Logical Database Design
Normalisation

Part - 1

Duration : 5 hrs

Detailed Syllabus

7.1 Introduction to data normalization and normal forms
- 7.1.1 What is normalization, Benefits of normalization, Normalization Rules
- 7.1.2 1NF, 2NF, 3NF and Higher NF.

7.2 First Normal Form
- 7.2.1 1NF, Why convert to 1NF, Conversion to 1NF;

7.3 Second Normal Form
- 7.3.1 2NF, Functional Dependence and Fully Functional Dependence, Why convert to 2NF, Conversion to 2NF

7.4 Third Normal Form
- 7.4.1 3NF, Transitive Dependence, Why convert to 3NF, Conversion to 3NF.

7.5 Normalization considerations
- 7.5.1 Good and bad decompositions
- 7.5.2 De-normalization
- 7.5.3 Multi-valued dependencies, Join dependencies

Normalization

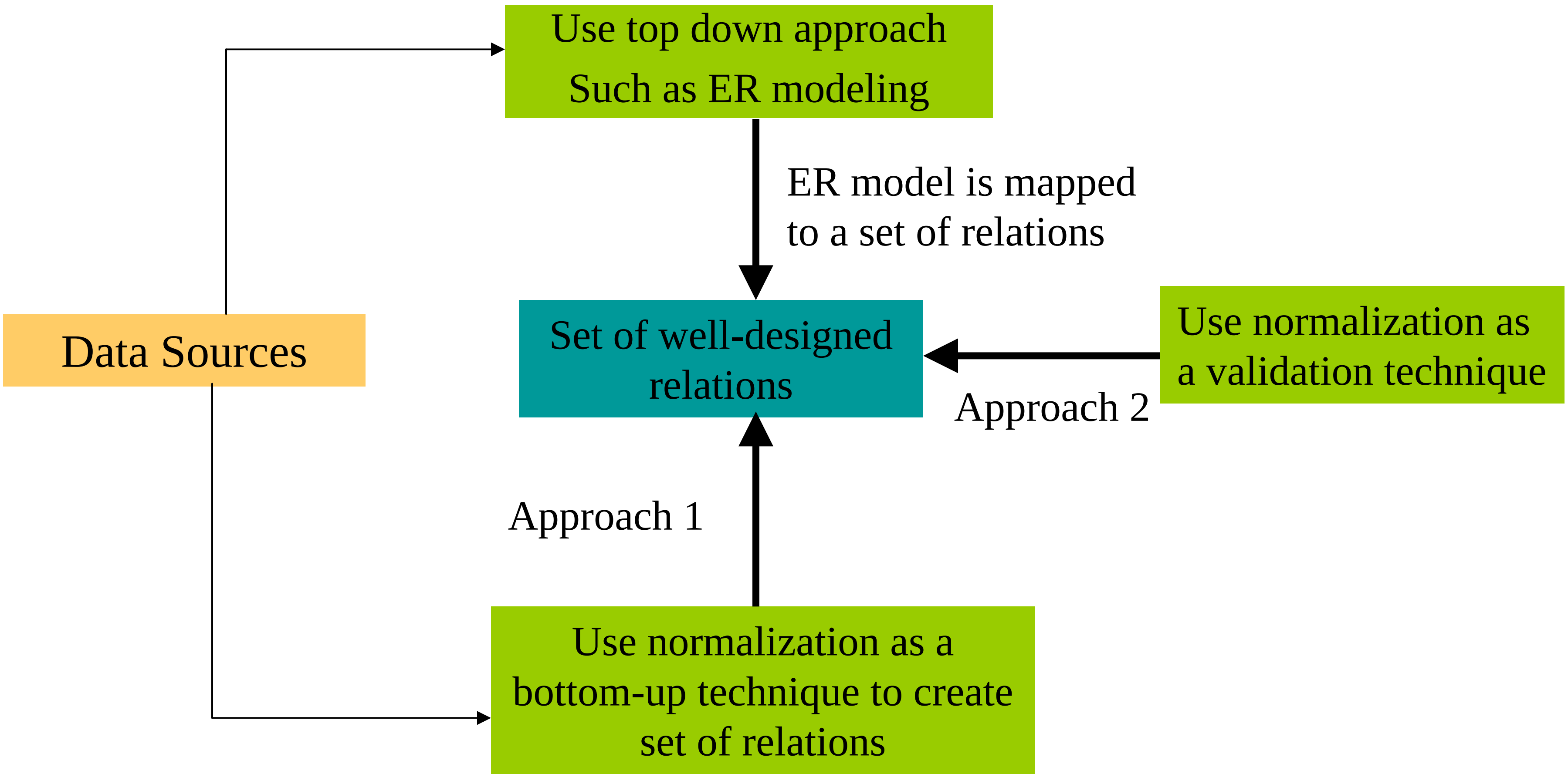
- Normalization is a database design technique which begins by examining the relationships (called functional dependencies) between attributes.

- Uses a series of tests (described as normal forms) to help identify the optimal grouping for these attributes to ultimately identify a set of suitable relations that supports the data requirements of the enterprise.

The purpose of normalization

- The purpose of normalization is to identify a suitable set of relations that support the data requirements of an enterprise. The characteristics of a suitable set of relations include the following:
  - The minimal number of attributes necessary to support the data requirements of the enterprise.
  - Attributes with a close logical relationship
  - Minimal redundancy

How Normalization Supports Database Design



Data Redundancy and Update Anomalies

- Aim is to group attributes into relations to minimize data redundancy.
- If this aim is achieved, the potential benefits for the implemented database include the following:
  - minimal number of update operations reducing data inconsistencies.
  - reduction in the file storage cost.

Data Redundancy and Update Anomalies

Employee {EmpId, Ename,BDate,Address, Dnumber}

Department {Dnumber, Dname, DmgrId}

- Emp_Dept

  {EmpId, Ename,BDate,Address,Dnumber, Dname, DmgrId}

Update Anomalies

- Update anomalies can be classified as insertion, deletion or modification anomalies.

- Insertion anomalies

  Can be differentiated into two types (illustrated using Emp_Dept)

  i. To insert a new employee tuple into Emp_Dept, we must include either the attribute values for the department that the employee works for or nulls.
  ii. It is difficult to insert a new department that has no employees.

Update Anomalies

- Deletion Anomalies
  - The problem of deletion anomalies is related to the second insertion anomaly situation.
  - If we delete from Emp_Dept the last employee working for a particular department, the information concerning that department is lost from the database.

Update Anomalies

- Modification Anomalies

In Emp_Dept, if we change the value of one of the attributes of a particular department, we must update the tuples of all employees who work in that department.

We can avoid these anomalies by decomposing the original relation into the *Employee* and *Department* relations.

Update Anomalies

- The process of normalization through decomposition must confirm the existence of the following properties :
  - The lossless join or nonadditive join property - disallows the possibility of generating spurious tuples with respect to the relation schema created after decomposition.
  - The dependency preservation property – ensures that each functional dependency is represented in some individual relation resulting after decomposition.

Generation of Spurious Tuples

- Consider the relation
  - Emp_Proj {Empid,Pnumber, Hours,Ename, Pname,Plocation}

Empid	Pnumber	Hours	Ename	Pname	Plocation
123	1	32	Perera	ProductX	Colombo
123	2	7	Perera	ProductY	Kandy
345	3	20	Silva	ProductZ	Kandy

Generation of Spurious Tuples

Consider the two relation schemas instead of Emp_Proj

- Emp_Locs{Ename, Plocation}
- Emp_Proj1{Empid,Pnumber,Hours,Pname,Plocation}

Emp_Proj1

Empid	Pnumber	Hours	Pname	Plocation
123	1	32	ProductX	Colombo
123	2	7	ProductY	Kandy
345	3	20	ProductZ	Kandy

Emp_Locs

Ename	Plocation
Perera	Colombo
Perera	Kandy
Silva	Kandy

Generation of Spurious Tuples

Empid	Pnumber	Hours	Ename	Pname	Plocation
123	1	32	Perera	ProductX	Colombo
123	2	7	Perera	ProductY	Kandy
345	**3**	**20**	**Perera**	**ProductZ**	**Kandy**
345	3	20	Silva	ProductZ	Kandy

Reason ?

Functional Dependencies

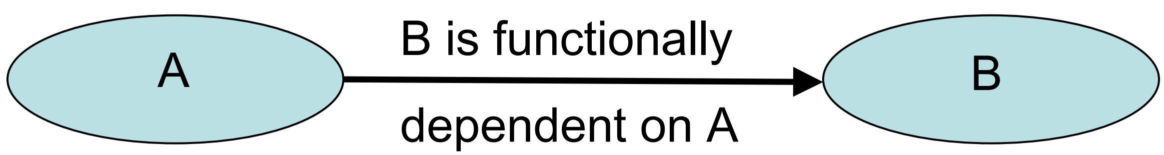
Functional dependency describes the relationship between attributes in a relation. For example,

- if A and B are attributes of relation R, B is functionally dependent on A (denoted A $\rightarrow$ B)

  if each value of A is associated with exactly one value of B.

Functional Dependencies

- When a functional dependency exists, the attribute or group of attributes on the left hand side of the arrow is called the determinant.



A is the determinant of B.

Functional Dependencies

- F – denotes the set of functional dependencies that are specified on relation schema R.
- There are functional dependencies that are semantically obvious.
- There are other dependencies that can be inferred or deduced from FDs in F.
- However, it is impossible to specify all possible functional dependencies for a given situation.

Functional Dependencies

- For example if each department has one manager, Dept_no uniquely determines Mgr_empid ;

  Dept_no → Mgr_empid

  Mgr_empid → Mgr_phone

These two dependencies together imply that

Dept_no → Mgr_phone

Functional Dependencies

- Formally, the set of all dependencies that include F as well as all dependencies that can be inferred from F called the **closure** of F; it is denoted by $\mathbf{F}^{+}$.

F = {Empid → {Ename, Bdate, Address, Dnumber}, Dnumber → { Dname, Mgrid}}

Inferred dependencies

Empid → { Dname, Mgrid}

Dnumber → Dname

Functional Dependencies

- Let A, B, and C be subsets of the attributes of relation R. Armstrong's axioms are as follows:

  1. Reflexivity

     If B is a subset of A, then A → B

  2. Augmentation

     If A → B, then A,C → B,C

  3. Transitivity

     If A → B and B → C, then A → C

Functional Dependencies

4. Projectivity

   If A $\rightarrow$ BC then A $\rightarrow$ B

5. Union

   If A $\rightarrow$ B and A $\rightarrow$ C, then A $\rightarrow$ BC