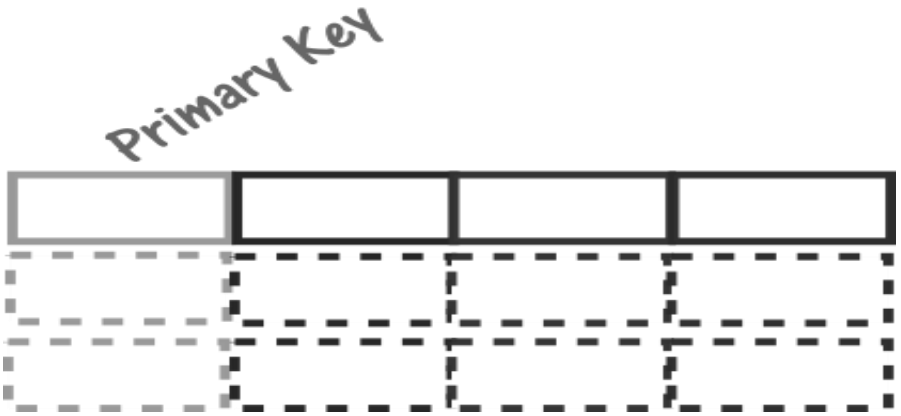
Build a Base simulate a relational database

Relational database (DB)

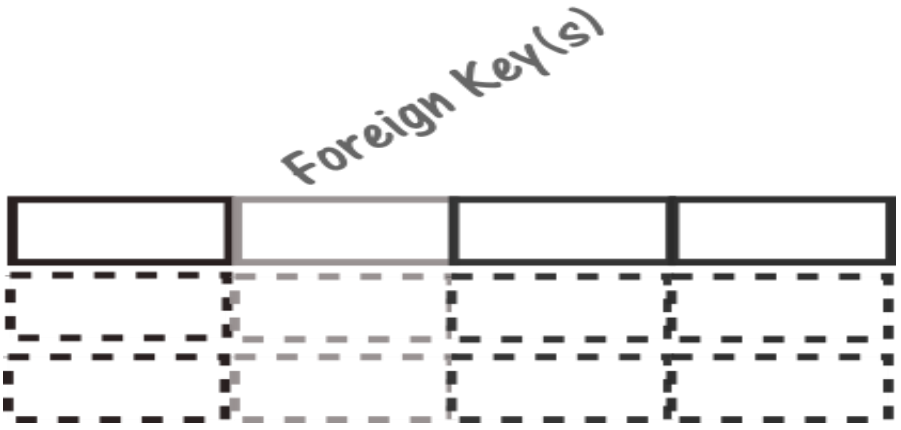
A relational database (DB) is an organized way to store data. Easy to store and easy to get again. And most important it is "easy" to combine data with other data.

A DB consist of tables. Each table is zero to many lines of data, each line conforming to the same specification.



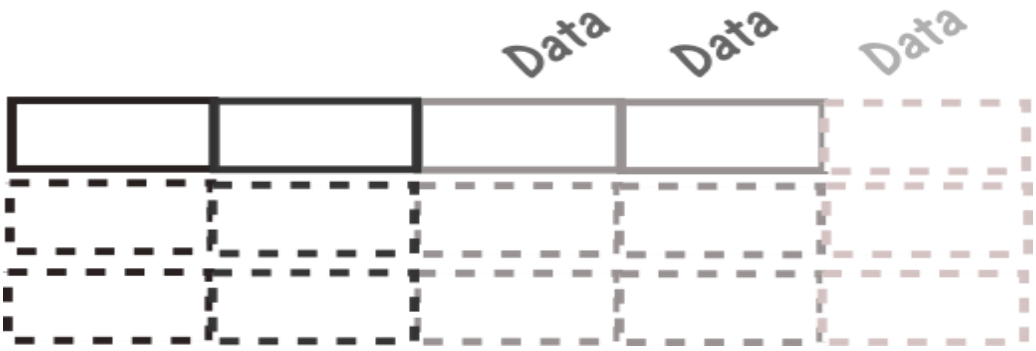
The first position is the Primary Key, which is a unique identification of the lines in that table. The Danish CPR-number (social security number) is a good example. Each tables lines about a person begins with the persons unique CPR-number.

The next positions can be Foreign Key(s), i.e. Primary Keys from other tables. Foreign Keys are not necessarily unique in the table, where they are foreign, as they are in their "own" table(s).



After the two types of Keys follows the data meant for that table. Each position has the same kind of information, i.e. if the Primary Key is the CPR number, the first Data field could be name, the second the address and so on for that person. And the next line would be the same information for the next person.

So if we want to find a certain persons information, we look for the Primary Key and read the Data fields.



The database software must then take care of the classical 4 :

- ✓ *C : create a new table or a new line in a table*
- ✓ *R : read specified lines from table*
- ✓ *U : update existing lines with new values*
- ✓ *D : delete an existing table or delete an existing line*

And the DB software must of course assure the uniqueness of the Primary Key. And take appropriate action if that rule is going to be violated.

Tables in huge DBs are usually indexed. That is a new table, which gives faster access/lookups to the data table. F.ex in a huge table with persons say 1.000.000 a new table with a few Primary Keys and the line number, where it is in the huge file makes it faster to find any value of the Foreign Key.

If anything is changed in the huge table, then a new index table is needed.

Modern databases can in addition to this combine data in a lot of different ways. This is often done with the assistance of SQL – Standard Query Language. The full SQL is beyond this simulation (but ok, if you have time for it too, then go ahead ;-)). But in the Exercise part, we will ask for a basic search.

Data

In this exercise we use data from www.IMDB.com, i.e. all kind of information on movies. The files are huge and can't f.ex. be completely imported into Excel. I.e. we are in a real life situation.

A '\N' *is used to denote that a particular field is missing or null for that title/name. The available datasets are as follows:*

---

title.akas.tsv.zip - Contains the following information for titles:

- titleId (string) - a tconst, an alphanumeric unique identifier of the title **(Primary Key)**
- ordering (integer) – a number to uniquely identify rows for a given titleId
- title (string) – the localized title
- region (string) - the region for this version of the title
- language (string) - the language of the title
- types (array) - Enumerated set of attributes for this alternative title. One or more of the following: "alternative", "dvd", "festival", "tv", "video", "working", "original", "imdbDisplay". New values may be added in the future without warning
- attributes (array) - Additional terms to describe this alternative title, not enumerated
- isOriginalTitle (boolean) – 0: not original title; 1: original title

title.basics.tsv.zip - Contains the following information for titles:

- tconst (string) - alphanumeric unique identifier of the title **(Primary Key)**
- titleType (string) – the type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc)
- primaryTitle (string) – the more popular title / the title used by the filmmakers on promotional materials at the point of release
- originalTitle (string) - original title, in the original language
- isAdult (boolean) - 0: non-adult title; 1: adult title
- startYear (YYYY) – represents the release year of a title. In the case of TV Series, it is the series start year
- endYear (YYYY) – TV Series end year. '\N' for all other title types
- runtimeMinutes – primary runtime of the title, in minutes
- genres (string array) – includes up to three genres associated with the title

title.crew.tsv.zip – Contains the director and writer information for all the titles in IMDb. Fields include:

- tconst (string) - alphanumeric unique identifier of the title **(Primary Key)**
- directors (array of nconsts) - director(s) of the given title **(Foreign Key)**

- writers (array of nconsts) – writer(s) of the given title **(Foreign Key)**

title.episode.tsv.zip – Contains the tv episode information. Fields include:

- tconst (string) - alphanumeric identifier of episode **(Primary Key)**
- parentTconst (string) - alphanumeric identifier of the parent TV Series **(Foreign Key)**
- seasonNumber (integer) – season number the episode belongs to
- episodeNumber (integer) – episode number of the tconst in the TV series

title.principals.tsv.gz – Contains the principal cast/crew for titles

- tconst (string) - alphanumeric unique identifier of the title **(Primary Key)**
- ordering (integer) – a number to uniquely identify rows for a given titleId
- nconst (string) - alphanumeric unique identifier of the name/person **(Foreign Key)**
- category (string) - the category of job that person was in
- job (string) - the specific job title if applicable, else '\N'
- characters (string) - the name of the character played if applicable, else '\N'

title.ratings.tsv.gz – Contains the IMDb rating and votes information for titles

- tconst (string) - alphanumeric unique identifier of the title **(Primary Key)**
- averageRating – weighted average of all the individual user ratings
- numVotes - number of votes the title has received

name.basics.tsv.gz – Contains the following information for names:

- nconst (string) - alphanumeric unique identifier of the name/person **(Primary Key)**
- primaryName (string)– name by which the person is most often credited
- birthYear – in YYYY format
- deathYear – in YYYY format if applicable, else '\N'
- primaryProfession (array of strings)– the top-3 professions of the person
- knownForTitles (array of tconsts) – titles the person is known for

---

Exercise ITO

Will be introduced by David.

Exercise Design

Design the basic software needed for CRUD of a DB working with files from IMDB. One file for each table.

Include a search tool which can combine and select information from 2 tables.

The Design must be covering, what is learnt up until now, i.e.

- Business Vision
- Use Cases (remember FURPS+)
- Use Case Diagram
- Domain Model
- System Sequence Diagram (eventually Operational Contracts)
- Sequence Diagram (eventually Operational Contracts)
- Design Class Diagram

The Domain Model can be replaced by a ERD (Entity Relation Diagram). It is illustrating the tables, attributes, type, Primary Key and eventually Foreign Keys.

The Build a Base Design part MUST be documented in a PDF file with text and diagrams before Wednesday, October 31rd, 12 o'clock and uploaded group wise to Fronter.

We receive a review from Dat18I Monday, November 5th 8:30 →

Exercise Programming

Implement your design.

The complete Build a Base incl. programming part must be uploaded group wise to Fronter as a link to a GitHub repository before Tuesday, November 16th 16:00 o'clock with revised Design documents and code.