Intro

DBMS:

Purposes of DB

Two main uses of databases:

- **online transaction processing**: large numer of users use the database, each user retrieving small amounts of data and performing small updates.
- **data analytics**: processing of data to draw conclusions, infer rules or decision procedures.

SQL DDL

```
create table department
  (dept_name  char(20),
   building   char(15),
   budget     numeric(12, 2));
```

SQL DDL supports integrity constraints

SQL DML

sql is declarative. Input several tables, output always a single table. Example:

single table input:

```
select instructor.name
from instructor
where instroctur.dept_name = 'History'
```

two tables as input:

```
select instructor.ID, department.dept_name
from instructor, department
where
  instructor.dept_name = department.dept_name and
  department.budget > 95000;
```

Steps to DB Design

Computer representation of some real world or imaginary complex system involves mainly two tasks:

- modelling the data aspect of the system (data requirements)
- modelling the behavior aspect of the system (functional requirements)

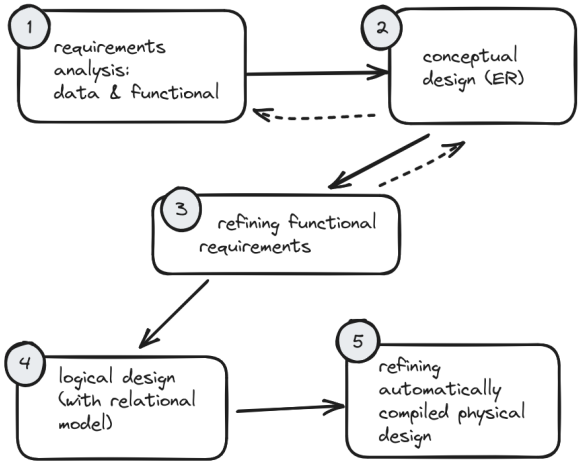
Modelling the data aspect falls under the domain of DBs design. Primary objects and their interrelationships must be determined according to the needs of the users of the system. Modelling behavior is a software design task. These tasks are interrelated and can reinforce/influence each other.

Steps of DB design:

1. **requirements analysis**: precise specification of client requirements (what data needs to be modelled) by consulting with domain experts and prospective clients.
2. **conceptual design**: a data model is chosen and the requirements are translated to the concepts of the data model (usually ER).

   the resulting schema is reviewed to confirm that all data requirements are satisfied and that there are no conflicts. Redundant features can be removed too.

   basically all necessary attributes to be captured are determined in this step. alternative to ER an automatic generation of tables can be achieved via a method called **normalization**.
3. **specification of functional requirements**: A precise conceptual schema facilitates determining the behavioral requirements. These in turn can reinforce and alter the data schema.
4. **logical-design phase**. Higher-level conceptual schema is mapped to the implementation data model of the database system (usually relational).
5. **physical-design phase**. The logical schema is automatically compiled to a physical implementation by the DBMS but the designer can manually refine and fine-tune the physical design.



DBMS architecture

The general architecture of a DBMS can be broadly regarded as consisting of **three** main components:

- **querry processor**
- **storage manager**
- **disk storage**

Querry processor

The querry processor realizes an easiy access to data via a querry language: users don't have to specify *how* to access the data but only *what* data to access. The particular algorithm/plan needed to access the data specified by the querry is automatically compiled by the querry processor. Querry processors try to optimize these algorithms as much as possible.

1. **DDL interpreter**
2. **DML compiler**: (query optimization)
3. **querry evaluation engine**: executes low-level instrocution generated by the DML compiler.

Storage Manager

DB are very large and cannot be stored entirely in the main memory. Also since persistent storage of data is always wanted, DB are stored on secomdary memory, usually disks.

Movement of data from secondary memory is much slower than movement from main memory to CPU registers. Therefore the DB tries to structure the data in a way that movement from secondary memory to main memory is minimized. This can be fine-tuned by the DB designer.

Storage Manager is responsible with the interaction with the file manager of the underlying OS. The raw data is ultimately stored on the disk using the file system of the OS. The storage manager translates the DML statements into low-level file-system commands.

Components of the storage manager:

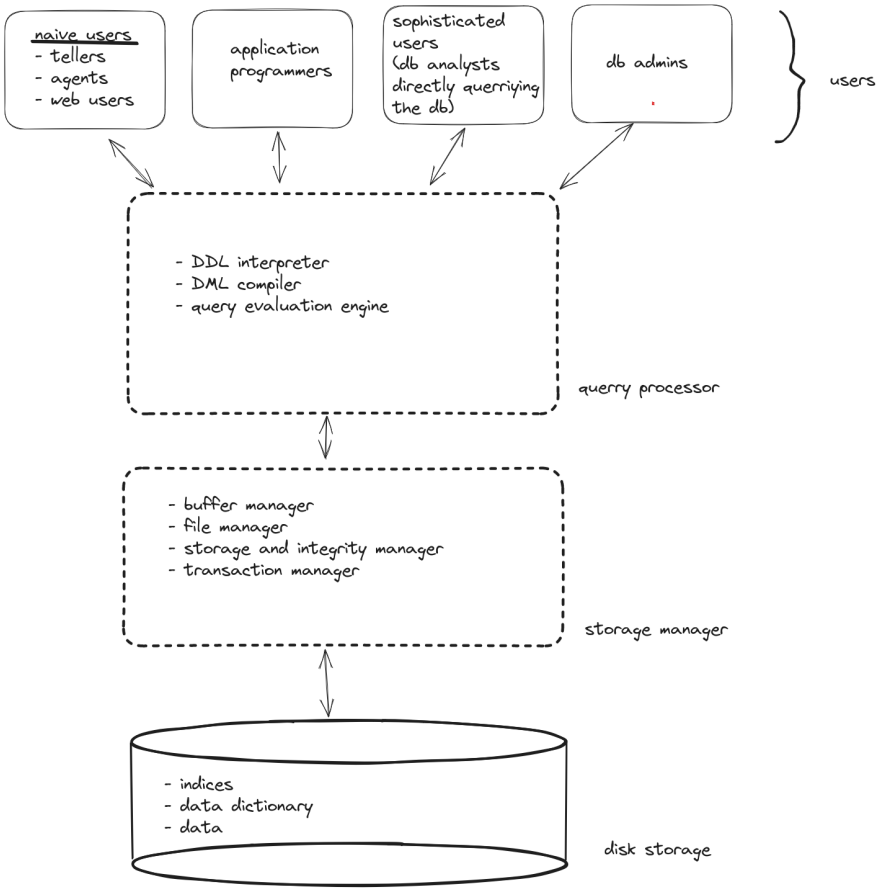
1. **authorization and integrity manager**
2. **transaction manager**: Realizes concurrent and parallel access to data as well as the grouping sequences of db access querries as single atomic units (transactions). Ensures that DB state is consistent after such operations but also in case of failures. The transaction manager thus realizes

    1. **atomicity**:
    2. **consistency**
    3. **durability**

3. **file manager**: allocation of space on disk and data structures used to represent information stored on disk
4. **buffer manager**: critical part of the DBS, since it is responsible for fetching data from disk storage into main memory and what data to cache in main memory. Enables the db to handle data sized much larger than the size of main memory.

Storage manager also implements several data structures as part of the physical system implementation:

- **data files**: store the DB itself.
- **data dictionary**: metadata about the structure of the db, i.e. the schema. Used by many other components like the querry processor
- **indices**: Just like an index of a book can provide fast access to data by storing actualy physical address of data items having some particular value on some attribute or multiple attributes.


naive users
- tellers
- agents
- web users
application programmers
sophisticated users (db analysts directly querriying the db)
db admins
users
- DDL interpreter
- DML compiler
- query evaluation engine
querry processor
- buffer manager
- file manager
- storage and integrity manager
- transaction manager
storage manager
- indices
- data dictionary
- data
disk storage