

# **Introduction to Database Systems**

## **Relational Database Design**

**Vijay Kumar**  
**SICE Computer Networking**  
**University of Missouri-Kansas City**

## Relational Database Design

A relational database is a set of good relational schema. Characteristics of a good relational schema

- Should contain, as far as possible, attributes of only the entity it represents.

**Example:** A bad relational schema. Interest rate attribute does not fit semantically.

### Department (bad schema)

Dnumber	Dlocation	Dname	Interest rate
---------	-----------	-------	---------------

- Should require minimum storage space and have minimum data redundancy.

**Example:** Suppose it takes one word to store the value of an attribute.

### Sport (good schema)

Sname	Inst_id	Inst_name	Expertise	Fee
Football	1	Tom	Football	200
Tennis	1	Tom	Tennis	200
Baseball	2	Peter	Baseball	300
Golf	2	Peter	Golf	300

This relation will require 20 words. We split this relation into **S1** and **S2** relations

#### S1

Sname	Instructor-id
Football	1
Tennis	1
Baseball	2
Golf	2

#### S2

Sname	Instructor-id	Expertise 1	Expertise 2	Fee
Tom	1	Football	Tennis	200
Peter	2	Baseball	Golf	300

This arrangement requires 18 words to store the same information. The amount of saving increases with database size. A smaller storage space also improves searching.

These are a few simple examples to illustrate the importance of good schema. Our aim is then to design good schema.

## Database Processing

Consider the following Student relation. The relation indicates that students with stu\_id take up some Activities and pay certain amount of fee for learning the activity. A student with any id can take any number of Activities provided they pay the fee. Assume

that this relation is stored in the database. We want to see what problems occur if this relation is modified.

### Student

Stu_id	Activity	Fee
100	Skiing	200
100	Golf	65
150	Swimming	50
175	Squash	50
175	Swimming	50
200	Swimming	50
200	Golf	65

**Modification:** Suppose Student 100 gave up skiing. The first record must be deleted from the database.

**Effect:** Database does not have information about skiing fee. More information is deleted than intended. This is not a good relation.

**Modification:** Suppose you want to add Baseball in the database and you are going to charge 200.

**Effect:** Cannot add this information since there is no student. We cannot add a fact about one entity until we have an additional fact about another entity. To add this record you must have a student willing to learn baseball. You are not allowed to insert null value in Stu\_id attribute. This is not a good relation.

These are called **Modification Anomalies**. We will study them in detail. First we define some useful terms

### Functional dependencies (FD)

A functional dependency is a dependency relationship between two sets of attributes. Formally, let  $R = \{A_1, A_2, \dots, A_n\}$ ,  $X \in R$  and  $Y \in R$ , where  $R$  is a relational schema over attributes  $A_i$ 's and  $X$  and  $Y$  are attribute sets.

Suppose the value of  $X$  determines the value of  $Y$ , that is, if we assign a value to  $X$  then  $Y$  will automatically acquire some value or the value of  $X$  will determine the value of  $Y$ . Thus, under this constraint, for two tuples  $t_1$  and  $t_2$  in  $r(R)$ , if  $t_1(X) = t_2(X)$  then we must have  $t_1(Y) = t_2(Y)$ . This means that the  $Y$  value of a tuple in  $r(R)$  is determined by the  $X$  value of that tuple in  $r(R)$ . This type of relationship is defined as Functional Dependency (FD).

**Notation:**  $X \rightarrow Y$ .  $X$  functionally determines  $Y$ .  $X$ : Left side of FD.  $Y$ : Right side of FD.  $Y \not\rightarrow X$ .  $Y$  does not functionally determine  $X$ , i.e.,  $X$  is not functionally dependent on  $Y$ .

**Full functional dependency:**  $X \rightarrow Y$  is a full functional dependency if all members of  $X$  must be present to hold the dependency. Suppose

$X = \{A, B, C\}$  and  $Y = \{D\}$ . If  $\{A, B\} \rightarrow \{D\}$  then  $Y$  is partially dependent on  $X$  because to know the value of  $Y$  only two attributes of  $X$  is required.

**Prime attribute:** An attribute  $\in$  primary key set. If primary key = {A, B, C}, then attributes A, B, and C are prime attributes.

**Non-prime attribute:** Attribute  $\notin$  primary key set. If R(A, B, C, D) and primary key = {A, B, C}, then attribute D is a non-prime attribute.

There is no formula to establish FD. The semantics of a relation should indicate how attributes of a relation are related.

### Examples

Suppose we create relation **Sport\_Fee**. It holds information about fee charged for each sport. The semantics of this relation indicates that the value of the attribute **Activity** determines the value of the **fee**. This means wherever Golf appears, its fee will be 65. From this semantics we can conclude that Activity  $\rightarrow$  Fee. We can also see that Fee does not determine Activity (Fee  $\rightarrow$  Activity), i.e., Fee cannot be the left side of FD. In Sport\_Fee fee value of 50 is associated with two different activities.

#### Sport\_Fee

Activity	Fee
Skiing	200
Golf	65
Squash	50
Swimming	50

Consider the following **Parts** relation.

The **Primary key** = {SSN, Pnumber}.

The FDs are: Pnumber  $\rightarrow$  {Plocation, Pname} and Plocation  $\rightarrow$  Pnumber

#### Parts

SSN	Pnumber	Plocation	Pname
1234	1	Kansas City	Car
1234	2	Mission	Rocket
1235	1	Kansas City	Car
1236	3	Wichita	DC 10
1237	1	Kansas City	Car
1238	5	Kansas City	Truck

### Graphical Representation of FDs using arrow

A FD is represented with an arrow. The head of the arrow points to the right side of FDs and the tail points to the left side of FDs. Thus, the tail is on the determinant side and the head is on the side of what is determined.

Consider the following **Emp\_Dept** relational schema.

#### Emp\_Dept

Ename	SSN	Bdate	Address	Dnumber	Dname	Dmgrssn
-------	-----	-------	---------	---------	-------	---------

Suppose the database designer has defined the following functional dependencies for the above schema.

**SSN** → {**Ename**, **Bdate**, **Address**, **Dnumber**}

**Dnumber** → {**Dname**, **Dmgrssn**}

These FDs are represented as follows using FD diagram.

### Emp\_Dept

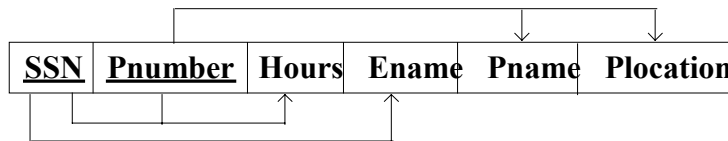


Similarly FDs in **Emp\_Proj** schema are represented as follows. Note that one of the prime attributes (e.g., Pnumber) can also be a determinant.

{**SSN**, **Pnumber**} → **Hours**. **SSN** → **Ename**

**Pnumber** → {**Pname**, **Plocation**}

### Emp\_Proj



**IMPORTANT:** One must remember that FDs in a relation are defined by the database designer. In the above examples these FDs may not be valid if they have not been defined.

## Normal Forms (NF) and Modification Anomalies

### Normal Forms (NF)

Normal forms are improved relations, which has minimum modification anomalies. Thus, a relation in a lower normal form (e.g., 1NF) may have more modification anomalies than a relation in 2NF, a relation in 2NF has more modification anomalies than a relation in 3NF, and so on. There are First normal form (1NF), Second normal form (2NF), Third normal form (3NF), Boyce-Codd normal form (BCNF), Fourth normal form (4NF), Domain/Key normal form (DK/NF) and Fifth normal form (5NF). We will study only 1NF through 4NF.

**Normalization:** The process of transforming a relation from a lower NF to upper NF.

### 1NF

*A relation is in 1NF if it does not contain repeating groups, i.e., all its attributes are atomic.*

To understand 1NF we define Repeating groups. In **Order** relation, for some **Ono** value, there are more than one **Pno** values (**BT04** and **BZ66**). Thus **Pno** is a **repeating group**. This is not allowed in relational model where every tuple must have values (including NULL) in all its attributes. Thus, for the 3rd and 6<sup>th</sup> tuples in **Order**, **Ono** and **Date** must have some appropriate values. If suitable values are inserted, then the repeating groups are eliminated. This is called **normalization**. To normalize **Order** copy the values of all other attributes from the previous tuple. Thus, the 5<sup>th</sup> tuple has no

value for Ono and Date attributes. These copies are copied from the previous tuple. This process is repeated until every tuple has values for all its attributes.

### Non-normalized: Order

<u>Ono</u>	<u>Date</u>	<u>Pno</u>	<u>No_Ordered</u>
12489	90287	AX12	11
12491	90287	BT04	1
12494	90487	BZ66	1
12495	90487	CB03	4
		CX11	2
12498	90587	AZ52	2
		BA74	4
12500	90587	BT04	1

### Normalized: Order

<u>Ono</u>	<u>Date</u>	<u>Pno</u>	<u>No_Ordered</u>
12489	90287	AX12	11
12491	90287	BT04	1
12494	90487	BZ66	1
12495	90487	CB03	4
1245	90487	CX11	2
12498	90587	AZ52	2
1248	90587	BA74	4
12500	90587	BT04	1

Normalized **Order** relation is in 1NF. The primary key to the not-normalized relation **Order** was **Ono** alone. The primary key to the normalized **Order** is now the combination of **Ono** and **Pno**. In general it is true that the primary key will expand in the transformation of not-normalize relation to 1NF. It will typically include the original primary key concatenated with the key to the repeating group.

### Modification Anomalies

Consider the following **Order** relation. It is in 1NF.

#### Order

<u>Ono</u>	<u>Date</u>	<u>Part_descrip</u>	<u>Pno</u>	<u>No_Ordered</u>
12489	90287	Iron	AX12	11
12491	90287	Stove	BT04	1
12491	90287	Washer	BZ66	1
12494	90487	Bike	CB03	4
12495	90487	Mixer	CX11	2
12498	90587	Skates	AZ52	2
12498	90587	Baseball	BA74	4
12500	90587	Stove	BT04	1

Relation **Order** has the following modification anomalies.

**Update** A change to the description of BT04 requires several changes since BT04 has been duplicated as a result of normalization.

**Inconsistent data:** In the absence of incomplete update, BT04 may have different values in other attributes.

**Additions:** We cannot add ZZ14 until we have an order for ZZ14.

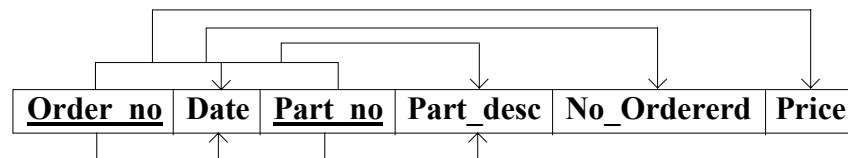
**Deletion:** By deleting BT04 we lose that BT04 represents Stove.

**Conclusion:** Order relation needs some improvement to minimize anomalies. It must be normalized to 2NF.

## 2NF

*A relation is in 2NF if it is in 1NF and non-prime attribute is fully functionally dependent on the primary key.*

### Dependency diagram of Order



**Primary key:** Order\_no, Part\_no.

Relation **Order** is in 1NF: All attributes are atomic.

Relation **Order** is not in 2NF, because non-key attribute **Part\_desc** is dependent upon **Part\_no** which is a portion of the primary key. Also **Date** is dependent upon **Order\_no** (part of the primary key). This type of dependency is often termed as **Partial Dependency**

**1NF to 2NF:** Split the relation into two or more via projection as follows.

### Steps

- Identify the set of attributes that makes up the primary key, i.e., {**Order\_no**, **Part\_no**}.
- Create all subsets of the above set, i.e., {**Order\_no**}, {**Part\_no**} and {**Order\_no and Part\_no**}.
- Designate each of these subset as the primary key of a relation that contains those attributes which are dependent on these primary keys:

**Primary Keys:** {Order\_no}, {Part\_no} and {Order\_no, Part\_no}

### New relational schemas

#### Order (Order\_no, Date)

Date is functionally dependent on **Order\_no** (see diagram).

#### Part (Part\_no, Part\_desc)

Part\_desc is functionally dependent on **Part\_no**.

#### Order\_line (Order\_no, Part\_no, No\_ordered, Price)

Occurrences of **Order Part\_desc**, and **Order\_line**

Order		Part		Order_line			
Order_no	Date	Part_no	Part_desc	Part_no	Order_no	No_ordered	Price
12489	90287	AX12	Iron	AX12	12489	11	14.95
12491	90287	AZ52	Skates	BT04	12491	1	402.99
12494	90487	BA74	Baseball	BZ66	12491	1	311.95
12495	90487	BH22	Toaster	CB03	12494	4	175.00
12498	90587	BT04	Stove	CX11	12495	2	57.95
12500	90587	BZ66	Washer	AZ52	12498	2	22.95
		CA14	Skillet	BA74	12498	4	4.95
		CB03	Bike	BT04	12500	1	402.99
		CX11	Mixer				

All these relations are in 2NF. Anomalies have been eliminated. To verify this we perform the same set of operations, which resulted in anomalies in the parent relation **Order**.

- Change:** Change to BT04 requires only one change in **Part** relation.
- Addition:** Adding a new part and its description do not need an order to exist.
- Deletion:** Deleting order 12489 does not cause AX12 to be deleted from Part.
- Information loss:** none.

**Q.** Does this imply that relations in 2NF do not have modification anomalies?

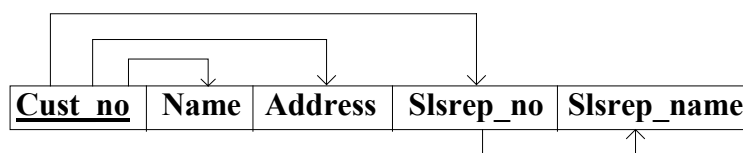
**A.** No. Relations in 2NF may suffer with all modification anomalies.

### Example

#### Customer

Cust_no	Name	Address	Slsrep_no	Slsrep_name
124	Sally A	4747 Troost	2	Tom J
256	Ann S	215 Oak	6	Bill S
311	Don C	48 College	12	Sam B
315	Tom D	914 Cherry	6	Bill S
405	Al W	519 Watson	12	Sam B
412	Sally A	16 Elm	3	Mary J
522	Mary N	108 Pine	12	Sam B
567	Joe B	808 Ridge	6	Bill S
587	Judy R	512 Pine	6	Bill S
622	Dan M	419 Chip	3	Mary J

The dependency diagram of **Customer**



**Customer** is in 2NF. It suffers with all the anomalies

**Update:** A change to **Slsrep\_name** requires multiple changes.

**Inconsistent data:** There is nothing in the design that would prohibit a **Slsrep\_name** from having two different names.

**Additions:** To add **Slsrep\_no 47**, there must be a customer for **47** first

**Deletions:** If we delete all the customers of a sales rep then we lose the name of the **Sales rep** also

### Reason for these anomalies and remedy

**Slsrep\_no** which is not a primary key, determines **Slsrep\_name**. As a result **Slsrep\_no** can appear many times in the relation. **Cust\_no** determines **Slsrep\_name** through **Slsrep\_no**. This is a transitive dependency. A transitive dependency exists among three or more attributes. For example, suppose there are three attributes X (primary key), Y, and Z. If  $X \rightarrow Y$  and  $Y \rightarrow Z$  then  $X \rightarrow Z$ . So Z is transitively dependent on X. To minimize anomalies normalize **Customer** to 3NF by removing transitive dependency.

### 3NF

*A relation  $r(R)$  is in 3NF if it is in 2NF and no non-prime attribute of  $r$  is transitively dependent on the primary key.*

**Customer** relation is further normalized to the following relation using the method described earlier.

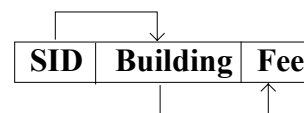
**Customer\_Profile:** Cust\_no Name Address Slsrep\_no

**Srep\_Profile:** Slsrep\_no Slsrep\_name

### Another example

#### Housing

SID	Building	Fee
100	Randolf	1200
150	Ingersol	1100
200	Randolf	1200
250	Pitkin	1100
300	Randolf	1200



In Housing relation which is in 2NF, Fee is transitively dependent on SID. This relation has all modification anomalies.

This relation can be converted to 3NF by normalization process. We convert Housing into **Housing** and **Fee** relations.

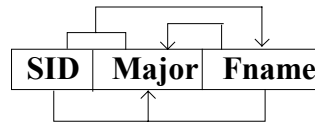
**Housing:** SID Building

**Fee:** Building Fee

3NF also have modification anomalies. Consider the following **Advisor** relation and its occurrence

### Advisor

SID	Major	Fname
100	Math	Cauchy
150	Psychology	Jung
200	Math	Riemann
250	Math	Cauchy
300	Psychology	Perl
300	Math	Riemann



### Relationships

- A student can have one or more majors.
- A **Major** can have several faculties as advisors.
- A faculty member advises in only one **Major** area.
- **SID** cannot be a key since a student can have many **Majors** and therefore many advisors.
- A student cannot have many advisors in the same area.

**Keys are:** (SID, Major) → Fname.

(SID, Fname) → Major

**Determinant:** Fname → Major

This relation does not have transitive dependency. One of these sets can be selected as a primary key. Advisor is in 3NF since there is no transitive dependency but it has modification anomalies.

**Deletion** Delete SID 300, we lose the fact that Perls advises in Psychology.

**Addition:** Cannot add the fact that Keynes advises in Economics if there is no student enrolled.

**Update:** If Cauchy advises in Physics then multiple changes are required.

**Inconsistency:** Any change in Cauchy-Math will make Advisor inconsistent.

**Solution:** Further normalization to Boyce/Codd Normal form.

### Boyce/Codd Normal Form (BCNF)

*A relation is in BCNF if every determinant is a candidate key*

**Advisor** is not in BCNF because **Fname → Major**, and **Fname** is not a candidate key. So **Advisor** is normalized to BCNF and the resultant relations are:

**Advisor:** SID Fname

**Major:** Major Fname

Relations in BCNF may not entirely free from anomalies. Consider the following **Student** relation:

#### Student

SID	Major	Activity
100	Music	Swimming
100	Accounting	Swimming
100	Music	Tennis
100	Accounting	Tennis
150	Math	Jogging

**Semantics:** A student can take more than one major and can enroll in more than one activity.

**Primary key:** All three attributes.

### What is the relationship between SID and Major?

It is not functional dependency, because students have several majors. There is some sort of relationship. This relationship can be illustrated by an example

Suppose: Student 100 wants to enroll in Skiing.

Add: tuple **100 Music Skiing**. Resulting relation

Student		
SID	Major	Activity
<b>100</b>	<b>Music</b>	<b>Skiing</b>
100	Music	Swimming
100	Accounting	Swimming
100	Music	Tennis
100	Accounting	Tennis
150	Math	Jogging

**Semantics:** It implies that **Student 100** can take up skis as a **Music** major but he/she does not know to ski as an **Accounting** major. Illogical.

**Solution:** Add tuple: **100 Accounting Skiing**. Resulting relation.

Student		
SID	Major	Activity
<b>100</b>	<b>Music</b>	<b>Skiing</b>
100	Music	Swimming
100	Accounting	Swimming
<b>100</b>	<b>Accounting</b>	<b>Skiing</b>
100	Music	Tennis
100	Accounting	Tennis
150	Math	Jogging

The resulting relation is consistent. The relationship between **SID** and **Major** is a **Multivalued dependency**. **SID** determines not a single value but several values. Thus **SID = 100** determines **Majors [Music, Accounting]** and **Activities [Skiing, Swimming, Tennis]**.

The relation is in BCNF since all attributes make the primary key. It has anomalies.

**Addition:** If one tuple is added then several other tuples must be added to preserve consistency.

**Delete:** If **100 Music Skiing** is deleted then **100 Accounting Skiing** also has to be deleted even though **Major** and **Activity** are not related.

### Student

SID	Major	Activity
<b>100</b>	<b>Music</b>	<b>Skiing</b>
100	Music	Swimming
100	Accounting	Swimming
<b>100</b>	<b>Accounting</b>	<b>Skiing</b>
100	Music	Tennis
100	Accounting	Tennis
150	Math	Jogging

**Solution:** As usual to minimize the anomalies, Student can be projected out into the following two relations.

### S\_major

SID	Major
100	Math
100	Accounting
150	Math

### S\_Activity

SID	Activity
100	Skiing
100	Swimming
100	Tennis
150	Jogging

**Multivalued dependency** always occur in pairs. For the Student relation **SID**  $\twoheadrightarrow$  **Major** because **Major** depends only on the value of **SID** and not on the value of **Activity**. Similarly **SID**  $\twoheadrightarrow$  **Activity**. **Activity** does not dependent on **Major** since having a particular major implies nothing about **Activity**. The no relationship between **Major** and **Activity** creates problem in the sense that whenever we add a new **Major**, we must add a tuple for every value of **Activity**.

The independence among attributes is not a problem if the attributes have a single value. For example, consider the following Student\_shoe relation.

### Student shoe

SID	Shoe-size	Marital status
100	8	M
150	10	S
200	5	S
250	12	S

**Primary Key: SID.**

In **Student\_shoe**, **Shoe\_size** and **Marital\_status** are independent and have single value, anomalies cannot occur. We can delete or add any tuple with no problem. This observation leads to the definition of 4NF.

### 4NF

*A relation is in 4NF if it is in BCNF and it has no multivalued dependencies.*