Expt. No.
Date

ENTITY RELATIONSHIP MODEL

Page No.

The ER model defines the conceptual view of a database. It works around real - world entities and the associations among them.

ENTITY

An entity can be a real - world object. that can be easily identifiable. For example, in a School database, Student, teacher, class, and course offered can be considered as entity. All these entities have some attributes or properties

An entity set is a collection of similar type of entities. For example, Students may contain all the students of a School; likewise a Teachers may contain all the teachers of a School Entity sets need not be disjoint.

DESIGN AN ENTITY RELATIONSHIP (ER) MODEL FOR A UNIVERSITY DATABASE.

1. An University contains many departments
2. Each department can offer any number of courses
3. Many instructions can work in a department
4. An instructor can work only in one department
5. For each department there is a Head.
6. An instructor can take head of only one department

Expt. No.
Date

Page No.

7. Each instructor can take any number of courses
8. A Course can be taken by only one instructor
9. A ~~Structure~~ student can enroll for any number of courses
10. Each Course can have any number of Students.

STEP 1: IDENTIFY THE ENTITIES

The entities are
1. Department
2. Course
3. Instructor
4. Student

STEP 2: IDENTIFY THE RELATIONSHIPS
1. One department offers many courses. But one particular course can be offered by only one department. hence the cardinality between department and Course is One to Many (1:N)
2. One department has multiple instructors. But instructor belongs to only one department. Hence the cardinality between department and instructor is One to Many (1:N)
3. One department has only one head and one head can be the head of only one department. Hence the cardinality is one to One. (1:1)

Expt. No. Page No.

Date

4. One Course can be enrolled by many Students and one Student can be enroll for many courses. Hence the cardinality between course and Student is many to Many (M:N)

5. One Course is taught by only one Instructor. But one Instructor teaches many Courses. Hence the Cardinality between course and Instructor is Many to One (N:1)

STEP 3: IDENTITY THE PRIMARY KEY ATTRIBUTE

• "Department ~~x~~ name" Can identityfy a department uniquely. Hence Department_Name is the primary key attribute for the Entity "Department".

• Course_ID is the primary key attribute for "Course" Entity.

• Student_ID is the primary key attribute for "Student" Entity.

• Instructor_ID is the primary key attribute for "Instructor" Entity.

STEP 4: IDENTIFY OTHER RELEVANT ATTRIBUTES

• For the department entity, other attributes are location.

• For Course entity, other attributes are name, duration

• For Instructor entity, other attributes are first_name, last_name, phone

• For student entity, first_name, last_name, phone

Expt. No.
Date
Page No.

STEP 5 : DRAW COMPLETE DIAGRAM

By Connecting all these details, we can now draw ER diagram as given below.

Expt. No.
Date
Page No.

DATA DEFINITION LANGUAGE (DDL) COMMANDS.

The DDL provides commands for defining relation schemas, deleting relations and modifying relation schemas.

DDL is used to:
- Create an object
- Alter the structure of an object.
- To drop object created.

The commands used are:
1. Create
2. Alter
3. Drop
4. Turncate
5. Rename

1. CREATE COMMAND

This command is used to create a table

Synatax:

```
create table tablename
(column_name1 data_type,
 column_name2 data_type...)
```

Expt. No.
Date
Page No.

2. ALTER COMMAND

This command is to add attributes to an existing relation

Syntax:

```
Alter table r add A D;
```

Where r - name of the existing relation,
A - Name of the attribute to be added
D - type of the added attribute

3. DROP COMMAND

Drop command deletes all information about the dropped relation from the database

Syntax:

```
DROP TABLE <Tablename>;
```

4. TRUNCATE COMMAND

Truncate command removes all the records from the table

Syntax:

```
TRUNCATE TABLE <Table_name>
```

5. RENAME COMMAND

Rename command is used to rename the objects.

Syntax:

```
RENAME <OldTableName> To <NewTableName>
```

OUTPUT:-
Table Created

OUTPUT:-

```
Name  Null?                Type
EMPNO                      NUMBER (4)
ENAME                      VARCHAR2(10)
DESIGINATION               VARCHAR2(10)
SALARY                     NUMBER (8, 2)
```

OUTPUT:
Table Created.
To view the EMP1 table

```
SQL > DESC EMP1
Name Null?            Type
EMPNO                 NUMBER (4)
ENAME                 VARCHAR2 (10)
DESIGINATION          VARCHAR2(10)
SALARY                NUMBER (8, 2)
```

Expt. No.
Date

Page No.

EXERCISES:

QUERY : 1

Write a query to create a table employee with employee number, employee name, designation and salary.

```
SQL > CREATE TABLE EMP ( EMPNO NUMBER(4), ENAME
VARCHAR2 (10), DESIGNATION VARCHAR2 (10), SALARY
NUMBER (8, 2));
```

QUERY : 2

Write a query to display the column name and data type of the table employee.

Syntax:

```
DESC < TABLE NAME>;
```

```
SQL >DESC EMP;
```

QUERY : 3

Write a query to create a table from an existing table with all the fields

```
SQL > CREATE TABLE EMP1 AS SELECT * FROM EMP;
```

OUTPUT:-
Table Created

```
To view EMP2
SQL>DESC EMP2
Name Null?        Type
  EMPNO           NUMBER(4)
  ENAME           VARCHAR2(10)
```

OUTPUT:
Table altered.

```
To view EMP
SQL>DESC EMP;

Name Null?        Type
  EMPNO           NUMBER(6)
  ENAME           VARCHAR2(10)
  DESIGNATION     VARCHAR2(10)
  SALARY          NUMBER(8,2)
```

Expt. No.
Date

Page No.

QUERY : 4

Write a query to create a table from an existing table with selected fields

syntax to create a table from an existing with selected fields

SQL > CREATE TABLE < TARGET TABLE NAME> SELECT EMPNO, ENAME FROM < SOURCE TABLE NAME>;

SQL > CREATE TABLE EMP2 AS SELECT EMPNO, ENAME FROM EMP;

ALTER & MODIFICATION ON TABLE

QUERY : 5

Write a Query to Alter the Column EMPNO NUMBER (4) TO EMPNO NUMBER (6).

Syntax for Alter & Modify on a Single column:

SQL > ALTER < TABLE NAME > MODIFY < COLUMN NAME> < DATATYPE > (SIZE);

SQL > ALTER TABLE EMP MODIFY EMPNO NUMBER (6);

OUTPUT :
Table altered

```
SQL > DESC EMP:

Name  Null?          Type
 EMPNO               NUMBER(7)
 ENAME               VARCHAR2(12)
 DESIGNATION         VARCHAR 2(10)
 SALARY              NUMBER (8,2);
```

OUTPUT:
Table altered

```
SQL> DESC EMP;

Name  Null?          Type
 EMPNO               NUMBER(7)
 ENAME               VARCHAR2 (12)
 DESIGNATION         VARCHAR2 (10)
 SALARY              NUMBER (8,2)
QUALIFICATION        VARCHAR2 (6).
```

Expt. No.
Date

Page No.

QUERY : 6

Write a Query to Alter the table employee with multiple Columns (EMPNO. ENAME)
Syntax for alter table with multiple Column:

```
SQL > ALTER <TABLE NAME> MODIFY
< COLUMN NAME1 >< DATA TYPE> (SIZE) , MODIFY
< COLUMN NAME2 > < DATA TYPE > (SIZE)........;
```

```
SQL > ALTER TABLE EMP MODIFY (EMPNO NUMBER (7),
ENAME VARCHAR2 (12);
```

QUERY : 7

Write a query to add a new column in employee relation.
Syntax for add a new column:

```
SQL > ALTER TABLE < TABLE NAME> ADD
( < COLUMN NAME >< DATA TYPE >< SIZE > );
```

```
SQL > ALTER TABLE EMP ADD QUALIFICATION
VARCHAR2 (6);
```

OUTPUT :

Table altered

```
SQL > DESC EMP;
Name Null?                Type
EMPNO                     NUMBER (7)
ENAME                     VARCHAR2 (12)
DESIGNATION               VARCHAR2 (10)
SALARY                    NUMBER (8,2)
QUALIFICATION             VARCHAR2(6)
DOB                       DATE
DOJ                       DATE
```

OUTPUT :

Table dropped

OUTPUT :

Table altered

```
SQL > DESC EMP;
Name Null?                Type
EMPNO                     NUMBER (7)
ENAME                     VARCHAR2(12)
DESIGNATION               VARCHAR2(10)
SALARY                    NUMBER (8,2)
QUALIFICATION             VARCHAR2(6)
DOB                       DATE
```

Expt. No.
Date

QUERY: 8

Write a query to add multiple columns in employee relation.

Syntax for add a new column:

```
SQL > ALTER TABLE < TABLE NAME > ADD (< COLUMN NAME1 >
< DATA SQL > < SIZE >, (< COLUMN NAME 2 > < DATA TYPE >
< SIZE >, .... ;
```

```
SQL > ALTER TABLE EMP ADD (DOB DATE , DOJ DATE);
```

DROP COMMAND

QUERY: 9

Write a query to drop the table

```
SQL > DROP TABLE STUDEND;
```

QUERY: 10

write a query to drop a column from an existing table employee

Syntax to drop a column in the existing table:

```
SQL > ALTER TABLE < TABLE NAME > DROP COLUMN
< COLUMN NAME >;
```

```
SQL > ALTER TABLE EMP DROP COLUMN DOJ;
```

OUTPUT:-

```
Table altered.
```

```
SQL >DESC EMP;
Name Null?            Type
 EMPNO                NUMBER(7)
 ENAME                VARCHAR2(12)
 DESIGNATION          VARCHAR2(10)
 SALARY               NUMBER(8,2)
```

OUTPUT:

```
SQL >DESC EMPLOYEE;

Name Null?            Type
 EMPNO                NUMBER(7)
 ENAME                VARCHAR2(12)
DESIGNATION           VARCHAR2(10)
 SALARY               NUMBER(8,2)
```

Expt. No.
Date

Page No.

QUERY :11

Write a query to drop multiple columns from employee

Syntax for add a new column:

```
SQL > ALTER TABLE < TABLE NAME >
DROP < COLUMNNAME1 >, < COLUMN NAME2 >, ........ ;

SQL > ALTER TABLE EMP DROP (DOB, QUALIFICATION);
```

RENAME COMMAND:

QUERY :12

Write a query to rename table emp to employee

Syntax to rename the table name:

```
RENAME < OldTableName > TO < New
TableName >

SQL > RENAME EMP TO EMPLOYEE;

SQL > DESC EMPLOYEE;
```

Expt. No.
Date

CONSTRAINTS

Page No.

Aim:

To execute constraints in SQL.

Integrity Contraints are part of the table definition that limits and restrict the value entered its columns. Integrity constraints ensure that changes made to the database by authorized users do not result in a loss of data consistency.

TYPES OF CONSTRAINTS:

1) Primary key
2) Foreign key / references
3) Check
4) Unique
5) Not null

PRIMARY KEY

A Primary key is a field in a table which identifies each row / record in a database table. Primary key must contain unique values. This column cannot have NULL values.

To create a PRIMARY KEY in Customer table.

```
CREATE TABLE CUSTOMERS (ID INT(5), NAME VARCHAR (20),
AGE INT (3) ADDRESS CHAR (25), SALARY DECIMAL (18, 2),
PRIMARY KEY (ID));
```

Expt. No.
Date
Page No.

To Create a PRIMARY KEY Constraint on the "ID" column when CUSTOMERS table already exists, following SQL syntax is used.

```
ALTER TABLE CUSTOMER ADD PRIMARY KEY (ID);
```

UNIQUE Constraint

The UNIQUE Constraint prevents two records from having identical values in a particular column.

The following query creates a new table called CUSTOMER. Here ID column is set to UNIQUE, so that two records cannot have same ID.

```
CREATE TABLE CUSTOMER (
ID INT NOT NULL UNIQUE,
NAME VARCHAR (20) NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR (25),
SALARY DECIMAL (18,2),
);
```

If CUSTOMER table has already been created, then to add a UNIQUE constraint to ID column,

```
ALTER TABLE CUSTOMER MODIFY ID INT NOT NULL
UNIQUE;
```

Expt. No.
Date

Page No.

FOREIGN KEY

To ensure that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation. This condition is called referential integrity. Foreign key can be specified as part of the SQL, create table statement by using the foreign key clause.

Consider the structure of the two tables as follows:

CUSTOMERS table:

```
CREATE TABLE CUSTOMERS (
ID INT NOT NULL,
NAMEVARCHAR (20) NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR (25),
SALARY DECIMAL (18, 2),
PRIMARY KEY (ID));
```

ORDERS table:

```
CREATE TABLE ORDERS (
ORD_ID INT NOT NULL,
ORD_DATE DATE,
FOREIGN KEY ID references CUSTOMERS);
```

Expt. No.
Date

Page No.

CHECK Constraint

The CHECK Constraint enables a Condition to check the value being entered into a record. If the condition evaluates to false, the record violates the constraint and doesn't entered into the table.

For example, the following query creates a new table called CUSTOMERS. Here, we add a CHECK with AGE Column. So that you cannot have any CUSTOMER below 18 years:

```
CREATE TABLE CUSTOMERS (
ID INT NOT NULL,
NAME VARCHAR (20) NOT NULL.
AGE INT NOT NULL CHECK (AGE >= 18),
ADDRESS CHAR (25),
SALARY DECIMAL (18, 2),
PRIMARY KEY (ID));
```

If CUSTOMERS table has already been created, then to add a CHECK constraint to AGE Column, the following statement can be given.

```
ALTER TABLE CUSTOMERS MODIFY AGE INT NOT
NULL CHECK (AGE >= 18);
```

Expt. No.
Date

Page No.

NOT NULL

The not null constraint prohibits the insertion of a null value for the attribute

Syntax:

```
Name varchar (20) not null
Budget varchar (12, 2) not null
```

For example, the following query creates a new table called CUSTOMERS. The NAME and AGE attributes is declared as NOT NULL.

```
CREATE TABLE CUSTOMERS (
ID INT,
NAME VARCHAR (20) NOT NULL,
AGE INT NOT NULL CHECK (AGE >= 18),
ADDRESS CHAR (25),
SALARY DECIMAL (18, 2),
PRIMARY KEY (ID));
```