

CHAPTER 7

Basic SQL

“Structured Query language”.

SQL Data Definition, Data Types, Standards

- Terminology:
 - **Table**, **row**, and **column** used for relational model terms relation, tuple, and attribute
- CREATE statement
 - Main SQL command for data definition
- Each statement in SQL ends with a **semicolon**

The CREATE TABLE Command in SQL

- Specifying a new relation
 - Provide name of table
 - Specify attributes, their types and initial constraints
- Can optionally specify schema:
 - `CREATE TABLE EMPLOYEE ...`

COMPANY relational database schema

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

One possible database state for the COMPANY relational database schema

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

One possible database state for the COMPANY relational database schema – continued

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

<u>Pname</u>	<u>Pnumber</u>	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

SQL CREATE TABLE data definition statements for defining the COMPANY schema

CREATE TABLE EMPLOYEE

(Fname	VARCHAR(15)	NOT NULL,
Minit	CHAR,	
Lname	VARCHAR(15)	NOT NULL,
Ssn	CHAR(9)	NOT NULL,
Bdate	DATE,	
Address	VARCHAR(30),	
Sex	CHAR,	
Salary	DECIMAL(10,2),	
Super_ssn	CHAR(9),	
Dno	INT	NOT NULL,

PRIMARY KEY (Ssn),

CREATE TABLE DEPARTMENT

(Dname	VARCHAR(15)	NOT NULL,
Dnumber	INT	NOT NULL,
Mgr_ssn	CHAR(9)	NOT NULL,
Mgr_start_date	DATE,	

PRIMARY KEY (Dnumber),

UNIQUE (Dname),

FOREIGN KEY (Mgr_ssn) **REFERENCES** EMPLOYEE(Ssn));

CREATE TABLE DEPT_LOCATIONS

(Dnumber	INT	NOT NULL,
Dlocation	VARCHAR(15)	NOT NULL,

PRIMARY KEY (Dnumber, Dlocation),

FOREIGN KEY (Dnumber) **REFERENCES** DEPARTMENT(Dnumber));

continued on next slide

SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 5.7 (Fig. 6.1)-continued

CREATE TABLE PROJECT

(Pname	VARCHAR(15)	NOT NULL,
Pnumber	INT	NOT NULL,
Plocation	VARCHAR(15),	
Dnum	INT	NOT NULL,

PRIMARY KEY (Pnumber),
UNIQUE (Pname),
FOREIGN KEY (Dnum) **REFERENCES** DEPARTMENT(Dnumber));

CREATE TABLE WORKS_ON

(Essn	CHAR(9)	NOT NULL,
Pno	INT	NOT NULL,
Hours	DECIMAL(3,1)	NOT NULL,

PRIMARY KEY (Essn, Pno),
FOREIGN KEY (Essn) **REFERENCES** EMPLOYEE(Ssn),
FOREIGN KEY (Pno) **REFERENCES** PROJECT(Pnumber));

CREATE TABLE DEPENDENT

(Essn	CHAR(9)	NOT NULL,
Dependent_name	VARCHAR(15)	NOT NULL,
Sex	CHAR,	
Bdate	DATE,	
Relationship	VARCHAR(8),	

PRIMARY KEY (Essn, Dependent_name),
FOREIGN KEY (Essn) **REFERENCES** EMPLOYEE(Ssn));

Attribute Data Types and Domains in SQL

- **Numeric data types**
 - Integer numbers: INTEGER, INT, and SMALLINT
 - Floating-point (real) numbers: FLOAT or REAL, and DOUBLE PRECISION, number (p, s), DECIMAL (p, s)
- **Character-string data types**
 - Fixed length: CHAR (n),
 - Varying length: VARCHAR (n), VARCHAR2 (n)
- **Boolean data type**
 - Values of TRUE or FALSE or NULL
- **DATE data type**
 - form YYYY-MM-DD=2021-02-18
 - Multiple mapping functions (in RDBMSs) to change date formats

Attribute Data Types and Domains in SQL (cont'd.)

- Additional data types
 - **Timestamp** data type
 - Includes the `DATE` and `TIME` fields
 - Optional `WITH TIME ZONE` qualifier
 - **INTERVAL** data type
 - Specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp
 - **DATE, TIME, Timestamp, INTERVAL** data types can be **cast** or converted to string formats for comparison.

Specifying Constraints in SQL

Basic constraints:

- **Key constraint:**
 - A primary key value cannot be duplicated
- **Entity Integrity Constraint:**
 - A primary key value cannot be null
- **Referential integrity constraints :**
 - The “foreign key “ must have a value that is already present as a primary key, or may be null.

Specifying Attribute Constraints

Other Restrictions on attribute domains:

- Default value of an attribute
 - **DEFAULT** <value>
 - **NULL** is not permitted for a particular attribute (**NOT NULL**)
- **CHECK** clause
 - `Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);`

Specifying Key and Referential Integrity Constraints

■ PRIMARY KEY clause

- Specifies one or more attributes that make up the primary key of a relation
- `Dnumber INT PRIMARY KEY;`

■ UNIQUE clause

- Specifies alternate (secondary) keys (called CANDIDATE keys in the relational model).
- `Dname VARCHAR(15) UNIQUE;`

Specifying Key and Referential Integrity Constraints (cont'd.)

- **FOREIGN KEY** clause
 - Default operation: reject update on violation
 - Attach **referential triggered action** clause
 - Options include SET NULL, CASCADE, and SET DEFAULT
 - Action taken by the DBMS for SET NULL or SET DEFAULT is the same for both ON DELETE and ON UPDATE
 - CASCADE option suitable for “relationship” relations

Giving Names to Constraints

- Using the Keyword **CONSTRAINT**
 - Name a constraint
 - Useful for later altering

Default attribute values and referential integrity triggered action specification (Fig. 6.2)

```
CREATE TABLE EMPLOYEE
(
    ...,
    Dno          INT          NOT NULL          DEFAULT 1,
    CONSTRAINT EMPPK
        PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
            ON DELETE SET NULL          ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
            ON DELETE SET DEFAULT      ON UPDATE CASCADE);

CREATE TABLE DEPARTMENT
(
    ...,
    Mgr_ssn CHAR(9)          NOT NULL          DEFAULT '888665555',
    ...,
    CONSTRAINT DEPTPK
        PRIMARY KEY (Dnumber),
    CONSTRAINT DEPTSK
        UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
            ON DELETE SET DEFAULT      ON UPDATE CASCADE);

CREATE TABLE DEPT_LOCATIONS
(
    ...,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
        ON DELETE CASCADE          ON UPDATE CASCADE);
```

Specifying Constraints on Tuples Using CHECK

- Additional Constraints on individual tuples within a relation are also possible using CHECK
- CHECK clauses at the end of a CREATE TABLE statement
 - Apply to each tuple individually
 - `CHECK (Dept_create_date <= Mgr_start_date);`

Basic Retrieval Queries in SQL

- **SELECT** statement
 - One basic statement for retrieving information from a database
- SQL allows a table to have two or more tuples that are identical in all their attribute values
 - Multiset or bag behavior(duplicate)

The SELECT-FROM-WHERE

Structure of Basic SQL Queries

- Basic form of the `SELECT` statement:

```
SELECT    <attribute list>  
FROM      <table list>  
WHERE     <condition>;
```

where

- <attribute list> is a list of attribute names whose values are to be retrieved by the query.
- <table list> is a list of the relation names required to process the query.
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.

The SELECT-FROM-WHERE Structure of Basic SQL Queries (cont'd.)

- Logical comparison operators
 - =, <, <=, >, >=, and <> (different)
- 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.

Basic Retrieval Queries

<u>Bdate</u>	<u>Address</u>
1965-01-09	731 Fondren, Houston, TX

<u>Fname</u>	<u>Lname</u>	<u>Address</u>
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

Query 0. Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

Q0: **SELECT** Bdate, Address
 FROM EMPLOYEE
 WHERE Fname='John' **AND** Minit='B' **AND** Lname='Smith';

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

Q1: **SELECT** Fname, Lname, Address
 FROM EMPLOYEE, DEPARTMENT
 WHERE Dname='Research' **AND** Dnumber=Dno;

Basic Retrieval Queries (Contd.)

(c)

<u>Pnumber</u>	<u>Dnum</u>	<u>Lname</u>	<u>Address</u>	<u>Bdate</u>
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

Q2: **SELECT** Pnumber, Dnum, Lname, Address, Bdate
 FROM PROJECT, DEPARTMENT, EMPLOYEE
 WHERE Dnum=Dnumber **AND** Mgr_ssn=Ssn **AND**
 Plocation='Stafford';

Ambiguous Attribute Names

- Same name can be used for two (or more) attributes in different relations
 - As long as the attributes are in different relations
 - Must **qualify** the attribute name with the relation name to prevent ambiguity

```
Q1A:  SELECT  Fname, EMPLOYEE.Name, Address
        FROM    EMPLOYEE, DEPARTMENT
        WHERE   DEPARTMENT.Name='Research' AND
                DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```

Aliasing, and Renaming

- **Aliases or tuple variables**

- Declare alternative relation names E and S to refer to the EMPLOYEE relation twice in a query:

Query 8. For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.

- **SELECT** E.Fname, E.Lname, S.Fname, S.Lname
FROM EMPLOYEE **AS** E, EMPLOYEE **AS** S
WHERE E.Super_ssn=S.Ssn;

E

ssn	name	Super-ssn
1	A	2
2	B	
3	C	1
4	D	1

S

ssn	name	Super-ssn
1	A	2
2	B	
3	C	1
4	D	1

E.ssn	E.name	E.Super-ssn	S.SSN	S.name	S.superSSN
1	A	2	2	B	
2	B				
3	C	1	1	A	2
4	D	1	1	A	2

Unspecified WHERE Clause and Use of the Asterisk

- Missing WHERE clause
 - Indicates no condition on tuple selection
- Effect is a CROSS PRODUCT
 - Result is Cartesian Product result

Queries 9 and 10. Select all EMPLOYEE Ssns (Q9) and all combinations of EMPLOYEE Ssn and DEPARTMENT Dname (Q10) in the database.

Q9: SELECT Ssn
 FROM EMPLOYEE;

Q10: SELECT Ssn, Dname
 FROM EMPLOYEE, DEPARTMENT;

Unspecified WHERE Clause and Use of the Asterisk (cont'd.)

- Specify an asterisk (*)
 - Retrieve all the attribute values of the selected tuples

Q1C: **SELECT** *

FROM EMPLOYEE

WHERE Dno=5;

Q1D: **SELECT** *

FROM EMPLOYEE, DEPARTMENT

WHERE Dname='Research' **AND** Dno=Dnumber;

Q10A: **SELECT** *

FROM EMPLOYEE, DEPARTMENT;

Tables as Sets in SQL

- SQL does not automatically eliminate duplicate tuples in query results
- Use the keyword **DISTINCT** in the `SELECT` clause
 - Only distinct tuples should remain in the result

Query 11. Retrieve the salary of every employee (Q11) and all distinct salary values (Q11A).

Q11: **SELECT** **ALL** Salary
 FROM **EMPLOYEE;**

Q11A: **SELECT** **DISTINCT** Salary
 FROM **EMPLOYEE;**

Substring Pattern Matching and Arithmetic Operators

- **LIKE** comparison operator

- Used for string **pattern matching**
- % replaces an arbitrary number of zero or more characters
- underscore (_) replaces a single character
- Examples: **WHERE** Address **LIKE** '%Houston,TX%';
- **WHERE** Ssn **LIKE** '__ 1__ 8901';

- **BETWEEN** comparison operator E.g., in Q14 :

WHERE(Salary **BETWEEN** 30000 **AND** 40000)
AND Dno = 5;

Arithmetic Operations

- Standard arithmetic operators:
 - Addition (+), subtraction (−), multiplication (*), and division (/) may be included as a part of **SELECT**
- **Query 13.** Show the resulting salaries if every employee working on the 'ProductX' project is given a 10 percent raise.

```
SELECT E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal  
FROM EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P  
WHERE E.Ssn=W.Essn AND W.Pno=P.Pnumber AND  
P.Pname='ProductX';
```

Ordering of Query Results

- Use **ORDER BY** clause
 - Keyword **DESC**:
 - result in a descending order of values
 - Keyword **ASC**:
 - ascending order explicitly
 - Typically placed at the end of the query

```
SELECT    <attribute list>
FROM      <table list>
[ WHERE   <condition> ]
[ ORDER BY <attribute list> ];
```

INSERT, DELETE, and UPDATE Statements in SQL

- INSERT
 - inserts a tuple (row) in a relation (table)
- UPDATE
 - update a number of tuples (rows) in a relation (table) that satisfy the condition
- DELETE
 - update a number of tuples (rows) in a relation (table) that satisfy the condition
- **Alter** table emp add column ssn int;(Add column to a table)
- **Alter** table emp drop column salary; remove column from a table
- Drop table emp;

The INSERT Command

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

```
U1:    INSERT INTO    EMPLOYEE
        VALUES      ( 'Richard', 'K', 'Marini', '653298653', '1962-12-30', '98
                        Oak Forest, Katy, TX', 'M', 37000, '653298653', 4 );
```

- The variation below inserts multiple tuples where a new table is loaded values from the result of a query.

```
U3B:    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;
```

DELETE

- Removes tuples from a relation
 - Includes a WHERE-clause to select the tuples to be deleted
 - 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 DELETE Command

- Removes tuples from a relation
 - Includes a `WHERE` clause to select the tuples to be deleted. The number of tuples deleted will vary.

U4A:	DELETE FROM	EMPLOYEE
	WHERE	Lname='Brown';
U4B:	DELETE FROM	EMPLOYEE
	WHERE	Ssn='123456789';
U4C:	DELETE FROM	EMPLOYEE
	WHERE	Dno=5;
U4D:	DELETE FROM	EMPLOYEE;

UPDATE

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

U5: UPDATE PROJECT

SET PLOCATION = 'Bellaire', DNUM = 5
WHERE PNUMBER=10;

UPDATE (contd.)

- Example: Give all employees in the 'Research' department a 10% raise in salary.

```
U6: UPDATE      EMPLOYEE
      SET        SALARY = SALARY *1.1
      WHERE      DNO = (SELECT      DNUMBER
                        FROM          DEPARTMENT
                        WHERE          DNAME='Research')
```

- In this request, the modified SALARY value depends on the original SALARY value in each tuple
 - The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification
 - The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification

Nested Queries

- Complete select-from-where blocks within WHERE clause of another query
 - **Outer query and nested subqueries**
- Comparison operator `IN`

```
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 (cont'd.)

- Use tuples of values in comparisons
 - Place them within parentheses

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

Nested Queries (cont'd.)

- Use other comparison operators to compare
 - = ANY (or = 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 (or SOME): >, >=, <, <=, and <>
 - ALL: value must exceed all values from nested query

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

Explicit Sets and Renaming of Attributes in SQL

- Can use explicit set of values in WHERE clause

Q17: SELECT DISTINCT Essn
 FROM WORKS_ON
 WHERE Pno IN (1, 2, 3);

- Use qualifier AS followed by desired new name
 - Rename any attribute that appears in the result of a query

Q8A: SELECT E.Lname AS Employee_name, S.Lname AS Supervisor_name
 FROM EMPLOYEE AS E, EMPLOYEE AS S
 WHERE E.Super_ssn=S.Ssn;

Specifying Joined Tables in the FROM Clause of SQL

- **Joined table**

- Permits users to specify a table resulting from a join operation in the FROM clause of a query

- **The FROM clause in Q1A**

- Contains a single joined table.

```
Q1A:  SELECT  Fname, Lname, Address
      FROM    (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
      WHERE   Dname='Research';
```

NATURAL JOIN

- Rename attributes of one relation so it can be joined with another using NATURAL JOIN:

```
Q1B:    SELECT    Fname, Lname, Address
        FROM      (EMPLOYEE NATURAL JOIN
                   (DEPARTMENT AS DEPT (Dname, Dno, Mssn,
                                         Msdate)))
        WHERE     Dname='Research';
```

The above works with $EMPLOYEE.Dno = DEPT.Dno$ as an implicit join condition

Aggregate Functions in SQL

- Used to summarize information from multiple tuples into a single-tuple summary
- Built-in aggregate functions
 - **COUNT**, **SUM**, **MAX**, **MIN**, and **AVG**
- **Grouping**
 - Create subgroups of tuples before summarizing
- To select entire groups, **HAVING** clause is used
- Aggregate functions can be used in the **SELECT** clause or in a **HAVING** clause

Renaming Results of Aggregation

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

Q19:

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

- The result can be presented with new names:

```
Q19A:      SELECT    SUM (Salary) AS Total_Sal, MAX (Salary) AS
                Highest_Sal, MIN (Salary) AS Lowest_Sal, AVG
                (Salary) AS Average_Sal
FROM    EMPLOYEE;
```

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

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

Grouping: The GROUP BY and HAVING Clauses (cont'd.)

- **HAVING** clause

- Provides a condition to select or reject an entire group:

- **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;
```

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> ];
```

The ALTER table command

- **Alter table actions** include:
 - Adding or dropping a column (attribute)
 - Adding or dropping table constraints
- **Example:**
 - `ALTER TABLE EMPLOYEE ADD Job
VARCHAR(12) ;`
 - `ALTER TABLE EMPLOYEE DROP COLUMN Job;`
 - `ALTER TABLE EMPLOYEE DROP constraint
fk1;`
 - `ALTER TABLE EMPLOYEE ADD constraint
Pk1 primary key(eno);`

Table 7.2 Summary of SQL Syntax

Table 7.2 Summary of SQL Syntax

```
CREATE TABLE <table name> ( <column name> <column type> [ <attribute constraint> ]
                             { , <column name> <column type> [ <attribute constraint> ] }
                             [ <table constraint> { , <table constraint> } ] )
```

```
DROP TABLE <table name>
ALTER TABLE <table name> ADD <column name> <column type>
```

```
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 )
```

```
INSERT INTO <table name> [ ( <column name> { , <column name> } ) ]
( VALUES ( <constant value> , { <constant value> } ) { , ( <constant value> { , <constant value> } ) }
| <select statement> )
```

continued on next slide

Table 7.2 (continued)

Summary of SQL Syntax

Table 7.2 Summary of SQL Syntax

DELETE FROM <table name>

[WHERE <selection condition>]

UPDATE <table name>

SET <column name> = <value expression> { , <column name> = <value expression> }

[WHERE <selection condition>]

CREATE [UNIQUE] INDEX <index name>

ON <table name> (<column name> [<order>] { , <column name> [<order>] })

[CLUSTER]

DROP INDEX <index name>

CREATE VIEW <view name> [(<column name> { , <column name> })]

AS <select statement>

DROP VIEW <view name>

NOTE: The commands for creating and dropping indexes are not part of standard SQL.