Chapter 03: Relational Model

1. Introduction :

A database is a finite set (of instances) of relations. The schema of the database is the set of schemas of the relations in that database.

Schema of a relational database: *The schema of a database is defined by the set of schemas of the relations that compose it.*

The creation of a database schema is straightforward once all the relations that make up the database have been identified. However, choosing these relations is a challenging task, as it largely determines the database's characteristics and qualities: performance, accuracy, completeness, availability of information, etc. One of the key aspects of relational database theory is precisely to define what constitutes a good schema and to provide formal tools to achieve this.

In practice, a less rigorous but more accessible approach is used: designing the schema using a conceptual data model and then translating the resulting conceptual schema into a relational schema. The most common technique involves building the database from an Entity/Relationship schema.

In this chapter, we outline the rules for the process of transforming an Entity-Relationship model into a relational model.

Indeed, the relational model can be directly used by starting with a universal relation and progressively decomposing this relation into smaller relations in canonical normal forms, using a rigorous mathematical process for the decomposition. This approach ensures an accurate and sound schema for the database being designed.

☺ Advantages of the relational model:

One of the main advantages of the relational model is its great simplicity. Indeed, there is only one structure: the relation. A relation can easily be represented in the form of a table.

Another advantage of the relational model is the independence of application programs and interactive sessions from the internal representation of data. Moreover, this model has a solid theoretical foundation for addressing issues of consistency and redundancy.

2. Principle of the relational model:

For a relational DBMS, the following properties apply:

☞ *There are only tables.*

☞ *A table consists of a relational schema that includes:*

- ✓ *A table name, which is the name of the relation.*
- ✓ *A set of columns, where each column represents an attribute of the relation. Each attribute has a name and a type, generally a scalar type.*
- ✓ *Each relation has an attribute or a group of attributes that represent the primary key of the relation.*
- ✓ *A content: this is a set of rows representing the actual values, or occurrences, of the relation.*

☞ *The order of rows and columns does not matter:*

- ✓ *The order of the columns: it does not matter, but as a rule, we usually begin with the attributes that identify the relation, followed by additional information.*
- ✓ *The order of the rows: it does not*
- ✓ *matter, but it reflects the temporal evolution of the table.*

Example 1: table of films

ID	Title	Director	Year	Genre
1	The Godfather	Francis Ford Coppola	1972	Crime, Drama
2	The Shawshank Redemption	Frank Darabont	1994	Drama
3	Schindler's List	Steven Spielberg	1993	Drama, History
4	Inception	Christopher Nolan	2010	Action, Sci-Fi

3. ***Relation schemas**: A relation schema is simply a name followed by a list of attributes, each associated with its domain. The syntax is:*

   R(A1 :D1, ..., An :Dn)

   Where Ai are the attribute names and Di are the domains. The arity of a relation is the number of its attributes. In a relation schema, the same domain can appear multiple times, but each attribute name can appear only once. The domain can be omitted during the definition phase.

   ***Instance of a relation**: An instance of a relation R, or simply a relation, is mathematically defined as a finite subset of the Cartesian product of the domains of the attributes of R. Recall that the Cartesian product $D1 \times D2 \times \ldots \times Dn$ of domains D1, D2, ..., Dn is the set of all tuples (v1, v2, ..., vn) where $vi \in Di$. One of the foundations of the relational model is set theory, and the notion of a relation in the model strictly corresponds to the mathematical concept in this theory. A relation is represented in the form of a table, and these two terms are often used interchangeably.*

   ***Example 2:** In Example 1, the tuple **film**("**Schindler's List**", "**Steven Spielberg**", **1993**, " **Drama, History** ") is an instance of the relation **film**.*

4. ***Mathematical interpretation of the relational model:***

   Let P be the Cartesian product of the domains D1, D2, ..., Dn. A table is a relation R between the domains D1, D2, ..., Dn. It is a subset of the Cartesian product P.
   Example: For instance, in a timetable management application, each teacher has a code, a first name, a last name, the course taught, day, and time.
   Professor(code, last name, first name, course, lecture-hall, day, time).

   4.1 Functional Dependency:

   - *Let R(A1, A2, ..., An) be a relation schema. Let X and Y be subsets of {A1, A2, ... An}. We say that Y is functionally dependent on X (X -> Y) if for every value of X, there is a unique corresponding value of Y.*
     We write: $X \rightarrow Y$.
     We say that: X determines Y.

Example 03:

Product(N°Prod, Name, UPrice)

N°Prod-> Name, UPrice

Mark(N°Exam, N°Pupil, Mark_p)

N°Exam, N°Pupil-> Mark_p

- *A functional dependency is a semantic property; it corresponds to a constraint that is assumed to always be true in the real world.*
- ***Elementary functional dependency**: Functional dependency X -> A, but A is a unique attribute not included in X, and there is no X' included in X such that X' -> A.*

***4.2 The key of a relation**:*

☞ *An attribute (or group of attributes) that determines all the others.*

☞ *A key uniquely determines a tuple.*

Example 04:

Let's take up example 3 again : N°Prod-> Name, UPrice

N°Prod is a key.

☞ *To find the key of a relation, one must carefully examine the assumptions about the real world.*

☞ *A relation can have multiple keys, which are called candidate keys.*

***Example 05:** in the relation Product name is a candidate key (provided that there are never two products with the same name).*

***4.2.1 Primary key**: The primary key of a relation is the smallest subset of attributes that uniquely identifies each row in a table. As we saw earlier in the **Film** table, where two rows are always different, the set of all attributes is itself a key, but we can almost always find the smallest subset that satisfies the condition. To distinguish the key, we will underline the (or the) attribute(s) of the key.*

***4.2.2 Foreign key**: A foreign key in a table Ti is an attribute in table Ti that is a primary key in another table Tj.*

***Example 6**: Here is a set of tables from a database:*

Client :

ID-Client	*Name*	*City*
01	*smith*	*paris*
02	*jones*	*paris*
03	*blake*	*nice*

Product :

ID-Product	*Name*	*UPrice*	*QStock*
01	*car hyundai eon*	*6000 euro*	*10*
02	*Motocycle MBK*	*10000 euro*	*20*
03	*Bicycle Detwin*	*300 euro*	*30*

sale :

ID-Client	*ID-Product*	*Date*	*Quantity*
01	*01*	*10/01/2004*	*01*
01	*02*	*23/11/2004*	*01*
01	*03*	*10/12/2004*	*01*
02	*01*	*10/02/2004*	*01*
02	*03*	*11/10/2005*	*02*
03	*01*	*03/10/2005*	*01*

Schemas of DataBase :

CLIENT (<u>ID-Client</u>, Name, City)

PRODUIT (ID-Product, Name,UPrice, Qstock)

VENTE (ID-Client, ID-Product, Date, Quantity)

*In this example, the **ID-Client** is a primary key in the **Client** relation and a foreign key in the **Sale** relation. Similarly, the **ID-Product** is a primary key in the **Product** relation and a foreign key in the **Sale** relation.*

5. *Theory of normalization*:

Normalization is a process in database design that aims to organize the attributes and relations of a database to reduce redundancy and improve data integrity. The goal is to ensure that the data is stored efficiently, with minimal duplication, and that the relationships between the data are well-structured.

*The normalization process involves decomposing complex relations into simpler, smaller ones while ensuring that the original data can still be reconstructed from the smaller relations. The process is carried out in several stages, known as **normal forms** (1NF, 2NF, 3NF, etc.), each addressing a specific type of anomaly or redundancy.*

Conclusion:

☞ *The theory of normalization is the ability to define a method for designing "good" tables, meaning without redundancies and without loss of information.*

☞ *It highlights "undesirable" relationships.*

☞ *In other words, it is a good decomposition of relations.*

☞ *It defines the normalization process, which allows for decomposing a non-normalized relation into an equivalent set of normalized relations.*

☞ *It defines the criteria for "desirable" relations, called normal forms.*

***Example 7**: This is an example of redundancy.*

Code	Last-name	First-name	Course	Lecture-Hall	Day	Time
Aaaaaa1	narssis	julie	DataBase	A3	Tuesday	11 :00
Aaaaaa1	narssis	julie	DataBase	A3	Tuesday	11 :00
Abaaaa1	mathiau	Frencois	Micro-CS	A2	Thursday	12 :30

Redundancy*: The course "Database" taught by Julie Narsis has been entered twice in the table.*

This kind of redundancy can lead to several problems, such as increased storage requirements and the risk of inconsistency if updates are not applied uniformly across all occurrences of the same data. To avoid redundancy, it's important to ensure that data is normalized so that each piece of information appears only once, and any duplicates are eliminated.

5.*1 Decomposition*:

☞ ***Objective****:*

- *Decompose the relations of the relational schema without loss of information.*
- *Obtain canonical or base relations of the real world.*
- *Achieve the normalized relational schema.*

☞ *The starting schema is the universal schema of the database.*

☞ *Through successive refinements, we obtain sub-relations without loss of information that will not be affected during updates (there is no redundancy). Indeed, at the end of the refinement process, we only obtain elementary functional dependencies.*

5.2 Normal Forms:

☞ ***Intuitive concept of Normal Form****:*

A "good relation" can be considered as a function from the primary key to the remaining attributes.

5.2.1 ***First Normal Form (1NF)****:*

A relation is in 1NF if every attribute is atomic (non-decomposable).

***Example 5.1**: This is a counterexample:*

***STUDENT** (ID-Stud, name, surname, List_Mark)*

An attribute cannot be a set of values.

***Decomposition**:*

***STUDENT** (ID-Stud, name, surname).*

GRADE (ID-Stud, ID-Subject, Mark).

5.2.2 ***Second Normal Form (2NF)**:*

☞ *A relation is in 2NF if:*

- *It is in 1NF.*
- *Every attribute that is not part of the key is fully dependent on every part of the key.*

✍ *This is the phase of identifying the keys.*

✍ *This step avoids certain redundancies.*

✍ *Every attribute must functionally depend on the entire key.*

***Example 8**: This is a counterexample of a relation in 1NF that is not in 2NF.*

***ORDER** (Date, ID-Client, ID-Product, Quantity, UPrice). It is not in 2NF because the key = (Date, ID-Client, ID-Poduct), and the unit price depends only on ID-Product.*

***Decomposition**:*

***ORDER** (date, ID-Client, ID-Product, Quantity).*

***PRODUCT** (ID-Product, UPrice).*

5.2.3 ***Third Normal Form (3NFC)**:*

☞ *A relation is in 3NF if:*

- *It is in 2NF.*

- *Every attribute that is not part of the key does not depend on a non-key attribute. This corresponds to the non-transitivity of functional dependencies, which avoids redundancies.*

☞ *In 3NF, a relation preserves functional dependencies without loss of information.*

***Example 5.3**: This is a counterexample of a relation in 2NF that is not in 3NF.*
***CAR** (registration, brand, model, power) – it is verified that it is in 2NF.*
It is not in 3NF because the key = registration, and power depends on (brand, model).

***Decomposition**:*
***CAR** (registration, brand, model)*
***MODEL** (brand, model, power)*

Example of Decomposition and Normalization of a Relation:

Let us consider a very simplified view of the information system for the Computer Science Department of Oum El Bouaghi:

- *The department welcomes students enrolled in different years of study (L1, L2, L3);*
- *For each student, we have a matriculation number, last name, first name, date of birth, address, and high school diploma series(BAC series);*
- *A student enrolled in a year is assigned to only one (tutorial/practical)group;*
- *A student is enrolled in multiple modules;*
- *A module has a responsible instructor.*
- *Each module is associated to a level.*

Questions:

1. *Define the universal relation schema.*
2. *To design the relational schema of the system: decompose the universal relation into relations under 3NF (Third Normal Form).*

Solution:

*R(**matriculation**, last-name, first-name, date-of-B, address, Sbac, groupe, level, **title-module**, name-responsable, level-module).**1NF***

*This relation is not in **2NF**.*

***Step 1:** Normalize to **2NF**.*

Step 1.1:

*R1(**matriculation**, last-name, first-name, date-of-B, address, Sbac, groupe, level, **title-module**)* $\neg 2NF$

*R2(**title-module** , name-responsable, level-module) **2NF**.*

Step 1.2:

*R1(**matriculation**, last-name, first-name, date-of-B, address, Sbac, groupe, level) **2NF**.*

*R2(**title-module** , name-responsable, level-module) **2NF**.*

*R3(**matriculation**, **title-module**) **2NF**.*

***Step 2:** Normalize to **3NF**.*

*R1(**matriculation**, last-name, first-name, date-of-B, address, Sbac, groupe, level) **3NF**.*

*R2(**title-module** , name-responsable, level-module) **3NF**.*

*R3(**matriculation**, **title-module**) **3NF**.*

***Step 3:** Renaming of relation*

*STUDENT(**matriculation**, last-name, first-name, date-of-B, address, Sbac, groupe, level) **3NF**.*

*MODULE(**title-module** , name-responsable, level-module) **3NF**.*

*ENROLLMENT(**matriculation**, **title-module**) **3NF**.*

6. Transformation of the Entity-Relationship Model to the Relational Model:

First Step: Classic Transformation Process

✍ An Entity:

An entity is represented by the relation: ***entity_name (list of entity attributes)****.*

✍ An Association:

An x,n association is represented by the relation: ***association_name (list of identifiers of participating entities, list of association attributes)****.*

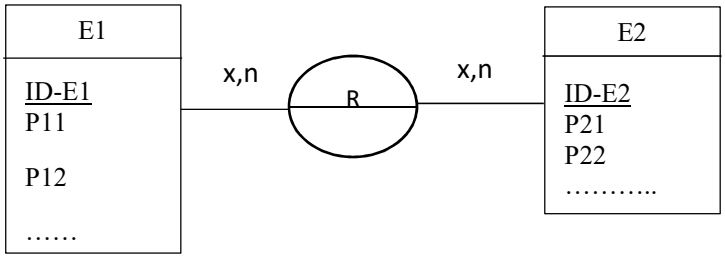
✍ Determining the Primary Key of a Relation:

In general:

☞ *The primary key of a relation formed from an entity is the identifier of that entity.*

☞ *The primary key of a relation formed from an association is the concatenation of the primary keys of the participating associated entities. However, it is important to verify whether this concatenation actually constitutes the primary key. If it does not, additional attributes from the association can be considered to determine the primary key.*

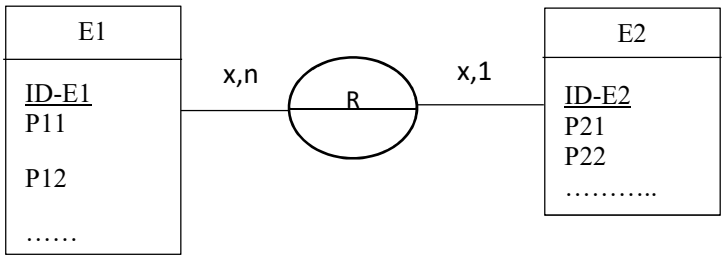
General rules of transformation:



E1(ID-E1 ,P11,P12,….)

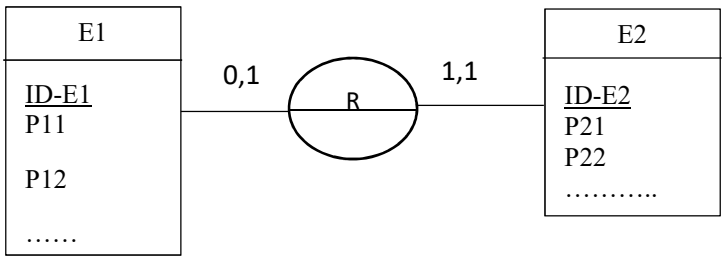
E2(ID-E2 ,P21,P22,….)

R(ID-E1, ID-E2)



E1(ID-E1 ,P11,P12,….)

E2(ID-E2 ,P21,P22,…, ID-E1)



E1(ID-E1 ,P11,P12,….)

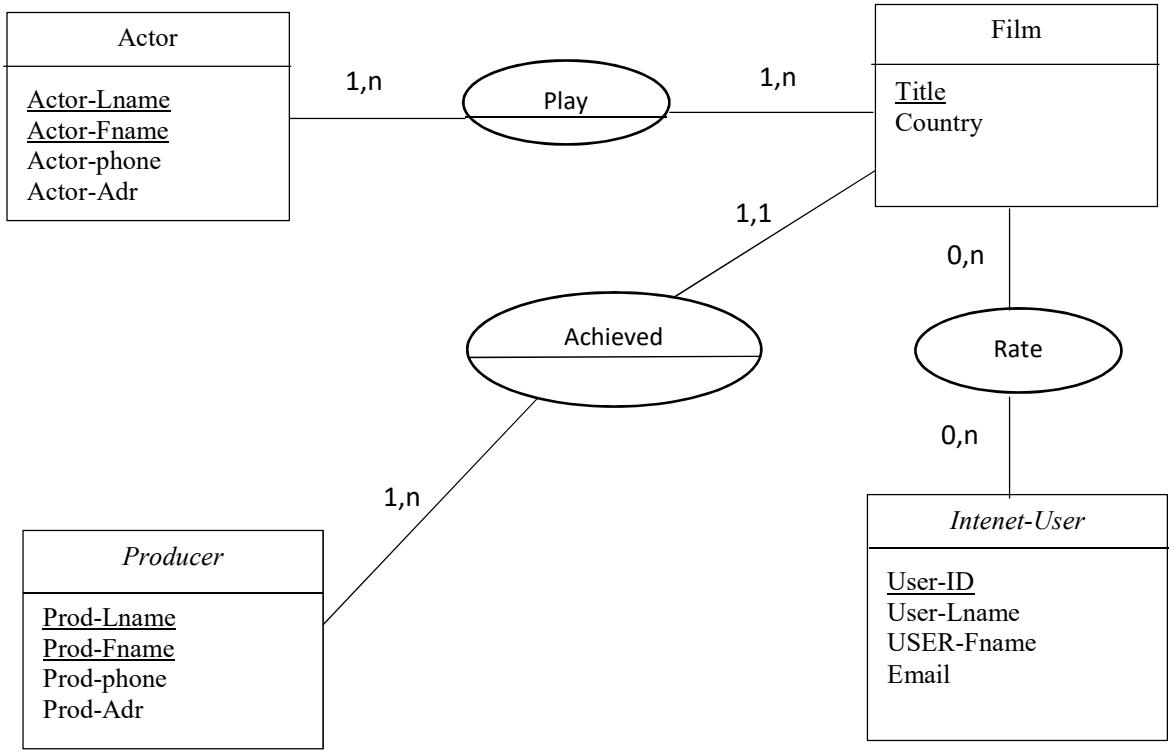
E2(ID-E2 ,P21,P22,…, ID-E1)

Second Step: Database Schema Optimization

✍ ***Eliminate information redundancy****, but be cautious of data loss when removing redundancies. To preserve all information, only remove data that can be derived through transitivity.*

✍ ***Ensure that all relations in the database schema*** *are in the Third Normal Form (3NF).*

Example of transformation of the Entity-Relationship Model to the Relational Model:



Actor(Actor-Lname, Actor-Fname, Actor-Adr, Actor-phone)3NF

Film(Title, Country, Prod-Lname, Prod- Fname) $\neg 2NF$

Play(Actor-Lname, Actor-Fname, Title,)3NF

Internet-User(User-ID, User-Lname, User-Fname, Email) 3NF

Rate(Title, User-ID) 3NF

Producer(Prod-Lname, Prod- Fname, Prod-phone, Prod-Adr) 3NF

Optimisation:

Actor(Actor-Lname, Actor-Fname, Actor-Adr, Actor-phone)3NF

Film(Title, Country)3NF

Play(Actor-Lname, Actor-Fname, Title,)3NF

Internet-User(User-ID, User-Lname, User-Fname, Email) 3NF

Rate(Title, User-ID)3NF

Producer(Prod-Lname, Prod- Fname, Prod-phone, Prod-Adr)3NF

Film(Title, Country, Prod-Lname, Prod- Fname)3NF

Achieved (Title, Prod-Lname, Prod- Fname) 3NF

7. *Database Integrity Rules:*

☞ ***Integrity rules*** *are assertions that must be verified by the data contained in a database.*

☞ *The relational model imposes the following structural constraints:*

- *Domain Integrity*
- *Key Integrity*
- *Referential Integrity*

☞ ***Automatic management of integrity constraints*** *is one of the most important tools in a database.*

7.1 Domain Integrity:

The values in a column of a relation must belong to the corresponding domain, and the following must be ensured:

- *Control of attribute values.*
- *Control between attribute values.*

7.2 Key Integrity:

☞ *The values of primary keys must be:*

- *Unique (uniqueness of key, uniqueness of n-tuples)*
- *Non-NULL*

☞ *In any possible instance of a relation, there cannot be two n-tuples with the same values for the key attributes, otherwise, two identical keys would determine two identical rows (according to the definition of a key), which is absurd (No redundancy).*

7.3 Referential Integrity:

☞ *The values of foreign keys are values of the primary key to which they reference (concept of dependent relations).*

☞ *Dependency: A primary key K1 in a relation R1 can be an attribute in another relation R2, where it acts as the primary key K2. As seen previously, K1 is called a foreign key. This concept allows us to establish dependency relationships between tables.* ***Referential constraints*** *have an important impact on operations applied to the database, as they help avoid anomalies.*

8. Conclusion:

The relational model has several advantages:

✍ ***Simplicity of presentation****: Data is presented in the form of tables.*

✍ ***Relational operations can be applied to the database****:*

- *Relational algebra.*
- *Assertion-based languages: Languages used for managing and querying the database, such as SQL.*

✍ ***Physical Independence**:*

- *Optimization of access during database management.*
- *Access strategy determined by the system.*

✍ ***Logical Independence**: Concept of a single view by different software.*

✍ ***Simple Integrity Control**: Integrity constraints defined at the schema level of the database.*