# Relational Databases

This is the main database concept since the 1980s.

Most mainstream databases use this concept or enhancements of it – more modern systems, etc. tend to be built on top of this concept.

It conceptually organises data into sets of two-dimensional tables.

## Relation:

Conceptually relational databases are based on *tables*.

This system is conceptually intuitive, simple, and elegant.

It also fits quite well with the underlying technology.

The relational database concept was not the first database concept, but was so superior to other systems at the time that it quickly became dominant.

## Anatomy of a Relational Database:

*Database:* A database consists of a set of named *relations* (aka tables)

*Relation:*

- A set of *tuples* (aka rows) with identical column structure
- Each column named with an *attribute* (must be distinct) [each row must have the same number of elements, and they must be in the right order]
- Values in each column must be of uniform "type" drawn from some set (*domain*) such as integer, text, date, and so on.

Example table:

| id_number | first_name | last_name | date_of_birth | hometown |
|---|---|---|---|---|
| 112345678 | Aoife | Ahern | 1993-01-25 | Cork |
| 112467389 | Barry | Barry | 1980-06-30 | Tralee |

## Some Technicalities

There should be no duplicate entries. There can be near duplicates (e.g. same name but different id number for students) but not exact ones. All rows should be distinct. Database systems don't always enforce this.

Values must be *atomic* (simple and unstructured).

- examples: integers, decimals, text, booleans, dates
- can't have lists, sets, records, arrays as single values
- (can represent lists etc. by other means later)

Every relation has a *key*: some subset of attributes such that no two tuples in the relation can ever have the same key value. In the example this was the student number.
It couldn't be a combined first name and last name, because there could be two students with the same name.

## Schema and Instances

The precise contents of a database generally vary over time. Example: addresses change, people are added and removed, etc.

We distinguish between:

- *Database Schema*: The "structure" of the database in terms of different tables and the organisation of each table in terms of attributes and domains and the "interpretation" of these in terms of the real-world situation the DB models. This remains fixed over time.
- *Database Instance*: Any specific collection of values populating the DB that conforms to the schema. This changes as the DB is modified. It's basically a snapshot of what the DB looks like right now (all data contained).

## Why "Relational"?

Binary relations in mathematics:

- Let $Z^2$ = Z x Z be the set of pairs of integers; any subset of $Z^2$ is a *binary relation*.
- For example: L = [(x, y) : x, y element of Z such that x<=y} is the set of integer pairs where the first is no larger than the second.

The terminology is just taken from maths. There was more on this slide but the specifics apparently aren't important.

## SQL

Our "instructions" to DBMS must be expressed in notation called SQL.

*Note that multiple people can query a database from separate computers simultaneously. DB needs to be accessible to a wide number of people.*

SQL is a computer language for expressing management, manipulation and querying of relational DBs. Pronounced either as "ess-cue-ell" or "sequel".

*Good News:* It's a pretty standard DB language. Most common DBMSs support it. There are free open source packages available.

*Bad News:* Most DBMSs adopt their own "dialects". DB software is generally not completely portable between DBMSs.

We use a computer language because natural language (e.g. English) can be very imprecise and ambiguous, so it's difficult to write software to "understand" and carry out instructions reliably.

Computer languages:

- are designed to be clear, concise, and unambiguous
- are amenable to automatic interpretation
- more here—missed it

SQL incorporates several aspects:

- **Data Manipulation Language (DML)**
  - expresses management of DB
  - expresses manipulations: adding/removing/updating tuples
  - expresses queries to interrogate DB
- **Data Definition Language (DDL)**
  - **missed it**

SQL Example -----------

*Query:*

**SELECT** first_name, last_name

**FROM** students

**WHERE** points >= 500;

*Result:*

- the result is a table (unnamed by default)
- includes specified columns and rows only
- the original table is left completely untouched

*Note: the third line (WHERE…) is a condition evaluated for each row. When it is true, the (specified bit of the) row is returned.*

SQL is very specific:

- keywords must be spelled correctly
- elements must be ordered correctly
- symbols need to be used correctly
- /*…*/ denotes (within symbols) a comment. Computer will ignore these.

## Using SQL

Typically SQL commands are embedded in other programs (as opposed to the operator/client writing the commands).