Data Manipulate Language

Another major component of SQL (standard relational database language)
Non Procedural language (transform-oriented language)
Command structure using standard English words, free format

DML – retrieve and update data
INSERT
UPDATE
DELETE
SELECT – express a query, most important statement
Insert Clause

Add new rows of data to a table
Two forms, first one to insert a single row

```
INSERT INTO #table_name#
VALUES (v_a1, v_a2, ...);
```

v_a1, v_a2, ... values of attributes in the same order as in original CREATE TABLE order

```
INSERT INTO #table_name# (c1, c2, ...) VALUES (v_a1, v_a2, ...);
```

- Column may be omitted if it is NULL or has a default value
- Number of values must be the same as the number of columns
- The position of the value corresponding to the position of the column in the list
- Data type of each item must be compatible with the data type of the corresponding column
Insert Clause

Student (id, fname, lname, phone, email, adrs, zcode, gender, gpa);
CREATE TABLE STUDENT
  (ID CHAR(8) PRIMARY KEY,
   F_NAME VARCHAR2(30) NOT NULL,
   L_NAME VARCHAR2(30) NOT NULL,
   PHONE CHAR(10),
   EMAIL VARCHAR2(50),
   ADRS VARCHAR2(100) NOT NULL,
   Z_CODE CHAR(5) NOT NULL,
   GENDER CHAR NOT NULL DEFAULT 'M'
   CHECK (GENDER IN ('M','F')),
   GPA NUMBER(2,1) NOT NULL CHECK (GPA BETWEEN 0 AND 4))
);
INSERT INTO STUDENT
VALUES (‘12345678’, ‘John’, ‘Smith’, null, null,
‘123 Main St.’, ‘11355’, ‘M’, 3.1);
Insert Clause

Student (id, fname, lname, phone, email, adrs, zcode, gender, gpa);
CREATE TABLE STUDENT
 (ID CHAR(8) PRIMARY KEY,
  F_NAME VARCHAR2(30) NOT NULL,
  L_NAME VARCHAR2(30) NOT NULL,
  PHONE CHAR(10),
  EMAIL VARCHAR2(50),
  ADRS VARCHAR2(100) NOT NULL,
  Z_CODE CHAR(5) NOT NULL,
  GENDER CHAR NOT NULL DEFAULT 'M'
  CHECK (GENDER IN ('M','F')),
  GPA NUMBER(2,1) NOT NULL
  CHECK (GPA BETWEEN 0 AND 4))
);
INSERT INTO (ID, L_NAME, F_NAME, ADRS, Z_CODE, GPA)
VALUES (‘12345678’, ‘Smith’, ‘John’, ‘123 Main St.’, ‘11355’, 3.1);
Insert Clause

Add new rows of data to a table
The other one to insert multiple rows that copied from another table

INSERT INTO #table_name# [(col_list)]
SELECT ...
Update Clause

Modify data in a table
UPDATE #table_name#
SET col1 = v1 [, col2 = v2...] 
[WHERE #condition#]

Update one more columns
Where is optional that if specified, only the rows that satisfy the condition in WHERE are updated

UPDATE STUDENT
SET GPA = 3.5
WHERE ID = ‘12345678’;
Delete Clause

delete data in a table
DELETE FROM #table_name#
[WHERE #condition#]

Where is optional that if specified, only the rows that satisfy the condition in WHERE are deleted.

If all rows are deleted, the table is still there, it’s just empty.

DELETE FROM STUDENT
WHERE F_NAME = ‘John’;
Select Clause

Retrieve and display data from one or more tables
Capable to perform the equivalent of relational algebra’s Selection, Projection, and Join operations in a single statement

SELECT #col_name# [AS #new_col_name#] [, ...] FROM #table_name#
[WHERE #condition#]

Relational Algebra equivalence
Selection → WHERE Clause
Project → SELECT Clause

AS – column alias
Select Clause

```sql
SELECT #col_name# [AS #new_col_name#] [, ...] FROM #table_name#
[WHERE #condition#]

SELECT ID, CONCAT(F_NAME, ' ', L_NAME) AS NAME, ROUND(GPA,0) AS ROUNDED_GAP
FROM STUDENT;

In oracle, concat only concatenate two string, may use || for concat

CONCAT(CONCAT(F_NAME, ' '), L_NAME)
F_NAME || ' ' || L_NAME

SELECT *
FROM STUDENT
WHERE ID = '12345678';
```
Where Clause

Single comparison
  gpa > 3.5
Compound comparison
  NOT, AND, OR
*Range
  gpa BETWEEN 2.5 AND 3.0
*Set
  gpa IN (3.5, 3.7)
Pattern
  f_name like ‘John%’
Null
Optional Clauses

SELECT [DISTINCT|ALL]
FROM
*WHERE
*GROUP BY
*HAVING
*ORDER BY

DISTINCT – unlike relational algebra, the result rows are not unique, use keyword DISTINCT to remove duplicates.
Order By Clause

The result are not arranged in any particular order. May specifies the order of the output use ORDER BY clause.

First sort by major sort key, it’s not unique, then sort by minor sort key.

The first element in the ORDER BY clause is major sort key, the second element is minor sort key.

Default is in ascending order, may specify ASC or DESC

SELECT *
FROM STUDENT
ORDER BY F_NAME, L_NAME;

Note:
May use alias or column number in order by
Order By Clause

Note:
May use alias or column number in order by

SELECT
  F_NAME AS "First Name", L_NAME AS "Last Name"
FROM STUDENT
ORDER BY "First Name", "Last Name" ;

SELECT  F_NAME AS FName, L_NAME AS LName
FROM STUDENT
ORDER BY 1 DESC, 2 ;
Aggregate Function

Like in Relational Algebra, ISO standard defines 5 aggregate functions
COUNT
SUM
AVG
MIN
MAX
Each take a single column parameter, returns a single value
Besides COUNT(*), each function operates on non-null value.
Default count duplicate rows, may use keyword DISTINCT before column name

GROUP BY and HAVING are to used with aggregate function

Note, no other columns besides the parameter column can be included in SELECT, unless it’s in the GROUP BY CLAUSE
Aggregate Function

SELECT GPA, COUNT(ID) AS STUDENT_COUNT
FROM STUDENT
GROUP BY GPA;

Look for the gpa above 3.0 only

SELECT GPA, COUNT(ID) AS STUDENT_COUNT
FROM STUDENT
WHERE GPA >= 3.0
GROUP BY GPA;

Look for the groups only containing at least 1 student

SELECT GPA, COUNT(ID) AS STUDENT_COUNT
FROM STUDENT
GROUP BY GPA
HAVING COUNT(ID) > 1;
Aggregate Function

What if we want to show each row with a column stating the value from an aggregate function?

Sample:
Print student’s id, his major code, and how many students major in such major?

```
SELECT CUNY_ID, MAJOR, (SELECT COUNT(CUNY_ID) FROM STUDENT WHERE MAJOR = S.MAJOR)
FROM STUDENT S;
```

May use `OVER PARTITION BY CLAUSE`

```
SELECT CUNY_ID, MAJOR, COUNT(CUNY_ID) OVER(PARTITION BY MAJOR)
FROM STUDENT;
```

Note:
Can’t use alias in group by

```
PROCESSING ORDER
FROM \ WHERE \ GROUP BY \ HAVING \ SELECT \ ORDER BY
```
Join

Include more than one table in the FROM clause, with WHERE clause on the matching columns or Key word NATURAL JOIN or USING if they have same column name

Sample:
SELECT D.DEPT_NAME, S.ID
FROM DEPARTMENT D, STUDENT S
WHERE D.DEPT_CODE = S.DEPT

For example if student’dept column rename to dept_code, then

SELECT DEPT_NAME, ID
FROM DEPARTMENT JOIN STUDENT USING DEPT_CODE
Outer Join

SELECT D.DEPT_NAME, S.ID
FROM DEPARTMENT D RIGHT JOIN STUDENT S
ON D.DEPT_CODE = S.DEPT_CODE
Subquery

A SELECT statement embedded in another SELECT statement. Maybe used in WHERE and HAVING clauses of an outer SELECT statement, may even in INSERT, UPDATE, and DELETE

Three types of subquery:

Scalar subquery
  – single column and single row
Row subquery
  – multiple columns but single row
Table subquery
  – one or more columns and multiple rows

Can’t use ORDER BY clause in a subquery
A subquery that returns more than one column may only be used with key word exists
Subquery

CREATE TABLE DEPARTMENT
(DEPT_CODE CHAR(4) PRIMARY KEY,
DEPT_NAME VARCHAR2(50) NOT NULL UNIQUE);

ALTER TABLE STUDENT
ADD DEPT CHAR(4) NOT NULL REFERENCES DEPARTMENT;

Scalar subquery
SELECT DEPT_NAME
FROM DEPARTMENT
WHERE DEPT_CODE = (SELECT DEPT
FROM STUDENT
WHERE ID = '12345678');
Scalar Subquery

SHOW ALL STUDENTS WHOSE GPA ARE HIGHER THAN AVERAGE GPA

SELECT *
FROM STUDENT
WHERE GPA > (SELECT AVG(GPA) FROM STUDENT)
;

NOTE:
WHERE GPA > AVG(GPA) is illegal
Subquery must be on the right hand side of operator
Subquery

SELECT DEPT_NAME
FROM DEPARTMENT
WHERE DEPT_CODE IN (SELECT DISTINCT DEPT
FROM STUDENT
WHERE GPA > (SELECT AVG(GPA)
FROM STUDENT)
);

May find student with max gpa without max
Use keyword ALL
SELECT *
FROM STUDENT S
WHERE GPA > ALL (SELECT GPA
FROM STUDENT
WHERE CUNY_ID <> S.CUNY_ID);

CoRelated subquery
Subquery

```sql
SELECT DEPT_NAME
FROM DEPARTMENT D
WHERE EXISTS (SELECT *
               FROM STUDENT S
               WHERE S.DEPT = D.DEPT_CODE)
;
May use join instead
SELECT DISTINCT DEPT_NAME
FROM DEPARTMENT D, STUDENT S
WHERE S.DEPT = D.DEPT_CODE;
```
Useful queries

SELECT table_name FROM user_tables;

SELECT column_name, data_type
FROM user_tab_columns
WHERE
  UPPER(table_name) = UPPER ('#table_name#');

SELECT * FROM user_constraints
WHERE
  UPPER(table_name) = UPPER ('#table_name#')
  AND constraint_type = 'C';

SELECT column_name FROM USER_cons_columns
WHERE constraint_name = (  
  SELECT constraint_name FROM user_constraints
  WHERE
    UPPER(table_name) = UPPER ('#table_name#')
    AND constraint_type = 'P'
);

DESC #table_name#;
Useful functions

SELECT LTRIM('  HI ', ' ') FROM DUAL;
SELECT LPAD('HI', 5, ' ') FROM DUAL;

SELECT SYSDATE FROM DUAL;
SELECT SYSDATE + 1 FROM DUAL;
SELECT NEXT_DAY(SYSDATE, 'MONDAY') FROM DUAL;
SELECT LAST_DAY(SYSDATE) FROM DUAL;
SELECT TO_CHAR(SYSDATE) FROM DUAL;
SELECT EXTRACT(YEAR FROM SYSDATE) FROM DUAL;
SELECT MONTHS_BETWEEN(SYSDATE, TO_DATE('7-MAY-20')) FROM DUAL;

QUESTION HOW TO CALCULATE AGE?
SELECT FLOOR((SYSDATE - dob) / 365.25) As age,
f_name || ' ' || l_name As “Name” from STUDENT;

SELECT FLOOR(MONTHS_BETWEEN(SYSDATE, dob) / 12) AS Age, f_name || ' ' || l_name As “Name” from STUDENT;