C2. Database Design. SQL DDL

As in case of other complex systems, the design phase is crucial in obtaining an efficient database structure and implementation. Therefore, several steps in database (DB) development are thought to support a good design. These steps are:

- Requirement analysis (implies functionality design)
- Conceptual design (e.g. using ER model)
- Logical DB design (e.g. using relational model)
- DB Schema refinement (normalization, optimization)
- Physical DB design (indexes)
- Client applications design (software design)

Requirement analysis is meant to extract the entities and data involved in a real world enterprise. During this step, main functionalities are identified. Conceptual models are based on requirement analysis and are used to analyze and understand application data. The most used model in this phase is the Entity-Relationship (ER). UML could be also considered, but is more appropriate for object oriented data models than for relational one. Logical data models are used to describe data in terms of domains and logical links. Database schema is the result of logical model refinement. It represents a description of a particular collection of data, using a given data model.

The relational model is the most widely used data model today. It is used in conjunction with relational data management systems (RDBMS), which are de-facto standard in industry.

To be efficient for a specific system, the relational model is refined further in a physical model. This model offers representation of a data considering the facilities and constraints of a specific DBMS.

1.1. Relational database design. A practical perspective.

For basic applications the relational database schema results directly from requirements analysis. In case of complex applications conceptual design is required. In this chapter only first case is considered. For logical design please refer the appropriate chapter form the database course.

1.2. The relational data model

The relational data model is based on the relational algebra proposed by E.F. Codd in early '70. A database consists of a collection of **relations** (or tables) on which is applied relational operators to manage the collection of data. A relation is a set of rows. Each row is a distinct tuple $\{a_1, a_2, \ldots a_n\}$. It has a fixed number of **attributes** (columns) and each attribute has a specific type or **domain**.

Two parts will describe a relation:

- **Instance**: a table, with rows and columns. Each row represents a record denoting a valid entity from the application. Each column corresponds to an attribute and can hold just one value. (#Rows = cardinality, #fields = degree)
- **Schema**: specifies name of relation, plus name and type of each attribute. E.g.: Students (sid: string, name: string, grade: real).

A set of fields is a superkey for a relation if no two distinct tuples can have same values in all its fields (all tuples are unique). A set of fields is a key if any its smaller subset does not keep this property. **Primary key** is a key selected by DBA to identify the relation. E.g.: *SID* is a primary key for the relation Students.

A **foreign key** is set of fields in one relation that is used to refer to a tuple in another relation. It must correspond to primary

key of the second relation. It is like a logical pointer – implements logical links between two relations.

A **functional dependency** between two attributes (X, Y) of a relation *R* is denoted by $X \rightarrow Y$, and holds if, and only if

$$\forall t_1, t_2 \in R, \quad t_1(X) = t_2(X) \Rightarrow t_1(Y) = t_2(Y),$$

where t_1 and t_2 represents two distinct tuples of *R*.

Several inference axioms and rules could be used to discover other dependencies based on a list of direct dependencies between relation's attributes.

- Reflexivity: $X \supseteq Y \Rightarrow X \rightarrow Y$
- Augmentation: $\{X \rightarrow Y\} \Rightarrow XZ \rightarrow YZ$
- Transitivity: $\{X \rightarrow Y \wedge Y \rightarrow Z\} \Rightarrow X \rightarrow Z$
- Decomposition: $\{X \rightarrow YZ\} \Rightarrow X \rightarrow Y, X \rightarrow Z$
- Union: $\{X \rightarrow Y \wedge X \rightarrow Z\} \Rightarrow X \rightarrow YZ$
- Pseudotransitivity: $\{X \rightarrow Y \wedge WY \rightarrow Z\} \Rightarrow WX \rightarrow Z$

1.3. Normalization

Normalization represents the process of organizing the fields and tables of a relational database that aims to reduce redundancy of information in the database. The redundancy is bad since it contributes to disk space wasting and could cause various data inconsistencies. Database normalization is a gradually process involving several normal forms. Each normal form takes all prior constraints and adds new conditions. There are five generally accepted normal forms, but a relational database is often described informally as "normalized" if all its tables satisfy the Third Normal Form. To ensure a normal form a table has to be transformed using decomposition. E.g. replacing ABCD with, say, AB and BCD, or ACD and ABD.

I) *The First Normal Form*

A relation is in first normal form if the domain of each attribute contains only **atomic** values, and the value of each attribute contains **only a single value from that domain**.

Solutions: Non-atomic attributes have to be split in two or more atomic ones. Multiple values attributes should be extracted in new relations. These relations will have the same primary key as the original one that will serve also as foreign key linked to original relation.

II) The Second Normal Form

A relation is in the second normal form if it satisfies the first normal form and all non-key attributes are **fully functional dependent** on the primary key. A relation fails to fulfill this form if exists at least one non-key attribute that depends on just a part of the primary key (a subset of attributes that form primary key).

Solution: Attributes that depend only partially by the primary key should be extracted in new relations. These relations will have the part of the primary key that determines that attributes as primary key (it will serve also as foreign key linked to original relation).

Note that if the primary key is not a composite key, all non-key attributes are always fully functional dependent on the primary key. A table that is in first normal form and has the primary key formed by a single attribute is automatically in the second normal form.

III) The Third Normal Form

A database is in the third normal form if it satisfies the following conditions: it is already in the second normal form and there is no **transitive functional dependency** between the primary key and a non-primary attribute.

By transitive functional dependency between A and C we understand that exists B so that A is functionally dependent on B, and B is functionally dependent on C. In this case, C is transitively dependent on A via B.

Solution: extract B and C into a new relation, with B as both primary key and foreign key linked to the original relation.

1.4. SQL DDL

A data definition language (DDL) in databases field is a collection of commands used for defining data structures in order to describe a database schema. The DDL subset of SQL is formed by a collection of statements whose effect is to modify the schema of the database by adding, changing, or deleting definitions of tables or other objects (e.g. views or indexes).

I) CREATE TABLE

Used to create a new relation (table) into an existing database.

The general syntax is:

```
CREATE TABLE table_name (
    column_name1 data_type[(size)] [attr_constraint],
    column_name2 data_type[(size)] [attr_constraint],
    ...
    column_nameN data_type[(size)] [attr_constraint]
    [,
    CONSTRAINT constraint_name {
                    CHECK (logical expression), |
                    PRIMARY KEY (attr list), |
                    FOREIGN KEY (logical expression)
                                REFERENCES table name,
                    }
    …
    ]
);
```

The values accepted by the data_type are system dependent. In case of Oracle it accepts (a selection): CHAR, VARCHAR2, INT, INTEGER, FLOAT, REAL, NUMBER(p,s), NUMERIC(p,s), DATE, BLOB.

For attr_constraint the following values could be used: PRIMARY KEY (only for single attribute PK), NOT NULL, UNIQUE, CHECK, REFERENCES (for FOREIGN KEYS).

II) DROP TABLE

Used to delete an existing relation (table) from a database.

The general syntax is:

DROP TABLE table_name [**PURGE**];

Depending on the system, it is very unlikely to recover the data from the original table. In Oracle, if **PURGE** is specified the table could not be recovered by rolling back the command.

III) ALTER TABLE

Used to modify an existing relation (table) definition from a database or to add additional structures (e.g. indexes).

The general syntax is:

ALTER TABLE table_name
[{**ADD** | **DROP COLUMN**} column_name,] |
[**MODIFY** column_name data_type[(size)],] |
[**ADD CONSTRAINT** constraint_name {
CHECK (logical expression), |
FOREIGN KEY (logical expression)
REFERENCES table name,
} |
[{**ENABLE** | **DISABLE**}
CONSTRAINT constraint_name,] |
[**DROP CONSTRAINT** constraint_name,];

1.5. Practical exercise

As a practical example we consider here a simplified database for University management. Suppose that requirements analysis has resulted in the following description of the entities involved:

Students

- has a name (comprising a first name and a last name)

- has a unique ID number (a Student ID, unique, allocated by the university in form of a character string: department abbreviation + unique number)
- is in some year of study (e.g. 2∈ {1,2,3,4})
- is enrolled at exactly one Department of the University (described by a Department ID: abbreviation up to 5 chars, a department name and an address)

Courses

- has a name (e.g., "Databases")
- has a unique identifier (e.g., "DB2EN")
- is worth some number of credits
- is offered by a department
- has zero or more prerequisite courses
- has zero or more students enrolled in a specific year (e.g. 2014), for each of them having an associated mark (that can be NULL before exams). The mark should be a float number >=1.00 and <=10.00 with 2 digits in the decimal part.

Using the above informal description, please create and normalize a corresponding relational schema. Based on this schema, implement the database using Oracle APEX SQL Workshop/SQL Commands interface to run all necessary SQL DDL statements.