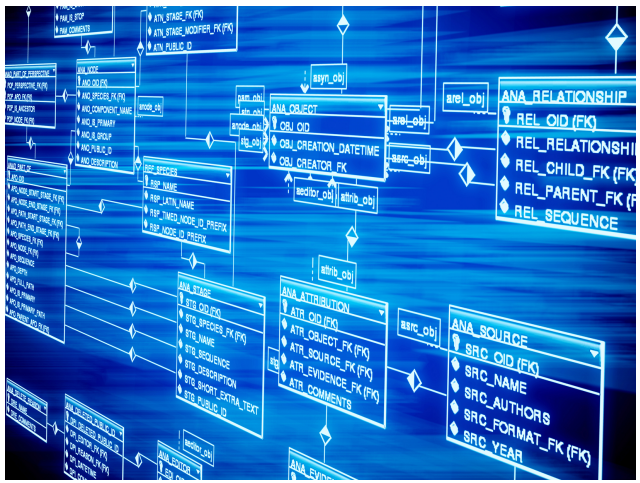
CSC 261/461
Introduction to Databases

Eustrat Zhupa



Structured Query Language - INSERT

- INSERT is used to add a single tuple to a table
  - have to specify the table name and a list of values for the tuple.
  - values should be listed in the same order in which they were defined with CREATE TABLE.

```
INSERT INTO EMPLOYEE
VALUES ('Richard','K','Marini','653298653',
'1962-12-30','98 Oak Forest, Katy, TX', 'M',
37000, '653298653', 4 );
```

Structured Query Language - INSERT

- the user can specify explicit attribute names that correspond to the values provided in the `INSERT` command.
- useful if a relation has many attributes but you assign only a few.
- the values must include all attributes with `NOT NULL` specification and no default value.
- Attributes with NULL or DEFAULT values are the ones that can be *left out*.

```
INSERT INTO EMPLOYEE (Fname, Lname, Dno, Ssn)
VALUES ('Richard', 'Marini', 4, '653298653');
```

Structured Query Language - INSERT

You can also add many tuples at a time

```
CREATE TABLE WORKS_ON_INFO
  ( Emp_name VARCHAR(15),
    Proj_name VARCHAR(15),
    Hours_per_week DECIMAL(3,1) );

INSERT INTO WORKS_ON_INFO ( Emp_name, Proj_name, Hours_per_week )
SELECT E.Lname, P.Pname, W.Hours
FROM PROJECT P, WORKS_ON W, EMPLOYEE E
WHERE P.Pnumber=W.Pno AND W.Essn=E.Ssn;
```

Structured Query Language - DELETE

- `DELETE` removes tuples from a relation.
- includes a `WHERE` clause to select the tuples to be deleted.
- Tuples are explicitly deleted from only one table at a time.
  - may propagate to other relations if referential triggered actions are specified.
- a missing `WHERE` deletes all tuples in the relation
  - table remains in the database as an empty table.

Structured Query Language - DELETE

```
DELETE FROM EMPLOYEE
WHERE Lname='Brown';

DELETE FROM EMPLOYEE
WHERE Ssn='123456789';

DELETE FROM EMPLOYEE
WHERE Dno=5;

DELETE FROM EMPLOYEE;
```

[md]

Structured Query Language - UPDATE

- `UPDATE` is used to modify attribute values of one or more selected tuples.
- a `WHERE` clause selects the tuples to be modified from a single relation.
- Updating a primary key value may propagate to the foreign key values of tuples in other relations.

Structured Query Language - Example

```
UPDATE PROJECT
SET Plocation = 'Bellaire', Dnum = 5
WHERE Pnumber=10;

UPDATE EMPLOYEE
SET Salary = Salary * 1.1
WHERE Dno = 5;
```

Note: *Old* salary on the right, *new* salary on the left.

[mu]

Structured Query Language - NULLs

- SQL has various rules for dealing with NULL values.
- NULL values represent:
  1. Unknown value.
  2. Unavailable or withheld value.
  3. Not applicable attribute.
- SQL *does not distinguish* between these cases.

Structured Query Language

Table 5.1 Logical Connectives in Three-Valued Logic

(a)

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

(b)

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

(c)

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

Structured Query Language

- SQL allows queries that check whether an attribute value is NULL.
- to compare an attribute value to NULL, use `IS` or `IS NOT`.
- **Query** Retrieve names (first, last) of all employees who do *not* have supervisors.

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Super_ssn IS NULL;
```

Nested Queries

Sometimes you need to compare against a (multi)set of values (tuples).

- SQL uses the comparison operator `IN` to compare a value v with a set (or multiset) of values V
  - evaluates to TRUE if v is in V.
- If a single attribute in a single tuple is returned, the query result will be a single value.
  - you can use $=$ instead of `IN`.

Q4: Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

[2]

Structured Query Language

```
Q4A:  ( SELECT    DISTINCT Pnumber
        FROM      PROJECT, DEPARTMENT, EMPLOYEE
        WHERE     Dnum=Dnumber AND Mgr_ssn=Ssn
                  AND Lname='Smith' )
        UNION
      ( SELECT    DISTINCT Pnumber
        FROM      PROJECT, WORKS_ON, EMPLOYEE
        WHERE     Pnumber=Pno AND Essn=Ssn
                  AND Lname='Smith' );
```

```
Q4A:  SELECT    DISTINCT Pnumber
      FROM      PROJECT
      WHERE     Pnumber IN
                ( SELECT    Pnumber
                  FROM      PROJECT, DEPARTMENT, EMPLOYEE
                  WHERE     Dnum=Dnumber AND
                            Mgr_ssn=Ssn AND Lname='Smith' )
                OR
                Pnumber IN
                ( SELECT    Pno
                  FROM      WORKS_ON, EMPLOYEE
                  WHERE     Essn=Ssn AND Lname='Smith' );
```

Nested Queries

- SQL allows the use of tuples of values in comparisons by placing them within parentheses.

```
SELECT DISTINCT Essn
FROM WORKS_ON
WHERE (Pno, Hours) IN ( SELECT Pno, Hours
                        FROM WORKS_ON
                        WHERE Essn='123456789' );
```

Other Operators

Other comparison operators can be used to compare a single value v to a set V.

- The = ANY (SOME) operator returns TRUE if the value v is equal to some value in the set V and is hence equivalent to IN.
- Other operators that can be combined with ANY include $>$, $>=$, $<$, $<=$, and $<>$.
- ALL can also be combined with each of these operators.

```
SELECT    Lname, Fname
FROM      EMPLOYEE
WHERE     Salary > ALL  ( SELECT   Salary
                          FROM     EMPLOYEE
                          WHERE    Dno=5 );
```

[a]

Correlated Nested Queries

Nested query condition references attributes of the outer query.

- `EXISTS` and `NOT EXISTS` are used in conjunction with a *correlated* nested query Q.
- `EXISTS(Q)` returns TRUE if there is at least one tuple in the result of Q, FALSE otherwise.
- `NOT EXISTS(Q)` returns TRUE if there are no tuples in the result of Q, FALSE otherwise.

Query: Retrieve names (first, last) of employees with no dependents.

NOT EXISTS

- `NOT EXISTS(Q)` returns TRUE if there are no tuples in the result Q, FALSE otherwise.

Query: Retrieve names (first, last) of employees with no dependents.

Query 6. Retrieve the names of employees who have no dependents.

```
Q6:   SELECT   Fname, Lname
      FROM     EMPLOYEE
      WHERE    NOT EXISTS ( SELECT   *
                            FROM     DEPENDENT
                            WHERE    Ssn=Essn );
```

Aggregate functions

Aggregate functions are used to summarize information from multiple tuples into a single-tuple summary.

- The `COUNT` function returns the number of tuples or values as specified in a query
- `SUM`, `MAX`, `MIN`, and `AVG` can be applied to a set (multi) of numeric values and return, respectively, the sum, maximum value, minimum value, and average (mean) of those values.

Aggregate Functions

Query: Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary.

```
SELECT SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
FROM EMPLOYEE;
```

Query: Find the sum of the salaries of all employees of the Research department, as well as the maximum salary, the minimum salary, and the average salary in this department.

```
SELECT SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
FROM EMPLOYEE, DEPARTMENT
WHERE Dname = 'Research';
```

Grouping

- In many cases we want to apply the *aggregate* functions to subgroups of tuples in a relation, with subgrouping based on some attribute values.
  - find the *average* salary of employees in each department.
- we need to partition the relation into *disjoint* subsets of tuples.
- each group consist of the tuples with the same value of some attribute(s), called *grouping* attribute(s).

Query 24. For each department, retrieve the department number, the number of employees in the department, and their average salary.

```
Q24:    SELECT     Dno, COUNT (*), AVG (Salary)
        FROM       EMPLOYEE
        GROUP BY   Dno;
```

GROUP BY

(a)

Fname	Minit	Lname	Ssn	...	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Result of Q24

HAVING

Query 26. For each project *on which more than two employees work,* retrieve the project number, the project name, and the number of employees who work on the project.

```
Q26:  SELECT    Pnumber, Pname, COUNT (*)
      FROM      PROJECT, WORKS_ON
      WHERE     Pnumber=Pno
      GROUP BY  Pnumber, Pname
      HAVING    COUNT (*) > 2;
```

Pname	Pnumber	...	Essn	Pno	Hours
ProductX	1		123456789	1	32.5
ProductX	1		453453453	1	20.0
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
ProductZ	3		666884444	3	40.0
ProductZ	3		333445555	3	10.0
Computerization	10	...	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

These groups are not selected by the HAVING condition of Q26.

After applying the WHERE clause but before applying HAVING

HAVING

Query 26. For each project *on which more than two employees work,* retrieve the project number, the project name, and the number of employees who work on the project.

```
Q26:    SELECT      Pnumber, Pname, COUNT (*)
        FROM        PROJECT, WORKS_ON
        WHERE       Pnumber=Pno
        GROUP BY    Pnumber, Pname
        HAVING      COUNT (*) > 2;
```

Pname	Pnumber	...	Essn	Pno	Hours
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
Computerization	10		333445555	10	10.0
Computerization	10	...	999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the HAVING clause condition

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Result of Q26
(Pnumber not shown)

Questions?

Thank you!