



THE LECTURE 4



SQL QUERIES

SQL

- Standardized
- Data Manipulation and Data Definition
- Can be embedded into general programming languages
- Statements specify what is to be done, NOT how to do it

DML - RETRIEVAL

- SELECT statement - can do MANY different things

- List whole table

```
SELECT * FROM STUDENT
```

- Relational Algebra PROJECT

```
SELECT StdID, LNAME, FNAME
```

```
FROM STUDENT
```

- Unlike RA, duplicates aren't (automatically) removed. E.g.

```
SELECT MAJOR
```

```
FROM STUDENT
```

- If we want duplicates removed, specify DISTINCT after SELECT

```
SELECT DISTINCT MAJOR
```

```
FROM STUDENT
```

DML - RESTRICT

- Relational Algebra RESTRICT/SELECT

- e.g. list all students who are freshmen

```
SELECT *
```

```
FROM STUDENT
```

```
WHERE YEAR = 'Fr'
```

- PROJECT and RESTRICT together

```
SELECT StdID, LNAME, FNAME
```

```
FROM STUDENT
```

```
WHERE YEAR = 'Fr'
```

MORE RETRIEVAL

- WHERE condition can be as complicated as you need it to be. E.g. freshmen with poor grades

```
SELECT StdID,LNAME,FNAME
```

```
FROM STUDENT
```

```
WHERE YEAR = 'Fr' AND GPA < 2.5
```

ORDERING

- With a little extra complexity, we can get our output in order by some particular attribute(s) E.g. order students by major

```
SELECT StdID,LNAME,FNAME, MAJOR  
FROM STUDENT  
ORDER BY MAJOR DESC
```

JOINING TABLES

- As you know, an important task with relational databases is relating info from more than one table (for instance, natural join in RA) E.g. show all students with the class indices of enrollments

```
SELECT StudID,LNAME,FNAME, INDEX  
FROM STUDENT, ENROLLMENTS  
WHERE StudID = Enrollments.Student
```

RESTRICT WITH JOIN

- Sometimes you don't want the entire join - you might want to restrict the results (sometimes called select-project-join) E.g. show all students enrolled in a particular section

```
SELECT StdID, LNAME, FNAME  
FROM STUDENT, ENROLLMENTS  
WHERE STUDENT.StdID = ENROLLMENTS.Student  
AND ENROLLMENTS.INDEX = 70238
```


MULTI-WAY JOIN

- We can join more than two tables together (which is a good thing; joining students with enrollments was a little unsatisfying because we didn't see any info about the section. Let's show all students with the section info for sections they enrolled in

```
SELECT STUDENT.*, SECTION.*  
FROM STUDENT, ENROLLMENTS, SECTION  
WHERE STUDENT.StdID = ENROLLMENTS.Student  
      AND ENROLLMENTS.INDEX = SECTION.INDEX
```

ALIAS

- Alternate name (for a table)
- `SELECT STUDENT.*, SECTION.*`
`FROM STUDENT A, ENROLLMENTS B, SECTION C`
`WHERE A.StdID = B.student AND B.index = C.index;`
- Here A,B, and C are aliases
- Used as a convenience, not necessary

QUERIES

- Special Operators
 - **BETWEEN** - used to define range limits.
 - **IS NULL** - used to check whether an attribute value is null
 - **LIKE** - used to check for similar character strings.
 - **IN** - used to check whether an attribute value matches a value contained within a (sub)set of listed values.
 - **EXISTS** - used to check whether an attribute has a value. In effect, **EXISTS** is the opposite of **IS NULL**.
 - Can also be used to check if a **subquery** returns any rows

QUERIES

- Special Operators

BETWEEN is used to define range limits.

```
SELECT *  
  FROM STUDENT  
 WHERE GPA BETWEEN 2.0 AND 2.1;
```

QUERIES

- Special Operators

IS NULL is used to check whether an attribute value is null.

```
SELECT INDEX, DEPT, CLASS, TIME  
FROM SECTION  
WHERE ROOM IS NULL;
```

QUERIES

- Special Operators

LIKE is used to check for similar character strings.

```
SELECT * FROM CATALOG_CLASS  
WHERE TITLE LIKE '%Lang%';
```

- % stands for 0 or more char wildcard
- _ stands for a one char wildcard
- e.g. WHERE TITLE LIKE '%Network%' finds classes whose title includes the substring 'Network'
- NOTE: While SQL commands are not case-sensitive, SQL strings are

QUERIES

- Special Operators

IN is used to check whether an attribute value matches a value contained within a (sub)set of listed values.

```
SELECT * FROM ENROLLMENT  
WHERE INDEX IN (66415, 66421);
```

EXISTS is used to check whether an attribute has value.

```
SELECT * FROM SECTIONs  
WHERE PROFESSOR EXISTS;
```

SOME SQL NUMERIC AGGREGATE FUNCTIONS

TABLE 6.10 SOME BASIC SQL AGGREGATE FUNCTIONS

FUNCTION	OUTPUT
COUNT	The number of rows containing “non null” values
MIN	The minimum attribute value encountered in a given column
MAX	The maximum attribute value encountered in a given column
SUM	The sum of all values for a given column
AVG	The arithmetic mean (average) for the specified column

AGGREGATE FUNCTIONS - AVG

- e.g. find ave GPA for all students

```
SELECT AVG(GPA)  
FROM STUDENT
```

- What is the average GPA of all freshmen

```
SELECT AVG(GPA)  
FROM STUDENT  
WHERE YEAR = 'Fr'
```

COUNT

- How many sections are offered

```
SELECT COUNT(*)  
FROM SECTION
```

- How many computer science majors are there?

```
SELECT COUNT(*)  
FROM STUDENT  
WHERE MAJOR = 'CS'
```

- How many distinct classes are being offered

```
SELECT COUNT(DISTINCT DEPT,CLASS)  
FROM SECTION
```

GROUP BY - SUBTOTALS

- Total Enrollments by Dept

```
SELECT DEPT, SUM(enroll)
FROM SECTION
GROUP BY DEPT
```

HAVING

- Total Enrollments by Dept for depts offering more than 10 sections

```
SELECT DEPT, SUM(enroll)
FROM SECTION
GROUP BY DEPT
HAVING COUNT(*) > 10
```

- Average Evening Enrollments by Depts with low ave enrollment

```
SELECT DEPT, AVG(enroll)
FROM SECTION
WHERE Sect >= 40
GROUP BY DEPT
HAVING AVG(enroll) < 15
```

HAVING

- Total Enrollments by Dept for depts offering more than 10 sections

```
SELECT DEPT, SUM(credits)
FROM CATALOGCOURSE
GROUP BY DEPT
HAVING COUNT(*) > 10
```

- Average Highest Enrollment (capacity) for upper division courses by Depts – for depts with many upper division sections

```
SELECT DEPT, AVG(MaxEnroll)
FROM SECTIONS
WHERE Course >= 200
GROUP BY DEPT
HAVING COUNT(*) >= 10
```

NESTED QUERIES

- Sometimes the result of one query can be used in another query - thus we can have nested queries
- e.g. find students whose GPA is above average

```
SELECT *
```

```
FROM STUDENT
```

```
WHERE GPA > ( SELECT AVG(GPA)
```

```
FROM STUDENT )
```

NESTED QUERIES

- It is fairly common for a nested query to use the set operation IN - which tests whether a value is a member of a set of values. So for instance, suppose we want to know all sections in which there are any freshman

```
SELECT DISTINCT CLASSINDEX  
FROM ENROLLMENTS  
WHERE STDSSN IN ( SELECT SSN  
                  FROM STUDENT  
                  WHERE YEAR = 'Fr' )
```

LEFT OUTER JOIN

```
SELECT P_CODE,VENDOR.V_CODE,V_NAME  
FROM VENDOR LEFT JOIN PRODUCT ON VENDOR.V_CODE = PRODUCT.V_CODE
```


RIGHT OUTER JOIN

```
SELECT PRODUCT.P_CODE,VENDOR.V_CODE,V_NAME  
FROM VENDOR RIGHT JOIN PRODUCT ON VENDOR.V_CODE = PRODUCT.V_CODE
```