Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology

(Deemed to be University Estd. u/s 3 of UGC Act, 1956) School of Computing

B.Tech.   – INFORMATION TECHNOLOGY

Academic Year: 2025–   2026

SUMMER SEMESTER - SS2526

Course Code : 10211IT201

Course Name : Database System Concepts

Slot No   : S12L5

DBMS TASK - 2 REPORT

Submitted by:

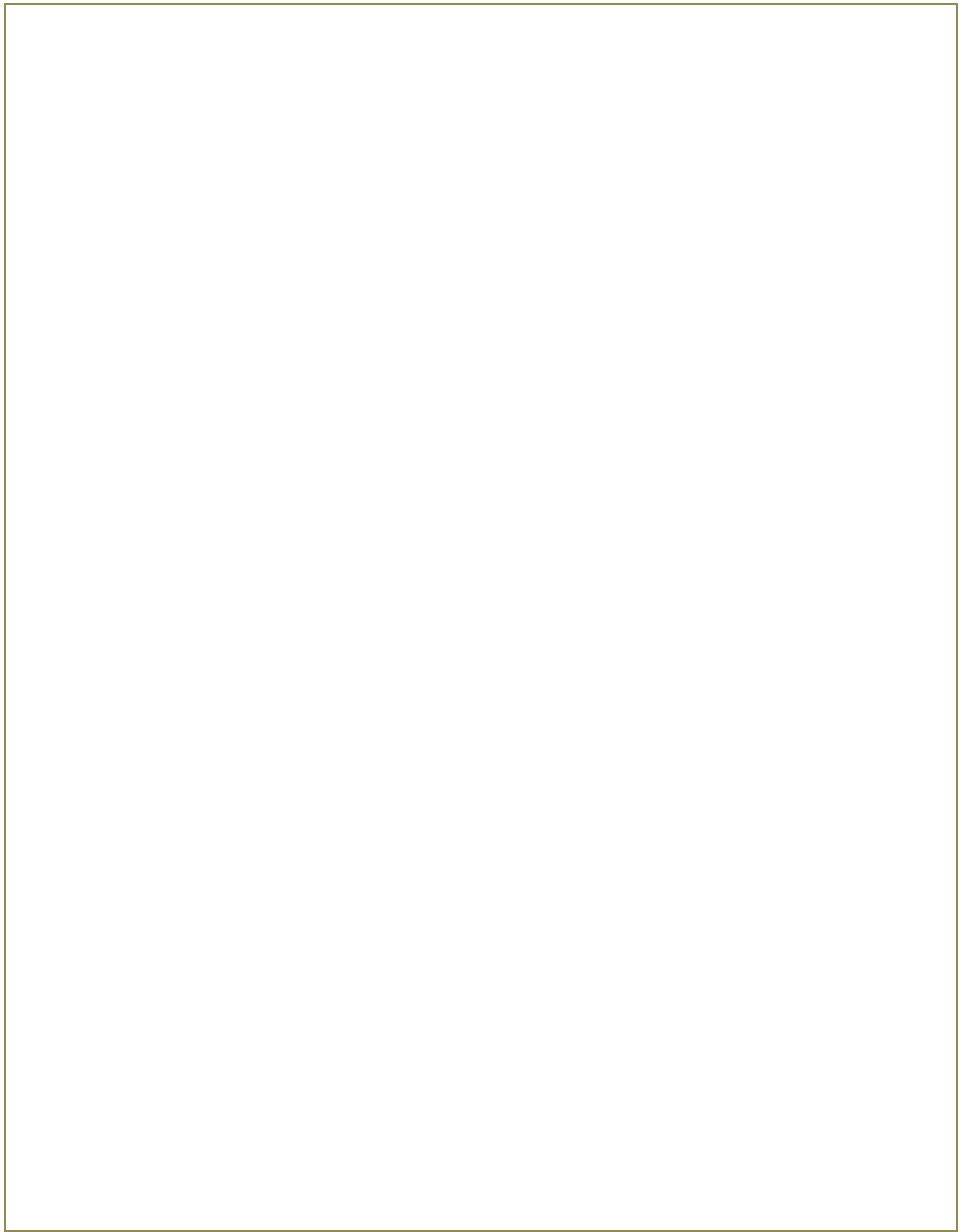| VTUNO	REGISTER NUMBER	STUDENT NAME
VTU29010	24UEIT0040	P.manikanta

ABSTRACT

The Student Book Management System aims to efficiently organize and manage the records of students and the books they borrow from an institution's library. The system is modeled using a Hierarchical and Network Database Model, which provides a structured and flexible data organization suitable for handling one-to-many and many-tomany relationships between entities such as Students, Books, Authors, and Borrow Records.

In the Hierarchical Model, data is represented in a tree-like structure where each student acts as a parent node with dependent child nodes representing the books issued. This structure simplifies record retrieval and ensures quick access to student-specific data.

The Network Model, on the other hand, allows more complex relationships—such as a book being issued to multiple students over time or authored by multiple writers— through the use of pointers and sets. This model enhances data connectivity and reduces redundancy compared to traditional file systems.

By implementing these models, the system achieves better data integrity, faster access paths, and improved management of student-book transactions. The project demonstrates how hierarchical and network models can be effectively applied to real-world academic systems where data relationships are both structured and interconnected.

Creating Hierarchical / Network Model for Student Book Management System

Creating Hierarchical /Network model of the database by enhancing the sound abstract data by performing following tasks using forms of inheritance:

    a.   Identify the specificity of each relationship, find and form surplus relations.

    b.   Check is-a hierarchy/ has-a hierarchy and performs generalization and/orspecialization relationship.

    c.   Find the domain of the attribute and perform a check constraint to theapplicable.

    d.   Rename the relations.

    e.   Perform SQL Relations using DDL, DCL commands.

e. Perform SQL Relations using DDL, DCL commands.

DATA DEFINITION COMMANDS

CREATION OF TABLE

Example

CREATE TABLE Customer

  (CustomerID INT PRIMARY KEY,

  FirstName VARCHAR(50) NOT NULL,

  LastName VARCHAR(50) NOT NULL,

  DateOfBirth DATE,

  Address VARCHAR(100),

  PhoneNumber VARCHAR(15)

);

```
SQL> desc customer;

 Name                          Null?    Type
 -------------------------------------- ---------- --------------------------------
 CUSTOMERID                    NOT NULL NUMBER(38)
 FIRSTNAME                     NOT NULL VARCHAR2(50)
 LASTNAME                      NOT NULL VARCHAR2(50)
 DATEOFBIRTH                            DATE
 ADDRESS                                VARCHAR2(100)
 PHONENUMBER                            VARCHAR2(15)
 EMAIL                                  VARCHAR2(50)
```

```
CREATE TABLE Account

  ( AccountID INT PRIMARY KEY,

  CustomerID INT,

  AccountNumber VARCHAR(20) NOT NULL,

  AccountType VARCHAR(10) NOT NULL,

  Balance DECIMAL(10, 2) CHECK (Balance >= 0),

  InterestRate DECIMAL(3, 2) CHECK (InterestRate >= 0.0 AND InterestRate <= 5.0),
```

CONSTRAINT fk_customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)

);

SQL> desc account;

```
 Name                              Null?    Type
 --------------------------------- -------- --------------------------------

  ACCOUNTID                        NOT NULL NUMBER(38)

  CUSTOMERID                       NOT NULL NUMBER(38)

  ACCOUNTNUMBER                    NOT NULL VARCHAR2(25)

  ACCOUNTTYPE                      NOT NULL VARCHAR2(10)

  BALANCE                                   NUMBER(10,2)

  INTERESTRATE                              NUMBER(3,2)
```


```sql
CREATE TABLE Transaction

  ( TransactionID INT PRIMARY KEY,

  AccountID INT,

  TransactionDate DATE NOT NULL,

  TransactionType VARCHAR(10) NOT NULL CHECK (TransactionType IN ('Deposit', 'Withdrawal')),

  Amount DECIMAL(10, 2) NOT NULL CHECK (Amount > 0),
```

CONSTRAINT fk_account FOREIGN KEY (AccountID) REFERENCES Account(AccountID)

);

SQL> desc transactiondetails;

Name	Null?	Type
TRANSACTIONID	NOT NULL	NUMBER(38)
ACCOUNTID	NOT NULL	NUMBER(38)
TRANSACTIONDATE	NOT NULL	DATE
TRANSACTIONTYPE	NOT NULL	VARCHAR2(10)
AMOUNT	NOT NULL	NUMBER(10,2)

Alter Tables

Add a Column to Customer Table

ALTER TABLE Customer ADD Email VARCHAR(50);

Modify Column in Account Table

ALTER TABLE Account MODIFY COLUMN AccountNumber VARCHAR(25);

Add a Check Constraint to Transaction Table

Drop Tables

Drop Transaction Table

DROP TABLE Transaction;

Drop Customer Table

DROP TABLE Customer;

DATA CONTROL LANGUAGE COMMANDS

Using Savepoint and Rollback

Following is the table class,

id	name
1	Abhi
2	Adam
4	Alex

Example

INSERT INTO class VALUES(5, 'Rahul');

COMMIT;

UPDATE class SET name = 'Abhijit' WHERE id = '5';

SAVEPOINT A;

INSERT INTO class VALUES(6, 'Chris');

SAVEPOINT B;

INSERT INTO class VALUES(7, 'Bravo');

SAVEPOINT C;

SELECT * FROM class;

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit
6	Chris
7	Bravo

The ROLLBACK command to roll back the state of data to the savepoint B.

ROLLBACK TO B;

SELECT * FROM class;

id	name
1	Abhi
2	Adam

4	Alex
5	Abhijit
6	Chris

Again using the ROLLBACK command to roll back the state of data to the savepoint A

ROLLBACK TO A;

SELECT * FROM class;

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit

c. Find the domain of the attribute and perform a check constraint to the applicable.

A check constraint in a database ensures that all values in a column satisfy a specific condition.

Sure! Let's work with an Account relation and provide examples of check constraints that ensure data integrity.

Account Table Structure

Assume we have an Account table with the following columns:

- AccountID (Primary Key)
- AccountNumber
- AccountType (e.g., 'Savings', 'Checking')
- Balance
- InterestRate

Example Check Constraints

Here are some examples of check constraints for the Account table:

1. Balance Constraint: Ensure that the account balance is non-negative.
2. Interest Rate Constraint: Ensure that the interest rate is between 0.0 and 5.0 percent.
3. Account Type Constraint: Ensure that the account type is either 'Savings' or 'Checking'.

SQL Statements

Create Table with Check Constraints:

```
CREATE TABLE Account
    ( AccountID INT PRIMARY KEY,
    AccountNumber VARCHAR(20) NOT NULL,
    AccountType VARCHAR(10) NOT NULL,
    Balance DECIMAL(10, 2),
    InterestRate DECIMAL(3, 2),
    CONSTRAINT chk_balance CHECK (Balance >= 0),
    CONSTRAINT chk_interest_rate CHECK (InterestRate >= 0.0 AND InterestRate <= 5.0),
    CONSTRAINT chk_account_type CHECK (AccountType IN ('Savings', 'Checking'))
);
```

Add Check Constraint to an Existing Table:

If the Account table already exists, you can add check constraints using the ALTER TABLE

SQL statement:

ALTER TABLE Account ADD CONSTRAINT chk_balance CHECK (Balance >= 0);

ALTER TABLE Account ADD CONSTRAINT chk_interest_rate CHECK (InterestRate >= 0.0 AND InterestRate <= 5.0);

ALTER TABLE Account ADD CONSTRAINT chk_account_type CHECK (AccountType IN ('Savings', 'Checking'));

d. Rename the relations.

In SQL, you can rename a table or a column within a table. Let's work with a Bank relation and provide examples for renaming both the table and columns.

Renaming a Table

Suppose you have a table named Bank and you want to rename it to FinancialInstitution.

SQL Statement to Rename a Table:

ALTER TABLE Bank RENAME TO FinancialInstitution;

Renaming a Column

Suppose you have a column AccountNumber in the Bank table and you want to rename it to AcctNum.

SQL Statement to Rename a Column:

ALTER TABLE Bank CHANGE COLUMN AccountNumber AcctNum VARCHAR(20);

Result:- The queries have been executed successfully.