MIS 150, Exam 2 Possible Definition Questions:
THE ULTIMATE COMPREHENSIVE GUIDE PART I

Name the database design that performs the table normalization for the database. **Logical design.**

Name the approach for the information systems requirement analysis that includes the entity relationship diagram. **Data Centered**

Name the database design that defines the access method for the database application. **Physical**

Name the testing method that intends to find the bottleneck of the entire database application. **Stress test**

Name the tool or process that builds a working model to define the end user's requirement for a database application. **Prototype**

Name the constraint that describes the total and partial relationship between two relations in the entity-relationship model. **Participation**

What is the attribute type for an attribute named student_ID that only allows one value in the entity relationship model. **Single**

Name the number of non primary key attribute in a relation that will not violate second normal form. **ZERO**

Name the referential integrity that is the best one for the end user to delete and update the database records. **Cascade**

Name the minimum number of attribute in a relation that might violate the third normal form. **Three**

Name the word that is used to indicate the non-overlapping relationship between subclasses in an enhanced entity relationship model. **OR**

Name the command in the SQL that allows a user to change the primary or foreign key. **Alter**

Name the syntax that can be used to change the domain of any attribute in an existing table of the Structured Query Language. **Alter**

Name the command in the SQL that allows an user to add data into a base table of the database. **INSERT**

Name the method that will eliminate the data insertion and deletion problems in the data centered approach. **Normalization**

Name the method that is used to document the database in the data centered approach. **ER diagram**

Name the storage structure that should be used for sequential and direct accesses in the physical database design. **B+ tree**

Name the CASE tool that is used to support the implementation, testing, and maintenance stages in the database application lifecycle. **Computer aided engineering**

Name the prototyping type for defining the user's needs of the large application system. **Requirement prototyping**

Name the process that builds working model of the database application. **Prototype**

Name the tool or process that builds a working model to define the end user's requirements for a database application. **Prototype**

Name the attribute type that is the calculated value in the entity relationship model. **Derived attribute**

Name the constraint that deals with mandatory and optional in the entity relationship model. **Participation**

Name the word that is used to indicate the overlapping relationship between subclasses in an enhanced relationship model. **AND**

Name the constraint that restricts the end user to enter he predefined values for an attribute of a table in the SQL. **Domain**

Name the syntax that will delete the structure of a table in the SQL. **Drop**

Name the constraint that determines the membership inclusion between subclasses and its superclass. **Participation**

Name the approach for the information systems requirement analysis that includes the table normalization technique. **Data centered**

Name the constraint that describes one to many or many to many relationship between two relations in the entity-relationship mode. **Cardinality**

What is the attribute type for an attribute named TELEPHONE that can have one or more values in the entity relationship model? **Mutlivalue composite**

Name the referential integrity that is the best one for the data administrator to manage deleting and updating the database records. **No action.**

Name the data conversion method that using software to translate the data of the old applications to meet the input requirements of a new application. **Bridge**

Name the most important database selection criterion of an end user. **Easy to use.**

Name the language for creating the table in a database. **DDL data definition language**

Name the theoretical syntax, keyword, to remove the record from a table. **DELETE**

Name the syntax, keyword to give the database access right from a user. **GRANT**

Name the most suitable referential integrity of updating the database from the database administrator's point of view. **No action**

Name one disadvantage of creating a view in a database. **Structure restriction or performance restriction.**

Name the constraint that restricts the null value for the primary key of a table. **Entity Integrity**

Name the improper relationship of entity relationship diagram that has missed a pathway. **Chasm Trap**

Name the improper relationship with an ambiguous pathway in the enhanced entity relationship model. **Fantrap**

What is the attribute type for an attribute named phoneNo that can have two or more values in the entity-relationship model? **Multi-value**
Name the privilege regarding the selecting records from a student table. **Object**

State the entire syntax for a Select statement in the Structure Query Language.

SELECT [DISTINCT | ALL] [columnExpression [AS newName]] [,…]}
FROM TableName [alias] [,…]
[**WHERE** condition]
GROUP BY columnList] [HAVING condition]
HAVING columnList]

Characteristics of E-R Model:

Semantic Data Model
Express the logical properties of an enterprise DB
Design tools and documentation
No physical DBMS
Unified Modeling Language (UML)
Proposed by Dr. Peter Chen

Components of E-R Model:

Entity, Attribute, Key, Relationship, Structural constraints on the relationship

Entity Type: a group of objects with the same properties, which are identified by the enterprise as having an independent existence. An **entity occurrence** is a uniquely identifiable object of an entity type. Diagram = Rectangle.
Relationship Type: a set of meaningful associates among entity types. A **relationship occurrence** is a uniquely identifiable association, which includes one occurrence from each participating entity type. Diagram = line.
Degree of a relationship type: the number of participating entity types in a relationship. (Unary, Binary, Ternary, etc)
Recursive Relationship: relationship where the *same* entity type participates more than once in *different* roles.
Attribute: a property of an entity or relationship type.

Attribute Domain: the set of allowablevalues for one or more attributes.
Attribute Types:

Composite Attribute: Composed of a single component each with an independent existence.
Single-value Attribute: holds a single value for each occurrence of an entity type.
Multi-value Attribute: holds multiple values for each occurrence of an entity type.
Derived Attribute: represents a value that is derivable from the value of a related attribute or set of attributes, not necessarily in the same entity.

Candidate Key: the minimal set of attributes that uniquely identifies each occurrence of an entity type that may relate to a single occurrence of an associated entity type through a particular relationship.
Primary Key: the candidate key that consists of two or more attributes.
Composite Key: is a candidate key that consists of one or more attributes.
Strong Entity Type: (parent, owner, dominant) is not existence-dependent on some other entity type whereas **Weak Entity Type** (child, dependent, subordinate) is existence dependent on some other entity.
Multiplicity: is the number (or range) of possible occurrences of an entity type that may relate to a single occurrence of an associated entity type through a particular relationship.
Cardinality: the maximum number of possible relationships. **One to One** or **One to Many** preferred.
Participation Constraints: determines whether all or only some entity occurrences participate in a given relationship. Ex) (m,or), (m,and), (o,or),(o,and)
Fan Trap: exists where a model represents a relationship between entity types, but the pathway between certain entity occurrences is ambiguous.
Chasm Trap: exists where a model suggests the existence of a relationship between entity types, but the pathway does not exist between certain entity occurrences.
SELECT statement: most important statement in the language and is used to express a query. Combines Selection, Projection and Join.
DOMAIN: is the set of allowable values for one or more attributes.
SUBSELECT: is a complete SELECT statement embedded in another query. A subselect may appear within the WHERE or HAVING clause of an outer SELECT statement, where it is called a **subquery** or **nested** query.
3 TYPES OF SUBQUERY
(**scalar** – returns a single column and a single row; **row**, **table**-returns one or more columns and multiple rows)
Order of operations: NOT, AND, OR
UNION: table containing all rows that are either the first table, second or both.
INTERSECTION: containing all rows that are common to both.
DIFFERENCE: table containing rows that are in first table but are not in second table.

Enhanced Entity Relationship Model

SuperClass: An entity type that includes one or more distinct subgroupings of its occurrences, which require to be represented in a data model. **Tsai says:** including one or more distinct subgroups in the data model.

SubClass: A distinct subgrouping of occurrences of an entity type, which require to be represented in a data model. **Tsai says:** a distinct subgroup of an entity type in the data model.

Attribute Inheritance: The process of maximizing the differences between members of an entity by identifying their distinguishing characteristics. **Tsai calls it:** Specialization Hierarchy.

Generalization: The process of minimizing the differences between entities by identifying their common characteristics. **Tsai calls it:** Generalization Hierarchy.

Constraints on specialization / generalization

- Participation (mandatory / optional)
- Disjoint (OR), Non-Disjoint (AND)

Aggregation (has a or is part of)

Composition (Strong ownership of aggregation)

Design Steps for an Enhanced Entity Relationship Model

Identify:

- Entity Types, relationship types
- Perform Cardinality and participation constraints (put in your one to manies and (m,or) / (o,or) stuff
- Specify Attributes
- Specify Keys
- Perform Specialization /Generalization
- Create your EER Diagram

Ex) Create an enhanced ER Diagram for a rental management using following entities:

- -Rental Agency
- -Staff (Part time, Full Time)
- -Owner
- -Renter
- -Property (Business, Home)

MIS 150, Exam 2 Possible SQL Questions:

THE ULTIMATE COMPREHENSIVE GUIDE PART II

(DML) Data Manipulation Language: for retrieving and updating data.

```
SELECT      [DISTINCT | ALL] {* | [columnExpression [AS newName]] [,…]}
FROM        TableName [alias] [,…]
[WHERE      condition(s)]
[GROUP BY   columnList] [HAVING condition]
[ORDER BY   columnList] [ASC (default in Oracle) | DESC]
```

Additional:

Data Manipulation Language (DML): **SELECT, UPDATE, INSERT, DELETE**

Aggregate Functions: **COUNT, SUM, AVG, MIN, MAX, Aggregate Functions using Group By need other attribute in Group By clause, it is illegal to mix aggregate functions with column names in a select clause unless GROUP BY is used**

Union Compatible Operations: **UNION, MINUS or EXCEPT, INTERSECT**

(DDL) Data Definition Language: Defines the database structure and controls access to the data.

Integrity Enhancement Feature

Required Data: **NOT NULL**

Domain Constraint: **CREATE DOMAIN** DomainName [**AS**] dataType
[**DEFAULT** defaultOption]
[**CHECK** (searchCondition)]

Ex) sex CHAR(1) NOT NULL CHECK (VALUE IN ('m', 'f'))
credit AS NUMBER(3) CHECK (VALUE BETWEEN 0 AND 999)

Ex) CREATE DOMAIN sexType AS CHAR(1) DEFAULT 'm' CHECK(VALUE IN('m','f'));
sex sexType NOT NULL;

Ex) DROP DOMAIN domainName [RESTRICT | CASCADE]
DROP DOMAIN sexType;

Entity Integrity PRIMARY KEY keyName(s) NOT NULL

Ex) PRIMARY KEY studentNo NOT NULL or studentNo INTEGER NOT NULL UNIQUE
Multiple primary keys: PRIMARY KEY (sNo, classNo, sDate) NOT NULL

Referential Integrity

Actions for ON UPDATE and ON DELETE: CASCADE, SET NULL, SET DEFAULT, NO ACTION

Ex) FOREIGN KEY hotelName REFERENCES hotel(hotelNo)
FOREIGN KEY hotelNo REFERENCES hotel(hotelNo) ON DELETE SET NULL
FOREIGN KEY hotelNo REFERENCES hotel(hotelNo) ON UPDATE CASCADE

Enterprise Constraint (ASSERTIONS)

CHECK clause, UNIQUE clause, CREATE ASSERTION statement

CREATE ASSERTION assertionName CHECK (assertion condition) is used to defined attribute constraint

Ex) CREATE ASSERTION tooMuch
CHECK (NOT EXIST (SELECT sno FROM enroll GROUP BY sno HAVING COUNT(*) >10));

SQL Data Definition Language (DDL): CREATE, ALTER, DROP

CREATE SCHEMA | DOMAIN | TABLE | INDEX | VIEW
ALTER TABLE | DOMAIN
DROP SCHEMA | DOMAIN | TABLE | INDEX | VIEW

CREATE SCHEMA [name | AUTHORIZATION creater-ID]

Ex) CREATE SCHEMA mis150
AUTHORIZATION Tsai;

DROP SCHEMA name [RESTRICT | CASCADE]
DROP SCHEMA mis150;

CREATE TABLE tableName (col dataType [NOT NULL, UNIQUE, DEFAULT option] [CHECK search condition][,…,],[PRIMARY KEY (col [,col,…]),[FOREIGN KEY (col [,col]) REFERENCES (parentTable [cols])

Ex) CREATE TABLE Student
(stuID NUMBER(5) NOT NULL CHECK (VALUE BETWEEN 00001 AND 99999),
facID NUMBER(5),
sched CHAR(10),
room CHAR(10),
CONSTRAINT pkClass PRIMARY KEY (course#),
CONSTRAINT fkClass FOREIGN KEY (facID) REFERENCES Faculty(FacID);

Ex) CREATE DOMAIN propertyNo AS SMALLINT;
CREATE DOMAIN staffNo AS CHAR(5) CHECK (VALUE IN (SELECT Sno FROM STAFF));
CREATE DOMAIN pRent AS DECIMAL(6,2) CHECK (VALUE BETWEEN 0 and 9999.99);
CREATE TABLE propertyForRent

(pNo propertyNo NOT NULL,
sNo staffNo CONSTRAINT StaffNotTooMuch CHECK (NOT EXIST (SELECT sNo FROM propertyForRent GROUP BY sNo HAVING COUNT(*) > 10)) NOT NULL,
rent pRent NOT NULL,
CONSTRAINT pkPropertyForRent PRIMARY KEY (pNo),
CONSTRAINT fkPropertyForRent FOREIGN KEY (sNo) REFERENCES Staff(sNo));

SQL DDL for Table

DROP TABLE tableName [RESTRICT | CASCADE];
ALTER TABLE tableName
[ADD [COLUMN] columnName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption] [CHECK (search condition)]]
[DROP [COLUMN] columnName [RESTRICT | CASCADE]]
[ADD [CONSTRAINT [constraintName]] tableConstraintDefinition]]
[DROP CONSTRAINT constraintName [RESTRICT | CASCADE]]
[ALTER [COLUMN] SET DEFAULT defaultOption]
[ALTER [COLUMN] DROP DEFAULT];

Ex) ALTER TABLE Enrollment
MODIFY (grade Number(3));
ALTER TABLE Enrollment
ADD (dateTaken DATE NOT NULL);
DROP TABLE Enrollment;

SQL DDL for INDEX

Index: structure that provides accelerated access to the rows of a table based on the values of one or more cols.

CREATE [UNIQUE] INDEX indexName ON baseTableName(col [ASC | DESC] [,…]);
DROP INDEX indexName;

Ex) CREATE INDEX studentName ON Student(stuName DESC);
CREATE INDEX majorCredit ON Student(major,credit);
DROP INDEX majorCredit;

Access Control

GRANT systemPrivilege [ALL PRIVILEGES] TO roleName [WITH GRANT OPTION];
GRANT objectPrivilege [ALL PRIVILEGES] ON [owner] objectName TO roleName [WITH GRANT OPTION];
GRANT roleName TO userName;
REVOKE privilege FROM roleName;
REVOKE roleName FROM userName;

System Privileges

CREATE TABLE, CREATE VIEW, CREATE USER, ALTER INDEX

Object Privileges

SELECT, INSERT, UPDATE, DELETE

GRANT CREATE TABLE, CREATE VIEW, CREATE USER TO manager; (manager is role name)
GRANT SELECT, INSERT, UPDATE, DELETE ON Student TO manager;

Ex) GRANT manager TO Tsai;
REVOKE DELETE ON Student FROM manager;
REVOKE manager FROM Tsai;

SAMPLE SQL (DML) QUESTIONS

Patient (patientNo, patName, patAddr, DOB, ssNo, driLicNo)
Building (buildingNo, buildingName, buildingType, noOfBeds)
Admitting (patientNo, buildingNo, admissionDate, doctorNo)
Drug (drugNo, drugName, costPerUnit)
Prescribing (patientNo, drugNo, unitsPerDay, startDate, doctorNo)
Doctor (doctorNo, docName, ssNo)

a. (8 points) Use the SQL to retrieve all patients who took drug named "vitamine" starting on 11/11/06. The report contains patientNo, patName, patAddress, and drugNo.

```
SELECT patientNo, patName, patAddress, drugNo

FROM Patient pa, Drug d, Prescribing pr

WHERE pa.patientNo = pr.patientNo

AND pr.drugNo = d.drugNo

AND drugName = 'Vitamine'

AND startDate => '11/11/06';
```

b. (8 points) Use SQL to list the doctors that <u>have not been</u> admitted any patient. The report contains doctorNo and docName.

<u>First way:</u>

```
SELECT doctorNo, doctName

FROM doctor d, admitting a

WHERE d.doctorNo  = a.doctorNo

AND a.doctorNo IS NULL;
```

<u>Second way:</u>

```
SELECT doctorNo, doctName

FROM doctor d

WHERE d.doctorNo NOT IN

        (SELECT a.doctorNo

        FROM admitting a);
```

c. (8 points) Use SQL to find the building name that has housed the most patient.

```
SELECT buildingName, MAX(patientCount)
FROM
        (SELECT buildingName, COUNT(a.patientNo) AS patientCount

        FROM Building b, Admitting a

        WHERE b.buildingNo = a.buildingNo

        GROUP BY buildingName)

GROUP BY buildingname;
```

d. (8 points) Use SQL to retrieve all patients who have been staying in the building named either "Recovery" or "Intensive-Care". The report contains patientNo and patName.

```
SELECT patientNo, patName

FROM Patient p, Admitting a, Building b

WHERE p.patientNo = a.patientNo

AND a.buildingNo = b.buildingNo

AND b.buildingName IN ('Recovery', 'Intensive-Care');
```

e. (12 points) Use SQL to create a view named Uselessbuilding. The Uselessbuilding contains those buildings that <u>have not been</u> used to house any patient. The Uselessbuilding contains buildingNo and buildingName.

<u>First way:</u>

```
CREATE VIEW Uselessbuilding (buildingNo, buildingName) AS
SELECT buildingNo, buildingName
FROM Building b, Admitting a
WHERE b.buildingNo = a.buildingNo
AND a.patientNo IS NULL;
```

<u>Second way:</u>

```
CREATE VIEW Uselessbuilding (b.buildingNo, b.buildingName) AS
SELECT b.buildingNo, b.buildingName
FROM Building b
WHERE b.buildingNo NOT IN
        (SELECT a.buildingNo
        FROM Admitting a);
```

SAMPLE SQL (DDL) QUESTIONS

f. (18 points) Assume that Patient, Building, and Doctor tables have properly been created. Use SQL to create the Admitting table using the following requirements and define primary key and every foreign key.

Admitting (patientNo, buildingNo, admissionDate, doctorNo)

patientNo: 6 integers and existed in the Patient table
buildingNo: 6 integers and existed in the Building table
admissionDate: a date field and equal to today
doctorNo: 6 integers, existed in Doctor table, and cannot admit more than 3 patients for the same day.

```
CREATE TABLE Admitting (
patientNo INT(6) NOT NULL,
buildingNo INT(6) NOT NULL,
admissionDate  DATE DEFAULT 'SYS_DATE' NOT NULL,
doctorNo INT(6) CHECK(COUNT(patientNo) <=3) NOT NULL,

CONSTRAINT pkadmitting PRIMARY KEY (patientNo, buildingNo, admissionDate),

CONSTRAINT fkpatient FOREIGN KEY (patientNo) REFERENCES Patient(patientNo),
CONSTRAINT fkbuilding FOREIGN KEY (buildingNo) REFERENCES Building(buildingNo),
CONSTRAINT fkadmitting FOREIGN KEY (doctorNo) REFERENCES Doctor(doctorNo)
        );
```

g. (8 points) Use SQL to remove the structure of table Admitting from the database. (Assume that Admitting has twenty thousand records.)

```
DROP TABLE Admitting;
```

Create a separate table with the same structure as the Booking table to hold archive records. Using the INSERT statement, copy the records from Booking table to the archive table relating to bookings before 1 January 2000. Delete all bookings before 1 January 2000 from the Booking table.

```
CREATE TABLE BookingHistory (
        hotelNo number(3) NOT NULL,
        guestNo number (3) NOT NULL,
        dateFrom date NOT NULL check (dateFrom > systemDate),
        dateTo  date NOT NULL check, (dateTo > systemDate),
        roomNo number(3) NOT NULL check roomNo between 1 and 100),
        CONSTRAINT pkguest PRIMARY KEY (hotelNo, guestNo, dateFrom, roomNo)
        CONSTRAINT fkbk FOREIGN KEY (roomNo, hotelNo)
        REFERENCES Room (roomNo, hotelNo);
INSERT INTO BookingHistory
        (SELECT *
        FROM Booking
        WHERE dateTo > '01/01/2000'
DELETE FROM BookingHistory
        WHERE dateTo < '01/01/2000';
```

Find identification number and anme of all students taking any course taught by the faculty member f110. Arrange in order by student identification number.

```
SELECT stuname, stuid
FROM student
WHERE stuid IN
        (SELECT stuid
        FROM enrollment
        WHERE course#
        IN (select course# from class where facid = 'F110))
```

```
ORDER BY stuID;
```

Find the name of all students who are not enrolled in csc201a.

```
SELECT stuname
FROM student
WHERE NOT EXISTS
        (SELECT *
        FROM enrollment
        WHERE student.stuid = enrollment.stuid
        AND course# = 'CSC201A');
```

Find the identification number and name of all students with the largest number of credits.

```
SELECT stuid, stuname
FROM student
WHERE credits =
        (SELECT MAX(credits)
        FROM student);
```

Find all course numbers in which fewer than three students are enrolled.

```
SELECT course#
FROM enrollment
GROUP BY course#
HAVING COUNT(*) <3;
```

Find all course numbers in which fewer than three students are enrolled.

```
SELECT course#
FROM enrollment
GROUP BY course#
HAVING COUNT(*) <3;
```

2. Given the following 5 relations:

EMPLOYEE (ENO, ENAME, ESSNO, BDATE, ADDRESS, SEX, SALARY, DEPTNO)
DEPARTMENT (DNAME, DEPTNO, MGRENO, DLOCATION)
PROJECT (PNAME, PRONO, DEPTNO)
ASSIGNMENT (ENO, PRONO, PDATE, HOURS)
DEPENDENT (ENO, DEPENDENT_NAME, SEX, BDATE, RELATIONSHIP)

Use SQL to list names of all employees with two or more dependents. The report should have the following information: ENAME.

```
SELECT ename, eno

FROM employee

WHERE eno IN

        (SELECT eno

        FROM dependent

        GROUP BY eno

        HAVING COUNT(eno) >= 2);
```

List cust_id, f_name, l_name, and city of those customers who have never rented a video and live in Sacramento.

```
SELECT cust_id, f_name, l_name, city

FROM customer c

WHERE cust_id NOT IN

        (SELECT video_id

        FROM rental r

        WHERE r.cust_id = c.cust_id

        AND city = 'Sacramento');
```

How many different customers have rented videotapes in October 2003?

```
        SELECT COUNT(DISTINCT cust_id) AS NumOfCust

        FROM rental

        WHERE rent_date BETWEEN '10/01/03' AND '10/31/03';
```

List the movie_id and the number of its video tape associated with each movie_id for the stores located in Davis.

```
SELECT movie_id, COUNT(video_id) AS NumOfTapes
FROM videotape v, movie m
WHERE store_id IN
        (SELECT store_id
        FROM store
        WHERE city = 'Davis')
GROUP BY movie_id;
```

What is the lost income from un-rented video tapes for all the stores today?

```
SELECT SUM(price) AS LostIncome
FROM videotape
WHERE video_id NOT IN
        (SELECT video_id
        FROM rental
        WHERE rentaldate = 'today');
```

Create a view containing the video_id and its title for the rating that is equal to G.

```
CREATE VIEW ratings
AS SELECT video_id, title
FROM videotape v, movie m
WHERE v.movie_id = m.movie_id AND m.rating = 'G';
```

Add every foreign key and proper primary key for the rental table. (Assume that the rental table has been created without any primary key and foreign keys and the customer table, videotape table, movie table and store table have been created with proper primary keys and foreign keys.)

```
ALTER TABLE rental

ADD (CONSTRAINT fkcustid FOREIGN KEY(cust_id) REFERENCES customer(cust_id)),

        (CONSTRAINT fkvideoid FOREIGN KEY(video_id) REFERENCES videotape(video_id)),

        (CONSTRAINT pkrental PRIMARY KEY(cust_id, video_id, rent_date));
```

Use Structure Query Language to list the number of course that could be offered by each department. The list should include department# and the number of course.

```
SELECT department#, COUNT(course#)

FROM course

GROUP BY department# ;
```

(d) (14 Points) Use Structure Query Langue to list every booktitle where the book is currently adopted by a course and published by a publisher located in the United States.

```
SELECT booktitle

FROM book

WHERE (book_isbn IN

        (SELECT book_isbn

        FROM book_adoption

        WHERE semesterdate = 'SPRING03'))

        AND (publisher# IN (

                SELECT publisher#

                FROM publisher

                WHERE country = 'United States'));
```

FROM CHAPTER 5 (DML) HOMEWORK (HANDOUT)

Branch (Bno, Street, Area, City, Pcode, Tel_No, Fax_NO)
Staff (Sno, FName, LName, Address, Tel_No, Position, Sex, DOB, Salary, NIN, Bno)
Property_for_Rent (Pno, Street, Area, City, Pcode, Type, Rooms, Rent, Ono, Sno, Bno)
Renter (Rno, Fname, Lname, Address, Tel_NO, Pref_Type, Max_Rent)
Owner (Ono, Fname, LName, Address, Tel_No)
Viewing (Rno, Pno, Date, Comment)

A list (Bno, Street, Area, Tel_No, Fax_No) of branches located in Bay Area.

```
SELECT (Bno, Street, Tel_No, Fax_No)

FROM Branch

WHERE Area = 'Bay';
```

A list (Pno, Street, Area, City) of properties for rent with 4 bedrooms or less than $1,000 per month rent.

```
SELECT (Pno, Street, Area, City)

FROM Property_for_rent

WHERE Rooms = 4 OR

Rent < 1000;
```

A list (Sno, FName, LName) of female mangers with salary between one to two million dollars.

```
SELECT (Sno, FName, LName)

FROM Staff

WHERE Position = 'manager' AND

Sex = 'f' AND

Salary BETWEEN 1000000 AND 2000000;
```

A list (Pno, Street, Area, City) of properties for rent located in SF, LA, NY, or DC.

```
SELECT (Pno, Street, Area, City)

FROM Property_for_rent

WHERE City IN ('SF', 'LA', 'NY', 'DC');
```

A list (Pno, Street, Area, City) of properties for rent not located in SF, LA, NY, or DC.

```
SELECT (Pno, Street, Area, City)
FROM Property_for_rent
WHERE City NOT IN ('SF', 'LA', 'NY', 'DC');
```

A list (Ono, Fname, Lname) of owners without any telephone.

```
SELECT (Ono, Fname, Lname)
FROM Owner
WHERE Tel_No IS NULL;
```

The total salary of male managers

```
SELECT SUM(Salary)
FROM Staff
WHERE Sex = 'm' AND
Position = 'manager';
```

The minimum salary of the female manager

```
SELECT MIN(Salary)
FROM Staff
WHERE Sex = 'f' AND
Position = 'manager';
```

The maximum rent to rent a property

```
SELECT MAX(Rent)
FROM Property_for_rent;
```

The average salary of male staff members

```
SELECT AVG(Salary)
FROM Staff
WHERE Sex = 'm';
```

The average number of rooms in the single house type.

```
SELECT AVG(Rooms)

FROM Property_for_rent

WHERE Type = 'single house';
```

The number of employee in each branch.

```
SELECT Bno, COUNT(Sno)
FROM Staff
GROUP BY Bno;
```

OR

```
SELECT Bno, COUNT(Sno)
FROM Branch b, Staff s
WHERE b.Bno=s.Bno
GROUP BY Bno;
```

The average salary of each branch

```
SELECT Bno, AVG(Salary)

FROM Staff

GROUP BY Bno;
```

The average salary in each branch with more than 10 staff members

```
SELECT Bno, AVG(Salary)

FROM Staff

GROUP BY Bno

HAVING COUNT(*) > 10;
```

The number of employee in each branch located in SF, NY, LA, and DC.

```
SELECT City, Bno, COUNT(Sno)

FROM Staff s, Branch b
```

```
WHERE s.Bno = b.Bno AND

City IN ('SF', 'LA', 'NY', 'DC')

GROUP BY (City, Bno);
```

The number of employees in each branch with more than 10 employees and located in SF, NY, LA, and DC.

```
SELECT City, Bno, COUNT(Sno)

FROM Branch b, Staff s

WHERE b.Bno = s.Bno

AND City IN ('SF', 'NY', 'LA', 'DC')

GROUP BY (City, Bno)

HAVING COUNT(*) > 10;
```

A list (Pno, Street, Area, City) of properties for rent managed by John Dow and owned by Sue Lee.

```
SELECT (Pno, Street, Area, City)

FROM Property_for_rent p, Owner o, Staff s

WHERE p.Ono = o.Ono

AND s.Sno = p.Sno

AND (s.FName = 'John' AND s.LName = 'Dow')

AND (o.FName = 'Sue' AND o.LName = 'Lee');
```

A list (Sno, FName, LName) of staff without managing any property for rent.

```
SELECT (Sno, FName, LName)

FROM Staff

WHERE Sno NOT IN

        (SELECT Sno

        FROM Property_for_rent);
```

A list (Sno, FName, LName) of staffs without managing any property for rent.

```
SELECT (Sno, FName, LName)
FROM Staff s
WHERE NOT EXIST
        (SELECT *
        FROM Property_for_rent p
        WHERE p.Sno = s.Sno);
```

A list (Sno, FName, LName) of staffs without managing any property for rent.

```
SELECT (Sno, FName, LName)
FROM Staff s LEFT JOIN Property_for_rent p
ON s.Sno = p.Sno
WHERE Pno IS NULL;
```

A list (Rno, Fname, Lname) of renters without looking any property for rent.

```
SELECT (Rno, FName, LName)
FROM Renter
WHERE Rno NOT IN
        (SELECT Rno
        FROM Viewing);
```

A list (Rno, Fname, Lname) of renters without looking any property for rent.

```
SELECT (Rno, FName, LName)
FROM Renter r
WHERE NOT EXIST
        (SELECT *
        FROM Viewing v
        WHERE r.Rno = v.Rno);
```

A list (Rno, Fname, Lname) of renters without looking any property for rent.

```
SELECT (Rno, FName, LName)
FROM Renter r LEFT JOIN Viewing v
ON r.Rno = v.Rno
WHERE Pno IS NULL;
```

A list (Pno, Street, Area, City) of property for rent without any viewing.

```
SELECT (Pno, Street, Area, City)
FROM Property_for_rent
WHERE Pno NOT IN
        (SELECT Pno
        FROM Viewing);
```

A list (Pno, Street, Area, City) of property for rent without any viewing.

```
SELECT (Pno, Street, Area, City)
FROM Property_for_rent p
WHERE NOT EXIST
        (SELECT *
        FROM Viewing v
        WHERE p.Pno = v.Pno);
```

A list (Pno, Street, Area, City) of property for rent without any viewing.

```
SELECT (Pno, Street, Area, City)
FROM Property_for_rent p LEFT JOIN Viewing v
ON p.Pno = v.Pno
WHERE v.Pno IS NULL;
```

A list (Sno, FName, LName) of staff managing a property for rent.

```
SELECT Distinct(Sno, FName, LName)
FROM Staff s, Property_for_rent p
WHERE s.Sno = p.Sno;
```

A list (Rno, Fname, Lname) of viewing renter.

```
SELECT (Rno, FName, LName)
FROM Renter r, Viewing v
WHERE r.Rno = v.Rno;
```

A list (Sno, FName, LName) of the highest salary staff.

```
SELECT (Sno, FName, LName)
FROM Staff s
WHERE Salary =
        (SELECT MAX(Salary)
        FROM Staff);
```

What is the most common property for rent type for all branches?

```
Select Type
        (Select Type, Max(X)
                (Select Type, Count(*) As X
        From Property_For_Rent
        Group by Type)
Group by Type);
```

What is the type of property that has been viewed the most by renter?

```
Select Type
        (Select Type, Max(X)
                (Select Type, Count(*) As X
        From Property_For_Rent p, Viewing v
        Where p.Pno = v.Pno
        Group by Type)
Group by Type);
```

Name the staff no and his/her branch no managing the most property.

```
Select Bno, Sno
From (Select Bno, Sno, Max(X)
        (Select Bno, Sno, Count(*) As X
        From Property_For_Rent
        Group by Bno, Sno)
Group by Bno, Sno);
```

FROM CHAPTER 5 (DML) HOMEWORK (BOOK)

Hotel (hotelno, hotelname, city)
Room (roomno, hotelno, type, price)
Booking (hotelno, guestno, datefrom, dateto, roomno)
Guest (guestno, guestname, guestaddress)

List the names and addresses of all guests in London, alphabetically ordered by name.

```
SELECT guestname, guestaddress
FROM guest
WHERE guestaddress like 'London'
ORDER BY guestname;
```

List all double or family rooms with a price below $40. 00 per night, in ascending order of price.

```
SELECT *
 FROM room
WHERE price < 40
AND type IN ('Double', 'Family')
ORDER BY price;
```

List the bookings for which no dateto has been specified.

```
SELECT *
FROM booking
WHERE dateto IS NULL;
```

How many hotels are there?

```
SELECT COUNT(hotelno)
FROM hotel;
```

What is the average price of a room?

```
SELECT AVG(price)
FROM room;
```

What is the total revenue per night from all double rooms?

```
SELECT SUM(price)
FROM room
WHERE type = 'Double' ;
```

How many different guests have made bookings for August?

```
SELECT COUNT(DISTINCT guestno)
FROM booking
WHERE (datefrom <= '8/31/06'
AND dateto >= '8/1/06');
```

List the price and type of all rooms at the Grosvenor Hotel.

```
SELECT price, type
FROM room
WHERE hotelno =
        (SELECT hotelno
        FROM hotel
        WHERE hotelname = 'Grosvenor');
```

OR

```
SELECT price, type
FROM room, hotel
WHERE hotel.hotelno = room.hotelno
AND hotelname = 'Grosvenor';
```

List all guests currently staying at the Grosvenor Hotel.

```
SELECT (guestno, guestname, guestaddress)
 FROM guest, booking, hotel
WHERE guest.guestno =booking.guestno
AND hotel.hotelno = booking.hotelno
AND (datefrom <= 'SYSTEM DATE'
AND dateto >= 'SYSTEM DATE')
AND hotelname = 'Grosvenor';
```

OR

```
SELECT *
FROM guest
WHERE guestno IN
        (SELECT guestno
        FROM booking
        WHERE datefrom <= 'SYSTEM DATE'
        AND dateto >= 'SYSTEM DATE'
        AND hotelno =
                (SELECT hotelno
                FROM hotel
                WHERE hotelname = 'Grosvenor')
);
```

List the details of all rooms at the Grosvenor Hotel, including the name of the guest staying in the room, if the room is occupied.

Create a view with every room having a guest (A).

```
CREATE VIEW roomocp (hotelno, roomno, type, price, guestname) AS
        SELECT r.hotelno, r.roomno, r.type, r.price, g.guestname
        FROM hotel h, room r, booking b, guest g
        WHERE h.name = 'Grosvenor'
        AND (b.datefrom <= 'SYSTEM DATE'
        AND b.dateto >= 'SYSTEM DATE')
        AND h.hotelno = r.hotelno
        AND r.hotelno = b.hotelno
        AND r.roomno = b.roomno
        AND b.guestno = g.guestno;
```

Create a view of every room (B).

```
CREATE VIEW roomall (hotelno, roomno, type, price) AS
        SELECT r.hotelno, r.roomno, r.type, r.price
        FROM hotel h, room r
        WHERE h.hotelname ='Grosvenor'
        AND h.hotelno = r.hotelno;
```

Find the answer (C).

```
SELECT r.roomno, r.hotelno, r.type, r.price, p.guestname
FROM roomall r LEFT JOIN roomocp p
ON r.roomno  = p.roomno;
```

What is the total income from bookings for the Grosvenor Hotel today ?

```
SELECT SUM(price)
FROM booking b, room r, hotel h
WHERE (b.datefrom <= 'SYSTEM DATE'
AND b.dateto >= 'SYSTEM DATE')
AND r.hotelno = h.hotelno
AND r.hotelno = b.hotelno
AND r.roomno = b.roomno
AND h.hotelname = 'Grosvenor';
```

List the rooms which are currently unoccupied at the Grosvenor Hotel.

```
SELECT (r.hotelno, r.roomno, r.type, r.price)
FROM room r, hotel h
WHERE r.hotelno = h.hotelno
AND h.hotelname = 'Grosvenor'
AND roomno NOT IN
        (SELECT roomno
        FROM booking b, hotel h
        WHERE (datefrom <= 'SYSTEM DATE'
        AND dateto >= 'SYSTEM DATE')
        AND b.hotelno=h.hotelno
        AND hotelname = 'Grosvenor');
```

OR

```
SELECT (r.hotelno, r.roomno, r.type, r.price)
FROM room r, hotel h
WHERE r.hotelno = h.hotelno
AND h.hotelname = 'Grosvenor'
AND NOT EXIST
        (SELECT *
        FROM booking b, hotel h
         WHERE (datefrom <= 'SYSTEM DATE'
        AND dateto >= 'SYSTEM DATE')
        AND r.hotelno=b.hotelno
        AND r.roomno=b.roomno
        AND r.hotelno=h.hotelno
        AND hotelname = 'Grosvenor');
```

What is the lost income from unoccupied rooms at the Grosvenor Hotel?

```
SELECT SUM(price)
FROM room r, hotel h
WHERE r.hotelno = h.hotelno
AND h.hotelname = 'Grosvenor'
AND roomno NOT IN
        (SELECT roomno FROM booking b, hotel h
        WHERE (datefrom <= 'SYSTEM DATE'
        AND dateto >= 'SYSTEM DATE')
        AND b.hotelno = h.hotelno
        AND r.hotelno=b.hotelno
```

```
        AND r.roomno=b.roomno
        AND h.hotelname = 'Grosvenor');
```

List the number of rooms in each hotel.

```
SELECT hotelno, COUNT(roomno)
FROM room
GROUP BY hotelno;
```

List the number of room in each hotel in London.

```
SELECT r.hotelno, COUNT(roomno)
FROM room r, hotel h
WHERE r.hotelno=h.hotelno
AND city = 'London'
GROUP BY r.hotelno;
```

What is the average number of bookings for each hotel in August?

```
SELECT hotelno, y/31
FROM
        (SELECT hotelno, COUNT(hotelno) AS y
        FROM booking
        WHERE (datefrom <= '8/31/06'
        AND dateto >= '8/1/06'
        GROUP BY hotelno);
```

What is the most commonly booked room type for all hotels in London?

```
SELECT type, MAX(y)
FROM
        (SELECT type, COUNT(type) AS y
        FROM booking b, hotel h, room r
        WHERE r.roomno = b.roomno
        AND r.hotelno = b.hotelno
        AND b.hotelno = h.hotelno
        AND  city = 'London'
        GROUP BY type)
GROUP BY type;
```

OR

```
SELECT hotelno, type, MAX(y)
FROM
        (SELECT  hotelno, type, COUNT(type) AS y
        FROM booking b, hotel h, room r
        WHERE r.roomno = b.roomno
        AND r.hotelno = b.hotelno
        AND b.hotelno = h.hotelno
        AND city = 'London'
        GROUP BY hotelno, type)
GROUP BY hotelno, type;
```

What is the lost income from unoccupied rooms at each hotel today?

```
SELECT r.hotelno, SUM(price)
FROM room r
WHERE NOT EXIST
        (SELECT *
        FROM booking b
        WHERE r.roomno = b.roomno
        AND r.hotelno = b.hotelno
        AND (datefrom <= 'SYSTEM DATE'
        AND dateto >= 'SYSTEM DATE'))
GROUP BY hotelno;
```

Insert rows into each of these tables.

```
INSERT INTO hotel
        VALUES ('h11', 'hilton', 'sacramento');
INSERT INTO room
        VALUES ('hr1111', 'h11', 'single', 120);
```

Update the price of all room by 5%.

```
UPDATE room
SET price = price*1.05;
```

FROM CHAPTER 6 (DDL) HOMEWORK (BOOK)

List the details of all rooms at the Grosvenor Hotel, including the name of the guest staying in the room, if the room is occupied.

Create a view with every room having a guest.

```
CREATE VIEW roomocp (hotelno, roomno, type, price, guestname) AS
        SELECT r.hotelno, r.roomno, r.type, r.price, g.guestname
        FROM hotel h, room r, booking b, guest g
        WHERE h.name = 'Grosvenor'
        AND (b.datefrom <= 'SYSTEM DATE'
        AND b.dateto >= 'SYSTEM DATE')
        AND h.hotelno = r.hotelno
        AND r.hotelno = b.hotelno
        ANDr.roomno = b. roomno
        AND b.guestno = g.guestno;
```

Create a view of every room.

```
CREATE VIEW roomall (hotelno, roomno, type, price) AS
        SELECT r.hotelno, r.roomno, r.type, r.price
        FROM hotel h, room r
        WHERE h.hotelname ='Grosvenor'
        AND h.hotelno = r.hotelno;
```

Find the answer.

```
SELECT r.roomno, r.hotelno, r.type, r.price, p.guestname
FROM roomall r LEFT JOIN roomocp p
ON r.roomno  = p.roomno;
```

Create the Hotel, Room, Booking and Guest tables

Hotel (hotelno, hotelname, city)

Room (roomno, ***hotelno***, Type, price)

Booking (***hotelno***, ***guestno***, datefrom, dateto, ***roomno***)

Guest (guestno, guestname, guestaddress)

```
CREATE TABLE hotel (hotelno   NUMBER(5)   NOT NULL,
hotelname   CHAR(20)   NOT NULL,
city  CHAR(30)   NOT NULL,
CONSTRAINT pkhotel PRIMARY KEY (hotelno));
```

CREATE TABLE room

- **Type must be one of Single, Double, or Family**
- **Price must be between $10 and $100**
- **Roomno must be between 1 and 100**

```
CREATE TABLE room  (roomno   NUMBER(3)  NOT NULL CHECK (VALUE BETWEEN 1 AND 100),
hotelno   NUMBER(5)  NOT NULL,
type   CHAR(8)  CHECK (VALUE IN ('Single', 'Double', 'Family')),
price   NUMBER(5,2)  CHECK (VALUE BETWEEN 10 AND 100),
CONSTRAINT pkroom PRIMARY KEY (roomno, hotelno),
CONSTRAINT fkrh FOREIGN KEY (hotelno) REFERENCES hotel (hotelno));
```

```
CREATE TABLE guest (guestno   NUMBER(10)   NOT NULL,
guestname   CHAR(20)   NOT NULL,
guestaddress    CHAR(30)   NOT NULL,
CONSTRAINT  pkguest PRIMARY KEY (guestno));
```

CREATE TABLE booking

•Datafrom and dateto must be greater than today's date

```
Create TABLE booking (hotelno   NUMBER(5) NOT NULL,
guestno   NUMBER(10)   NOT NULL,
datefrom   DATE  CHECK (datefrom > 'systemdate'),
dateto   DATE  CHECK (dateto > 'systemdate'),
roomno   NUMBER(3)   NOT NULL,
CONSTRAINT  pkbooking PRIMARY KEY (hotelno, guestno, roomno, datefrom),
CONTRAINT fkbh FOREIGN KEY (hotelno) REFERENCES hotel (hotelno),
CONTRAINT fkbg FOREIGN KEY (guestno) REFERENCES guest (guestno),
CONTRAINT fkbr FOREIGN KEY (roomno, hotelno) REFERENCES room (roomno, hotelno));
```

The same room cannot be double-booked

```
CONSTRAINT roombooked CHECK (NOT EXISTS
        (SELECT *
        FROM booking b
        WHERE b.datefrom <= booking.dateto
        AND b.dateto >= booking.datefrom
        AND b.roomno = booking.roomno
        AND b.hotelno = booking.hotelno))
```

The same guest cannot have overlapping booking

```
CONSTRAINT guestbooked CHECK (NOT EXISTS
        (SELECT *
        FROM booking b
        WHERE b.datefrom <= booking.dateto
        AND b.dateto >= booking.datefrom
```

```
        AND b.guestno = booking.guestno))
```

Create a separate table with the same structure as the Booking table to hold archive records. Using the INSERT statement, copy the records from the Booking table to the archive table relating to bookings before 1st January 2003. Delete all bookings before 1st January 2003 from the Booking table.

Create table to hold archive records.

```
CREATE TABLE bookinghis (hotelno   NUMBER(5)   NOT NULL,
guestno   NUMBER(10)   NOT NULL,
datefrom   DATE,
dateto   DATE,
roomno   NUMBER(3)  NOT NULL,
CONSTRAINT pkbhh PRIMARY KEY (hotelno, guestno, datefrom),
CONSTRAINT fkbh FOREGN KEY (hotelno) REFERENCES hotel (hotelno),
CONSTRAINT fkbg FOREIGN KEY (guestno) REFERENCES guest (guestno),
CONSTRAINT fkbr FOREIGN KEY (roomno, hotelno) REFERENCES room (roomno, hotelno));
```

Using INSERT, copy the records from Booking table to Archive table.

```
INSERT INTO bookinghis (hotelno, guestno, datefrom, dateto, roomno) VALUES
        (SELECT *
        FROM booking
        WHERE dateto < '1/1/2003');
```

Delete all bookings before 1st January 2003 from Booking table.

```
DELETE FROM booking
WHERE dateto < '1/1/2003';
```

Create a view containing the hotel name and the names of the guests staying at the hotel.

```
CREATE VIEW hotgst (hotelname, guestname)AS
        SELECT hotelname, guestname
        FROM hotel h, booking b, guest g
```

```
WHERE h.hotelno = b.hotelno
AND b.guestno = g.guestno
AND (datefrom <= 'systemdate')
AND (dateto >='systemdate');
```

Create a view containing the account for each guest at the Gorsvenor Hotel.

```
CREATE VIEW gact (guestno, guestname guestaddress)
    AS SELECT guestno, guestname, guestaddress
    FROM hotel h, booking b, guest g
    WHERE h.hotelno = b.hotelno
    AND b.guestno = g.guestno
    AND h.hotelname='Grosvenor';
```

Give the users Manager and Director full access to these views, with the privilege to pass the access on to other users.

```
GRANT ALL PRIVILEGES
ON gact
TO Manger, Director WITH GRANT OPTIONS;
GRANT ALL PRIVILEGES
ON hotgst
TO Manger, Director WITH GRANT OPTIONS;
```

Give the users Accounts SELECT access to these views. Now revoke the access from user.

```
GRANT SELECT
ON gact, hotgst
TO Accounts;
REVOKE SELECT
ON gact, hotgst
 FROM Accounts;
```

FROM CHAPTER 6 (DDL) HOMEWORK (HANDOUT)

Define City for Branch according to the following constraints:

–Has to have a two character string value

–Has to be equal to SF, NY, LA, or DC

–Set default value to SF

City AS CHAR(2) DEFAULT 'SF' CHECK (VALUE IN ('SF', 'NY', 'LA', 'DC'))

Define Sno for Staff according to the following constraints:

–Has to have a three digits integer

–Has to be between 111 and 999

–Has to be unique

Sno AS SMALLINT NOT NULL UNIQUE CHECK (VALUE BETWEEN 111 AND 999)

Define a domain citylocation for City according to the following constraints:

–Has a two character string value

–Has to be equal to SF, NY, LA, or DC

–Set default value to SF

Define the City in the Branch or Property_for_Rent using the citylocation

CREATE DOMAIN citylocation AS CHAR(2) DEFAULT 'SF' CHECK (VALUE IN ('SF', 'NY', 'LA', 'DC'));

City citylocation NOT NULL

Define a domain noforsno according to the following constraints:

–Has to have a three digits integer

–Has to be equal to one of the Sno in the Staff

Define the sno in the Property_for_rent using the domain noforsno

```
CREATE DOMAIN noforsno AS NUMBER(3) CHECK (VALUE IN
        (SELECT Sno FROM Staff));
Sno  noforsno NOT NULL
```

Define an assertion that will limit 10 staff members in any branch.

```
CREATE ASSERTION bnolimit
        CHECK (NOT EXIST
                (SELECT bno
                FROM staff GROUP BY bno
                HAVING COUNT (*) >10));
```

Create staff table using following constraints

–Sno is three digits integer, between 111 and 999, a primary key

–Name and position are 30 alphanumeric fields

–Sex has a default value (m) limits to m or f

–Salary is between 50,000 and & 100,000

–Bno is equal to one value of bno in branch with 10 or less staff member, set null for delete, set cascade for update.

```
CREATE TABLE staff (Sno  AS SMALLINT NOT NULL UNIQUE CHECK (VALUE BETWEEN 111 AND 999),
name CHAR(30) NOT NULL,
position CHAR(30) NOT NULL,
sex CHAR(1) NOT NULL DEFAULT 'm' CHECK (VALUE IN ('m', 'f')),
Salary NUMBER (8,2) NOT NULL CHECK (VALUE BETWEEN 50000 AND 100000),
Bno SMALLINT NOT NULL CONSTRAINT bnolimit CHECK (NOT EXIST
        (SELECT bno FROM staff GROUP BY bno HAVING COUNT (*) >10),
CONSTRAINT pkstaff PRIMARY KEY (sno),
CONSTRAINT fkstaffbranch FOREIGN KEY (bno) REFERENCES branch (bno)
        ON DELETE SET NULL ON UPDATE CASCADE);
```

Add a new attribute named HireDate to the existing table Staff.

```
ALTER TABLE Staff
ADD HireDate DATE;
```

Add a new attribute named ViewNo for the table Viewing

Assign ViewNo as the primary key of the table Viewing

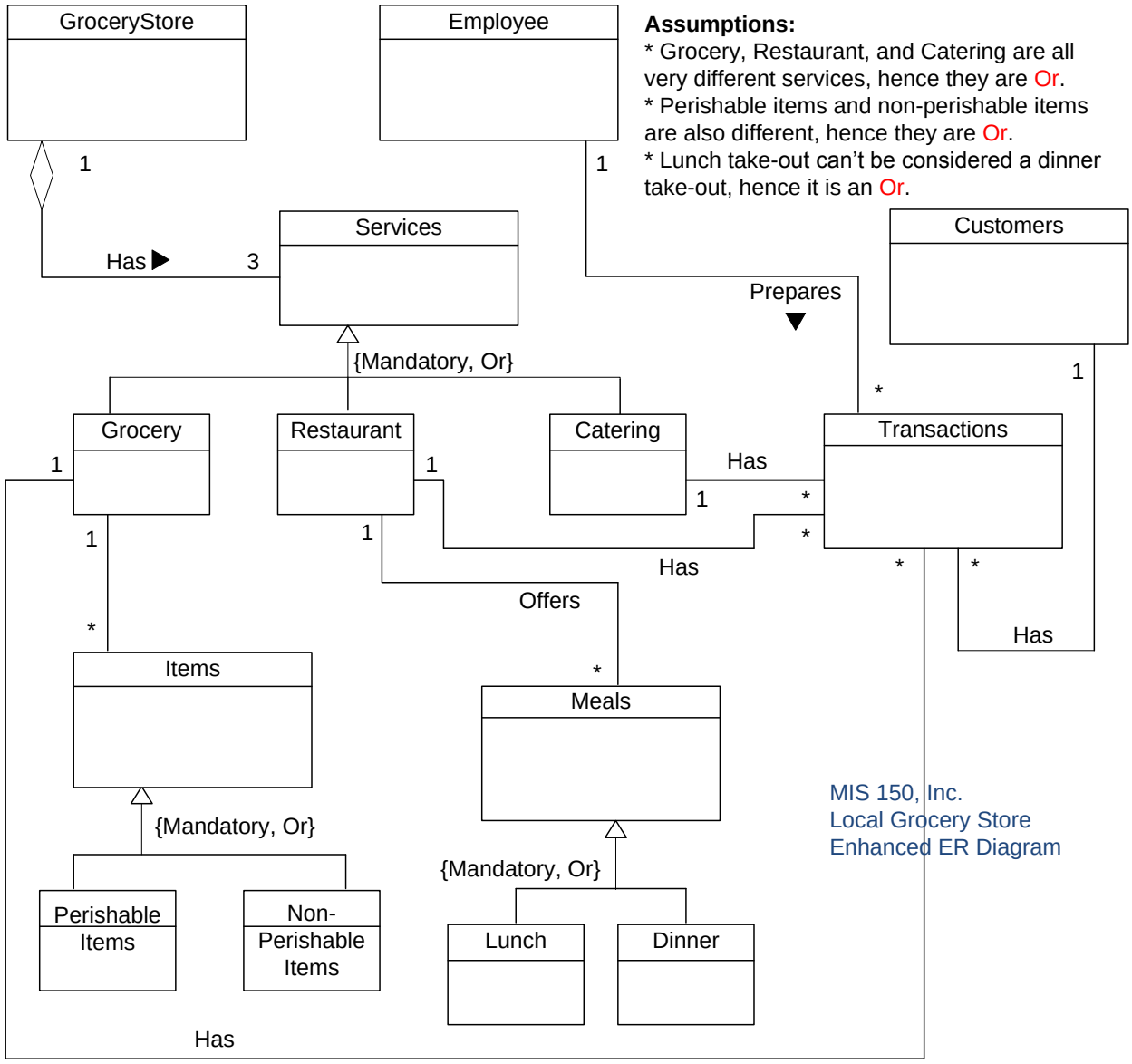
```
ALTER TABLE Viewing
ADD ViewNo NUMBER(6);

ALTER TABLE Viewing
ADD CONSTRAINT pkviewing PRIMARY KEY (ViewNo);
```

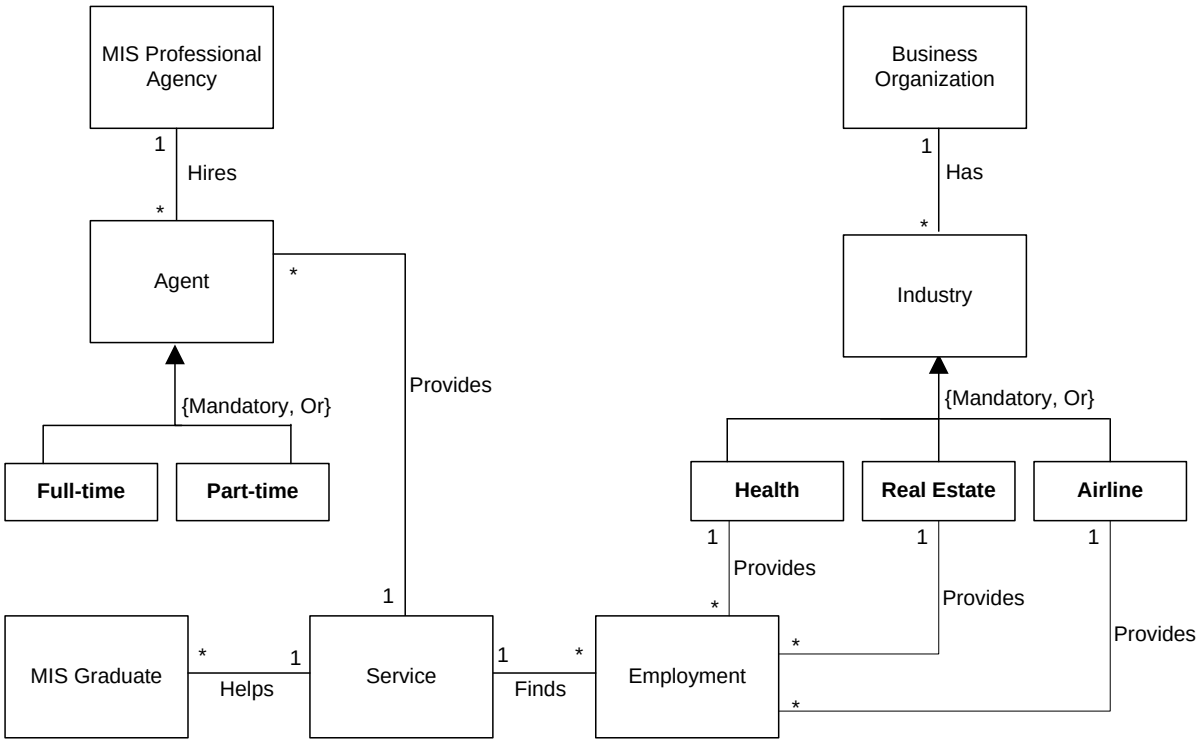
MIS 150, Exam 2 ER Diagram:

THE ULTIMATE COMPREHENSIVE GUIDE PART III

MIS150, Inc. is a local grocery store located in Sacramento City. The MIS150 has three divisions including catering, restaurant, and grocery. The catering division provides party services. The restaurant division offers lunch and dinner meals for takeout. The grocery division sells two types of merchandises including perishable items and non-perishable items. The MIS150 hires employee to help its customer getting the services. Develop an Enhanced Entity-Relationship diagram to document the database of the MIS150.

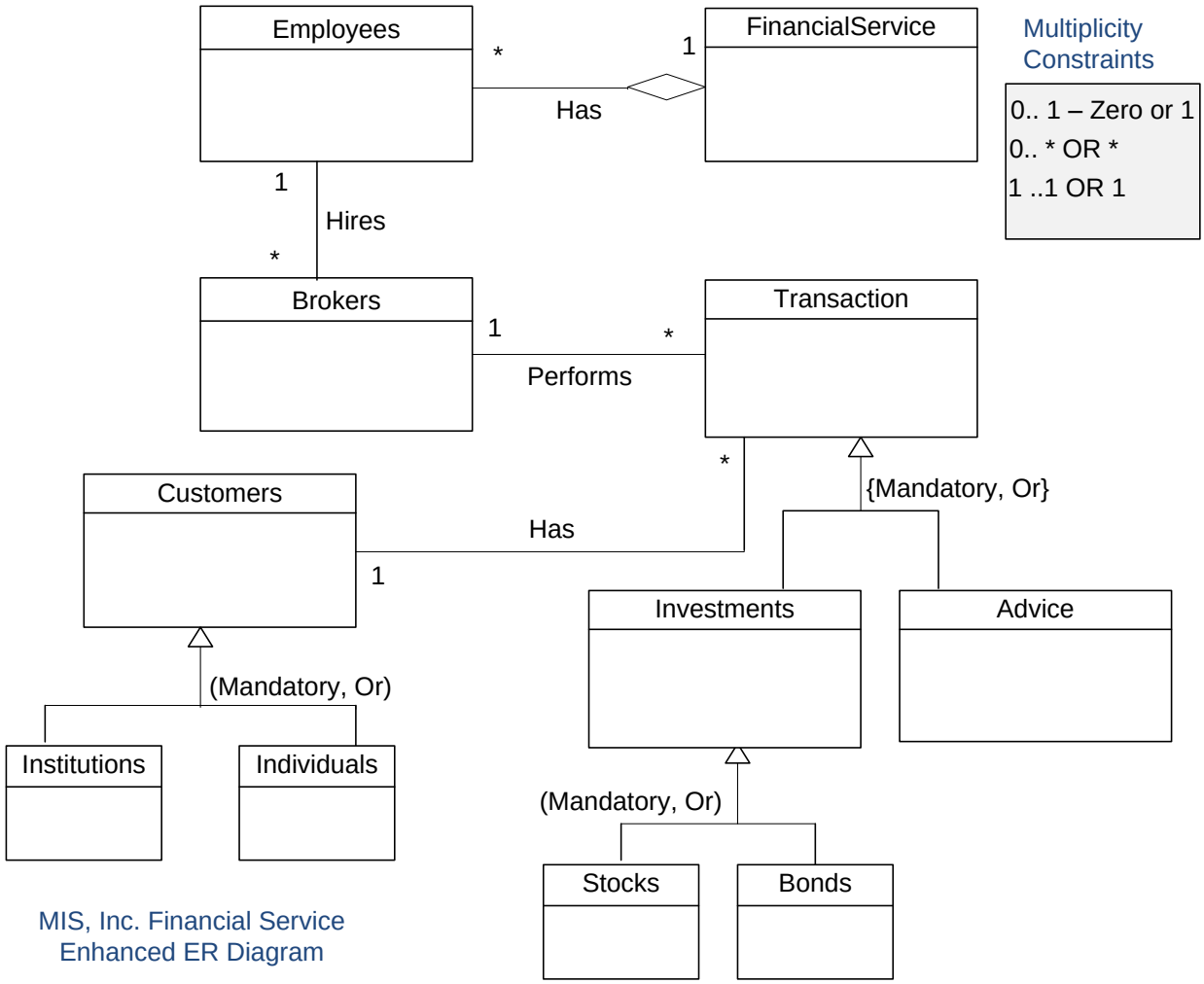


MIS Professional Agency is a non-profit organization. It has several full time and part time agents to help MIS graduates finding information technology employment from business organizations. There are three types of business organizations that are health, real estate, and airline. Develop an enhanced entity relationship diagram for the database needed by the MIS Professional Agency.



Assumptions:
* Service contains MIS graduate records such as applications, names, addresses, etc.
* Service also contains employment records such as employmentNo, hireDate, which industry does that particular employee goes.

MIS, Inc. is a medium size financial service company located in San Francisco. MIS offers selling/buying stocks and/or bonds for its customers. The MIS hires competent brokers not only to perform the selling/buying transactions for customers, but also to advice customers in terms of potential monetary benefits that could be generated by different types of stocks and bonds. The MIS has two different types of customers including institutions and individual.
Develop an Enhanced Entity-Relationship diagram to document the database of the MIS.

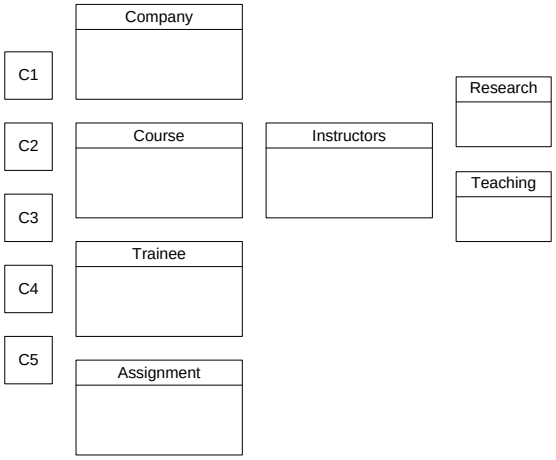


MIS, Inc. Financial Service
Enhanced ER Diagram

Assumptions:
* Investment and advice are different, hence it is an OR.
* Institutions is a public or private company and cannot be an individual.
* Stocks and bonds are different type of investments, hence it is an OR.

You are required to create a conceptual data model of the data requirements for a company that specialized in IT training. The company has 30 instructors and can handle up to 100 trainees per training session. The company offers five advanced technology courses, each of which is taught by a teaching team of two or more instructors. Each instructor is assigned to a maximum of two teaching teams or may be assigned to do research. Each trainee undertakes one advanced technology course per training session.

(a.) Identify the *main entity types* for the company.



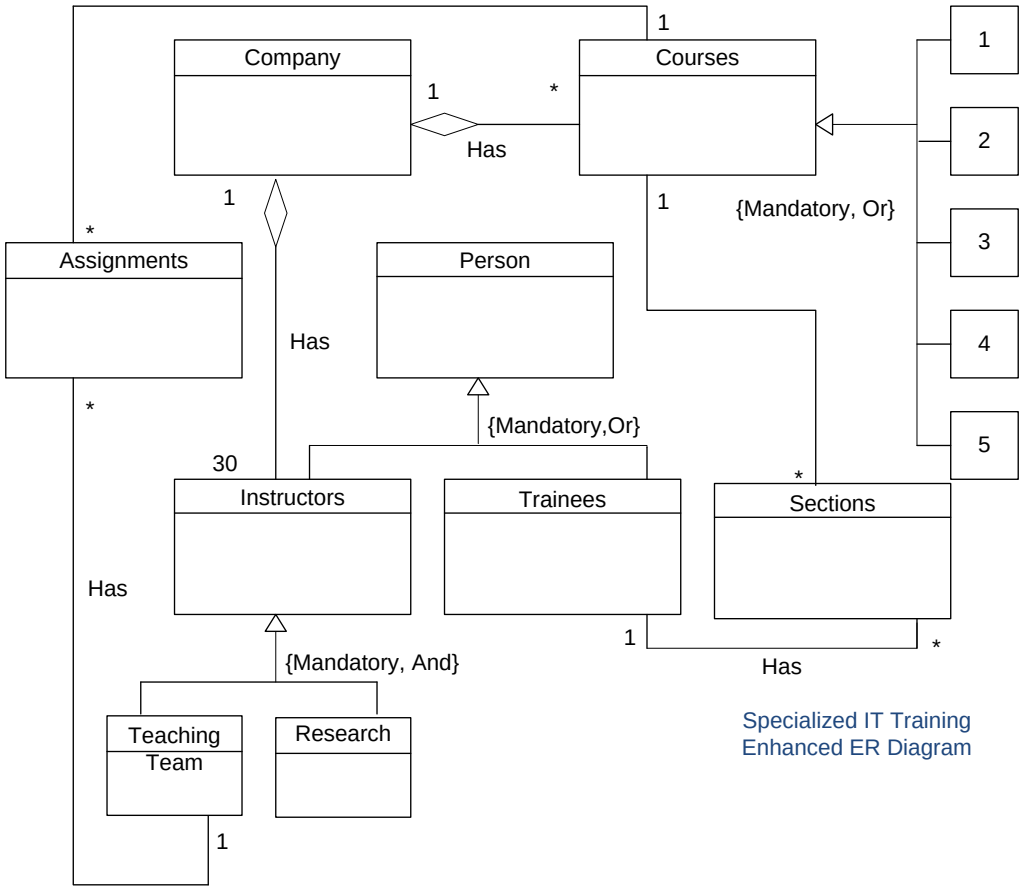
(b.) Identify the main relationship types and specify the multiplicity for each relationship. State any assumptions you make about the data.

- Company has Trainees (1..1, 0..*)
- Company offers Course (1..1, 1..*)
- Company has Instructor (1..1, 1..*)
- Course has Trainee (1..1, 0..*)
- Instructor – Teaching has Assignment (1..1, 0..*)
- Course has Assignment (1..1, 1..*)
- Instructor is Teaching or Researching (1..1, 1..*)
- Course C1, or, C2, or C3, or C4, or C5 (1..1, 0..*)

Assumptions:

* Trainers cannot be Instructors and vice-versa, hence it's Or.
* TeachingTeam can also be in a research team, hence it's And.
* Courses are different, hence it's Or.

(c.) Using your answers for (a) and (b), draw a single ER diagram to represent the data requirements for the company.



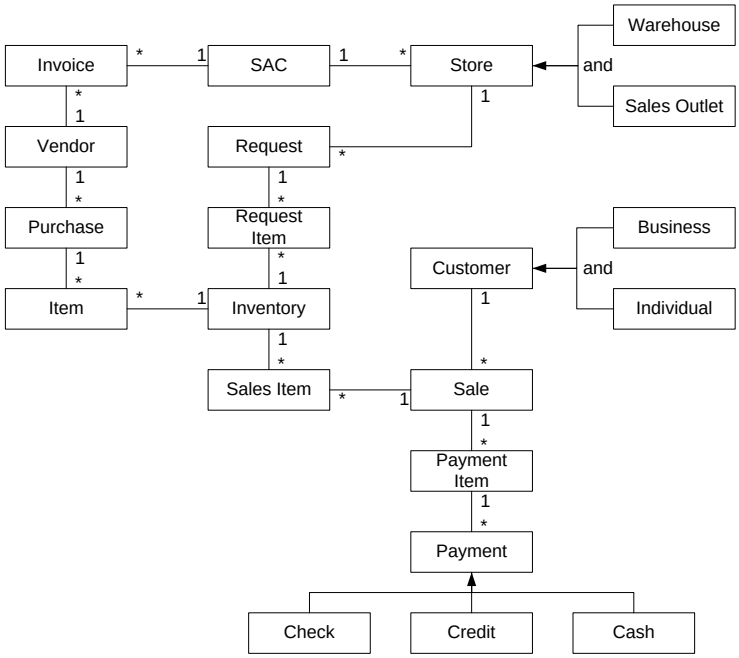
Assumptions:
* Trainers cannot be Instructors and vice-versa, hence it's Or.
* TeachingTeam can be also a research team, hence it's And.
* Courses are unique, hence it's Or.

The **SAC Winery** has **10 stores** and **two warehouses** located in the California State. SAC buys top quality California wines by issuing purchase orders to several famous California vineyards. The ordered wines are delivered to SAC with invoices requesting for payments from vineyards. SAC has established excellent credit with its wine suppliers. This allows SAC to accumulate vineyard's invoices and pay them every two months. On the other hand, SAC keeps a good size inventory either in the stores or in the warehouses to meet the seasonal demand from its individual and business customers. Warehouse will ship wines to individual store based on the store request. SAC has a sales policy that accepts cash, check, or credit as its sales payments. (Minus two points for each mistake)

) Identify and list every class (super class and subclass) for designing a database for the SAC Winery.

- SAC
- Invoice
- Vendor
- Purchase
- Item
- Inventory
- Request
- Request Item
- Store
  - Warehouse (subclass)
  - Sales Outlet (subclass)
- Customer
  - Business (subclass)
  - Individual (subclass)
- Sales Item
- Sale
- Payment Item
- Payment
  - Cash (subclass)
  - Check (subclass)
  - Credit(subclass)

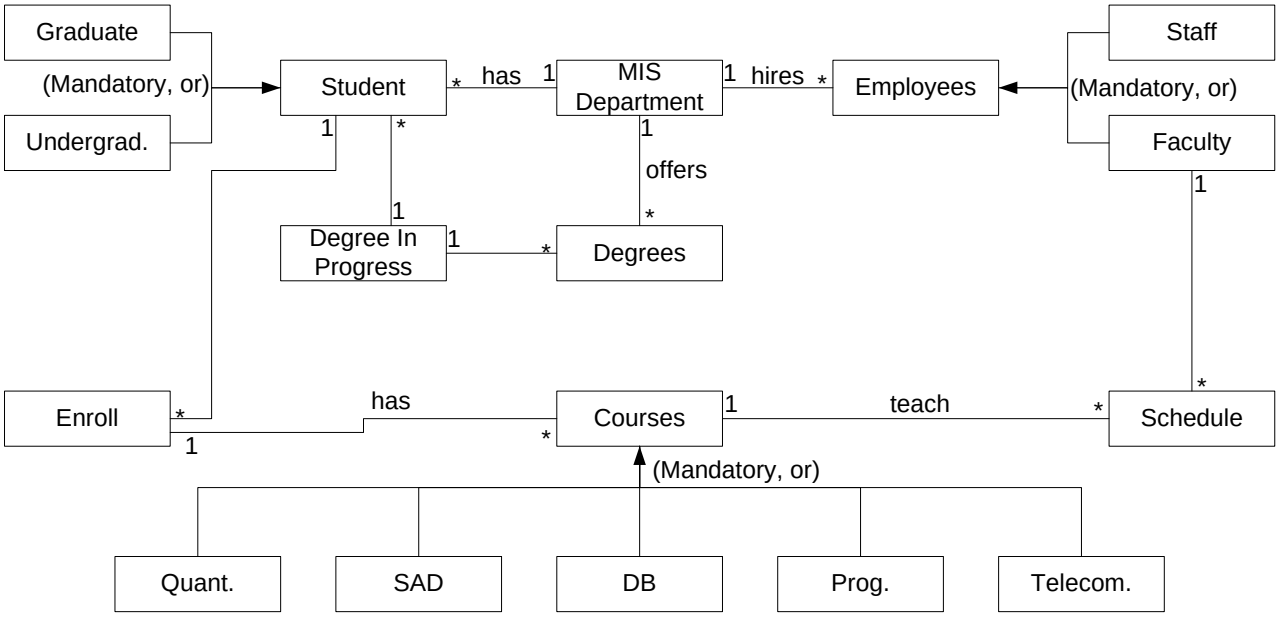
Develop an enhanced entity-relationship diagram for the database for the SAC Winery.



Management Information Science (MIS) Department offers graduate and undergraduate degrees to its graduate and undergraduate students. MIS Department hires two types of employees. The first type includes qualified faculty members to teach its courses. The second type comprises competent staff members for administration work. MIS Department offers five different types of courses that are quantitative, system analysis and design, database, programming, and telecommunication. Students have to enroll and complete all the required courses in order to earn their degrees.

(a) Identify and list every class (super class and subclass) for designing the database of MIS Department. (Minus 1 point for each mistake.)

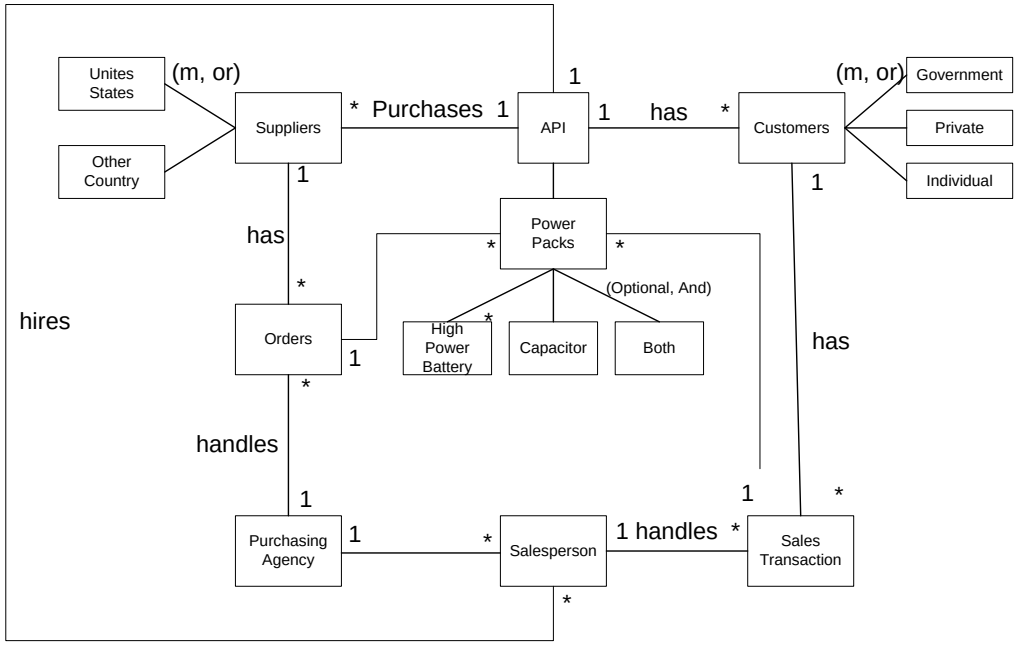
MIS
Student
Degree (Graduate, Undergraduate)
Employees
- Faculty → teach
- Staff → admin
Courses (System Analysis and Design, Database, Programming, Telecommunication)

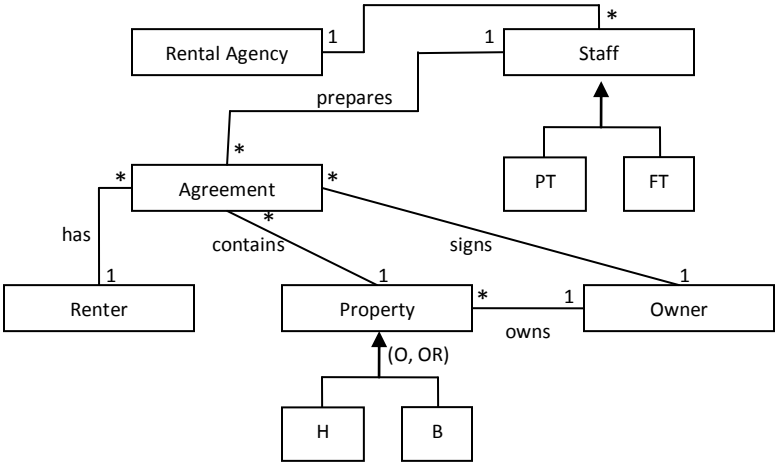


Apogee Power Incorporation (API) is a medium size power source company. It purchases different and high quality power packs from several suppliers to fill its customer's orders. Its suppliers are either located in the United States or in the other foreign countries. The customer base includes the private business, government agency, and individuals. The power packs can be the high power battery, capacitor, or both. API hires salesperson to handle the sales transactions for its customers. API also hires purchasing agency to manage the orders for its suppliers. Develop an Enhanced Entity-Relationship diagram for the database of API. (Minus 1 point for each mistake.)

Entity Types:

1. API
2. Power Packs – High Power, Capacitor, Both
3. Suppliers – United States, Other Country
4. Customers – Private, Government, Individuals
5. Salesperson
6. Sales Transactions
7. Purchasing Agency
8. Orders




Rental Agency
1
*
1
Staff
prepares
*
*
Agreement
*
PT
FT
has
*
contains
signs
1
1
1
Renter
Property
*
1
Owner
owns
(O, OR)
H
B