Database solutions

Database environment

Marzena Nowakowska
Faculty of Management and Computer Modelling
Kielce University of Technology
room: 3.21 C

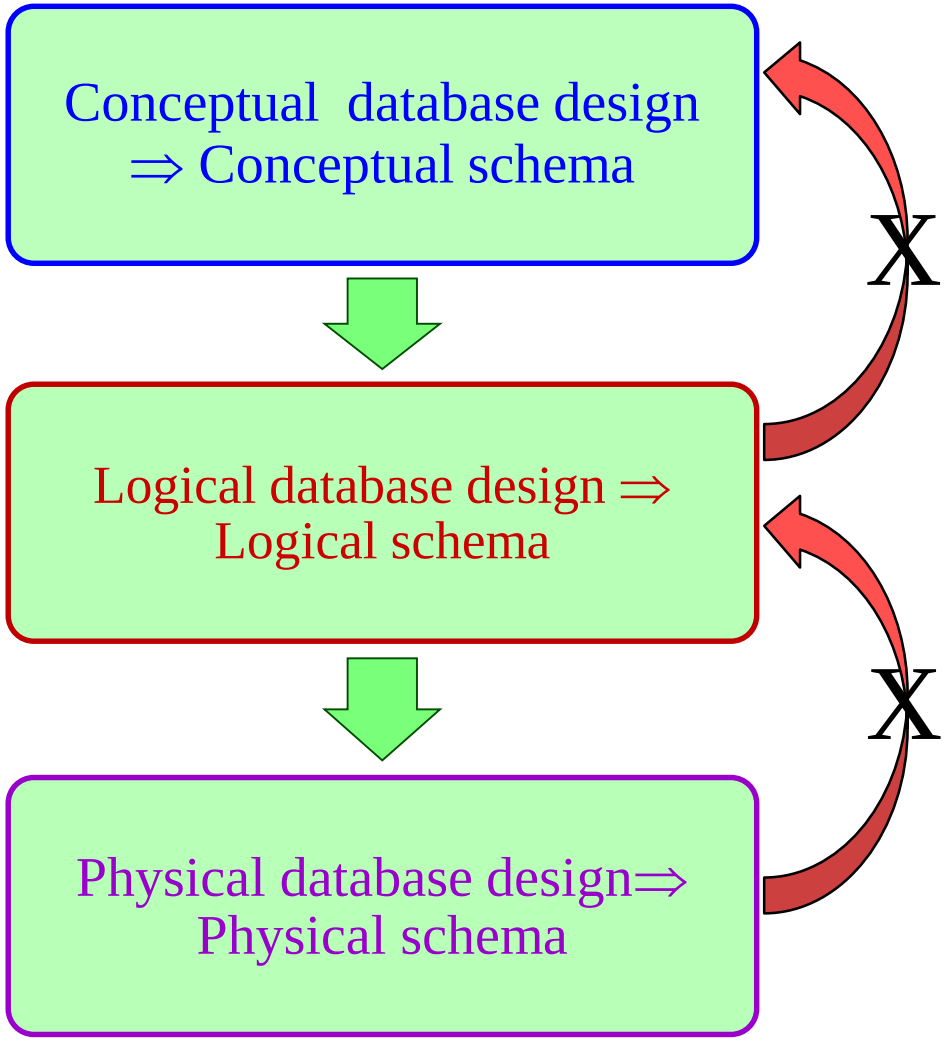
Data independence – general concept

Database schema is the overall description of a database. The schema is specified during the database design process and is not expected to change frequently.

Logical data independence refers to the immunity of conceptual schemas to changes in the logical schema.

Physical data independence refers to the immunity of logical schemas to changes in the physical schema.

Data independence means that upper level of database schema are unaffected by changes to lower level.



Data sublanguage

The definition of a relational system requires that a single language – sometimes called a **comprehensive data sublanguage** – be able to handle all communications with the database.

In the world of relational database management system (RDBMS) that language is **SQL** – Structured Query Language.

This is non-procedural language – the language allows user specify **what** should be done rather than **how** it is to be done.

SQL has been standardized by the International Standards Organization (ISO).

SQL commands classification:

- DDL – Data Definition Language
- DML – Data Manipulation Language
- DAL – Data Administration Language

Data dictionary

Data dictionary (also known as **system catalog**) is a repository of information describing the data in the database. The amount of information and the way the information is used vary with the DBMS.

Typically, the system catalog stores:

- metadata: names, types, and sizes of data items,
- names of relationships,
- integrity constrains on the data,
- names of authorised users who have access to the data.

List of functions for a DBMS

1. **Data storage, retrieval, and update services**
2. **A user-accessible catalog**
3. **Transaction support** (all the updates corresponding to a given transaction are made or none of them are made)
4. **Concurrency control services** (a database is updated correctly when multiple users are updating the database at the same time)
5. **Backup and recovery services** (recovering the database in the event that the database is damaged in any way)
6. **Authorization (security) services** (only authorized users can access the database or certain parts of the database)
7. **Replication support** (managing copies of the same database at multiple locations)
8. **Integrity services** (both data in the database and changes to the database follow certain constraints rules)
9. **Services to promote data independence** (supporting the independence of programs from the actual structure of the database)
10. **Utility services (**assistance in the general maintenance of the database)

The cornerstone of relational databases

Dr. Edgar Frank Codd revolutionary idea was to structure data in tables with defined relationships, allowing for efficient storage and retrieval of information.

Prior to the idea, databases were primarily hierarchical or network-based, which posed significant challenges in data retrieval and management.

Codd formulated rules that were named after their creator. The rules comprise a set of guidelines (outlines, principles) that ensure the integrity, reliability, and efficiency of a relational database system.

The rules have played an instrumental role in shaping the field of database management.

13 Codd rules provide specific criterions for evaluating the effectiveness of a relational database.

Codd rules for a RDBMS

- **Essential rules**: Rule 0 – Foundational rule
  Rule 12 – Nonsubversion rule
- **Structural rules**: Rule 1 – Information representation
  Rule 6 – View updating
- **Integrity rules**: Rule 3 – Systematic treatment of null values
  Rule 10 – Integrity independence
- **Data manipulation rules**:
  Rule 2– Guaranteed access
  Rule 4 – Dynamic on-line catalog based on the relational model
  Rule 5 – Comprehensive data sublanguage
  Rule 7 – High-level insert, update, delete
- **Data independence rules**:
  Rule 8– Physical data independence
  Rule 9 – Logical data independence
  Rule 11 – Distribution independence

Essential rules: Rule 0

Rule 0 – Foundational rule

For any system that is advertised as, or claimed to be, a relational database management system, that system must be able to manage databases entirely through its relational capabilities, no matter what additional capabilities the system may support.

Rule Zero is used for classifying relational database management systems (RDBMSs). The rule ensures that the underlying databases are truly relational. Even if other factors come into play, the **relational nature** of the way data is managed **has to be the core function**.

It's called "Rule Zero" because it is considered the fundamental rule for all RDBMSs.

Essential rules: rule 12

Rule 12 – Nonsubverstion rule

If a relational system has a low-level (single-record-at-a-time) interface (language), that level cannot be used to bypass (overthrow, subvert, damage) the integiry rules or constrains expressed in the higher-level relational (multiple-records-at-a-time) interface.

The rule states that if a RDMS has an interface that provides access to low level records, this interface then must not be able to subvert the system and bypass security and integrity constraints.
Overall, there should not be a way in any form to violate the integrity constraints defined on a database.
Only the language/sublanguage which was used to define those constraints can be able to redefine them.

Structural rules: Rule 1

Rule 1 – Information representation rule

All information in a relational database is represented explicitly at a logical level and in exactly one way – by values in tables.

The phrase “logical level” indicates that there’s nothing about:

- how physical storage is done,
- pointer (it is a tool that allows fast and predictable accesses of data without violating neither temporal or logical consistency nor transaction serialization) chains,
- the physical position of data in arrays, files, or anything else.

Structural rules: Rule 6

Rule 6 – View updating rule

All views that are theoretically updatable are also updatable by the system.

A view is the dynamic result of one or more relational operations (relational algebra) acting on the base relation (table) to produce another relation – a virtual one.

A virtual relation does not exist in the database but it is produced upon request by a particular user. A view has a name and is created using the SELECT command of SQL language.

Rule 6 is one of the most challenging rules to be implemented in practice, and no commercial product fully satisfies it today. A view is theoretically updateable as long as it's made up of columns that directly correspond to real table columns.

Integrity rules: Rule 3

Rule 3 – Systematic treatment of null values

Null values (distinct form the empty character string and form the zero value) are supported for representing missing information and inapplicable information in a systematic way, independent of a data type.

The rule demands that it must be able to use a NULL placeholder irrespective of data type used. NULLs are distinct from an empty character string or any other number, and they are always to be considered as unknown on non-existing values. This rule insists provisions for manipulating NULL values in all the possible ways.

Integrity rules: Rule 10

Rule 10 – Integrity independence

Integrity constraints specific to a particular relational database must be definable in a relational data sublanguage and storable in the catalog, not in the application programs.

The following two integrity constraints must be supported:

1. **Entity integrity.** No component of a primary key is allowed to have a null value. *That is, no records can have NULL values in its Primary Key attribute.*
2. **Relational integrity.** For each distinct non-null foreign key value in a relational database, there must exist a matching primary key value from the same domain. *In other words, if a foreign key cannot have null values as its component then it must refer a matching primary key value with the same set of permitted values to accept any new records.*

Data manipulation rules: Rule 2

Rule 2 – Guaranteed access

Each and every datum (atomic value) is a relational database is guaranteed to be a logically accessible by restoring to a combination of table name, primary key value and column name.

It must be possible to access every single piece of information from the tables in some way. The combination “Table name + Primary key + Column name” must allow to find the information what is needed. In the case of multiple records in the result, the combination of “Table name + Column name” or the “Table name + * “ would find what is needed.

Data manipulation rules: Rule 4

Rule 4 – Dynamic on-line catalog based on the relational model

The database description is represented at the logical level in the same way as ordinary data, so authorized users can apply the same relational language to its integration as they apply to the regular data.

The rule requires that a relational database be self-describing. In other words, the database must contain certain system tables whose columns describe the structure of the database itself, or alternatively, the database description is contained in user-accessible tables. In other words, this rule insists a data dictionary that stores meta data.

Data manipulation rules: Rule 5

Rule 5 – Comprehensive data sublanguage

There must be at least one language the statements of which can express all of the following items:

(1) data definition,

(2) view definition,

(3) data manipulation,

(4) integrity constraints,

(5) authorization,

(6) transaction management operations.

This rule requires the existence of SQL like language to manipulate data. Manipulation should involve all the above said things (highlighted by numbers).

Data manipulation rules: Rule 7

Rule 6 – High-level insert, update, delete

The capability of handling a base relation or a virtual relation (view) as a single operand applies not only to the retrieval of data but also to the insertion, update, and deletion of data.

This rule stresses the set-oriented nature of a relational database. It requires that rows be treated as sets in insert, delete, and update operations. The rule is designed to prohibit implementations that support only row-at-a-time, navigational modification of the database.

The SQL language covers this via the INSERT, UPDATE, and DELETE statements.

Data independence rules: Rule 8

Rule 8 – Physical data independence

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods.

Changes in the physical level must not lead to a change in the next higher level called logical level.

Applications that are defined on the physical level should be able to continue to work even when changes are made to the internal implementation of data storage and access methods.

The way the data are stored physically must be independent of how its accessed logically.

Data independence rules: Rule 9

Rule 9 – Logical data independence

Application programs and terminal activites remain logically unimpaired when information-preserving changes of any kind that theoretically permit unimpairment are made to the base tables.

The addition or removal of new entities, attributes, or relationships to the conceptual schema should be possible without having to change existing external schemas or having to rewrite existing application programs.

For example, for a table with the schema Employee(Eno, Ename, Street, City, Salary), a view with the attributes (Eno, Ename) will not be affected if any other attributes of Employee is altered.

Data independence rules: Rule 11

Rule 11 – Distribution independence

The data manipulation sub-language of a RDBMS must enable application programs and inquires (terminal activities) remain logically the same (unimpaired) whether and whenever data are physically centralized or distributed.

The distribution of the parts of any database to different sites/locations should be invisible to the end users. That is, like in distributed database, it should give a centralized effect to the end users who access it.

Also, the existing applications are able to continue their operations when the database is distributed or redistributed.