Database Management Systems

Review

Topics

- SQL Assignments
- Midterm exam questions
- Term projects

Assignment 4

- Assume the following table is given.

  Student: <StudentID, Name, Surname, Major, YearStarted>

- Create the table and define necessary constraints (primary key, null constraint, etc.)

Assignment 4

```
CREATE TABLE Student
{
    StudentID   integer PRIMARY KEY,
    Name        varchar(50) NOT NULL,
    Surname     varchar(50) NOT NULL,
    Major       varchar(50) NOT NULL,
    YearStarted  varchar(4)  NOT NULL
}
```

Assignment 4

- Insert at least 5 records into the table using **Insert** SQL command

INSERT INTO Student values (12345, 'John', 'Lewis' , 'Computer Science' , '2020')

INSERT INTO Student values (23456, 'David', 'Smith' , 'Computer Science' , '2020')

Assignment 4

- Select all students having a major in Computer Engineering, print their name and student ID

```
SELECT name, StudentID  FROM student
        where major = 'Computer Engineering'
```

Assignment 4

- Update start year of all Engineering students to 2015 (**hint:** use like clause)

```
UPDATE student set YearStarted = 2015
        where major like '%Engineering%'
```

Assignment 4

- Delete all students with starting year less than 2010

Delete from student where YearStarted < 2010

Assignment 4

- List all records and verify the results

  Select * from student

Assignment 4

- Drop the table

```
Drop Table Student
```

Assignment 5

- Assume three tables are given as follows:
  - **Student:** <StID, Name, Department>
  - **Course:** < CID, CName, Credit>
  - **Grading:** <StID, CID, Year, Grade >

Assignment 5

- Write SQL commands to create the tables. Consider necessary requirements such as primary key, foreign key, unique, not null, etc.

```
CREATE  TABLE Student
{
    StID integer PRIMARY KEY,
    Name  varchar(50) NOT NULL,
    Department varchar(50) NOT NULL
}
```

```
CREATE TABLE Course
  {  CID integer PRIMARY KEY,
     CName  varchar(50) NOT NULL,
     Credit integer NOT NULL
  }

CREATE TABLE Grading
  {  StID integer  References Student(StID),
     CID  integer  References Course(CID),
     Year char(4)   NOT NULL,
     Grade integer,
     PRIMARY KEY (StID, CID, Year)
  }
```

Assignment 5

- Insert some records

INSERT INTO Student Values (1234, 'John', 'Computer Science')

INSERT INTO Course Values (CS1234, 'Programming', 5)

INSERT INTO Grading (1234, CS1234, 2020, 85)

Assignment 5

- Find the number of students taking course CENG356 in 2020

```
Select count(*)
From Grading
WHERE CID='ceng356'and Year=2020
```

Assignment 5

- Find the average grade for CENG356 in 2020

```
SELECT avg(Grade)
FROM Grading
WHERE CID = 'CENG356' and year = 2020
```

Assignment 5

- Find the average grade for CENG356 during years 2000 and 2010 inclusive

```
SELECT avg(Grade)
FROM Grading
WHERE CID = 'CENG356' and year between 2000 and 2010
```

Assignment 5

- How many courses did a student named 'John' take in 2018?

```
SELECT count(*)
FROM Student Join Grading ON Student.StID = Grading.StID
and Year = 2018 and Student.Name='John'
```

Assignment 5

- What is the average grade of students in each course in 2018?

```
Select Avg(Grade), CName
FROM Grading Join Course on Grading.CID = Course.CID
GROUP By CName
```

Exam Questions

- Which operation in pile files can be carried out in a short time independently from the file size?
  - **Insertion**
  - Deletion
  - Fetch
  - Fetch Next

Exam Questions

- If a query is required to include non-repeated values, what keyword can be used?
  - Cascade
  - **Distinct**
  - Join
  - Intersect

Exam Questions

- Which one is the reason why linear indexing is not suitable for large data files
  - The linear index files have limit in their size
  - Linear indexes cannot refer to non-numeric key values
  - **Linear indexes should be in memory for fast search and for large files, linear index can be too large**
  - Linear index is used to store the key values and location of each record. Therefore, search in linear index files is slow.

Exam Questions

- A B+Tree index is fast in locating records because:
  - **The number of tree levels is small (tree has a shallow depth)**
  - The keys are not used in searching for records
  - The search in a B+Tree is based on the binary search
  - B+Tree needs less memory because it is more compact.

Exam Questions

- There may be collisions in creating a hash table because:
  - Hash tables use exhaustive search to find records
  - **Hash functions may return the same value for different key attributes**
  - Hash functions can be replaced with chaining algorithms
  - Hash tables cannot use bucketing when we create them

Exam Questions

- Which one is **NOT** a property of a view:
  - A view is a virtual table. It does not physically exist
  - A view is created by a query which may join one or more tables
  - A view can be a subset of a the records of table
  - **The data in a view does not change if we change the records in the related table(s).**

Exam Questions

- Which command(s) is used for restricting the number of output records in a SELECT statement
  - WHERE
  - AVG
  - HAVING
  - **Both WHERE and HAVING**

Exam Questions

- Which relational algebra operations are used to find the names of students taking database course?
  - **Select students taking database, then project on their names**
  - Project the names of the students taking database
  - Join student names and course name (database), then project on student name
  - Join student name, then project database

Exam Questions

- Which records will appear in the output if we run this query?
- SELECT Name from Table1 EXCEPT SELECT name from Table2
  - Every name in table1 and table2
  - Names in both table1 and table2
  - Names in table2 but not in table1
  - **Names in table1 but not in table2**

Exam Questions

- Which one is the difference between the equi join and the natural join?
  - Equi join does not consider null attributes but natural join does
  - Equi join uses only unequal values in attributes
  - Equi join combines attributes with the same name but natural join can combine attributes with different names
  - **Equi join combines attributes with different names but natural join can combine attributes with the same names**

Exam Questions

- A supermarket wants to store the details of its products on a table. The details include productID, product name, product category, and price. Create a table for this data. Include necessary constraints.

```
CREATE TABLE Product
{  Pid  integer PRIMARY KEY,
   pname varchar(128) NOT NULL unique,
   category varchar(128)  NOT NULL,
   price  decimal NOT NULL
}
```

Exam Questions

- A car manufacturing company wants to update the price of its automobiles. The table is as shown below:

  Automobile:<CarID, Model, Make, Price>

  The prices will be reduced by **10%** if the model is less than **2015**, and by **5%** otherwise. Write the necessary SQL query to update the records.

```
UPDATE Automobile Set Price=Price*0.95 WHERE Model >= 2015
UPDATE Automobile Set Price=Price*0.90 WHERE Model < 2015
```

Exam Questions

- A university instructor offers a few courses each year. They use the table below to store the grades.

CourseGrades: < StudentID, CourseCode, Year, grade>

The instructor wants to find the minimum, the maximum, and the average grade in each course in the year 2020. Write the necessary SQL query to find the required data.

Exam Questions

- Select CourseCode, min(grade), max(grade), avg(grade)
  From CourseGrades
  Where Year=2020
  Group by CourseCode

Exam Questions

- Assume in a company an employee table is used to store employee data, as given below:

  Employee:< EmpID, Name, AssignedTask, Salary>

  Write a SQL query to find all employees who are paid less than the average salary in this company and print their names and assigned tasks.

Exam Questions

```
SELECT Name, AssignedTask
From Employee
where Salary < (Select Avg(salary) From Employee )
```

Term Projects

- Follow the steps shown below in your term project:
- Choose your teammate (groups of two at most)
- Choose a title, e.g. Library Database
- Send your team members, and project title to me by email.

Term Projects

- Create a list of **data items** that will be stored in your project. For example, in library project, you may ask the library. Items need not be put in any order.
- Find and list all **entities**.
- Find the **attributes** of entities (the attributes are data items that you already found)
- Define **restrictions** for each attribute (e.g. Number of digits in student ID, range of values for Age attribute, …)

Term Projects

- Define **relationships** between entities and their types (Explain how you found the type of the relationships)
- Create **Entity-Relationship (ER)** model of your database.
- Create necessary **tables** for your database.
- Design and write necessary **queries** (example: find a book in a library, or find a product in a store).

Term Projects

- Write necessary SQL commands to create tables and queries
- Add some data to your database to test your queries
- Write a report for your project (put all steps explained above)
- Implement your project using a DBMS (use **MySQL**)

Term Projects

Evaluation

- Gathering data items (10)
- Defining entities, the attributes of the entities, the restrictions on the attributes (15)
- Relationships (10)
- ER model (10)
- Queries (15)
- Implementation (15)
- Report (10)
- Presentation (15) (is necessary)

Presentations

- Will be during the last week of the semester on December 8
- The presentation schedule will be posted on BB.

Questions?