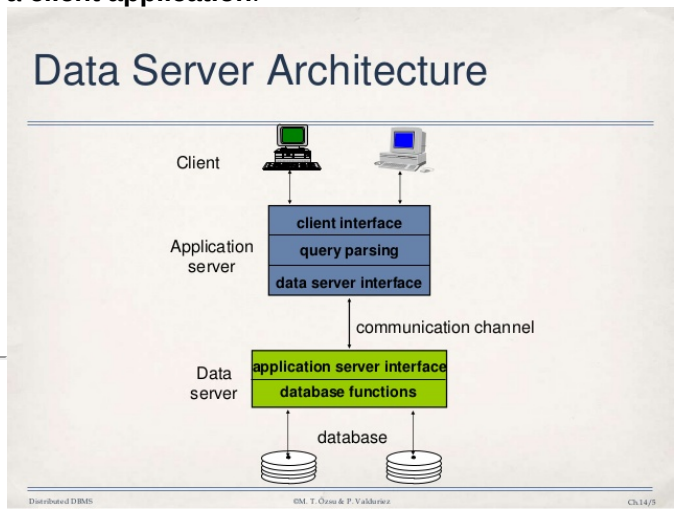
- Make a View (VDL):

```
CREATE VIEW my_view AS
    SELECT F_name, L_name FROM student;
```



Languages and Interfaces

- The data within a **DBMS** will typically be **accessed** via a **client application**.
- This application will require an **interface**.
- Some examples:
  - Menu-based systems/Web-clients
  - Form-based interfaces
  - GUIs
  - Natural Language Interfaces
  - Speech I/O



The DBMS Environment

- Interface environments include:

- **Interactive Query interface**.
  - Executes queries as entered.
  - Typically a console/terminal.
- **Application program**.
  - Queries are grouped and sent as batch transactions.
  - Typically graphical user interfaces (GUI).

The DBMS Environment

* Compiling involves **Query Optimisation**. * Operations **re-ordered**. * **Redundant** operations removed. * **Runtime Processor** executes the query. * Meta-data and actual data are stored in secondary storage. * **Managed** by operating system.

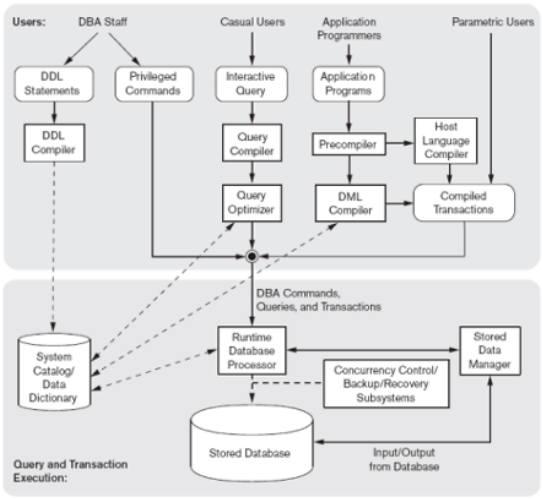


Figure 2.3
Component modules of a DBMS and their interactions.

* Provides buffer management (caching) to reduce disk access.

The DBMS Environment

- Most **DBMSs** typically provide a suit of **utilities**, including:
  - Tranfers.
  - Uploading.
  - Backups & restores.
- Data **upload** utilities:
  - In **PostgreSQL** this is achieved using the **COPY statement**:

```
COPY my_table FROM '/path/to/csv/my_file.csv' DELIMITER ',' CSV;
```

The DBMS Environment

- Database **backup** and **restore** utilities:
  - The **`pg_dump`** utility provides this functionality in PostgreSQL:

```
pg_dump dbname > outfile
```

- Restore is achieved using the **`psql`** interactive client:

```
psql dbname < infile
```

- Postgres actually has a range of backup/restore utilities

The DBMS Environment

- Most DBMSs will provide **utilities** for managing data storage and file layout.
- PostgreSQL has a range of utilities:
  - Files and directories.
  - Page Layout.
  - Index types.

The DBMS Environment

- Most DBMSs will provide a utilities for **monitoring the performance**.
- This becomes important as usage scales up.
- PostgreSQL has a range of utilities:
  - Statistics collecting.
  - Progress reporting.
  - Viewing locks.
  - Dynamic tracing.

Quick Look: Client/Server Centralized Architectures

* Centralized Architecture: * Mainframe.

* Multiple displays.

Quick Look: Specialized Server Architectures

* Phyisical two-tier:

* Basic client and server architectures.

Logical Three-Tier Client/Server Architectures

* Logical three-tier: * With mid-tier. * (a) for engineers. * (b) for package vendors.

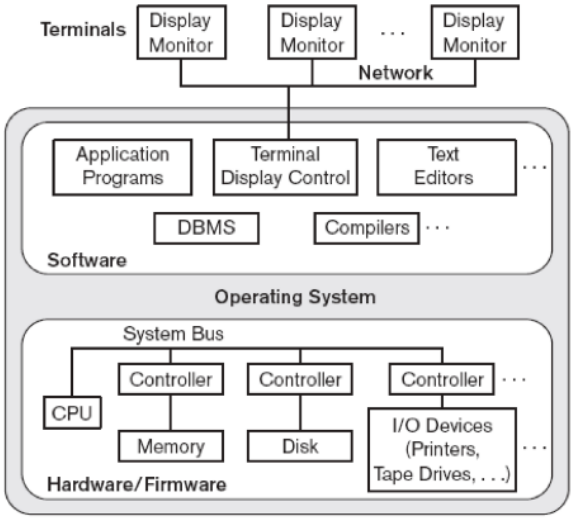


Figure 2.4
A physical centralized architecture.

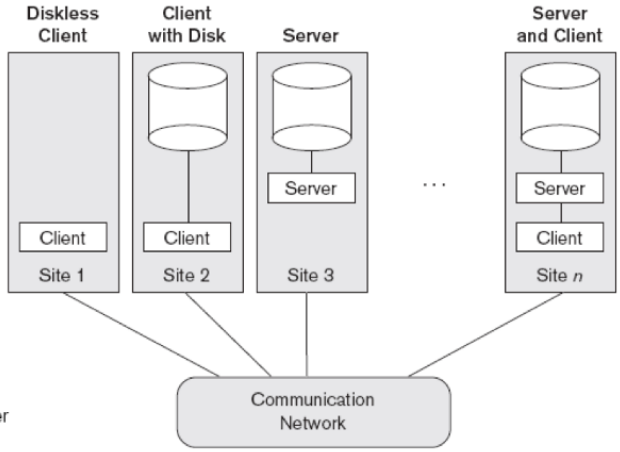


Figure 2.6
Physical two-tier client/server architecture.

Summary

- Data Models, Schemas and Instances
- The Three Schema Architecture
- Languages and Interfaces
- A Quick Look At Client/Server Architectures

Questions?

Next Lecture

- The Relational Data Model and Relational Database Constraints

---

Reading

- Chapter 2 from ***Fundamentals of Database Systems***
- Chapter 5 from ***Fundamentals of Database Systems*** for next lecture.

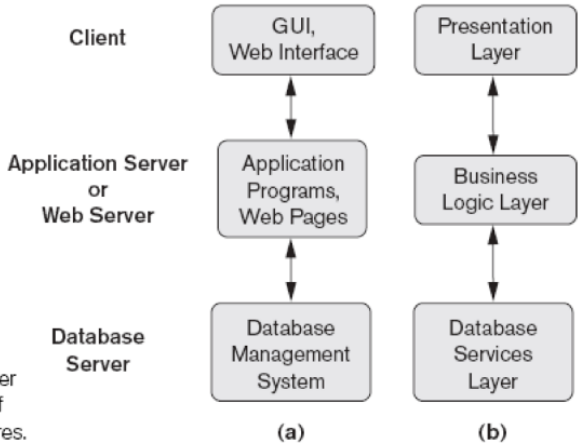


Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.
