SQL – Data Definition

Chapter Seven

ISO SQL Data Type

DATA TYPE	DECLARATIONS				
boolean	BOOLEAN				
character	CHAR	VARCHAR			
bit†	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		

†BIT and BIT VARYING have been removed from the SQL:2003 standard.

Integrity Enhancement Feature

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

Integrity Enhancement Feature

Required Data

position VARCHAR(10) NOT NULL

Domain Constraints

(a) CHECK

(b) CREATE DOMAIN

CREATE DOMAIN DomainName [AS] dataType

[DEFAULT defaultOption]

[CHECK (searchCondition)]

Domain Constraints Example

- sex CHAR NOT NULL
  CHECK (sex IN ('M', 'F')

- 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

- Any INSERT/UPDATE attempting to create FK value in child table without matching CK value in parent is rejected.
- Action taken attempting to 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. 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.
- 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)
CREATE ASSERTION StaffNotHandlingTooMuch
CHECK (NOT EXISTS    (SELECT staffNo
                      FROM PropertyForRent
                      GROUP BY staffNo
                      HAVING COUNT(*) > 100))
```

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

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.

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.

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).

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*.

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.

DROP VIEW

```
DROP VIEW ViewName [RESTRICT | CASCADE]
```

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

```
DROP VIEW Manager3Staff;
```

- 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.

VIEW Resolution

Count number of properties managed by each member at branch B003.

```
SELECT staffNo, cnt
FROM StaffPropCnt
WHERE branchNo = 'B003'
ORDER BY staffNo;
```

- View column names in SELECT list are translated into their corresponding column names in the defining query:

```
SELECT s.staffNo As staffNo, COUNT(*) As cnt
```

- View names in FROM are replaced with corresponding FROM lists of defining query:

```
FROM Staff s, PropertyForRent p
```

View Resolution

- WHERE from user query is combined with WHERE of defining query using AND:

  WHERE s.staffNo = p.staffNo AND branchNo = 'B003'

- GROUP BY and HAVING clauses copied from defining query:

  GROUP BY s.branchNo, s.staffNo

- ORDER BY copied from query with view column name translated into defining query column name

  ORDER BY s.staffNo

View Resolution

- Final merged query is now executed to produce the result:

```
SELECT s.staffNo AS staffNo, COUNT(*) AS cnt
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo AND
                  branchNo = 'B003'
GROUP BY s.branchNo, s.staffNo
ORDER BY s.staffNo;
```

View Updateability

- 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.

Transactions

- SQL defines transaction model based on COMMIT and ROLLBACK.
- Transaction is logical unit of work with one or more SQL statements guaranteed to be atomic with respect to recovery.
- An SQL transaction automatically begins with a *transaction-initiating* SQL statement (e.g., SELECT, INSERT).
- Changes made by transaction are not visible to other concurrently executing transactions until transaction completes.
- Transaction can complete in one of four ways:
  - COMMIT ends transaction successfully, making changes permanent.
  - ROLLBACK aborts transaction, backing out any changes made by transaction.
  - For programmatic SQL, successful program termination ends final transaction successfully, even if COMMIT has not been executed.
  - For programmatic SQL, abnormal program end aborts transaction.

Transactions

- New transaction starts with next transaction-initiating statement.
- SQL transactions cannot be nested.
- SET TRANSACTION configures transaction:

```
SET TRANSACTION
   [READ ONLY  |  READ WRITE]  |
   [ISOLATION LEVEL READ UNCOMMITTED  |
   READ COMMITTED | REPEATABLE READ  | SERIALIZABLE ]
```

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.

- 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.
- 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.