



Chapter 5 - Relational Database Design

- First Normal Form
- Pitfalls in relational database design
- Functional Dependencies
- Armstrong Axioms
- 2nd, 3rd, BCNF, and 4th normal form
- Decomposition, Desirable properties of decomposition
- Overall database design process



Pitfalls in relational database design

- Redundancy of data
- Unable to express certain information
- Wastage of storage



Anomalies

- Redundant information
- Insert anomalies
- Delete anomalies
- Update anomalies
- Null values in tuple
- Generation of Spurious tuples

Figure 10.2

Example database state for the relational database schema of Figure 10.1.

EMPLOYEE

Ename	<u>Ssn</u>	Bdate	Address	Dnumber
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

<u>Ssn</u>	<u>Pnumber</u>	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

EMPLOYEE

F.K.

Ename	<u>Ssn</u>	Bdate	Address	Dnumber
-------	------------	-------	---------	---------

P.K.

DEPARTMENT

F.K.

Dname	<u>Dnumber</u>	Dmgr_ssn
-------	----------------	----------

P.K.

DEPT_LOCATIONS

F.K.

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

P.K.

PROJECT

F.K.

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

P.K.

WORKS_ON

F.K.

F.K.

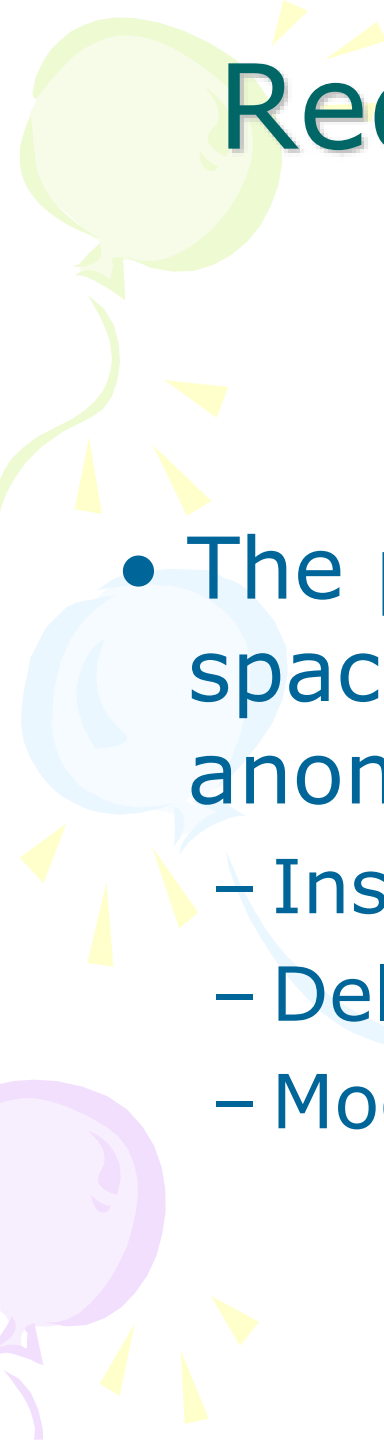
<u>Ssn</u>	<u>Pnumber</u>	Hours
------------	----------------	-------

P.K.

EMP_DEPT

Redundancy

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555



Redundant Information in Tuples and Update Anomalies

- The previous example not only waste spaces but also cause some anomalies:
 - Insert Anomaly
 - Delete Anomaly
 - Modify Anomaly

2. Redundant Information in Tuples and Update Anomalies

- Consider the relation:

EMP_PROJ

<u>Ssn</u>	<u>Pnumber</u>	Hours	Ename	Pname	Plocation
------------	----------------	-------	-------	-------	-----------

- Insert Anomaly:
 - For a new employee, you have to assign NULL to projects
 - Cannot insert a project unless an employee is assigned to it.

2. Redundant Information in Tuples and Update Anomalies

- Consider the relation:

EMP_DEPT

Ename	<u>Ssn</u>	Bdate	Address	Dnumber	Dname	Dmgr_ssn
-------	------------	-------	---------	---------	-------	----------

- Delete Anomaly:
 - If we delete from EMP_DEPT an employee tuple that happens to represent the last employee, the information containing that department is lost from the database

2. Redundant Information in Tuples and Update Anomalies

- Consider the relation:

EMP_DEPT

Ename	<u>Ssn</u>	Bdate	Address	Dnumber	Dname	Dmgr_ssn
-------	------------	-------	---------	---------	-------	----------

- Modify Anomaly:
 - If we change the value of the manager of department 5, we must update the tuples of all employees who work in the department



3. Null Values in Tuples

- As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL
- For example: if only 10% of employees have individual offices, DO NOT include a attribute OFFICE_NUMBER in the EMPLOYEE relation
- Rather, a relation EMP_OFFICES(ESSN, OFFICE_NUMBER) can be created. (just like WEAK entity type)

Generation of Spurious Tuples

- Let us consider two relation schema
Emp_Locs(ename, plocation)
Emp_proj1(eno, pnumber, hours, pname,
plocation)

If we attempt a natural join operation on above relation schema, the result produces many more tuples than the original set of tuples. Additional tuples that were not there in Emp_proj are called spurious tuples because they represent wrong information which is not valid



Guidelines For Relation Schema

- **Guideline 1** – Design a relation schema so that it is easier to explain its meaning. Do not combine attributes from multiple entity types and relationship types into a single relation
- **Guideline 2** – Design the base schema so that no insertion, deletion, or modification anomalies are present in the relations
- **Guideline 3** – As far as possible, avoid placing attributes in a base relation whose values may frequently be null. If nulls are unavoidable make sure that they apply in exceptional cases only and do not apply to majority of the tuples in the relation
- **Guideline 4** – Design relation schemas so that they can be joined with equality conditions on attributes that are either primary keys or foreign key that guarantees that no spurious tuples are generated



Functional Dependencies

- It is a constraint between two sets of attributes from the database. Let us consider that our relation schema has n attributes A_1, A_2, \dots, A_n . The whole database is described by a single universal relation schema

$$R = \{A_1, A_2, \dots, A_n\}$$

A **functional dependency** denoted by $\mathbf{X \rightarrow Y}$ between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R . The constraint is for any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$ must also have $t_1[Y] = t_2[Y]$

Functional Dependencies

- Values of Y component of a tuple in r depend on, or are determined by, the values of the X component **OR** The values of X component of a tuple uniquely or functionally determine the values of Y component
- There is a functional dependency from X to Y or Y is functionally dependent on X
- The abbreviation of functional dependency is FD or f.d.

Functional Dependency

- Social security number determines employee name
 - SSN \rightarrow ENAME
- Project number determines project name and location
 - PNUMBER \rightarrow {PNAME, PLOCATION}
- Employee ssn and project number determines the hours per week that the employee works on the project
 - {SSN, PNUMBER} \rightarrow HOURS

EMPLOYEE					F.K.
Ename	<u>Ssn</u>	Bdate	Address	Dnumber	
P.K.					

DEPARTMENT			F.K.
Dname	<u>Dnumber</u>	Dmgr_ssn	
P.K.			

DEPT_LOCATIONS		F.K.
<u>Dnumber</u>	<u>Dlocation</u>	
P.K.		

PROJECT				F.K.
Pname	<u>Pnumber</u>	Plocation	Dnum	
P.K.				

WORKS_ON			F.K.	F.K.
<u>Ssn</u>	<u>Pnumber</u>	Hours		
P.K.				



Functional Dependencies

- If the values of an attribute "Marks" is known then the values of an attribute "Grade" are determined since

Marks \rightarrow Grade

- X is called the left-hand side of FD, and Y is called the right-hand side

Inference Rules for Functional Dependencies

- Let F is the set of functional dependencies that are specified on relation schema R . However, numerous other dependencies can be deduced or inferred from the FDs in R . In real life it is impossible to specify all them for given situation
e. g. if each department has one manager, so dept_no uniquely determines manager_no, and a manager has a unique phone number then manager_no uniquely determines mgr_phone, these two dependencies together imply that dept_no determines mgr_phone. This is an inferred FD and need not be explicitly stated.

If $A \rightarrow B$ and $B \rightarrow C$, then we can infer that $A \rightarrow C$

Closure of Functional Dependencies

- *The set of all dependencies that include F as well as all dependencies that can be inferred from F is called closure of F . It is denoted by F^+*

Example:

$F = \{eno \longrightarrow \{ename, dob, address, dnumber\},$
 $dnumber \longrightarrow \{dname, dmgrno\}$

Inferred functional dependencies are

$eno \longrightarrow \{dname, dmgrno\}$

$eno \longrightarrow eno$

$dnumber \longrightarrow dname$



Inference Rules for Functional Dependencies

- *A set of rules that can be used to infer new dependencies from a given set of dependencies*
- *We use the notation $F \models X \rightarrow Y$ to denote that the functional dependency $X \rightarrow Y$ is inferred from the set of functional dependencies F*

Inference Rules for Functional Dependencies

- IR1 (Reflexive rule) If $X \supseteq Y$ then $X \longrightarrow Y$
- IR2 (Augmentation rule) $\{X \longrightarrow Y\} \mid = XZ \longrightarrow YZ$
- IR3 (Transitive rule) $\{X \longrightarrow Y, Y \longrightarrow Z\} \mid = X \longrightarrow Z$
- IR4 (Decomposition or Projective rule)
 $\{X \longrightarrow YZ\} \mid = X \longrightarrow Y$
- IR5 (Union or Additive rule)
 $\{X \longrightarrow Y, X \longrightarrow Z\} \mid = \{X \longrightarrow YZ\}$
- IR6 (Pseudo transitive rule)
 $\{X \longrightarrow Y, WY \longrightarrow Z\} \mid = \{WX \longrightarrow Z\}$

Proof of IR4 (Using IR1 through IR3).

1. $X \rightarrow YZ$ (given).
2. $YZ \rightarrow Y$ (using IR1 and knowing that $YZ \supseteq Y$).
3. $X \rightarrow Y$ (using IR3 on 1 and 2).

Proof of IR5 (Using IR1 through IR3).

1. $X \rightarrow Y$ (given).
2. $X \rightarrow Z$ (given).
3. $X \rightarrow XY$ (using IR2 on 1 by augmenting with X ; notice that $XX = X$).
4. $XY \rightarrow YZ$ (using IR2 on 2 by augmenting with Y).
5. $X \rightarrow YZ$ (using IR3 on 3 and 4).

Proof of IR6 (Using IR1 through IR3).

1. $X \rightarrow Y$ (given).
2. $WY \rightarrow Z$ (given).
3. $WX \rightarrow WY$ (using IR2 on 1 by augmenting with W).
4. $WX \rightarrow Z$ (using IR3 on 3 and 2).



Types of Functional Dependencies

- Full Functional dependency
- Partial Functional dependency
- Transitive Functional dependency

Definitions of Keys and Attributes Participating in Keys

- **Key**- The difference between Key and Super key is that a key has to be minimal.

$\{ssn\}$

- **Super Key**- A key K is a super key with the additional property that removal of any attribute from K will cause K not to be a super key any more.

$\{ssn, Ename\}, \{ssn, Ename, Bdate\}$

Ename

ssn

Bdate

Address

dnumber

Definitions of Keys and Attributes Participating in Keys

- **candidate key** - *If a relation schema has more than one key, each is called a candidate key*
- **Primary And Secondary Key** – *One of the candidate key is designated to be a primary key and others are called secondary keys*

Definitions of Keys and Attributes Participating in Keys

- **Prime Attribute** – *If an attribute of a relation R is member of some candidate key of R*
- **Nonprime** – *If an attribute of a relation R is not member of any candidate key of R*



Normalization of Relations

- **Normalization:**

- The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

- **Normal form:**

- Condition using keys and FDs of a relation to certify **whether a relation schema is in a particular normal form**

A decorative graphic on the left side of the slide featuring three balloons: a light green one at the top, a light blue one in the middle, and a light purple one at the bottom. Yellow streamers and triangular flags are attached to the balloons.

Benefits of Normalization

- Less storage space
- Quicker updates
- Less data inconsistency
- Clearer data relationships
- Easier to add data
- Flexible Structure



Normalization

- The normalization process was first proposed by Codd in 1972
- **It takes relation schema through a series of tests to certify whether it satisfies a certain normal form**
- The process proceeds in a **top-down fashion** by evaluating each relation against the criteria for normal forms and decomposing relations as necessary
- **Codd proposed three normal forms which he called First, Second, Third normal form**
- A strong definition of 3NF is called Boyce-Codd normal (BCNF) form was proposed by Boyce and Codd

Normalization

- **All these normal forms are based on the functional dependencies among the attributes of the relation**
- Later, the 4NF and 5NF were proposed, based on the concept of multi-valued and join-dependencies, respectively
- **Normalization is a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties**
 - Minimizing redundancy
 - Minimizing insertion, deletion, modification anomalies
- A relation that do not meet normal form tests are decomposed into smaller relation schemas that meet the tests and posses the desirable properties



List of Normal Forms

- First Normal Form (1NF)
 - Atomic values
- 2NF, 3NF
 - based on primary keys
- 4NF
 - based on keys, multi-valued dependencies
- 5NF
 - based on keys, join dependencies



First Normal Form

- It was designed **to disallow multi-valued attributes, composite attributes and their combinations**
- It states that the domain of an attribute must include only atomic i.e. simple or indivisible values and that value of any attribute in a tuple must be a single value from the domain of that attribute



First Normal Form

- Historically, it is designed to disallow
 - **composite attributes**
 - **multivalued attributes**
 - **Or the combination of both**
- All the values need to be **atomic**

First Normal Form (1NF)

- Consider **Department** relation schema whose **primary key** is **dnumber**. If extended by adding dlocations attribute. **Each department can have a number of locations**

Department



- The **Department** schema is not in 1NF
 - The domain of dlocations contains atomic values, but some tuples can have set of these values. Dlocations is not functionally depend on the primary key

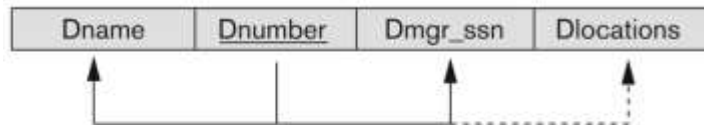
First Normal Form (1NF)

- There are 3 techniques to achieve 1NF for such a relation
 - Remove the attribute that violates 1NF and place it in a separate relation along with the primary key of original relation. The primary key of new relation is the combination.
 - Expand the key so that there will be a separate tuple in the original relation for each location of a department. The primary key becomes the combination. Disadvantage of introducing redundancy
 - **If maximum number of values is known for the attribute. Replace that attribute by those many atomic attributes . Disadvantages of introducing null**

First Normal Form

(a)

DEPARTMENT



(b)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	<u>Dlocation</u>
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Figure 10.8

Normalization into 1NF.

(a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

Second Normal Form (2NF)

- **Second Normal form is based on the concept of full functional dependencies**
- A functional dependency $X \rightarrow Y$ is a full functional dependency if removal of any attribute A from X means that the dependency does not hold for any more

$\{\text{eno}, \text{pnumber}\} \rightarrow \text{hours}$

is a full functional dependency

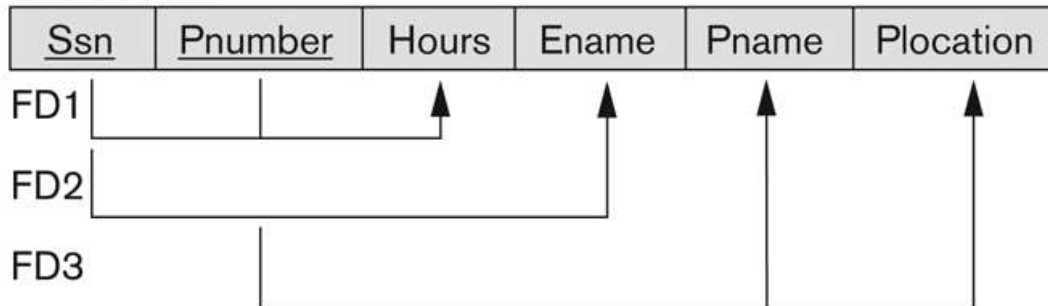
Neither $\text{eno} \rightarrow \text{hours}$ nor
 $\text{pnumber} \rightarrow \text{hours}$

Second Normal Form

- In this example, $\{Ssn, Pnummber\} \rightarrow Hours$ is a fully dependency
- However, the dependency $\{Ssn, Pnumber\} \rightarrow Ename$ is partial because $Ssn \rightarrow Ename$ holds

(b)

EMP_PROJ

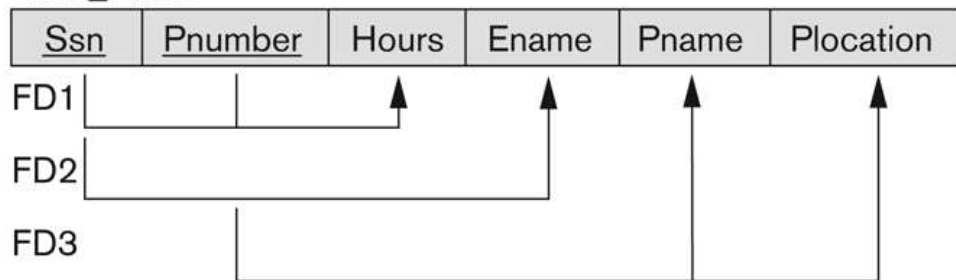


Second Normal Form

- A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on the primary key
- A functional dependency $X \rightarrow Y$ is a **partial dependency** if some attribute A belong X can be removed from X and the dependency still holds

(b)

EMP_PROJ

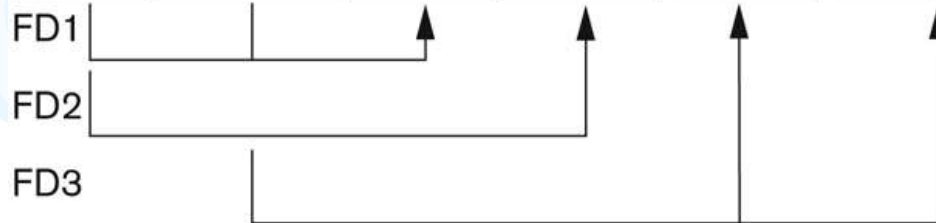


Second Normal Form

(a)

EMP_PROJ

<u>Ssn</u>	<u>Pnumber</u>	Hours	Ename	Pname	Plocation
------------	----------------	-------	-------	-------	-----------



2NF Normalization

EP1

<u>Ssn</u>	<u>Pnumber</u>	Hours
------------	----------------	-------



EP2

<u>Ssn</u>	Ename
------------	-------



EP3

<u>Pnumber</u>	Pname	Plocation
----------------	-------	-----------

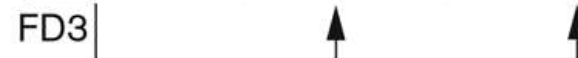


Figure 10.10

Normalizing into 2NF and 3NF.
(a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.



Third Normal Form

- A relation schema R is in **third normal form (3NF)** if it is in 2NF *and* no non-prime attribute A in R is transitively dependent on the primary key
 - **Transitive functional dependency:** a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$



Third Normal Form

- Examples:

- SSN \rightarrow DMGRSSN is a **transitive** FD

- Since SSN \rightarrow DNUMBER and DNUMBER \rightarrow DMGRSSN hold

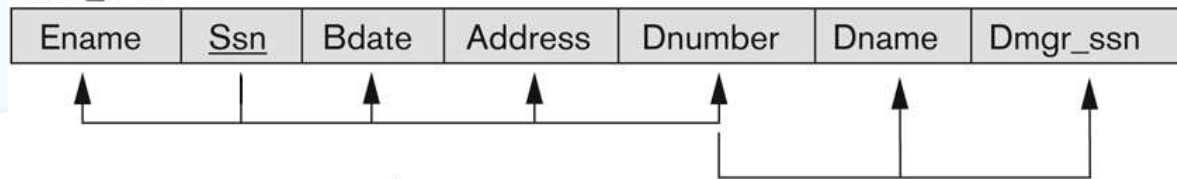
- SSN \rightarrow ENAME is **non-transitive**

- Since there is no set of attributes X where SSN \rightarrow X and X \rightarrow ENAME

Third Normal Form

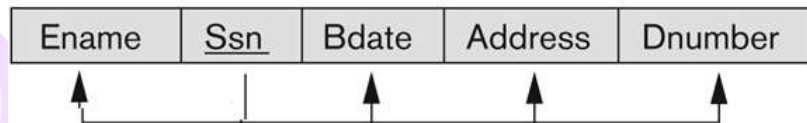
(b)

EMP_DEPT

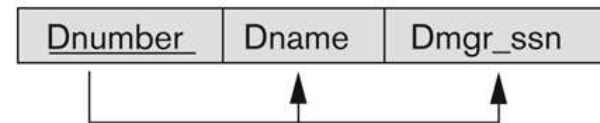


3NF Normalization

ED1



ED2






SUMMARY OF NORMAL FORMS based on Primary Keys

Table 10.1

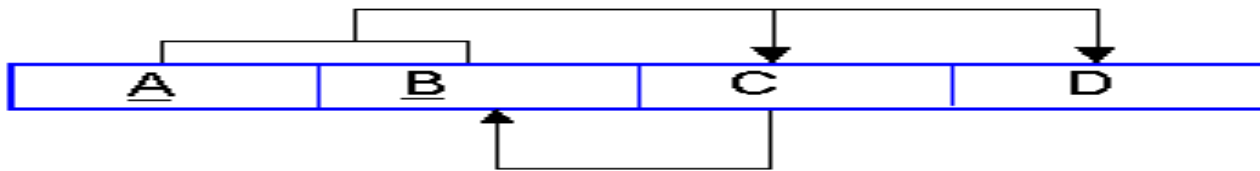
Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multi-valued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).



- ***BCNF:***

Boyce Codd Normal Form (BCNF) is considered a special condition of third Normal form. **A table is in BCNF if every determinant is a candidate key.** A table can be in 3NF but not in BCNF. This occurs when a non key attribute is a determinant of a key attribute. The dependency diagram may look like the one below



The table is in 3NF. A and B are the keys and C and D depend on both A and B. There are no transitive dependencies existing between the non key attributes, C and D.

The table is not in BCNF because a dependency exists between C and B. In other words if we know the value of C we can determine the value of B.

• *EXAMPLE*

If we know the Student Number and Teacher Code we know the Offering (class) the student is in. We also know the review date for that student and teacher (Student progress is reviewed for that class by the teacher and student).

<u>S_Num</u>	<u>T_Code</u>	Offering#	Review Date
123599	FIT104	01764	2nd March
123599	PIT305	01765	12th April
123599	PIT107	01789	2nd May
346700	FIT104	01764	3rd March
346700	PIT305	01765	7th May

- *EXAMPLE*

- The dependencies are

$S_Num, T_Code \longrightarrow Offering\#, Review\ Date$ which means that the table is in third normal form. The table is not in BCNF as if we know the offering number we know who the teacher is. Each offering can only have one teacher!

$Offering\# \longrightarrow T_Code$

A non key attribute is a determinant.

- If we look at the table we can see a combination of T_Code and $Offering\#$ is repeated several times. eg FIT104 and 01764.

- *EXAMPLE*

Converting to BCNF

- 1** The determinant, Offering#, becomes part of the key and the dependant attribute T_Code, becomes a non key attribute. So the Dependency diagram is now
 $S_Num, Offering\# \rightarrow T_Code, Review\ Date$
- 2** There are problems with this structure as T_Code is now dependant on only part of the key. This violates the rules for 2NF, so the table needs to be divided with the partial dependency becoming a new table. The dependencies would then be
 $S_Num, Offering\# \rightarrow Review\ Date$
 $Offering\# \rightarrow T_Code$
- 3** The original table is divided into two new tables. Each is in 3NF and in BCNF.



STUDENT REVIEW

<u>S Num</u>	<u>Offering#</u>	Review Date
123599	01764	2nd March
123599	01765	12th April
123599	01789	2nd May
346700	01764	3rd March
346700	01765	7th May

OFFERING TEACHER

Offering#	<u>T Code</u>
01764	FIT104
01765	PIT305
01789	PIT107