PAGE NO. :
DATE : / /

# Practical No. 01

* Aim : Design database using Network Model, Hierarchical model, Relational model and E-R model

* Theory :

## Database model -

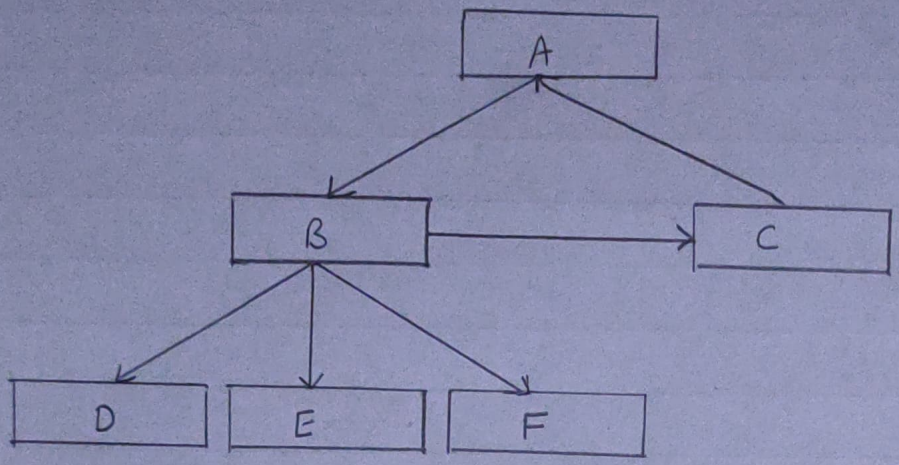
A database model is a type of model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organised and manipulated. The most popular example of database model is relational model.

## Network model -

- This model uses pointers towards data but there is no need of parent to child association so it does not necessarily use a downward tree structure.
- This model is used in network databases.
- This database model is similar to hierarchical model up to some aspects.
- There are some concepts involved in network model as follows:

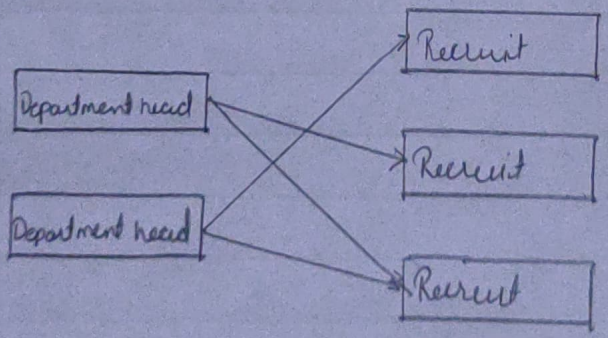
i) SET - A relationship between any two record types is called as a set.

ii) Record - ① Owner record is like parent record in hierarchical model
② Parent record is like child record in hierarchical model




NETWORK MODEL

Example: Two department heads giving instruction to the new recruits!

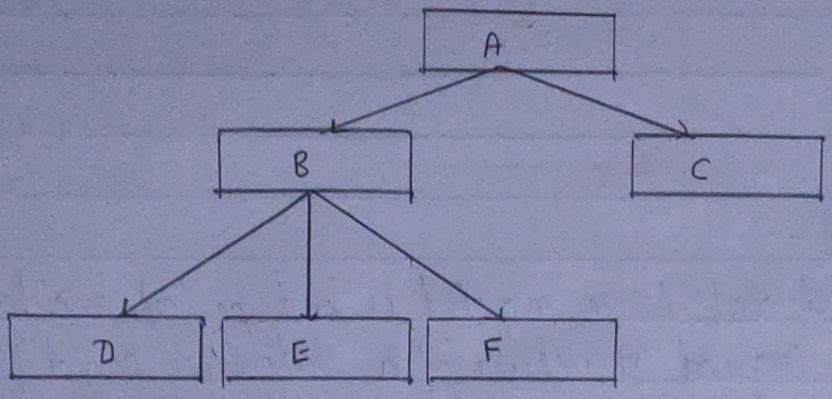


PAGE NO.:
DATE: / /

- A relationship model can be 1:N, or M:N relationship.

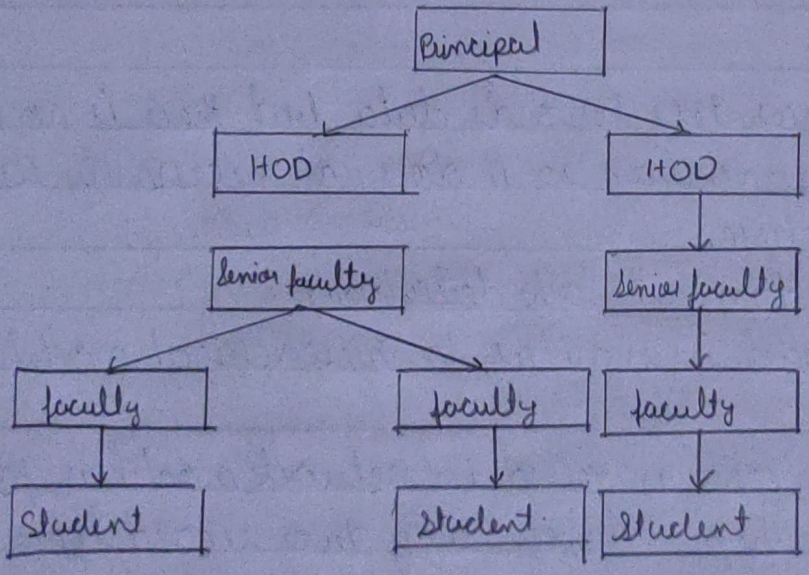
Hierarchical Model –

- It was developed by joint efforts of IBM and North American Rockwell known as Information Management System.
- It was the first DBMS model.
- The data is stored hierarchically, either in top down or bottom up approach of designing.
- This model uses pointers to navigate between stored data.
- This model represents data as a hierarchical tree.
- This business rule of the Hierarchical model states that - one parent node may have many child nodes, but one child node cannot have more that one parent.
- Advantages of hierarchical model
  1. Conceptual simplicity.
  2. Simple creation, updation and access
  3. Database security.
  4. Database integrity.
  5. Data independence.
  6. Efficiency
- Disadvantages of hierarchical model.
  1. Complex implementation
  2. Difficult to manage.
  3. Lack of structural independence
  4. Complex application programming
  5. Limitations in implementation




Hierarchical Model

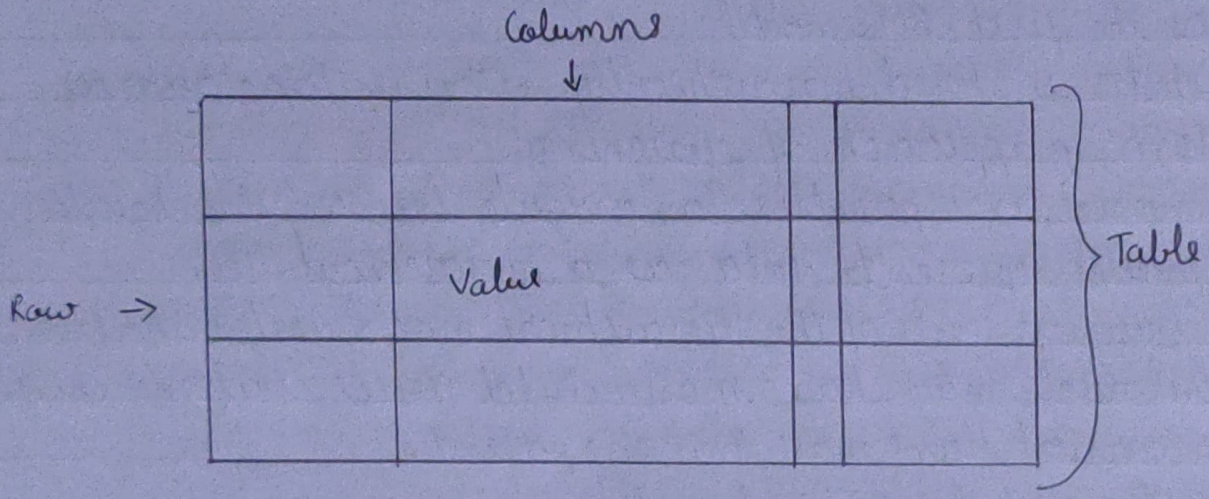
Example: Let us consider simple organisational structure.



PAGE NO. :
DATE : / /

## Relational model -

- The relational model was first proposed by E.F. Coddthence.
- Relational database was an attempt to simplify database by making use of tables and columns.
- Tables are known as 'relations'.
- Columns are known as 'attributes'.
- Rows are known as 'tuples'.
- A relational database is a collection of 2-dimensional tables consisting of rows and columns
- This model we uses collection of tables to represent relationship amongst the data.
- In this model, each database item is viewed as record with attributes.
- A set of records with similar attributes is called a table. Each table contains a record of a particular type




Relational model

PAGE NO. :
DATE : / /

## ER model -

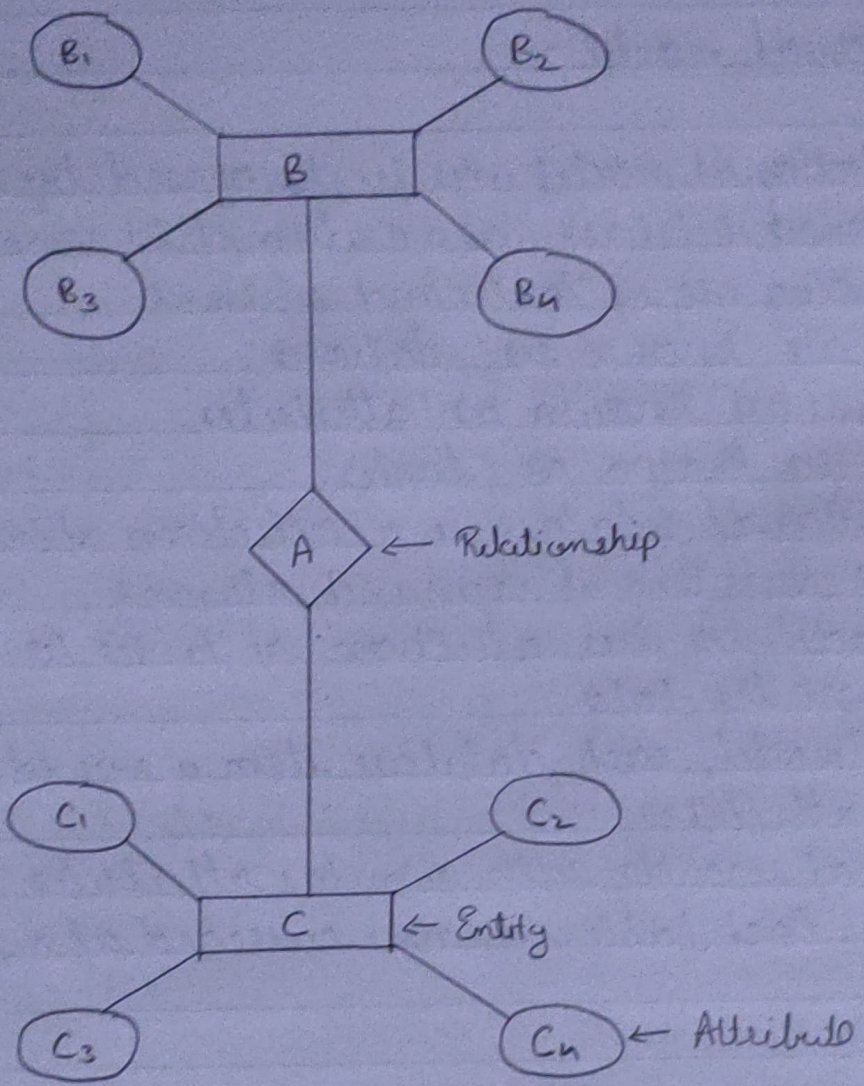
- ER model stands for entity-relationship model.
- It is a high level data model.
- This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database.
- It also develops a very simple and easy to design view of data.
- In E-R modeling, the database structure is portrayed as a diagram called an entity relationship diagram.
- The components of E-R diagram are:

① Entity
② Attribute
③ Relationship

- ① Entity:
An entity may be any object, class, person or place. It can be represented as rectangles in ER diagrams.

② Attribute:
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

② Relationships:
A relationship is used to describe the relation between entities. Diamonds or rhombus is used to represent relationship



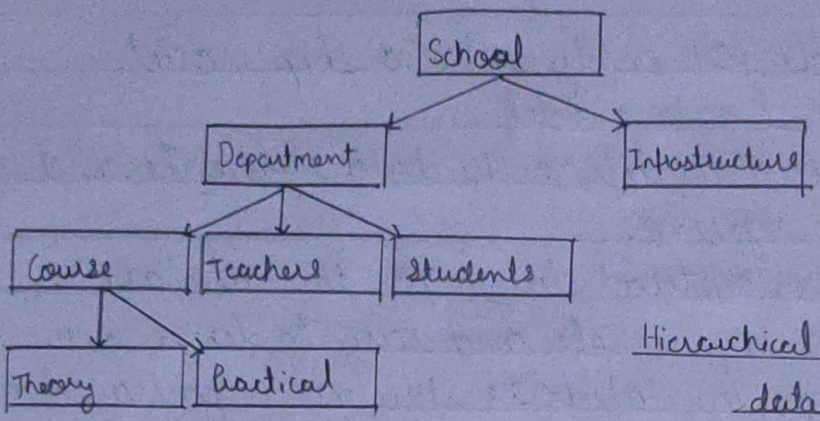

ER Model (Entity-Relationship-Model)

PAGE NO. :
DATE : / /

- Compare Network and Hierarchical Model.

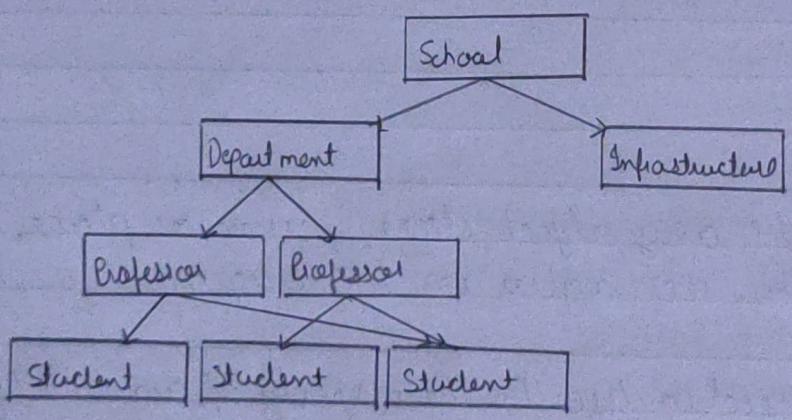
| Hierarchical Model | Network model |
|---|---|
| ① Relationship between record is of parent child type. | ① Relationship between records is expressed in the form of pointers or links. |
| ② It can have data inconsistency during updation and deletion of the data. | ② No data inconsistency. |
| ③ Transversing of data is complex. | ③ Data transversing is easy because node can be accessed from parent to child or child to parent. |
| ④ It does not support many to many relationship | ④ It supports many to many relationship. |
| ⑤ It creates tree like structure | ⑤ It creates graph like structure. |

Problem statement 1: Design school database by Hierarchical Model




Hierarchical model of school database

Problem statement 2: Design school database by Network model



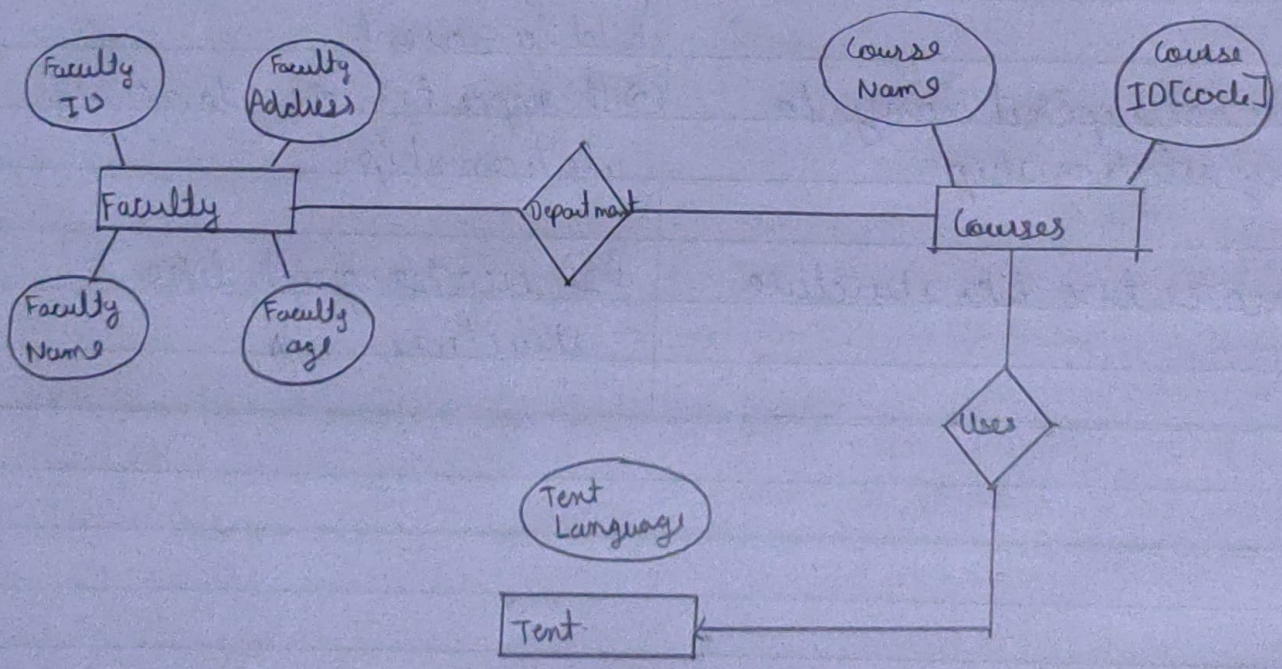

Network model of School Database

Problem Statement 3: Design School Database by Relation Model

Employees

| Roll No | First Name | Last Name | Department |
|---|---|---|---|
| 18621 | Naruto | Uzumaki | Leaf |
| 18622 | Gaara | Yuzawa | Sand |
| 18623 | Killer | Bee | Clouds |
| 18624 | Itachi | Uchiha | Leaf |
| 18625 | Zaluza | Momochi | Mist |

Problem Statement 4: Design School Database by relational Model



PAGE NO. :
DATE : / /

Conclusion:

Hence, I designed a database of school system using Network model, Hierarchical model, Relational model and ER model.

PAGE NO.:
DATE: / /

# Practical No. 02

**Aim:** Identify entities, Attributes, Tuple, Domains and Prime keys availble in above created model

**Theory:**

- What is an entity?

→ An entity can be a real world object, either animate or inanimate, that can easily be identifiable. For example, in a school database students, teachers, classes and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities.

- What is an attribute?

→ Entities are represented by means of their properties, called attributes. All attributes have value. For example an employee may have name, age, salary, etc. as the value, i.e. as their attribute.

There exists a domain or range of values that can be assigned to attributes. For example, a students name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

PAGE NO. :
DATE : / /

• What is a tuple?

→ A table has rows and columns, where rows represent records and columns represent attributes.

Tuple is a single row of table, which contains a single record for that relation. A finite set of tuples in the relational database system represents relational instance.

• What is a domain?

→ A table in DBMS is a set of rows and columns that contain its data. Columns in a table have a unique name, often referred as attributes in DBMS.

A domain is a unique set of values permitted for an attribute in a table. For example, a domain of month-of-a-year can accept January, February, ... December as possible values.

• What is a prime key?

→ A prime key is a column in a table whose values uniquely identify the rows in the table. The primary key is chosen from this list of candidates based on its preferred value to the business as an identifier.

If the primary key is a combination of more than one column then it is called as composite key.

PAGE NO. :
DATE : / /

Primary key  Domain

| Roll No | First Name | Last Name | Department |
|---|---|---|---|
| 18621 | Naruto | Uzumaki | Leaf |
| 18622 | Gaara | Yuzawa | Sand |
| 18623 | Killer | Bee | Clouds |
| 18624 | Itachi | Uchiha | Leaf |
| 18625 | Zabuza | Momochi | Mist |

Entity → (18622)

← Tuple or Row.

↑ Columns or Attributes

- The above diagram is the problem statement 3 i.e. Design school database by relational model.
- The roll no is the primary key because it is unique to every student and teacher.
- The first name, last name and department are domains because they can only be stored value of a specific type. For example, in the above mentioned table only alphabetical values can be stored.
- The data inside the table is called as entity.
- The rows of the above table can be also be reffered to as tuples.
- The columns of the table can also be reffered to as attributes.

PAGE NO. :
DATE : / /

* Conclusion :

Hence, by performing this practical I identified the entities, Attributes, Tuples, domains and prime keys in the above created ~~Typ~~ model.

PAGE NO. :
DATE : / /

# Practical No. 03

* Aim : Design a normalize database. Identify available dependencies in created database. Identify types of used normal form.

* Theory :

• What is normalisation of database?

→ Normalisation is a process of organising the data in a database to avoid data redundancy, insertion anomaly, update anomaly and deletion anomaly

Types of normalisation :

The most commonly used normal forms are :
First normal form (1NF)
Second normal form (2NF)
Third normal form (3NF)
Boyce & Codd normal form (BCNF)

• First normal form :

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

Example : Suppose a school wants to store the roll nos. Name and courses of student. It creates a table that look like this

PAGE NO.:
DATE: / /

| Roll no. | Name | Courses |
|---|---|---|
| 01 | Levi Ackerman | Python, SQL |
| 02 | Mikasa Ackerman | Javascript. |
| 03 | Eren Jaeger. | C#, C++ |

Two students (Eren and Levi) are having two courses so the school stored them in the same field as you could see in the table above.

Thus the above table is not in 1NF as the rule says "each attribute of the table must have atomic values". So to make the table compatible with 1NF, we should have the data as follows:

| Roll No. | Name | Courses. |
|---|---|---|
| 01 | Levi Ackerman | Python |
| 01 | Levi Ackerman | SQL |
| 02 | Mikasa Ackerman | Javascript |
| 03 | Eren Jaeger | C# |
| 03 | Eren Jaeger | C++ |
| | | |

- Second Normal Form:

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

PAGE NO. :
DATE : / /

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

Example:

| profesor_id | Courses | Name |
|---|---|---|
| 161 | HTML CSS | Alex |
| 162 | Javascript | Alex |
| 163 | Python | Steve |
| 163 | C# | Harvey |
| 163 | C++ | Harvey |

Candidate key: { profesor_id, classes }

Non-prime attribute: { Name }

The above table is in 1NF because it meets the conditions for 1NF. But, it is not in 2NF because non-prime attribute name is dependent on profesor_id alone. which is proper subset of candidate key.

This validates the rule of 2NF which states - "No non-prime attribute is dependant on the proper subset of any candidate key on the table." To make the tables compiles with 2NF we can break it into two tables like this,

profesor_details table:

| profesor_id | Name |
|---|---|
| 161 | Alex |
| 162 | Steve |
| 163 | Harvey. |

PAGE NO. :
DATE : / /

professor_courses
~~Teacher_classes~~ table :

| professor_id | courses |
|---|---|
| 161 | HTML CSS |
| 161 | Javascript |
| 162 | Python |
| 163 | C# |
| 163 | C++ |

Now, the tables comply with the second normal form (2NF).

- Third Normal Form (3NF) :

A table design is said to be ~~not~~ 3NF if both the conditions are met :

- The table must be in 2NF
- Transitive functional dependancy of non prime attribute on any super key should be removed.

Example :

| roll no. | Name | Zip code | State | city |
|---|---|---|---|---|
| 196 | Stephen | 440016 | MH | Nagpur-S |
| 197 | Hossain | 440042 | MH | Nagpur-W |
| 198 | Dan | 440036 | MH | Hingna |
| 199 | George | 440032 | MH | MIDC, NGP |
| 200 | Caleb | 440010 | MH | Central, NGP |

PAGE NO. :
DATE : / /

Candidate-key : roll no.

Here, the non-prime attributes, state and city are transitively dependant upon super key (roll no.). It violates the rule of third normal form. That's why, we need to move the state and city to new table with zipcode as a primary key.

Student's table :

| Roll No. | Name | Zipcode |
|---|---|---|
| 196 | Stephen | 440016 |
| 197 | Howrah | 440042 |
| 198 | Dan | 440036 |
| 199 | George | 440032 |
| 200 | Caleb | 440010 |

Residence's table :

| Zipcode | State | City |
|---|---|---|
| 440016 | MH | Nagpur-S |
| 440042 | MH | Nagpur-W |
| 440036 | MH | Hingna |
| 440032 | MH | MIDC, NGP |
| 440010 | MH | Central NGP. |

- Boyce Codd Normal Form (BCNF) :
  - BCNF is an advance version of 3NF. It is stricter than 3NF.
  - A table is in BCNF if every functional dependency $X \rightarrow Y$, X is the superkey of the table.

PAGE NO. :
DATE : / /

Example: Let's assume there's a school and the teacher teach in different classes.

Teacher table:

| ID | Name | Class | Subject | Students |
|---|---|---|---|---|
| 181 | Sakuta | 1st yr A | C | 30 |
| 181 | Sakuta | 1st yr B | C | 32 |
| 186 | Maisan | 2nd yr A | RDBMS | 29 |
| 186 | Maisan | 2nd yr B | RDBMS | 31 |

The above table is not in BCNF because neither ID & nor class alone are keys..

To convert the above table into BCNF, we decompose it into three tables as follows:

Name Table:

| ID | Name |
|---|---|
| 181 | Sakuta |
| 186 | Maisan |

Candidate Key : ID
Functional dependency: ID → Name.

Class Table:

| Class | Subject | Student |
|---|---|---|
| 1st yr A | C | 30 |
| 1st yr B | C | 32 |
| 2nd yr A | RDBMS | 29 |
| 2nd yr B | RDBMS | 31 |

Candidate Key : Class
Functional dependency:
class → subject, students.

PAGE NO.:
DATE: / /

Name, class relation table:

| ID | Class |
|---|---|
| 181 | 1st yr A |
| 181 | 1st yr B |
| 186 | 2nd yr A |
| 186 | 2nd yr B |

Candidate-Key : ID, Class.

* Conclusion:

Hence, I designed a normalize database, identified available dependencies in created database and identified the types used normal form.

PAGE NO.:
DATE: / /

# Practical No. 04.

* **Aim:** Create and execute DDL commands in SQL and apply various integrity constrains on above created table.

* **Theory:**

* What is SQL?

→ • Structured Query Language (SQL) is the standard and most widely used programming language for relational databases.
• It is used to manage and organise data in all sorts of systems in which various data relationship exists.
• SQL is a value programming language with strong career prospects.

* Components of SQL:

A DBMS has appropriate language and interfaces to express database queries and updates. Database language can be used to read, store and update data in the database.

* Types of database languages:

① Data definition language (DDL):

It is used to define database structure or pattern. The various commands in this language are: Create, alter, drop, truncate, rename, comment.

PAGE NO. :
DATE : / /

② Data manipulation language (DML):

It is used for accessing and manipulating data in a database. The various command in this language are SELECT, INSERT, UPDATE, DELETE, MERGE, CALL, Explain, Lock Table.

③ Data Control Language (DCL):

It is used to retrieve the stored or saved data. The commands used in this language are GRANT, REVOKE.

④ Transaction Control Language (TCL):

TCL is used to run the changes made by the DML. The commands used in this language are COMMIT, GRANT.

★ Data Definition Language commands:

i) Alter: It is used to alter/add objects in a database.

Syntax: ALTER TABLE table_name.
ADD column_name datatype;

Example:

Name table:

| ID | Name. |
|---|---|
| 23 | Sakura |
| 24 | Hinata |
| 25 | Tenten. |

PAGE NO. :
DATE : / /

ALTER TABLE name
ADD age INTEGER;

| ID | Name | Aage |
|---|---|---|
| 23 | Sakura | 29 |
| 24 | Hinata | 21 |
| 25 | Tenten | 24. |

2) Create: It is used to create objects in the database.
SYNTAX: CREATE TABLE table_name (
column1 datatype
column2 datatype.
....);
Example: CREATE TABLE Orders (
order_id INTEGER PRIMARY KEY,
order VARCHAR(20),
Name VARCHAR(20),
);

| order_id | order | Name. |
|---|---|---|

3) Drop: It is used to delete object from the database.
SYNTAX: DROP TABLE table_name.
Example:
Student Table:

| ID | Name | AGE |
|---|---|---|
| 23 | Sakura | 29 |
| 24 | Hinata | 21. |

PAGE NO. :
DATE : / /

DROP TABLE Student;
Thus the table will be deleted.

4) TRUNCATE :
It is used to delete or remove all data from a table.
Syntax : TRUNCATE TABLE table-name;

Example :
Student table -

| ID | Name | Age. |
|---|---|---|
| 23 | Sakura | 23 |
| 24 | Hinata | 26. |

TRUNCATE TABLE Student;

| ID | Name | Age. |
|---|---|---|

5) Rename :
It is used to rename an object.
SYNTAX : RENAM COLUML old-name to new-name

Example :

| ID | Name | Age |
|---|---|---|
| 23 | Sakura | 23 |
| 24 | Hinata | 26. |

PAGE NO.:
DATE: / /

RENAME COLUMN ID to Student_id.

| Student_id | Name | Age |
|---|---|---|
| 23 | Sakura | 23 |
| 24 | Hinata | 26. |

6) COMMENT:

It is used to comment on the data directory.

Syntax: COMMENT ON COLUMN column_name IS 'comment';

Example:

COMMENT ON COLUMN students.name IS 'studentname here';

* Conclusion:

Hence, I created DDL commands in SQL and applied various integrity constrains on above created table.

PAGE NO. :
DATE : / /

# Practical No. 05

* **Aim:** Create and execute DML command in SQL.

* **Theory:**

DML stands for Data Manipulation Language.

The commands used in DML language are:

1. SELECT
2. INSERT
3. UPDATE
4. DELETE
5. MERGE
6. CALL
7. EXPLAIN PLAN
8. LOCKTABLE.

## 1. SELECT:

It is used to retrive data from a database.

Syntax: SELECT column_1, column_2, ....
FROM table_name;

Example:

SELECT * ~~from~~ FROM students;

| id | Name | age. |
|---|---|---|
| 69 | Killua | 18 |
| 68 | Jon | 12 |
| 67 | Haruta | 23. |

## 2. INSERT:

It is used to insert data in the table.

SYNTAX: INSERT INTO table_name (column 1, column 2, ...)
VALUES (value 1, value 2, value 3, ...).

PAGE NO.:
DATE: / /

Example
Student's table:

| ID | Name | age. |
|---|---|---|

INSERT INTO Student VALUES (196, "Kenma", 16).

| ID | Name | age |
|---|---|---|
| 196 | Kenma | 16. |

2) UPDATE:
It is used to update data in existing table.
Syntax:
UPDATE table_name.
SET column 1 = value, column 2 = value, ...
WHERE condition.

Example:
Player's ~~Ator~~ table:

| ID | Name | age. |
|---|---|---|
| 126 | Kuru | 19. |

UPDATE Player
SET Name = "Kuro"
WHERE ID = 126;

| ID | Name | age |
|---|---|---|
| 126. | Kuro | 19 |

PAGE NO. :
DATE : / /

WHEN NOT MATCHED BY TARGET
THEN merge_not_matched
WHEN NOT MATCHED BY SOURCE
THEN merge_matched;

Example:

PRODUCT_LIST

| ID | Name | Price |
|---|---|---|
| 161 | 18GBRAM | 6,000 |
| 162 | 1TBHDD | 1,800 |
| 163 | SMPS. | 3,500 |

Product List (Target)

UPDATE_LIST

| ID | Name | PRICE |
|---|---|---|
| 161 | 16 GB RAM | 6,200 |
| 162 | 1TB HDD | 1,800 |
| 163 | SMPS. | 3,600 |

Updated List (Source)

```
MERGE PRODUCT_LIST AS TARGET
USING UPDATE_LIST AS SOURCE

ON (TARGED.ID = SOURCE.ID)
WHEN MATCHED
    TARGET.name <> SOURCE.name.
    TARGET.price <> SOURCE.price.
THEN UPDATE.
    SET TARGET.name = SOURCE.name.
        TARGET.name = SOURCE.price.
WHEN NOT MATCHED BY TARGET.
    THEN INSERT (ID, name, price).
        VALUE (SOURCE.ID, SOURCE.name, SOURCE.price)
WHEN NO MATCHED BY SOURCE
THEN DELETE
```

PAGE NO.:
DATE: / /

4) DELETE:

It is used to delete all records from a table.

Syntax:

DELETE FROM TABLE WHERE Condition;

Example:

Wrestler's table.

| ID | Name | Weight |
|---|---|---|
| 261 | Russev | 195 lbs |
| 169 | Rey Mysterio | 169 lbs |

DELETE FROM Wrestler WHERE ID=261;

| ID | Name | Weight |
|---|---|---|
| 169 | Rey Mysterio | 169 lbs. |

5) MERGE:

It performs upsert operation; i.e. insert or update operation.

Syntax:

```
MERGE target_table AS TARGET
USING source_table AS SOURCE
ON condition
WHEN MATCHED
 THEN merge_matched.
```

| ID | name | Price. |
|---|---|---|
| 161 | 16GB RAM | 6,200 |
| 162 | 1TB HDD | 1,800 |
| 163 | SMPS | 3,600 |

6) Call:

It is used to call a structured query language or a Java subprogram.

Syntax: [variable] CALL procedure-name [expression])

Example:

```
EXCEL SQL
CALL GETEMPSVR (:V1, V2)
END-EXEC;
```

7) EXPLAIN-PLAN:

It has the parameter of explaining data.

Syntax:

```
SELECT PLAN-TABLE-OUTPUT
FROM TABLE (DBMS-XPLAN.DISPLAY (NULL,
                                'statement-id'.)
```

Example:

```
EXPLAIN PLAN
    SELECT statement-id = 'ex-plan1' FOR
    SELECT phone-number FROM employees
    WHERE phonenumber LIKE '650 %';
```

PAGE NO. :
DATE : / /

8) LOCK TABLE :

It controls concurrency.

Syntax :

LOCK TABLE table-name.
IN { SHARE | EXCLUSIVE } MODE

Example :

LOCK TABLE students
IN share MODE.
SHARE.

* Conclusion :

Hence, I created various locks and executed them using the DML commands such as SELECT, INSERT, UPDATE, DELETE, MERGE, CALL, EXPLAIN PLAN, LOCK table in SQL.

PAGE NO.:
DATE: / /

# Practical No. 06

* **Aim:** Write queries using various operators arithmetic, set operator, relational operator and comparison operators to retrieve data

* **Theory:**

What are operators?

→ An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operations. These operators are used to specify conditions in an SQL statement and to serve as statement.

1) Arithmetic operators:

| Operator | Their function |
|---|---|
| + | to add |
| - | to subtract |
| * | to multiply |
| \ | to divide |
| % | Returns remainder. |

2) Set operator:

Set operators are used to join the results of two (or more) SELECT statements. The SET operators available in Oracle are:

PAGE NO.:
DATE: / /

i) UNION
ii) UNION ALL
iii) INTERSECT
iv) MINUS

3) Relational operators:

| Operators | Description |
|---|---|
| = | Check if the values are equal, if yes then returns true. |
| != | Check if the values are not equal, if yes then return true. |
| <> | checks if the values are not equal, if yes then returns true |
| > | Checks if the value is greater than or not |
| < | Checks if the value is smaller than or not |
| >= | Checks if the value is greater than and equal to or not. |
| <= | Checks if the value is smaller than and equal to or not. |
| !< | Checks if the value is not less than |
| !> | Checks if the value is not greater than. |

① Arithmetic operators:

(i) '+' operator

```
SELECT 20+10;
```

| 20+10 |
|---|
| 30 |

PAGE NO. :
DATE : / /

(ii) '-' operators:

SELECT 20-10;

| 20-10 |
|---|
| 10 |

(iii) '/' operator.

SELECT 20/10;

| 20/10 |
|---|
| 2 |

(iv) '*' operator.

SELECT 20*10;

| 20*10 |
|---|
| 200 |

(v) '%' operator.

SELECT 20%2;

| 20%2 |
|---|
| 0 |

PAGE NO.:
DATE: / /

① Relational operators.

Code:

```
CREATE TABLE info (id INTEGER PRIMARY KEY
                   name TEXT,
                   marks INTEGER
                   );

INSERT INTO info VALUES (1, "Deku", 49)
INSERT INTO info VALUES (2, "Todoroki", 50)
INSERT INTO info VALUES (3, "Bakugou", 46)
INSERT INTO info VALUES (4, "Kamenari", 40)
INSERT INTO info VALUES (5, "Lida", 42)
INSERT INTO info VALUES (6, "Mineta", 32)
INSERT INTO info VALUES (7, "Tokoyami", 38)
INSERT INTO info VALUES (8, "Uraraka", 42)
INSERT INTO info VALUES (9, "Kirishima", 35)
INSERT INTO info VALUES (10, "Izuku", 33)

SELECT * FROM info WHERE marks = 42;      Fig 6.1
SELECT * FROM info WHERE marks != 42;     Fig 6.2
SELECT * FROM info WHERE marks > 45;      Fig 6.3
SELECT * FROM info WHERE marks < 35;      Fig 6.4
SELECT * FROM info WHERE marks >= 40;     Fig 6.5
SELECT * FROM info WHERE marks <= 40;     Fig 6.6
```

| id | Name | Marks |
|---|---|---|
| 5 | Lida | 42 |
| 8 | Uraraka | 42 |

Fig. 6.1

| id | Name | Marks |
|---|---|---|
| 1 | Deku | 49 |
| 2 | Todoroki | 50 |
| 3 | Bakugou | 46 |
| 4 | Kamenari | 40 |
| 6 | Mineta | 32 |
| 7 | Tokoyami | 38 |
| 9 | Kirishima | 35 |
| 10 | Tsuyu | 33 |

Fig. 6.2

| id | Name | Marks |
|---|---|---|
| 1 | Deku | 49 |
| 2 | Todoroki | 50 |
| 3 | Bakugou | 46 |

Fig. 6.3.

| id | Name | Marks |
|---|---|---|
| 6 | Mineta | 32 |
| 10 | Tsuyu | 33 |

Fig. 6.4

| id | Name | Marks |
|---|---|---|
| 1 | Deku | 49 |
| 2 | Todoroki | 50 |
| 3 | Bakugou | 46 |
| 4 | Kamenari | 40 |
| 5 | Lida | 42 |
| 8 | Uraraka | 42 |

Fig. 6.5

| id | Name | Marks |
|---|---|---|
| 4 | Kumenai | 40 |
| 6 | Mineta | 32 |
| 7 | Takayami | 38 |
| 9 | Kirishima | 35 |
| 10 | Tsuyu | 33 |

Fig. 6.6.

PAGE NO. :
DATE : / /

Conclusion:

Hence I wrote queries using various operators such as arithmetic, relational, conditional, set operators to retrieve the data.

PAGE NO. :
DATE: / /

# Practical No. 07

* **Aim:** Use different aggregate functions, string functions, Date-time functions, Data conversion functions such as to To_char(), To_number(), and To_date(). And also display special date formats using To_char() function.

* **Theory:**

What are aggregate?

→ In database management an aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

Examples of aggregate functions are:

(i) Count()
(ii) Sum()
(iii) Avg()
(iv) Min()
(v) Max()

**(i) Count():**

Syntax:

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

PAGE NO. :
DATE : / /

Code:

```
CREATE TABLE classes (
        class id INTEGER PRIMARY KEY
        class student_count INTEGER
        )

INSERT INTO classes VALUES (1, 58)
INSERT INTO classes VALUES (2, 60)
INSERT INTO classes VALUES (3, 62)
INSERT INTO classes VALUES (4, 59)
INSERT INTO classes VALUES (5, 60)
INSERT INTO classes VALUES (6, 61)

SELECT *FROM classes
SELECT COUNT(student_count) FROM classes;
SELECT SUM(student_count) FROM classes;
SELECT AVG(student_count) FROM classes;
SELECT MIN(student_count) FROM classes;
SELECT MAX(student_count) FROM classes;
```

| class | student_count |
|---|---|
| 1 | 58 |
| 2 | 60 |
| 3 | 62 |
| 4 | 59 |
| 5 | 60 |
| 6 | 61 |

| Count |
|---|
| 6 |

| Min |
|---|
| 58 |

| Sum |
|---|
| 360 |

| Max |
|---|
| 62 |

| Avg |
|---|
| 60 |

PAGE NO. :
DATE : / /

## String functions:

SQL string function is used primarily for string manipulation.

Code:

```
SELECT CHAR(112) AS ansiToChar;
```

| ansiToChar. |
| --- |
| p |

```
SELECT LOWER("HELLO SQL!") as lowercase;
```

| lowercase |
| --- |
| hello sql |

```
SELECT UPPERCASE("hello sql!") as uppercase;
```

| uppercase |
| --- |
| HELLO SQL! |

## Date-time function:

Date time functions help in querying date and time in the database. The following are some examples of SQL date time function:

PAGE NO. :
DATE : / /

i) NOW()
ii) CURDATE()
iii) CURTIME()

Example

SELECT NOW(), CURDATE(), CURTIME();

| NOW() | CURDATE() | CURTIME() |
|---|---|---|
| 2021-1-25 09:43:32 | 2021-1-25 | 09:43:32 |

## Data Conversion functions:

### i) To Char:

In oracle, TO_CHAR function converts a date-time value to a string using a specified format.

Code:

```
SELECT TO_CHAR(SYSDATE, 'YYYY-MM-DD') FROM Dual;
```

Output:

| 2011-11-26 |
|---|

### ii) To Number:

The oracle/PLSQL TO_NUMBER function converts a string to a number.

PAGE NO.:
DATE: / /

Syntax:
TO_NUMBER(string1, [, format_mask] [, nls-language]);

Code:
TO_NUMBER('1210.73');

Result:
1210.73

iii) To Date:
In Oracle, TO_DATE function converts a string value to DATE data type value using the specified format.

Syntax:
TO_DATE (string, format);

Code:
TO_DATE;
Syntax:
TO_DATE (string, format);
Code:
SELECT TO_DATE('2016-05-19', 'yyyy-MM-DD') FROM dual.

PAGE NO. :
DATE : / /

* Conclusion:

Hence, I used different aggregate functions, string functions, Date-time functions, data conversion functions such as To_char(), To_number() and To_date(). Also display special date formats using To_char() function.