

Database Design and Development

DATABASE DEVELOPMENT – SQL DML



Data Manipulation Language (DML)

INSERT, DELETE, and UPDATE Statements in SQL

Three
commands
used to
modify the
database:
INSERT,
DELETE,
and
UPDATE.

- **INSERT** typically inserts a tuple (row) in a relation (table)
- **UPDATE** may update a number of tuples (rows) in a relation (table) that satisfy the condition
- **DELETE** may delete a number of tuples (rows) in a relation (table) that satisfy the condition

In its simplest form, it is used to add one or more tuples to a relation

Attribute values should be listed in the same order as the attributes were specified in the **CREATE TABLE** command

Constraints on data types are observed automatically

Any integrity constraints as a part of the DDL specification are enforced

INSERT

Specify the relation name and a list of values for the tuple. All values including nulls are supplied.

```
INSERT INTO      employee

VALUES          ('653298653', 'Richard', 'K', 'Marini','1962-12-30',
'98 Oak Forest, Katy,TX','M', 37000, '987654321', 4 );

INSERT INTO      employee(SSN, fname, mname, lname, birthdate, address,
gender, salary, supervisor_SSN, dept_number)

VALUES          ('653298651', 'Chang', 'L', 'Will', '1987-10-09',
'98 Oak Forest, Katy,TX','M', 35000, '987654321', 4 );
```

The INSERT Command

Removes tuples from a relation

Includes a WHERE-clause to select the tuples to be deleted

Referential integrity should be enforced

Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)

A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table

The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause

Delete

Includes a WHERE clause to select the tuples to be deleted. The number of tuples deleted will vary.

```
DELETE FROM employee  
WHERE         lname = 'Wong';
```

```
DELETE FROM employee  
WHERE         SSN = '123456789';
```

```
DELETE FROM employee  
WHERE         dept_number = 5;
```

```
DELETE FROM employee;
```

The DELETE Command

Used to modify attribute values of one or more selected tuples

A WHERE-clause selects the tuples to be modified

An additional SET-clause specifies the attributes to be modified and their new values

Each command modifies tuples *in the same relation*

Referential integrity specified as part of DDL specification is enforced

Update

Example: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively

```
UPDATE      department
SET         manager_SSN = '653298653', manager_start_date = '1988-05-22'
WHERE       number=1;
```

The UPDATE Command

Basic Retrieval Queries in SQL

SELECT statement

- One basic statement for retrieving information from a database

```
SELECT <attributes list>  
FROM <tables list>  
WHERE <condition>
```

The SELECT-FROM-WHERE Structure of Basic SQL Queries

Logical comparison operators

- =, <, <=, >, >=, and <>

Projection attributes

- Attributes whose values are to be retrieved

Selection condition

- Boolean condition that must be true for any retrieved tuple. Selection conditions include join conditions when multiple relations are involved.

Comments

Comments are used to explain sections of SQL statements, or to prevent execution of SQL statements.

Single line comments

- start with --.
- Any text between -- and the end of the line will be ignored (will not be executed).
 - **SELECT** fname, lname **FROM** employee; -- Employee information

Multi-line comments

- start with /* and end with */
- Any text between /* and */ will be ignored.
 - /*Select first and last name
of all the records
in the employee table:*/
- **SELECT** fname, lname **FROM** employee;

Basic Retrieval Queries

Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

birthdate	address
1965-01-09	731 Fondren, Houston,TX

```
SELECT      birthdate, address
FROM        employee
WHERE       fname = 'John'
           AND mname = 'B'
           AND lname = 'Smith';
```

Use of the Asterisk

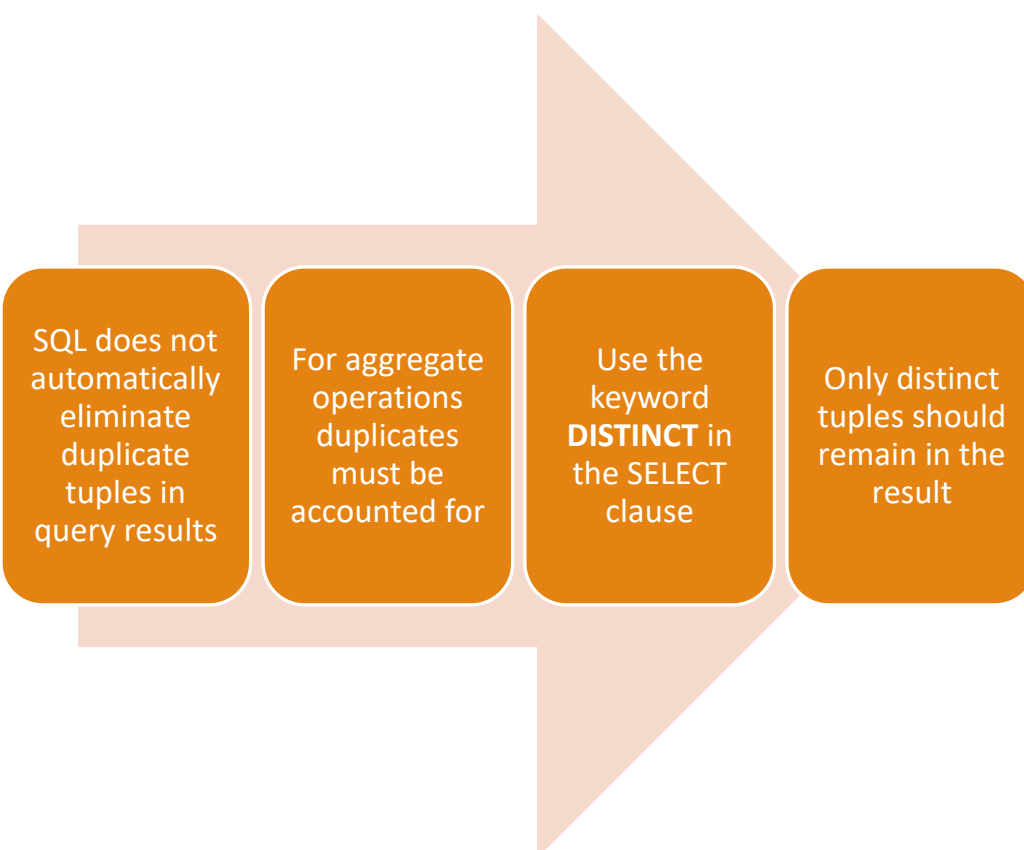


Specify an asterisk (*)

- Retrieve all the attribute values of the selected tuples
- The * can be prefixed by the relation name; e.g., employee.*

```
SELECT          *  
FROM           employee  
WHERE          dept_number = 5;
```

DISTINCT



SQL does not automatically eliminate duplicate tuples in query results

For aggregate operations duplicates must be accounted for

Use the keyword **DISTINCT** in the SELECT clause

Only distinct tuples should remain in the result

Retrieve the salary of every employee.

```
SELECT      salary
FROM        employee;
```

Retrieve all distinct salary values.

```
SELECT      DISTINCT salary
FROM        employee;
```

Explicit Set of values

Can use explicit set
of values in the
WHERE clause

SELECT
FROM
WHERE

DISTINCT employee_SSN
employee_project
project_number
IN (1, 2, 3);

LIKE comparison operator



Used for
string
**pattern
matching**

% replaces
an arbitrary
number of
zero or more
characters

underscore
(_) replaces
a single
character

Examples:

```
SELECT  *  
FROM    employee  
WHERE   address LIKE '%Houston, TX%';
```

```
SELECT  *  
FROM    employee  
WHERE   SSN LIKE '__3__5555';
```

BETWEEN comparison operator

The MySQL BETWEEN Condition is used to retrieve values within a range in a SELECT, INSERT, UPDATE, or DELETE statement.

Syntax:
expression BETWEEN
value1 AND value2;

- Retrieve all employees in department 5 whose salary is between \$30,000 and \$40,000.

```
SELECT      *  
FROM        employee  
WHERE       (Salary BETWEEN  
            30000 AND 40000)  
            AND dept_number = 5;
```

Comparisons Involving NULL



SQL allows queries that check whether an attribute value is NULL

- IS or IS NOT NULL

Retrieve the names of all employees who do not have supervisors.

```
SELECT  fname, lname
FROM    employee
WHERE    supervisor_SSN IS NULL;
```

Limit

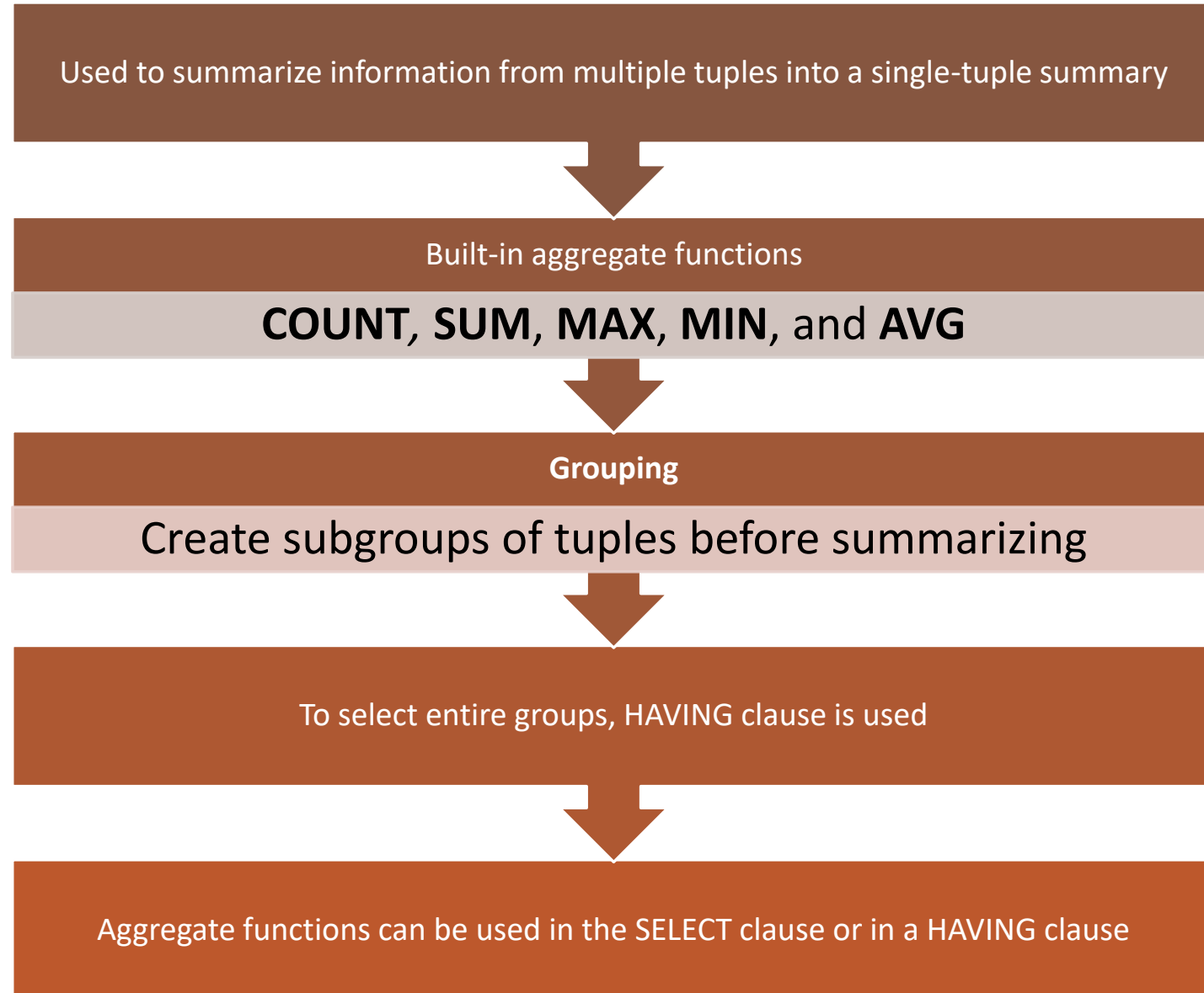
The LIMIT clause is used to specify the number of records to return.

The LIMIT clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

```
SELECT      *  
FROM        employee  
LIMIT       5;
```

Note: LIMIT is used in MySQL to return the number of records whereas SELECT TOP and FETCH FIRST clauses are used in SQL Server and Oracle, respectively.

Aggregate Functions in SQL



Aggregate Functions in SQL

Retrieve the total number of employees in the company

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

Renaming Results of Aggregation

Following query returns a single row of computed values from employee table:

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;
```

The result can be presented with new names:

```
SELECT          SUM(salary) AS Total_Sal, MAX(salary) AS
Highest_Sal, MIN(salary) AS Lowest_Sal,
AVG(salary) AS Average_Sal
FROM            employee;
```

The GROUP BY Clause

Partition relation into subsets of tuples

- Based on **grouping attribute(s)**
- Apply function to each such group independently

GROUP BY clause

- Specifies grouping attributes
- The grouping attribute must appear in the SELECT clause:
- COUNT (*) counts the number of rows in the group

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

```
SELECT
dept_number,
COUNT (*),
AVG (salary)

FROM      employee

GROUP BY  dept_number;
```

Note: If the grouping attribute has NULL as a possible value, then a separate group is created for the null value



Question

Retrieve the supervisor SSN, the number of employees the supervisor supervises, and their sum salary.

Combining the WHERE and the HAVING Clause

Consider the query: we want to count the *total* number of employees whose salaries exceed \$35,000 in each department, but only for departments where more than two employees work.

INCORRECT QUERY:

```
SELECT      dept_number, COUNT (*)
FROM        employee
WHERE       salary>35000
GROUP BY    dept_number
HAVING      COUNT (*) > 2;
```

Combining the WHERE and the HAVING Clause

Correct Specification of the Query:

Note: the WHERE clause applies tuple by tuple whereas HAVING applies to entire group of tuples

For each department that has more than two employees, retrieve the department number and the number of its employees who are making more than \$35,000.

```
SELECT      dept_number, COUNT(*)
FROM        employee
WHERE        salary>35000 AND dept_number IN
              (SELECT      dept_number
FROM        employee
GROUP BY    dept_number
HAVING      COUNT(*) > 2)
GROUP BY    dept_number;
```

EXPANDED Block Structure of SQL Queries

```
SELECT      <attribute and function list>
FROM        <table list>
[WHERE      <condition>]
[GROUP BY   <grouping attribute(s)>]
[HAVING     <group condition>]
[ORDER BY   <attribute list>;
```

Summary of DML SQL Syntax

- INSERT INTO <table name> [(<column name> { , <column name> })]
 (VALUES (<constant value> , { <constant value> }) { , (<constant value> { , <constant value> }) }
 | <select statement>)
- DELETE FROM <table name>
 [WHERE <selection condition>]
- UPDATE <table name>
 SET <column name> = <value expression> { , <column name> = <value expression> }
 [WHERE <selection condition>]

Summary of DML SQL Syntax

- SELECT [DISTINCT] <attribute list>
FROM (<table name> { <alias> } | <joined table>) { , (<table name> { <alias> } | <joined table>) }
[WHERE <condition>]
[GROUP BY <grouping attributes> [HAVING <group selection condition>]]
[ORDER BY <column name> [<order>] { , <column name> [<order>] }]
- <attribute list> ::= (* | (<column name> | <function> (([DISTINCT] <column name> | *))))
{ , (<column name> | <function> (([DISTINCT] <column name> | *))) }
- <grouping attributes> ::= <column name> { , <column name> }
- <order> ::= (ASC | DESC)

References

Elmasri, R., & Navathe, S. (2017). Fundamentals of database systems (Vol. 7). Pearson

Nugent, D. (2017). Higher Nationals in Computing Core Textbook. Pearson Education Custom Content.