Database Systems 2

Lecture 17

Functional Dependency

This Lecture - Objectives

- Purpose of normalization.
- Problems associated with redundant data.
- Identification of various types of update anomalies such as insertion, deletion, and modification anomalies.
- Functional Dependencies
- Dependency Diagrams
- Identifying Keys

Next Lecture - Objectives

- How functional dependencies can be used to group attributes into relations that are in a known normal form.
- How to undertake process of normalization.
- How to identify most commonly used normal forms, namely 1NF, 2NF, 3NF, and Boyce–Codd normal form (BCNF).

Normalization

The main objective in developing a logical data model for relational database systems is to create an accurate representation of the data, its relationships, and constraints.

To achieve this objective, we must identify a suitable set of relations.

Up to now, we have been analysing the scenario in a top-down way. This approach is bottom-up – it starts with the data.

Remember:

- Relation = Table
- Tuple = Row
- Attribute = Field

Normalization

The four most commonly used normal forms are:

- First Normal Form (1NF),
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce–Codd normal form (BCNF).

We are not looking at relationships between tables any more.

We are looking at the functional dependencies between the attributes of a single table.

A relation can be normalized to a specific form to prevent possible occurrence of update anomalies.

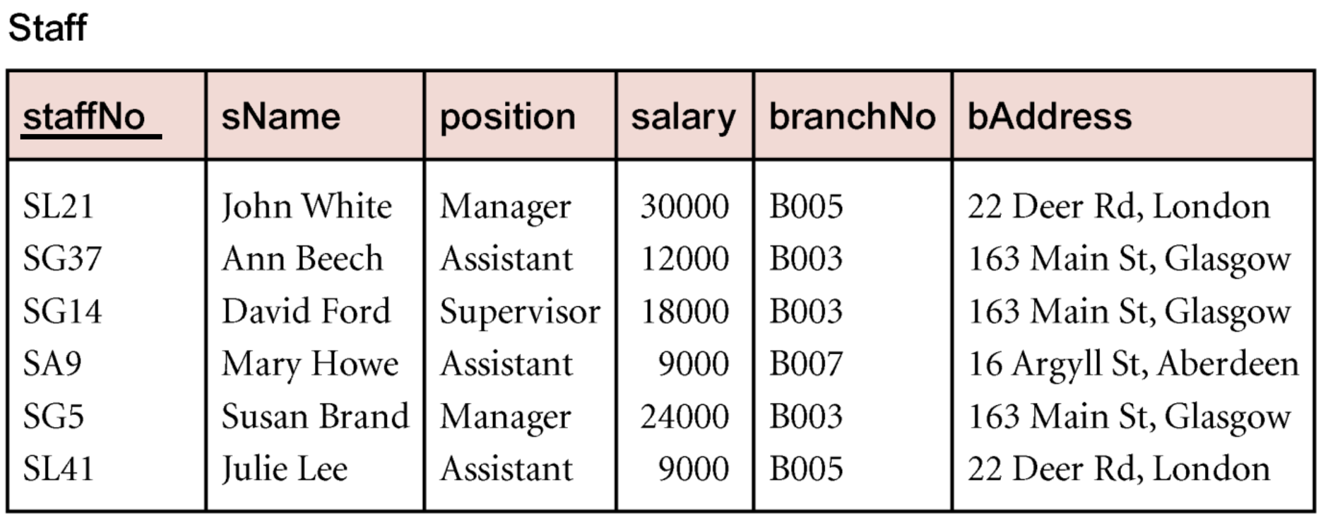
Data Redundancy

A major aim of relational database design is to group attributes into relations to minimize data redundancy and reduce file storage space required by base relations.

Base relations are the ones from which all user views (ie manager, supervisor) are derived.

Problems associated with data redundancy are illustrated by looking at the following example:

Staff Table



Staff

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

Why is this not a good way of storing this data?
Where are the redundant copies of the data?
Which attributes depend on which other attributes?

How many entities are buried in this table?

Here is another example

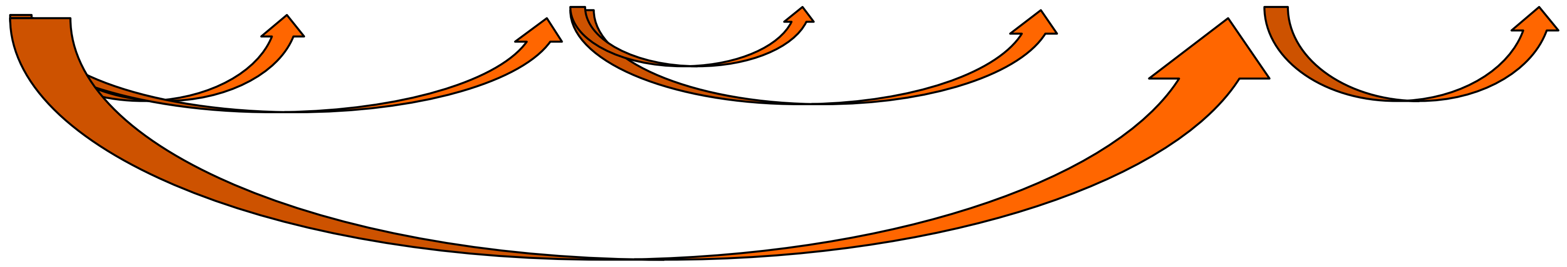
StudentNo	StudentName	UnitName	Teacher	Room	CourseNo	CourseName
S01	Adams	Java Prog	Stanley	160	C01	CompNet
S02	Brown	Databases	Wilson	164	C02	BusComp
S03	Corden	Project	Scott	75	C01	CompNet
S04	Davies	Java Prog	Stanley	160	C01	CompNet
S05	Eastman	Project	Scott	75	C01	CompNet
S06	Forsyth	Databases	Wilson	164	C02	BusComp
S07	Grantham	Databases	Wilson	164	C02	BusComp

What are the dependencies?

How many entities? What are the relationships?

Here is another example

StudentNo	StudentName	UnitName	Teacher	Room	CourseNo	CourseName
S01	Adams	Java Prog	Stanley	160	C01	CompNet
S02	Brown	Databases	Wilson	164	C02	BusComp
S03	Corden	Project	Scott	75	C01	CompNet
S04	Davies	Java Prog	Stanley	160	C01	CompNet
S05	Eastman	Project	Scott	75	C01	CompNet
S06	Forsyth	Databases	Wilson	164	C02	BusComp
S07	Grantham	Databases	Wilson	164	C02	BusComp



Update Anomalies

A table that has a 1:M relationship embedded in it is likely to contain redundant information.

Tables that contain redundant information may potentially suffer from update anomalies.

There are various types of update anomaly:

- Insertion
- Deletion
- Modification.

Insertion Anomalies

Staff

<u>staffNo</u>	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

New member of staff joins branch B005
- Insert new row into Staff table
- Type wrong address: 163 Main St, Glasgow.
- Database is now inconsistent!

Establish new branch with no members of staff
- B008, 57 Princes St, Edinburgh
- No staff members, so staffNo must be NULL
- But staffNo is the primary key of the Staff table, so cannot be NULL!

Deletion Anomaly

Staff

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

Mary Howe, staffNo SA9, leaves the company

- Delete the appropriate row of Staff
- This also deletes details of branch B007 where Mary Howe works
- But no-one else works at branch B007, so we no longer know the address of this branch!

Modification Anomaly

Staff

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

Branch B003 has transferred to a new location

- New address is 145 Main St, Glasgow
- Must change *three* rows of the Staff relation
- What if we miss one?

Data Redundancy

Staff relation contains redundant data: details of a branch are repeated for every member of staff.

We have effectively got a 1:M relationship embedded in one table.

This is what causes data redundancy, and this is what can give rise to update anomalies.

This is what normalisation is designed to prevent.

What would be a better way of storing this data?

Data Redundancy

Staff

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

Lossless-join and Dependency Preservation Properties

If we have decomposed one relation into two smaller relations, we must preserve:

- ***Lossless-join property***
  enables us to find any instance of original relation from corresponding instances in the smaller relations.

- ***Dependency preservation property***
  enables us to enforce a constraint on original relation by enforcing some constraint on each of the smaller relations.

Questions

Summer 2014	Q1 a, c
Summer 2013	Q2 a

Functional Dependency

Main concept associated with normalization.

Essentially, it is concerned with identifying possible primary key attributes.

<u>Functional Dependency</u>

- Describes relationships between attributes in a relation.

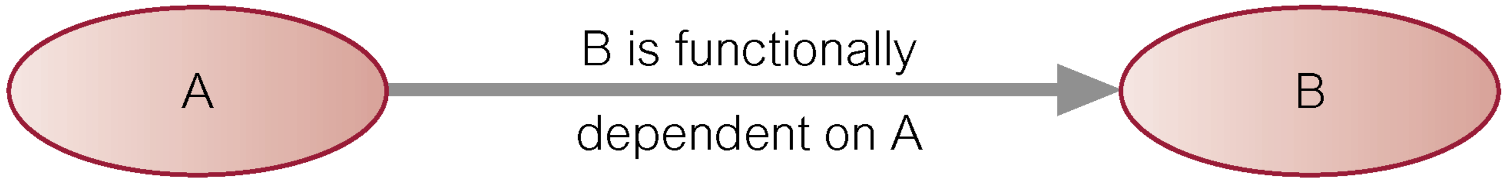
- If A and B are attributes of relation R,
  B is functionally dependent on A, (denoted $A \rightarrow B$),
  if each value of A is associated with exactly one value of B.

Functional Dependency

Property of the meaning (or semantics) of the attributes in a relation.

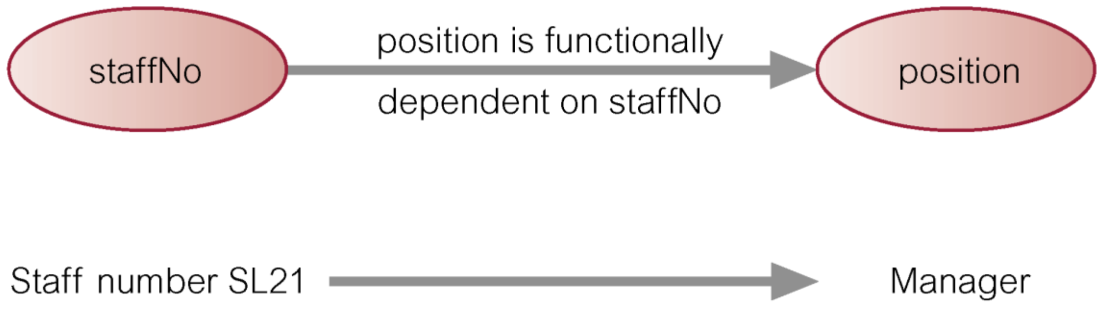
Diagrammatic representation:

Note – this is <u>not</u> an ERD



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

Example - Functional Dependency



Identify the dependencies in this table

staffNo	sName	position	salary	branchNo	bAddress	Tel_No
SL21	John White	Manager	30000	B005	22 Deer Rd, London	0171-886-1212
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow	0141-339-2178
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow	0141-339-2178
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen	01224-67125
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow	0141-339-2178
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London	0171-886-1212

Try starting from the right hand end.

Functional Dependencies of the Staff_Branch Relation

Staff_No ➔ SName
Staff_No ➔ SAddress
Staff_No ➔ Position
Staff_No ➔ Salary
Staff_No ➔ Branch_No
Staff_No ➔ Baddress
Staff_No ➔ Tel_No

Branch_No ➔ Baddress
Branch_No ➔ Tel_No
Baddress ➔ Branch_No
Baddress ➔ Tel_No
Tel_No ➔ Branch_No
Tel_No ➔ Baddress

Functional Dependency

Complete set of functional dependencies for a given relation can be very large.

Important to find an approach that can reduce set to a manageable size.

Need to identify set of functional dependencies (X) for a relation that is:

- smaller than complete set of functional dependencies (Y) for that relation and
- has property that every functional dependency in Y is implied by functional dependencies in X.

Functional Dependency

Main characteristics of functional dependencies used in normalization:

- have a 1:1 relationship between attribute(s) on left and right-hand side of a dependency;

- hold for all time;

- are nontrivial.

Staff_No ➔ SName, SAddress, Position, Salary, Branch_No, Baddress, Tel_No

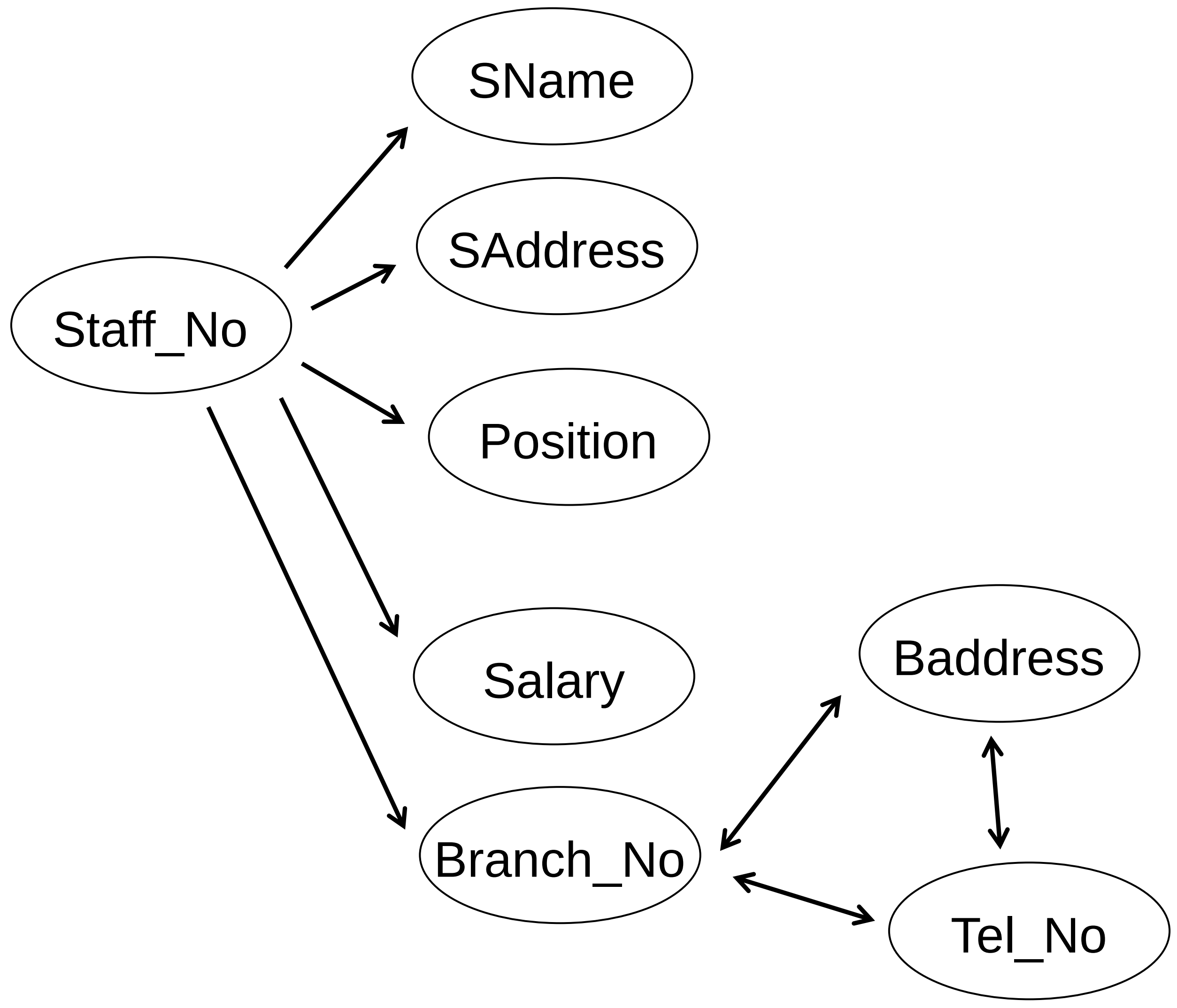
Branch_No ➔ Baddress, Tel_No

Baddress ➔ Branch_No, Tel_No

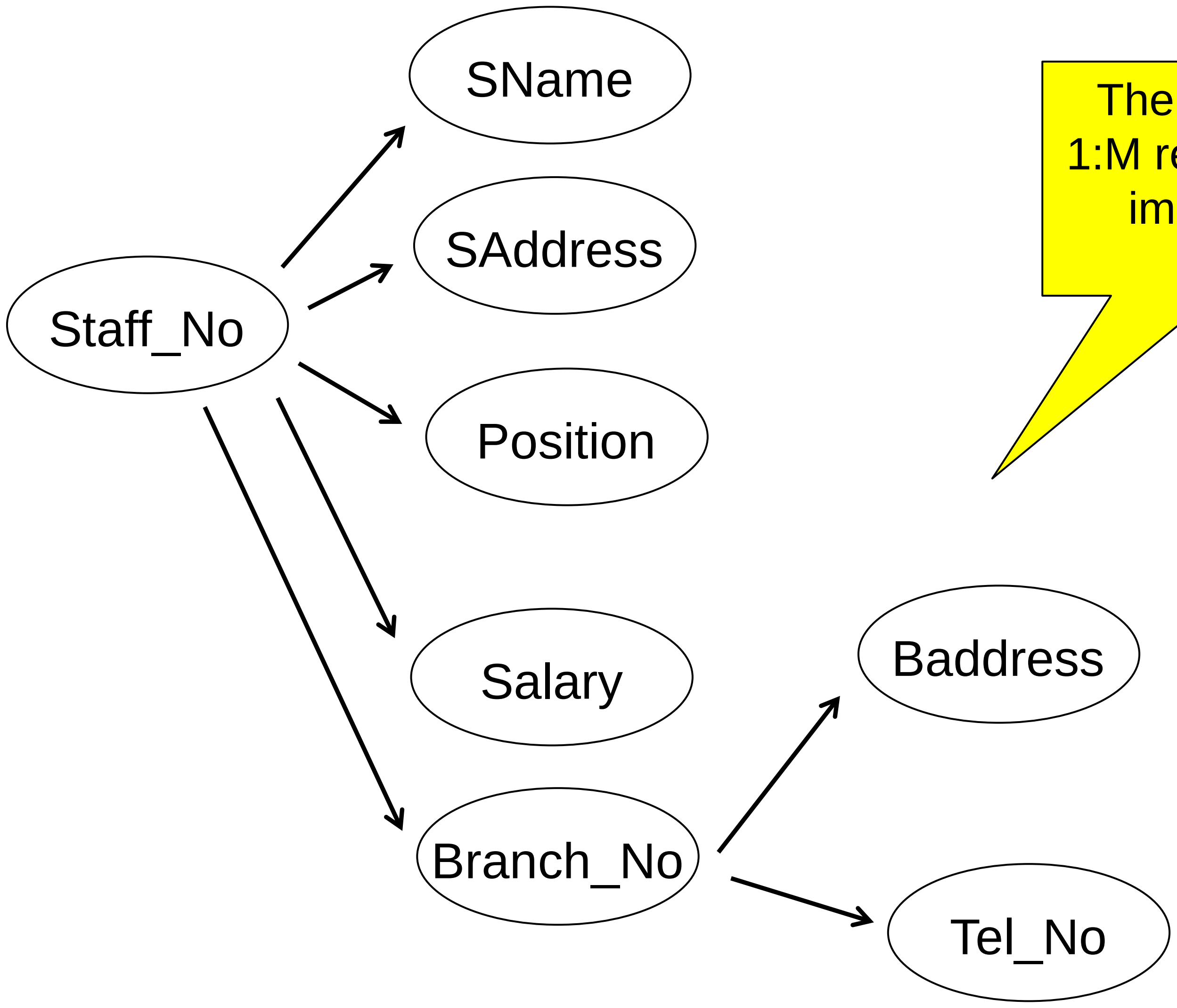
Tel_No ➔ Branch_No, Baddress

Which of these possible keys will hold for all time?

Dependency Diagram



Dependency Diagram



The embedded 1:M relationship is immediately visible

Identifying Candidate Keys

A candidate key is an attribute, (or set of attributes,) that uniquely identifies a row

- Must be *irreducible (no bigger than it needs to be)*
- *No part of a candidate can ever be NULL*

An attribute A that functionally determines *every other* attribute of the relation is a candidate key

- Each value of A determines exactly one value of each of the other attributes
- So each value of A must identify a whole row

Identifying Primary Keys

A primary key is a candidate key chosen to identify rows uniquely within a table

- Other candidate keys called *alternate keys*

Some guidelines on choosing the primary key

- Pick the candidate key with fewest attributes
- Pick the candidate key with shortest length
- Pick the candidate key that makes most sense for the business!

There is only one candidate key for the Staff relation.

This Lecture - Objectives

- Purpose of normalization.
- Problems associated with redundant data.
- Identification of various types of update anomalies such as insertion, deletion, and modification anomalies.
- Functional Dependencies
- Dependency Diagrams
- Identifying Keys