Database Normalization

Contents

1. *Purpose of Normalization*
2. *Data Redundancy and Update Anomalies*
3. *Functional Dependencies*
4. *Normalization Process*
5. *First Normal Form (1NF)*
6. *Second Normal Form (2NF)*
7. *Third Normal Form (3NF)*

1. Purpose of Normalization

- Normalization is a process for evaluating and correcting table structures to minimize data redundancies, thereby reducing the likelihood of data anomalies.
- 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:
    - Each table represents a single subject. For example, a course table will contain only data that directly pertains to courses. Similarly, a student table will contain only student data.
    - No data item will be *unnecessarily* stored in more than one table (in short, tables have minimum controlled redundancy). The reason for this requirement is to ensure that the data are updated in only one place.
    - All nonprime attributes in a table are dependent on the primary key—the entire primary key and nothing but the primary key. The reason for this requirement is to ensure that the data are uniquely identifiable by a primary key value.
    - Each table is void of insertion, update, or deletion anomalies. This is to ensure the integrity and consistency of the data.

2. Data Redundancy and Update Anomalies

- A major aim of relational database design 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:

- updates to the data stored in the database are achieved with a minimal number of operations thus reducing the opportunities for data inconsistencies occurring in the database;
- reduction in the file storage space required by the base relations thus minimizing costs.
- Relations that have redundant data may have problems called update anomalies, which are classified as insertion, deletion, or modification anomalies.
- Example of relation that has redundancy and update anomalies:

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

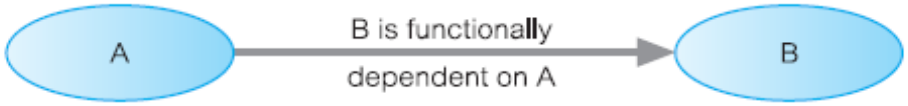
- In the StaffBranch relation there is redundant data; the details of a branch are repeated for every member of staff located at that branch. In contrast, the branch details appear only once for each branch in the Branch relation, and only the branch number (branchNo) is repeated in the Staff relation to represent where each member of staff is located.
- **Insert Anomalies :**
  - 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. For example, to insert the details of new staff located at branch number B007, we must enter the correct details of branch number B007 so that the branch details are consistent with values for branch B007 in other tuples of the StaffBranch relation.

- To insert details of a new branch that currently has no members of staff into the StaffBranch relation, it is necessary to enter nulls into the attributes for staff, such as staffNo.

- **Deletion Anomalies:**
  - If we delete a tuple from the StaffBranch relation that represents the last member of staff located at a branch, the details about that branch are also lost from the database.
- **Update Anomalies:**
  - If we want to change the value of one of the attributes of a particular branch in the StaffBranch relation, for example the address for branch number B003, we must update the tuples of all staff located at that branch. If this modification is not carried out on all the appropriate tuples of the StaffBranch relation, the database will become inconsistent.

3. Functional Dependencies

- Assume that a relational schema has attributes (A, B, C, . . . , Z) and that the database is described by a single universal relation called R = (A, B, C, . . . , Z). This assumption means that every attribute in the database has a unique name.
- 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 → B), if each value of A is associated with exactly one value of B. (A and B may each consist of one or more attributes.)



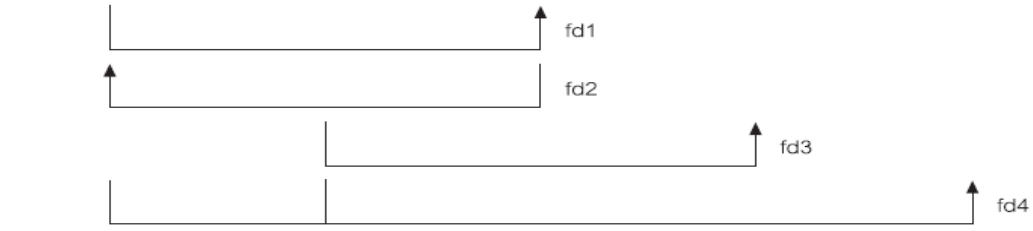
Determinant **Dependent**

- Functional dependency is a property of the meaning or semantics of the attributes in a relation. The semantics indicate how attributes relate to one another, and specify the functional dependencies between attributes.
- When a functional dependency is present, the dependency is specified as a constraint between the attributes.
- The functional dependencies that we use in normalization have the following characteristics:
  - There is a one-to-one relationship between the attribute(s) on the left-hand side (**determinant** (Refers to the attribute, or group of attributes, on the left-hand side of the arrow of a functional dependency)) and those on the right-hand side of a functional dependency. (Note that the relationship in the opposite direction, that is from the right- to the left-hand side attributes, can be a one-to-one relationship or one-to-many relationship.)
  - They hold 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. In other words, there must be a full functional dependency between the attribute(s) on the left- and right-hand sides of the dependency.
- Fully Functionally Dependent indicates that if A and B are attributes of a relation, B is fully functionally dependent on A if B is functionally dependent on A, but not on any proper subset of A.

- A functional dependency A → B is a *full* functional dependency if removal of any attribute from A results in the dependency no longer existing. A functional dependency A→B is a **partially dependency** if there is some attribute that can be removed from A and yet the dependency still holds.
- **Transitive dependency** 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:

Sample Relation

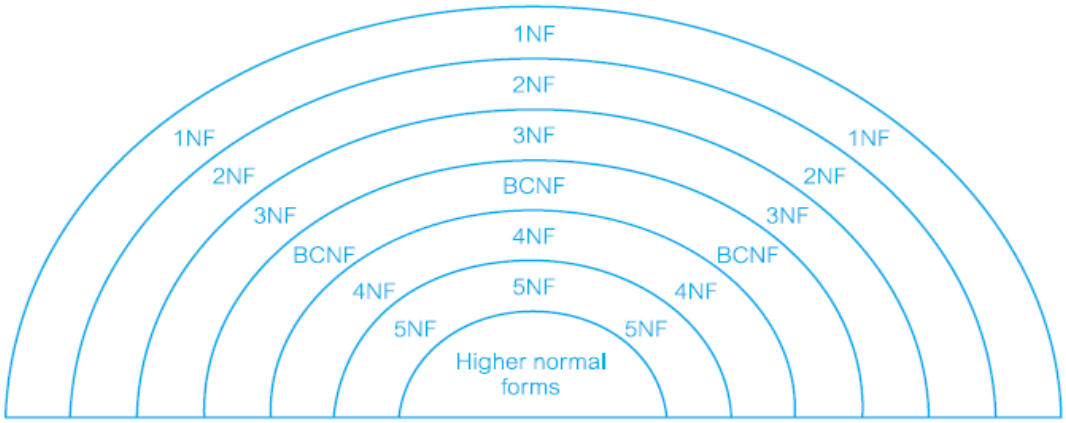
A	B	C	D	E
a	b	z	w	q
e	b	r	w	p
a	d	z	w	t
e	d	r	w	q
a	f	z	s	t
e	f	r	s	t



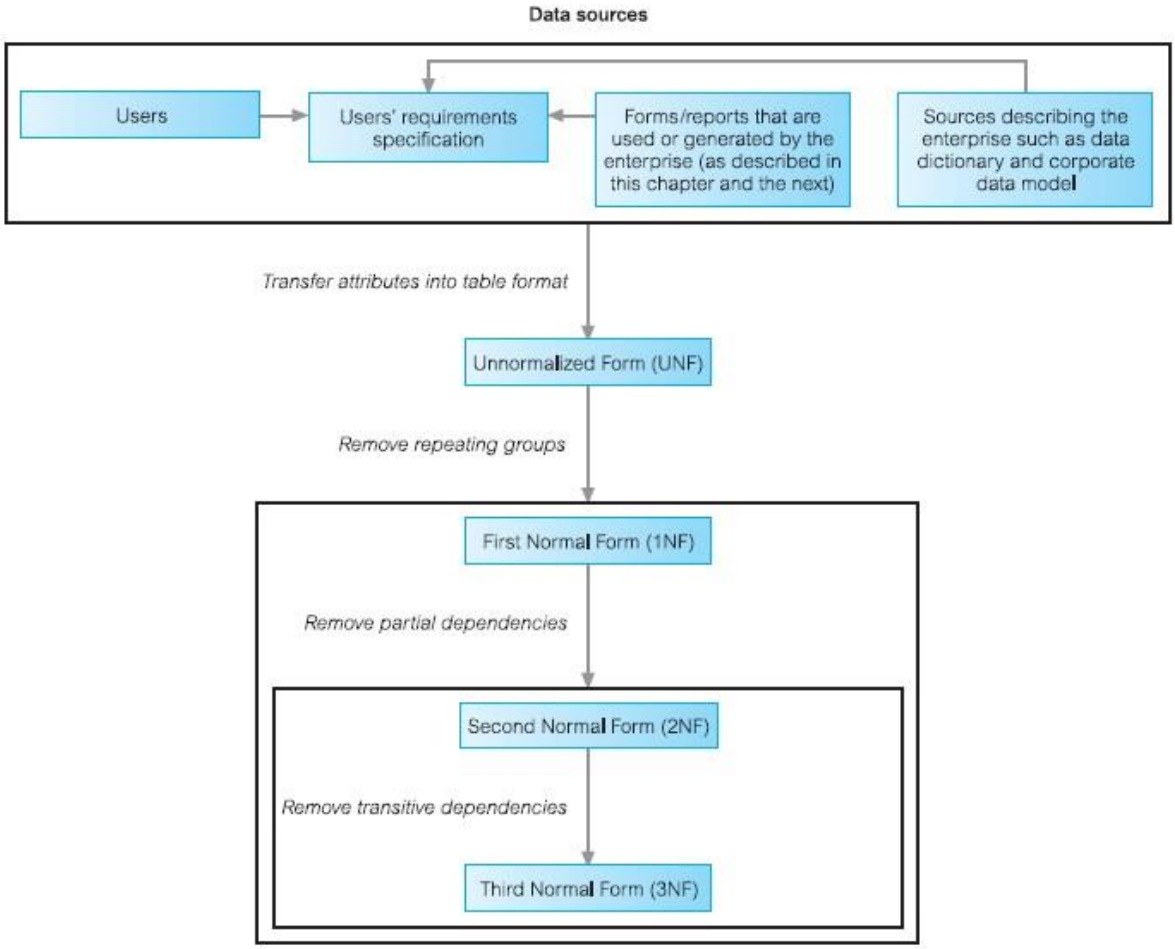
4. Normalization Process

- Normalization is a formal technique for analyzing relations based on their primary key (or candidate keys) and functional dependencies
- . The technique involves a series of rules that can be used to test individual relations so that a database can be normalized to any degree.
- When a requirement is not met, the relation violating the requirement must be decomposed into relations that individually meet the requirements of normalization.

- Three normal forms were initially proposed called First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF). Subsequently, R. Boyce and E.F. Codd introduced a stronger definition of third normal form called Boyce Codd Normal Form (BCNF).
- With the exception of 1NF, all these normal forms are based on functional dependencies among the attributes of a relation. Higher normal forms that go beyond BCNF were introduced later such as Fourth Normal Form (4NF) and Fifth Normal Form (5NF).
- Normalization is often executed as a series of steps. Each step corresponds to a specific normal form that 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 can be explained by the figure:



5. **First Normal Form (1NF)**

- **Unnormalized Form (UNF)** A table that contains one or more repeating groups.
- **First Normal Form (1NF)** A relation in which the intersection of each row and column contains one and only one value.
- The table is in Unnormalized Form referred to an unnormalized table. To transform the unnormalized table to First Normal Form we identify and remove repeating groups within the table.

- A repeating group is an attribute, or group of attributes, within a table that occurs with multiple values for a single occurrence of the nominated key attribute(s) for that table.
- To convert UNF table into 1NF table:

  (1) *By entering appropriate data in the empty columns of rows containing the repeating data.* In other words, we fill in the blanks by duplicating the nonrepeating data, where required. This approach is commonly referred to as 'flattening' the table.

  (2) *By placing the repeating data, along with a copy of the original key attribute(s), in a separate relation.*
- Example:

  UNF Table

ClientRental

clientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	John Kay	PG4	6 Lawrence St, Glasgow	1-Jul-03	31-Aug-04	350	CO40	Tina Murphy
		PG16	5 Novar Dr, Glasgow	1-Sep-04	1-Sep-05	450	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Glasgow	1-Sep-02	10-June-03	350	CO40	Tina Murphy
		PG36	2 Manor Rd, Glasgow	10-Oct-03	1-Dec-04	375	CO93	Tony Shaw
		PG16	5 Novar Dr, Glasgow	1-Nov-05	10-Aug-06	450	CO93	Tony Shaw

1NF Table:

ClientRental

clientNo	propertyNo	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	John Kay	6 Lawrence St, Glasgow	1-Jul-03	31-Aug-04	350	CO40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-04	1-Sep-05	450	CO93	Tony Shaw
CR56	PG4	Aline Stewart	6 Lawrence St, Glasgow	1-Sep-02	10-Jun-03	350	CO40	Tina Murphy
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-03	1-Dec-04	375	CO93	Tony Shaw
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Nov-05	10-Aug-06	450	CO93	Tony Shaw

7. Second Normal Form (2NF)

- Second Normal Form (2NF) is based on the concept of full functional dependency. Second Normal Form applies to relations with composite keys, that is, relations with a primary key composed of two or more attributes. A relation with a single-attribute primary key is automatically in at least 2NF.
- 2NF table is a relation that is in First Normal Form and every non-primary-key attribute is fully functionally dependent on the primary key.
- Based on the previous example:

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-03	31-Aug-04
CR76	PG16	1-Sep-04	1-Sep-05
CR56	PG4	1-Sep-02	10-Jun-03
CR56	PG36	10-Oct-03	1-Dec-04
CR56	PG16	1-Nov-05	10-Aug-06

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

7. Third Normal Form (3NF)

- A relation that is in First and Second Normal Form and in which no non-primary-key attribute is transitively dependent on the primary key.
- The normalization of 2NF relations to 3NF involves the removal of transitive dependencies. If a transitive dependency exists, we remove the transitively dependent attribute(s) from the relation by placing the attribute(s) in a new relation along with a copy of the determinant.
- Example:
-

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-03	31-Aug-04
CR76	PG16	1-Sep-04	1-Sep-05
CR56	PG4	1-Sep-02	10-Jun-03
CR56	PG36	10-Oct-03	1-Dec-04
CR56	PG16	1-Nov-05	10-Aug-06

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

- The functional dependencies for the Client, Rental, and PropertyOwner relations, derived in Example:

Client

fd2	clientNo → cName	(Primary key)

Rental

fd1	clientNo, propertyNo → rentStart, rentFinish	(Primary key)
fd5′	clientNo, rentStart → propertyNo, rentFinish	(Candidate key)
fd6′	propertyNo, rentStart → clientNo, rentFinish	(Candidate key)

PropertyOwner

fd3	propertyNo → pAddress, rent, ownerNo, oName	(Primary key)
fd4	ownerNo → oName	(Transitive dependency)

- All the non-primary-key attributes within the PropertyOwner relation are functionally dependent on the primary key, with the exception of oName, which is transitively dependent on ownerNo (represented as fd4).
- The PropertyForRent and Owner relations are in 3NF as there are no further transitive dependencies on the primary key.

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

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-03	31-Aug-04
CR76	PG16	1-Sep-04	1-Sep-05
CR56	PG4	1-Sep-02	10-Jun-03
CR56	PG36	10-Oct-03	1-Dec-04
CR56	PG16	1-Nov-05	10-Aug-06

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