IT360: Applied Database Systems

SQL: Structured Query Language
(Chapter 4)

Goals

- SQL: Data Definition Language
  - CREATE
  - ALTER
  - DROP
- SQL: Data Manipulation Language
  - INSERT
  - DELETE
  - UPDATE
  - SELECT
Relational Query Languages

- A major strength of the relational model:
  - supports simple, powerful querying of data
- Queries can be written intuitively, and the 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
Creating Tables

CREATE TABLE `table_name`
(`column_name1` `column_type1` [constraints1],
 ....
[[CONSTRAINT `constraint_name` `table_constraint`]
)

Table constraints:
- NULL/NOT NULL
- PRIMARY KEY (columns)
- UNIQUE (columns)
- CHECK (conditions)
- FOREIGN KEY (local columns) REFERENCES foreign_table
  (foreign_columns) [ON DELETE action_d ON UPDATE action_u]

Specify surrogate key in SQL Server:
`column_name` `int_type` IDENTITY (seed, increment)

Specify surrogate key in MySQL:
`column_name` `int_type` AUTO_INCREMENT

CREATE TABLE Example
CREATE TABLE Students
(`StudentNumber` integer NOT NULL,
`StudentLastName` varchar(18) NOT NULL,
`StudentFirstName` varchar(18) NOT NULL,
Email varchar(50),
`PhoneNumber` char(18),
`MajorDepartmentName` char(18),

CONSTRAINT PK_Students PRIMARY KEY (`StudentNumber`),
CONSTRAINT U_Email UNIQUE (Email),
CONSTRAINT FK_Dept FOREIGN KEY(`MajorDepartmentName`) REFERENCES DEPARTMENTS(`DepartmentName`) ON DELETE NO ACTION ON UPDATE CASCADE
)
### FOREIGN KEY Constraints

**DEPARTMENTS**

- **DepartmentName**: char(18)
- **Phone**: char(18)
- **Building**: char(18)
- **Room**: integer

**STUDENTS**

- **StudentNumber**: integer
- **StudentLastName**: char(18)
- **StudentFirstName**: char(18)
- **Email**: varchar(50)
- **PhoneNumber**: char(18)
- **MajorDepartmentName**: char(18)

---

**CREATE TABLE Departments**

```sql
CREATE TABLE Departments
(DepartmentName char(18),
 Phone char(18) NOT NULL,
 Building char(18),
 Room integer,
 PRIMARY KEY (DepartmentName)
)"
```

**CREATE TABLE Students**

```sql
CREATE TABLE Students
(StudentNumber integer,
 StudentLastName char(18) NOT NULL,
 StudentFirstName char(18) NOT NULL,
 Email varchar(50) NULL,
 PhoneNumber char(18) NULL,
 MajorDepartmentName char(18) NULL,
 PRIMARY KEY (StudentNumber),
 UNIQUE(Email),
 FOREIGN KEY (MajorDepartmentName) REFERENCES Departments (DepartmentName)
)"
```

---

- **NO ACTION** (delete/update is rejected)
- **CASCADE**
- **SET NULL**
- **SET DEFAULT**

---

**4 options on deletes and updates:**

- **NO ACTION**
- **CASCADE**
- **SET NULL**
- **SET DEFAULT**
Modifying Tables

- **ALTER TABLE** table\_name clause

Clauses:
- **ADD COLUMN** column\_name column\_type [constraints]
- **DROP COLUMN** column\_name
- **ALTER COLUMN / MODIFY** – DBMS specific!
- **ADD CONSTRAINT** constraint
- **DROP CONSTRAINT** constraint\_name

ALTER TABLE Examples

- **ALTER TABLE** Students **ADD COLUMN** BirthDate datetime NULL

- **ALTER TABLE** Students **DROP COLUMN** BirthDate

- **ALTER TABLE** Student **ADD CONSTRAINT** FK\_Department
  FOREIGN KEY (MajorDepartmentName)
  REFERENCES Departments (DepartmentName)
  ON DELETE NO ACTION
  ON UPDATE CASCADE
Removing Tables

- **DROP TABLE** *table_name*

  ```sql
  DROP TABLE Departments;
  ```

- If there are constraints dependent on table:
  - Remove constraints
  - Drop table

  ```sql
  ALTER TABLE Students
    DROP CONSTRAINT FK_Department;
  ```

  ```sql
  DROP TABLE Departments;
  ```

---

SQL DDL and DML

- **Data definition language (DDL)**
  statements
  - Used for creating and modifying tables, views, and other structures
  - CREATE, ALTER, DROP

- **Data manipulation language (DML)**
  statements.
  - Used for queries and data modification
  - INSERT, DELETE, UPDATE, SELECT
SQL DML

- Data manipulation language (DML) statements.
  - Used for queries and data modification
    - INSERT
    - DELETE
    - UPDATE
    - SELECT

INSERT Statement

```
INSERT INTO table_name [ (column_list) ] VALUES (data_values)
INSERT INTO table_name [ (column_list) ] select_statement
```

INSERT command:

```
INSERT INTO Students (StudentNumber, StudentLastName, StudentFirstName)
VALUES (190, 'Smith', 'John');
```

```
INSERT INTO Students VALUES(190, 'Smith', 'John', 'jsmith@usna.edu', '410-431-3456')
```

- Bulk INSERT:

```
INSERT INTO Students (StudentNumber, StudentLastName, StudentFirstName, Email, PhoneNumber)
SELECT *
FROM Second_Class_Students;
```
UPDATE Statement

**UPDATE table_name**
SET column_name1 = expression1 [,column_name2 = expression2,... ]
[ WHERE search_condition ]

- **UPDATE command:**
  ```sql
  UPDATE Students
  SET PhoneNumber = '410-123-4567'
  WHERE StudentNumber = 673;
  ```

- **BULK UPDATE command:**
  ```sql
  UPDATE Students
  SET PhoneNumber = '410-123-4567'
  WHERE StudentLastName = 'Doe';
  ```

<table>
<thead>
<tr>
<th>StudentNumber</th>
<th>StudentLastName</th>
<th>StudentFirstName</th>
<th>Email</th>
<th>PhoneNumber</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>
</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>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td>Bob</td>
<td><a href="mailto:bred@usna.edu">bred@usna.edu</a></td>
<td>443-451-7865</td>
</tr>
</tbody>
</table>

DELETE Statement

**DELETE FROM table_name**
[ WHERE search_condition ]

- **DELETE command:**
  ```sql
  DELETE FROM Students
  WHERE StudentNumber = 190;
  ```

  If you omit the WHERE clause, you will delete every row in the table!!!

- **Another example:**
  ```sql
  DELETE FROM Departments
  WHERE DepartmentName = 'ComSci'
  ```

  Integrity constraints?!
The SQL SELECT Statement

- Basic SQL Query:
  
  \[
  \text{SELECT} \ [\text{DISTINCT}] \ column\_name(s) \ | \ * \\
  \text{FROM} \ \text{table\_name(s)} \\
  [\text{WHERE} \ \text{conditions}] 
  \]

Selecting All Columns: The Asterisk (*) Keyword

\[
\text{SELECT} \ \ast \\
\text{FROM} \ \text{Students};
\]

<table>
<thead>
<tr>
<th>Student Number</th>
<th>Student LastName</th>
<th>Student FirstName</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;
```

<table>
<thead>
<tr>
<th>StudentLastName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
</tr>
<tr>
<td>Doe</td>
</tr>
<tr>
<td>Doe</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StudentLastName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
</tr>
<tr>
<td>Doe</td>
</tr>
</tbody>
</table>
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

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'
```
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.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>IT340</td>
<td>Spring2006</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>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>IT360</td>
<td>Fall2005</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>IT340</td>
<td>Spring2006</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>Fall2005</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>IT340</td>
<td>Spring2006</td>
</tr>
<tr>
<td>190</td>
<td>Smith</td>
<td><a href="mailto:jsmith@usna.edu">jsmith@usna.edu</a></td>
<td>190</td>
<td>IT360</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>312</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>IT360</td>
<td>Fall2005</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>IT340</td>
<td>Spring2006</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>Fall2005</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>IT340</td>
<td>Spring2006</td>
</tr>
<tr>
<td>190</td>
<td>Smith</td>
<td><a href="mailto:jsmith@usna.edu">jsmith@usna.edu</a></td>
<td>190</td>
<td>IT360</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>312</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>IT360</td>
<td>Fall2005</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>IT340</td>
<td>Spring2006</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>Fall2005</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 ‘Spring2010’ semester
- Find the names of all students enrolled in ‘ComSci’ courses
Sorting the Results

\[
\text{SELECT} \quad [\text{DISTINCT}] \quad \text{column\_name(s)} \mid * \\
\text{FROM} \quad \text{table\_name(s)} \\
[\text{WHERE} \quad \text{conditions}] \\
[\text{ORDER BY} \quad \text{column\_name(s)}] \quad [\text{ASC/DESC}] \\
\]

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 ‘9’

SELECT SNb, SName
FROM Students
WHERE SNb LIKE ‘9%’ 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

<table>
<thead>
<tr>
<th>Charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
</tr>
<tr>
<td>400</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>400</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(No column name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
</tr>
<tr>
<td>Brad Johnson</td>
</tr>
<tr>
<td>Jessica Smith</td>
</tr>
<tr>
<td>Mary Davis</td>
</tr>
</tbody>
</table>
Aggregate Operators

- SELECT COUNT(*)
  FROM Students

- SELECT COUNT(DISTINCT SName)
  FROM Students
  WHERE SNb > 700

- SELECT AVG(Age)
  FROM Students
  WHERE SNb LIKE '09____'

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!**

Select students with age higher than average

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

- SELECT *  
  FROM Students  
  WHERE Age > (SELECT AVG(Age)  
  FROM Students)  
  **Correct!**
Class Exercise

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

- List SNb of all students enrolled in ‘IT360’ or ‘IT340’, 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 = 2009

| (no column name) | 21 |
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>2009</td>
<td>21</td>
</tr>
<tr>
<td>2012</td>
<td>17</td>
</tr>
<tr>
<td>2011</td>
<td>18</td>
</tr>
<tr>
<td>2010</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(SN\#, SName, Email)
- Courses(Cid, CName, Dept)
- Enrolled(SN\#, 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(Alpha, LName, FName, Class, Age)
- Enroll(Alpha, CourseID, Semester, Grade)

1. Find alpha for students enrolled in **both** ‘IT360’ and ‘IT334’
2. Find name of students enrolled in **both** ‘IT360’ and ‘IT334’
Class Exercise

- Students(Alpha, LName, FName, Class, Age)
- Enroll(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]