3ISY402 – DATABASE SYSTEMS

Lecture 6 - SQL: Data Definition

Material from essential text:

T CONNOLLY & C BEGG. Database Systems – A Practical Approach to Design, Implementation and Management, 4th Edition. Addison-Wesley, 2005.

Leena Gulabivala

Lecture - Objectives

- Introduction to some basic SQL DDL statements.
- How to define table structures using CREATE TABLE statements.
- How to define Primary and Foreign Keys.
- How to amend tables using ALTER statements.
- How to delete tables using DROP statements.
- How to rename tables using RENAME statements.
- How to create and delete views using CREATE VIEW and DROP VIEW .
- Under what conditions views are updatable.
- Advantages and disadvantages of views.

Data Definition

- SQL DDL allows database objects such as schemas, domains, tables, views, and indexes to be created and destroyed.
- Relations and other database objects exist in an *environment*.
- Each environment contains one or more *catalogs*, and each catalog consists of a set of schemas.
- A *schema* is a named collection of related database objects.
- Objects in a schema can include tables, views, domains, etc.

CREATE TABLE Syntax

```
CREATE TABLE TableName (
{colName dataType [NOT NULL]  [,...]}
[PRIMARY KEY (listOfColumns),]
{[FOREIGN KEY (listOfFKColumns)
    REFERENCES ParentTableName
    [(listOfCKColumns)]  [,…]}  );
```

CREATE TABLE

- Creates a table with one or more columns of the specified *dataType*.
- NULL (default) indicates whether column can contain *nulls*.
- With NOT NULL, system rejects any attempt to insert a null in the column.
- *Primary keys should always be specified as NOT NULL.*
- Foreign keys are often (but not always) candidates for NOT NULL.
- FOREIGN KEY clause specifies FK.

Some Common SQL Data Types

CHAR(*size*)	Fixed length character data of length *size* bytes.
VARCHAR2(*size*)	Variable length character string having maximum length *size* bytes.
NUMBER (*l, d*)	Fixed-point or floating-point numbers, where ***l*** stands for length and ***d*** stands for the number of decimal digits. *Eg – NUMBER(5,2) allows max 999.99.* *Eg – NUMBER(3) allows max 999.*
DATE	Stores dates ranging from 01 January 4712 BC to 31 December 4712 AD

Example 1 - CREATE TABLE Branch

Create the structures for the Branch table.

```
CREATE TABLE Branch (
      branchNo     CHAR(5)              NOT NULL,
      street       VARCHAR2(25),
      city         VARCHAR2(15),
      postcode     VARCHAR2(10) );
```

Example 2 - CREATE TABLE Staff

Create the structures for the Staff table.

```
CREATE TABLE Staff (
        staffNo     CHAR(5)          NOT NULL,
        fName       VARCHAR2(20)     NOT NULL,
        lName       VARCHAR2(25)     NOT NULL,
        position    VARCHAR2(10)     NOT NULL,
        sex         CHAR(1)          NOT NULL,
        DOB         DATE,
        salary      NUMBER(8,2)      NOT NULL,
        branchNo    CHAR(5)          NOT NULL );
```

Defining Primary and Foreign Keys

- Primary Key
  - A column or a set of columns that is selected to uniquely identify a row within a table.
  - The primary key of a table must contain a unique, non-null value for each row. (Use NOT NULL)
  - A primary key comprising more than one column is called a *composite key* or *compound key*.
- Foreign Key
  - A column or a set of columns within one table that matches the candidate key of another table.

Referential Integrity

- *Referential integrity means that, if the foreign key contains a value, that value must refer to an existing row in the parent table.*
- Any INSERT/UPDATE attempting to create a foreign key value in the child table without a matching key value in the parent is rejected with the following error message.
  - **SQL Error: ORA-02291: integrity constraint violated - parent key not found**

Example 3 –CREATE TABLE with a Simple Primary Key

- Create the structures for the Staff table and define staff number (staffNo) as its *primary key*.

```
CREATE TABLE Staff (
        staffNo     CHAR(5)          NOT NULL,
        fName       VARCHAR2(20)     NOT NULL,
        lName       VARCHAR2(25)     NOT NULL,
        position    VARCHAR2(10)     NOT NULL,
        sex         CHAR(1)          NOT NULL,
        DOB         DATE,
        salary      NUMBER(8,2)      NOT NULL,
        branchNo    CHAR(5)          NOT NULL,
        PRIMARY KEY (staffNo)  );
```

Example 4 – CREATE TABLE with a Compound Key

- Create the structures for the Viewing table and define property number (propertyNo), client number (clientNo) and the viewing date as its *primary key (compound key).*

- CREATE TABLE Viewing (
  clientNo CHAR(4) NOT NULL,
  propertyNo CHAR(4) NOT NULL,
  viewDate DATE NOT NULL,
  comment VARCHAR2(30),
  PRIMARY KEY (clientNo, propertyNo, viewDate));

Example 5 – CREATE TABLE with Foreign Keys

- Create the structures for the Viewing table and define property number, client number and viewdate as its *primary key (compound key)* and property number and client number as *foreign keys.*

```
CREATE TABLE Viewing (
        clientNo      CHAR(4)     NOT NULL,
        propertyNo CHAR(4)     NOT NULL,
        viewDate     DATE         NOT NULL,
        comment     VARCHAR2(30),
        PRIMARY KEY (clientNo, propertyNo, viewDate),
        FOREIGN KEY (clientNo) REFERENCES
              Client(clientNo),
        FOREIGN KEY (propertyNo) REFERENCES
              PropertyForRent(propertyNo) );
```

ALTER TABLE

- Allows modification of table structures already defined.
- Amend an existing column data definition in a table.
- Adding new column to existing tables.
- Drop a column from a table.
- Rename a column in a table.
- Add a Primary or Foreign Key.

Example 6 - ALTER TABLE – Change Existing Column Data Definition

```
ALTER TABLE TableName
MODIFY (colName newDataType);
```

- Increase the size of column *position* in the Staff table from 10 to 20 to accommodate the new post of Managing Director.

```
ALTER TABLE   Staff
MODIFY        position      VARCHAR2(20);
```

Example 7 - ALTER TABLE – Adding Columns to Existing Tables

```
ALTER TABLE TableName
ADD (colName dataType [NOT NULL] );
```

- Add a column called *nationality* to the Staff table.

```
ALTER TABLE   Staff
ADD           (nationality   VARCHAR2(20));
```

ALTER TABLE – Deleting a Column or Renaming a Column

- Delete a Column:

```
ALTER TABLE TableName
DROP COLUMN colName;
```

- Rename a Column:

```
ALTER TABLE    TableName
RENAME COLUMN oldColumnName
              TO newColumnName;
```

- Example - rename the column from *sex* to *gender*:

```
ALTER TABLE    Staff
RENAME COLUMN sex TO gender;
```

Example 8 - ALTER TABLE – Add a Primary Key or Foreign Key Constraint

- Define *branchNo* as the primary key of the existing *Branch* table.
- ALTER TABLE Branch
  ADD PRIMARY KEY(branchNo);

- Define *branchNo* as the foreign key of the existing *Staff* table.
- ALTER TABLE Staff
  ADD FOREIGN KEY (branchNo)
  REFERENCES Branch(branchNo);

DROP TABLE

DROP TABLE TableName [RESTRICT | CASCADE] [CONSTRAINTS];

e.g. DROP TABLE PropertyForRent;

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

RENAME TABLE

RENAME OldTableName
TO NewTableName;

e.g. RENAME Property TO PropertyForRent;

- Renames named table to new named table.
- DBMS automatically transfers to the new table the integrity constraints, indexes and privileges that referenced the old table.
- *BUT* objects that referenced the old table (eg views, stored procedures and functions) become invalid.

Views

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

SQL - CREATE VIEW

CREATE VIEW ViewName [(newColumnName [,...])]

AS subselect

- Can assign a name to each column in view.
- If a list of column names is specified, it must have the same number of items as the number of columns produced by the *subselect*.
- If omitted, each column takes the name of the corresponding column in the *subselect*.
- List must be specified if there is any ambiguity in a column name.
- The *subselect* is known as the *defining query.*

Example 9 - 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     ManagerStaff
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 10 - Create Vertical View

- Create view of staff details at branch B003 excluding salaries.

```
CREATE VIEW Staff
AS SELECT  staffNo, fName, lName, position, sex
     FROM     Staff
     WHERE  branchNo = 'B003';
```

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

Example 11 - 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;
```

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 ManagerStaff;

- 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.
- Grouped view may never be joined with a base table or a view.
  - Eg StaffPropCnt view is a grouped view, so cannot join this view with another table or view.
- If a column in a view is based on an aggregate function:
  - Column may appear only in SELECT and ORDER BY clauses of queries accessing view.
  - Column may not be used in WHERE nor be an argument to an aggregate function in any query based on view.

Example 12 - 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;
```

View Updatability

- All updates to the base table are reflected in all views that encompass the base table.
- Similarly, may expect that if a view is updated then base table(s) will reflect change.
- *For a view to be updatable, DBMS must be able to trace any row or column back to its row or column in the source or base table.*

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.

Advantages and Disadvantages of Views

- Advantages of Views
  - Data independence
  - Currency
  - Improved security
  - Reduced complexity
  - Convenience
  - Customization
  - Data integrity
- Disadvantages of Views
  - Update restriction
  - Structure restriction
  - Performance

References and Further Reading:

- T CONNOLLY & C BEGG. *Database Systems – A Practical Approach to Design, Implementation and Management,* 4th Edition. Addison-Wesley, 2005.
  - Chapter 6 – SQL Data Definition.
- T CONNOLLY & C BEGG. *Database Solutions – A step-by-step guide to building databases,* 2nd Edition. Addison-Wesley, 2004.
  - Chapter on SQL Data Definition.
- J MORRISON, M MORRISON & R CONRAD. *Guide to ORACLE 10g*. Thomson Learning, 2006.
  - Chapter 2 – Creating and Modifying Database Tables.