SQL: Structured Query Language (Chapter 2)

SQL - The Language of Databases

- Developed by IBM in the 1970s
- Create and process database data
- SQL programming is a critical skill !!!!
Facebook and Databases

- Relational databases are accessed in much the same way across the board: SQL. **Learning how SQL works is crucial** to getting anything done in databases, and any GUI is largely a wrapper around the SQL statements one uses to make those actions happen.
- Knowing a little about database design (layout, B-trees, file storage, normalization) is good, mostly for helping you understand good queries.
- We run the LAMP stack here, so we primarily use MySQL databases across the site.
- I hope this helps a little. Another good motivation may be found in the requirements for most engineering positions here on [http://www.facebook.com](http://www.facebook.com)

Thanks!
Nick from Facebook

Relational Query Languages

- A major strength of the relational model:
  - support simple, powerful querying of data
- Ad-hoc queries
- High-level (declarative) languages
  - Queries can be written intuitively
  - DBMS is responsible for efficient evaluation.
SQL DDL and DML

- SQL statements can be divided into two categories:
  - **Data definition language (DDL)** statements
    - Used for creating and modifying tables, views, and other structures
    - CREATE, DROP, ALTER
  - **Data manipulation language (DML)** statements.
    - Used for queries and data modification
    - INSERT, DELETE, UPDATE, SELECT

The SQL SELECT Statement

- Basic SQL Query:

  ```sql
  SELECT   [DISTINCT] column_name(s) | *
  FROM      table_name(s)
  [WHERE    conditions]
  ```
Selecting All Columns: The Asterisk (*) Keyword

```
SELECT *
FROM Students;
```

<table>
<thead>
<tr>
<th>StudentNumber</th>
<th>StudentLastName</th>
<th>StudentFirstName</th>
<th>Email</th>
<th>PhoneNumber</th>
<th>MajDeptName</th>
</tr>
</thead>
<tbody>
<tr>
<td>190</td>
<td>Smith</td>
<td>John</td>
<td><a href="mailto:jsmith@usna.edu">jsmith@usna.edu</a></td>
<td>410-431-3456</td>
<td>ComSci</td>
</tr>
<tr>
<td>673</td>
<td>Doe</td>
<td>Jane</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
<td></td>
<td>ComSci</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td>Jane</td>
<td><a href="mailto:jdoe2@usna.edu">jdoe2@usna.edu</a></td>
<td>443-451-7865</td>
<td>Math</td>
</tr>
</tbody>
</table>

Specific Columns and Rows from One Table

```
SELECT StudentNumber, StudentLastName, StudentFirstName
FROM Students
WHERE MajDeptName = 'ComSci';
```

<table>
<thead>
<tr>
<th>StudentNumber</th>
<th>StudentLastName</th>
<th>StudentFirstName</th>
</tr>
</thead>
<tbody>
<tr>
<td>190</td>
<td>Smith</td>
<td>John</td>
</tr>
<tr>
<td>673</td>
<td>Doe</td>
<td>Jane</td>
</tr>
</tbody>
</table>
The DISTINCT Keyword

```
SELECT SName
FROM Students;
```

```
SELECT DISTINCT SName
FROM Students;
```

Class Exercise

- Department(`DeptName`, ChairName, WebAddress, DivName)
- Find the name of the Chair of the ‘Math’ Department
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

SELECT from Two or More Tables

Find the names of students enrolled in IT360

SELECT SName FROM Students S, Enrolled E WHERE S.Snb = E.SNb AND E.Cid = 'IT360'

<table>
<thead>
<tr>
<th>Students</th>
<th>Courses</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNb</td>
<td>SName</td>
</tr>
<tr>
<td>190</td>
<td>Smith</td>
</tr>
<tr>
<td>673</td>
<td>Doe</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enrolled</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNb</td>
<td>Cid</td>
</tr>
<tr>
<td>190</td>
<td>IC322</td>
</tr>
<tr>
<td>312</td>
<td>IT360</td>
</tr>
</tbody>
</table>
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.

Example Conceptual Evaluation

```
SELECT SName
FROM Students S, Enrolled E
WHERE S.Snb = E.SNb AND E.Cid = 'IT360'
```

<table>
<thead>
<tr>
<th>S.Nb</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:jsmith@usna.edu">jsmith@usna.edu</a></td>
<td>190</td>
<td>IC322</td>
<td>Spring2011</td>
</tr>
<tr>
<td>190</td>
<td>Smith</td>
<td><a href="mailto:jsmith@usna.edu">jsmith@usna.edu</a></td>
<td>312</td>
<td>IT360</td>
<td>Spring2012</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>IC322</td>
<td>Spring2011</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>IT360</td>
<td>Spring2012</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td><a href="mailto:jdoe2@usna.edu">jdoe2@usna.edu</a></td>
<td>190</td>
<td>IC322</td>
<td>Spring2011</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td><a href="mailto:jdoe2@usna.edu">jdoe2@usna.edu</a></td>
<td>312</td>
<td>IT360</td>
<td>Spring2012</td>
</tr>
</tbody>
</table>
Example Conceptual Evaluation

```
SELECT SName
FROM Students S, Enrolled E
WHERE S.Snb = E.SNb AND E.Cid = 'IT360'
```

<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:jsmith@usna.edu">jsmith@usna.edu</a></td>
<td>190</td>
<td>IC322</td>
<td>Spring2011</td>
</tr>
<tr>
<td>190</td>
<td>Smith</td>
<td><a href="mailto:jsmith@usna.edu">jsmith@usna.edu</a></td>
<td>312</td>
<td>IT360</td>
<td>Spring2012</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>IC322</td>
<td>Spring2011</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>IT360</td>
<td>Spring2012</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td><a href="mailto:jdoe2@usna.edu">jdoe2@usna.edu</a></td>
<td>190</td>
<td>IC322</td>
<td>Spring2011</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td><a href="mailto:jdoe2@usna.edu">jdoe2@usna.edu</a></td>
<td>312</td>
<td>IT360</td>
<td>Spring2012</td>
</tr>
</tbody>
</table>

Example Conceptual Evaluation

```
SELECT SName
FROM Students S, Enrolled E
WHERE S.Snb = E.SNb AND E.Cid = 'IT360'
```

<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:jsmith@usna.edu">jsmith@usna.edu</a></td>
<td>190</td>
<td>IC322</td>
<td>Spring2011</td>
</tr>
<tr>
<td>190</td>
<td>Smith</td>
<td><a href="mailto:jsmith@usna.edu">jsmith@usna.edu</a></td>
<td>312</td>
<td>IT360</td>
<td>Spring2012</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>IC322</td>
<td>Spring2011</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>IT360</td>
<td>Spring2012</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td><a href="mailto:jdoe2@usna.edu">jdoe2@usna.edu</a></td>
<td>190</td>
<td>IC322</td>
<td>Spring2011</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td><a href="mailto:jdoe2@usna.edu">jdoe2@usna.edu</a></td>
<td>312</td>
<td>IT360</td>
<td>Spring2012</td>
</tr>
</tbody>
</table>
Modified Query

SELECT SNb
FROM Students S, Enrolled E
WHERE S.Snb = E.SNb AND E.Cid = 'IT360'

- Would the result be different with DISTINCT?

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 ‘Spring2011’ semester
- Find the names of all students enrolled in ‘ComSci’ courses
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 ‘12’

SELECT SNb, SName
FROM Students
WHERE SNb LIKE ‘12%’ 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 '12____'
Aggregate Operators Limitations

- Return only one row
- Not in WHERE clause

Select oldest students and their age

- SELECT S.SName, MAX (Age) FROM Students S
  - Illegal!

- SELECT S.SName, S.Age FROM Students S
  WHERE S.AGE = (SELECT MAX(Age) FROM Students)
  - Correct!

Sub-query
Select students with age higher than average

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

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

Class Exercise

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

- List SNb of all students enrolled in ‘IT360’ or ‘IC322’, ordered by SNb
Grouping rows

- **Find the age of the youngest student for each class year**
- SELECT MIN (S.Age)
  FROM Students S
  WHERE S.ClassYear = 2012

<table>
<thead>
<tr>
<th>(no column name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</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:
  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>2014</td>
<td>21</td>
</tr>
<tr>
<td>2012</td>
<td>17</td>
</tr>
<tr>
<td>2011</td>
<td>18</td>
</tr>
<tr>
<td>2013</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 ‘IT360’ and ‘IT334’

- SELECT FName + ' ' + LName AS StudentName
  FROM Students, (SELECT Alpha
                   FROM Enroll
                   WHERE CourseID = 'IT360'
                   AND Alpha IN
                   (SELECT Alpha
                    FROM Enroll
                    WHERE CourseID = 'IT334')
               ) AS ResultAlphaTable
  WHERE Students.Alpha = ResultAlphaTable.Alpha
Subqueries Exercise

- Students(\textit{Alpha}, LName, FName, Class, Age)
- Enroll(\textit{Alpha}, CourseID, Semester, Grade)

1. Find alpha for students enrolled in \textbf{both} ‘IT360’ and ‘IT334’
2. Find name of students enrolled in \textbf{both} ‘IT360’ and ‘IT334’

Class Exercise

- Students(\textit{Alpha}, LName, FName, Class, Age)
- Enroll(\textit{Alpha}, CourseID, Semester, Grade)

- Find the name of students enrolled in ‘IT360’
  - 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 = 'IT360'
      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]