Assignment A4

Chapter8:

---

1 _____ is a relational set operator.

a. MINUS
b. ALL
c. EXISTS
d. PLUS

---

2 What is meant by "union-compatible"?

a. The number of attributes must be the same but the names and data types can be different.
b. The names of the relation attributes must be the same but the data types can be different.
c. The number of attributes must be the same and their data types must be alike.
d. The names of the relation attributes can be different, as well as the number of attributes, but data types must be similar.

---

3 Which data type is considered compatible with VARCHAR(35)?

a. CHAR(15)
b. DATE
c. TINYINT
d. INT

---

4 Which data type is compatible with NUMBER?

a. DATE
b. VARCHAR(15)
c. SMALLINT
d. CHAR(10)

---

5 The _____ statement combines rows from two or more queries without including duplicate rows.

a. UNION ALL
b. MINUS
c. UNION
d. INTERSECT

---

6 What is the syntax for the UNION statement?

a. UNION: query query
b. query UNION query
c. UNION(query, query)
d. query + query

---

7 Assume you are using the UNION statement to combine the results from two identical tables CUSTOMER and CUSTOMER_2. The CUSTOMER table contains 10 rows, while the CUSTOMER_2 table contains seven rows. There are two customers included in both tables. How many records are returned when using the UNION statement?

a. 7
b. 10
c. 15
d. 17

---

8 Assume you are using the UNION ALL statement to combine the results from two identical tables CUSTOMER and CUSTOMER_2. The CUSTOMER table contains 10 rows, while the CUSTOMER_2 table contains seven rows. There are two customers included in both tables. How many records are returned when using the UNION ALL statement?

a. 7
b. 10
c. 15
d. 17

---

9 A(n) _____ query can be used to produce a relation that retains duplicate rows.

a. INTERSECT
b. UNION
c. UNION ALL
d. MINUS

---

10 The _____ statement can be used to combine rows from two queries, returning only the rows that appear in both sets.

a. UNION
b. MINUS
c. UNION ALL
d. INTERSECT

---

11 The _____ statement in SQL combines rows from two queries and returns only the rows that appear in the first set but not in the second.

a. MINUS
b. UNION ALL
c. INTERSECT
d. UNION

---

12 What type of subquery could be used in place of INTERSECT if the RDBMS does not support it?

a. AND
b. UNION
c. OF
d. IN

---

13 What type of subquery could be used in place of MINUS if the RDBMS does not support it?

a. UNION
b. AND
c. IN
d. NOT IN

---

14 The following SQL statement uses a(n) _____.
SELECT P_CODE, P_DESCRIPT, P_PRICE, V_NAME
FROM PRODUCT, VENDOR
WHERE PRODUCT.V_CODE = VENDOR.V_CODE;

a. "old-style" join
b. procedural statement
c. set operator
d. natural join

---

15 The _____ join is the traditional join in which only rows that meet a given criteria are selected.

a. inner
b. set
c. full
d. outer

---

16 The statement SELECT * FROM T1, T2 produces a(n) _____ join.
a. natural
b. full
c. cross
d. equi-

---

17 A _____ join of two tables returns rows with matching values and includes all rows from both tables with unmatched values.

a. full outer
b. natural
c. cross
d. left outer

---

18 How many different types of outer joins exist?

a. 1
b. 2
c. 3
d. 4

---

19 How many rows would be returned from a cross join of tables A and B if A contains 10 rows and B contains 20?

a. 10
b. 20
c. 30
d. 200

---

20 A(n) _____ join will select only the rows with common values in the common column(s), excluding rows with unmatched values and duplicate columns.

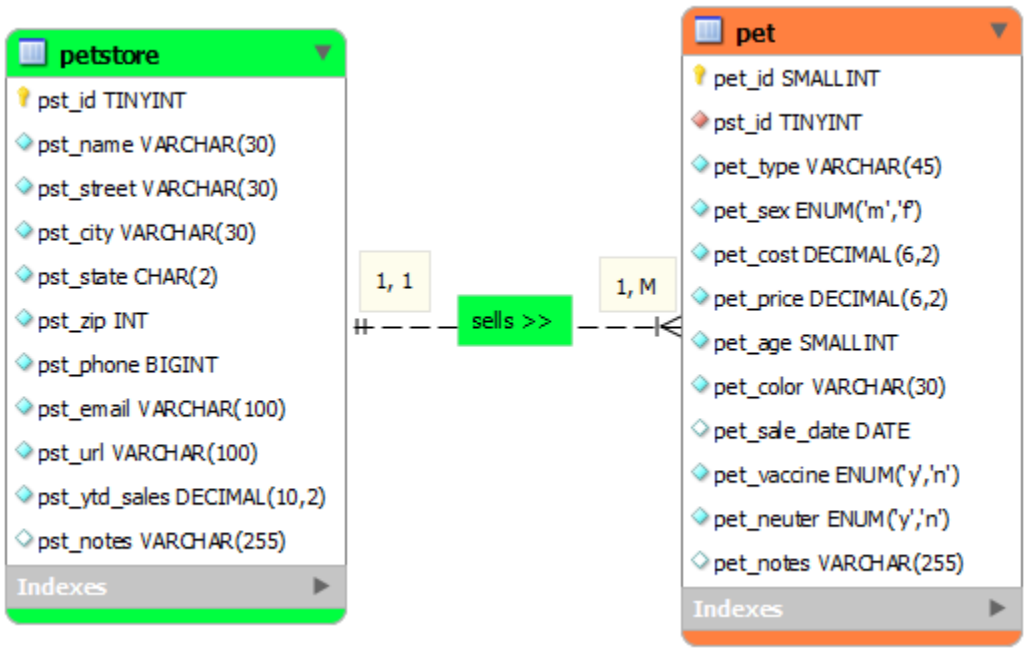
a. natural
b. cross
c. outer
d. full

Deliverables

Important!: Two DBMSs Employed!

1) MS SQL Server

T-SQL Statements for A4 (Using following ERD)



1. **Log into SQL Server:** Demo in class
2. Backward-engineer the above ERD using *only* **T-SQL**
   NOTES: Must include the following checks and defaults:
   petstore:
   - pst_state: default = AZ
   - pst_zip: > 0 and <= 999999999
   - pst_ytd_sales: > 0

   pet:
   - pet_sex: m or f
   - pet_cost: > 0
   - pet_price: > 0
   - pet_age: >0 and <= 10500
   - pet_vaccine: y or n
   - pet_neuter: y or n

   FK: *Must* require **ON DELETE CASCADE, ON UPDATE CASCADE**

3. Include **five *unique* records** for each table
4. Copy and paste query result sets with headers for **both tables** (select * from tablename;).

Resources:
Logging into MS SQL Server (using RemoteLabs):
http://www.qcitr.com/vids/MS_SQL_Server_Login.mp4
A4 Helper Video: http://www.qcitr.com/vids/LIS3784_A4_MS_SQL_Server.mp4

Important!: Two DBMSs Employed!

2) MySQL Server

Must include query result sets!
Joins *must* include all 4 types of Inner Joins (See Table 8.1)
For each SELECT statement

1. **Log into MySQL Server: Using PuTTY (PC) or Terminal (Mac). See Assignment 1 for uploading finance.sql to your db directory!**

2. Drop *ALL* existing tables in your database!

3. Upload the following file (see Assignment 1) to your db directory. Then type:
   source db/finance.sql
   --or...
   \. db/finance.sql

4. Perform the following steps:
   a. Display all tables
   b. Display each table structure
   c. Display all data for each table

   NOTE: All dollar amounts must be formatted to two decimal places, including a dollar sign ($). All phone numbers and zip codes must include proper hyphens (-).

5. Create SQL queries to generate the following reports, and display their associated query resultsets, copy from Secure Shell (PC) or Terminal (Mac):
   a. Create a **view** that only displays users' ids, first and last names, phone numbers and e-mail addresses, sort in ascending order by user last name (name it **v_user_info**):

   b. Create a **view** that only displays institutions' names, streets, cities, states, zip codes, phone numbers and url addresses, display all address info. under one header, sort in ascending order by institution name (name it **v_institution_info**):

   c. Create a **view** that only displays *unique* category types used (i.e., in transactions), not all of the category types available (name it **v_category_types**):

   d. Create a **stored procedure** that accepts a user's id, and displays a user's name, the institution names, and the account types to which that user is associated, sort in descending order by account type. (name it **UserInstitutionAccountInfo**):

   e. Create a **stored procedure** that only displays users' names, account types, and transactions for each account (include transaction type, method, amount, and date/time), (include date/time under one header) for each user, sort in descending order by user last name, ascending order of amount
   (name it **UserAccountTransactionInfo**):

   f. Create a **view** that only displays users' names and total of all debits, for each user, sort in descending order by total debits (name it **v_user_debits_info**):

   g. Create a **transaction** that inserts a record in the transaction table, updates it, and then removes it.

   Helper Videos:
   LIS3784 A4 MySQL1: https://youtu.be/i-gtYkfs198

   LIS3784 A4 MySQL2: https://youtu.be/w29h5Kzhx6Q