

## Structured Query Language (Chapter 8)

Database Management  
COP4540, SCS, FIU

## SQL Introduction

- Most important relational data manipulation language endorsed by American National Standard Institute (ANSI)
- SEQUEL developed by IBM in mid 70's, renamed SQL in 80's
- Input: one or more relations  
Output: a table
- Syntax may vary slightly among different DBMS. (The examples appearing in lecture notes comply with ANSI SQL2/SQL92)

Database Management  
COP4540, SCS, FIU

## Basic Components

- A standard relational database language
  - Data definition language (DDL)
    - Define relation schemes, delete relations, create indices, and Modify schemes.
  - Data manipulation language (DML)
    - Insert, delete, and modify tuples.
  - Query language (QL)
    - select

Database Management  
COP4540, SCS, FIU

## SQL as DDL

SQL Statement	Task
CREATE DATABASE	Create a database
CREATE TABLE	Define a new table
DROP TABLE	Destroy a table
ALTER TABLE	Modify definition of a table
CREATE INDEX	Define an index
DROP INDEX	Destroy an index
CREATE VIEW	Define a logical table
DROP VIEW	Destroy a view
CREATE SCHEMA	Define a database owned by a user

Database Management  
COP4540, SCS, FIU

## Example of DDL Commands (1)

- Defining a database  
**CREATE DATABASE UNIVERSITY**
- Creating tables:  
**CREATE TABLE CLASS**  
(Name **VARCHAR(40)** **NOT NULL**,  
Time **DATE** **NOT NULL**,  
**CONSTRAINT CLASS\_PK PRIMARY KEY (Name)**);

Database Management  
COP4540, SCS, FIU

## Example of DDL Commands (2)

```
CREATE TABLE ENROLLMENT
(SNo, INTEGER NOT NULL,
CName VARCHAR(40) NOT NULL,
PositionNo INTEGER NOT NULL,
CONSTRAINT ENROLLMENT_PK
PRIMARY KEY (SNo, CName),
CONSTRAINT ENROLLMENT_FK1
FOREIGN KEY (SNo) REFERENCES STUDENT(SID)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT ENROLLMENT_FK2
FOREIGN KEY (CName) REFERENCES CLASS(Name)
ON DELETE CASCADE ON UPDATE CASCADE,
);
```

Database Management  
COP4540, SCS, FIU

### Example of DDL Commands (3)

- Changing table definitions  
**ALTER TABLE CLASS**  
**ADD** (Room VARCHAR(5));
- Removing tables:  
**DROP TABLE CLASS;**  
**DROP TABLE CLASS CASCADE;**  
**DROP TABLE CLASS RESTRICT;**

Database Management  
COP4540, SCS, FIU

### SQL as DML

SQL Statement	Task
INSERT INTO	Insert row(s)
DELETE FROM	Delete row(s)
UPDATE	Modify row(s)

Relations Used for SQL Statements:

- JUNIOR(Snum, Name, Major)
- HONOR\_STU(Number, Name, Interest)
- STUDENT(SID, Name, Major, Year, Age)
- CLASS(CID, Name, Time, Room)
- ENROLLMENT(SNo, CName, PositionNo)
- FACULTY(FID, Name, Dept, Salary)

Database Management  
COP4540, SCS, FIU

### Insert One Row

- Inserting one row  
**INSERT INTO ENROLLMENT**  
**VALUES** ( 400, 'COP4540', 50);  
  
**INSERT INTO CLASS** (CID, Name, Room)  
**VALUES** ( COP3338, 'Programming III', 'ECS134');

Database Management  
COP4540, SCS, FIU

### Inserting Rows in Groups

```
INSERT INTO JUNIOR  
VALUES  
(SELECT SID, Name, Major  
FROM STUDENT  
WHERE Year = 'JR'  
);
```

Database Management  
COP4540, SCS, FIU

### Deleting Database Contents

- Deleting rows that meet a certain criterion  
**DELETE FROM STUDENT WHERE** SID = 100;  
**DELETE FROM STUDENT**  
**WHERE** Major = 'Biochemistry';
- Deleting all rows from a table  
**DELETE FROM STUDENT;**

Database Management  
COP4540, SCS, FIU

### Changing Database Contents

```
UPDATE CLASS  
SET Room = 'PC 213'  
WHERE Name = 'Database';  
  
UPDATE FACULTY  
SET Salary = Salary * 1.1  
WHERE Dept = 'Computer Science';
```

Database Management  
COP4540, SCS, FIU

## Querying with SQL

- General Syntax of SELECT Statement

```
SELECT [ALL| DISTINCT] column_list
FROM table_list
[WHERE conditional expression]
[GROUP BY group_by_column_list]
[HAVING condition expression]
[ORDER BY order_by_column_list];
```

Database Management  
COP4540, SCS, FIU

## Simple Queries in SQL

- Basic structure
  - *select* clause lists attributes to be copied
    - Corresponds to projection in relational algebra.
  - *from* clause lists relations to be used.
  - *where* clause
    - Corresponds to selection in relational algebra.
  - Typical simple query has the form:

```
SELECT A1, A2, ..., An
FROM R
WHERE Condi
```

where each  $A_i$  represents an attribute in  $R$ ,  $R$  is a relation, and  $Condi$  is a condition.

Database Management  
COP4540, SCS, FIU

## Selecting Columns (1)

1. Name the relation to be projected
2. List the columns to be shown

```
SELECT SID, Name, Major
FROM STUDENT;
```

```
SELECT Name, SID, Major (attribute
FROM STUDENT; order changed)
```

Database Management  
COP4540, SCS, FIU

## Selecting Columns (2)

- Changing column headers

```
SELECT SID AS Student_ID, Name
FROM STUDENT;
```
- Duplicates not eliminated automatically. If duplicate rows must be removed, do:

```
SELECT DISTINCT Major
FROM STUDENT;
```

Database Management  
COP4540, SCS, FIU

## Using Expressions and Constants

- Expressions are mathematical manipulations of data in the table

```
SELECT Product_Name, Unit_Price, On_Hand,
Unit_Price * On_Hand AS Value
FROM PRODUCT;
```

```
SELECT name, price/1.6 as price_in_US, 'US$' as unit
FROM Merchandise
WHERE type = 'laptop';
```

Database Management  
COP4540, SCS, FIU

## Using Wildcards

- Star (\*) as lists of All attributes

```
SELECT SID, Name, Major, Year, Age
FROM STUDENT
WHERE Major = 'MATH';
```

```
SELECT *
FROM STUDENT
WHERE Major = 'MATH';
```

Database Management  
COP4540, SCS, FIU

## The where clause

- The predicates can be more complicated, and can involve
  - Logical connectives: *AND*, *OR* and *NOT*.
  - Common comparison operators: =, <>, <, >, <=, >=
  - String comparison operators: *LIKE*
  - Arithmetic expressions on constant or tuple values.
  - The *BETWEEN* operator for ranges of values.
  - The *IN* operator for set/bag.

Database Management  
COP4540, SCS, FIU

## Comparison of strings

- The strings are compared in lexicographic order.
  - Given two strings  $A = a_1a_2\dots a_n$  and  $B = b_1b_2\dots b_m$ , we say  $A < B$  if:
    - $a_1 < b_1$ , or  $a_1 = b_1$  and  $a_2 < b_2$ , or  $a_1 = b_1$ ,  $a_2 = b_2$  and  $a_3 < b_3$ , and so on.
    - If  $n < m$  and  $a_1a_2\dots a_n = b_1b_2\dots b_n$ .
- Pattern Match
  - Format: *string* *LIKE* *pattern*
  - Two special characters in the *pattern*
    - % any sequence of 0 or more characters in the *string*.
    - \_ any one character in the *string*

Database Management  
COP4540, SCS, FIU

## LIKE

```
SELECT *
FROM STUDENT
WHERE SID LIKE '___-__-5678';
```

```
SELECT *
FROM STUDENT
WHERE SID LIKE '%5678';
```

Database Management  
COP4540, SCS, FIU

## Ranges

```
SELECT Name, Age
FROM STUDENT
WHERE Age >= 20 AND Age <= 31;
```

```
SELECT Name, Age
FROM STUDENT
WHERE Age BETWEEN 21 AND 31
```

Database Management  
COP4540, SCS, FIU

## IN and NOT IN Lists

```
SELECT Name
FROM STUDENT
WHERE Major IN ('MATH', 'ACCOUNTING');
```

*IN* is particular useful in SQL Statements that use sub-queries.

Database Management  
COP4540, SCS, FIU

## SQL Built-in (Aggregation) Functions

- COUNT: count number of rows in a table
- SUM: sum up numeric columns
- AVG: compute average
- MAX: obtain maximum value of a column
- MIN: obtain minimum value of a column

Database Management  
COP4540, SCS, FIU

## Using Built-in Functions

```
SELECT COUNT(*)  
FROM STUDENT;
```

- Cannot be mixed with column names in SELECT
- Cannot be used in WHERE clause

× **SELECT** Name, COUNT(\*)  
**FROM** STUDENT;  
unless a GROUP BY clause is used.

√ **SELECT** Name, COUNT(\*)  
**FROM** STUDENT  
**GROUP BY** Name;

Database Management  
COP4540, SCS, FIU

## Categorizing Results: GROUP BY

Count the number of students in each major

```
SELECT Major, COUNT(*)  
FROM STUDENT  
GROUP BY Major;
```

1. Divides a table into subsets
2. An aggregate function can be used to provide summary information for that group.

Database Management  
COP4540, SCS, FIU

## Qualifying Results by Categories HAVING

Find only department with more than 10 student

```
SELECT Major  
FROM STUDENT  
GROUP BY Major  
HAVING COUNT(*) > 10;
```

Database Management  
COP4540, SCS, FIU

## Ordering the output

- *Order by* clause
  - ORDER BY <list of attributes> [ASC | DESC]
- The order is based on the value of attributes specified in the <list of attributes>
- The order is by default *ascending*.  
**SELECT** Name, Major, Age  
**FROM** STUDENT  
**ORDER BY** Major ASC, Age DESC;

Database Management  
COP4540, SCS, FIU