Relational Databases

Dr. Rabah Mokhtari

Department of Computer Science,
UNIVERSITY OF M'SILA

October 30, 2022

Content

Introduction

Relational DB Model

The relational data model was first introduced by Ted Codd of IBM Research in 1970 in a paper titled, ***A Relational Model of Data for Large Shared Data Banks***, and it attracted immediate attention due to its simplicity and mathematical foundation.

simplicity of a mathematically founded model

1. The model uses the concept of a mathematical relation and has its theoretical basis in **set theory** and **first-order predicate logic**.
2. **Simplicity**: The concept of tables with rows and columns is extremely simple and easy to understand.
3. **Data independence**: Data independence is ability to modify data structure (in this, case, tables) without affecting existing programs.
4. **Declarative data access**: The relational model introduced a declarative language called Structured Query Language (SQL), also known as “sequel”, to simplify data access and manipulation.

Database Example

KnowWare Inc. Database

Almost illustration examples used in the rest of this presentation are taken from an example of database of a manufacturing company called **KnowWare Inc.** inspired by Date, C. in his book titled *An Introduction to Database Systems.* (2003) AW, 8th edition.

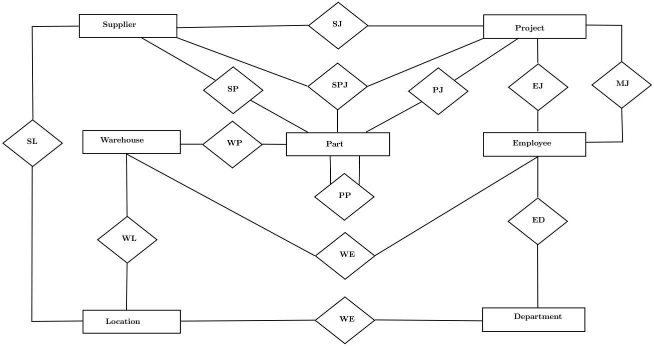

Figure: KnowWare Inc. Database

Formalization

Relation scheme, Relation and Tuple

Relation scheme, Attribute, and Domain

1. A *relation scheme* **R** is a finite set of ***attribute names*** $\{A_1, A_2, ..., A_n\}$.
2. Corresponding to each attribute name A_i is a set $D_i, 1 \leqslant i \leqslant n$, called the ***domain*** of A_i.
3. We also donate the domain of A_i by $dom(A_i)$. Attribute names are sometimes called simply *attributes*. The domains are non-empty set, finite, or countably infinite. Let $D = D_1 \cup D_2 \cup ... \cup D_n$.

Relation and Tuple

- A *relation* (or **relation state**) r on relation scheme R, denoted by $r(R)$, is a finite set of mappings (or **tuples**) $\{t_1, t_2, ..., t_p\}$ from R to D.
- Each mapping (or **tuple**) $t \in r, t(A_i)$ must be in $D_i, 1 \leqslant i \leqslant n$.
- The mappings are called ***n-tuples***.

Examples

Supplier Relation scheme

Considering the relation scheme "**Supplier**" of KnowWare Inc. company database. We can write :

1. $Supplier = \{Sno, Sname, Status, City\}$. Or
2. $Supplier(Sno, Sname, Status, City)$.

Table: Relation of five suppliers

Sno	Sname	Status	City
1	Smith	20	London
2	Jones	10	Paris
3	Blake	30	Paris
4	Clark	20	London
5	Adams	30	NULL

Examples

5 suppliers (or tuples)

Last table shows a relation of **five tuples** (or rows) where each row presents one supplier.
The first two tuples $t1$ and $t2$ may be written as

1. $t1 =< 1, Smith, 20, London >$
2. $t2 =< 2, Jones, 10, Paris >$.

NULL values in tuples

An important concept is that of **NULL** values, which are used to represent the values of attributes that may be unknown or may not apply to a tuple. A special value, called NULL, is used in these cases. (see the **City** of the supplier Adams in the Supplier relation.

Sno	Sname	Status	City
..	..	..	..
5	Adams	30	**NULL**

Examples

FLIGHT relation scheme

Every flight listed in the airline schedule table has an origin and a destination and it is scheduled to depart at a specific time and arrive at a later time. We can write

1. FLIGHTS = {NUMBER, FROM, TO, DEPARTS, ARRIVES }.
2. FLIGHTS(NUMBER, FROM, TO, DEPARTS, ARRIVES).

Table: FLIGHTS - Airline schedule.

NUMBER	FROM	TO	DEPARTS	ARRIVES
83	JFK	O'Hare	11:30a	1:43p
84	O'Hare	JFK	3:00p	5:55p
109	JFK	Los Angeles	9:50p	2:52a
213	JFK	Boston	11:43a	12:45p
214	Boston	JFK	2:20p	3:12p

Key and superkey (SK)

Definition

A *key* of relation r on relation scheme R is a subset $K = \{B_1, B_2, ..., B_n\}$ of R with the following property.

1. For any two distinct tuples t_1 and t_2 in r, $t_1(K) \neq t_2(K)$, and
2. No proper subset K' of K shares this property.

If r has key K', and $K' \subseteq K$, then K is also a key of r. SK is called a *superkey* if SK contains a key of r.

Example 1

In the FLIGHTS relation shown above, $\{NUMBER\}$ is a key (and a superkey), so $\{NUMBER, FROM\}$ is a superkey but not a key.

Example 2

$\{Sno\}$ is a key of the SUPPLIER relation.

Integrity Constraints (ICs)

Integrity Constraints

Definition

- An **integrity constraint** is a boolean expression that is associated with some database and is required to evaluate at all time to TRUE.
- A DBMS should provide capabilities for defining and enforcing these constraints.

IC Types

Integrity constraints can generally be divided into two main categories:

1. Constraints that can be formally declared in the database scheme and the DBMS must then enforce them.
2. Constraints that cannot be directly expressed in the database scheme. These types of constraints are not understood by the DBMS but they specify what the data means to the **users**.

KnowWare Inc. database Integrity Constraints

Example of ICs on KnowWare Inc. database

Here are some integrity constraints, expressed in natural language, all based on the **KnowWare Inc.** database.

1. Every supplier status value is in the range 1 to 100. (***Category 1***)
2. Every supplier in London has status 20. (***Category 2***)
3. No two distinct suppliers have the same number. (***Category 1***)
4. No supplier with status less then 20 supplies any part in a quantity greater than 500. (***Category 2***)

Important Integrity Constraints

Domain Constraint

- The domain constraint involves specifying a data type for each data item.
- This kind of constraint is implicitly guaranteed by the DBMS.

Key and Primary Key Constraints

- A superkey *SK* specifies a *uniqueness constraint* that no two distinct tuples, in a relation **r**, t_1 and t_2 can have the same value for *SK*, $t_1(SK) \neq t_2(SK)$.
- If this *SK* is a *key*, it specifies a particular uniqueness constraint called ***primary key constraint***.

Referencial and Foreign Key Constraints

- A referential constraint involves specifying that each record in a file must be related to one record in other file.
- A ***foreign key*** is a referencial key that allows to join two tables together by using a ***primary key*** in one table with a ***non key field*** in another table.

Primary and Foreign Key Constraints Examples

Supplier, Part, and SP

- Supplier(**Sno**, Sname, Status, City).
- Part(**Pno**, Name, Color, Weight, City).
- SP(**#Sno**, **#Pno**, QTY).

SP

Sno	Pno	QTY
1	1	300
1	2	200
1	3	400
1	4	200
1	5	100
1	6	100
2	1	300
2	2	400
2	3	500
3	2	200
3	3	200
3	4	300
4	2	300
4	3	400

Supplier

Sno	Sname	Status	City
1	Smith	20	London
2	Jones	10	Paris
3	Black	30	Paris
4	Clak	20	London
5	Adams	30	Athens

Part

Pno	Name	Color	Weight	City
1	Nut	Red	12.0	London
2	Bolt	Green	17.0	Paris
3	Screw	Blue	17.0	Oslo
4	Screw	Red	14.0	London
5	Cam	Blue	12.0	Paris
6	Cog	Red	19.0	London

Figure: Supplier, Part and SP relations

Realational DataBase Scheme and Database

Realational Database Scheme

A relational database scheme **S** is a set of relation schemes $S = \{R_1, R_2, ..., R_m\}$ and a set of integrity constraints **IC**s.

Relational Database

A relational database state **DB** of **S** is a set of relation states $\mathbf{DB} = \{r_1, r_2, ..., r_m\}$ such that each r_i is a state of R_i and such that the r_i relation states satisfy the specified **IC**s.

Functional Dependencies and Normalization

DFs and Normalization

Definition

- Let X and Y be subsets of the relation scheme R;
- then the functional dependency (FD) **X** $\rightarrow$ **Y** holds in R if and only if, whenever two tuples of R agree on X, they also agree on Y.
- X and Y are the **determinant** and the **dependant**, respectively, and the FD overall can be read as either "**X** functionally determines **Y**" or "**Y** is functionally dependent on **X**", or more simply just "**X** arrow **Y**"

DFs and Normalization

Example

The relation shown in the following table satisfies the FD $\{SNO\#\} \rightarrow \{CITY\}$.

Table: Sample values for relation variable SCP

SNO#	CITY	PNO#	QTY
S1	London	P1	90
S1	London	P2	100
S2	Paris	P1	200
S2	Paris	P2	200
S3	Paris	P2	300
S4	London	P2	400
S4	London	P4	400
S4	London	P5	400

Closure of a set of functional dependencies

Closure of set of FDs and Amstrongs axioms

Definition. Formally, the set of all dependencies that include F as well as all dependencies that can be inferred from F is called the **closure** of F. It is denoted by F^+ [8].

To compute F^+ from F, Amstrong [1] gave a set of **inference rules** (more usually called **Amstrong's axioms**) by which new FDs can be inferred from given ones [4].

Let A, B, and C be arbitrary subsets of the set of attributes of the given scheme R, and let us agree to write AB to mean the union of A and B. Then:

1. **Reflexivity:** if B is a subset of A, then $A \to B$.
2. **Augmentation:** if $A \to B$, then $AC \to B$.
3. **Transitivity:** if $A \to B$ and $B \to C$, then $A \to C$.

Several further rules can be derived from the three given above, the following among them. (Let D is another arbitrary subset of the set of attributes of R)

1. **Self-determination:** $A \to A$.

Closure of a set of functional dependencies

Definition and Amstrong's axioms

2. **Decomposition:** if $A \rightarrow BC$, then $A \rightarrow B$ and $A \rightarrow C$.
3. **Union:** if $A \rightarrow B$ and $A \rightarrow C$, then $A \rightarrow BC$.
4. **Composition:** if $A \rightarrow B$ and $C \rightarrow D$, then $AC \rightarrow BD$.

Closure of a set of functional dependencies

Amstrongs axioms

Example

The following set of FDs F is specified on the relation scheme "EMP-DEPT" in Figure 9 [8].

$$F = \{Ssn \rightarrow \{Ename, Bdate, Address, Dnumber\}, Dnumber \rightarrow \{Dname, Dmgr\_ssn\}\}$$

Some of the additional functional dependencies that we can infer from F are the following:

$Ssn \rightarrow \{Dname, Dmgr\_ssn\}$

$Ssn \rightarrow Ssn$

$Dnumber \rightarrow Dname$

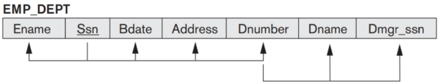

Figure 9: Relation scheme EMP-DEPT

Closure of a set of functional dependencies

Closure of set of FDs with Amstrong's axioms

Exercise2.1

Suppose we are given shceme $R(ABCDEF)$ and the FDs:

A $\rightarrow$ **BC**

B $\rightarrow$ **E**

CD $\rightarrow$ **EF**

Prove that the FD **AD** $\rightarrow$ **F** $\in$**F**$^+$ (meaning that we can infer **AD** $\rightarrow$ **F** from **F**).

Closure of a set of attributes X under a set of FDs F

Definition

Let F a set of FDs over scheme R. If an FD $X \rightarrow Y$ can be *implied* by F, we write $F \models X \rightarrow Y$. To determine if $F \models X \rightarrow Y$, we need only test if $X \rightarrow Y \in F^+$ [10]. Because F^+ can be considerably large than F, we would like to find a means to test if $X \rightarrow Y \in F^+$. The solution is to compute the **closure** X^+ of the set of attributes X under F and test if Y is in X^+. Formally, we write $X \rightarrow Y \in F^+$ if and only if $Y \subseteq X^+$.

Closure of a set of attributes X under a set of FDs F

Algorithm

Algorithm 1 Compute the closure X^+ under the set of FDs F

INPUT: A set of attributes X and a set of FDs F
OUTPUT: X^+ the closure of X under F

```
function CLOSURE(X,F)
    X⁺ ← X;
    oldX⁺ ← ∅;
    while (oldX⁺ ≠ X⁺) do
        oldX⁺ ← X⁺;
        for every FD Y → Z in F do
            if X⁺ ⊇ Y then
                X⁺ ← X⁺ ∪ Z;
            end if
        end for
    end while
    return X⁺;
end function
```

Closure of a set of attributes X under a set of FDs F

Example

Example

Let $F = \{A \rightarrow D, AB \rightarrow E, BI \rightarrow E, CD \rightarrow I, E \rightarrow C\}$. $CLOSUE(F, \{A, E\}) = \{A, E\}^+ = \{A, C, D, E, I\}$.

Exercise

Exercise

Let F and G two sets of FDs,

- $F = \{A \rightarrow BC, E \rightarrow CF, B \rightarrow E, CD \rightarrow EF\}$.
- $G = \{A \rightarrow B, C \rightarrow DE, AC \rightarrow F\}$.

– Compute the closure $\{A, B\}^+$ under F and $\{A, C\}^+$ under G.

Cover and equivalence

Definition

Let F and G two sets of FDs over scheme R. If every FD $X \rightarrow Y \in G$ is *implied* by F, we say that G is implied by F and we write $F \models G$.

Definition of cover. Let F and G two sets of FDs. If every FD implied by F is implied by G –i.e., if F^+ is a subset of G^+– we say that G is a *cover* for F [4].

Definition of equivalence. Two sets of FDs F and G over scheme R are *equivalent*, written $F \equiv G$, if and only if $F^+ = G^+$. If $F \equiv G$, then F is a cover for G [10].

Lemma Given sets of FDs over scheme R, $F \equiv G$ if and only if $F \models G$ and $G \models F$

Cover and equivalence

Exercise

Exercise2.2

F and G two sets of FDs, $F = \{A \rightarrow BC, A \rightarrow D, CD \rightarrow E\}$ and $G = \{A \rightarrow BCE, A \rightarrow ABD, CD \rightarrow E\}$

Are F and G equivalent?

Minimal Cover

Definition

A set F of FDs is **irreducible**, called also **minimal cover**, if only if it satisfies the following three properties:

1. The right-hand side (the **dependent**) of every FD in F involves just one attribute (i.e., it is a singleton set).
2. The left-hand side (the **determinant**) of every FD in F is irreducible in turn, meaning that no attribute can be discarded from the determinant without changing the closure F^+ (i.e., without converting F into some set not **equivalent** to F.
3. No FD in F can be discarded from F without changing the closure F^+ (i.e., without converting F into some set not equivalent to F).

Minimal Cover

Algorithm

Algorithm 3 Find the minimal cover F of the set of FDs E

INPUT: A set of FDs E.

OUTPUT: The minimal cover F of E.

```
F ← E;
Replace each FD X → {A1, A2, ..., An} in F by the n FDs X → A1, X → A2,
X → An;
for each FD X → A in F do
    for each attribute B in X do
        if {{F − {X → A}} ∪ {(X − {B}) → A}} is equivalent to F then
            replace X → A with (X − {B}) in F;
        end if
    end for
end for
for each remaining FD X → A in F do
    if {{F − {X → A}} is equivalent to F then
        remove X → A from F;
    end if
end for
```

Minimal Cover

Definition

Exercise

Compute the minimal cover of the set of FDs E on the scheme R(ABCD). $E = \{A \to BC, B \to C, A \to B, AB \to C, AC \to D\}$.

Answer of exercise

1. Step 1,

a) $A \to C$

b) $A \to B$ (removed because it occurs twice)

c) $B \to C$

d) $A \to B$

e) $AB \to C$

f) $AC \to D$

2. Step 2,

$AC \to D$ (**f**) is replaced by $A \to D$ because:

- $A \to AC$ is implied by composition of $A \to A$ and $A \to C$ (**a**).
- Then, $A \to D$ is implied by transitivity of $A \to AC$ and $AC \to D$ (**f**).

3. Step 3,

$AB \to C$ (**e**) is removed because

- We can imply $AB \to CB$ by composition of $A \to C$ (given in **a**) and $B \to B$.
- Then, $AB \to C$ is implied by decomposition of $AB \to CB$.

4. Step 4,

$A \to C$ is removed because it is implied by transitivity of $A \to B$ (**d**) and $B \to C$ (**c**)

Minimal Cover

Algorithm

Algorithm 3 Find the minimal cover F of the set of FDs E

INPUT: A set of FDs E.

OUTPUT: The minimal cover F of E.

$F \leftarrow E$;
Replace each FD $X \rightarrow \{A_1, A_2, ..., A_n\}$ in F by the n FDs $X \rightarrow A_1$, $X \rightarrow A_2$, $X \rightarrow A_n$;
for each FD $X \rightarrow A$ in F **do**
    for each attribute B in X **do**
        if $\{\{F - \{X \rightarrow A\}\} \cup \{(X - \{B\}) \rightarrow A\}\}$ is equivalent to F **then**
            replace $X \rightarrow A$ with $(X - \{B\})$ in F;
        end if
    end for
end for
for each remaining FD $X \rightarrow A$ in F **do**
    if $\{\{F - \{X \rightarrow A\}\}$ is equivalent to F **then**
        remove $X \rightarrow A$ from F;
    end if
end for

Normalization

Definition

A normal form is a restriction on the database scheme that presumably precludes certain undesirable properties from the database [10].

First Normal Form (1NF)

Let relation scheme $R(A_1A_2...A_n)$. A relation $r(R)$ is in *first normal form* (lNF) if and only if for all tuple t appearing in r, the value of each attribute A_i of type $dom(A_i)$ is atomic. To say it in different words, 1NF means that every tuple contains exactly one value for each attribute [4].

Normalization

1NF - Normalization

Example

NAME	SEXE
{John, Jean, Ivan}	Male
{Mary, Marie}	Female

Table 4: *gender* relation not in 1FN

NAME	SEXE
John	Male
Jean	Male
Ivan	Male
Mary	Female
Marie	Femal

Table 5: *gender* relation in 1FN

Normalization

Second Normal Form (2NF)

- **Definition :** A scheme R is in **second normal form** (2NF) if and only if, for every key K of R and every nonkey attribute A of R, the FD $K \rightarrow \{A\}$ (which holds in R, necessarily) is irreducible [5].

Example

SNO#	CITY	PNO#	QTY
S1	London	P1	90
S1	London	P2	100
S2	Paris	P1	200
S2	Paris	P2	200
S3	Paris	P2	300
S4	London	P2	400
S4	London	P4	400
S4	London	P5	400

The scheme SCP isn't in 2NF because its key is $\{SNO, PNO\}$ and the FD $\{SNO, PNO\} \rightarrow CITY$ isn't irreducible

Normalization

2NF - Normalization

SNO#	PNO#	QTY
S1	P1	90
S1	P2	100
S2	P1	200
S2	P2	200
S3	P2	300
S4	P2	400
S4	P4	400
S4	P5	400

SNO#	CITY
S1	London
S2	Paris
S3	Paris
S4	London

Normalization

3NF

Third Normal Form (3NF)

A relation that is in First and Second Normal Form and in which no non-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3NF).

Example

The scheme Supplier isn't in 3NF.

Supplier

SNO	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	30	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Normalization

3NF - Normalization

SNC

SNO	SNAME	CITY
S1	Smith	London
S2	Jones	Paris
S3	Blake	Paris
S4	Clark	London
S5	Adams	Athens

CT

CITY	STATUS
Athens	30
London	20
Paris	30