SQL

- The most widely used commercial language
- SQL is NOT a Turing machine equivalent language
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database
Data Definition Language (DDL)

- Specification notation for defining the database schema
- Example: `create table instructor (
  ID char(5),
  name varchar(20),
  dept_name varchar(20),
  salary numeric(8,2))`
- DDL compiler generates a set of table templates stored in a data dictionary
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
- DML also known as query language
- Two classes of languages
  - Pure – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - Commercial – used in commercial systems
    - SQL is the most widely used commercial language
SQL, an interactive query language

<table>
<thead>
<tr>
<th>Course-Name</th>
<th>Instructor</th>
<th>Room-Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Aref</td>
<td>DS1</td>
</tr>
<tr>
<td>Operating Syst.</td>
<td>Rodriguez</td>
<td>N3</td>
</tr>
<tr>
<td>Networks</td>
<td>Fahmy</td>
<td>N3</td>
</tr>
<tr>
<td>Security</td>
<td>Spafford</td>
<td>G</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Room-Name</th>
<th>Building</th>
<th>Floor</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS1</td>
<td>Recitation</td>
<td>1</td>
</tr>
<tr>
<td>N3</td>
<td>Recitation</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>Univ. Hall</td>
<td>2</td>
</tr>
</tbody>
</table>

SELECT Course-Name, Room-Name, Floor
FROM Courses, Rooms WHERE
Courses.Room-Name = Rooms.Room-Name
AND Instructor = ‘Fahmy’;

### Basic Query Structure

- A typical SQL query has the form:

  ```
  select A_1, A_2, ..., A_n
  from r_1, r_2, ..., r_m
  where P
  ```

  - \( A_i \) represents an attribute
  - \( R_i \) represents a relation
  - \( P \) is a predicate.

- The result of an SQL query is a relation.
The select Clause

- The `select` clause lists the attributes desired in the result of a query
  - corresponds to the projection operation of the relational algebra
- Example: find the names of all instructors:
  ```sql
  select name
  from instructor
  ```
- NOTE: SQL names are case insensitive (i.e., you may use upper- or lower-case letters.)
  - E.g., `Name ≡ NAME ≡ name`
  - Some people use upper case wherever we use bold font.

The select Clause (Cont.)

- SQL allows duplicates in relations as well as in query results.
- To force the elimination of duplicates, insert the keyword `distinct` after `select`.
- Find the department names of all instructors, and remove duplicates
  ```sql
  select distinct dept_name
  from instructor
  ```
- The keyword `all` specifies that duplicates should not be removed.
  ```sql
  select all dept_name
  from instructor
  ```
The select Clause (Cont.)

- An asterisk in the select clause denotes “all attributes”

```sql
select *
from instructor
```

- An attribute can be a literal with no `from` clause

```sql
select '437'
```

  - Results is a table with one column and a single row with value “437”
  - Can give the column a name using:

```sql
select '437' as FOO
```

- An attribute can be a literal with `from` clause

```sql
select 'A'
from instructor
```

  - Result is a table with one column and N rows (number of tuples in the `instructors` table), each row with value “A”

The select Clause (Cont.)

- The `select` clause can contain arithmetic expressions involving the operation, +, −, *, and /, and operating on constants or attributes of tuples.

  - The query:

```sql
select ID, name, salary/12
from instructor
```

  would return a relation that is the same as the `instructor` relation, except that the value of the attribute `salary` is divided by 12.

  - Can rename “salary/12” using the `as` clause:

```sql
select ID, name, salary/12 as monthly_salary
```
The where Clause

- The where clause specifies conditions that the result must satisfy
  - Corresponds to the selection predicate of the relational algebra.
- To find all instructors in Comp. Sci. dept
  
  ```
  select name
  from instructor
  where dept_name = 'Comp. Sci.'
  ```
- Comparison results can be combined using the logical connectives and, or, and not
  - To find all instructors in Comp. Sci. dept with salary > 80000
    
    ```
    select name
    from instructor
    where dept_name = 'Comp. Sci.' and salary > 80000
    ```
- Comparisons can be applied to results of arithmetic expressions.

String Operations

- SQL includes a string-matching operator for comparisons on character strings. The operator like uses patterns that are described using two special characters:
  - percent (%) . The % character matches any substring.
  - underscore (_). The _ character matches any character.
- Find the names of all instructors whose name includes the substring “dar”.
  
  ```
  select name
  from instructor
  where name like '%dar%'
  ```
- Match the string “100%”
  
  ```
  like '100\%'
  ```
  in that above we use backslash (\) as the escape character.
String Operations (Cont.)

- Patterns are case sensitive.
- Pattern matching examples:
  - `'Intro%'` matches any string beginning with “Intro”.
  - `%Comp%` matches any string containing “Comp” as a substring.
  - `_ _ _` matches any string of exactly three characters.
  - `_ _ _ %` matches any string of at least three characters.

- SQL supports a variety of string operations such as
  - concatenation (using “||”)
  - converting from upper to lower case (and vice versa)
  - finding string length, extracting substrings, etc.

Where Clause Predicates

- SQL includes a **between** comparison operator
- Example: Find the names of all instructors with salary between $90,000 and $100,000
  
  ```sql
  select name 
  from instructor 
  where salary between 90000 and 100000 
  ```
Ordering the Display of Tuples

- List in alphabetic order the names of all instructors
  
  ```sql
  select distinct name
  from instructor
  order by name
  ```

- We may specify `desc` for descending order or `asc` for ascending order, for each attribute; ascending order is the default.
  - Example: `order by name desc`

- Can sort on multiple attributes
  - Example: `order by dept_name, name`

And now, some examples

- Ideas on a favorite table?