Name : Kalaiselvan P

SQL Job Preparation Assignment 1

1. **Table DEPT:**

DEPTNO(PK)	DNAME
10	AC

Table EMP:

EMPNO	ENAME	DEPTNO(FK)
101	ROCK	10
102	JACK	10
103	MARK	10
104	JERRY	10

What is the difference between the parent and child tables, and why?

A parent is the table that stores the primary key, A child is any table that references the parent with a foreign key.

A foreign key is a way to enforce referential integrity within the SQL Server database. A foreign key means that values in one table must also appear in another table. The referenced table is called the parent table while the table with the foreign key is called the child table.

2. **What are the four components of a database management system?**

Below are four components in a database management system

1. Software
2. Hardware
3. Data & Procedures
4. Users

3. **What is the distinction between SQL and SQL plus?**

SQL is a query language is used to communicate with data base.

SQL Plus is a command line tool which can send sql queries to server.

4. **What is the definition of normalization?**

In database, Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

Normalization divides the larger table into smaller and links them using relationships. normal form is used to reduce redundancy from the database table.

5. **Give examples of 1NF, 2NF, 3NF, and BCNF**

Here are the most commonly used normal forms:

- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

- **First Normal Form (1NF)**

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Example: Let's say a company wants to store the names and contact details of its employees. It creates a table in the database that looks like this:

Emp_Id	Emp_Name	Emp_Address	Emp_Mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 ,
			9900012222

Employee Jon have two mobile numbers that caused the Emp_Mobile field to have multiple values for these two employees.

This table is not in 1NF as the rule says “each attribute of a table must have atomic (single) values”, the Emp_Mobile values for employees Jon & Lester violates that rule.

To make the table complies with 1NF we need to create separate rows for the each mobile number in such a way so that none of the attributes contains multiple values.

Emp_Id	Emp_Name	Emp_Address	Emp_Mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222

- **Second Normal Form (2NF)**

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

Example: Let’s say a school wants to store the data of teachers and the subjects they teach. They create a table Teacher that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

Teacher_Id	Subject	Teacher_Age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys: {Teacher_Id, Subject}
Non prime attribute: Teacher_Age

This table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute Teacher_Age is dependent on Teacher_Id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says **“no non-prime attribute is dependent on the proper subset of any candidate key of the table”**.

To make the table complies with 2NF we can disintegrate it in two tables like this:
Teacher_Details table:

Teacher_Id	Teacher_Age
111	38
222	38
333	40

Teacher_Subject table:

Teacher_Id	Subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Now the tables are in Second normal form (2NF).

- **Third Normal form (3NF)**

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

Example: Let’s say a company wants to store the complete address of each employee, they create a table named Employee_Details that looks like this:

Emp_Id	Emp_Name	Emp_Zip	Emp_State	Emp_City	Emp_District
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

Super keys: {Emp_Id}, {Emp_Id, Emp_Name}, {Emp_Id, Emp_Name, Emp_Zip}…so on
Candidate Keys: {Emp_Id}
Non-prime attributes: all attributes except Emp_Id are non-prime as they are not part of any candidate keys.

Here, Emp_State, Emp_City & Emp_District dependent on Emp_Zip. Further Emp_zip is dependent on Emp_Id that makes non-prime attributes (Emp_State, Emp_City & Emp_District) transitively dependent on super key (Emp_Id). This violates the rule of 3NF.

To make this table complies with 3NF we have to disintegrate the table into two tables to remove the transitive dependency:

Employee Table:

Emp_Id	Emp_Name	Emp_Zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

Employee_Zip table:

Emp_Zip	Emp_State	Emp_City	Emp_District
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

- **Boyce Codd normal form (BCNF)**

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Example: Suppose there is a company wherein employees work in **more than one department**. They store the data like this:

Emp_Id	Emp_Nationality	Emp_Dept	Dept_Type	Dept_No_Of_Emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above:
Emp_Id -> Emp_Nationality
Emp_Dept -> {Dept_Type, Dept_No_Of_Emp}

Candidate key: {Emp_Id, Emp_Dept}

The table is not in BCNF as neither Emp_Id nor Emp_Dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:
Emp_Nationality table:

Emp_Id	Emp_Nationality
1001	Austrian
1002	American

Emp_Dept table:

Emp_Dept	Dept_Type	Dept_No_Of_Emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

Emp_Dept_Mapping table:

Emp_Id	Emp_Dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:
Emp_Id -> Emp_Nationality
Emp_Dept -> {Dept_Type, Dept_No_Of_Emp}

Candidate keys:
For first table: Emp_Id
For second table: Emp_Dept
For third table: {Emp_Id, Emp_Dept}

This table is now in BCNF as in both the functional dependencies left side part is a key.