**Database Management System**

**(CSL 214)**

**Lab Workbook**

Logo

Description automatically generated

Faculty Name: Ms. Anuradha Dhull Student Name: Piyush Gambhir

Roll No.: 21CSU349

Semester: IV

Group: AL-3

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122017, India

Session 2022-23

**Student Name: Piyush Gambhir**

**Student Roll No.: 21CSU349**

**INDEX**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **Experiment Title** | **Date of Experiment** | **Date of Submission** | **Sign Student** | **CO Covered** | **Sign Faculty** |
| **1** | Design an ER/EER diagram for the COMPANY and SPORTS TEAM database. |  |  |  | **CO2** |  |
| **2** | Design a Relational Database Schema for the COMPANY and SPORTS TEAM database from the ER/EER diagram. |  |  |  | **CO2** |  |
| **3** | To apply SQL integrity constraints as per the DDL statements given below for COMPANY database. |  |  |  | **CO4** |  |
| **4** | To familiarize with SELECT-FROM-WHERE SQL simple queries on the COMPANY database. |  |  |  | **CO4** |  |
| **5** | To familiarize with JOIN operations in SQL on the COMPANY database. |  |  |  | **CO4** |  |
| **6** | To understand Aggregate functions using SQL queries on the COMPANY database. |  |  |  | **CO4** |  |
| **7** | To familiarize with nested SQL queries on the COMPANY database. |  |  |  | **CO4** |  |
| **8** | Identifying contrast between Relational Databases and NoSQL, thereby recognizing their applications. |  |  |  | **CO5** |  |
| **9** | Create COMPANY database using NoSQL database - MongoDB. |  |  |  | **CO5** |  |
| **10** | Retrieve data from NoSQL database - MongoDB. |  |  |  | **CO5** |  |
| **11** | VA - Design an ER/ EER Diagram, relational schema and implement the database in MongoDB. |  |  |  | **CO2, CO5** |  |

**Experiment No: 1**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section:** Semester-IV – AIML-B (AL-3) |
| **Link to Code:** [Piyush-Gambhir/NCU-CSL214-DBMS-Lab\_Manual (github.com)](https://github.com/Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual) |
| **Date:** 30.01.2023 |
| **Faculty Signature:** |
| **Marks:** |

**Objective**

Design an ER/EER diagram for the COMPANY and SPORTS TEAM database.

**Program Outcome**

* The students will be able to draw conceptual database design using ERD Plus.

**Problem Statement**

1. The COMPANY database keeps track of a company’s employees, departments, and projects. Suppose that after the requirements collection and analysis phase, the database designers provide the following description of the mini world—the part of the company that will be represented in the database.

* A department controls several projects, each of which has a unique  
  name, a unique number, and a single location.
* The company is organized into departments. Each department has a unique  
  name, a unique number, and a particular employee who manages the  
  department. We keep track of the start date when that employee began managing the department. A department may have several locations.
* We store each employee’s name, Social Security number, address, salary, sex  
  (gender), and birth date. An employee is assigned to one department, but  
  may work on several projects, which are not necessarily controlled by the  
  same department. We keep track of the current number of hours per week  
  that an employee works on each project. We also keep track of the direct  
  supervisor of each employee (who is another employee).
* We want to keep track of the dependents of each employee for insurance  
  purposes. We keep each dependent’s first name, sex, birth date, and relationship to the employee. `

2. Design an ER/EER diagram for keeping track of the exploits of your favourite sports team. You should store the matches played, the scores in each match, the players in each match and individual player statistics for each match. Summary statistics should be modelled as derived attributes. Further, extend the E-R diagram of the previous question to track the same information for all teams in a league.

Design the ER/EER model by identifying the following from the above requirements:

1. Entities (Strong and Weak)
2. Relationships
3. Participation constraints
4. Various types of attributes
5. Recursive relations
6. Mapping cardinalities
7. Binary/Ternary relationship
8. Specialization/Generalization etc.

**Background Study**

ER Diagram Symbols and Notations:

1. Entity:
   * Real-world object distinguishable from other objects.

* An entity is described using a set of *attributes*.

1. Entity Set: A collection of similar entities. Eg: all employees.

* All entities in an entity set have the same set of attributes.
* Each entity set has a *key*.
* Each attribute has a *domain*.

1. Attributes

* Attributes are properties used to describe an entity.
* Example: EMPLOYEE entity may have a Name, SSN, Address, Sex, BirthDate.

Diagram

Description automatically generated

1. Relationship

* A relationship relates two or more distinct entities with a specific meaning.
* Relationships of the same type are grouped or typed into a relationship type.
* 3 types of relationships : Unary, Binary & Ternary.

1. Recursive Relationship

* A relationship with the same participating entity type in distinct roles.
* Example: the SUPERVISION relationship

1. Structural Constraints – Semantics of Relationships

* Cardinality Ratio : The number of instances of an entity from a relation that can be associated with the relation.
* Participation Constraints
* Total Participation − Each entity is involved in the relationship.
* Partial participation − Not all entities are involved in the relationship.

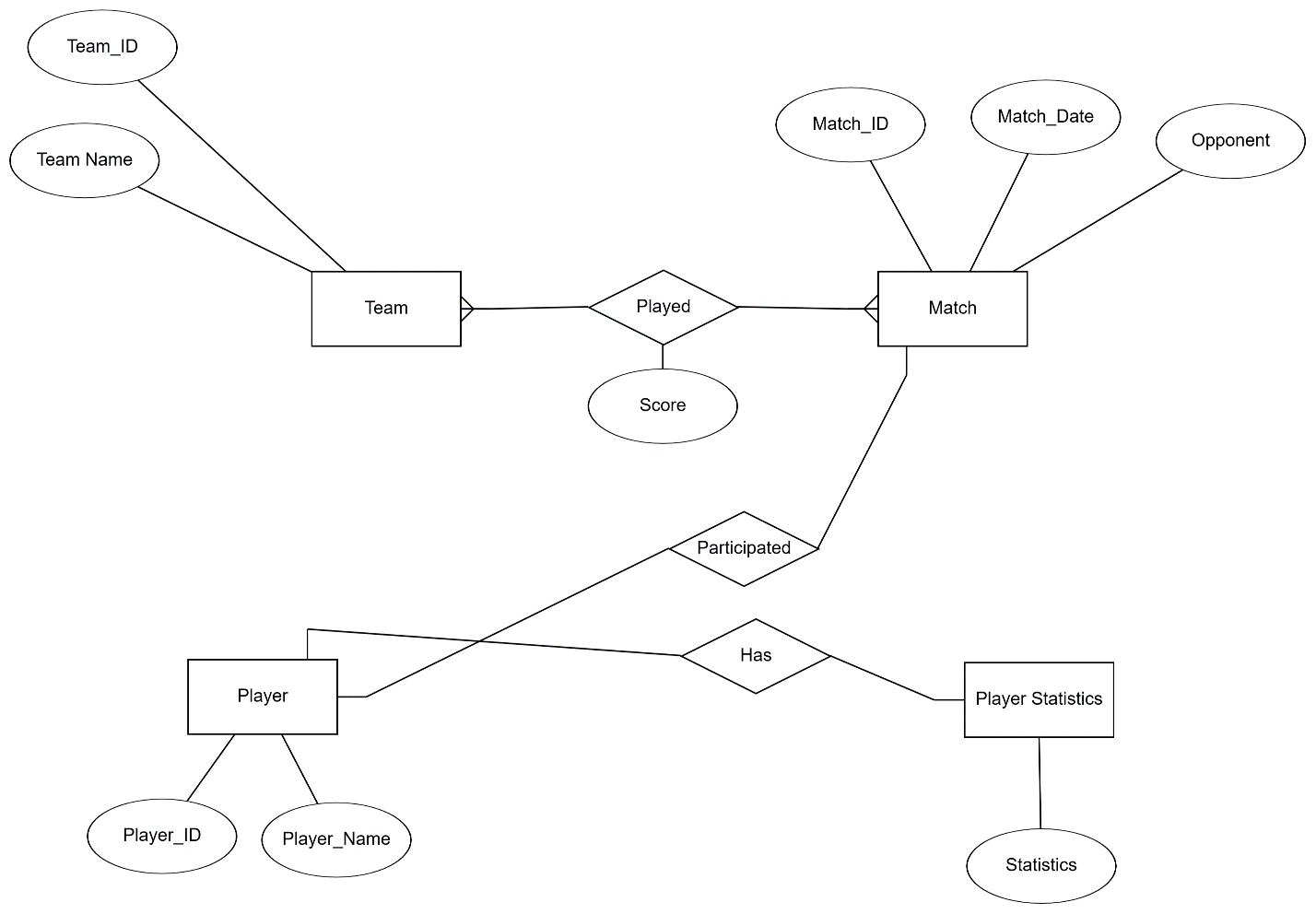
**ER Diagram**

**Company Database**

Diagram, schematic

Description automatically generated

**Sports Team Database**

****

**Preparatory Questions**

Q1) Given the basic ER and relational models, which of the following is INCORRECT?

1. An attribute of an entity can have more than one value.
2. An attribute of an entity can be composite.
3. In a row of a relational table, an attribute can have more than one value.
4. In a row of a relational table, an attribute can have exactly one value or a NULL value.

Q2) Consider a directed line(-->) from the relationship set advisor to both entity sets instructor and student. This indicates \_\_\_\_\_\_\_\_\_ cardinality  
 1) One to many  
 2) One to one  
 3) Many to many  
 4) Many to one

Q3) An entity set that does not have sufficient attributes to form a primary key is termed as:  
 1) Strong entity set  
 2) Variant set  
 3) Weak entity set  
 4) Variable set

Q4) Which of the following indicates the maximum number of entities can be involved in a relationship?

1) Minimum cardinality

2) Maximum Cardinality

3) ERD

4) Greater Entity Count (GEC)

Q5) State true or false: Every weak entity must be associated with and identifying entity.

1. True
2. False

**Experiment No: 2**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section:** Semester-IV – AIML-B (AL-3) |
| **Link to Code:** [Piyush-Gambhir/NCU-CSL214-DBMS-Lab\_Manual (github.com)](https://github.com/Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual) |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

**Objective**

Design a Relational Database Schema for the COMPANY and SPORTS TEAM database from the ER/EER diagram.

**Program Outcome**

* The students will be able to map the conceptual database design to logical (relational) database design.

**Problem Statement**

Map the ER/EER diagram of the COMPANY and SPORTS TEAM database created in the previous experiment to Relational Database Schema. Follow the below mentioned steps to successfully map ER/EER model to relational tables:

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types

Step 5: Mapping of Binary M:N Relationship Types

Step 6: Mapping of Multivalued attributes

Step 7: Mapping of N-ary Relationship Types

Step 8: Mapping EER Model Constructs to Relations

Step 9: Options for Mapping Specialization or Generalization

Step 10: Mapping of Union Types (Categories)

**Background Study**

1. Mapping of Regular Entity Types

* For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Choose one of the key attributes of E as the primary key for R. If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

1. Mapping of Weak Entity Types

* For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R.
* Include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).

1. Mapping of Binary 1:1 Relationship Types

* For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. This has 3 approaches :
* Foreign Key Approach
* Merged Relation Option
* Cross-reference or Relationship Relation Option

1. Mapping of Binary 1:N Relationship Types

* For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
* Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.

1. Mapping of Binary M:N Relationship Types

* For each regular binary M:N relationship type R, create a new relation S to represent R.
* Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.

1. Mapping of Multi-valued attributes

* For each multi-valued attribute, A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.

1. Mapping of N-ary Relationship Types

* For each n-ary relationship type R, where n>2, create a new relationship S to represent R.
* Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.

1. Options for Mapping Specialization or Generalization

* Convert each specialization with m subclasses {S1, S2,….,Sm} and generalized superclass C, where the attributes of C are {k,a1,…an} and k is the (primary) key, into relational schemas using 1 of the following :

1. Multiple relations-Superclass and subclasses.
2. Multiple relations-Subclass relations only :
3. Single relation with *one type attribute*
4. Single relation with multiple type attributes.
5. Mapping of Union Types (Categories)

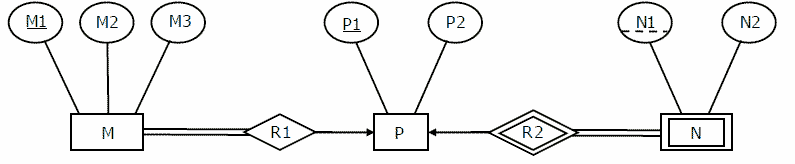
* For mapping a category whose defining superclass have different keys, it is customary to specify a new key attribute, called a surrogate key, when creating a relation to correspond to the category.

**Relational Database Design**

**Preparatory Questions**

Q1)  In which of the following, a separate schema is created consisting of that attribute and the primary key of the entity set.  
 1) A many-to-many relationship set  
 2) A multivalued attribute of an entity set  
 3) A one-to-many relationship set  
 4) All of the mentioned

Q2) Consider the following ER diagram.



The minimum number of tables needed to represent M, N, P, R1, R2 is

1. 2
2. 3
3. 4
4. 5

Q3) Consider the data given in above question. Which of the following is a correct attribute set for one of the tables for the correct answer to the above question?

1. {M1, M2, M3, P1}
2. {M1, P1, N1, N2}
3. {M1, P1, N1}
4. {M1, P1}

Q4) Let E1 and E2 be two entities in an E/R diagram with simple single-valued attributes. R1 and R2 are two relationships between E1 and E2, where R1 is one-to-many and R2 is many-to-many. R1 and R2 do not have any attributes of their own. What is the minimum number of tables required to represent this situation in the relational model?

1. 2
2. 3
3. 4
4. 5

Q5) What is the min and max number of tables required to convert an ER diagram with 2 entities and 1 relationship between them with partial participation constraints of both entities?

1. Min 1 and max 2
2. Min 1 and max 3
3. Min 2 and max 3
4. Min 2 and max 2

**Experiment No: 3**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section:** Semester-IV – AIML-B (AL-3) |
| **Link to Code:** [Piyush-Gambhir/NCU-CSL214-DBMS-Lab\_Manual (github.com)](https://github.com/Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual) |
| **Date:** 23.01.2023 |
| **Faculty Signature:** |
| **Marks:** |

**Objective**

To apply SQL integrity constraints as per the DDL statements given below for COMPANY database.

**Program Outcome**

* The students will understand how a database is created followed by insertion of relevant data.
* The students will understand the need of applying various types of integrity constraints such as primary key, foreign key, unique key, NOT NULL, default and CHECK etc

**Problem Statement**

Table

Description automatically generated

Table

Description automatically generated

A picture containing table

Description automatically generated

Implement the following types of integrity constraints:

1. Primary Key
2. Foreign Key
3. Unique
4. Default
5. Auto-increment
6. Check
7. Not Null

**Background Study**

1. Primary Key Constraint: A column or group of columns in a table which helps us to uniquely identifies every row in that table is called a primary key. This DBMS can't be a duplicate. The same value can't appear more than once in the table.

**Syntax to define a Primary key at column level:**

*column name datatype [CONSTRAINT constraint\_name] PRIMARY KEY*

**Syntax to define a Primary key at table level:**

*[CONSTRAINT constraint\_name] PRIMARY KEY (column\_name1,column\_name2,..)*

Rules for defining Primary key:

* Two rows can't have the same primary key value
* It must for every row to have a primary key value.
* The primary key field cannot be null.
* The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

1. Foreign Key (Referential integrity constraint): This constraint identifies any column referencing the PRIMARY KEY in another table. It establishes a relationship between two columns in the same table or between different tables. For a column to be defined as a Foreign Key, it should be a defined as a Primary Key in the table which it is referring. One or more columns can be defined as Foreign key.

**Syntax to define a Foreign key at column level:**

*[CONSTRAINT constraint\_name] REFERENCES Referenced\_Table\_name(column\_name)*

**Syntax to define a Foreign key at table level:**

*[CONSTRAINT constraint\_name] FOREIGN KEY(column\_name) REFERENCES referenced\_table\_name(column\_name);*

1. SQL Not Null Constraint : This constraint ensures all rows in the table contain a definite value for the column which is specified as not null. Which means a null value is not allowed.

Syntax to define a Not Null constraint:

*[CONSTRAINT constraint name] NOT NULL*

## SQL Unique Key: This constraint ensures that a column or a group of columns in each row have a distinct value. A column(s) can have a null value but the values cannot be duplicated.

**Syntax to define a Unique key at column level:**

*[CONSTRAINT constraint\_name] UNIQUE*

**Syntax to define a Unique key at table level:**

*[CONSTRAINT constraint\_name] UNIQUE(column\_name)*

## SQL Check Constraint : This constraint defines a business rule on a column. All the rows must satisfy this rule. The constraint can be applied for a single column or a group of columns.

**Syntax to define a Check constraint:**

*[CONSTRAINT constraint\_name] CHECK (condition)*

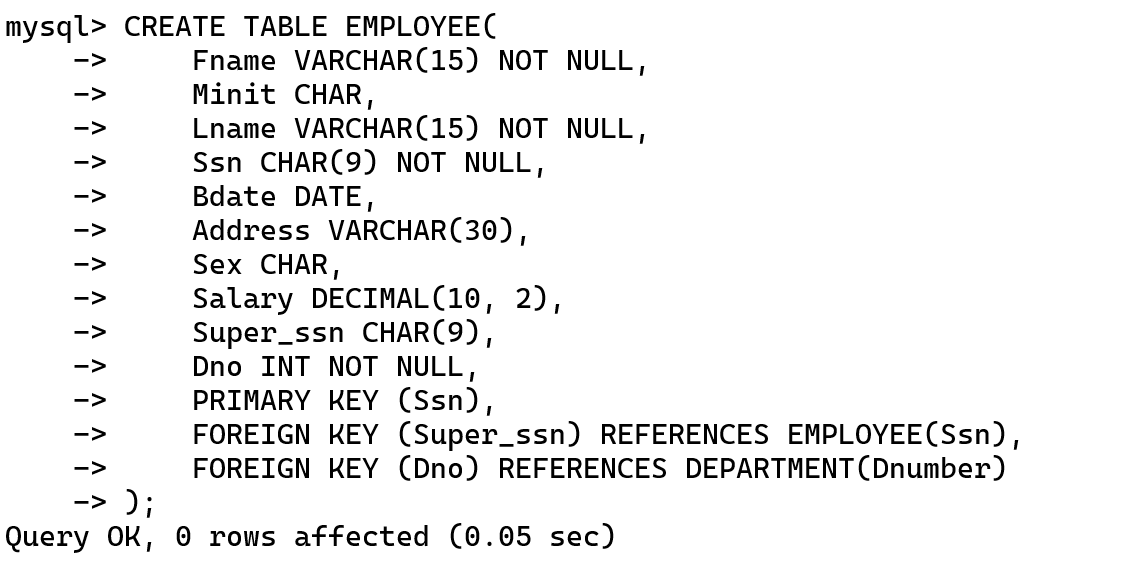
**Output: Screenshots**

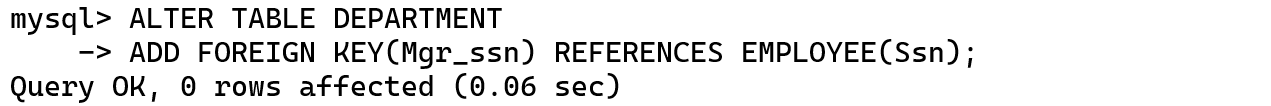
**Create Table Department**

Text

Description automatically generated

**Create Table Employee**



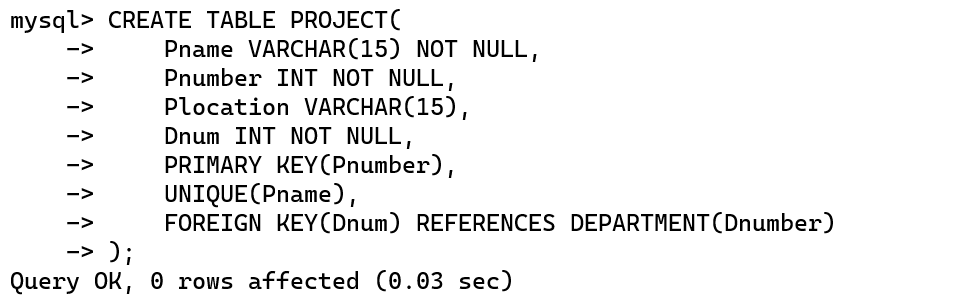


**Create Table Department Locations**

**Text

Description automatically generated with medium confidence**

**Create Table Projects**

****

**Create Table Works On**

**Text

Description automatically generated**

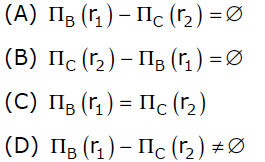
**Create Table Dependent**

**Text

Description automatically generated**

**Preparatory Questions**

1. Suppose (A, B) and (C,D) are two relation schemas. Let r1 and r2 be the corresponding relation instances. B is a foreign key that refers to C in r2. If data in r1 and r2 satisfy referential integrity constraints, which of the following is ALWAYS TRUE?



1. A
2. B
3. C
4. D
5. Given the following statements:

S1: A foreign key declaration can always be replaced by an equivalent check assertion in SQL.

S2: Given the table R(a,b,c) where a and b together form the primary key, the following is a valid table definition.

*CREATE TABLE S (*

*a INTEGER,*

*d INTEGER,*

*e INTEGER,*

*PRIMARY KEY (d),*

*FOREIGN KEY (a) references R)*

Which one of the following statements is CORRECT?

1. S1 is TRUE and S2 is FALSE.
2. Both S1 and S2 are TRUE.
3. S1 is FALSE and S2 is TRUE.
4. Both S1 and S2 are FALSE

Q3) Which of the following is not an integrity constraint?  
 1) Not null  
 2) Positive  
 3) Unique  
 4) Check ‘predicate’

Q4) *CREATE TABLE Manager(ID NUMERIC,Name VARCHAR(20),budget NUMERIC,Details VARCHAR(30));*

In order to ensure that the value of budget is non-negative which of the following should be used?  
 1) Check(budget>0)  
 2) Check(budget<0)  
 3) Alter(budget>0)  
 4) Alter(budget<0)

Q5) The following table has two attributes A and C where A is the primary key and C is the foreign key referencing A with on-delete cascade.

|  |  |
| --- | --- |
| A | C |
| 2 | 4 |
| 3 | 4 |
| 4 | 3 |
| 5 | 2 |
| 7 | 2 |
| 9 | 5 |
| 6 | 4 |

The set of all tuples that must be additionally deleted to preserve referential integrity when the tuple (2,4) is deleted is:

1. (3,4) and (6,4)
2. (5,2) and (7,2)
3. (5,2), (7,2) and (9,5)
4. (3,4), (4,3) and (6,4)

**Experiment No: 4**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section:** Semester-IV – AIML-B (AL-3) |
| **Link to Code:** [Piyush-Gambhir/NCU-CSL214-DBMS-Lab\_Manual (github.com)](https://github.com/Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual) |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

**Objective**

To familiarize with SELECT-FROM-WHERE SQL simple queries on the COMPANY database.

**Program Outcome**

* The students will be able to retrieve zero or more rows from one or more [database tables](https://en.wikipedia.org/wiki/Database_Tables) or database [views](https://en.wikipedia.org/wiki/View_(database)).

**Problem Statement**

From the COMPANY database as mentioned and described in the previous database:

1. Retrieve the birth date and address of the employee(s) whose name  
   is ‘John B. Smith’.
2. Retrieve the name and address of all employees who work for the ‘Research’ department.
3. For every project located in ‘Stafford’, list the project number, the  
   controlling department number, and the department manager’s last name,  
   address, and birth date.
4. Select all combinations of EMPLOYEE Ssn and DEPARTMENT Dname in the database.
5. Retrieve all the attribute values of any EMPLOYEE who works in DEPARTMENT number 5.
6. Retrieve all distinct salary values.
7. Make a list of all project numbers for projects that involve an employee whose last name is ‘Smith’, either as a worker or as a manager of the department that controls the project.
8. Retrieve all employees whose address is in Houston, Texas.
9. Find all employees who were born during the 1950s
10. Show the resulting salaries if every employee working on the ‘ProductX’ project is given a 10 percent raise.
11. Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.

**Background Study**

1. Structured Query Language SQL contains statements for data definitions, queries, and updates (both DDL and DML)
2. Domain

* Name used with the attribute specification.
* Makes it easier to change the data type for a domain that is used by numerous attributes.

1. Inserting values in our table using the commands:

*INSERT INTO EMPLOYEE (Fname, Lname, Dno, Ssn)*

*VALUES (‘Richard’, ‘Marini’, 4, ‘653298653’)*

1. Out of the complete Database we create we can easily pick out and filter certain amount of data by using the SQL queries as :

*SELECT <attribute list>*

*FROM <table list>*

*WHERE <condition>;*

* <attribute list> is a list of attribute names whose values are to be retrieved by the query.
* <table list> is a list of the relation names required to process the query.
* <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.

1. Ambiguous Attribute Names: Same name can be used for two (or more) attributes as long as the attributes are in different relations, these are made to differ by mentioning their table names before the attribute names.
2. Aliasing, Renaming Tuple variables: We can rename the tables into some smaller easier names as per choice when it has to be used multiple times in the query.
3. DISTINCT keyword is used when the query wishes to derive no two duplicate values.
4. SET operations such as UNION and INTERSECT can also be applied on the tables to filter out the values as per choice.
5. Substring Pattern matching is carried out by the use of the keyword LIKE which helps in retrieving the column values of the tuple matching our mentioned substring.
6. ORDERBY <attributes>

* Helps in sorting the tuple values of the attributes by default set to ascending and can be changed to descending .

**Output: Screenshots**



Text

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated with low confidence

Text, letter

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

Table

Description automatically generated

Table

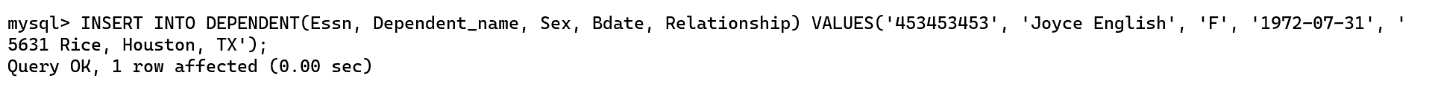
Description automatically generated with medium confidence

Text, letter

Description automatically generated

Text

Description automatically generated with medium confidence



Text, email

Description automatically generatedText, letter

Description automatically generatedText

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generatedBackground pattern

Description automatically generated with low confidenceTable

Description automatically generated

Table

Description automatically generatedTable

Description automatically generatedText, letter

Description automatically generated

**Preparatory Questions**

Q1) Which operator performs Pattern matching?

1. BETWEEN operators
2. LIKE operator
3. EXISTS operator.
4. None of the above

Q2) In SQL which commands are used to change the storage characteristics of the table?

1. ALTER TABLE
2. MODIFY TABLE
3. CHANGE TABLE
4. All of the above

Q3) \_\_\_\_\_\_\_\_\_\_\_removes all rows from a table without logging the individual row deletions.  
 1) DELETE  
 2) REMOVE  
 3) DROP  
 4) TRUNCATE

Q4) If you don’t specify ASC or DESC after a SQL ORDER BY clause, the following is used by default :   
1) ASC  
2) DESC  
3) There is no default value  
4) None of the mentioned

Q5) What is the purpose of the SQL AS clause?  
1) The AS SQL clause is used to change the name of a column in the result set or to assign a name to a derived column  
2) The AS clause is used with the JOIN clause only  
3) The AS clause defines a search condition  
4) All of the mentioned

**Experiment No: 5**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section:** Semester-IV – AIML-B (AL-3) |
| **Link to Code:** [Piyush-Gambhir/NCU-CSL214-DBMS-Lab\_Manual (github.com)](https://github.com/Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual) |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

**Objective**

To familiarize with JOIN operations in SQL on the COMPANY database.

**Program Outcome**

* The students will be to establish a connection between two or more database tables based on matching columns, thereby creating a relationship between the tables.

**Problem Statement**

Consider the Sample database given below and answer the queries as stated:

**Pack\_grades**

|  |  |  |  |
| --- | --- | --- | --- |
| Grade\_id | Grade\_name | Min\_price | Max\_price |

**Customers**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Customer\_id | First\_name | Last\_name | Birth\_date | Join\_date | City | Pack\_id | State |

**Packages**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pack\_id | Speed | Start\_date | Monthly\_payment | Sector\_id |

**Sectors**

|  |  |
| --- | --- |
| Sector\_id | Sector\_name |

1. Write a query to display first name, last name, package number and internet speed for all customers.
2. Write a query to display first name, last name, package number and internet speed for all customers whose package number equals 22 or 27. Order the query in ascending order by last name.
3. Display the package number, internet speed, monthly payment and sector name for all packages (*Packages* and *Sectors* tables).
4. Display the customer name, package number, internet speed, monthly payment and sector name for all customers (*Customers*, *Packages* and *Sectors* tables).
5. Display the customer name, package number, internet speed, monthly payment and sector name for all customers in the business sector (*Customers*, *Packages* and *Sectors* tables).
6. Display the last name, first name, join date, package number, internet speed and sector name for all customers in the private sector who joined the company in the year 2006.
7. Display the package number, internet speed, monthly payment and package grade for all packages (*Packages* and *Pack\_Grades* tables).
8. Display the first name, last name, internet speed and monthly payment for all customers. Use INNER JOIN to solve this exercise.
9. Display the last name, first name and package number for all customers who have the same package number as customer named ‘Amado Taylor’ (*Customers* table).
10. Display the package number and internet speed for all packages whose internet speed is equal to the internet speed of package number 10 (*Packages* table).

**Background Study**

1. JOINED TABLE: Permits users to specify a table resulting from a join operation in the FROM clause of a query

* The attributes of such a table are all the attributes of the first table followed by all the attributes of the second table.
* The default type of join in a joined table is called an inner join, where a tuple is included in the result only if a matching tuple exists in the other relation.

1. NATURAL JOIN on two relations R and S

* No join condition specified.
* Implicit EQUIJOIN condition for each pair of attributes with same name from R & S
* It is possible to rename the attributes so that they match, if the names of the join attributes are not the same in the base relations.

1. LEFT OUTER JOIN

* Every tuple in left table must appear in result
* If no matching tuple, padded with NULL values for attributes of right table

1. RIGHT OUTER JOIN

* Every tuple in right table must appear in result
* If no matching tuple, padded with NULL values for the attributes of left table

1. FULL OUTER JOIN

* Used when both the tables are taken completely

1. CROSS JOIN – for Cartesian Product

**Output: Screenshots**

Background pattern

Description automatically generated with low confidenceA picture containing graphical user interface

Description automatically generatedText

Description automatically generated with medium confidence

Text, letter

Description automatically generatedTable

Description automatically generated

A picture containing text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generatedTable

Description automatically generated

Text

Description automatically generatedText

Description automatically generated with medium confidenceText, letter, email

Description automatically generated

**Preparatory Questions**

Q1) Database table by name Loan\_Records is given below.

Borrower Bank\_Manager Loan\_Amount

Ramesh Sunderajan 10000.00

Suresh Ramgopal 5000.00

Mahesh Sunderajan 7000.00

What is the output of the following SQL query?

*SELECT Count(\*)*

*FROM ( ( SELECT Borrower, Bank\_Manager*

*FROM Loan\_Records) AS S*

*NATURAL JOIN ( SELECT Bank\_Manager, Loan\_Amount*

*FROM Loan\_Records) AS T );*

1) 3

2) 9

3) 5

4) 6

Q2) Which product is returned in a join query have no join condition:  
1) Equijoins  
2) Cartesian  
3) Both Equijoins and Cartesian  
4) None of the mentioned

Q3) Which join refers to join records from the write table that have no matching key in the left table are include in the result set:  
1) Left outer join  
2) Right outer join  
3) Full outer join  
4) Half outer join

Q4) Which operation are allowed in a join view:  
1) UPDATE  
2) INSERT  
3) DELETE  
4) All of the mentioned

Q5) Which view that contains more than one table in the top-level FROM clause of the SELECT statement:  
1) Join view  
2) Datable join view  
3) Updatable join view  
4) All of the mentioned

**Experiment No: 6**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section:** Semester-IV – AIML-B (AL-3) |
| **Link to Code:** [Piyush-Gambhir/NCU-CSL214-DBMS-Lab\_Manual (github.com)](https://github.com/Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual) |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

**Objective**

To understand Aggregate functions using SQL queries on the COMPANY database.

**Program Outcome**

* The students will understand the need of applying an aggregate function to get a single value from a set of values, thus, expressing the significance of the data it is computed from.

**Problem Statement**

Consider the COMPANY database in Experiment 1 and execute the following queries:

1. Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary.
2. Find the sum of the salaries of all employees of the ‘Research’ department, as well as the maximum salary, the minimum salary, and the average salary in this department.
3. Retrieve the total number of employees in the company.
4. Retrieve the number of employees in the ‘Research’ department.
5. Count the number of distinct salary values in the database.
6. For each department, retrieve the department number, the number of employees in the department, and their average salary.
7. For each project, retrieve the project number, the project name, and the number of employees who work on that project.
8. For each project *on which more than two employees work,* retrieve the project number, the project name, and the number of employees who work on the project
9. For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project.
10. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than $40,000.

**Background Study**

1. Aggregate Functions: Used to summarize information from multiple tuples into a single-tuple summary
2. Grouping: Create subgroups of tuples before summarizing
3. Built-in aggregate functions: COUNT*,* SUM, MAX, MIN, and AVG (NULL values discarded when aggregate functions are applied to a particular column)
4. Functions can be used in the SELECT clause or in a HAVING clause
5. Partition relation into subsets of tuples

* Based on grouping attribute(s)
* Apply function to each such group independently

1. GROUP BY clause: Specifies grouping attributes
2. If NULLs exist in grouping attribute, then separate group created for all tuples with a NULL value in grouping attribute
3. HAVING clause provides a condition on the summary information
4. SELECT clause includes only the grouping attribute and the aggregate functions to be applied on each group of tuples.
5. WHERE clause limit the *tuples* to which functions are applied, the HAVING clause serves to choose *whole groups*

**Output: Screenshots**

Background pattern

Description automatically generated with low confidenceText

Description automatically generated with medium confidenceText, letter

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generatedTable

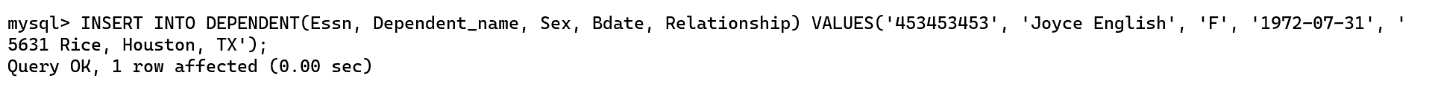
Description automatically generated

Table

Description automatically generated with medium confidenceText, letter

Description automatically generated

Text

Description automatically generated with medium confidenceGraphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, letter, email

Description automatically generated

Text

Description automatically generated with low confidenceGraphical user interface, text, application, email

Description automatically generated

**Preparatory Questions**

Q1) Which of the following statements are TRUE about an SQL query? P: An SQL query can contain a HAVING clause even if it does not have a GROUP BY clause Q : An SQL query can contain a HAVING clause only if it has a GROUP BY clause R : All attributes used in the GROUP BY clause must appear in the SELECT clause S : Not all attributes used in the GROUP BY clause need to appear in the SELECT clause

1. P and R
2. P and S
3. Q and R
4. Q and S

Q2) Consider a database table T containing two columns X and Y each of type integer. After the creation of the table, one record (X=1, Y=1) is inserted in the table. Let MX and My denote the respective maximum values of X and Y among all records in the table at any point in time. Using MX and MY, new records are inserted in the table 128 times with X and Y values being MX+1, 2\*MY+1 respectively. It may be noted that each time after the insertion, values of MX and MY change. What will be the output of the following SQL query after the steps mentioned above are carried out?

SELECT Y FROM T WHERE X=7;

1. 127
2. 255
3. 129
4. 257

Q3) The employee information in a company is stored in the relation

*Employee (name, sex, salary, deptName)*

Consider the following SQL query

*select deptName*

*from Employee*

*where sex = 'M'*

*group by deptName*

*having avg (salary) > (select avg (salary) from Employee)*

It returns the names of the department in which

|  |  |
| --- | --- |
| 1) | the average salary is more than the average salary in the company |
| 2) | the average salary of male employees is more than the average salary of all male employees in the company |
| 3) | the average salary of male employees is more than the average salary of employees in the same department |
| 4) | the average salary of male employees is more than the average salary in the company |

Q4) Which of the following is aggregate function in SQL?

1) Avg

2) Select

3) Ordered by

4) distinct

Q5) Observe the given SQL query and choose the correct option.

***SELECT*** *branch\_name,* ***COUNT*** *(****DISTINCT*** *customer\_name)*

***FROM*** *depositor, account*

***WHERE*** *depositor.account\_number = account.account\_number*

***GROUP******BY*** *branch\_id*

1) The query is syntactically correct but gives the wrong answer  
2) The query is syntactically wrong  
3) The query is syntactically correct and gives the correct answer  
4) The query contains one or more wrongly named clauses.

**Experiment No: 7**

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Link to Code:** |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

**Objective**

To familiarize with nested SQL queries on the COMPANY database.

**Program Outcome**

* The students will understand how to devise independent and correlated nested queries.

**Problem Statement**

Consider the COMPANY database in Experiment 1 and execute the following queries:

1. Make a list of all project numbers for projects that involve an employee whose last name is ‘Smith’, either as a worker or as a manager of the department that controls the project.
2. Select the Essns of all employees who work on the same project and hours as some project that employee ‘John Smith’ (whose Ssn = ‘123456789’) works on.
3. Return the names of employees whose salary is greater than the salary of all the employees in department 5
4. Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee
5. Retrieve the names of employees who have no dependents
6. List the names of managers who have at least one dependent using EXISTS and NOT EXISTS functions
7. Retrieve the name of each employee who works on all the projects controlled by department number 5 using EXISTS and NOT EXISTS functions
8. Retrieve the names of all employees who have two or more dependents
9. Retrieves the names of all employees who work on only one project

**Background Study**

1. SQL allows queries that check whether an attribute value is NULL: IS or IS NOT NULL
2. Nested Queries: Complete select-from-where blocks within WHERE clause of another query

* Nested Queries generally return a table (relation)

1. Comparison Operator IN:

* Compares value *v* with a set (or multiset) of values *V*
* Evaluates to TRUE if *v* is one of the elements in *V*

1. = ANY (or = SOME) Operator: Returns TRUE if the value *v* is equal to some value in the set *V* (equivalent to IN)
2. ALL Operator: (*v* > ALL *V*) returns TRUE if the value *v* is greater than *all* the values in the set (or multiset) *V*.

* Other operators that can be combined with ANY (or SOME) and ALL: >, >=, <, <=, and <>

1. Possible ambiguity among attribute names if attributes of the same name exist—one in a relation in the FROM clause of the outer query, and another in a relation in the FROM clause of the nested query.

* Thumb Rule: reference to an unqualified attribute refers to the relation declared in the innermost nested query.

1. Whenever a condition in the WHERE clause of a nested query references some attribute of a relation declared in the outer query, the two queries are said to be correlated.

* In a correlated query, the nested query is evaluated once for each tuple (or combination of tuples) in the outer query

1. EXISTS function: Check whether the result of a correlated nested query is empty or not.
2. EXISTS and NOT EXISTS are typically used in conjunction with a correlated nested query.
3. EXISTS(Q): returns TRUE if there is at least one tuple in the result of the nested query Q, and it returns FALSE otherwise.

1. NOT EXISTS(Q): returns TRUE if there are no tuples in the result of nested query Q, and it returns FALSE otherwise.

**Output: Screenshots**

**Preparatory Questions**

Q1) Database table by name Loan\_Records is given below.

Borrower Bank\_Manager Loan\_Amount

Ramesh Sunderajan 10000.00

Suresh Ramgopal 5000.00

Mahesh Sunderajan 7000.00

What is the output of the following SQL query?

*SELECT Count(\*)*

*FROM ( ( SELECT Borrower, Bank\_Manager*

*FROM Loan\_Records) AS S*

*NATURAL JOIN ( SELECT Bank\_Manager, Loan\_Amount*

*FROM Loan\_Records) AS T );*

1. 3
2. 9
3. 5
4. 6

Q2) A relational schema for a train reservation database is given below. Passenger (pid, pname, age) Reservation (pid, class, tid)

Table: Passenger

pid pname age

-----------------

0 Sachin 65

1 Rahul 66

2 Sourav 67

3 Anil 69

Table : Reservation

pid class tid

---------------

0 AC 8200

1 AC 8201

2 SC 8201

5 AC 8203

1 SC 8204

3 AC 8202

What pids are returned by the following SQL query for the above instance of the tables?

*SELECT pid*

*FROM Reservation ,*

*WHERE class ‘AC’ AND*

*EXISTS (SELECT \**

*FROM Passenger*

*WHERE age > 65 AND*

*Passenger. pid = Reservation.pid)*

1. 1,0
2. 1,2
3. 1,3
4. 1,5

Q3) Consider the following relational schema:

Suppliers(sid:integer, sname:string, city:string, street:string)

Parts(pid:integer, pname:string, color:string)

Catalog(sid:integer, pid:integer, cost:real)

Consider the following relational query on the above database:

*SELECT S.sname*

*FROM Suppliers S*

*WHERE S.sid NOT IN (SELECT C.sid*

*FROM Catalog C*

*WHERE C.pid NOT IN (SELECT P.pid*

*FROM Parts P*

*WHERE P.color<> 'blue'))*

Assume that relations corresponding to the above schema are not empty. Which one of the following is the correct interpretation of the above query?

|  |  |
| --- | --- |
| 1) | Find the names of all suppliers who have supplied a non-blue part. |
| 2) | Find the names of all suppliers who have not supplied a non-blue part. |
| 3) | Find the names of all suppliers who have supplied only blue parts. |
| 4) | Find the names of all suppliers who have not supplied only blue parts. |
| 5) | None |

Q4) Consider the table employee(empId, name, department, salary) and the two queries Q1 ,Q2 below. Assuming that department 5 has more than one employee, and we want to find the employees who get higher salary than anyone in the department 5, which one of the statements is TRUE for any arbitrary employee table?

Query1: *Select e.empId*

*From employee e*

*Where not exists*

*(Select \* From employee s where s.department = “5” and s.salary >=e.salary)*

Query2: *Select e.empId*

*From employee e*

*Where e.salary > Any*

*(Select distinct salary From employee s Where s.department = “5”)*

1. Q1 is the correct query
2. Q2 is the correct query
3. Both Q1 and Q2 produce the same answer.
4. Neither Q1 nor Q2 is the correct query

Q5) Consider the following relational schema:

employee(empId, empName, empDept)

customer(custId, custName, salesRepId, rating)

salesRepId is a foreign key referring to empId of the employee relation. Assume that each employee makes a sale to at least one customer. What does the following query return?

*SELECT empName*

*FROM employee E*

*WHERE NOT EXISTS ( SELECT custId*

*FROM customer C*

*WHERE C.salesRepId = E.empId*

*AND C.rating <> `GOOD`);*

|  |  |
| --- | --- |
| 1) | Names of all the employees with at least one of their customers having a ‘GOOD’ rating. |
| 2) | Names of all the employees with at most one of their customers having a ‘GOOD’ rating. |
| 3) | Names of all the employees with none of their customers having a ‘GOOD’ rating. |
| 4) | Names of all the employees with all their customers having a ‘GOOD’ rating. |

**Experiment 8**

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Link to Code:** |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

**Objective**

Identifying contrast between Relational Databases and NoSQL, thereby recognizing their applications.

**Program Outcome**

* The students will install MongoDB shell and familiarize themselves with it.

**Problem Statement**

MongoDB installation and shell familiarity.

**Background Study**

MongoDB is a Schema less database. A database in MongoDB contains collections and inside collections we have documents.

* Database is a physical container for collections. A single MongoDB server typically has multiple databases.
* Collection is a group of MongoDB documents.
* Documents within a collection can have different fields. A document is a set of key-value pairs and have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

Diagram, text

Description automatically generated

**Output: Screenshots**

**Preparatory Questions**

Q1) Point out the correct statement:

1) A database is a set of key-value pairs

2) A MongoDB deployment hosts a number of databases

3) A document holds a set of collections

4) All of the mentioned

Q2) MongoDB stores all documents in:

1) tables

2) collections

3) rows

4) all the mentioned

Q3) BSON is a binary representation of \_\_\_\_\_\_\_\_ documents,

1) JSON

2) XML

3) JScript

4) All of the mentioned

Q4) The maximum size of a MongoDB document is

1) 2 MB

2) 16 MB

3) 12 MB

4) There is no maximum size. It depends on RAM

Q5) Which of the following statements is true?

1) MongoDB cannot be used as a file system.

2) MongoDB can run over single servers only.

3) Embedded documents and arrays reduce need for joins.

4) None

**Experiment 9**

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Link to Code:** |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

**Objective**

Create COMPANY database using NoSQL database - MongoDB.

**Program Outcome**

* The students will be able to perform basic CRUD operations in MongoDB.

**Problem Statement**

1. Create a COMPANY database with the following collections:

* Employee with fields Emp ID, Ename, Age, Mobile, Email, Address, Dno
* Department with fields Dnumber, Dname, Dlocation
* Project with fields Pname, Pnumber, Plocation, ControlDept

1. Insert 5 documents in each collection using different versions of “Insert” command.
2. Select all documents in a collection.
3. Retrieve the details of all projects that are being run in department number 50.
4. Retrieve the details of the first project that is being run in department number 50 in formatted manner.
5. Return the formatted/structured project details of all projects in department number “5” and project name as “ProductZ”.
6. Return the project details (including project name, project number and department number and excluding project location and default primary key) of all projects with department number “5” or project name as “ProductZ”.
7. Return the formatted/structured project details of all projects with department number less than “10” and project name as “ProductZ” or project location as “Delhi”.

**Background Study**

1. **use:** This command is used to create a database in MongoDB. It will return an existing database or will create a new database if it dosen’t exist.
2. **createCollection( ):** To create a collections in a database
3. MongoDB provides the following methods to insert documents into a collection:
4. **db.collection.insert():** used to insert one or multiple documents. To insert multiple documents in a single query, pass an array of documents in insert() command.
5. **db.collection.insertOne():** used to insert only one document.
6. **db.collection.insertMany():** used to insert multiple documents. To insert multiple documents in a single query, pass an array of documents in insert() command.
7. **db.collection.find():**  this method is provided in MongoDB to read documents from a collection. It returns a **cursor** to the matching documents. db.collection.find(<query filter>, <projection>) accepts second optional parameter that is list of fields that you want to retrieve in the form { field1: <value>, field2: <value> ... }.
8. **db.collection.findOne()**: it returns the first occurrence of document, otherwise null.

**Output: Screenshots**

**Preparatory Questions**

Q1) Which of the following method is used to query documents in collections?

1) find

2) move

3) shell

4) replace

Q2) When you query a collection, MongoDB returns a \_\_\_\_\_\_\_\_ object that contains the results of the query.

1) row

2) cursor

3) colums

4) none of the mentioned

Q3) Which of the following method is called while accessing documents using the array index notation ?

1) cur.toArray()

2) cursor.toArray()

3) doc.toArray()

4) all of the mentioned

Q4) The mongo shell and the drivers provide several cursor methods that call on the cursor returned by the \_\_\_\_\_\_\_ method to modify its behavior.

1) cursor()

2) find()

3) findc()

4) none of the mentioned

Q5) Which of the following method corresponds to Order by clause in SQL?

1) sort()

2) order()

3) orderby()

4) all of the mentioned

**Experiment 10**

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Link to Code:** |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

**Objective**

Retrieve data from NoSQL database - MongoDB.

**Program Outcome**

* The students will be able to perform advanced CRUD operations in MongoDB.

**Problem Statement**

Create a database with the following collections:

**Project**

Possible Fields:

* Project Name (Pname)
* Project Number (Pnumber)
* Department Number in which project is being run (Dnum)
* Project Locations (Plocation)
  + Plocation can be an array of multiple locations (or)
  + Plocation can be a composite field with sub-fields – city, state, country

**Department**

Possible Fields:

* Department Number (Dno)
* Department Name (Dname)
* Department Manager (Dmanager)

**Execute the following queries:**

1. Insert multiple documents (2-3) together in the Project collection with atleast 1 project location as a composite attribute
2. Write the “Select \* from Project” equivalent query in MongoDB.
3. Write a MongoDB query to update the project location of all projects to “Pune” and department number to “20” with project name “ProjectX”.
4. Write a MongoDB query to delete all the documents of collection “Project” with department number “4”.
5. Write a MongoDB query to retrieve the project details of all projects with project location as “Delhi” and “Mumbai” both assuming the following documents exist.

{ "\_id" : ObjectId("5cc7fe36e37496f1a5e7e268"),"Pname" : "ProductK", "Pnumber" : 45,"Plocation" : ["Delhi","Mumbai","Pune"], "Dnum" : 3}

{ "\_id" : ObjectId("5cc7fe36e37496f1a5e7e268"),"Pname" : "ProductT", "Pnumber" : 56,"Plocation" : ["Delhi","Mumbai"], "Dnum" : 15}

{ "\_id" : ObjectId("5cc7fe36e37496f1a5e7e268"),"Pname" : "ProductU", "Pnumber" : 5,"Plocation" : "Delhi", "Dnum" : 6}

{ "\_id" : ObjectId("5cc7fe36e37496f1a5e7e268"),"Pname" : "ProductL", "Pnumber" : 4,"Plocation" : "Mumbai", "Dnum" : 7}

1. Write a MongoDB query creating a One to Many Relation between a document of “Project” Collection and “Department” Collection.
2. Write a MongoDB query to retrieve the project details 🡪 project name, department number, department name and department manager of projects with department manager (for department in which the project is being run) as “ABC”.

**Background Study**

1. pretty() is used to display the query obtained in a neat manner with all the fields clearly depicted one below the other.
2. Data for a particular field can also be stored as an array with multiple values for the field.
3. $all is used to match the array items for the particular field.

* *db.project.find({Plocation:{$all:["Delhi","Mumbai"]}}).pretty()*

1. $elemMatch: This operator matches documents that contain an array field with at least one element that matches all the specified query criteria.

* Syntax: { <field>: { $elemMatch: { <query1>, <query2>, ... } } }

1. Fields can be embedded in the field too and while using find in such a case, always refer the field embedded in another field by using a dot notation and in quotes.
2. AND/OR Operations
3. Documents can either be embedded in one to one relation or one to many relation

* In one to one, the new field contains only a single embedded document.
* In one to many, the new field contains an array of multiple embedded documents.

**Output: Screenshots**

**Preparatory Questions**

Q1) The order of documents returned by a query is not defined unless you specify a \_\_\_\_\_\_  
 1) sortfind()  
 2) sortelse()  
 3) sort()  
 4) none of the mentioned

Q2) In aggregation pipeline, the \_\_\_\_\_\_\_ pipeline stage provides access to MongoDB queries.  
 1) $catch  
 2) $match  
 3) $batch  
 4) All of the mentioned

Q3) Point out the wrong statement.  
 1) sort() modifier sorts the results by age in ascending order  
 2) Queries in MongoDB return all fields in all matching documents by default  
 3) To scale the amount of data that MongoDB sends to applications, include a projection in the queries.  
 4) None of the mentioned

Q4) Which of the following query selects documents in the records collection that match the condition { “user\_id”: { $lt: 42 } }?  
 1) db.records.findOne( { “user\_id”: { $lt: 42 } }, { “history”: 0 } )  
 2) db.records.find( { “user\_id”: { $lt: 42 } }, { “history”: 0 } )  
 3) db.records.findOne( { “user\_id”: { $lt: 42 } }, { “history”: 1 } )  
 4) db.records.select( { “user\_id”: { $lt: 42 } }, { “history”: 0 } )

Q5) Which of the following is not a projection operator?  
 1) $slice  
 2) $elemMatch  
 3) $  
 4) None of the mentioned

**Value Added Experiment**

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Link to Code:** |
| **Date:** |
| **Faculty Signature:** |
| **Marks:** |

**Objective**

Design an ER/ EER Diagram, relational schema and implement the database in MongoDB.

**Program Outcome**

* The students will design and implement DBMS concepts on a real-world project.

**Problem Statement**

The NorthCap University has recently launched new specializations in B-Tech (Computer Science & Engineering). To facilitate this, a new admission application needs to be developed, where an Admin should be able to create a new application, update an existing application and delete any application. The application should contain the below mentioned fields:

1) Student Name

2) Student Id

3) Father’s Name

4) Mother’s Name

5) Gender

6) DOB

7) Nationality

8) Address

9) Marks in Class X. XII and JEE.

10) Email

11) Contact No.

12) B-Tech CSE Specialization Choice 1 (IOT, Data Science, Cyber Security, Full Stack and Game

Tech)

13) B-Tech CSE Specialization Choice 2 (IOT, Data Science, Cyber Security, Full Stack and Game

Tech)

Based on the application requirements and your specific assumptions, draw an ER/EER model for the application’s database and further map it to appropriate relational schema.

**Database for B.Tech Specialization**

**Collections**

1. Student

Possible Fields:

* Student Name (Sname)
* Student Id should be unique (Sid)
* Address
* Address can be an array of multiple address(or)
* Address can be a composite field with sub field-city, state, country
* Marks in JEE. (Marks\_JEE)
* B-Tech CSE Specialization Choice 1 (Choise\_1)
* B-Tech CSE Specialization Choice 2 (Choise\_2)

2. Specialization

Possible Fields:

* B-Tech CSE Specialization Choice IOT (IOT)
* B-Tech CSE Specialization Data Science (Data\_Science)
* B-Tech CSE Specialization Cyber Security (Cyber\_Security)
* B-Tech CSE Specialization Full Stack (Full\_Stack)
* B-Tech CSE Specialization Game Tech (Game\_Tech)

**Queries:**

* + - 1. Insert multiple documents (2-3) together in the Student collection with at least 1 student Address as a composite attribute.
      2. Write the “Select \* from Student” equivalent query in MongoDB.
      3. Write a MongoDB query to return the formatted/structured Student details of all Students with Specialization choice 1 “IOT” and Student name as “RAM”.
      4. Write a MongoDB query to return the formatted/structured Student details (including Student name, choice1 and choice2 and excluding other fields) of all students with choice1 as “IOT” or Student name as “RAM”.
      5. Write a MongoDB query to return the formatted/structured Student details of all students with choice1 equal to “Full\_Stack” and Student JEE Marks less than 200 or student address as “Delhi”.
      6. Write a MongoDB query to update the student choice1 to “IOT” and JEE Marks to “80” with student name “XYZ”.
      7. Write a MongoDB query to delete all the documents of collection “student” with choice1 as “IOT”.
      8. Write a MongoDB query to retrieve the Student details of all students with address as “Delhi” and “Mumbai”.
      9. Write a MongoDB query creating a One to Many Relation between a document of“student” Collection and “Specialization” Collection.
      10. Write a MongoDB query to retrieve the student details  student name, Marks\_JEE, of student with specialization as Cyber\_Security (for department in which the project is being run) as “ABC”.

**Output: Screenshots**