Structured Query Language (SQL) DDL

SWEN 304
Trimester 2, 2017

Lecturer: Dr Hui Ma

Engineering and Computer Science

Slides by: Pavle Mogin & Hui Ma

Outline

- Data Definition Language
  - Database schema definition
  - Domain definition
  - Table definition
  - Constraints
  - Schema Evolution
- Reading: Chapter 8 of the textbook

Structured Query Language (SQL)

- We are concerned with three major problems, namely:
  - How to define a database schema and its relation schemas?
  - How to store and manipulate data in relations?
  - How to retrieve data from a database?
- SQL is a standardised language used in almost all relational database systems
  - developed at IBM and later defined as a standard by ANSI and ISO
  - SQL stands for Structured Query Language (some also say SEQUEL for Structured English QUEry Language)
  - versions in use: SQL-86, SQL-92, SQL:1999 (also known as SQL-3, the ANSI and ISO standard), SQL:2003, SQL:2006, SQL:2008, SQL:2011

Structured Query Language (SQL)

- We will discuss three different aspects of SQL, namely the usage of SQL as
  - a Data Definition Language (DDL)
  - a Data Manipulation Language (DML)
  - a Query Language (QL)
- SQL provides some further features like the definition of external views (View Definition Language (VDL) on the database, authorisation and access rights, and coupling with programming
- Declarative and set oriented

SQL as Data Definition Language

- Catalog
- Schemas (Relational database schemas)
- Domains
- Tables (Relation schemas)
- Constraints
- Modify schemas (schema evolution)

Databases and Catalogs in DBMS

- A DBMS creates a catalog for every database that it manages
  - The catalog records essential information about this database
    - This information is also called meta data of the database
  - The catalog is used by DBMS modules like the DDL compiler module, the query optimisation module, or the authorisation and security module, etc.
- The catalog includes in particular the following:
  - Schema table: information on all relation schemas
  - Attribute table: information on all attributes
  - Data types table: information on all data types (domains)
  - Constraints table: information on all constraints

Databases and Catalogs in DBMS

- Other information recorded in the catalog:
  - Descriptions of views (external schemas) and their queries
  - Descriptions of storage structures and indexes to enable fast access to the data in the database
  - Security and authorization information that records the owner of each relation and describes each user's privileges to access specific relations or views of the database
- Catalogs and SQL:
  - In a relational DBMS, the catalog is stored in form of tables
    - This allows the DBMS software (and authorised users such as DBAs) to access and manipulate the catalog using SQL
  - SQL'92 defines a standard format for the catalog, called the information schema
  - The precise layout of the catalog differs from DBMS to DBMS

Defining a Schema

- A RDB Schema describes that part of a database used by one “application” and its users

```
CREATE SCHEMA UNIVERSITY AUTHORIZATION huima;
```

- Followed by definitions of the parts of the database schema:
  - domains,
  - tables,
  - constraints,
  - views,
  - authorization grants
- Use CREATE SCHEMA only in the case you need more than one schema in the same database

Notational Conventions

We shall use BNF notation to describe SQL syntax:

- Non terminal symbols are shown in angled brackets 〈data_type〉 ,
- Optional parts shown in square brackets [DEFAULT 〈value〉],
- Alternatives shown in parentheses separated by bars (FULL | PARTIAL)
- Lists or repeated elements shown in braces { 〈constraint_declaration〉 ,… }
- Note: In SQL, names are not case-sensitive

Domain Definition Syntax

- CREATE DOMAIN 〈domain_name〉
    [AS] 〈data_type〉
    [DEFAULT 〈value〉]
    [{CONSTRAINT 〈name〉 〈constraint〉 , ...}]
- 〈data_type〉 ::
  - Numeric: INT , REAL, DECIMAL(d, p)
  - Character-string: CHAR(n), VARCHAR(n)
  - Bit-string: BIT(n), BIT VARYING(n)
  - Date: DATE(format)
  - Time: TIME(format)

Domain Example

```
CREATE DOMAIN idno
    AS  INT
    DEFAULT 300001
    NOT NULL
    CONSTRAINT idnoconstr
        CHECK (VALUE > 300000 AND VALUE
                        <= 399999);
```

Base Table (Relation Schema) Definition

```
CREATE TABLE 〈table_name〉 (
      { 〈attribute_declaration〉 , …}
      [, { [ CONSTRAINT 〈name〉 ] 〈table_constraint〉 ,… } ]
);
```

```
〈attribute_declaration〉 ::
   〈attribute_name〉
      ( 〈data_type〉 [(max_length)] | 〈domain_name〉 )
      [DEFAULT ( 〈value〉 | 〈function〉 | NULL) ]
      [ { [CONSTRAINT 〈name〉 ] 〈att_constraint〉 , …} ]
```

Attribute Constraint Declarations

```
⟨att_constraint⟩  ::
  NOT NULL
  |  (PRIMARY KEY | UNIQUE)
  |  REFERENCES  ⟨referenced_table_name⟩
     [ ⟨referenced_table_attribute⟩ ]
         [ON DELETE (NO ACTION | CASCADE | SET NULL
                              | SET DEFAULT )]
         [ON UPDATE (NO ACTION | CASCADE | SET NULL
                                       | SET DEFAULT )]
         [MATCH FULL | MATCH PARTIAL]
  | CHECK ( ⟨conditional_expression⟩ )
```

Examples:

http://www.postgresql.org/docs/8.0/interactive/ddl-constraints.html

Table Constraints Declarations

```
⟨table_constraint⟩   ::=
  PRIMARY KEY  ( ⟨attribute_list⟩ )
  | UNIQUE ( ⟨attribute_list⟩ )
  | FOREIGN KEY  ⟨attribute_list⟩
        REFERENCES  ⟨referenced_table_name⟩
        [ ⟨referenced_table_attribute_list⟩ ]
        [MATCH (FULL | PARTIAL)]
        [ON DELETE (NO ACTION | CASCADE | SET NULL |
                   SET DEFAULT )]
        [ON UPDATE (NO ACTION | CASCADE | SET NULL |
                   SET DEFAULT )]
  CHECK  ⟨conditional_expression⟩
```

UNIVERSITY Database

UNIVERSITY ={STUDENT(<u>StudId</u>, Lname, Fname, Major),
COURSE(<u>CourId</u>, Cname, Points, Dept),
GRADES(<u>StudId</u>, <u>CourId</u>, Grade)}

STUDENT

StudId	Lname	Fname	Major
300111	Smith	Susan	COMP
300121	Bond	James	MATH
300143	Bond	Jenny	MATH
300132	Smith	Susan	COMP

COURSE

CourId	Cname	Points	Dept
COMP302	DB sys	15	Engineering
COMP301	softEng	20	Engineering
COMP201	Pr & Sys	22	Engineering
MATH214	DisMat	15	Mathematics

GRADES

StudId	CourId	Grade
300111	COMP302	A+
300111	COMP301	A
300111	MATH214	A
300121	COMP301	B
300132	COMP301	C
300121	COMP302	B+
300143	COMP201	ω
300132	COMP201	ω
300132	COMP302	C+

University Database Schema: COURSE

```
CREATE TABLE COURSE (
    CourId  CHAR(7)  CONSTRAINT cspk  PRIMARY KEY,
    CName  CHAR(15) NOT NULL,
    Points  INT NOT NULL CONSTRAINT pointschk
      CHECK (Points >= 0 AND Points <= 50),
    Dept  CHAR(25)
  );
```

University Database Schema: STUDENT

- Without using idno DOMAIN:

```
CREATE TABLE STUDENT (
    StudId  INT
            NOT NULL
            DEFAULT 30000
            CONSTRAINT stpk PRIMARY KEY
            CONSTRAINT StIdRange CHECK
                    (StudId  BETWEEN 300000 AND 399999),
    LName  CHAR(15) NOT NULL,
    FName  CHAR(15) NOT NULL,
    Major  CHAR(25) DEFAULT ‘Comp’
    );
```

University Database Schema: STUDENT

- Using idno DOMAIN:

```
CREATE TABLE STUDENT (
   StudId  idno CONSTRAINT stpk PRIMARY KEY,
   LName  CHAR(15) NOT NULL,
   FName  CHAR(15) NOT NULL,
   Major  CHAR(25)
);
```

A Question for You

- SQL allows defining only one relation schema key – PRIMARY KEY
- Suppose *A* is defined as a primary key
- Can we define *B* as another key (not primary one)?
- Answers: ?

```
CREATE TABLE STAFF (
        staff_id  INT PRIMARY KEY,
        ird_number  CHAR(7) NOT NULL UNIQUE,
        address VARCHAR(255)
);
```

Example UNIQUE Constraints

```
CREATE TABLE PERSONBS(
    Id INT PRIMARY KEY,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255),
    Address VARCHAR(255),
    DoB DATE NOT NULL,
    CONSTRAINT uc_Person UNIQUE (LastName, DoB)
);
```

Conditional Expressions: CHECK (Ref)

- Conditional expression of the CHECK clause can be any plausible combination of the following:

  $[A \;\; \theta \; a] \quad [A \;\; \theta \;\; B\,]$

  $[A$ [NOT] BETWEEN a_1 AND $a_2\,]$

  $[A$ [NOT] LIKE ⟨pattern⟩ $]$

  $[A$ [NOT] SIMILAR TO ⟨regular expression⟩ $]$

  $[A$ [NOT] IN ⟨value_list⟩ $]$

  $[A \;\; \theta$ ANY ⟨value_list⟩ $]$ $[A \;\; \theta$ SOME ⟨value_list⟩ $]$

  $[A \;\; \theta$ ALL ⟨value_list⟩ $]$

  Combined with AND or OR or NOT

- where $\theta \in \{ =, <, <=, >, >=, <> \}$,
  A and B attributes or functions of attributes,
  $a_i \in dom\,(A\,)$,

University Database Schema: GRADES

```
CREATE TABLE GRADES (
    StudId  INT  NOT NULL
          CONSTRAINT Gstidrange  CHECK
                    (StudId BETWEEN 300000 and 399999),
          CONSTRAINT gsri REFERENCES STUDENT
                    ON DELETE CASCADE,
    CourId  CHAR(8) NOT NULL
          CONSTRAINT gpri  REFERENCES COURSE
                    ON DELETE NO ACTION,
    Grade  CHAR(2)
          CONSTRAINT grd  CHECK
                    (Grade IN ('A+', 'A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', NULL)),

    CONSTRAINT gpk  PRIMARY KEY (StudId, CourId )
    );
```

CHECK: refer to several columns

- A CHECK constraint can also refer to several columns

```
CREATE TABLE BOOK (
    book_no INT PRIMARY KEY,
    title VARCHAR(30) NOT NULL,
    price NUMERIC CHECK (price > 0),
    discounted_price NUMERIC CHECK (discounted_price > 0),
   CHECK (price > discounted_price)
);
```

Modifying a Schema: DROP

- DROP ⟨construct⟩ ⟨construct_name⟩ ⟨drop_behavior⟩
- ⟨construct⟩ ::= (SCHEMA | TABLE | DOMAIN)
- ⟨drop behavior⟩ ::= (CASCADE | RESTRICT)
- Generally:
  - CASCADE behavior means deleting the construct itself and all the other constructs related to it,
  - RESTRICT behavior means that the construct will be deleted only if it is empty (schema), or not referenced by any other construct (like: table, attribute, view)
- e.g. DROP TABLE COURSE RESTRICT

  DROP TABLE STUDENTS CASCADE

Modifying a Schema: ALTER

- ALTER TABLE 〈table_name〉 …

  allows you to modify a table after it has been defined

e.g.

```
ALTER TABLE STUDENT ADD NoOfPoints INT DEFAULT 320;

ALTER TABLE GRADES ALTER Grade SET DEFAULT 'C';

ALTER TABLE GRADES DROP CONSTRAINT gsri ;

ALTER TABLE GRADES ADD CONSTRAINT gsfk FOREIGN KEY StudId
    REFERENCES STUDENT ON DELETE NO ACTION;

ALTER TABLE STUDENT DROP CONSTRAINT stpk CASCADE;
```

Summary

- Structured Query Language (DDL, DML, QL and VDL)
- Data Definition Language to create schema constructs (CREATE SCHEMA, CREATE DOMAIN, CREATE TABLE)
- CREATE TABLE command is used to define relation schema attributes and constraints
  - Attribute constraints
  - Table constraints
- Schema evolution commands allow altering attributes and constraints

Next Lecture

- SQL DML
  - Single table queries
  - Multiple table queries
  - Nested queries
  - Aggregate functions