IT420: Database Management and Organization

SQL - SELECT
Chapter 2

Class Exercise
- Division(Name, Building, OfficeNb)
- Department(DeptName, ChairName, WebAddress, DivName)
- Create tables
- Modify Department to add a FK constraint for DivName
- Create table Colleges with same structure as Division
- Insert everything from Division into Colleges
- Remove Division table
- Find the name of the Chair of the ‘Math’ Department

Last Time vs. Today
- Last Time:
  - CREATE, DROP, ALTER
  - INSERT, DELETE, UPDATE
  - SELECT
- Today: More about SELECT
  - Joins, sub-queries
  - Sorting, wild cards
  - Arithmetic operations, aggregates
  - Groups

The SQL SELECT Statement
- Basic SQL Query:
  SELECT [DISTINCT] column_name(s) | *
  FROM table_name(s)
  [WHERE conditions]

SELECT from Two or More Tables
Find the names of students enrolled in IT420

SELECT SName
FROM Students S, Enrolled E
WHERE S.Snb = E.SNb AND E.Cid = 'IT420'

SELECT - Conceptual Evaluation Strategy
- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - Compute the cross-product of table_names
  - Discard resulting rows if they fail condition
  - Delete columns that are not in column_names
  - If DISTINCT is specified, eliminate duplicate rows
  - This strategy is probably the least efficient way to compute a query!
    - An optimizer will find more efficient strategies to compute the same answers.

Students
<table>
<thead>
<tr>
<th>Snb</th>
<th>SName</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>190</td>
<td>Smith</td>
<td><a href="mailto:jsmith@usna.edu">jsmith@usna.edu</a></td>
</tr>
<tr>
<td>673</td>
<td>Doe</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
</tr>
<tr>
<td>312</td>
<td>Snie</td>
<td><a href="mailto:jsnell@usna.edu">jsnell@usna.edu</a></td>
</tr>
</tbody>
</table>

Enrolled
<table>
<thead>
<tr>
<th>Snb</th>
<th>Cid</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>190</td>
<td>IT420</td>
<td>Spring2006</td>
</tr>
<tr>
<td>312</td>
<td>IT430</td>
<td>Fall2005</td>
</tr>
</tbody>
</table>
Example Conceptual Evaluation

SELECT SName FROM Students S, Enrolled E WHERE S.Snb = E.SNb AND E.Cid = 'IT420'

<table>
<thead>
<tr>
<th>S.Snb</th>
<th>SName</th>
<th>Email</th>
<th>E.SNb</th>
<th>Cid</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>190</td>
<td>Smith</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>190</td>
<td>IT340</td>
<td>Spring2006</td>
</tr>
<tr>
<td>190</td>
<td>Smith</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>312</td>
<td>IT420</td>
<td>Fall2005</td>
</tr>
<tr>
<td>673</td>
<td>Doe</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>190</td>
<td>IT340</td>
<td>Spring2006</td>
</tr>
<tr>
<td>673</td>
<td>Doe</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>312</td>
<td>IT420</td>
<td>Fall2005</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>190</td>
<td>IT340</td>
<td>Spring2006</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>312</td>
<td>IT420</td>
<td>Fall2005</td>
</tr>
</tbody>
</table>

Example Conceptual Evaluation

SELECT SName FROM Students S, Enrolled E WHERE S.Snb = E.SNb AND E.Cid = 'IT420'

<table>
<thead>
<tr>
<th>S.Snb</th>
<th>SName</th>
<th>Email</th>
<th>E.SNb</th>
<th>Cid</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>190</td>
<td>Smith</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>190</td>
<td>IT340</td>
<td>Spring2006</td>
</tr>
<tr>
<td>190</td>
<td>Smith</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>312</td>
<td>IT420</td>
<td>Fall2005</td>
</tr>
<tr>
<td>673</td>
<td>Doe</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>190</td>
<td>IT340</td>
<td>Spring2006</td>
</tr>
<tr>
<td>673</td>
<td>Doe</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>312</td>
<td>IT420</td>
<td>Fall2005</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>190</td>
<td>IT340</td>
<td>Spring2006</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td>312</td>
<td>IT420</td>
<td>Fall2005</td>
</tr>
</tbody>
</table>

Example Conceptual Evaluation

SELECT SName FROM Students S, Enrolled E WHERE S.Snb = E.SNb AND E.Cid = 'IT420'

How would the result be different with DISTINCT?

Modified Query

SELECT SNb FROM Students S, Enrolled E WHERE S.Snb = E.SNb AND E.Cid = 'IT420'

Class Exercise

- Students(SNb, SName, Email)
- Courses(Cid,CName, Dept)
- Enrolled(SNb,Cid, Semester)

- Find the student number and name for each student enrolled in ‘Spring2007’ semester
- Find the names of all students enrolled in ‘ComSci’ courses

Sorting the Results

SELECT (DISTINCT column_name(s)) * FROM table_name(s) [WHERE conditions] [ORDER BY column_name(s) [ASC/DESC]]

Example:

Students(SNb, SName, Email, Major)

SELECT SNb, SName FROM Students ORDER BY SName ASC, SNb DESC
WHERE Clause Options

- **AND, OR**
- **IN, NOT IN, BETWEEN**
- **LIKE**

  Wild cards:
  - SQL-92 Standard (SQL Server, Oracle, etc.):
    - _ = Exactly one character
    - % = Any set of one or more characters
  - MS Access
    - ? = Exactly one character
    - * = Any set of one or more characters

  Example:
  Students(SNb, SName, Email, Major)
  Find alpha and name of SCS or SIT students with SNb starting with '8'

```
SELECT SNb, SName
FROM Students
WHERE SNb LIKE '8%' AND
      Major IN ('SIT', 'SCS')
```

Calculations in SQL

- **Simple arithmetic**
- **Five SQL Built-in Functions:**
  - COUNT
  - SUM
  - AVG
  - MIN
  - MAX

**Simple Arithmetic**

```
SELECT NbHours* HourlyRate AS Charge
FROM FlightEvents
```

```
SELECT SFirstName + ' ' + SLastName FROM Students
```

**Aggregate Operators**

- **SELECT COUNT(*) FROM Students**
- **SELECT COUNT(DISTINCT SName) FROM Students WHERE SNb > 700**
- **SELECT AVG(Age) FROM Students WHERE SNb LIKE '08____'**

**Aggregate Operators Limitations**

- Return only one row
- Not in WHERE clause

**Select oldest students and their age**

```
SELECT SName, MAX(Age)
FROM Students S
```

```
SELECT SName, S.Age
FROM Students S
WHERE S.Age = (SELECT MAX(Age) FROM Students)
```

```
SELECT SName, MAX(Age) FROM Students S
```

```
SELECT SName, S.Age
FROM Students
WHERE S.Age = (SELECT MAX(Age) FROM Students)
```

```Sub-query```

```Correct!```
Select students with age higher than average

- Correct!
  ```sql
  SELECT *
  FROM Students
  WHERE Age > (SELECT AVG(Age) FROM Students)
  ```

- Illegal!

Class Exercise

- Students(SNb, SName, Email)
- Courses(Cid, CName, Dept)
- Enrolled(SNb, Cid, Semester)

- List SNb of all students enrolled in 'IT420' or 'IT340', ordered by SNb

Grouping rows

- Find the age of the youngest student for each class year

- Correct!
  ```sql
  SELECT MIN (S.Age)
  FROM Students S
  WHERE S.ClassYear = 2007
  ```

- Example:
  ```sql
  SELECT ClassYear, MIN(Age)
  FROM Students
  GROUP BY ClassYear
  ```

<table>
<thead>
<tr>
<th>ClassYear</th>
<th>(no column name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>21</td>
</tr>
<tr>
<td>2010</td>
<td>17</td>
</tr>
<tr>
<td>2009</td>
<td>18</td>
</tr>
<tr>
<td>2008</td>
<td>20</td>
</tr>
</tbody>
</table>

GROUP-BY Clause

- SELECT [DISTINCT] column_name(s) | aggregate_expr
  FROM table_name(s)
  [WHERE conditions]
  GROUP BY grouping_columns

- Example:
  ```sql
  SELECT ClassYear, MIN(Age)
  FROM Students
  GROUP BY ClassYear
  ```

<table>
<thead>
<tr>
<th>ClassYear</th>
<th>(no column name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>21</td>
</tr>
<tr>
<td>2010</td>
<td>17</td>
</tr>
<tr>
<td>2009</td>
<td>18</td>
</tr>
<tr>
<td>2008</td>
<td>20</td>
</tr>
</tbody>
</table>

Conceptual Evaluation

- Semantics of an SQL query defined as follows:
  - Compute the cross-product of tables in FROM (table_names)
  - Discard resulting rows if they fail WHERE conditions
  - Delete columns that are not in SELECT or GROUP BY(column_names or grouping-columns)
  - Remaining rows are partitioned into groups by the value of the columns in grouping-columns
  - One answer row is generated per group
  - Note: Does not imply query will actually be evaluated this way!

HAVING Clause

- SELECT [DISTINCT] column_name(s) | aggregate_expr
  FROM table_name(s)
  [WHERE conditions]
  GROUP BY grouping_columns
  HAVING group_conditions

- GROUP BY groups the rows
- HAVING restricts the groups presented in the result
Example - HAVING

- SELECT ClassYear, MIN(Age)
  FROM Students
  WHERE MajDeptName = 'ComSci'
  GROUP BY ClassYear
  HAVING COUNT(*) > 20

Conceptual Evaluation

- SQL query semantics:
  - Compute the cross-product of table_names
  - Discard resulting rows if they fail conditions
  - Delete columns that are not specified in SELECT, GROUP BY
  - Remaining rows are partitioned into groups by the value of the columns in grouping-columns
  - One answer row is generated per group
  - Discard resulting groups that do not satisfy group_conditions

Example

- SELECT Class, MIN(Age)
  FROM Students
  WHERE MajDeptName = 'ComSci'
  GROUP BY Class
  HAVING COUNT(*) > 2

Class Exercise

- Students(SNb, SName, Email)
- Courses(Cid, CName, Dept)
- Enrolled(SNb, Cid, Semester)

List all course names, and the number of students enrolled in the course

Subqueries

- SELECT *
  FROM Students
  WHERE Age > (SELECT AVG(Age)
  FROM Students)

- Second select is a subquery (or nested query)
- You can have subqueries in FROM or HAVING clause also

Subqueries in FROM Clause

- Find name of students enrolled in both 'IT420' and 'IT334'
  - SELECT FName + ' ' + LName AS StudentName
    FROM Students, (SELECT Alpha
    FROM Enroll
    WHERE CourseID = 'IT420'
    AND Alpha IN
    (SELECT Alpha
    FROM Enroll
    WHERE CourseID = 'IT334')) AS ResultAlphaTable
    WHERE Students.Alpha = ResultAlphaTable.Alpha
Subqueries Exercise

- Students(Alpha, LName, FName, Class, Age)
- Enroll(Alpha, CourseID, Semester, Grade)

1. Find alpha for students enrolled in both ‘IT420’ and ‘IT334’
2. Find name of students enrolled in both ‘IT420’ and ‘IT334’

Class Exercise

- Students(Alpha, LName, FName, Class, Age)
- Enroll(Alpha, CourseID, Semester, Grade)

- Find the name of students enrolled in ‘IT420’
  - Usual way
  - Use subqueries

Class Exercise

What does this query compute:

- SELECT FName, LName
  FROM Students S, Enroll E1, Enroll E2
  WHERE S.Alpha = E1.Alpha
    AND S.Alpha = E2.Alpha
    AND E1.CourseID = 'IT420'
    AND E2.CourseID = 'IT344'

Summary

- SELECT [DISTINCT] column_name(s) / aggregate_expr
  FROM table_name(s)
  WHERE conditions
  GROUP BY grouping_columns
  HAVING group_conditions
  ORDER BY column_name(s) [ASC/DESC]