Chapter 14
Normalization

1

COMP211

Objectives

- The purpose of normalization.
- Data Redundancy and Update Anomalies
- The concept of functional dependency, which describes the relationship between attributes.
- The concept of transitive dependency.
- The process of normalization.

2

COMP211

Purpose of Normalization

- When designing a database, the main objective is to create an accurate representation of the data, relationships between the data, and constraints on the data that is relevant to the enterprise.
- One of the database design techniques we have covered is ER modeling.
- This chapter introduces techniques to evaluate the table structures to control data redundancies, thereby avoiding data anomalies.
- **Normalization** is a technique for producing a set of suitable relations *with desirable properties*, given the data requirements of an enterprise.

Characteristics of a suitable set of relations

- Characteristics of a suitable set of relations include:
  - the *minimal* number of attributes necessary to support the data requirements of the enterprise;
  - attributes with a close logical relationship are found in the same relation;
  - *minimal* redundancy, with each attribute represented only once, with the important exception of attributes that form all or part of foreign keys, which are essential for the joining of related relations.

Benefits of a suitable set of relations

- The benefits of using a database that has a suitable set of relations is that the database will be:
  - easier for the user to access and maintain the data;
  - take up minimal storage space on the computer.

Example on benefits of a suitable set of relations

Compare the storage space as a result of minimal redundancy.

If room for MENG111 is changed from 560 to 606, compare the maintenance cost.

Student	CourseCode	CourseName	Teacher	Room
Dora	COMP111	Computer	Peter	610
Dora	MATH111	Maths	Thomas	851
Susan	MENG111	English	Janice	560
Rachael	MENG111	English	Janice	560
Monica	MATH111	Maths	Thomas	851
Monica	MENG111	English	Janice	560

Foreign key

Student	CourseCode
Dora	COMP111
Dora	MATH111
Susan	MENG111
Rachael	MENG111
Monica	MATH111
Monica	MENG111

Primary key

CourseCode	CourseName	Teacher	Room
COMP111	Computer	Peter	610
MATH111	Maths	Thomas	851
MENG111	English	Janice	560

Objectives

- The purpose of normalization.
- Data Redundancy and Update Anomalies.
- The concept of functional dependency, which describes the relationship between attributes.
- The concept of transitive dependency.
- The process of normalization.

Problems with Data Redundancy – An Example

- Problems associated with data redundancy are illustrated by comparing the Staff and Branch relations with the StaffBranch relation.
- In the StaffBranch relation, staffNo is the primary key.

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Problems with Data Redundancy – An Example (Cont'd)

- StaffBranch relation has redundant data; the details of a branch are repeated for every member of staff.
- Relations that contain redundant information may potentially suffer from update anomalies.
- Types of update anomalies include
  - Insertion
  - Deletion
  - Modification

9

COMP211

Insertion Anomalies

- Using the StaffBranch relation, there are 2 main types of insertion anomaly:
  - To insert details of new members of staff, details of the branch at which the staff are located **must be included correctly** so that the details are **consistent** with the values in other tuples of the relation.
  - We cannot enter a tuple for a new branch that currently has no members of staff because staffNo is the primary key, attempting to enter nulls for staffNo violates entity integrity.

10

COMP211

Deletion Anomalies

- Using the StaffBranch relation, there is one type of deletion anomaly:
  - If we delete a tuple that represents the last member of staff located at a branch, the details of that branch are lost from the database as well.

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
~~SA9~~	~~Mary Howe~~	~~Assistant~~	~~9000~~	~~B007~~	~~16 Argyll St, Aberdeen~~
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Modification Anomalies

- Using the StaffBranch relation, there is one type of modification anomaly:
  - Updating the address of a branch requires updating the tuples of all staff located at that branch, otherwise, data will become inconsistent.

Decomposing the original relation

- We can avoid all these anomalies by decomposing the original relation into the Staff and Branch relations.
- Here, the branch information appears only once for each branch in the Branch relation and only the branch number (branchNo) is repeated in the Staff relation.

Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

Branch

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

13

COMP211

Lossless-join Property

- In decomposing a larger relation into smaller relations, ***lossless-join*** property ensures that any instance of the original relation can be identified from corresponding instances in the smaller relations.

14

COMP211

Objectives

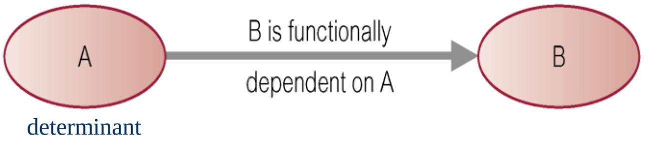
- The purpose of normalization.
- Data Redundancy and Update Anomalies
- The concept of functional dependency, which describes the relationship between attributes.
- The concept of transitive dependency.
- The process of normalization.

Functional Dependencies

- An important concept associated with normalization.
- Functional dependency describes relationship between attributes.
- For example, if A and B are attributes of relation R, B is functionally dependent on A (denoted A $\rightarrow$ B), **if each value of A in R is associated with exactly one value of B in R**. (A and B may each consist of one or more attributes.)

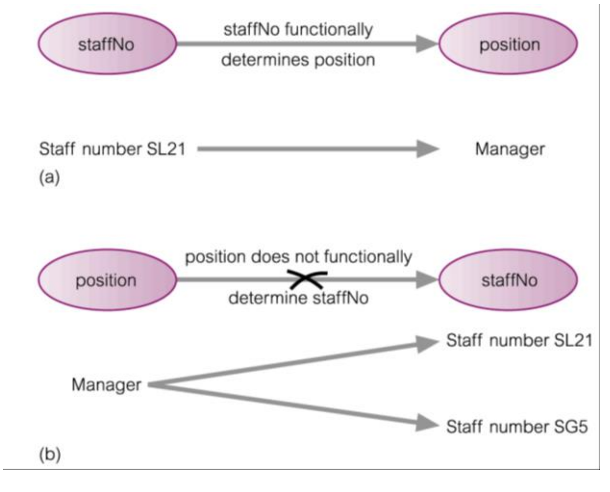
Determinant of Functional Dependencies

- When a functional dependency is present, it is specified as a **constraint** between the attributes.
- Diagrammatic representation.



- The ***determinant*** of a functional dependency refers to the attribute or group of attributes on the left-hand side of the arrow of a functional dependency.

An Example of a Functional Dependency



The relationship between staffNo and position is one-to-one.

The relationship between position and staffNo is one-to-many.

Example of a Functional Dependency that Holds for All Time

- Consider the values shown in staffNo and sName attributes of the Staff relation.

Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

- Based on sample data, the following functional dependencies appear to hold.

staffNo → sName

sName → staffNo

Example of a Functional Dependency that Holds for All Time

- However, the only functional dependency **that remains true for all possible values** for the staffNo and sName attributes of the Staff relation is:

staffNo → sName

Characteristics of Functional Dependencies

- Main characteristics of functional dependencies used in normalization:
  - There is a *one-to-one* relationship between the attribute(s) on the left-hand side (determinant) and those on the right-hand side of a functional dependency.
  - Is True for *all* time.
  - The determinant has the ***minimal*** number of attributes necessary to maintain the dependency with the attribute(s) on the right hand-side (i.e. **full functional dependency**).
  - Used to identify primary key for a relation.

Objectives

- The purpose of normalization.
- Data Redundancy and Update Anomalies
- The concept of functional dependency, which describes the relationship between attributes.
- The concept of transitive dependency.
- The process of normalization.

Transitive Dependencies

- Important to recognize a transitive dependency because its existence in a relation can potentially cause update anomalies.
- Transitive dependency describes a condition where A, B, and C are attributes of a relation such that if A → B and B → C, then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C).

Example of Transitive Dependency

- List out all the functional dependencies in the StaffBranch relation.

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

- Identify the primary key for this relation.
- Can you identify any transitive dependency?

Example of Transitive Dependency

staffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

- staffNo is the primary key for this relation because all other attributes of the relation are functionally dependent on staffNo.
- **Transitive dependency, branchNo → bAddress** exists on staffNo via branchNo.
- staffNo attribute functionally determines the bAddress via the branchNo attribute

Objectives

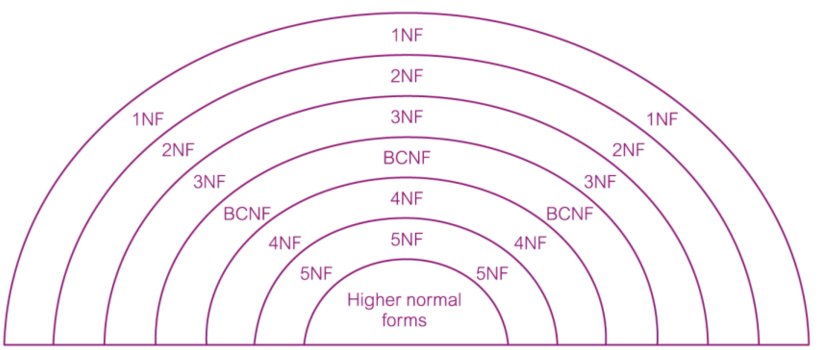
- The purpose of normalization.
- Data Redundancy and Update Anomalies
- The concept of functional dependency, which describes the relationship between attributes.
- The concept of transitive dependency.
- The process of normalization.

The Process of Normalization

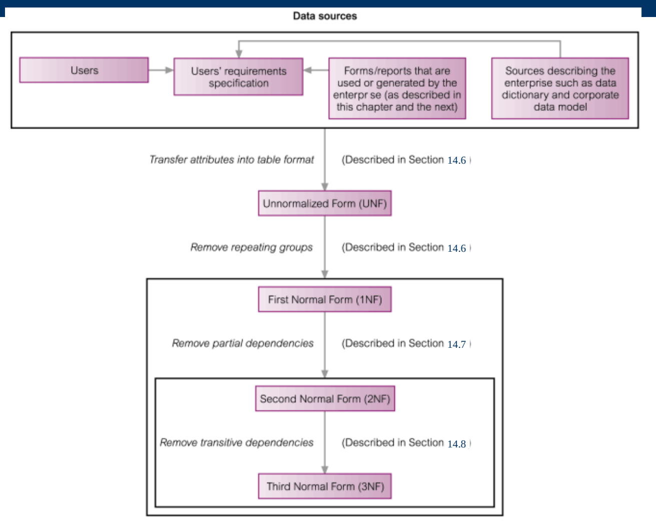
- Formal technique for analyzing a relation based on its primary key (or candidate keys) and functional dependencies.
- Often executed as a series of steps. Each step corresponds to a specific normal form, which has known properties.
- As normalization proceeds, the relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.

The Process of Normalization

- To avoid update anomalies, it is generally recommended that we proceed to at least Third Normal Form (3NF).



The Process of Normalization



Unnormalized Form (UNF)

- Transfer the data from the information source (e.g. form) into table format with columns and rows.

ClientRental

clientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	John Kay	PG4	6 Lawrence St, Glasgow	1-Jul 07	31-Aug-08	350	CO40	Tina Murphy
		PG16	5 Novar Dr, Glasgow	1-Sep-08	1-Sep-09	450	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Glasgow	1-Sep-06	10-June-07	350	CO40	Tina Murphy
		PG36	2 Manor Rd, Glasgow	10-Oct-07	1-Dec-08	375	CO93	Tony Shaw
		PG16	5 Novar Dr, Glasgow	1-Nov-09	10-Aug-10	450	CO93	Tony Shaw

First Normal Form (1NF)

- 1NF: **A relation in which the intersection of each row and column contains one and only one value**.

UNF to 1NF

Step 1: Identify key of the unnormalized table.

Step 2: Identify the repeating group(s) in the unnormalized table, i.e. those having more than one values for the corresponding key value.

Step 3: Remove the repeating group by
- Entering appropriate data into the empty columns of rows containing the repeating data ('flattening' the table), **OR**
- Placing the repeating data along with a copy of the original key attribute(s) into a separate relation.

UNF to 1NF: Remove repeating groups

UNF:

- ClientRental (clientNo, cName, propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

- **Repeating group:** (propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

1NF:

- PropertyRentalOwner(clientNo, propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName)
- Client (clientNo, cName)

UNF to 1NF

- 'Flattening' the table introduces more redundancy into the original UNF table. (this will be your class practice)
- Approach 2 creates 2 or more relations.

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

PropertyRentalOwner

clientNo	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	6 Lawrence St, Glasgow	1-Jul-07	31-Aug-08	350	CO40	Tina Murphy
CR76	PG16	5 Novar Dr, Glasgow	1-Sep-08	1-Sep-09	450	CO93	Tony Shaw
CR56	PG4	6 Lawrence St, Glasgow	1-Sep-06	10-Jun-07	350	CO40	Tina Murphy
CR56	PG36	2 Manor Rd, Glasgow	10-Oct-07	1-Dec-08	375	CO93	Tony Shaw
CR56	PG16	5 Novar Dr, Glasgow	1-Nov-09	10-Aug-10	450	CO93	Tony Shaw

- However, no matter which initial approach is taken, the original UNF table will be normalized into the same set of 3NF relations.

Second Normal Form (2NF)

- Based on the concept of full functional dependency.
- Full functional dependency indicates that if
  - A and B are attributes of a relation,
  - B is fully dependent on A if B is functionally dependent on A but not on any proper subset of A.
- Applies to relations with composite primary key.
- **A relation with a single-attribute primary key is automatically in at least 2NF**.

35

COMP211

Second Normal Form (2NF)

- 2NF: **A relation that is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key**.

36

COMP211

1NF to 2NF

- Step 1: Identify the primary key for the 1NF relation.
- Step 2: Identify the partial dependencies in the relation(s), if any.
- Step 3: If partial dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their determinant. Otherwise, the relation is already in 2NF.

37

COMP211

1NF to 2NF: Remove partial dependencies

1NF:

- PropertyRentalOwner(clientNo, propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName)
  - **Partial dependencies?**
    - clientNO -> (NO)
    - propertyNo -> pAddress, rent, ownerNo, oName
    - rentStart -> (NO)
- Client (clientNo, cName)
  - **Partial dependencies?** (NO, because the primary key is single attribute.)

38

COMP211

1NF to 2NF

2NF:

- Client (clientNo, cName)
- PropertyOwner (propertyNo, pAddress, rent, ownerNo, oName)
- Rental(clientNo, propertyNo, rentStart, rentFinish)

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-07	31-Aug-08
CR76	PG16	1-Sep-08	1-Sep-07
CR56	PG4	1-Sep-06	10-Jun-07
CR56	PG36	10-Oct-07	1-Dec-08
CR56	PG16	1-Nov-09	10-Aug-10

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	375	CO93	Tony Shaw

Figure 14.14 Second normal form relations derived from the ClientRental relation.

Note: *In fact, it is possible to have rent in both PropertyOwner and Rental. The rent in PropertyForRent records the current rent of a property whereas the rent in Rental is the historical record of the rent in different periods of time.*

39

COMP211

Third Normal Form (3NF)

- Based on the concept of transitive dependency.
- Transitive Dependency is a condition where
  - A, B and C are attributes of a relation such that if $A \rightarrow B$ and $B \rightarrow C$,
  - then C is transitively dependent on A through B. (Provided that A is not functionally dependent on B or C).

40

COMP211

Third Normal Form (3NF)

- 3NF: **A relation that is in 1NF and 2NF and in which no non-primary-key attribute is transitively dependent on the primary key**.

2NF to 3NF

- Step 1: Identify the primary key in the 2NF relation.
- Step 2: Identify transitive dependencies in the relation(s), if any.
- Step 3: If transitive dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their determinant. Otherwise, the relation is already in 3NF.

2NF to 3NF: Remove transitive dependencies

2NF:

- Client (clientNo, cName)
  - **Transitive dependencies?** (NO)
- PropertyOwner (propertyNo, pAddress, rent, ownerNo, oName)
  - **Transitive dependencies?**
    - ownerNo -> oName
- Rental(clientNo, propertyNo, rentStart, rentFinish)
  - **Transitive dependencies?** (NO)

2NF to 3NF

3NF:

- Client (clientNo, cName)
- Rental(clientNo, propertyNo, rentStart, rentFinish)
- Owner (ownerNo, oName)
- PropertyForRent (propertyNo, pAddress , rent, ownerNo)

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-07	31-Aug-08
CR76	PG16	1-Sep-08	1-Sep-09
CR56	PG4	1-Sep-06	10-Jun-07
CR56	PG36	10-Oct-07	1-Dec-08
CR56	PG16	1-Nov-09	10-Aug-10

PropertyForRent

propertyNo	pAddress	rent	ownerNo
PG4	6 Lawrence St, Glasgow	350	CO40
PG16	5 Novar Dr, Glasgow	450	CO93
PG36	2 Manor Rd, Glasgow	375	CO93

Owner

ownerNo	oName
CO40	Tina Murphy
CO93	Tony Shaw

Figure 14.17 A summary of the 3NF relations derived from the ClientRental relation.

Why the need for Normal forms that go beyond 3NF?

- Relations in 3NF are normally sufficiently well structured to prevent the problems associated with data redundancy.
- However, later normal forms were created to identify relatively rare problems with relations, that, if not corrected, might result in undesirable data redundancy.

Boyce-Codd Normal Form (BCNF)

- Based on functional dependencies that take into account all candidate keys in a relation, however BCNF also has additional constraints compared with the general definition of 3NF.
- Boyce–Codd normal form (BCNF)
  - **A relation is in BCNF if and only if every determinant is a candidate key.**

3NF to BCNF

- Step 1: Identify functional dependencies in the relation.
- Step 2: Identify all the determinants and making sure that they are candidate keys.
- Step 3: If there are determinants not being candidate key, remove them by placing them in a new relation.

47

COMP211

3NF to BCNF

3NF:

- Client (clientNo, cName)
  - clientNo -> cName
- Rental(clientNo, propertyNo, rentStart, rentFinish)
  - clientNo, propertyNo, rentStart -> rentFinish
- Owner (ownerNo, oName)
  - ownerNo -> oName
- PropertyForRent (propertyNo, pAddress, rent, ownerNo)
  - propertyNo -> pAddress, rent, ownerNo

48

COMP211

3NF to BCNF

BCNF:
- Client (<u>clientNo</u>, cName)
- Rental(<u>clientNo</u>, <u>propertyNo</u>, <u>rentStart</u>, rentFinish)
- Owner (<u>ownerNo</u>, oName)
- PropertyForRent (<u>propertyNo</u>, pAddress, rent, ownerNo)

- All the tables above are in BCNF because all of the determinants are candidate keys.

49

COMP211

Summary

- Normalization is a process for evaluating and correcting table structures to **minimize data redundancies**, thereby reducing the likelihood of **data anomalies**.
- A properly normalized set of table structures is less complicated to use than an un-normalized set.

50

COMP211