**Structured Query Language (SQL)**

**Example Schema:**

S(SNO, SNAME, CITY, STATUS)

P(PNO, PNAME, COLOR, WEIGHT, CITY)

SP(SNO, PNO, SDATE, QUANTITY)

**SQL:**

o ANSI standard 92 -- more or less portable.

o de facto standard for relational databases.

o Non-faithful implementation of the relational model.

o Many variations exist.

o Relatively easy to learn.

o Essentially non-procedural.

o Include operations for data definition, query and management.

o Two kinds:

o Direct SQL (interactive)

o Embedded SQL

**DDL**

1. Domains:

CREATE DOMAIN domain [AS] data-type

[default-definition] [domain-constraint-definition-part]

o May be shared by several attributes.

o May include default value.

o May include constraints.

o DEFAULT {literal | niladic-function | NULL}

o Example:

CREATE DOMAIN CITIES CHAR(20) DEFAULT '???'

CREATE DOMAIN QUANTITIES SMALLINT DEFAULT 0

CREATE DOMAIN CITIES CHAR(20)

CONSTRAINT CITY\_VALUES

CHECK (VALUE IN ('Houston', 'New York', 'Dallas'))

CREATE DOMAIN CITIES CHAR(20) DEFAULT NULL

CREATE DOMAIN CITIES CHAR(20)

CHECK (VALUE IS NOT NULL)

o Domains may be altered or dropped.

ALTER DOMAIN domain domain-alteration-action

o Actions can to taken to change default and constraint.

o Example:

ALTER DOMAIN CITIES SET DEFAULT 'Unknown'

ALTER DOMAIN CITIES DROP CITY\_VALUES

ALTER DOMAIN CITIES ADD CHECK (VALUE IS NOT NULL)

DROP DOMAIN CITIES RESTRICTED

DROP DOMAIN CITIES CASCADED

o In drop, if the domain is referenced,

o in RESTRICTED, drop fails.

o in CASCADED:

o referencing views and integrity constraints are dropped.

o referencing columns are not dropped: they get the underlying data type, default and integrity constraint directly.

2. Create tables:

CREATE TABLE table-name (base-table-element-commalist)

column-definition ::=

column {data\_type | domain}

[default-definition]

[column-constraint-definition-list]

o Tables are not relations: may contain duplicate rows, columns are ordered left to right.

o Column descriptions:

o types: smallint, integer, decimal(p,q), numeric, etc.

note that ANSI standard does not include date and logical data types.

o default values:

o Constraints:

(1) NOT NULL.

(2) PRIMARY or UNIQUE.

(3) CHECK constraint.

(4) Reference constraint.

Example:

CREATE TABLE SP(SNO SMALLINT NOT NULL,

PNO SMALLINT NOT NULL,

SDATE DATE NOT NULL,

QUANTITY INTEGER,

PRIMARY KEY(SNO, PNO, SDATE),

FOREIGN KEY(PNO) REFERENCES PART,

FOREIGN KEY(SNO) REFERENCES SUPPLIER);

Alternatives in table SP:

CHECK (SNO IS NOT NULL),

QUANTITY INTEGER CHECK (VALUE > 0),

CHECK (QUANTITY > 0)

SDATE INTEGER DEFAULT CURRENT\_DATE,

Others:

CHECK (S.STATUS > 0)

CHECK (P.COLOR <> 'GREEN' OR P.CITY <> 'HOUSTON')

o Tables may be altered or dropped.

ALTER TABLE base-table base-table-alteration-action

DROP TABLE base-table {RESTRICTED | CASCADED}

DROP TABLE S RESTRICTED

3. Views

o Views are virtual tables -- supported by base tables.

o When views are created, only the definition is stored.

Example:

CREATE VIEW GOOD\_SUPPLIER

AS SELECT \*

FROM S

WHERE STATUS >= 5

SELECT SNAME

FROM GOOD\_SUPPLIER

WHERE CITY = 'HOUSTON'

o View operations are translated into base table operations.

o No recursion in view definition.

o Explicit column names may be given if needed (for changes).

o Table expressions can be from more than one tables.

CREATE VIEW CITY\_PAIR(SNO, PNO, CITY)

AS SELECT DISTINCT (S.SNO, P.PNO, P.CITY)

FROM S, P, SP

WHERE (S.CITY = P.CITY)

AND (S.SNO = SP.SNO)

AND (P.PNO = SP.PNO);

o A given view may not be updatable:

UPDATE GOOD\_SUPPLIER

SET S.STATUS = 10

WHERE S.STATUS > 10;

-- Translated to base table operations.

o Updatable conditions:

o table expression only SELECT.

o No DUPLICATE option.

o FROM only one table, which must be a base table or an updatable view.

o Unique mapping of columns to base table columns.

o No nested expression in the WHERE option.

o No Having.

o No Grouped by.

o CHECK OPTION

o only for updatable views.

o To reject update and insert operations to insert new rows satisfying the view definition.

o Example:

INSERT

INTO GOOD\_SUPPLIER (SNO, NAME, STATUS, CITY)

VALUES('S100', 'So and so', 3, 'Houston');

UPDATE GOOD\_SUPPLIER

SET STATUS = 3

WHERE SNO = '10';

o Views may be dropped:

DROP VIEW View {RESTRICTED | CASCADED}

o RESTRICTED option: drop fails if there are other views dependent on the current view.

o CASCADED option: drop all dependent views also.

**DML**

1. The SELECT Command

o for query

o more than the select, project and join operators of relational algebra.

o Basic form:

SELECT <columns>

FROM <relations>

WHERE <condition>;

o Some special operators:

IS NULL

BETWEEN lower\_range AND upper\_range

EXISTS -- not null

IN -- a set/table

LIKES -- inexact string matching.

wild cards: % any and all following

\_ any one character.

SELECT \*

FROM S

WHERE (0 < SNO) AND (SNO < 100)

-- All suppliers with S# between 0 to 100.

SELECT \*

FROM S

WHERE SNO BETWEEN 0 AND 100

-- All suppliers with unknown city names.

SELECT \*

FROM S

WHERE CITY IS NULL

-- All suppliers with name SMITH??????.

SELECT \*

FROM S

WHERE SNAME LIKE "SMITH%"

-- All suppliers with name SMITH??????: better

SELECT \*

FROM S

WHERE UPPER(SNAME) LIKE "SMITH%"

-- All supplier info for 10, 20, 30 and 40.

SELECT \*

FROM S

WHERE SNO IN (10, 20, 30, 40)

o IN allows subqueries.

o Examples of subqueries.

o GROUP BY and HAVING

o Allow aggregate functions (group functions, column functions) to be performed by the groups defined.

o one row for each group.

o HAVING to GROUP is like WHERE to SELECT: specifying conditions for group.

o Example aggregate functions: MAX, MIN, AVG, COUNT, SUM

o Using GROUP BY, the columns of the SELECT clause can only have:

(1) Columns named in GROUP BY.

(2) Aggregate functions on other columns in the tables.

(3) Constant expressions.

Examples:

SELECT CITY, 'Number of suppliers =', COUNT(\*),

'Average Status = ', AVG(STATUS)

FROM S

GROUP BY CITY;

SELECT COLOR, 'Number of parts', COUNT(\*),

'Average Weight = ', AVG(WEIGHT)

FROM S

GROUP BY COLOR

HAVING CITY = 'Houston';

Example:

Sales(Sale\_No, DeptId, Item\_Id, S\_Date, Amount):

List the total amount of sales and its percentage of total sales made by each department.

INSERT INTO SALE\_SUMMARY(TOTAL\_SALE)

SELECT SUM(AMOUNT)

FROM SALES;

SELECT S.DEPTID, S.SUM(AVG), 'Percentage of total = ',

S.SUM(AVG)/T.TOTAL\_SALE \* 100.0

FROM SALE S, SALE\_SUMMARY T

GROUP BY S.DEPT;

o Many SQL allows ORDER BY:

SELECT \*

FROM S

ORDER BY SNO

SELECT \*

FROM SP

ORDER BY QUANTITY DESC

2. UNION, DIFFERENCE, INTERSECT

o DIFFERENCE -- EXCEPT

o Syntax: options:

(a) CORRESPONDING BY (colmnn-commalist): only specified columns are selected.

(b) CORRESPONDING: only common attributes are selected.

(c) default: by ordinal position.

o Examples:

A report showing number of employees in a department with its name:

-- Error: a department with no employees is not shown.

SELECT D.DEPTNO, D.DNAME, COUNT(\*)

FROM DEPT D, EMPLOYEE E

WHERE D.DEPTNO = E.DEPTNO

GROUP BY D.DEPTNO, D.NAME;

-- Correct solution

(SELECT D.DEPTNO, D.DNAME, COUNT(\*)

FROM DEPT D, EMPLOYEE E

WHERE D.DEPTNO = E.DEPTNO

GROUP BY D.DEPTNO, D.NAME)

UNION

(SELECT D.DEPTNO, D.DNAME, 0

FROM DEPT D

WHERE DEPTNO NOT IN

(SELECT DISTINCT DEPTNO IN EMP));

3. JOINS

o Cartesian product: use SELECT.

o table-reference [NATURAL] [join-type] JOIN

table-reference

[ON conditional-expression

| USING (column-commalist) ]

join-type ::= INNER

| LEFT [OUTER]

| RIGHT [OUTER]

| FULL [OUTER]

| UNION

o A JOIN B ON C

Theta-Join

SELECT \*

FROM A, B

WHERE C

o A NATURAL JOIN B

A |x| B

o A JOIN B USING (C1, C2, C3)

Equi-join of A and B on C1, C2 and C3 and then remove duplication of C1, C2 and C3.

o A LEFT JOIN OUTER B ON C

approximately:

SELECT A.\*, B.\*

FROM A, B

WHERE C

UNION ALL

SELECT A.\*, NULL, NULL, ..., NULL

FROM A

WHERE A.\* NOT IN (SELECT A.\*

FROM A, B

WHERE C)

o A UNION JOIN B

approximately:

SELECT A.\*, NULL, NULL, ..., NULL

FROM A

UNION ALL

SELECT A.\*, NULL, NULL, ..., NULL

FROM A

4. INSERT, UPDATE, DELETE

o INSERT INTO table [(column-commalist)] source

o Examples:

INSERT

INTO S

VALUES(100, 'K&K', 'Houston', 5)

INSERT

INTO S

VALUES(:SNO\_VAR, :SNAME\_VAR, :SNAME\_CITY, DEFAULT)

INSERT

INTO TEMP(SNO, CITY)

SELECT SNO, CITY

FROM S

WHERE STATUS > 5

o UPDATE table

SET assignment-commalist

[WHERE conditional-expression]

UPDATE S

SET STATUS = 10, CITY = 'HOUSTON'

WHERE SNO = 18

o DELETE

FROM table

[WHERE conditional-expression]

DELETE FROM S

WHERE SNO = 20

DELETE FROM S

WHERE CITY = 'DALLAS'

**Security**

o Views to hide sensitive data from unauthorized users.

o Access rules:

o define privilege for accessing data objects and operations.

o GRANT and REVOKE by the schema owner.

o Authorization Identifiers (authID): users that can grant and revoke privilege.

**Embedded SQL**

o SQL may be embedded in a host language.

o Most host languages do not support a relation data type directly.

o Stepping through each row of a table: a cursor.

o A cursor: a pointer to run through an ordered collection of rows.

o A cursor is positioned in a row.

o Host language parameters used in embedded SQL:

o must be declared in EXEC SQL BEGIN and END DECLARED SECTION.

o must be prefixed by : when used.

o Cursor's definition:

DECLARE cursor [INSENSITIVE] [SCROLL] CURSOR

FOR cursor-specification

cursor-specification ::=

table-expression

[ORDER BY order-item-commalist]

[FOR {READ ONLY | UPDATE [OF column-commalist]}]

order-item ::=

{column | unsigned-integer} [ASC | DESC]

o INSENSITIVE: insensitive to other concurrent changes

separate copy of table made.

o SCROLL: all form of FETCH allowed (otherwise, only FETCH NEXT allowed).

o READ ONLY: UPDATE CURRENT and DELETE CURRENT not allowed.

Example:

DECLARE X CURSOR

FOR SELECT SP.PNO, AVG(QUANTITY) AS AVG\_Q

FROM SP

GROUP BY PNO

ORDER BY AVG\_Q

o Cursor's operations:

o OPEN

o FETCH

o UPDATE

o DELETE

o CLOSE

OPEN cursor

o OPEN C

o evaluate table expression

o Put C into open state, and

o position cursor to the first row.

FETCH [[row-selector] FROM] cursor INTO target-commalist

row-selector ::=

NEXT | PRIOR | FIRST | LAST

| ABSOLUTE n | RELATIVE n

o NEXT is the default.

Example:

FETCH NEXT X INTO :PNO\_PARA, :AVG\_Q\_PARA;

UPDATE table

SET assignment-commalist

CURRENT OF cursor

Example:

UPDATE SP

SET QUANTITY = QUANTITY + :INC\_PARA

CURRENT OF X;

o Cannot update column mentioned in the GROUP BY clause in the cursor definition.

o

DELETE FROM table

WHERE CURRENT OF cursor

Example:

DELETE FROM SP

WHERE CURRENT OF X;

CLOSE cursor

o CLOSE X: put S into the close state.

o Host language program must include SQLCODE (integer) and/or SQLSTATE (a 5 char string)

o The WHENEVER statement can be used to test SQL statement execution results:

EXEC SQL WHENEVER condition action terminator

condition ::= {NOT FOUND | SQLERROR}

action ::= {CONTINUE | GO TO label}

**Some Important SQL Commands**

**DDL**

CREATE DOMAIN domain [AS] data-type

[default-definition] [domain-constraint-definition-part]

ALTER DOMAIN domain domain-alteration-action

DROP DOMAIN domain {RESTRICTED | CASCADED}

CREATE TABLE base-table (base-table-element-commalist)

column-definition ::=

column {data\_type | domain}

[default-definition]

[column-constraint-definition-list]

ALTER TABLE base-table base-table-alteration-action

DROP TABLE base-table {RESTRICTED | CASCADED}

CREATE VIEW View [(column-commalist)]

AS table-expression

[WITH [CASCADED | LOCAL] CHECK OPTION]

DROP VIEW View {RESTRICTED | CASCADED}

**Select:**

select-expression

::= SELECT [ALL | DISTINCT] select-item-commalist

[INTO target\_commalist]

FROM table-reference-commalist

[ WHERE conditional-expression ]

[GROUP BY column-commalist

[HAVING conditional-expression]]

**Union, Difference, Intersect**:

nonjoin-table-expression

::= nonjoin-table-term

| table-expression {UNION | EXCEPT} [ALL]

[ CORRESPONDING [BY (column-commalist)]] table-term

nonjoin-table-term

::= table-expression INTERSECT [ALL]

[ CORRESPONDING [BY (column-commalist)]] table-primary

**Joins**

table-reference [NATURAL] [join-type] JOIN

table-reference

[ON conditional-expression

| USING (column-commalist) ]

join-type ::= INNER | LEFT [OUTER] | RIGHT [OUTER]

| FULL [OUTER] | UNION

**Insert, Update, Delete**

INSERT INTO table [(column-commalist)] source

UPDATE table

SET assignment-commalist

[WHERE conditional-expression]

DELETE

FROM table

[WHERE conditional-expression]

**Embedded SQL Commands**

DECLARE cursor [INSENSITIVE] [SCROLL] CURSOR

FOR cursor-specification

cursor-specification ::=

table-expression

[ORDER BY order-item-commalist]

[FOR {READ ONLY | UPDATE [OF column-commalist]}]

order-item ::= {column | unsigned-integer} [ASC | DESC]

FETCH [[row-selector] FROM] cursor INTO target-commalist

row-selector ::= NEXT | PRIOR | FIRST | LAST

| ABSOLUTE n | RELATIVE n

UPDATE table

SET assignment-commalist

CURRENT OF cursor

DELETE FROM table WHERE CURRENT OF cursor

CLOSE cursor

EXEC SQL WHENEVER condition action terminator

condition ::= {NOT FOUND | SQLERROR}

action ::= {CONTINUE | GO TO label}

**Examples Of SQL DDL Commands**

1. Domains

CREATE DOMAIN CITIES CHAR(20) DEFAULT '???'

CREATE DOMAIN QUANTITIES SMALLINT DEFAULT 0

CREATE DOMAIN CITIES CHAR(20)

CHECK (VALUE IN ('Houston', 'New York', 'Dallas'))

ALTER DOMAIN CITIES SET DEFAULT 'Unknown'

ALTER DOMAIN CITIES DROP CITY\_VALUES

ALTER DOMAIN CITIES ADD CHECK (VALUE IS NOT NULL)

DROP DOMAIN CITIES RESTRICTED

DROP DOMAIN CITIES CASCADED

2. Tables

CREATE TABLE SP(SNO SMALLINT NOT NULL,

PNO SMALLINT NOT NULL,

SDATE DATE NOT NULL,

QUANTITY INTEGER,

PRIMARY KEY(SNO, PNO),

FOREIGN KEY(PNO) REFERENCES PART,

FOREIGN KEY(SNO) REFERENCES SUPPLIER)

DROP TABLE S RESTRICTED

3. Views

CREATE VIEW GOOD\_SUPPLIER

AS SELECT \*

FROM S

WHERE STATUS >= 5

SELECT SNAME

FROM GOOD\_SUPPLIER

WHERE CITY = 'HOUSTON'

CREATE VIEW CITY\_PAIR(SNO, PNO, CITY)

AS SELECT DISTINCT (S.SNO, P.PNO, P.CITY)

FROM S, P, SP

WHERE (S.CITY = P.CITY)

AND (S.SNO = SP.SNO)

AND (P.PNO = SP.PNO)

DROP VIEW CITY\_PAIR CASCADED

4. SELECT

SELECT CITY, 'Number of suppliers =', COUNT(\*),

'Average Status = ', AVG(STATUS)

FROM S

GROUP BY CITY

SELECT COLOR, 'Number of parts', COUNT(\*),

'Average Weight = ', AVG(WEIGHT)

FROM S

GROUP BY COLOR

HAVING CITY = 'Houston'

A report showing number of employees in a department with its name:

-- Error: a department with no employees is not shown.

SELECT D.DEPTNO, D.DNAME, COUNT(\*)

FROM DEPT D, EMPLOYEE E

WHERE D.DEPTNO = E.DEPTNO

GROUP BY D.DEPTNO, D.NAME;

-- Correct solution

(SELECT D.DEPTNO, D.DNAME, COUNT(\*)

FROM DEPT D, EMPLOYEE E

WHERE D.DEPTNO = E.DEPTNO

GROUP BY D.DEPTNO, D.NAME)

UNION

(SELECT D.DEPTNO, D.DNAME, 0

FROM DEPT D

WHERE DEPTNO NOT IN

(SELECT DISTINCT DEPTNO IN EMP));

5. Others

A LEFT JOIN OUTER B ON C

(approximately: SELECT A.\*, B.\* FROM A, B WHERE C)

UNION ALL

(approximately: SELECT A.\*, NULL, NULL, ..., NULL

FROM A

WHERE A.\* NOT IN (SELECT A.\* FROM A, B WHERE C))

INSERT INTO S VALUES(100, 'K&K', 'Houston', 5)

INSERT INTO S

VALUES(:SNO\_VAR, :SNAME\_VAR, :CITY\_VAR, DEFAULT)

INSERT INTO TEMP(SNO, CITY)

SELECT SNO, CITY

FROM S

WHERE STATUES > 10