Chapter 7

SQL: Data Definition

Chapter 7 - Objectives

- How to create and drop tables and views using SQL.
- Purpose of integrity enhancement feature of SQL.
- How to define integrity constraints using SQL.
- How to use the integrity enhancement feature in the CREATE and ALTER TABLE statements.
- Restrictions on Views
- Under what conditions views are updatable.

Data Definition

- SQL DDL allows database objects such as schemas, domains, tables, views, and indexes to be created and destroyed.
- Main SQL DDL statements are:

CREATE SCHEMA	DROP SCHEMA
CREATE/ALTER DOMAIN	DROP DOMAIN
CREATE/ALTER TABLE	DROP TABLE
CREATE VIEW	DROP VIEW

- Many DBMSs also provide:

CREATE INDEX DROP INDEX

Data Definition

- Relations and other database objects exist in an *environment*.
- Each environment contains one or more *catalogs*, and each catalog consists of set of *schemas*.
- Schema is named collection of related database *objects*.
- Objects in a schema can be tables, views, domains, assertions and character sets. All have same owner.

CREATE SCHEMA

CREATE SCHEMA [Name |
AUTHORIZATION CreatorId]
DROP SCHEMA Name [RESTRICT | CASCADE]

- **With RESTRICT (default), schema must be empty or operation fails.**
- **With CASCADE, operation cascades to drop all objects associated with schema in order defined above. If any of these operations fail, DROP SCHEMA fails.**

CREATE TABLE

```
CREATE TABLE TableName
{(colName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption]
[CHECK searchCondition] [,...]}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listOfColumns),] […,]}
{[FOREIGN KEY (listOfFKColumns)
  REFERENCES ParentTableName [(listOfCKColumns)],
  [ON UPDATE referentialAction]
  [ON DELETE referentialAction ]] [,…]}
 {[CHECK (searchCondition)] [,…] })
```

CREATE TABLE

- **Creates a table with one or more columns of the specified *dataType s (domains)*.**
  - CREATE DOMAIN PNumber AS VARCHAR(5);
- **With NOT NULL, system rejects any attempt to insert a null in the column.**
- **Can specify a DEFAULT value for the column.**
- **Primary keys should always be specified as NOT NULL.**
- **FOREIGN KEY clause specifies FK along with the referential action.**

Example 7.1 - CREATE TABLE

```
CREATE TABLE PropertyForRent (
    propertyNo  PNumber   NOT NULL, ….
    rooms    PRooms       NOT NULL  DEFAULT 4,
    rent     PRent        NOT NULL, DEFAULT 600,
    ownerNo     OwnerNumber          NOT NULL,
    staffNo     StaffNumber
                Constraint StaffNotHandlingTooMuch ….
    branchNo    BranchNumber         NOT NULL,
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff
     ON DELETE SET NULL ON UPDATE CASCADE ….);
```

ALTER TABLE

- **Add a new column to a table.**
- **Drop a column from a table.**
- **Add a new table constraint.**
- **Drop a table constraint.**
- **Set a default for a column.**
- **Drop a default for a column.**

Example 7.2(a) - ALTER TABLE

Change Staff table by removing default of 'Assistant' for position column and setting default for sex column to female ('F').

```
ALTER TABLE Staff
        ALTER position DROP DEFAULT;
ALTER TABLE Staff
        ALTER sex SET DEFAULT 'F';
```

Example 7.2(b) - ALTER TABLE

Remove constraint from PropertyForRent that staff are not allowed to handle more than 100 properties at a time. Add new column to Client table.

```
ALTER TABLE PropertyForRent
    DROP CONSTRAINT StaffNotHandlingTooMuch;
ALTER TABLE Client
    ADD prefNoRooms PRooms;
```

DROP TABLE

DROP TABLE TableName [RESTRICT | CASCADE]

e.g. DROP TABLE PropertyForRent;

- **Removes named table and all rows within it.**
- **With RESTRICT, if any other objects depend for their existence on continued existence of this table, SQL does not allow request.**
- **With CASCADE, SQL drops all dependent objects (and objects dependent on these objects).**

ISO SQL Data Types

Table 6.1 ISO SQL data types.

Data type	Declarations			
boolean	BOOLEAN			
character	CHAR	VARCHAR		
bit	BIT	BIT VARYING		
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION	
datetime	DATE	TIME	TIMESTAMP	
interval	INTERVAL			
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT	

Integrity Enhancement Feature

- **Consider five types of integrity constraints:**
  - **required data (not null)**
  - **domain constraints (check)**
  - **entity integrity**
  - **referential integrity**
  - **general constraints.**

Integrity Enhancement Feature

Required Data

position VARCHAR(10) NOT NULL

Domain Constraints

(a) CHECK

sex CHAR NOT NULL

CHECK (sex IN ('M', 'F'))

Integrity Enhancement Feature

(b) CREATE DOMAIN

CREATE DOMAIN DomainName [AS] dataType
[DEFAULT defaultOption]
[CHECK (searchCondition)]

For example:

CREATE DOMAIN SexType AS CHAR
CHECK (VALUE IN ('M', 'F'));

sex SexType NOT NULL

Integrity Enhancement Feature

- ***searchCondition*** **can involve a table lookup:**

```
CREATE DOMAIN BranchNo AS CHAR(4)
CHECK (VALUE IN (SELECT branchNo
                 FROM Branch));
```

- **Domains can be removed using DROP DOMAIN:**

```
DROP DOMAIN DomainName
  [RESTRICT | CASCADE]
```

IEF - Entity Integrity

- **Primary key of a table must contain a unique, non-null value for each row.**
- **ISO standard supports FOREIGN KEY clause in CREATE and ALTER TABLE statements:**

  PRIMARY KEY(staffNo)

  PRIMARY KEY(clientNo, propertyNo)

- **Can only have one PRIMARY KEY clause per table. Can still ensure uniqueness for alternate keys using UNIQUE:**

  UNIQUE(telNo)

IEF - Referential Integrity

- FK is column or set of columns that links each row in child table containing foreign FK to row of parent table containing matching PK.
- Referential integrity means that, if FK contains a value, that value must refer to existing row in parent table.
- ISO standard supports definition of FKs with FOREIGN KEY clause in CREATE and ALTER TABLE:

FOREIGN KEY(branchNo) REFERENCES Branch

IEF - Referential Integrity

- **Any INSERT/UPDATE attempting to create FK value in child table without matching PK value in parent is rejected.**
- **Action taken attempting to update/delete a PK value in parent table with matching rows in child is dependent on referential action specified using ON UPDATE and ON DELETE subclauses:**

  – **CASCADE** - **SET NULL**

  – **SET DEFAULT** - **NO ACTION**

IEF - Referential Integrity

CASCADE: Delete row from parent and delete matching rows in child, and so on in cascading manner.

SET NULL: Delete row from parent and set FK column(s) in child to NULL. Only valid if FK columns are NOT NULL.

SET DEFAULT: Delete row from parent and set each component of FK in child to specified default. Only valid if DEFAULT specified for FK columns.

NO ACTION: Reject delete from parent. Default.

IEF - Referential Integrity

FOREIGN KEY (staffNo) REFERENCES Staff ON DELETE SET NULL

FOREIGN KEY (ownerNo) REFERENCES Owner ON UPDATE CASCADE

IEF - General Constraints

- Could use CHECK/UNIQUE in CREATE and ALTER TABLE.
- Similar to the CHECK clause, also have:

```
CREATE ASSERTION AssertionName
CHECK (searchCondition)
```

IEF - General Constraints

```
CREATE ASSERTION StaffNotHandlingTooMuch
CHECK (NOT EXISTS (SELECT staffNo
                   FROM PropertyForRent
                   GROUP BY staffNo
                   HAVING COUNT(*) > 100))
```

Views

- A view is a virtual relation that does not necessarily actually exist in the database but is produced upon request, at time of request.

- Contents of a view are defined as a query on one or more base relations.

SQL - CREATE VIEW

CREATE VIEW ViewName [(newColumnName [,...])]
AS subselect
[WITH [CASCADED | LOCAL] CHECK OPTION]

- **Can assign a name to each column in view.**
- **If list of column names is specified, it must have same number of items as number of columns produced by *subselect*.**
- **If omitted, each column takes name of corresponding column in subselect.**

SQL - CREATE VIEW

- **WITH CHECK OPTION ensures that if a row fails to satisfy WHERE clause of defining query, it is not added to underlying base table.**
- **Need SELECT privilege on all tables referenced in subselect and USAGE privilege on any domains used in referenced columns.**

Example 7.3 - Create Horizontal View

Create view so that manager at branch B003 can only see details for staff who work in his or her office.

```
CREATE VIEW Manager3Staff
AS      SELECT *
        FROM Staff
        WHERE branchNo = 'B003';
```

Table 6.3 Data for view Manager3Staff.

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003

Example 7.5 - Grouped and Joined Views

Create view of staff who manage properties for rent, including branch number they work at, staff number, and number of properties they manage.

```
CREATE VIEW StaffPropCnt (branchNo, staffNo, cnt)
AS SELECT s.branchNo, s.staffNo, COUNT(*)
    FROM Staff s, PropertyForRent p
    WHERE s.staffNo = p.staffNo
    GROUP BY s.branchNo, s.staffNo;
```

Example 7.3 - Grouped and Joined Views

Table 6.5 Data for view StaffPropCnt.

branchNo	staffNo	cnt
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

SQL - DROP VIEW

DROP VIEW ViewName [RESTRICT | CASCADE]

- **Causes definition of view to be deleted from database.**
- **For example:**

DROP VIEW Manager3Staff;

SQL - DROP VIEW

- **With CASCADE, all related dependent objects are deleted; i.e. any views defined on view being dropped.**

- **With RESTRICT (default), if any other objects depend for their existence on continued existence of view being dropped, command is rejected.**

Restrictions on Views

SQL imposes several restrictions on creation and use of views.

(a) If column in view is based on an aggregate function:

- **Column may appear only in SELECT and ORDER BY clauses of queries that access view.**
- **Column may not be used in WHERE nor be an argument to an aggregate function in any query based on view.**

Restrictions on Views

- For example, following query would fail:

```
SELECT COUNT(cnt)
FROM StaffPropCnt;
```

- Similarly, following query would also fail:

```
SELECT *
FROM StaffPropCnt
WHERE cnt > 2;
```

Restrictions on Views

(b) Grouped view may never be joined with a base table or a view.

- **For example, StaffPropCnt view is a grouped view, so any attempt to join this view with another table or view fails.**

View Updatability

- All updates to base table reflected in all views that encompass base table.
- Similarly, may expect that if view is updated then base table(s) will reflect change.

View Updatability

- **ISO specifies that a view is updatable if and only if:**

  - DISTINCT is not specified.

  - Every element in SELECT list of defining query is a column name and no column appears more than once.

  - FROM clause specifies only one table, excluding any views based on a join, union, intersection or difference.

  - No nested SELECT referencing outer table.

  - No GROUP BY or HAVING clause.

  - Also, every row added through view must not violate integrity constraints of base table.