a **relation** is a table, a **tuple** is a row

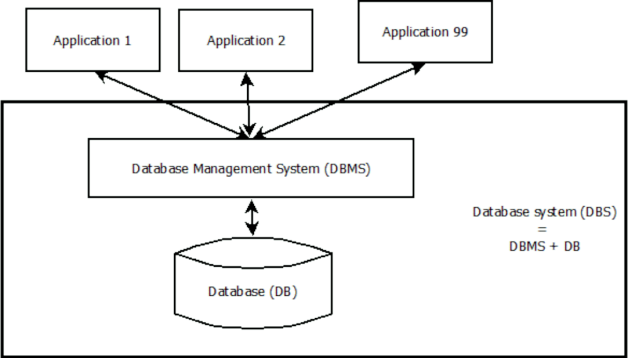
**Database** – structured collection of data.

**Data Base Management System (DBMS)** –

a collection of methods to access (read, modify, write) data into a DB

The DBMS (a piece of software) manages permanently the data of a database. The application developer just have to learn the language of the DBMS.

The DBMS provides already all relevant functionality for the data.



These data is **saved only once** and references between each other are possible.

**Involved user groups:**

* Requests: End user and application developer
* Database programming: application developer
* Database definition: database administrator
* File organization: system administrator, database administrator
* View definition (for end user): application administrator

Operational practice in companies has three main components:

* Business processes
* Application software
* Data

All of them change over time. The fundamental renovation come in:

* Every 1-2 years for business processes
* Every 2-5 years for application software
* Every 10-20 years for database schemes

**Development cycle for database applications:**

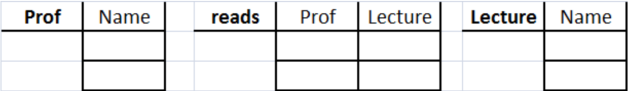
* Design database by logic (What has to be in the database?)
* Design database for system (How should the data be saved?)
* Develop database applications (How is the data processed?)
* Fill database (How to insert the data?)
* Maintain database (the database runs and runs and runs..)

**Entity-Relationship-Model** for logical draft of the database. The focus is on the business requirements. This language is not implemented in any DBMS.



**SQL** as language for definition of ”relational DBMS”. The aspect of an ERM are transformed into a form that can be ”understand” by a

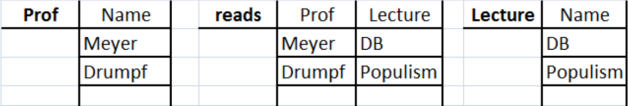
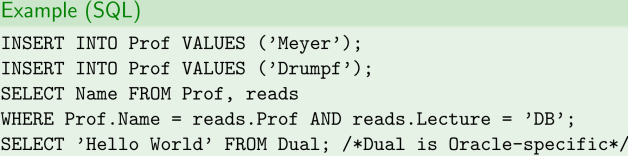
DMBS. The model underlying SQL is quite simple: in general tables (called: **relations**) are managed. The language SQL is implemented by almost every DBMS.

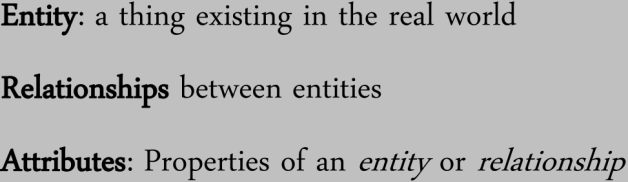


The models that are created with language Entity-Relationship-Model and SQL are called **schema**.

These languages are used to define data structures and are therefore called **Data Definition Language (DDL)**.

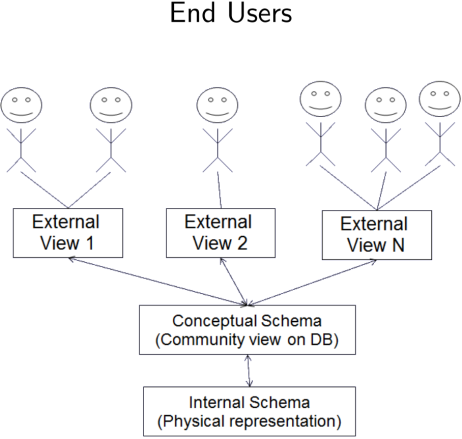
SQL as language for Manipulation and Search in the database. This sublanguage is as well part of SQL and is called **Data Manipulation Language (DML)**.

** **

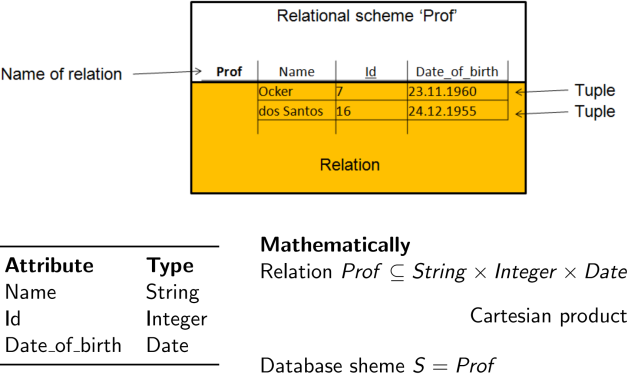
****

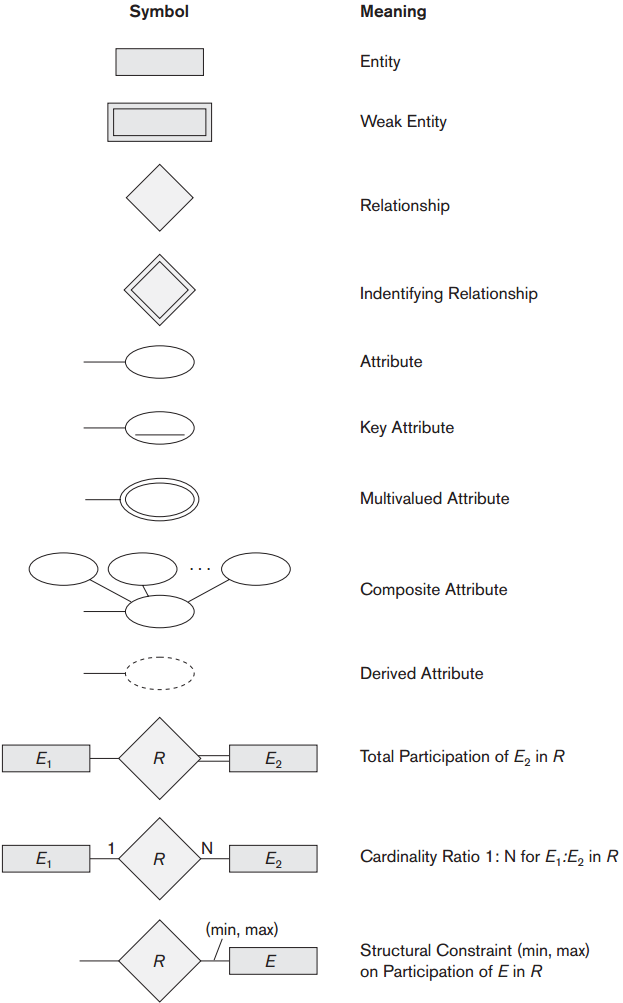
**ANSI/SPARC 3-Level-Architecture**

* External Level
* Conceptual Level
* Internal Level

****

* The **internal scheme** describes the system specific realization of the database, thus the storage location and DBMS-specific way of accessing the data
* The **conceptual scheme** (also called the logical scheme) contains the DBMS-independent modeling of the whole database.
* Based on the conceptual scheme there are **external schemes** defined for different applications or different users. These schemes typically provide application related sections of the whole data pool. It is also possible to support different database languages at this level.

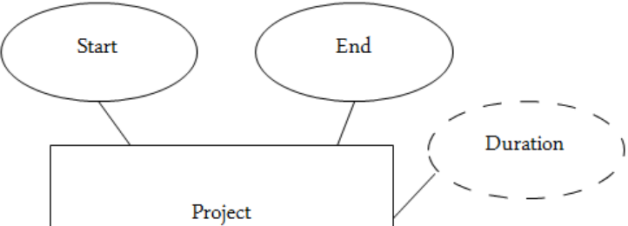




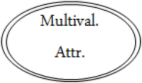












**Codd’s Rule #1**

The information rule:

* All information in the database is to be represented in one and only one way, (namely by values in column positions within rows of tables).
* No redundancies
* Only one datum per field
* Applies to application data and meta data

**Codd’s Rule #2**

The guaranteed access rule:

* All data must be accessible with no ambiguity. This rule is essentially a restatement of the fundamental requirement for primary keys. It says that every individual scalar value in the database must be logically addressable by specifying the name of the containing table, the name of the containing column and the primary key value of the containing row.[Wikipedia]
* Remember: No pointers to data.

**Codd’s Rule #3**

Systematic treatment of null values:

* The DBMS must support a representation of ”missing information and inapplicable information” that is systematic, distinct from all regular values, and independent of data type.
* NULL is distinct from empty string or zero
* Oracle treats empty string

(a zero length varchar / ”) as NULL

**Codd’s Rule #4**

Active online catalog based on the relational model:

* The system must support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language. That is, users must be able to access the database’s structure (catalog) using the same query language that they use to access the database’s data.
* Data Dictionary

**Codd’s Rule #5**

Comprehensive data sublanguage rule:

* Data Definition
* View Definition
* Data Manipulation (Interactive and by program).
* Integrity Constraints
* Authorization
* Transaction boundaries
* SQL

**Codd’s Rule #6**

The view updating rule:

* All views that are theoretically updatable must be updatable by the system.[Wikipedia]
* That’s hard.
* Own research topic

**Codd’s Rule #7**

High-level insert, update, and delete:

* The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data.
* Data manipulation not only for single row, but for many rows in one operation.

**Codd’s Rule #8**

Physical data independence:

* Changes to the physical level (how the data is stored, whether in arrays or linked lists etc.) must not require a change to an application based on the structure.[Wikipedia]
* Very important feature of RDBMS.

**Codd’s Rule #9**

Logical data independence:

* Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure.[Wikipedia]
* Important (but difficult) feature.
* May depend on DB and application design.

**Codd’s Rule #10**

Integrity independence:

* Integrity constraints specific to a particular relational data base must be definable in the relational data sub-language and storable in the catalog, not in the application programs.
* One of the key features of RDBMS
* But: Not all integrity constraints can be expressed in such a way as to be storable in the DBMS.

**Codd’s Rule #11**

Distribution independence:

* Data may be stored in different locations
* This must be transparent to the user (meaning invisible)!
* Existing applications should continue to operate successfully :
* When a distributed version of the DBMS is first introduced; and
* When existing distributed data are redistributed around the system.

Problem: CAP Theorem a distributed computer system can not guarantee all of:

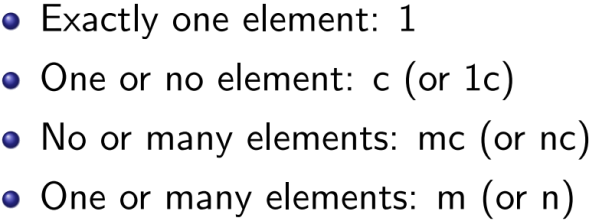
* Consistency
* Availability
* Partition tolerance

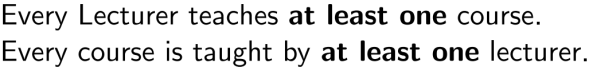
**Codd’s Rule #12**

The nonsubversion rule:

* If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system, for example, bypassing a relational security or integrity constraint.
* Problem, if the DB can access data files in the file system.

















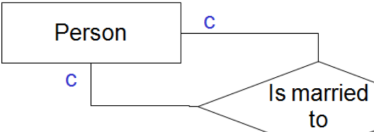


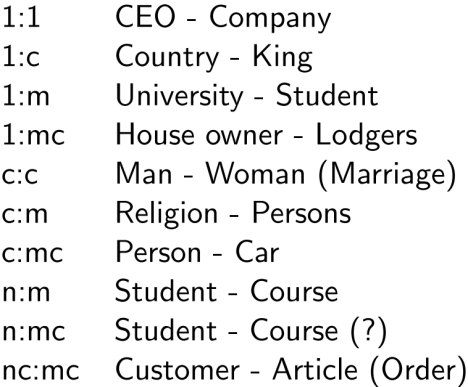






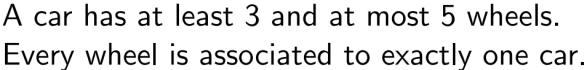




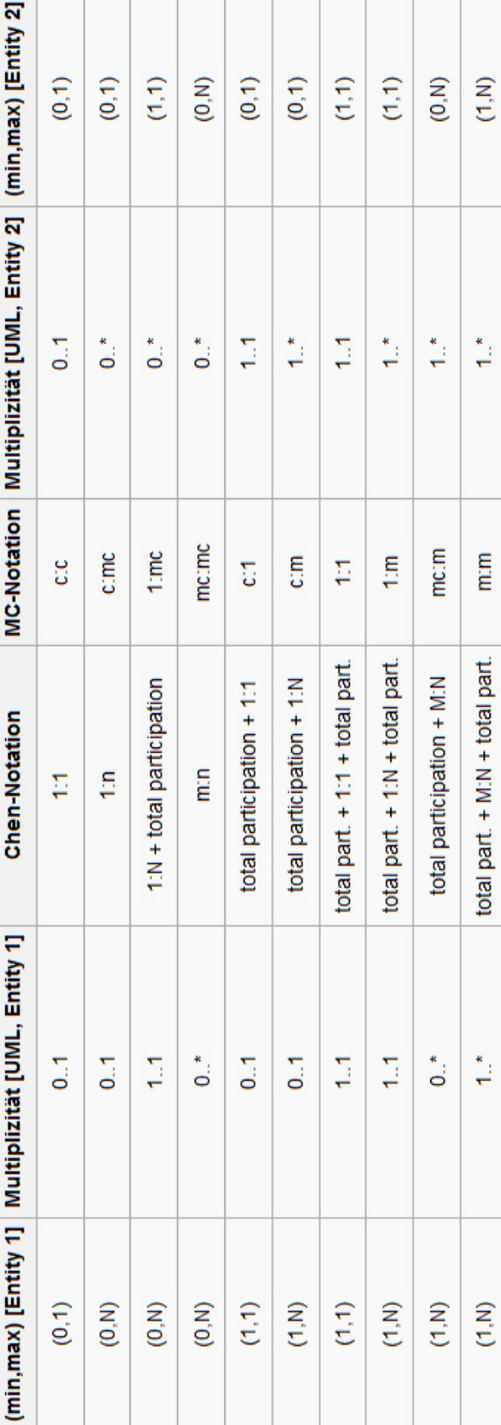




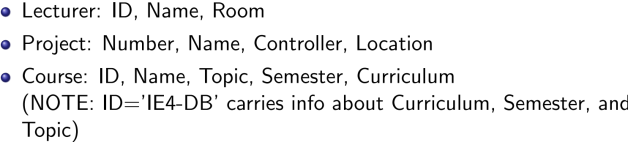






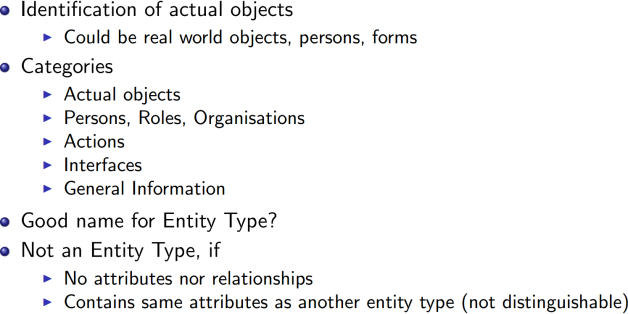




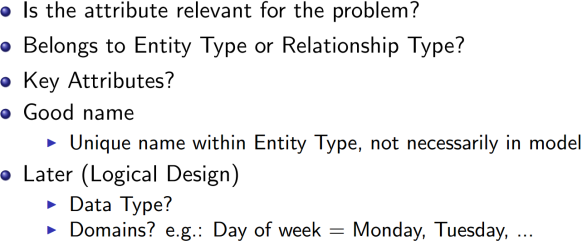




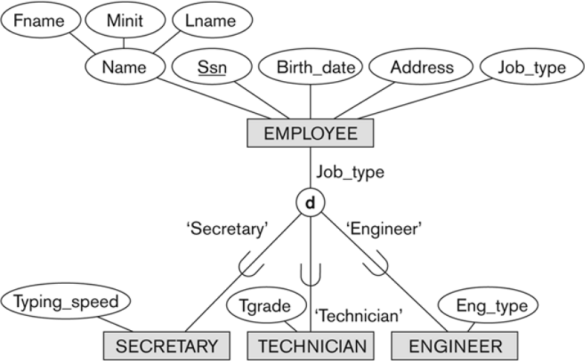




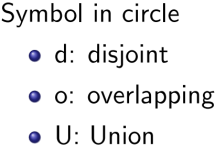


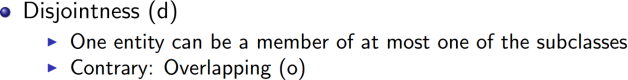




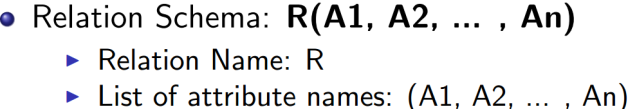


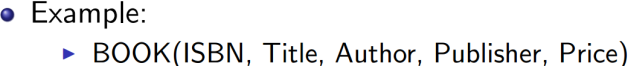




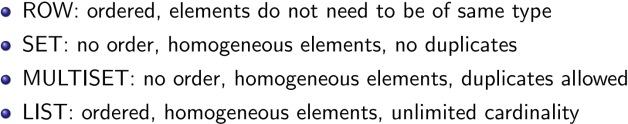




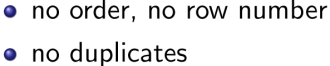




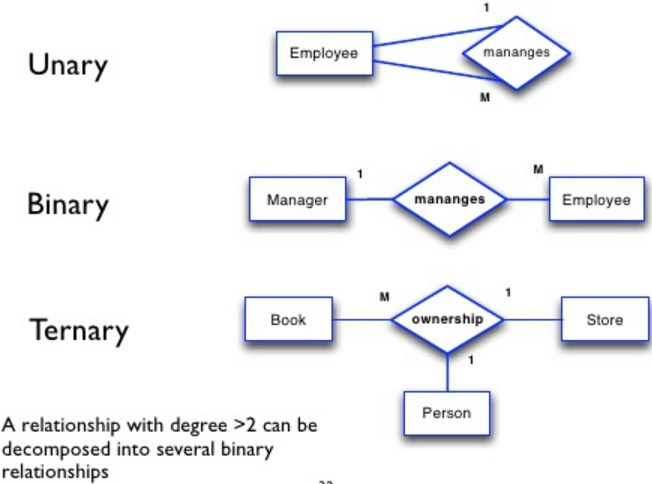


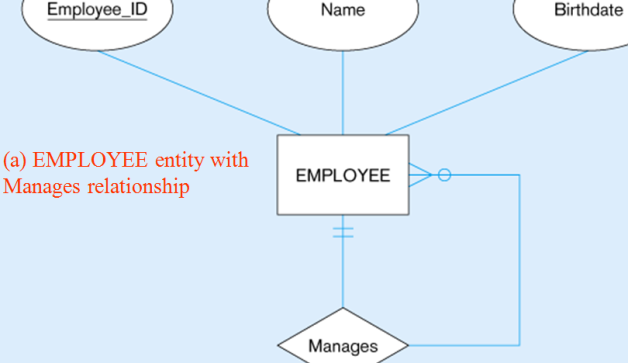


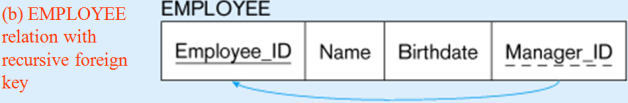




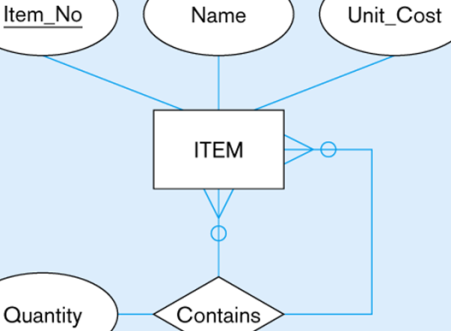


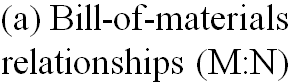




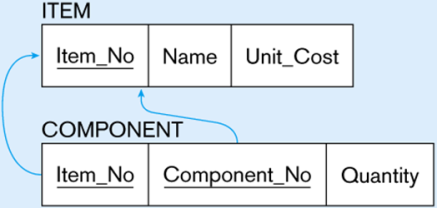








**(b) ITEM and COMPONENT relations**



Mapping the binary relationship







