Unit VI
Functional Dependencies and Normalization

The **normalization** was first developed by Codd in the year 1972. *Normalization* is the process of efficiently organizing the data in a database. There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). The database community has developed a series of guidelines for ensuring that the databases are normalized. These are referred to as normal forms such as 1NF, 2NF, 3NF, and BCNF.

Informal Design Guidelines for Relation Schemas

Four important informal design guidelines for relations schema design are:

1. Semantics of the attributes
2. Reducing the redundant information in tuples
3. Reducing the NULL values in tuples
4. Disallowing the possibility of generating spurious tuples

Semantics of the attributes:

The word semantics refers to the meaning of the attributes. Attributes mean how the attribute values in a tuple relate to one another. For example, the meaning of the EMPLOYEE relational schema is that each tuple represents an employee record. The attribute SSN represents the Social Security Number that is used as a key, Bdate represents the birth date, and so on.

Guideline – 1

Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity types and relationship types into a single relation.

Reducing the redundant information in tuples:

Data redundancy is means that some data fields appear more than once in the database. Data redundancy is wasteful and insufficient for several reasons. One goal of the schema design is to minimize the storage space that the base relations occupy. One of the main reasons for the data redundancy to appear in database is due to anomalies.

There are three major anomalies.

1. Insertion Anomalies
2. Deletion Anomalies
3. Updation Anomalies

Let us consider an example based on a relation EMP_DEPT

Redundancy

EMP_DEPT

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

Insertion Anomalies:
Suppose for a new employee tuple insertion, until and unless an employee is assigned to a particular department, data of the employee cannot be inserted, or else we will have to set the department information as NULL.
Also, if we have to insert data of 100 employees for the same department, then the department information will be repeated for all those 100 employees.
The above discussed situations are called insertion anomalies.

Delete Anomalies:
In the above EMP_DEPT table, two different data are kept together such as Employee information and Department information. At certain point of time, if some of the employees leave the company then we will lose the department information and if a particular department is stopped functioning, then we will lose employee information. This is called Deletion anomalies.

Updation Anomalies:
In the above table, if we change the value of one of the attributes of a particular department then we must update the tuples of all employees who work in that department otherwise the database will become inconsistent.
Based on the three anomalies, we can state the following guideline.

Guideline – 2
Design the base relation schemas so that no insertion, deletion, or Updation anomalies are present in the relations.

Reducing the NULL values in tuples:
If some of the attributes do not apply to all tuples in the relation, NULL values are stored in those attributes. If NULL values are present, the results become unpredictable. Moreover, NULLs can have multiple interpretations, such as the following:

- The attributes *does not apply* to this tuple.
- The attribute value for this tuple is *unknown*.
- The value is *known but absent;* that is, it has not been recorded yet.

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 a majority of tuples in the relation.

Disallowing the possibility of generating spurious tuples
When there are two relations not having any common attributes between them and if we try to join those two relations, the resultant relation after joining will contain spurious tuples. (A spurious tuple is a record in a database that gets created when two tables are joined badly. Spurious tuples are created when two tables are joined on attributes that are neither primary keys nor foreign keys.)

Guideline – 4
Design relation schemas so that they can be joined with equality conditions on attributes that are (primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated. Avoid relations that contain matching attributes that are not (primary key, foreign key) combinations because joining on such attributes may produce spurious tuples.

Functional Dependencies:

A *functional dependency* is a constraint between two sets of attributes from the database. Suppose that our relational database schema has *n* attributes A_1, A_2,…, A_n. Let us think of the whole database as being described by a single universal relation schema R = {A_1, A_2, … , A_n}.

A *functional dependency*, denoted by $X \rightarrow Y$, 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 that, for any two tuples t_1 and t_2 in *r* that have $t_1[X] = t_2[X]$, they must also have $t_1[Y] = t_2[Y]$.

$X \rightarrow Y$ means that X uniquely determines Y or Y is uniquely determined by X.
This means that the values of the Y component of a tuple in *r* depend on, or are *determined by*, the values of the X component; alternatively, the values of the X component of a tuple uniquely (or functionally) determine the values of the Y component. We also say that there is a functional dependency from X to Y, or that Y is functionally dependent on X.
The left-hand side of the Functional Dependency is sometimes called the determinant (X) and the right-hand side is called dependent (Y).

Functional Dependencies are divided into two types.

1. Full Functional Dependency
2. Partial Functional Dependency

Redundancy Redundancy

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

Consider the relation schema EMP_PROJ in the above figure. From the semantics of the attributes, we know that the following functional dependencies should hold:

a. Ssn → Ename
b. Pnumber → {Pname, Plocation}
c. {Ssn, Pnumber} → Hours

These functional dependencies specify that (a) the value of an employee's social security number (Ssn) uniquely determines the employee name (Ename), (b) the value of a project's number (Pnumber) uniquely determines the project name (Pname) and location (Plocation), and (c) a combination of Ssn and Pnumber values uniquely determines the number of hours the employee currently works on the project per week (Hours).

Full Functional Dependency:
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 any more; that is, for any attribute $A \in X$, (X – {A}) does not functionally determine Y.

Example:
In the above relation EMP_PROJ, {Ssn, Pnumber} → Hours is full functional dependency whereas Ssn → Hours or Pnumber → Hours are not fully functional dependent.
Ssn → Hours is not fully functional dependent because we removed Pnumber and hence, Ssn alone cannot uniquely determine the hours. In the same way, Pnumber → Hours is not fully functional dependent because we removed Ssn and hence, Pnumber alone cannot uniquely determine the hours.

Partial Functional Dependency:
A functional dependency $X \rightarrow Y$ is a ***partial functional dependency*** if some attribute $A \in X$ can be removed from X and the dependency still holds; that is, for some $A \in X$, (X – {A}) → Y.

Example:
The dependency {Ssn, Pnumber} → Ename is partial functional dependency because Ssn → Ename holds. Even after removing Pnumber, Ssn alone uniquely determines Ename. This is called Partial functional dependency.

Prime and Nonprime Attributes:
If a relation schema has more than one key, each is called a **candidate key**. One of the candidate keys is arbitrarily designated to be the **primary key**, and the others are called secondary keys. Each relation schema must have a primary key.
The above table EMP_PROJ does not have single attributes that can find the tuple uniquely. In such cases a combination of two or more attributes are used as a primary key. Such keys are called Composite keys. Hence EMP_PROJ relation contains combination of Ssn and Pnumber as a primary key.

An attribute of a relation schema R is called a **prime attribute** of R if it is a member of some *candidate key* of R.
Example: In EMP_PROJ relation, Ssn and Pnumber are called prime attributes.

An attribute of a relation schema R is called a **nonprime attribute** of R if it is not a member of some *candidate key* of R. (that is, if it is not a prime attribute)
Example: In EMP_PROJ relation, other than Ssn and Pnumber all other attributes are nonprime attributes. Hours, Ename, Pname, and Plocation are nonprime attributes.

Normal Forms Based on Primary Keys:
Normalization of data can be considered a process of analyzing the given relation schemas based on their Functional Dependencies (FDs) and primary keys to achieve the desirable properties of (1) minimizing redundancy and (2) minimizing the insertion, deletion, and update anomalies. Unsatisfactory relation schemas that do not meet certain conditions are decomposed into smaller relation schemas that meet the desirable properties.

To design a good database, In 1972 Codd proposed three normal forms which are called First Normal Form, Second Normal Form, Third Normal Form. A stronger definition of Third Normal Form (Boyce-Codd Normal Form) was proposed later by Boyce and Codd.

Hence Normalization is categorized into following normal forms:

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce-Codd Normal Form (BCNF)

Redundancy

EMP_DEPT

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

Redundancy Redundancy

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

1. First Normal Form (1NF)

- First Normal Form was defined to disallow multivalued attributes, composite attributes and their combinations.
- It states that the domain of an attribute must include only *atomic (simple, indivisible)* values.

DEPARTMENT

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

1 NF ⇩

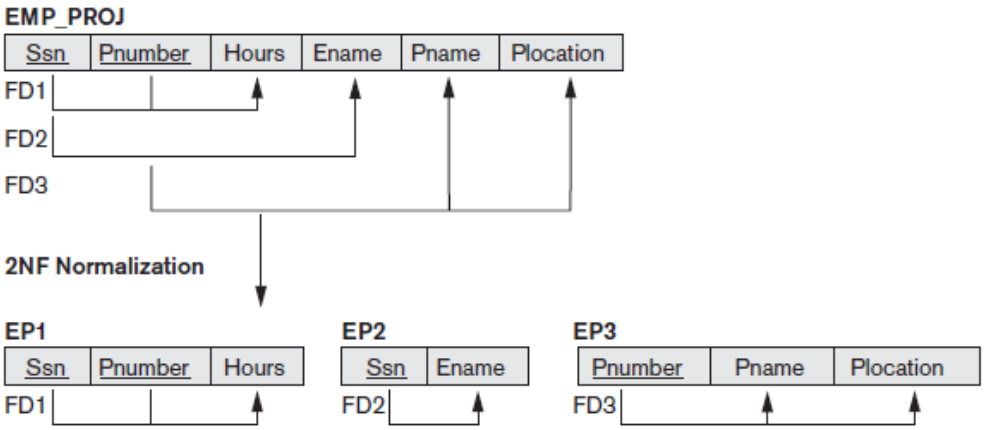
DEPARTMENT

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

Consider the DEPARTMENT relation schema, it is not in 1NF because the attribute Dlocation contains multiple values for the Dnumber 5. We modify the relation where Dlocation can have only atomic values. After applying 1NF, the DEPARTMENT relation contains atomic values for the attribute Dlocation.

2. Second Normal Form (2NF)

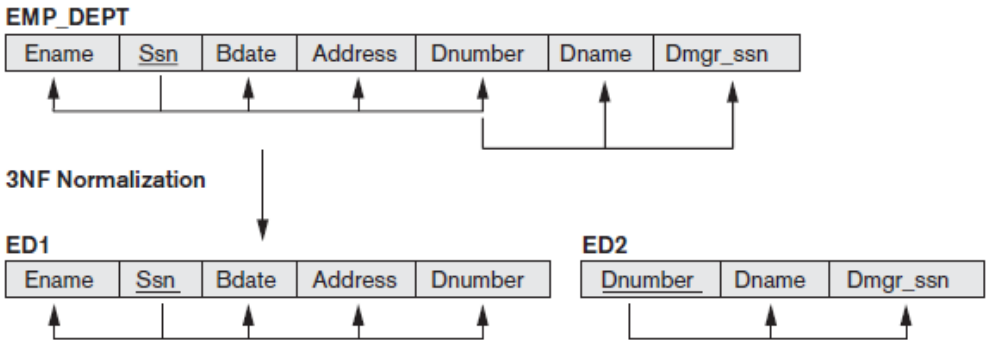
- **Second Normal Form** is based on the concept of *full functional dependency*.
- The relation should be in 1NF.
- A relation schema R is in 2NF if every nonprime attribute A in R is *fully Functionally dependent* on the primary key of R.



The test for 2 NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key. The above EMP_PROJ relation is in 1NF but is not in 2NF. The nonprime attribute Ename violates 2NF because of FD2, and the nonprime attributes Pname and Plocation because of FD3. The functional dependencies FD2 and FD3 make Ename, Pname, and Plocation partially dependent on the primary key {Ssn, Pnumber} thus violating the 2NF test.
Hence the functional dependencies FD1, FD2, and FD3 lead to the decomposition of EMP_PROJ into the three relation schemas EP1, EP2, and EP3 as shown above, each of which is in 2NF.

3. Third Normal Form (3NF)

- **Third Normal Form** is based on the concept of *transitive dependency.*
- The relation should be in 2NF.
- According to Codd, a relation schema R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent of the primary key.



The functional dependency $X \rightarrow Y$ in a relation schema R is a **transitive dependency** if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R, and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold.
The dependency Ssn → Dmgr_ssn is transitive through Dnumber in EMP_DEPT because both the dependencies Ssn → Dnumber and Dnumber → Dmgr_ssn hold and Dnumber is neither a key itself nor a subset of the key of EMP_DEPT.
The relation schema EMP_DEPT is in 2NF, since no partial dependencies on a key exist. However, EMP_DEPT is not in 3NF because of the transitive dependency of Dmgr_ssn (and also Dname) on Ssn via Dnumber. We can normalize EMP_DEPT by decomposing it into the two 3NF relation schemas ED1 and ED2 as shown in the above figure.

4. Boyce-Codd Normal Form (BCNF)

- **Boyce-Codd Normal Form (BCNF)** was proposed as a simpler form of 3NF, but it was found to be stricter than 3NF.
- The relation should be in 3NF.
- A relation is in BCNF, if and only if every determinant in the table is a candidate key.

TEACH

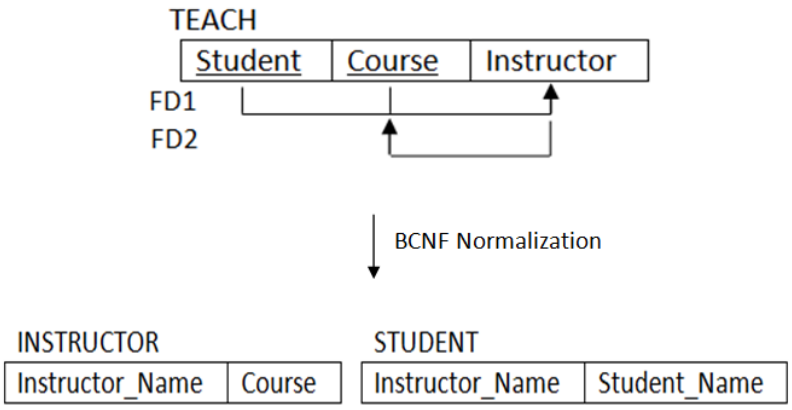
Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

Consider a relation TEACH with the following dependencies:

FD1: {Student, Course} → Instructor

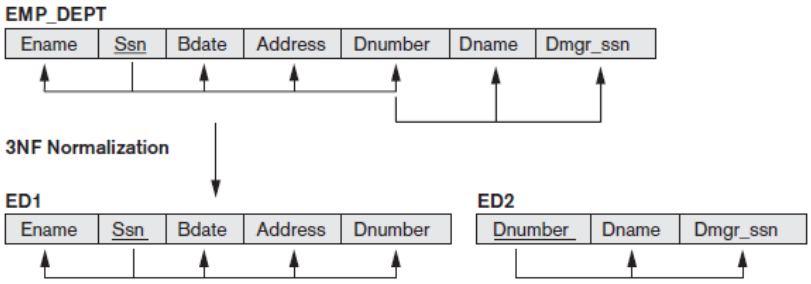
FD2: Instructor → Course

Note that {Student, Course} is a candidate key for this relation and that the dependencies shown follow the pattern below.



The relation TEACH is in 3NF but not in BCNF. The relation TEACH can be decomposed into two relations INSTRUCTOR and STUDENT which fall into the BCNF.
Note: Whenever a nonprime attribute determines a prime attribute then the relation is not in BCNF.

Inference Rules:



1. **Reflexive Rule (IR_1)**
   In the reflexive rule, if Y is a subset of X, then X determines Y.
   If $X \supseteq Y$ then $X \rightarrow Y$
   Example:
   X = {Ename, Ssn, Bdate, Address, Dnumber}
   Y = {Dname, Dmgr_ssn}

2. **Augmentation Rule (IR_2)**
   The augmentation is also called a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.
   If $X \rightarrow Y$ then $XZ \rightarrow YZ$
   Example:
   if {Ssn} → {Ename} then {Ssn, Bdate}→ {Ename, Bdate}

3. **Transitive Rule (IR_3)**
   In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.
   If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$
   Example:
   if {Ssn} → {Dnumber} and {Dnumber} → {Dname} then {Ssn}→ {Dname}

4. **Union Rule (IR_4)**
   Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.
   If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$
   Example:
   if {Ssn} → {Ename} and {Ssn} → {Address} then {Ssn}→ {Ename, Address}

5. **Decomposition Rule (IR_5)**
   Decomposition rule is also known as project rule. It is the reverse of union rule.
   This Rule says, if X determines Y and Z, then X determines Y.
   If $X \rightarrow YZ$ then $X \rightarrow Y$

6. **Pseudotransitive Rule (IR_6)**
   This rule says, if X determines Y and WY determines Z, then WX must also determine Z.
   If $X \rightarrow Y$ and $WY \rightarrow Z$ then $WX \rightarrow Z$