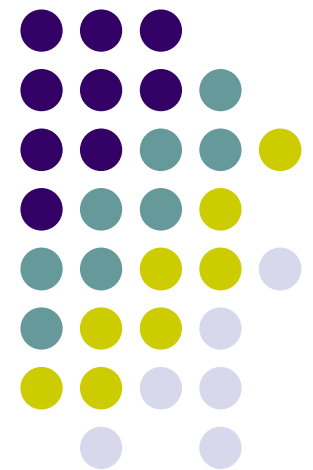


Select statement and the like

**CIS 331:
Introduction to
Database Systems**





Topics:

- **SELECT**
- **LIKE ...**
- **... JOIN ...**
- **Set operations**
 - **INTERSECT**
 - **UNION**
 - **MINUS**



Select

- Of all statements, SELECT's are most frequently used in SQL queries:

```
SELECT ssn, first_name, last_name  
    FROM students  
    WHERE last_name = 'Khan'  
  
;
```

- * stands for “all attributes:”

```
SELECT *  
    FROM classes  
  
;
```



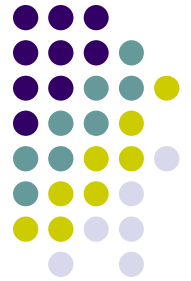
Attribute values

- **Values** of attributes **are** case sensitive:

```
SELECT first_name, middle_name, last_name
      FROM students
      WHERE major = 'CIS'
;
```

- is not the same as:

```
SELECT first_name, middle_name, last_name
      FROM students
      WHERE major = 'cis'
;
```



Attribute names

- Attribute **names** themselves **are not** case sensitive:

```
SELECT first_name, middle_name, last_name
      FROM students
      WHERE major = 'CIS'
;
```

- is the same as:

```
SELECT first_name, middle_name, last_name
      FROM students
      WHERE MAJOR = 'CIS'
;
```



Attributes

- Notice how string attribute values are in **single quotes**:

```
SELECT student
  FROM students_classes
     WHERE code = 'CIS525'
;
```

- Double quotes will not work!

```
SELECT student
  FROM students_classes
     WHERE code = "CIS525"
;
```



Numerical attributes

- But **numbers** are **not** in quotation marks:

```
SELECT student
  FROM students_classes
  WHERE grade >= 3.67
;
```



Like, Not Like

- Sometimes we may want to find an approximate match:

```
SELECT class_code, name
      FROM classes
      WHERE name LIKE '%Data%'
;
```

- Percent signs (%) function as wild-card characters; they can match any string or character.

```
SELECT class_code, name
      FROM classes
      WHERE name NOT LIKE '%Computation'
;
```

Join



- But to unleash the full potential of relational databases we need to be able to combine data stored in several tables. We can do this by **joining** tables:

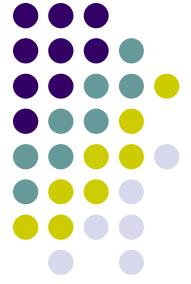
```
SELECT first_name, middle_name, last_name
       FROM students, students_classes
       WHERE students.ssn =
              students_classes.student
       AND students_classes.class = 'CIS525'
;
```



Aliases

- Aliases are used to abbreviate names of tables and attributes (saves time in typing):

```
SELECT first_name, middle_name, last_name
       FROM students S, students_classes SC
       WHERE S.ssn = SC.student
       AND SC.class = 'CIS525'
;
```

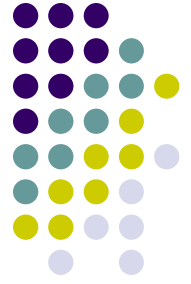


Select

- Conditions can be liberally combined to obtain the desired result (getting names of students who are attending either CIS525 or CIS526)

```
SELECT first_name, middle_name, last_name
       FROM students S, students_classes SC
       WHERE S.ssn = SC.student
       AND (
           SC.class = 'CIS525'
          OR SC.class = 'CIS526'
        )
;

```



In (... , ...)

- That is equivalent to:

```
SELECT first_name, middle_name, last_name
       FROM students S, students_classes SC
       WHERE S.ssn = SC.student
       AND SC.class IN ('CIS525', 'CIS526')
;
```



Union

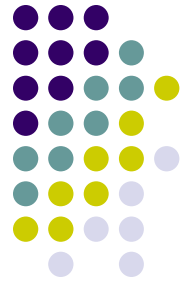
- That is equivalent to:

```
SELECT first_name, middle_name, last_name
      FROM students S, students_classes SC
      WHERE S.ssn = SC.student
      AND SC.class = 'CIS525'
```

UNION

```
SELECT first_name, middle_name, last_name
      FROM students S, students_classes SC
      WHERE S.ssn = SC.student
      AND SC.class = 'CIS526'
```

```
;
```



Intersection

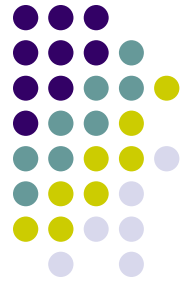
- To get a list of students who are attending both CIS525 and CIS526, we can intersect the two sets:

```
SELECT first_name, middle_name, last_name
       FROM students S, students_classes SC
       WHERE S.ssn = SC.student
             AND SC.class = 'CIS525'
```

INTERSECT

```
SELECT first_name, middle_name, last_name
       FROM students S, students_classes SC
       WHERE S.ssn = SC.student
             AND SC.class = 'CIS526'
```

;



Intersection

- Be careful not to do something like this:

```
SELECT first_name, middle_name, last_name
       FROM students S, students_classes SC
       WHERE S.ssn = SC.student
             AND (
                 SC.class = 'CIS525'
                 AND SC.class = 'CIS526'
             )
;

```

- Because no attribute can assume two different values at the same time, this query will return nothing.



Set difference

- And finally, to get a list of students who are attending CIS525 but are not attending CIS526:

```
SELECT first_name, middle_name, last_name
       FROM students S, students_classes SC
       WHERE S.ssn = SC.student
             AND SC.class = 'CIS525'
```

MINUS

```
SELECT first_name, middle_name, last_name
       FROM students S, students_classes SC
       WHERE S.ssn = SC.student
             AND SC.class = 'CIS526'
```

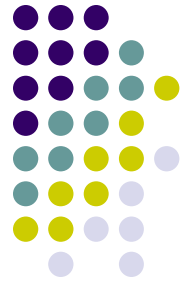
;



Note: set operations

- Operands of union (UNION), intersection (INTERSECT) and difference (MINUS) operations have to be **union compatible**, i.e. they have to come from the same **domain**:

```
SELECT first_name, middle_name, last_name
      FROM students S, students_classes SC
      WHERE S.ssn = SC.student
      AND SC.class = 'CIS525'
[ UNION | INTERSECT | MINUS ]
SELECT first_name, middle_name, last_name
      FROM students S, students_classes SC
      WHERE S.ssn = SC.student
      AND SC.class = 'CIS526'
;
```



Note: set operations

- This, for example, would not work:

```
SELECT ssn, last_name
  FROM students S, students_classes SC
   WHERE S.ssn = SC.student
     AND SC.class = 'CIS525'
[ UNION | INTERSECT | MINUS ]
SELECT first_name, middle_name, last_name
  FROM students S, students_classes SC
   WHERE S.ssn = SC.student
     AND SC.class = 'CIS526'
;
```

- Because (**ssn, last_name**) and (**first_name, middle_name, last_name**) do not come from the same domain.