DBMS Lab Manual

Sl.No	Experiments
1.	Create an ER diagrams **Company Database system** and **Banking Database System using** drawio tool.
2.	i. Consider the **Company database** and create the below tables by properly specifying the primary keys and the foreign keys. **Employee** (Fname: varchar, Minit: Char, Lname: varchar, ssn:int, Bdate: Date, Address: varchar, Sex: char, salary: decimal,Super_ssn:int, DNO:int) **Department** (Dname: varchar, Dnumber: int, mgr_ssn: int, mgr_start_date: date) **Dept_location** (Dnumber: int, Dlocation: varchar) **Project** (pname: varchar, pnumber: int, plocation: varchar, dnum:int) **Works_on** (Essn: int, pno:int, hours: decimal) **Dependent** (Essn: char, dependent_name: varchar, sex: char, Bdate: date, relationship: varchar) ii. Insert at least five tuples in each relation.
3.	i. Retrieve the name and address of all employees who work for the ‘Research’ department. ii. 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. iii. For each employee, retrieve the employee’s first and last name and the first and last name of his or her immediate supervisor. iv. 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. v. Retrieve all employees whose address is in Houston, Texas. vi. Retrieve all employees in department 5 whose salary is between $30,000 and $40,000. **Execute above quires for the Company database defined in Unit-II.**
4.	i. Retrieve the names of all employees who do not have supervisors. ii. Retrieve the name of each employee who has a dependent with the same first name and is the same gender as the employee iii. Retrieve the names of employees who have no dependents. iv. List the names of managers who have at least one dependent. v. Retrieve the Social Security numbers of all employees who work on project numbers 1, 2, or 3. vi. 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. vii. For each department, retrieve the department number, the number of employees in the department, and their average salary. **Execute above quires for the Company database defined in Unit-II.**

5.	Consider the following database for a Banking enterprise: **BRANCH** (branch-name: string,branch-city: string,assets: real) **ACCOUNT** (accno:int,branch-name: string,balance: real) **DEPOSITOR** (customer-name: string,accno:int) **CUSTOMER** (customer-name: string,customer-street: string,city:string) **LOAN** (loan-number:int,branch-name: string,loan-number-int) **BORROWER** (customer-name: string,customer-street: string,city: string) 1) Create the above tables by properly specifying the primary and foreign keys 2) Enter 5 tuples for each relation 3) Find all the customers who have atleast two accounts at the main branch 4) Find all the customers who have an account at all the branches located in a specified city 5) Demonstrate how you delete all account tuples at every branch located in a specified city

Basic concepts

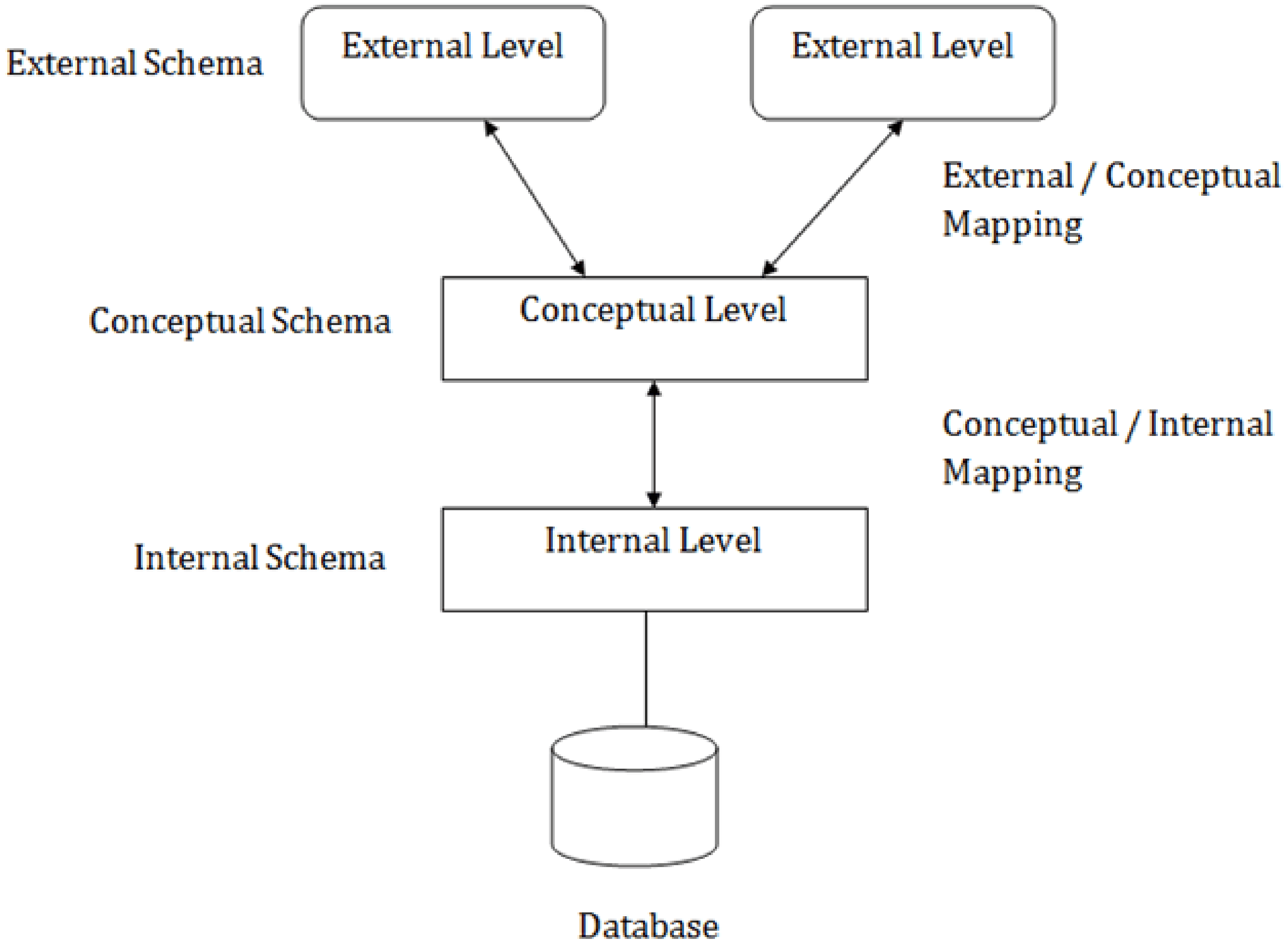
Schema

A database schema is considered the "blueprint" of a database which describes how the data may relate to other tables or other data models. However, the schema does not actually contain data. A sample of data from a database at a single moment in time is known as a database instance.

Types of database schemas

While the term schema is broadly used, it is commonly referring to three different schema types—a conceptual database schema, a logical database schema, and a physical database schema.

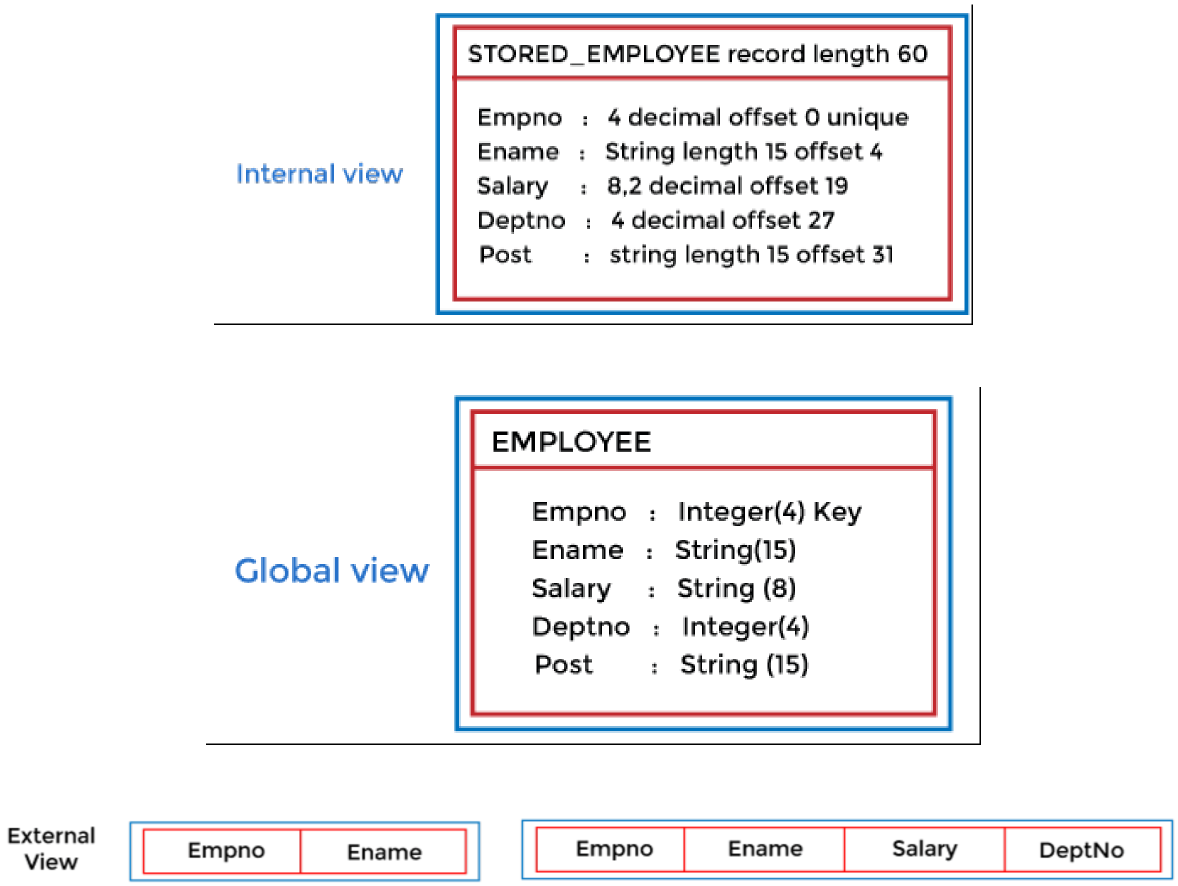
- **Conceptual schemas** offer a big-picture view of what the system will contain, how it will be organized, and which business rules are involved. Conceptual models are usually created as part of the process of gathering initial project requirements.
- **Logical database schemas** are less abstract, compared to conceptual schemas. They clearly define schema objects with information, such as table names, field names, entity relationships, and integrity constraints—i.e. any rules that govern the database. However, they do not typically include any technical requirements.
- **Physical database schemas** provide the technical information that the logical database schema type lacks in addition to the contextual information, such as table names, field names, entity relationships, et cetera. That is, it also includes the syntax that will be used to create these data structures within disk storage.
- **Three schema architecture:**



Objectives of three schema architecture:

- Different users need different views of the same data.
- The approach in which a particular user needs to see the data may change over time.
- The users of the database should not worry about the physical implementation and internal workings of the database such as data compression and encryption techniques, hashing, optimization of the internal structures etc.
- All users should be able to access the same data according to their requirements.
- DBA should be able to change the conceptual structure of the database without affecting the user's

- Internal structure of the database should be unaffected by changes to physical aspects of the storage.



Internal view

STORED_EMPLOYEE record length 60

Empno : 4 decimal offset 0 unique
Ename : String length 15 offset 4
Salary : 8,2 decimal offset 19
Deptno : 4 decimal offset 27
Post : string length 15 offset 31

Global view

EMPLOYEE

Empno : Integer(4) Key
Ename : String(15)
Salary : String (8)
Deptno : Integer(4)
Post : String (15)

External View

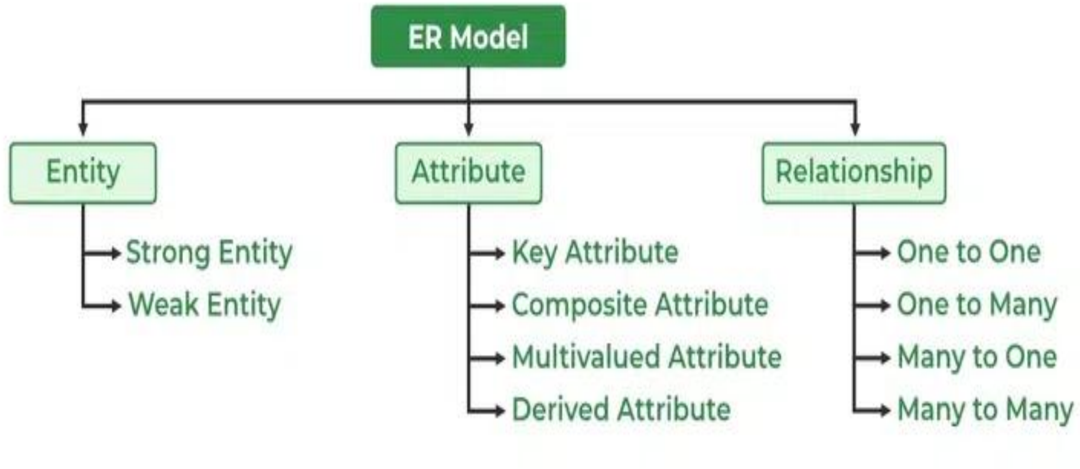
Empno	Ename

Empno	Ename	Salary	DeptNo

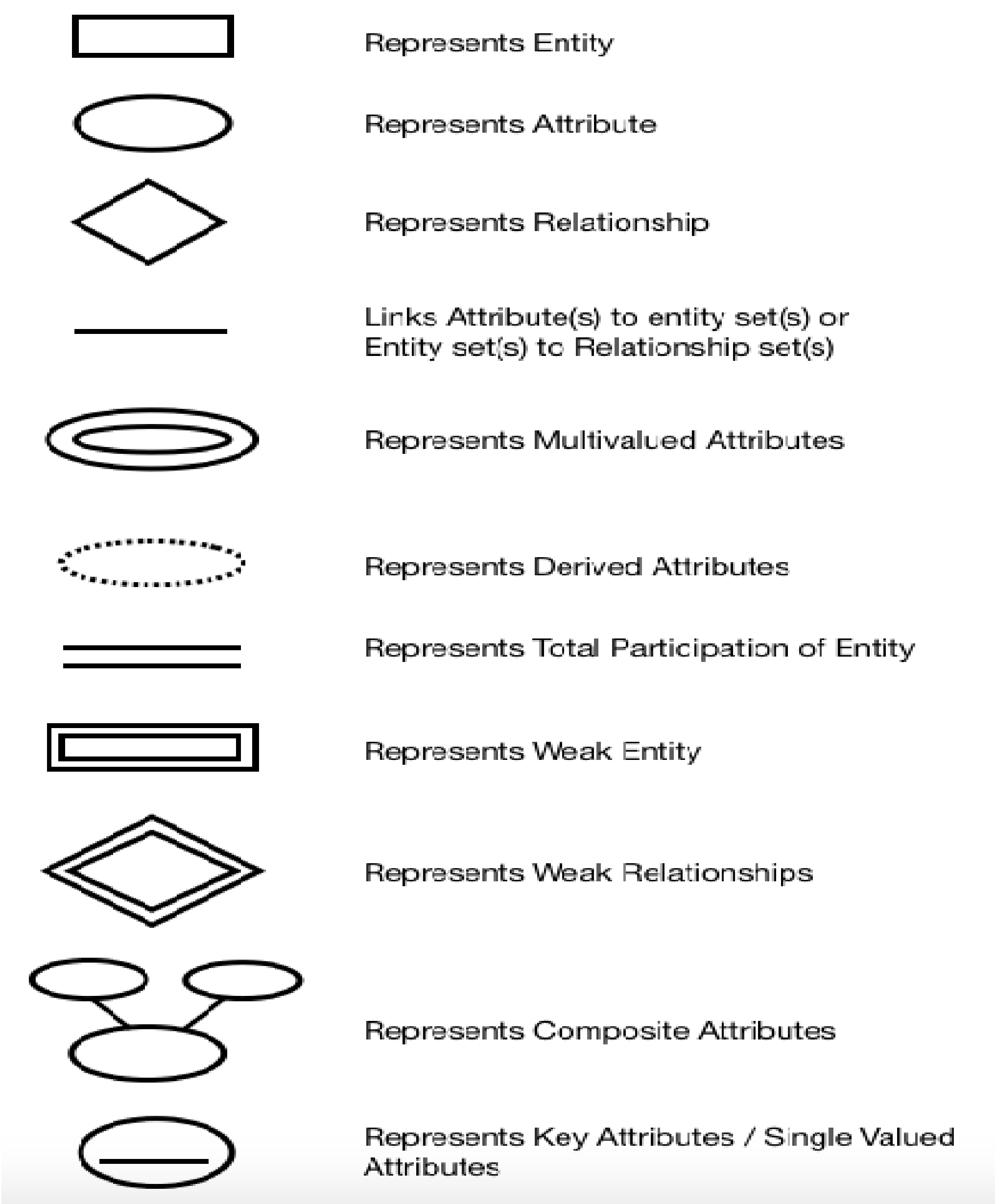
ER diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

Components of ER diagram



Symbols used in ER diagram



Represents Entity

Represents Attribute

Represents Relationship

Links Attribute(s) to entity set(s) or
Entity set(s) to Relationship set(s)

Represents Multivalued Attributes

Represents Derived Attributes

Represents Total Participation of Entity

Represents Weak Entity

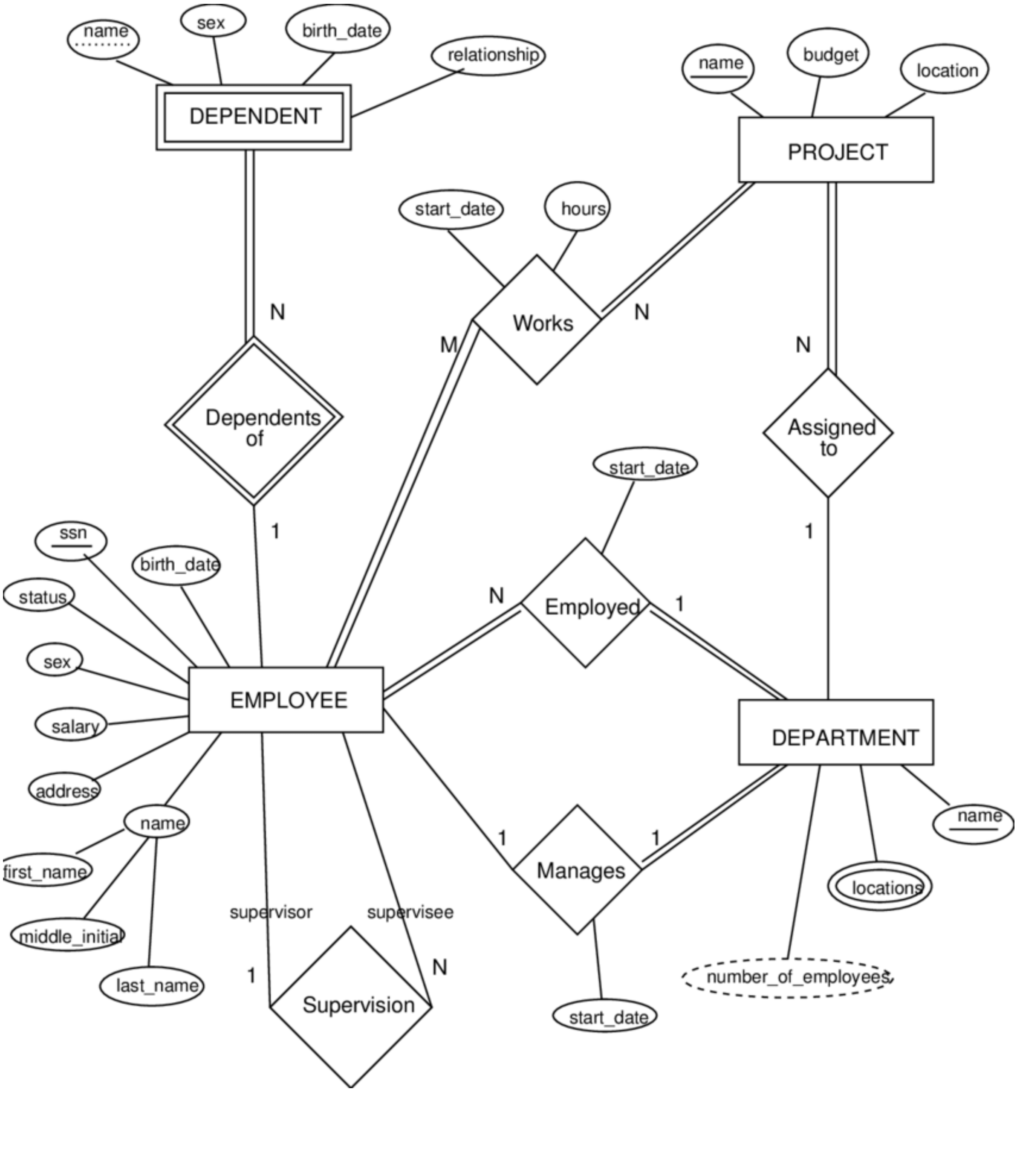
Represents Weak Relationships

Represents Composite Attributes

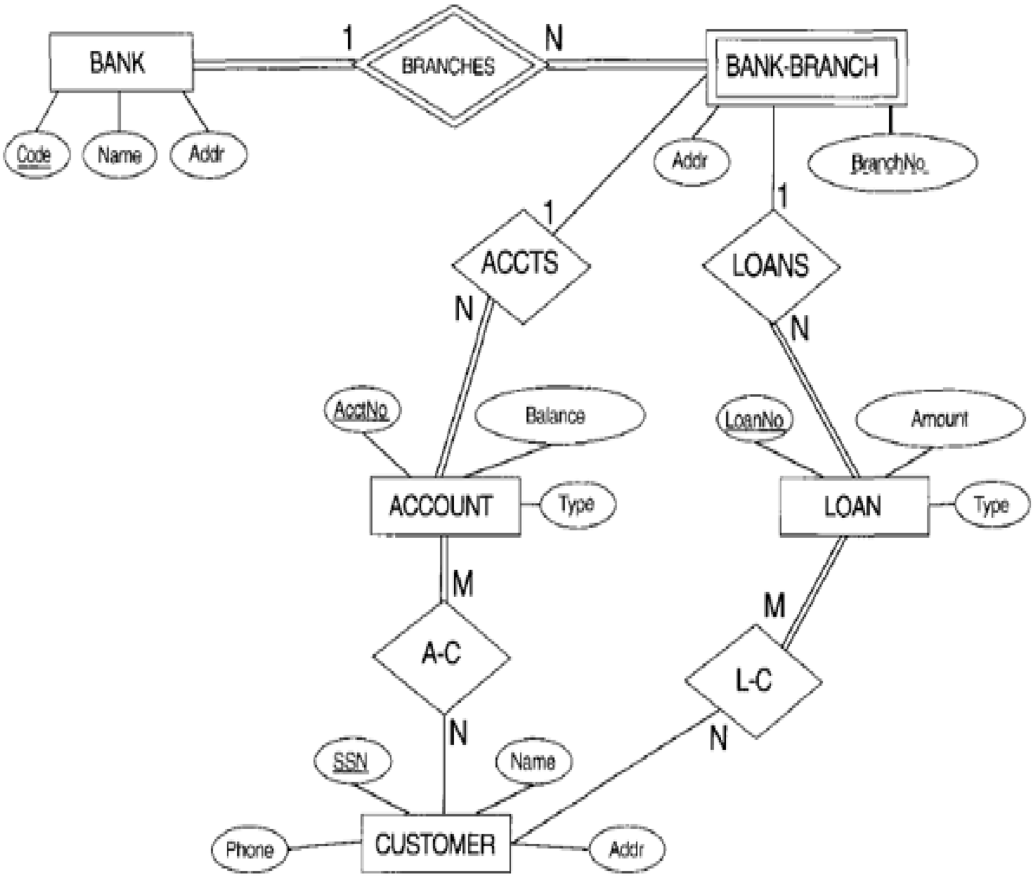
Represents Key Attributes / Single Valued
Attributes

Experiment 1

Create ER diagram for company database using Drawio.



Create ER diagram for Bank database using Drawio.



Basic SQL commands

To Create database: CREATE database_name;

Drop command to remove existing database: DROP DATABASE IF EXISTS data base_name;

To use created database: USE database_name;

To list tables in a database: SHOW tables;

Drop command to remove existing table: DROP DATABASE IF EXISTS table_name;

To describe a database: DESC table_name;

PROBLEM STATEMENT

2) Consider the **Company database** and create the below tables by properly specifying the primary keys and the foreign keys.

Employee (Fname: varchar, Minit: Char, Lname: varchar, ssn:int, Bdate: Date, Address: varchar, Sex: char, salary: decimal,Super_ssn:int, DNO:int)

Department (Dname: varchar, Dnumber: int, mgr_ssn: int, mgr_start_date: date)

Dept_location (Dnumber: int, Dlocation: varchar)

Project (pname: varchar, pnumber: int, plocation: varchar, dnum:int)

Works_on (Essn: int, pno:int, hours: decimal)

Dependent (Essn: char, dependent_name: varchar, sex: char, Bdate: date, relationship: varchar)

```
CREATE TABLE DEPARTMENT (
  dname        varchar(25) not null,
  dnumber      int not null,
  mgrssn      char(9),
  mgrstartdate date ,
  CONSTRAINT pk_Department primary key (dnumber),
  CONSTRAINT uk_dname UNIQUE (dname));

CREATE TABLE EMPLOYEE (
  fname    varchar(15) not null,
  minit    varchar(1),
  lname    varchar(15) not null,
  ssn     char(9) not null,
  bdate    date,
  address  varchar(50),
  sex      char,
  salary   decimal(10,2),
  superssn char(9),
  dno      int not null ,
  CONSTRAINT pk_employee primary key (ssn),
  CONSTRAINT emp_superssn foreign key(superssn)references EMPLOYEE(ssn)
  On delete not NULL on update cascade,
  CONSTRAINT fk_employee_department foreign key (dno) references
DEPARTMENT(dnumber)
  on delete cascade on update cascade);
```

```
ALTER TABLE DEPARTMENT
    ADD CONSTRAINT Dept_mgrfk
    foreign key (mgrssn)references EMPLOYEE(ssn) on delete  cascade ON
UPDATE CASCADE ;



CREATE TABLE DEPENDENT (
  essn           char(9),
  dependent_name varchar(15),
  sex            char,
  bdate          date,
  relationship   varchar(8),
  CONSTRAINT pk_essn_dependent_name primary key (essn,dependent_name),
  CONSTRAINT fk_dependent_employee foreign key (essn) references
EMPLOYEE(ssn) );
CREATE TABLE DEPT_LOCATIONS (
  dnumber   int,
  dlocation varchar(15),
  CONSTRAINT pk_dept_locations primary key (dnumber,dlocation),
  CONSTRAINT fk_deptlocations_department foreign key (dnumber) references
DEPARTMENT(dnumber) on delete cascade on update cascade
);
```

```
CREATE TABLE PROJECT (
  pname      varchar(25) not null,
  pnumber    int,
  plocation  varchar(15),
  dnum       int not null,
  CONSTRAINT ok_project primary key (pnumber),
  CONSTRAINT uc_pnumber unique (pname),
  CONSTRAINT fk_project_department foreign key (dnum) references
DEPARTMENT(dnumber)
);
```

```
CREATE TABLE WORKS_ON (
  essn   char(9),
  pno    int,
  hours  decimal(4,1),
  CONSTRAINT pk_worksOn primary key (essn,pno),
  CONSTRAINT fk_workson_employee foreign key (essn) references
EMPLOYEE(ssn),
  CONSTRAINT fk_workson_project foreign key (pno) references
PROJECT(pnumber)
);
```

ii. Insert at least five tuples in each relation.

Syntax of INSERT command: INSERT INTO table_name VALUES(list of values corresponding to each attribute);

Ex: INSERT INTO DEPARTMENT VALUES ('Research','5','333445555','1978-05-22');

3)

i.**Retrieve the name and address of all employees who work for the 'Research' department.**

select e.fname,e.address from employee e, department d where d.dname="Research" and d.dnumber = e.dno;

ii. **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.**

select p.pnumber,p.dnum,e.lname,e.address,e.bdate from project p, department d, employee e where p.plocation="Stafford" and p.dnum= d.dnumber and d.mgrssn=e.ssn;

iii. **For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.**

select e.fname,e.lname,s.fname,s.lname from employee as e, employee as s where s.superssn=e.ssn;

iv. 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.

```
(select distinct pnumber from project,department,employee where dnum=dnumber and
mgrssn=ssn and lname="Smith")

union

(select distinct pnumber from project,works_on,employee where pnumber=pno and
essn=ssn and lname="Smith");
```

v. Retrieve all employees whose address is in Houston, Texas.

```
select fname,mname,lname from employee where address="Houston,Texas";
```

vi. Retrieve all employees in department 5 whose salary is between $30,000 and $40,000.

```
select * from employee where dno=5 and salary >=30000 and salary <=40000;
```

PROBLEM STATEMENT

5. Consider the following database for a banking enterprise

BRANCH (branch_name: string, branch_city: string, assets: real)
ACCOUNT (accno: int, branch_name: string, balance: real)
CUSTOMER (customer_name: string, customer_street: string, customer_city: string)
DEPOSITOR (customer_name: string, accno: int)
LOAN (loan_number: int, branch_name: string, amount: real)
BORROWER (customer_name: string, loan_number: int)

i) Create the above tables by properly specifying the primary keys
and the foreign keys.
ii) Enter atleast five tuples for each relation.
iii) Find all the customers who have atleast two accounts at the ***Main*** branch.
iv) Find all the customers who have an account at ***all*** the branches located in a specific city.
v) Demonstrate how you delete all account tuples at every branch located in a specific city.

TABLE CREATION

```
SQL> create table branch
(
branch_name varchar(25) primary key,
branch_city varchar(20) not null,
assets decimal(10,2) not null
);
Table created.
```

TABLE DESCRIPTION

```
SQL> describe branch;
Name Null? Type
-----------------------------------------------------------------------------
BRANCH_NAME NOT NULL VARCHAR2(25)
BRANCH_CITY NOT NULL VARCHAR2(20)
ASSETS NOT NULL NUMBER(10,2)
```

INSERTING TUPLES

```
SQL> insert into branch values('jaynagar’,'bangalore','15000000');
SQL>insert into branch values(‘basavanagudi’,’bangalore’,’25000000’);
SQL>insert into branch values(‘noida’,’delhi’,’50000000’);
SQL>insert into branch values(‘marine drive’,’mumbai’,’40000000’);
SQL>insert into branch values(‘grren park’,’delhi’,’30000000’);
```

```
BRANCH_NAME      BRANCH_CITY          ASSETS
------------------------------------------------------------------------------------
Jaynagar                bangalore               15000000
basavanagudi            bangalore               25000000
noida                   delhi                   50000000
marine drive            mumbai                  40000000
green park              delhi                   30000000
```

TABLE CREATION

```
SQL> create table account
(
accno int primary key,
branch_name varchar(25) not null,
balance decimal(10,2) not null,
foreign key(branch_name) references branch(branch_name)
);
Table created.
```

TABLE DESCRIPTION

```
SQL> describe account;

Name                    Null?                Type
--------------------------------------------------------------------------------
ACCNO                NOT NULL             NUMBER(38)
BRANCH_NAME    NOT NULL             VARCHAR2(25)
BALANCE             NOT NULL             NUMBER(10,2)
```

INSERTING TUPLES

```
SQL>insert into account values(‘123’,’jaynagar’,’25000’);
SQL>insert into account values(‘156’,’jaynagar’,’30000’);
SQL>insert into account values(‘456’,‘basavanagudi’,’15000’);
SQL>insert into account values(‘789’,‘noida’,’25000’);
SQL>insert into account values(‘478’,’marine drive’,’48000’);
SQL>insert into account values(‘778’,’green park’,’60000’);
SQL>insert into account values('189','basavanagudi','48888');

SQL> select * from account;
ACCNO              BRANCH_NAME           BALANCE
---------- ------------------------ ----------------------------------
123                    jaynagar                        25000
156                    jaynagar                        30000
456                  basavanagudi                      15000
789                     noida                          25000
478                  marine drive                      48000
778                  green park                        60000
189                  basavanagudi                      48888
```

TABLE CREATION

```
SQL> create table customer
(
customer_name varchar(25) primary key,
customer_street varchar(25) not null,
customer_city varchar(25) not null
);
Table created.
```

TABLE DESCRIPTION

```
SQL> describe customer;
Name                          Null?              Type
-----------------------------------------------------------------------------
CUSTOMER_NAME            NOT NULL           VARCHAR2(25)
CUSTOMER_STREET         NOT NULL           VARCHAR2(25)
CUSTOMER_CITY             NOT NULL           VARCHAR2(25)
```

INSERTING TUPLES

```
SQL> insert into customer values(‘ramu’,’jaynagar’,’bangalore’);
SQL> insert into customer values(‘kumar’,’basavanagudi’,’bangalore’);
SQL> insert into customer values(‘john’,’noida’,’delhi’);
SQL> insert into customer values(‘mike’,’marine drive’,’mumbai’);
SQL> insert into customer values(‘sachin’,’green park’,’delhi’);

SQL> select * from customer;

CUSTOMER_NAME              CUSTOMER_STREET          CUSTOMER_CITY
----------------------------------------------------------------------------------------------------
ramu                                    jaynagar                         bangalore
kumar                               basavanagudi                    bangalore
john                                    noida                               delhi
mike                                marine drive                       mumbai
sachin                                   green park                        delhi
```

TABLE CREATION

```
SQL> create table depositor
(
customer_name varchar(25) not null,
accno int not null,
foreign key (customer_name) references customer (customer_name)
foreign key(accno) references account(accno) on delete cascade
);
Table created.
```

TABLE DESCRIPTION

```
SQL> describe depositor;

Name                              Null?                         Type
------------------------------------------------------------------------
CUSTOMER_NAME        NOT NULL                VARCHAR2(25)
ACCNO                         NOT NULL                  NUMBER(38)
```

INSERTING TUPLES

```
SQL> insert into depositor values(‘ramu’,’123’);
SQL> insert into depositor values(‘ramu’,’156’);
SQL> insert into depositor values(‘ramu’,’189’);
SQL> insert into depositor values(‘kumar’,’456’);
SQL> insert into depositor values(‘john’,’789’);
SQL> insert into depositor values(‘mike’,’478’);
SQL> insert into depositor values(‘sachin’,’778’);

SQL> select *from depositor;

CUSTOMER_NAME          ACCNO
------------------------------------------------
ramu                     123
ramu                     156
ramu                     189
kumar                    456
john                     789
mike                     478
sachin                   778
```

TABLE CREATION

```
SQL> create table loan
(
loan_number int primary key,
branch_name varchar(25) not null,
amount decimal(10,2) not null,
foreign key(branch_name) references branch (branch_name)
);
Table created.
```

TABLE DESCRIPTION

```
SQL> describe loan;



Name                        Null?                       Type
-----------------------------------------------------------------------
LOAN_NUMBER      NOT NULL          NUMBER(38)
BRANCH_NAME      NOT NULL          VARCHAR2(25)
AMOUNT                NOT NULL          NUMBER(10,2)
```

INSERTING TUPLES

```
SQL> insert into loan values(‘1111’,’jaynagar’,’250000’);
SQL> insert into loan values(‘2222’,’basavanagudi’,’350000’);
SQL> insert into loan values(‘3333’,’noida’,’150000’);
SQL> insert into loan values(‘4444’,’marine drive’,’1500000’);
SQL> insert into loan values(‘5555’,’green park’,’7500000’);

SQL> select *from loan;

LOAN_NUMBER           BRANCH_NAME                AMOUNT
--------------------------------------------------------------------------------
1111                             jaynagar                          250000
2222                         basavanagudi                      350000
3333                               noida                            150000
4444                          marine drive                    1500000
5555                           green park                     7500000
```

TABLE CREATION

```
SQL> create table borrower
(
customer_name varchar(25) not null,
loan_number int not null,
foreign key(customer_name) references customer (customer_name),
foreign key (loan_number) references loan (loan_number),
primary key(customer_name,loan_number)
);
Table created.
```

TABLE DESCRIPTION

```
SQL> describe borrower;

Name                         Null?                     Type
-------------------------------------------------------------------------
CUSTOMER_NAME           NOT NULL              VARCHAR2(25)
LOAN_NUMBER             NOT NULL              NUMBER(38)
```

INSERTING TUPLES

```
SQL> insert into borrower values(‘ramu’,’1111’);
SQL> insert into borrower values(‘kumar’,’2222’);
SQL> insert into borrower values(‘john’,’3333’);
SQL> insert into borrower values(‘mike’,’4444’);
SQL> insert into borrower values(‘sachin’,’5555’);

SQL> select *from borrower;

CUSTOMER_NAME              LOAN_NUMBER
-----------------------------------------------------------
ramu                                1111
kumar                               2222
john                                3333
mike                                4444
sachin                              5555
```

QUERIES

1. Find all the customers who have atleast two accounts at the *Main* branch.

```
SQL> select distinct(customer_name), count(*)
from account a, depositor d
where a.accno=d.accno
and d.accno in (select accno from account
where branch_name=’jaynagar’)
group by d.customer_name
having count(*)>=2;
```

RESULT

```
CUSTOMER_NAME           COUNT(*)
--------------------------------------------------
ramu                               2
```

EXPLANATION

The above query is similar to a SELECT-JOIN-PROJECT sequence of relational algebra operations and such queries are called select-join queries. In the WHERE clause, branch_name = ‘jaynagar’ specifies the main branch and a.accno = d.accno is a join condition for the join operation on the two relations account and depositor. Then the GROUP BY clause is used to sub-group the tuples based on the grouping attributes branch_name and customer_name. The HAVING clause provides a condition count (*) >= 2 on the groups of tuples. Only the groups that satisfy the condition are retrieved in the result of the query.

2. Find all the customers who have an account at all the branches located in a specific city.

```
SQL> select d.customer_name
from account a, depositor d, branch b
where b.branch_name=a.branch_name and a.accno=d.accno
and b.branch_city=’bangalore’
having count(distinct b.branch_name)=(select count(branch_name)
from branch
where branch_city=’bangalore’)
group by customer_name;
```

RESULT

```
CUSTOMER_NAME
------------------------
ramu
```

EXPLANATION

The inner query counts the number of branches in ‘bangalore’ which is used to compare with the number of branches in Bangalore in which a customer has accounts. We join the account, depositor and branch tables by specifying the appropriate join conditions and selecting only those tuples having branches in Bangalore grouped by the customer name.

3. Demonstrate how you delete all account tuples at every branch located in a specific city.

```
SQL> delete from account
where branch_name in (select branch_name
from branch
where branch_city='delhi');
```

2 rows deleted. **RESULT**

```
SQL> select *from account;


ACCNO           BRANCH_NAME         BALANCE
-------------------------------------------------------------------
123                 jaynagar              25000
156                 jaynagar              30000
456               basavanagudi            15000
478               marine drive           48000
189               basavanagudi            48888

SQL> select *from depositor;

CUSTOMER_NAME           ACCNO
---------------------------------------------
ramu                       123
ramu                       156
ramu                       189
kumar                      456
mike                       478
```

EXPLANATION

The nested query selects the tuples that satisfy the selection condition branch_city = ‘delhi’ from the relation branch .The IN operator compares the subtuple of value branch_name for each tuple in account relation with the tuples produced by the nested query. Finally, the selected tuples are deleted from the account relation. Here the account tuples containing the branches in delhi, i.e, in noida and green park are deleted. Also, since depositor references the accno from the account relation, for the accno deleted from the account table, corresponding tuples containing the same accno in depositor relation are deleted.