Week 2

Data Storage and Databases

William Cooper
w.cooper@herts.ac.uk

Plan for today

What are databases?

Relational vs Non-relational

Keys and constraints

Schema

What is a database?

This is an introduction to databases and their access through Structured Query Language (SQL), which you will encounter along this module.

- A database is a collection of related data.
- A database management system (DBMS) is a piece of software enabling access and control of this data.

Duties of a DBMS

What is a DBMS responsible for?

`https://www.menti.com/aljcpboqtqmy`

Duties of a DBMS

What is a DBMS responsible for?

1. Defining rules to validate and manipulate data.
2. Interacting with databases, applications, and end users.
3. Retrieving, storing, and analysing data.
4. Updating data.

Functions of a DBMS

What does a DBMS do?

1. Efficient storage.

Efficient storage

- Accessing data on a disk takes a long time (relatively).
- Operational speeds: CPU $>>$ Memory $>>$ Disk.
- Indexing and correct variable sizing improves access speeds.
- Parallel disks to allow scaling.

Functions of a DBMS

What does a DBMS do?

1. Efficient storage.
2. Provide a logical view of the data.

Provide a logical view of the data

- Translating between logical and physical view of data (abstraction).
- Logical representation: tables, fields, etc.
- Physical representation: bytes on disks, index structure, etc.

Functions of a DBMS

What does a DBMS do?

1. Efficient storage.
2. Provide a logical view of the data.
3. Query processing.

Query processing

A DBMS always has some type of query language. This includes Structured Query Language (SQL).

- Adding new records.
- Amending existing records, i.e. modifying or deleting.
- Retrieving data by given criteria.

Functions of a DBMS

What does a DBMS do?

1. Efficient storage.
2. Provide a logical view of the data.
3. Query processing.
4. Transaction management.

Transaction management

- A transaction is a series of operations treated as **one** logical operation.
- Transactions are ‘all-or-nothing’.

Example: Transferring money between two bank accounts. Think about what could happen if this is interrupted mid transaction.

Relational databases

- Data is stored in different tables, with rows (*tuples*) and columns (*attributes*).
- Tuples describe entities or relationships between entities.
- Attributes must be uniquely named.
- All entries within an attribute must be of the same domain, i.e. same type of data.
- Each tuple should be distinct, identical tuples are allowed but **bad** for data science.

Relational databases

- Data is stored in different tables, with rows (*tuples*) and columns (*attributes*).
- Tuples describe entities or relationships between entities.
- Attributes must be uniquely named.
- All entries within an attribute must be of the same domain, i.e. same type of data.
- Each tuple should be distinct, identical tuples are allowed but **bad** for data science.

`https://www.menti.com/aljcpboqtqmy`

Non-relational databases

- Often similar in appearance to relational databases.
- Document stores, key-values pairs are examples of NoSQL.

Keys

- Keys are attributes that can be used to **uniquely** identify a tuple.
- Note that multiple keys can identify the same tuple (one/many to one, not one to many).

Keys

- Keys are attributes that can be used to **uniquely** identify a tuple.
- Note that multiple keys can identify the same tuple (one/many to one, not one to many).
- A *candidate* key is a minimal (often singular) collection of attributes that can create a key.

Keys

- Keys are attributes that can be used to **uniquely** identify a tuple.
- Note that multiple keys can identify the same tuple (one/many to one, not one to many).
- A *candidate* key is a minimal (often singular) collection of attributes that can create a key.
- A *primary* key is a candidate key which is used to arrange data such that any tuple can be rapidly retrieved.

Keys

- Keys are attributes that can be used to **uniquely** identify a tuple.
- Note that multiple keys can identify the same tuple (one/many to one, not one to many).
- A *candidate* key is a minimal (often singular) collection of attributes that can create a key.
- A *primary* key is a candidate key which is used to arrange data such that any tuple can be rapidly retrieved.
- A *foreign* key is a key that originates from a separate table.

Constraints

- Depends on your exact case but varies by attribute.

Constraints

- Depends on your exact case but varies by attribute.
- Acts like a `python` *assert* command, ensures a data entry is valid before committing a transaction.

Constraints

- Depends on your exact case but varies by attribute.
- Acts like a `python` *assert* command, ensures a data entry is valid before committing a transaction.
- The foreign key constraint ensures any entries match the origin table primary key.
- The uniqueness constraint means there can not be any duplicate value in that attribute.
- Not NULL enforces the fact that no entry within an attribute can have a missing value.

Relations

- *Relations* are tables within a database.
- The schema of a relation consists of the name of the relation followed by the names of the attributes.
- The schema of a database explains how each relation is connected, and is often shown as a spider diagram.

Relations

Example:

1. Student(**nr**, name, address, email)
2. Staff(**nr**, name, office, email, phone)
3. Department(**name**, school, building)

Relations

Example:

1. Student(**nr**, name, address, email, *supervisor*)
2. Staff(**nr**, name, office, email, phone)
3. Department(**name**, school, building)

Relations

Example:

1. Student(**nr**, name, address, email, *supervisor*)
2. Staff(**nr**, name, office, email, phone)
3. Department(**name**, school, building)
4. Subject(**student, department**)