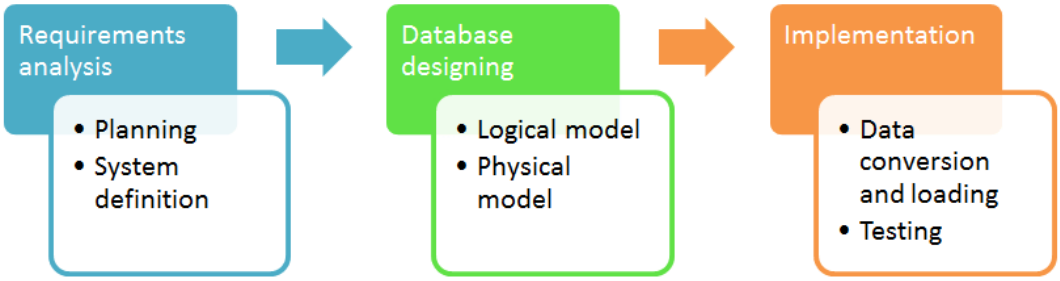
II B.Sc Computer Science
18BCS42C – DATABASE MANAGEMENT SYSTEM
UNIT-IV

DATABASE DESIGN THEORY: Data base design process – Relational Database Design – Relation Schema – Anomalies in a database – Functional dependencies – Axioms – closure of a set of FD's - minimal covers - Normal forms based on primary keys – Second Normal form, Third Normal form, Boyce–Codd Normal form.

Database Design Process

- **Database Design** is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems.
- Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space.
- The database designer decides how the data elements correlate and what data must be stored.
- The main objectives of database designing are to produce logical and physical designs models of the proposed database system.
- The logical model concentrates on the data requirements and the data to be stored independent of physical considerations.
- It does not concern itself with how the data will be stored or where it will be stored physically.
- The physical data design model involves translating the logical design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).
- Database design is important as it produces database systems
    - That meet the requirements of the users
    - Have high performance
- Database designing is crucial to **high performance** database system.

Database development life cycle



Requirements analysis

- **Planning** - This stages concerns with planning of entire Database Development Life Cycle.

    - It takes into consideration the Information Systems strategy of the organization.
- **System definition** - This stage defines the scope and boundaries of the proposed database system.

Database designing

- **Logical model** - This stage is concerned with developing a database model based on requirements. The entire design is on paper without any physical implementations or specific DBMS considerations.
- **Physical model** - This stage implements the logical model of the database taking into account the DBMS and physical implementation factors.

Implementation

- **Data conversion and loading** - this stage is concerned with importing and converting data from the old system into the new database.
- **Testing** - this stage is concerned with the identification of errors in the newly implemented system .It checks the database against requirement specifications.

Two Types of Database Techniques

1. Normalization
2. ER Modeling

Relational Database Design (RDD)

- Relational database design (RDD) models information and data into a set of tables with rows and columns.
- Each row of a relation/table represents a record, and each column represents an attribute of data.
- The Structured Query Language (SQL) is used to manipulate relational databases.
- The design of a relational database is composed of four stages, where the data are modeled into a set of related tables.
- The stages are:
    - Define relations/attributes
    - Define primary keys
    - Define relationships
    - Normalization
- Relational databases differ from other databases in their approach to organizing data and performing transactions.
- In an RDD, the data are organized into tables and all types of data access are carried out via controlled transactions.
- Relational database design satisfies the ACID (atomicity, consistency, integrity and durability) properties required from a database design.

- Relational database design mandates the use of a database server in applications for dealing with data management problems.
- The four stages of an RDD are as follows:
    - Relations and attributes: The various tables and attributes related to each table are identified. The tables represent entities, and the attributes represent the properties of the respective entities.
    - Primary keys: The attribute or set of attributes that help in uniquely identifying a record is identified and assigned as the primary key
    - Relationships: The relationships between the various tables are established with the help of foreign keys. Foreign keys are attributes occurring in a table that are primary keys of another table. The types of relationships that can exist between the relations (tables) are:
        - One to one
        - One to many
        - Many to many
- An entity-relationship diagram can be used to depict the entities, their attributes and the relationship between the entities in a diagrammatic way.

Normalization

- This is the process of optimizing the database structure.
- Normalization simplifies the database design to avoid redundancy and confusion.
- The different normal forms are as follows:
    - First normal form
    - Second normal form
    - Third normal form
    - Boyce-Codd normal form
    - Fifth normal form
- By applying a set of rules, a table is normalized into the above normal forms in a linearly progressive fashion.
- The efficiency of the design gets better with each higher degree of normalization.

Relation Schema

- Relation schema defines the design and structure of the relation like it consists of the relation name, set of attributes/field names/column names.
- Every attribute would have an associated domain.
- Example: There is a student named Geeks, she is pursuing B.Tech, in the 4th year, and belongs to IT department (department no. 1) and has roll number 1601347.She is proctored by Mrs. S Mohanty. If we want to represent this using databases we would have to create a student table

with name, sex, degree, year, department, department number, roll number and proctor (adviser) as the attributes.

student (rollNo, name, degree, year, sex, deptNo, advisor)

Note –

If we create a database, details of other students can also be recorded.
Similarly, we have the IT Department, with department Id 1, having Mrs. Sujata Chakravarty as the head of department. And we can call the department on the number 0657 228662 .
This and other departments can be represented by the department table, having department ID, name, hod and phone as attributes.

department(deptId, name, hod, phone)

The course that a student has selected has a courseid, course name, credit and department number.

course(courseId, ename, credits, deptNo)

The professor would have an employee Id, name, sex, department no. and phone number.

professor(empId, name, sex, startYear, deptNo, phone)

We can have another table named enrollment, which has roll no, courseId, semester, year and grade as the attributes.

enrollment(rollNo, coursId, sem, year, grade)

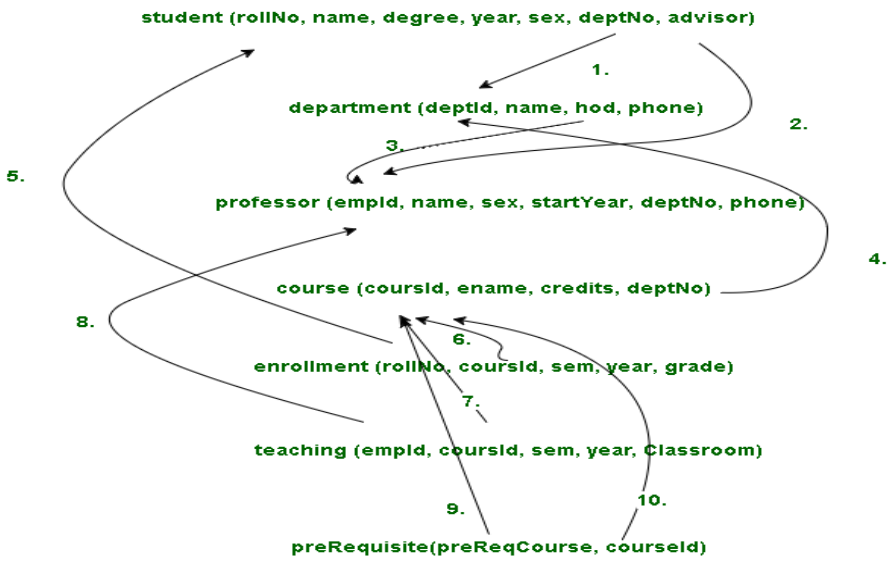
Teaching can be another table, having employee id, course id, semester, year and classroom as attributes.

teaching(empId, coursed, sem, year, Classroom)

When we start courses, there are some courses which another course that needs to be completed before starting the current course, so this can be represented by the Prerequisite table having prerequisite course and course id attributes.

prerequisite(preReqCourse, courseId)

The relations between them is represented through arrows in the following **Relation diagram**,



1. This represents that the deptNo in student table table is same as deptId used in department table. deptNo in student table is a foreign key. It refers to deptId in department table.
2. This represents that the advisor in student table is a foreign key. It refers to empId in professor table.
3. This represents that the hod in department table is a foreign key. It refers to empId in professor table.
4. This represents that the deptNo in course table table is same as deptId used in department table. deptNo in student table is a foreign key. It refers to deptId in department table.
5. This represents that the rollNo in enrollment table is same as rollNo used in student table.
6. This represents that the courseId in enrollment table is same as courseId used in course table.
7. This represents that the courseId in teaching table is same as courseId used in course table.
8. This represents that the empId in teaching table is same as empId used in professor table.
9. This represents that preReqCourse in preReqiusite table is a foreign key. It refers to courseId in course table.
10. This represents that the deptNo in student table is same as deptId used in department table.

Anomalies in Relational Model

- **Anomalies**

  There are different types of anomalies which can occur in referencing and referenced relation which can be discussed as:

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

- **Insertion anomaly:** If a tuple is inserted in referencing relation and referencing attribute value is not present in referenced attribute, it will not

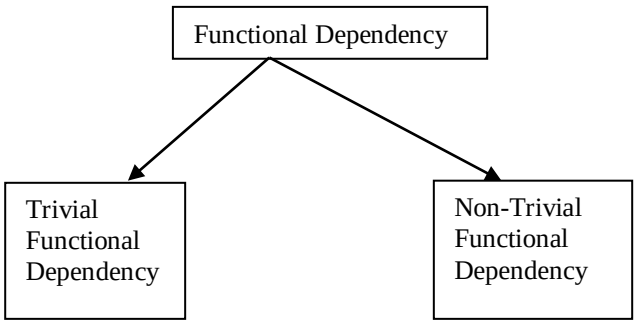
allow inserting in referencing relation. For Example, If we try to insert a record in STUDENT_COURSE with STUD_NO =7, it will not allow.

- **Deletion and Updation anomaly:** If a tuple is deleted or updated from referenced relation and referenced attribute value is used by referencing attribute in referencing relation, it will not allow deleting the tuple from referenced relation. For Example, If we try to delete a record from STUDENT with STUD_NO =1, it will not allow. To avoid this, following can be used in query:
    - **ON DELETE/UPDATE SET NULL:** If a tuple is deleted or updated from referenced relation and referenced attribute value is used by referencing attribute in referencing relation, it will delete/update the tuple from referenced relation and set the value of referencing attribute to NULL.
    - **ON DELETE/UPDATE CASCADE:** If a tuple is deleted or updated from referenced relation and referenced attribute value is used by referencing attribute in referencing relation, it will delete/update the tuple from referenced relation and referencing relation as well.

FUNCTIONAL DEPENDENCY

- The functional dependency is a relationship that exists between two attributes.
- It typically exists between the primary key and non-key attribute within a table.
- X → Y The left side of FD is known as a determinant, the right side of the production is known as a dependent.
- For example: Assume we have an employee table with attributes:
  Emp_Id, Emp_Name, Emp_Address.
  Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.
- Functional dependency can be written as:
  Emp_Id → Emp_Name

Types of Functional dependency



Trivial functional dependency

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B
- Example: Consider a table with two columns Employee_Id and Employee_Name.

  {Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as Employee_Id is a subset of {Employee_Id, Employee_Name}.
- Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are trivial dependencies too.

Non-trivial functional dependency

- A → B has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then A → B is called as complete non-trivial.
- Example:

  ID → Name,
  Name → DOB

Normalization of Database

- Database Normalization is a technique of organizing the data in the database.
- Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion anomalies.
- It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.
- Normalization is used for mainly two purposes, •
  - Eliminating reduntant(useless) data.
  - Ensuring data dependencies make sense i.e data is logically stored.

Problems Without Normalization

- If a table is not properly normalized and have data redundancy then it will not only eat up extra memory space but will also make it difficult to handle and update the database, without facing data loss.
- Insertion, Updation and Deletion Anomalies are very frequent if database is not normalized.
- To understand these anomalies let us take an example of a Student table.

ROLLNO	NAME	BRANCH	HOD	OFFICE_TEL
401	Akon	CSE	Mr.X	53337
402	Bkon	CSE	Mr.X	53337
403	Ckon	CSE	Mr.X	53337
404	Dkon	CSE	Mr.X	53337

In the table above, we have data of 4 Computer Sci. students. As we can see, data for the fields branch, hod(Head of Department) and office_tel is repeated for the students who are in the same branch in the college, this is Data Redundancy.

Insertion Anomaly

- Suppose for a new admission, until and unless a student opts for a branch, data of the student cannot be inserted, or else we will have to set the branch information as NULL.
- Also, if we have to insert data of 100 students of same branch, then the branch information will be repeated for all those 100 students.
- These scenarios are nothing but Insertion anomalies.

Updation Anomaly

- What if Mr. X leaves the college? or is no longer the HOD of computer science department?
- In that case all the student records will have to be updated, and if by mistake we miss any record, it will lead to data inconsistency.
- This is Updation anomaly.

Deletion Anomaly

- In our Student table, two different informations are kept together, Student information and Branch information.
- Hence, at the end of the academic year, if student records are deleted, we will also lose the branch information.
- This is Deletion anomaly.

Normalization Rule

- Normalization rules are divided into the following normal forms:
    1. First Normal Form
    2. Second Normal Form
    3. Third Normal Form
    4. BCNF
    5. Fourth Normal Form
    6. Fifth Normal Form

First Normal Form (1NF)

- For a table to be in the First Normal Form, it should follow the following 4 rules:
    1. It should only have single(atomic) valued attributes/columns.
    2. Values stored in a column should be of the same domain
    3. All the columns in a table should have unique names.
    4. And the order in which data is stored, does not matter.

Rules for First Normal Form

- The first normal form expects you to follow a few simple rules while designing your database, and they are:

- Rule 1: Single Valued Attributes Each column of your table should be single valued which means they should not contain multiple values. We will explain this with help of an example later, let's see the other rules for now.
- Rule 2: Attribute Domain should not change This is more of a "Common Sense" rule. In each column the values stored must be of the same kind or type. For example: If you have a column dob to save date of births of a set of people, then you cannot or you must not save 'names' of some of them in that column along with 'date of birth' of others in that column. It should hold only 'date of birth' for all the records/rows.
- Rule 3: Unique name for Attributes/Columns This rule expects that each column in a table should have a unique name. This is to avoid confusion at the time of retrieving data or performing any other operation on the stored data. If one or more columns have same name, then the DBMS system will be left confused.
- Rule 4: Order doesn't matters This rule says that the order in which you store the data in your table doesn't matter.
- EXAMPLE: Create a table to store student data which will have student's roll no., their name and the name of subjects they have opted for.

ROLL_NO	NAME	SUBJECT
101	Akon	OS, CN
103	Ckon	Java
102	Bkon	C, C++

- Here is our updated table and it now satisfies the First Normal Form.

ROLL_NO	NAME	SUBJECT
101	Akon	OS
101	Akon	CN
103	Ckon	Java
102	Bkon	C
102	Bkon	C++

Second Normal Form (2NF)

- For a table to be in the Second Normal Form
    1. It should be in the First Normal form.
    2. And, it should not have Partial Dependency.

Dependency

- Let's take an example of a Student table with columns student_id, name, reg_no(registration number), branch and address(student's home address).

Student_id	name	Reg_no	branch	Address

- In this table, student_id is the primary key and will be unique for every row, hence we can use student_id to fetch any row of data from this table
- Even for a case, where student names are same, if we know the student_id we can easily fetch the correct record.

Student_id	name	Reg_no	branch	Address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat

- Hence we can say a Primary Key for a table is the column or a group of columns(composite key) which can uniquely identify each record in the table.

Partial Dependency

- Now that we know what dependency is, we are in a better state to understand what partial dependency is.
- For a simple table like Student, a single column like student_id can uniquely identfy all the records in a table.
- But this is not true all the time. So now let's extend our example to see if more than 1 column together can act as a primary key.
- Let's create another table for Subject, which will have subject_id and subject_name fields and subject_id will be the primary key.

Subject_id	Subject_name
1	Java
2	C++
3	Php

- Now we have a Student table with student information and another table Subject for storing subject information.
- Let's create another table Score, to store the marks obtained by students in the respective subjects.
- We will also be saving name of the teacher who teaches that subject along with marks.

Score_id	Student_id	Subject_id	Marks	Teacher
1	10	1	70	Java Teacher
2	10	2	75	C++ Teacher
3	11	1	80	Java Teacher

- In the score table we are saving the student_id to know which student's marks are these and subject_id to know for which subject the marks are for.

- Together, student_id + subject_id forms a Candidate Key which can be the Primary key.
- To get me marks of student with student_id 10, can you get it from this table? No, because you don't know for which subject. And if I give you subject_id, you would not know for which student. Hence we need student_id + subject_id to uniquely identify any row.
- Now if you look at the Score table, we have a column names teacher which is only dependent on the subject, for Java it's Java Teacher and for C++ it's C++ Teacher & so on.
- Now as we just discussed that the primary key for this table is a composition of two columns which is student_id & subject_id but the teacher's name only depends on subject, hence the subject_id, and has nothing to do with student_id.
- This is Partial Dependency, where an attribute in a table depends on only a part of the primary key and not on the whole key.
- There can be many different solutions for this, but our objective is to remove teacher's name from Score table.
- The simplest solution is to remove columns teacher from Score table and add it to the Subject table.
- Hence, the Subject table will become:

Subject_id	Subject_name	Teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

- And our Score table is now in the second normal form, with no partial dependency.

Score_id	Student_id	Subject_id	Marks
1	10	1	70
2	10	2	75
3	11		

Third Normal Form (3NF)

- A table is said to be in the Third Normal Form when,
    1. It is in the Second Normal form.
    2. And, it doesn't have Transitive Dependency.

Student Table

Student_id	name	Reg_no	branch	Address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat
12	Bkon	09-WY	IT	Rajasthan

Subject Table

Subject_id	Subject_name	Teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

Score Table

In the Score table, we need to store some more information, which is the exam name and total marks, so let's add 2 more columns to the Score table.

Score_id	Student_id	Subject_id	Marks
1	10	1	70
2	10	2	75
3	11	1	80

Transitive Dependency

- With exam_name and total_marks added to our Score table, it saves more data now.
- Primary key for the Score table is a composite key, which means it's made up of two attributes or columns → student_id + subject_id.
- The new column exam_name depends on both student and subject.
- For example, a mechanical engineering student will have Workshop exam but a computer science student won't.
- And for some subjects you have Practical exams and for some you don't.
- So we can say that exam_name is dependent on both student_id and subject_id.
- The column total_marks depends on exam_name as with exam type the total score changes.
- For example, practicals are of less marks while theory exams are of more marks. But, exam_name is just another column in the score table.
- It is not a primary key or even a part of the primary key, and total_marks depends on it. This is Transitive Dependency.

- When a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key.

  Score Table: In 3rd Normal Form

Score_id	Student_id	Subject_id	Marks	Exam_id

  The new Exam table

Exam_id	Exam_name	Total_marks
1	Workshop	200
2	Mains	70
3	Practicals	30

Advantage of removing Transitive Dependency

- The advantage of removing transitive dependency is,
  - Amount of data duplication is reduced.
  - Data integrity achieved.

Boyce and Codd Normal Form (BCNF)

- Boyce and Codd Normal Form is a higher version of the Third Normal form.
- This form deals with certain type of anomaly that is not handled by 3NF.
- A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.
- For a table to be in BCNF, following conditions must be satisfied:
  - R must be in 3rd Normal Form
  - For each functional dependency ($X \rightarrow Y$), X should be a super Key.
  - In simple words, it means, that for a dependency $A \rightarrow B$, A cannot be a non-prime attribute, if B is a prime attribute.
- Example

  College enrolment table with columns student_id, subject and professor.

Student_id	Subject	Professor
101	Java	P.Java
101	C++	P.C++
102	Java	P.Java2

103	C#	P.Chash
104	Java	P.Java

In the table above: One student can enroll for multiple subjects. For example, student with student_id 101, has opted for subjects - Java & C++

- Hence, there is a dependency between subject and professor here, where subject depends on the professor name.
- This table satisfies the 1st Normal form because all the values are atomic, column names are unique and all the values stored in a particular column are of same domain.
- This table also satisfies the 2nd Normal Form as there is no Partial Dependency.
- And, there is no Transitive Dependency, hence the table also satisfies the 3rd Normal Form.
- But this table is not in Boyce-Codd Normal Form.
- In the table above, student_id, subject form primary key, which means subject column is a prime attribute.
- But, there is one more dependency, professor → subject.
- And while subject is a prime attribute, professor is a non-prime attribute, which is not allowed by BCNF.
- To make this relation(table) satisfy BCNF, we will decompose this table into two tables, student table and professor table.

Student Table

Student_id	**P_id**
101	1
101	2

Professor Table

P_id	**Professor**	**Subject**
1	P.Java	Java
2	P.Cpp	C++

- And now, this relation satisfy Boyce-Codd Normal Form.

Fourth Normal Form (4NF)

- A table is said to be in the Fourth Normal Form when,
    1. It is in the Boyce-Codd Normal Form.
    2. And, it doesn't have Multi-Valued Dependency.

Multi-valued Dependency

- A table is said to have multi-valued dependency, if the following conditions are true,

  1. For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.

  2. Also, a table should have at-least 3 columns for it to have a multi-valued dependency.

  3. And, for a relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

- Example

  Below we have a college enrolment table with columns s_id, course and hobby.

S_id	Course	Hobby
1	Science	Cricket
1	Maths	Hockey
2	C#	Cricket
2	Php	Hockey

- From the table above, student with s_id 1 has opted for two courses, Science and Maths, and has two hobbies, Cricket and Hockey.
- Well the two records for student with s_id 1, will give rise to two more records, as shown below, because for one student, two hobbies exists, hence along with both the courses, these hobbies should be specified.

S_id	Course	Hobby
1	Science	Cricket
1	Maths	Hockey
2	Science	Cricket
2	Maths	Hockey

- And, in the table above, there is no relationship between the columns course and hobby.
- They are independent of each other.
- So there is multi-value dependency, which leads to un-necessary repetition of data and other anomalies as well.
- To make the above relation satify the 4th normal form, we can decompose the table into 2 tables.

CourseOpted Table

S_id	Course
1	Science
1	Maths
2	C#
2	Php

Hobbies Table

S_id	Hobby
1	Cricket
1	Hockey
2	Cricket
2	Hockey

Fifth Normal Form (5NF)

- A database is said to be in 5NF, if and only if,
    1. It's in 4NF
    2. If we can decompose table further to eliminate redundancy and anomaly, and when we re-join the decomposed tables by means of candidate keys, we should not be losing the original data or any new record set should not arise. In simple words, joining two or more decomposed table should not lose records nor create new records.
- If a table can be recreated by joining multiple tables and each of this table have a subset of the attributes of the table, then the table is in Join Dependency.
- It is a generalization of Multivalued DependencyJoin Dependency can be related to 5NF, wherein a relation is in 5NF, only if it is already in 4NF and it cannot be decomposed further.
- Example
  <Employee>

EmpName	EmpSkills	EmpJob(Assigned Work)
Tom	Networking	EJ001
Harry	Web Development	EJ002
Katie	Programming	EJ002

<EmployeeSkills>

EmpName	EmpSkills
Tom	Networking
Harry	Web Development
Katie	Programming

<EmployeeJob>

EmpName	EmpJob(Assigned Work)
Tom	EJ001
Harry	EJ002
Katie	EJ002

<JobSkills>

EmpSkills	EmpJob(Assigned Work)
Networking	EJ001
Web Development	EJ002
Programming	EJ002

Our Join Dependency:
{(EmpName, EmpSkills), (EmpName, EmpJob), (EmpSkills, EmpJob)}
The above relations have join dependency, so they are not in 5NF. That would mean that a join relation of the above three relations is equal to our original relation .

REFERENCES

1. Silberschatz A., Korth H. and Sudarshan S., "Database System Concepts", Tata McGraw Hill, 2011.
2. Elmasri R. and Navathe S.B., "Fundamentals of Database Systems", Pearson Education, 2016.
3. www.w3schools.com
4. www.guru99.com
5. www.tutorialspoint.com
6. www.beginnersbook.com
7. www.ecomputernotes.com

----------**********----------