Chapter 7

SQL – Data Definition

Chapter 7 - Objectives

- Data types supported by SQL standard.
- 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.

Chapter 7 - Objectives

- Purpose of views.
- How to create and delete views using SQL.
- How the DBMS performs operations on views.
- Under what conditions views are updatable.
- Advantages and disadvantages of views.
- How the ISO transaction model works.
- How to use the GRANT and REVOKE statements as a level of security.

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, collations, translations, 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*.
- 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  VARCHAR(5),
    rooms           SMALLINT,
    rent            DECIMAL (6,2),
    ownerNo         VARCHAR(5),
    staffNo         VARCHAR(5),
    branchNo        CHAR(4),
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff);
```

ISO SQL Data Types

DATA TYPE	DECLARATIONS				
boolean	BOOLEAN				
character	CHAR	VARCHAR			
bit[1]	BIT	BIT VARYING			
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT	BIGINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION		
datetime	DATE	TIME	TIMESTAMP		
interval	INTERVAL				
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT		

[1]BIT and BIT VARYING have been removed from the SQL:2003 standard.

CHAR or VARCHAR

- Data to store ‘ABCD’
  - CHAR(4) 4 bytes
  - VARCHAR(4) 5-6 bytes (1-2 bytes eos)
  - CHAR (200) 200 bytes
  - VARCHAR(200) 5-6 bytes
- So, use CHAR for short, fixed length items (e.g., SSN)
  - Get benefits of data type checking on insert
- If input string is too long, get error message (STRICT SQL enabled) or truncation (STRICT not enabled)

ENUMERATIONS

- In MySQL, (if invalid insert, '' empty string is inserted instead)

```
CREATE TABLE Beverages (
    Name VARCHAR (20)
    Price DECIMAL(4,2)
    Size ENUM ('venti', 'grande', 'tall')
    );
```

- In Oracle
  - CREATE TABLE Beverages (
    - Name VARCHAR (20)
    - Price DECIMAL(4,2)
    - Size VARCHAR (6) CHECK Size IN ('venti', 'grande', 'tall')
    -);

Integrity Enhancement Feature

- Consider five types of integrity constraints:
  - required data
  - domain constraints
  - entity integrity
  - referential integrity
  - general constraints.

DECIMAL/NUMERIC

- DECIMAL (precision, scale)
  - DECIMAL (5,2)
    - 123 -> 123.00
    - ~same as NUMERIC
    - Too many digits prior to '.'
      - Throws an error (e.g., 1234)
    - Too many digits after '.'
      - Truncates (e.g., 123.456 -> 123.45)

Integrity Enhancement Feature

Required Data

position VARCHAR(10) NOT NULL

Domain Constraints

(a)CHECK

```
student_type   CHAR NOT NULL
                        CHECK  (student_type  IN  ('U',
    'G'))
```

Integrity Enhancement Feature

(b) CREATE DOMAIN

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

For example:

```
CREATE DOMAIN StudentType AS CHAR
        CHECK (VALUE IN ('U', 'G'));
```

In CREATE TABLE Student:

```
student_type    StudentType    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];
// RESTRICT: only if empty; CASCADE: drop column
```

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 CK value in parent is rejected.
- Action taken when update/delete a CK 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.

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));
```

Example 7.1 - CREATE TABLE

```
CREATE DOMAIN OwnerNumber AS VARCHAR(5)
  CHECK (VALUE IN (SELECT ownerNo FROM PrivateOwner));
CREATE DOMAIN StaffNumber AS VARCHAR(5)
  CHECK (VALUE IN (SELECT staffNo FROM Staff));
CREATE DOMAIN PNumber AS VARCHAR(5);
CREATE DOMAIN PRooms AS SMALLINT;
        CHECK(VALUE BETWEEN 1 AND 15);
CREATE DOMAIN PRent AS DECIMAL(6,2)
        CHECK(VALUE BETWEEN 0 AND 9999.99);
```

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 gender 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).

Views

View

Dynamic result of one or more relational operations operating on base relations to produce another relation.

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

Views

- Contents of a view are defined as a query on one or more base relations.
- With view resolution, any operations on view are automatically translated into operations on relations from which it is derived.
- With view materialization, the view is stored as a temporary table, which is maintained as the underlying base tables are updated.

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

- List must be specified if there is any ambiguity in a column name.
- The *subselect* is known as the defining query.
- 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.

"Horizontal" i.e., a subset of the rows

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

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 7.4 - Create Vertical View

Create view of staff details at branch B003 excluding salaries.

Subset of columns (also example of dependent View);

```
CREATE VIEW Staff3Staff
AS SELECT staffNo, fName, lName, position, sex
    FROM Manager3Staff;
```

staffNo	fName	lName	position	sex
SG37	Ann	Beech	Assistant	F
SG14	David	Ford	Supervisor	M
SG5	Susan	Brand	Manager	F

SQL - DROP VIEW

DROP VIEW ViewName [RESTRICT | CASCADE]

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

DROP VIEW Manager3Staff RESTRICT;

- should not be allowed due to Staff3Staff;

DROP VIEW Manager3Staff; (Default: CASCADE?]

DROP VIEW Manager3Staff CASCADE;

- will also drop Staff3Staff;

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.

SQL - DROP VIEW

DROP VIEW Manager3Staff;

- should drop Manager3Staff, leave Staff3Staff

DROP VIEW Manager3Staff RESTRICT;

- should not be allowed due to Staff3Staff;

DROP VIEW Manager3Staff CASCADE;

- will also drop Staff3Staff;

NOTE: In mysql, CASCADE/RESTRICT are parsed and ignored:

https://dev.mysql.com/doc/refman/8.0/en/drop-view.html

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

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

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

- However, consider again view StaffPropCnt.
- If we tried to insert record showing that at branch B003, SG5 manages 2 properties:

```
INSERT INTO StaffPropCnt
VALUES ('B003', 'SG5', 2);
```

- Have to insert 2 records into PropertyForRent showing which properties SG5 manages. However, do not know which properties they are; i.e. do not know primary keys! So, this is not allowed.

Updatable View

For view to be updatable, DBMS must be able to trace any row or column back to its row or column in the source table.

Advantages of Views

- Data independence
- Currency
- Improved security
- Reduced complexity
- Convenience
- Customization
- Data integrity

Disadvantages of Views

- Update restriction
- Structure restriction
- Performance

View Materialization

- View resolution mechanism may be slow, particularly if view is accessed frequently.
- View materialization stores view as temporary table when view is first queried.
- Thereafter, queries based on materialized view can be faster than recomputing view each time.
- Difficulty is maintaining the currency of view while base tables(s) are being updated.

Access Control - Authorization Identifiers and Ownership

- Authorization identifier is normal SQL identifier used to establish identity of a user. Usually has an associated password.
- Used to determine which objects user may reference and what operations may be performed on those objects.
- Each object created in SQL has an owner, as defined in AUTHORIZATION clause of schema to which object belongs.
- Owner is only person who may know about it.

Privileges

- Actions user permitted to carry out on given base table or view:

SELECT Retrieve data from a table.

INSERT Insert new rows into a table.

UPDATE Modify rows of data in a table.

DELETE Delete rows of data from a table.

REFERENCES Reference columns of named table in integrity constraints.

USAGE Use domains, collations, character sets, and translations.

Privileges

- Can restrict INSERT/UPDATE/REFERENCES to named columns.
- Owner of table must grant other users the necessary privileges using GRANT statement.
- To create view, user must have SELECT privilege on all tables that make up view and REFERENCES privilege on the named columns.

GRANT

```
GRANT       {PrivilegeList | ALL PRIVILEGES}
ON          ObjectName
TO          {AuthorizationIdList | PUBLIC}
[WITH GRANT OPTION]
```

- *PrivilegeList* consists of one or more of above privileges separated by commas.
- ALL PRIVILEGES grants all privileges to a user.

GRANT

- PUBLIC allows access to be granted to all present and future authorized users.
- *ObjectName* can be a base table, view, domain, character set, collation or translation.
- WITH GRANT OPTION allows privileges to be passed on.

Example 7.7/8 - GRANT

Give Manager full privileges to Staff table.

```
GRANT ALL PRIVILEGES
ON Staff
TO Manager WITH GRANT OPTION;
```

Give users Personnel and Director SELECT and UPDATE on column salary of Staff.

```
GRANT SELECT, UPDATE (salary)
ON Staff
TO Personnel, Director;
```

Example 7.9 - GRANT Specific Privileges to PUBLIC

Give all users SELECT on Branch table.

```
GRANT SELECT
ON Branch
TO PUBLIC;
```

REVOKE

- REVOKE takes away privileges granted with GRANT.

```
REVOKE [GRANT OPTION FOR]
                {PrivilegeList | ALL PRIVILEGES}
ON ObjectName
FROM {AuthorizationIdList | PUBLIC}
                [RESTRICT | CASCADE]
```

- ALL PRIVILEGES refers to all privileges granted to a user by user revoking privileges.

Example 7.10/11 - REVOKE Specific Privileges

Revoke privilege SELECT on Branch table from all users.

```
REVOKE SELECT
ON Branch
FROM PUBLIC;
```

Revoke all privileges given to Director on Staff table.

```
REVOKE ALL PRIVILEGES
ON Staff
FROM Director;
```