CSCI235 – Database Systems

Normalization in practice

Normalization

What is it?

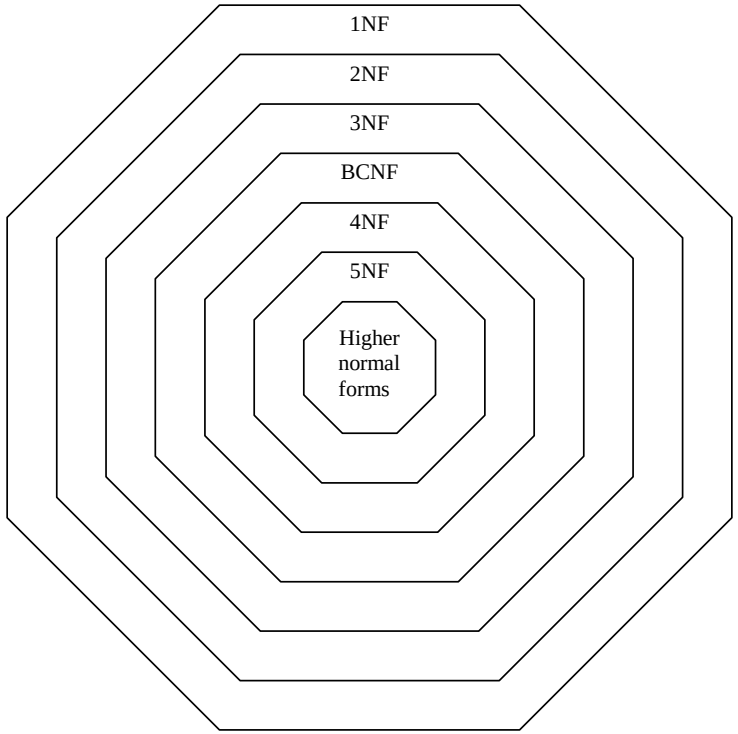
- A technique for producing a set of relations with desirable properties, given the data requirements of an organization.
- Often performed as a series of tests on a relation to determine whether it satisfies or violates the requirements of a given normal form.
- Three normal forms were initially proposed, called First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF).

Normalization

- A stronger definition of third normal form was introduced by R. Boyce and E.F. Codd in 1974, and is referred to as Boyce-Codd Normal Form (BCNF).

The Process of Normalization

Normalization is often performed as a series of tests on a relation to determine whether it satisfies or violates the requirements of a given normal form, starting from 1NF to 2NF then 3NF and so on.



The Process of Normalization

As normalization proceeds, the relations become progressively more restricted (stronger) in format, and also less vulnerable to update anomalies.

In relational database design, it is important to recognize that it is only first normal form (1NF) that is critical in creating appropriate relations.

All of the subsequent normal forms are optional. However, to avoid the update anomalies, it is normally recommended that we proceed to 3NF or BCNF (if needed).

Normal Forms

Normal form is a way of measuring the levels, or depth, to which a database has been normalized.

Four common normal forms:

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

1NF, 2NF and 3NF depends on each others; i.e., a subsequent normal form depends on normalization steps taken in the previous normal form.

Update Anomalies

Update Anomalies

- Three forms of **update** anomalies:
  - **Insert Anomaly**
  - **Delete Anomaly** and
  - **Modification Anomaly**

Update anomalies are problems that arise when information is inserted, deleted, or updated into a relation with a **flawed** structure.

Update Anomalies

Taking the STAFFBRANCH relation for illustration of the three anomalies:

STAFFBRANCH

SNum	SName	SAddress	Position	Salary	BrNum	BrAddress	PhoneNum
SL21	Aaron Kwok	12 Bk Timah Rd.	Dy. Manager	30000	B5	Suntec City, Tower A	333-8863
SG37	Andy Lau	19 Katong R.	Manager	32000	B3	UE Square, #27-18	123-5432
SG14	Taufik B.	7 Linden St.	Asst. Manager	28000	B3	UE Square, #27-18	123-5432
SA9	Jacky Cheung	1 George Park	Director	40000	B7	Ocean Tower, #07-09	334-1234
SG5	Olinda C.	12 Princess Rd.	Asst. Manager	28000	B3	UE Square, #27-18	123-5432
SL41	Ocean Aw	9 Bk Timah	Manager	32000	B5	Suntec City, Tower A	333-8863

Insert Anomalies

There are two problems that we will encounter when we try to insert records into the STAFFBRANCH relation:

Problem Number 1:

To insert the details of new members of staff into the STAFFBRANCH relation, we must include the details of the branch at which the staff are to be located.

Consequence:

- Leads to data duplication (redundancy).
- Need Multiple updates.

Insert Anomalies

Problem Number 2:

Cannot insert the details of a new branch that currently has **no** members of staff into the STAFFBRANCH relation.

Why?

Since the staff details are not available at the time, NULL value have to be inserted for the SNum, SName, SAddress, Position and Salary. However, SNum is the primary key for the STAFFBRANCH relation, this violates the entity integrity of the relation.

Deletion Anomalies

To delete records from the STAFFBRANCH relation causes problem too:

Problem:

We cannot delete a staff record from the STAFFBRANCH relation if the record represents the last member of staff located at a branch.

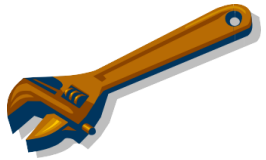
Reason being:

Removing the record causes the related branch details being lost.

Modification Anomalies

Modification to the STAFFBRANCH relation causes problem too:

Problem: Changes to any of the attributes related to branch details may require multiple updates.



Consequences:

If changes is not carried out on all the appropriate rows of the STAFFBRANCH relation, the database will become inconsistent.

Relational Data Model Integrity

Relational Data Model - Integrity

- An important aspect of RDBMS is support for integrity.
- Two important aspects of integrity rules in relational data model (that are enforced by a RDBMS) are:
  - **Entity integrity**
  - **Referential integrity**

- A third type of integrity rules (an equally important integrity, but is not enforced by most RDBMS) is the **semantic integrity**

Concept of a superkey

Terminology and processes

Closure of an Attribute Set

What is a closure of an attribute set?

The closure of a set of attribute(s) {X} is the set of attributes (in a relation) {Y} that can be functionally determined from the attribute(s) set {X}. (Note: an attribute set is a collection of attributes that describe all the characteristics of an entity in a domain.)

Closure of attributes set $\{X\}$ is denoted as $\{X\}^+$

Terminology and processes

- The closure of attributes that includes all the attributes of a relation R in the result is one of the keys (minimal super key / candidate keys).

Terminology and processes

Steps to find closure of an attribute set

Following steps are followed to find the closure of an attribute set:

Step 1:

- Add the attributes contained in the attribute set for which closure is being calculated to be result set.

Step 2:

- Recursively add the attributes to the result set which can be functionally determined from the attributes already contained in the result set.

Terminology and processes

The algorithm:

Step 1:

- $X^+ = X$

Step 2:

- Repeat until X^+ does not change
  - For each FD $Y \rightarrow Z$ in F
    - If $Y \subseteq X^+$ then $X^+ = X^+ \cup Z$

Terminology and processes

Example:
Consider a relation $R(A, B, C, D, E, F, G)$ with the functional dependencies:

- $A \rightarrow BC$
- $BC \rightarrow DE$
- $D \rightarrow F$
- $CF \rightarrow G$

Terminology and processes

Next, let us find the closure of some attributes and attribute sets:

Closure of attribute {A}:

$$\{A\}^+ = \{A\}$$
$$= \{A, B, C\}\ (Using\ A \rightarrow BC)$$
$$= \{A, B, C, D, E\}\ (Using\ BC \rightarrow DE)$$
$$= \{A, B, C, D, E, F\}\ (Using\ D \rightarrow F)$$
$$= \{A, B, C, D, E, F, G\}\ (Using\ CF \rightarrow G)$$

Hence, $\{A\}^+ = \{A, B, C, D, E, F, G\}$

Terminology and processes

Next, let us find the closure of some attributes and attribute sets:

Closure of attribute {B,C}:

$$\{B,C\}^+ = \{B,C\}$$
$$= \{B,C,D,E\}\,(Using\ BC \rightarrow DE)$$
$$= \{B,C,D,E,F\}\,(Using\ D \rightarrow F)$$
$$= \{B,C,D,E,F,G\}\,(Using\ CF \rightarrow G)$$

Hence, $\{B,C\}^+ = \{B,C,D,E,F,G\}$

Terminology and processes

Next, let us find the closure of some attributes and attribute sets:

Closure of attribute $\{D\}$:

$\{D\}^+ = \{D\}$

$= \{D, F\}\ (Using\ D \rightarrow F)$

We cannot determine any other attribute using attributes D and F contained in the result set.

Hence, $\{D\}^+ = \{D, F\}$

Terminology and processes

Next, let us find the closure of some attributes and attribute sets:

Closure of attribute $\{C, F\}$:

$\{C, F\}^+ = \{C, F\}$

$= \{C, F, G\}\ (Using\ CF \rightarrow G)$

We cannot determine any other attribute using attributes C, F and G contained in the result set.

Hence, $\{C, F\}^+ = \{C, F, G\}$

Super key vs Minimal Super key

Super key

- A super key is an attribute or a set of attributes such that its closure is all attributes in the relation. In other words, the closure of a super key is the entire relation schema.

Super key

- To find a super key or minimal super key of a relation, we need to find the closure of certain attributes (in most of the cases, the attribute(s) on the left hand side of a functional dependency).
- The closure that includes all the attributes of a relational table R in the result is one of the keys (minimal keys / candidate keys).

Super key vs Minimal super key

- A minimal super key is defined as that part of super-key from which any attribute cannot be removed without sacrificing the uniqueness. In other words, if we take any attributes out of a super key, it will not be a super key anymore.
- A relation can have more than one minimal super key.

Determining candidate keys

- Find out attribute closure for each of the determinant.
- Check which determinant closure derive all the attributes of the relation.
- Declare all the determinant whose closure set contains all the attributes, as candidate key
- In case non of the determinant of the given functional dependencies derives all the attributes, then we shall choose the best closure based on more number of attribute derived and then combine the missing attributes to the best closure and make sure it is the minimal subset before it gets declared as candidate key.

Examples

$R(A, B, C, D)$
$\{A \rightarrow C, B \rightarrow D\}$

Closure of attribute {A}:

$\{A\}^+ = \{A\}$
$= \{A, C\}$

$\{B\}^+ = \{B\}$
$= \{B, D\}$

Examples

Since none of the determinant's closure set derives all the attributes, hence, we can combine AB, using composition rule, and get the closure.

$$AB^{+} = \{A, C, B, D\}$$

AB^{+} derives all the attribute, hence, AB is the **candidate key**.

Back to Relational Data Model Integrity Constraints

Relational Data Model - Integrity

Entity integrity

- Entity integrity dictates that each relational table have exactly one primary key, and it **cannot** have null values in any tuple of the relational table. This is because primary key values are used to *identify* the individual tuple.
- A primary key is a combination of one or more attributes whose value unambiguously locates each row in a table.
- The primary key is always a candidate key.
- A candidate key is a minimal set of attributes that uniquely identifies a tuple.

Relational Data Model - Integrity

Candidate Key

Student table

StudentId	Name	NRIC	Region	Counselor
3399059	Benjamin Bayer	S8630561A	Singapore	227499
3331696	Katherine Ashly	S8938162E	Singapore	
3415983	Charles Cooper	S8637698B	Singapore	227500
3399023	Barbara Benson	S8883984D	Singapore	227499

Primary Key

Candidate Key

Staff table

StaffId	Name	NRIC	Region
227499	PS Liew	S7525984D	Singapore
227500	PK Loo	S7065972G	Singapore
227501	S Japit	S2598469B	Singapore
115102	W Susilo	373-62-1245	Australia

Relational Data Model - Integrity

Referential Integrity

- Before we look at referential integrity, let's look at the concept of a foreign key.
- A foreign key is an attribute or set of attribute within one relational table that matches the primary key of some relational table.

  For example, in the relational tables shown earlier (in slide 14), tuples in the referencing relational table Student have attribute Counselor (also known as foreign key attribute) that references the primary key attribute StaffId of the referenced relational table Staff.

Relational Data Model - Integrity

- Referential integrity requires that the RDBMS keep each foreign key consistent with its corresponding primary key.
- It is a constraint involving *two* relational tables.
- It is used to specify a *relationship* among tuples in two relational tables: the **referencing relational table** and the **referenced relational table**.

Relational Data Model - Integrity

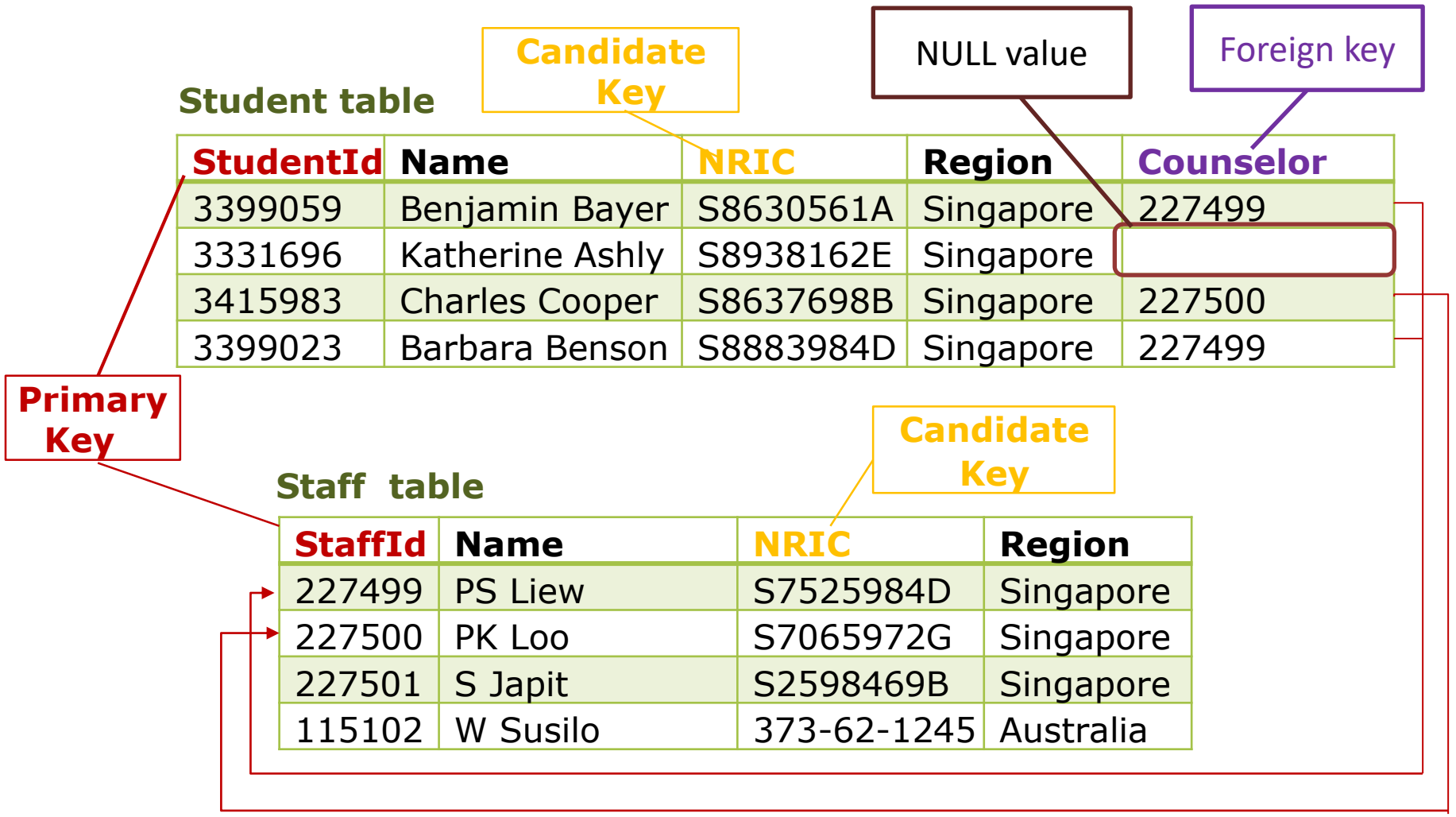
Statement of the Referential Integrity constraint

The value in the foreign key column (or columns) of the **referencing relational table** can be either:

1) a value of an existing primary key value of the corresponding primary key in the **referenced** relational table, or
2) a null.

In case (2), the foreign key in the referencing relational table should **NOT** be a part of its own primary key.

Relational Data Model - Integrity



Other Types of Constraints

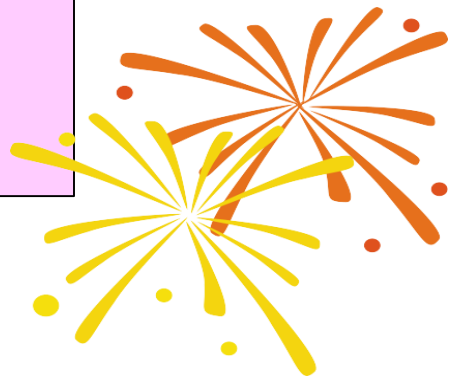
Semantic Integrity Constraints:

- based on application semantics and cannot be expressed by the model per se
- E.g., "the max. no. of hours per employee for all projects he or she works on is 56 hrs per week"
- A *constraint specification language* may have to be used to express these
- SQL-99 allows triggers and ASSERTIONS to allow for some of these (Note: Oracle does not support ASSERTIONS, but various type of constraints are implemented using triggers.)

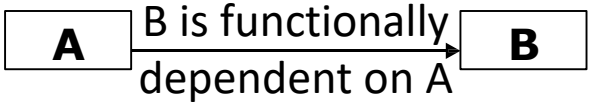
Functional Dependency

Functional Dependencies

The concept of functional dependencies is the basis for normal forms relations. It describes the relationship between attributes in relations.



For example, if A and B are attributes of relation R, B is functionally dependent on A, if each value of A in R is associated with exactly one value of B in R.



A is known as determinant.

Full Functional Dependency

Full functional dependency applies to relations with composite key.

Attribute B in relation R is fully functional dependent on attribute A (composite key) of R if it is **functionally dependent on A and NOT functionally dependent upon any subset of A.**

(SNum, SName) → BrNum

(SNum, PNum) → HoursWorked

Snum → BrNum

Fully functional dependency

Partial functional dependency

Transitive Dependency

Transitive dependency indicates a dependency in which if A, B and C are attributes of a relation R such that if $A \rightarrow B$ and $B \rightarrow C$ and B is neither a candidate key nor a subset of any key of the relation R, then C is transitively dependent on A via B.

$$Snum \rightarrow BrNum \; and$$
$$BrNum \rightarrow BrAddress$$

BrAddress is transitively dependent on SNum via BrNum

Process of Normalization – An example

38

Normal Forms

We will use example to show the process of the normalization:

A company obtains parts from a number of suppliers. Each supplier is located in one city. A city can have more than one supplier located there and each city has a status code associated with it. Each supplier may provide many parts. The company creates a simple relation to store this information that can be expressed in relational notation as

FIRST (s#, status, city, p#, qty)

Normal Forms

A relation is in **first normal form (1NF)** if and only if all intersection of each tuple and attribute contains one and only one value (atomic).

FIRST

s#	status	city	p#	qty
s1	20	London	p1	300
s1	20	London	p2	200
s1	20	London	p3	400
s1	20	London	p4	200
s1	20	London	p5	100
s1	20	London	p6	100
s2	10	Paris	p1	300
s2	10	Paris	p2	400
s3	10	Paris	p2	200
s4	20	London	p2	200
s4	20	London	p4	300
s4	20	London	p5	400

Normal Forms

A relation in 1NF will encounter update anomalies:

- **Insert anomalies** – The fact that a certain supplier (s5) is located in a particular city (Athens) cannot be added until they supplied a part.
- **Delete anomalies** – if a row is deleted, then not only is the information about quantity and part lost but also information about the supplier.
- **Update anomalies** – if supplier s1 moved from London to New York, then six rows would have to be updated with this new information.

Normal Forms

A relation is in second normal form (2NF) if it is in first normal form (1NF) and every non-primary-key attribute is fully functionally dependent on the primary key.

Every non-key attributes must be dependent upon the entire primary key.

FIRST

s#	status	city	p#	qty
s1	20	London	p1	300
s1	20	London	p2	200
s1	20	London	p3	400
s1	20	London	p4	200
s1	20	London	p5	100
s1	20	London	p6	100
s2	10	Paris	p1	300
s2	10	Paris	p2	400
s3	10	Paris	p2	200
s4	20	London	p2	200
s4	20	London	p4	300
s4	20	London	p5	400

Normal Forms

The relation FIRST is in $1NF$ but not in $2NF$ because status and city are functionally dependent upon only on the attribute $s\#$ of the composite key $(s\#, p\#)$.

FIRST

s#	status	city	p#	qty
s1	20	London	p1	300
s1	20	London	p2	200
s1	20	London	p3	400
s1	20	London	p4	200
s1	20	London	p5	100
s1	20	London	p6	100
s2	10	Paris	p1	300
s2	10	Paris	p2	400
s3	10	Paris	p2	200
s4	20	London	p2	200
s4	20	London	p4	300
s4	20	London	p5	400

$$s\# \rightarrow city, status$$

$$City \rightarrow status$$

$$(s\#, p\#) \rightarrow qty$$

Normal Forms

The process for transforming a 1NF relation to 2NF relation:

1. Identify any determinants other than the composite key and the attributes they determine.

$$s\# \rightarrow city, status$$
$$City \rightarrow status$$

Normal Forms

2. Create and name a new relation for each determinant and the unique attributes it determines.

SUPPLIER (s#, status, city)

Normal Forms

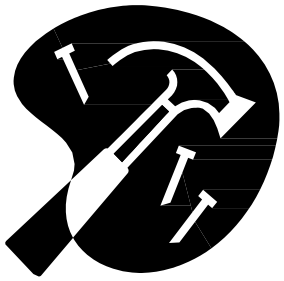
3. Moves the determined attributes from the original relation to the new relation. The determinant becomes the primary key of the new relation.

SUPPLIER

S#	Status	City
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

Normal Forms

4. Delete the attribute you just moved from the original relation except for the determinant which will serve as a foreign key.



SUPPLIER

S#	Status	City
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

FIRST

S#	P#	qty
S1	p1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Normal Forms

5. The original relation may be renamed to maintain semantic meaning.

SUPPLIER

S#	Status	City
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

PARTS

S#	P#	qty
S1	p1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Normal Forms

The process for transforming a 1NF relation to 2NF relation:

1. Identify any determinants other than the composite key and the attributes they determine.
2. Create and name a new relation for each determinant and the unique attributes it determines.
3. Moves the determined attributes from the original relation to the
   new relation. The determinant becomes the primary key of the new relation.
4. Delete the attribute you just moved from the original relation except
   for the determinant which will serve as a foreign key.
5. The original relation may be renamed to maintain semantic meaning.

Normal Forms

Relations in 2NF still contain modification anomalies:

- **Insert anomalies**. The fact that a particular city has a certain status (Rome has a status of 50) cannot be inserted until there is a supplier in the city.
- **Delete anomalies.** Deleting any row in SUPPLIER destroys the status information about the city as well as the association between supplier and city.

Normal Forms

A relation is in 3NF if it is already in 2NF and in which no non-primary-key attribute is transitively dependent upon its primary key.

A transitive dependency occurs when a non-key attribute that is a determinant of the primary key is the determinant of other attributes.

SUPPLIER

S#	Status	City
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

Normal Forms

The process of transforming a relation into 3NF is:

1. Identify any determinants, other than the primary key, and the attributes they determine.

$$SUPPLIER.S\# \rightarrow SUPPLIER.STATUS$$
$$SUPPLIER.S\# \rightarrow SUPPLIER.CITY$$
$$SUPPLIER.CITY \rightarrow SUPPLIER.STATUS$$

$$SUPPLIER.CITY \rightarrow SUPPLIER.STATUS$$

Normal Forms

2. Create and name a new relation for each determinant and the unique attributes it determine.

CITYSTATUS(CITY, STATUS)

Normal Forms

3. Move the determined attributes from the original relation to the new relation. The determinant becomes the primary key of the new relation.

CITYSTATUS

city	status
London	20
Paris	10
Athens	30
Rome	50

Normal Forms

4. Delete the attributes you just moved from the original relation except for the determinant that will serve as a foreign key.

CITYSTATUS

city	status
London	20
Paris	10
Athens	30
Rome	50

SUPPLIER

S#	City
S1	London
S2	Paris
S3	Paris
S4	London
S5	Athens

Normal Forms

5. The original relation may be renamed to maintain semantic meaning.

CITYSTATUS

city	status
London	20
Paris	10
Athens	30
Rome	50

SUPPLIERCITY

S#	City
S1	London
S2	Paris
S3	Paris
S4	London
S5	Athens

Normal Forms

3NF relations should have removed or minimized update anomalies.

- **Insert Anomalies**. Facts about the status of a city, Rome, have a status of 50 can be added even though there is no supplier in that city. Likewise, facts about new suppliers can be added even though they have not yet supplied parts.
- **Delete Anomalies**. Information about parts supplied can be deleted without destroying information about a supplier or a city.
- **Update Anomalies**. The location of a supplier or the status of a city requires modifying only one row.

Normal Forms

Unfortunately not all 3NF relations are free from update anomalies.

- Consider the following CLIENTINTERVIEW Relation:

CLIENTINTERVIEW

ClientNum	InterviewDate	InterviewTime	StaffNum	RoomNum
CR76	13-Nov-05	10:30	SG5	G101
CR56	13-Nov-05	12:00	SG5	G101
CR74	14-Nov-05	12:00	SG37	G102
CR56	01-Dec-05	10:30	SG5	G102

Normal Forms

- The CLIENTINTERVIEW relation does not have any partial or transitive dependencies on the primary key (ClientNum, InterviewDate) and is therefore in 3NF.
- However, this relation suffer an update anomalies. For if we try to change the room number a staff is using for the interview on a particular date.

CLIENTINTERVIEW

ClientNum	**InterviewDate**	**InterviewTime**	**StaffNum**	**RoomNum**
CR76	13-Nov-05	10:30	SG5	G101
CR56	13-Nov-05	12:00	SG5	G101
CR74	14-Nov-05	12:00	SG37	G102
CR56	01-Dec-05	10:30	SG5	G102

Normal Forms

- Analyzing the CLIENTINTERVIEW relation, we notice that there are three functional dependencies as follows:

(ClientNum, InterviewDate) → InterviewTime, StaffNum, RoomNum

(StaffNum, InterviewDate, InterviewTime) → ClientNum

(StaffNum, InterviewDate) → RoomNum

- From the above dependencies, only (ClientNum, InterviewDate) and (StaffNum, InterviewDate, InterviewTime) are candidate key.
- (StaffNum, InterviewDate) is not a candidate key.

Normal Forms

- To avoid update anomalies in the CLIENTINTERVIEW relation, we can split the relation into two relations as follow:

Normal Forms

INTERVIEW

ClientNum	**InterviewDate**	**InterviewTime**	**StaffNum**
CR76	13-Nov-05	10:30	SG5
CR56	13-Nov-05	12:00	SG5
CR74	13-Nov-05	12:00	SG37
CR56	01-Dec-05	10:30	SG5

STAFFROOM

StaffNum	**InterviewDate**	**RoomNum**
SG5	13-Nov-05	G101
SG37	13-Nov-05	G102
SG5	01-Dec-05	G102

Normal Forms

- The relations INTERVIEW and STAFFROOM shown above is said to be in Boyce-Codd Normal Form.

The definition of Boyce-Codd Normal Form (BCNF) states that a relation is in BCNF if and only if every determinant is a candidate key.

(ClientNum, InterviewDate → InterviewTime, StaffNum, RoomNum

(StaffNum, InterviewDate, InterviewTime) → ClientNum

(StaffNum, InterviewDate) → RoomNum

This determinant is not a candidate key.

Normal Forms

- Note that BCNF does not state that the relation must first be in 3NF and then satisfy an additional constraint.
- Also note that every relation in BCNF is also in 3NF, however, a relation in 3NF is not necessary in BCNF.
- The next two examples will give you a better understanding on BCNF.

Normal Forms

- Violation of BCNF is quite rare, since it may only happen under specific conditions.
- The potential to violate BCNF may occur in a relation that:
  - Contains two (or more) composite candidate keys,
  - Which overlap and share at least one attribute in common.