

SQL DDL

Part I

September 13, 14, 2021

SQL as DDL

- To define
 - relational databases, schemas
 - data types, constraints
 - other constructs (such as domains, triggers, assertions)
 - a virtual table (or view) of data
- To remove an object that's no longer needed
- To build an index to make table access faster
- To change the definition of an existing object
- To control the physical storage of data by the DBMS

SQL as DDL

- The core of the DDL is based on 3 SQL verbs:
 - **CREATE**: defines and creates a DB object
 - **DROP**: removes an existing DB object
 - **ALTER**: changes the definition of a DB object

Creating a Database

- **CREATE DATABASE <databasename>**
- You accept the defaults associated with the many clauses that are part of the creation of the DB
- **CREATE DATABASE <databasename>**
DEFAULT CHARACTER SET <characterSetName>
DEFAULT COLLATE <collatingsequencename>
- This creates a new database without tables

Connect to a Database

- If you want to use this database, do not forget to make it the current database

- **CONNECT TO** <dbname> in DB2
 - **USE** <dbname> in MySQL

- Privileges and authorities are associated with the creation of the DB

Updating a Database

- You can change the existing default character set and collating sequence with the command:
- **ALTER DATABASE <databasename>
DEFAULT CHARACTER SET <charsetname>
DEFAULT COLLATE <CollatingSeqname>**
- New defaults apply only to the tables and columns that will be created after the update

Dropping a Database

- Most drastic SQL statement

```
DROP DATABASE <dbname>
```

- This statement removes the entire database at once
- All tables of that database disappear permanently, so be careful !

Catalog Tables

- Most products have a number of tables for their own use in which this data is stored: **catalog tables** or **system tables**
- The **system catalog** is a collection of special tables in a database that are owned, created and maintained by the DBMS itself
- These system tables contain data that describes the structure of the database
- The tables in the system catalog are automatically created when the database is created
- The DBMS constantly refers to the data in the system catalog while processing SQL statements

Catalog Tables

- Like “ordinary” tables that can be queried
- User access to system catalog is read-only
- The DBA may restrict system catalog access to provide an additional measure of database security
- DBMS itself takes care of inserting, deleting, and updating rows of the system tables as it modifies the structure of a database
- Almost all commercial SQL products include system tables that describe dbs, tables, columns, users, views, privileges

Table Creation

- The CREATE TABLE statement defines a new table, and DBMS prepares it to accept data added with the INSERT statement
- Column Definition :
 - Column definition appear in a comma separated list enclosed in parentheses
 - The order of the column definition determines the left-to-right order of the columns in the table
 - Each column definition specifies:
 - Column Name, Data Type, Required Data, Default Value

Table Creation

- **Column name**: every column in a table must have unique name, but the names may duplicate those of columns in other tables
- **Data type**: specifies the domain values for the attribute
- **Required data**: determines whether the column contains required data, and prevents NULL values from appearing in the column; otherwise, NULL values are allowed
- **Default value**: uses an optional default value for the column when an INSERT statement for the table does not specify a value for the column

Table Creation

- **CREATE TABLE <table-name>**
 (attribute_name datatype [(max_length)],
 attribute_name datatype attrib_constraints,
 ...
 attribute_name datatype table_constraints);
- In addition SQL2 standards allows you to specify a domain instead of a data type within a column defn

Primitive Data Types

- **Numeric**

- INTEGER, INT, SMALLINT - subsets of the integers (m/c dependent)
INTEGER ($-2,147,483,647 \leq n \leq 2,147,483,647$)
SMALLINT ($-32,767 \leq n \leq 32,767$)
BIGINT (-2^{63} up to and including $+2^{63}-1$)
- REAL, DOUBLE PRECISION are floating-point and double-precision floating-point (machine dependent)
- FLOAT(N) is floating-point with at least N digits
- DECIMAL(P,D) (or DEC(P,D), or NUMERIC(P,D)), with P digits of which D are to the right of the decimal point

- **Boolean**

- **Bit-strings**

- BIT(N) is a fixed-length bit string
- VARBIT(N) or BIT VARYING(N): bit string with at most N bits

Primitive Data Types (cont.)

- **Character-string**

- CHAR(N) (or CHARACTER(N)) is a fixed-length character string
- VARCHAR(N) (or CHAR VARYING(N), or CHARACTER VARYING(N)) is a variable-length character string with at most N characters

- **Time**

- DATE is a date: YYYY-MM-DD (2005-03-30)
- TIME, a time of day: HH-MM-SS (12:02:20)
- TIME(I), a time of day with I decimal fractions of a second: HH-MM-SS-F....F
- TIME-STAMP, date, time, fractions of a second and an optional WITH TIME ZONE qualifier:

YYYY-MM-DD-HH-MM-SS-F...F{-HH-MM}
(TIMESTAMP `1999-12-31 23:59:59.000000')

Table Creation

- An attribute may have constraints, such as **NOT NULL**, key, entity integrity and referential integrity
- **Missing and Default values**

```
CREATE TABLE WORKS_ON  
(ESSN      INT(9)      NOT NULL,  
HOURS  DECIMAL(3,1)  NOT NULL DEFAULT 00.0,  
REGION VARCHAR(10)  NOT NULL DEFAULT 'Eastern' );
```

- **Check Constraint:** restricts data in the table so that its rows meet a specified search condition

```
YEAR INT NOT NULL, DEFAULT 2003  
CHECK (YEAR BETWEEN 1960 AND 2003)
```

Table Creation

■ Specifying Key Constraints

- Primary keys:

PRIMARY KEY (<attribute(s)>)

- Secondary keys:

UNIQUE <attribute(s)>

- Foreign keys:

FOREIGN KEY <attribute> REFERENCES <table(attribute)>

■ For secondary keys:

- **UNIQUE (attribute) or**
Attribute Datatype Requireddata UNIQUE

Table Creation

```
CREATE TABLE WORKS_ON  
( ESSN          INT(9)          NOT NULL,  
  PNO           INT            NOT NULL,  
  HOURS        DECIMAL(3,1)    NOT NULL,  
  PRIMARY KEY (ESSN, PNO),  
  FOREIGN KEY (ESSN) REFERENCES  
    EMPLOYEE(SSN),  
  FOREIGN KEY PNO REFERENCES  
    PROJECT(PNUMBER) );
```

Populating Table

```
INSERT INTO WORKS_ON  
(ESSN, PNO, Hours) VALUES (115, 5,14)
```

Table Creation: Entity Integrity

- The requirement that primary keys have unique values is called the entity integrity constraint
- When a primary key is specified for a table, the DBMS automatically checks the uniqueness of the primary key value for every INSERT and UPDATE statement performed on the table

Uniqueness and NULL value

- Because of the NULL value, the DBMS cannot conclusively decide whether the PK duplicates one that is already in the table
- The answer must be “may be”, depending on the “real” value of the missing (NULL) data
- SQL standard requires that every column that is part of a primary key must be declared NOT NULL
- The same restriction applies for every column that is named in a uniqueness constraint

Table Creation: Referential Integrity

■ Referential Triggered Action

- When referential integrity is violated by inserting, deleting, updating tuples
- Referential integrity problems: **four update situations**
 - Inserting a new child row
 - Updating the foreign key in a child row
 - Deleting a parent row
 - Updating the primary key in a parent row

Table: Delete & Update Rules

- **RESTRICT delete/update rule**

**FOREIGN KEY EMPDNO REFERENCES DEPTT(DNUMBER)
ON DELETE RESTRICT**

- You can't delete/update a row from the parent table if the row has any children
 - » You can't delete/change a department if any sales person are assigned to it

Table: Delete & Update Rules

- CASCADE delete/update rule

**FOREIGN KEY EMPDNO REFERENCES DEPTT(DNUMBER)
ON DELETE CASCADE**

- Deleting/updating a parent row also deletes /updates all of its child row automatically from the child table
 - » Deleting/Changing a department no. automatically deletes/changes the department no. for all the sales person assigned to that department

Table :Delete & Update Rules

- SET NULL delete/update rule

**FOREIGN KEY EMPDNO REFERENCES DEPTT(DNUMBER)
ON DELETE SET NULL**

- When a parent row is deleted/updated, the foreign key values in all of its child rows should automatically be set to null
 - » If a department number is changed, the current department assignment of its sales persons are unknown

Table :Delete & Update Rules

- SET DEFAULT delete/update rule

**FOREIGN KEY EMPDNO REFERENCES DEPTT(DNUMBER)
ON DELETE SET DEFAULT '101'**

- When a parent row is deleted/updated, the foreign key values in all of its child rows should automatically be set to the default value for that particular column
 - » If a department no. is deleted/updated, automatically change the department assignment of its sales persons to the default department specified in the definition

Referential Cycles

- **SALESPERSON table:** ENUM, NAME, AGE, **EOFFICE**, TITLE
- **OFFICE table:** OFFICE, CITY, REGION, **MGR**, TARGET, SALE
- These two relationship form a referential cycle
- Regardless of the no. of tables that they involve, referential cycles pose special problems for referential integrity constraints
- Suppose NULL values are not allowed in the primary or foreign keys of the two tables

Referential Cycles

- Company hires a new salesperson, Ravi (enum 115), who is the manager of a new sales office in Bombay (office no. 14)
- **INSERT INTO SALESPERSON (ENUM, NAME, EOFFICE)
VALUES (115,'RAVI',14)**
- **INSERT INTO OFFICE (OFFICE, CITY, REGION, MGR, TARGET,
SALES) VALUES (14,'BOMBAY','WESTERN',115,0.00,0.00)**
- Insertion deadlock
- To prevent this at least one of the foreign keys in a referential cycle must permit NULL values

Referential Cycles

- Say MGR column does not permit NULL but EOFFICE does
- These insertions can be accomplished with two INSERTS and an UPDATE
- INSERT INTO SALESPERSON (ENUM, NAME, EOFFICE) VALUES (115,'RAVI', NULL)
- INSERT INTO OFFICE (OFFICE, CITY, REGION, MGR, TARGET, SALES) VALUES (14,'BOMBAY','WESTERN',115,0.00,0.00)
- UPDATE SALESPERSON SET EOFFICE = 14 WHERE ENUM = 115

FOREIGN Keys and NULL Values

- One should be very careful with NULL values in compound foreign key
- CREATE TABLE statement provides two options
 - MATCH FULL option
 - MATCH PARTIAL option

Delete Tables

- With proper permission, you can also drop a table owned by another user by specifying a qualified table name
- **DROP TABLE <tablename>**
- **DROP TABLE <tablename> CASCADE**
 - removes a table and the related constraints (or views)
- **DROP TABLE <name> RESTRICT**
 - removes a table only if it is not referenced
 - This provides an added precaution against unintentional side-effects of a drop view statements
- By default RESTRICT

Changing a Table Definition

■ Alter Tables

- To add or delete attributes and constraints
- To change a column definition

- **ALTER TABLE PROJECT ADD PMGR CHAR(9);**
- **ALTER TABLE PROJECT ALTER PLOCATION SET DEFAULT 'NEW YORK';**
- **ALTER TABLE PROJECT DROP DNUM CASCADE (or RESTRICT);**
- **ALTER TABLE PROJECT ADD CONSTRAINT IN DEPTMGR FOREIGN KEY (DEPTMGR) REFERENCES DEPTT (MGR);**
- **ALTER TABLE PROJECT DROP PRIMARY KEY (DEPTMGR);**
Foreign key corresponding to this primary key must be dropped

Constraint Definitions

- DB constraints (uniqueness, primary and foreign key, and check) are closely associated with a single DB table
- Specified as part of the create table statement and can be modified or dropped using the ALTER TABLE statement
- The other two types of DB integrity constraints:
 - Assertions
 - Domains
- These constraints are created as stand-alone objects within a DB, independent of any individual table definition