Introduction

- Structured Query Language (SQL) is a data sublanguage that has constructs for defining and processing a database
- It can be
  - Used stand-alone within a DBMS command
  - Embedded in triggers and stored procedures
  - Used in scripting or programming languages

SQL-92

- SQL was developed by IBM in late 1970s
- SQL-92 was endorsed as a national standard by ANSI in 1992
- SQL3 incorporates some object-oriented concepts but has not gained acceptance in industry
- Data Definition Language (DDL) is used to define database structures
- Data Manipulation Language (DML) is used to query and update data
- SQL statement is terminated with a semicolon

Sample Database

<table>
<thead>
<tr>
<th>PROJECT</th>
<th>EMPLOYEE</th>
<th>ASSIGNMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
<td>Phone</td>
</tr>
<tr>
<td>100</td>
<td>Mary Jacobs</td>
<td>288-8879</td>
</tr>
<tr>
<td>200</td>
<td>Kenji Hiroto</td>
<td>287-0308</td>
</tr>
<tr>
<td>300</td>
<td>Heather Jones</td>
<td>287-9983</td>
</tr>
<tr>
<td>400</td>
<td>Rosalie Jackson</td>
<td>288-1273</td>
</tr>
<tr>
<td>500</td>
<td>James Nester</td>
<td>287-9243</td>
</tr>
<tr>
<td>600</td>
<td>Richard Wu</td>
<td>287-9232</td>
</tr>
<tr>
<td>700</td>
<td>Kim Sung</td>
<td>287-3222</td>
</tr>
</tbody>
</table>

Sample Data
CREATE TABLE

CREATE TABLE statement is used for creating relations
- Each column is described with three parts: column name, data type, and optional constraints
- Example

```
CREATE TABLE PROJECT (
    ProjectID Integer            Primary Key,
    Name           Char(25)         Unique Not Null,
    Department VarChar(100)  Null,
    MaxHours Numeric(6,1)   Default 100);
```

Data Types

- Standard data types
  - Char for fixed length character
  - VarChar for variable length character
    - It requires additional processing than Char data types
  - Integer for whole number
  - Numeric
- There are many more data types in the SQL-92 standard

Constraints

- Constraints can be defined within the CREATE TABLE statement, or they can be added to the table after it is created using the ALTER table statement
- Five types of constraints:
  - PRIMARY KEY may not have null values
  - UNIQUE may have null values
  - NULL/NOT NULL
  - FOREIGN KEY
  - CHECK

ALTER Statement

- ALTER statement changes table structure, properties, or constraints after it has been created
- Example

```
ALTER TABLE ASSIGNMENT
ADD CONSTRAINT EmployeeFK
FOREIGN KEY (EmployeeNum) REFERENCES
EMPLOYEE (EmployeeNumber)
ON UPDATE CASCADE
ON DELETE NO ACTION;
```

DROP Statements

- DROP TABLE statement removes tables and their data from the database
- A table cannot be dropped if it contains foreign key values needed by other tables
  - Use ALTER TABLE DROP CONSTRAINT to remove integrity constraints in the other table first
- Example:
  - DROP TABLE CUSTOMER;
  - ALTER TABLE ASSIGNMENT DROP CONSTRAINT ProjectFK;

SELECT Statement

- SELECT can be used to obtain values of specific columns, specific rows, or both
- Basic format:

```
SELECT (column names or *)
FROM (table name(s))
[WHERE (conditions)];
```
WHERE Clause Conditions

- Require quotes around values for Char and VarChar columns, but no quotes for Integer and Numeric columns
- AND may be used for compound conditions
- IN and NOT IN indicate ‘match any’ and ‘match all’ sets of values, respectively
- Wildcards _ and % can be used with LIKE to specify a single or multiple unknown characters, respectively
- IS NULL can be used to test for null values

Example: SELECT Statement

SELECT Name, Department, MaxHours FROM PROJECT;
- Insert Figure 6-2
(PROJECT Table only)

Example: SELECT DISTINCT

SELECT DISTINCT Department
FROM PROJECT;
- Insert Figure 6-2
(PROJECT Table only)

Example: BETWEEN

SELECT Name, Department
FROM EMPLOYEE
WHERE Department IN
('Accounting', 'Finance', 'Marketing');
- Insert Figure 6-2 (EMPLOYEE Table only)

Example: IN/NOT IN

SELECT Name, Phone, Department
FROM EMPLOYEE
WHERE Department IN
('Accounting', 'Finance', 'Marketing');
- Insert Figure 6-2 (EMPLOYEE Table only)

Example: BETWEEN

SELECT Name, Department
FROM EMPLOYEE
WHERE EmployeeNumber BETWEEN 200 AND 500;
- Or WHERE EmployeeNumber >= 200 AND EmployeeNumber <= 500;
- Insert Figure 6-2 (EMPLOYEE table only)
Example: LIKE

```
SELECT * 
FROM EMPLOYEE 
WHERE Phone LIKE '285__ ___';
```

```
SELECT * 
FROM EMPLOYEE 
WHERE Phone LIKE '285%';
```

Example: IS NULL

```
SELECT Name, Department 
FROM EMPLOYEE 
WHERE Phone IS NULL;
```

Sorting the Results

- ORDER BY phrase can be used to sort rows from SELECT statement
  
  ```
  SELECT Name, Department 
  FROM EMPLOYEE 
  ORDER BY Department;
  ```

- Two or more columns may be used for sorting purposes
  
  ```
  SELECT Name, Department 
  FROM EMPLOYEE 
  ORDER BY Department DESC, Name ASC;
  ```

Built-in Functions

- Five built-in functions for SELECT statement:
  - COUNT counts the number of rows in the result
  - SUM totals the values in a numeric column
  - AVG calculates an average value
  - MAX retrieves a maximum value
  - MIN retrieves a minimum value

- Result is a single number (relation with a single row and a single column)

- Column names cannot be mixed with built-in functions

- Built-in functions cannot be used in WHERE clauses

Example: Built-in Functions

```
SELECT COUNT (DISTINCT Department) 
FROM PROJECT;
```

```
SELECT MIN(MaxHours), MAX(MaxHours), SUM(MaxHours) 
FROM PROJECT 
WHERE ProjectID < 1500;
```

Built-in Functions and Grouping

- GROUP BY allows a column and a built-in function to be used together

- GROUP BY sorts the table by the named column and applies the built-in function to groups of rows having the same value of the named column

- WHERE condition must be applied before GROUP BY phrase

Example

```
SELECT Department, Count(*) 
FROM EMPLOYEE 
WHERE EmployeeNumber < 600 
GROUP BY Department 
HAVING COUNT(*) > 1;
```
Querying Multiple Tables

- Multiple tables can be queried by using either subqueries or joins.
- If all of the result data comes from a single table, subqueries can be used.
- If results come from two or more tables, joins must be used.
- Joins cannot substitute for correlated subqueries nor for queries that involve EXISTS and NOT EXISTS.

Subqueries

- Subqueries can be extended to include many levels.
- Example:

```sql
SELECT DISTINCT Name
FROM EMPLOYEE
WHERE EmployeeNumber IN
(SELECT EmployeeNumber
FROM ASSIGNMENT
WHERE HoursWorked > 40
AND ProjectID IN
(SELECT ProjectID
FROM PROJECT
WHERE Department = 'Accounting'));
```

Joins

- The basic idea of a join is to form a new relation by connecting the contents of two or more other relations.
- This joined table can be processed like any other table.
- Example:

```sql
SELECT PROJECT.Name, HoursWorked, EMPLOYEE.Name
FROM PROJECT, ASSIGNMENT, EMPLOYEE
WHERE PROJECT.ProjectID = ASSIGNMENT.ProjectID
AND EMPLOYEE.EmployeeNumber = ASSIGNMENT.EmployeeNumber;
```

Alternate Join Syntax

- SQL 92's alternative join syntax substitutes the words JOIN and ON for WHERE.
- Using aliases for table names improves the readability of a join.
- Example: alias E is assigned to the EMPLOYEE table.

```sql
SELECT P.Name, HoursWorked, E.Name
FROM PROJECT P JOIN ASSIGNMENT A
ON P.ProjectID = A.ProjectID
JOIN EMPLOYEE E
ON A.EmployeeNumber = E.EmployeeNumber;
```

Outer Joins

- Outer joins can be used to ensure that all rows from a table appear in the result.
- Left (right) outer join: every row on the table on the left (right) hand side is included in the results even though the row may not have a match.
- Outer joins can be nested.

Example: Outer Join

- Left outer join

```
SELECT Name, HoursWorked
FROM PROJECT LEFT JOIN ASSIGNMENT
ON PROJECT.ProjectID = ASSIGNMENT.ProjectID;
```

- Nested outer join

```
SELECT PROJECT.Name, HoursWorked, EMPLOYEE.Name
FROM (PROJECT LEFT JOIN ASSIGNMENT
ON PROJECT.ProjectID = ASSIGNMENT.ProjectID)
LEFT JOIN EMPLOYEE
ON EMPLOYEE.EmployeeNumber = Assignment.EmployeeNumber;
```
INSERT INTO Statement

- The order of the column names must match the order of the values.
- Values for all NOT NULL columns must be provided.
- No value needs to be provided for a surrogate primary key.
- It is possible to use a select statement to provide the values for bulk inserts from a second table.
- Examples:
  - INSERT INTO PROJECT VALUES (1600, 'Q4 Tax Prep', 'Accounting', 100);
  - INSERT INTO PROJECT (Name, ProjectID) VALUES ('Q1+ Tax Prep', 1700);

UPDATE Statement

- UPDATE statement is used to modify values of existing data.
- Example:
  UPDATE EMPLOYEE
  SET Phone = '287-1435'
  WHERE Name = 'James Nestor';
- UPDATE can also be used to modify more than one column value at a time.
  UPDATE EMPLOYEE
  SET Phone = '285-0091', Department = 'Production'
  WHERE EmployeeNumber = 200;

DELETE FROM Statement

- Delete statement eliminates rows from a table.
- Example:
  DELETE FROM PROJECT
  WHERE Department = 'Accounting';
- ON DELETE CASCADE removes any related referential integrity constraint of a deleted row.

Chapter 6
Introduction to Structured Query Language (SQL)