CSC 443 – Data Base Management Systems

Lecture 6 – SQL As A Data Definition Language

Basic SQL

- SQL language
  - Considered one of the major reasons for the commercial success of relational databases
- **SQL**
  - **Structured Query Language**
  - Statements for data definitions, queries, and updates (both DDL and DML)
  - **Core specification – Core of the language found in all implementations**
  - Plus specialized **extensions** added in various implementations

SQL Data Definition and Data Types

- Terminology:
  - **Table**, **row**, and **column** used for relational model terms relation, tuple, and attribute
- `CREATE` statement
  - Main SQL command for data definition

Schema and Catalog Concepts in SQL

- **SQL schema**
  - Identified by a **schema name**
  - Includes an **authorization identifier** and **descriptors** for each element
- Schema **elements** include
  - Tables, constraints, views, domains, and other constructs
- Each statement in SQL ends with a semicolon

Schema and Catalog Concepts in SQL (continued)

- **`CREATE SCHEMA`** statement
  - **`CREATE SCHEMA COMPANY;`**
- **Catalog**
  - Named collection of schemas in an SQL environment
- SQL **environment**
  - Installation of an SQL-compliant RDBMS on a computer system

The CREATE TABLE Command in SQL

- Specify a new relation
  - Provide name
  - Specify attributes and initial constraints
- Can optionally specify schema:
  - **`CREATE TABLE COMPANY.EMPLOYEE ...`**

    or
  - **`CREATE TABLE EMPLOYEE ...`**

The CREATE TABLE Command in SQL (continued)

- **Base tables** (**base relations**)
  - Relation and its tuples are actually created and stored as a file by the DBMS
- **Virtual relations**
  - Created through the `CREATE VIEW` statement

Defining the **`COMPANY`** Schema Using SQL

```
mysql> create table employee
    -> (fname          varchar(15)       not null,
    -> Minit           char,
    -> Lname           varchar(15)       not null,
    -> ssn             char(9)           not null,
    -> Bdate           date,
    -> Address         varchar(30),
    -> Sex             char,
    -> Salary          decimal(10,2),
    -> Super_ssn       char(9),
    -> Dno             int               not null);
Query OK, 0 rows affected (0.19 sec)
```

Defining the **Department** Table

```
mysql> create table department
    -> (Dname         varchar(15)          not null,
    -> Dnumber        int                  not null,
    -> Mgr_ssn        char(9)              not null,
    -> Mgr_start_date   date,
    -> Primary key (Dnumber),
    -> Unique (Dname));
Query OK, 0 rows affected (0.14 sec)
```

Adding Primary and Foreign Keys

```
mysql> alter table employee add primary key (Ssn);
Query OK, 0 rows affected (0.25 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table employee add foreign key
(Super_ssn) references employee(ssn);
Query OK, 0 rows affected (0.20 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table employee add foreign key (Dno)
references department (Dnumber);
Query OK, 0 rows affected (0.20 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Adding a Foreign Key

```
mysql> alter table department add foreign key
(Mgr_ssn) references employee (Ssn );
Query OK, 0 rows affected (0.27 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Defining the **`Dept_Locations`** Table

```
mysql> create table Dept_Locations
    -> (Dnumber         int             not null,
    -> Dlocation        varchar(15)     not null,
    -> primary key (Dnumber, Dlocation),
    -> foreign key (Dnumber) references
department(Dnumber));
Query OK, 0 rows affected (0.20 sec)
```

Defining the **`Project`** Table

```
mysql> create table project
   -> (Pname       varchar(15)     not null,
   -> Pnumber      int             not null,
   -> Plocation    varchar(15),
   -> Dnum         int             not null,
   -> primary key (Pnumber),
   -> unique (Pname),
   -> foreign key (Dnum) references department
   -> (Dnumber));
Query OK, 0 rows affected (0.11 sec)
```

Defining the **`Works_On`** Table

```
mysql> create table works_on
    -> (Essn            char(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));
Query OK, 0 rows affected (0.13 sec)
```

Defining the **`Dependent`** Table

```
mysql> create table dependent
    -> (Essn            char(9)         not null,
    -> Dependent_name   varchar(15)     not null,
    -> Sex              char,
    -> Bdate            date,
    -> Relationship     varchar(8),
    -> primary key (Essn, Dependent_name));
Query OK, 0 rows affected (0.09 sec)
```

The **`CREATE TABLE`** Command in SQL (continued)

- Some foreign keys may cause errors
  - Specified either via:
    - Circular references
    - Or because they refer to a table that has not yet been created

Attribute Data Types and Domains in SQL

- Basic **data types**
  - **Numeric** data types
    - Integer numbers: `INTEGER`, `INT`, and `SMALLINT`
    - Floating-point (real) numbers: `FLOAT` or `REAL`, and `DOUBLE PRECISION`
  - **Character-string** data types
    - Fixed length: `CHAR(`*n*`)`, `CHARACTER(`*n*`)`
    - Varying length: `VARCHAR(`*n*`)`, `CHAR VARYING(`*n*`)`, `CHARACTER VARYING(`*n*`)`

Attribute Data Types and Domains in SQL (continued)

- **Boolean** data type
  - Values of `TRUE` or `FALSE` or `NULL`
- **DATE** data type
  - Ten positions
  - Components are `YEAR`, `MONTH`, and `DAY` in the form YYYY-MM-DD

Attribute Data Types and Domains in SQL (continued)

- Domain
  - Name used with the attribute specification
  - Makes it easier to change the data type for a domain that is used by numerous attributes
  - Improves schema readability
  - Example:
    - **`CREATE DOMAIN SSN_TYPE AS CHAR(9);`**

INSERT Instruction

```
mysql> insert into employee values ('John', 'X',
'Jones', '222334444','1960-03-15', '3010 Broadway,
New York, NY', 'M', 52000, '333445555', 5);
Query OK, 1 row affected (0.08 sec)

mysql>
```

- This may not work if there are foreign key constraints - this can be corrected by using the command

  `Set foreign_key_checks = 0;`

Load File

```
mysql> load data local infile 'Data.txt' into table employee;
Query OK, 1 row affected, 1 warning (0.08 sec)
Records: 1  Deleted: 0  Skipped: 0  Warnings: 1
```

- The file must be located in the home directory of mysql (which is in

  C:\Program Files\MySQL\MySQL Server 5.5\bin

  This may not work on Panther because of the permission settings.

The State for the COMPANY Relational Database

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

The State for the COMPANY Relational Database (continued)

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Specifying Constraints in SQL

- Basic constraints:
  - Key and referential integrity constraints
    - Primary keys must be unique
    - Foreign keys must correspond to a primary key value that exists in the other table.
  - Restrictions on attribute domains and NULLs
    - We expect attributes to be within a specific range
    - The primary key may not necessarily be the only attribute not allowed to be NULL.
  - Constraints on individual tuples within a relation

Specifying Attribute Constraints and Attribute Defaults

- **NOT NULL**
  - **NULL** is not permitted for a particular attribute
- Default value
  - **DEFAULT** <value>
- **CHECK** clause
  - **Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);**

Specifying Key and Referential Integrity Constraints

- **PRIMARY KEY** clause
  - Specifies one or more attributes that make up the primary key of a relation
  - **Dnumber INT PRIMARY KEY;**
- **UNIQUE** clause
  - Specifies alternate (secondary) keys
  - **Dname VARCHAR(15) UNIQUE;**

Specifying Key and Referential Integrity Constraints (continued)

- **`FOREIGN KEY`** clause
  - Default operation: reject update on violation
  - Attach ***referential triggered action*** clause
    - Options include `SET NULL`, `CASCADE`, and `SET DEFAULT`
    - Action taken by the DBMS for `SET NULL` or `SET DEFAULT` is the same for both `ON DELETE` and `ON UPDATE`
    - `CASCADE` option suitable for "relationship" relations

Specifying Key and Referential Integrity Constraints (continued)

- **`ON DELETE SET NULL`**
  - If the tuple containing the primary key is deleted, then the corresponding value in another table where that attribute is a foreign key is set to null.
  - *E.g., if the supervisor's record is deleted, his subordinates' Super_ssn value is set to null.*
- **`ON UPDATE CASCADE`**
  - If the tuple has its primary key changed, that change cascades into the records where it is a foreign key.
  - E. g., if the supervisor's social security number is changed, this change cascades into his subordinates' records.
- If **`SET DEFAULT`** were used for either case, the foreign key's value would be reset to the default value.

Giving Names to Constraints

- Keyword **CONSTRAINT**
  - Name a constraint
  - It is used to identify a particular constraint if the constraint must be dropped later and replaced with another constraint .

Specifying Constraints on Tuples Using **CHECK**

- **CHECK** clauses at the end of a **CREATE TABLE** statement
  - Apply to each tuple individually
  - **CHECK (Dept_create_date <= Mgr_start_date);**

Basic Retrieval Queries in SQL

- **`SELECT`** statement
  - One basic statement for retrieving information from a database
- SQL allows a table to have two or more tuples that are identical in all their attribute values
  - Unlike relational model
  - Multiset or bag behavior

The SELECT-FROM-WHERE Structure of Basic SQL Queries

- Basic form of the `SELECT` statement:

```
SELECT    <attribute list>
FROM      <table list>
WHERE     <condition>;
```

where

- <attribute list> is a list of attribute names whose values are to be retrieved by the query.
- <table list> is a list of the relation names required to process the query.
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.

The SELECT-FROM-WHERE Structure of Basic SQL Queries (continued)

- Logical comparison operators
  - =, <, <=, >, >=, and <>
- **Projection attributes**
  - Attributes whose values are to be retrieved
- **Selection condition**
  - Boolean condition that must be true for any retrieved tuple

Simples Queries - Example

- Retrieve the birthdate and Address for John B.Smith

```
SELECT Bdate, Address
FROM Employee
WHERE Fname = 'John' AND Minit = 'B'
          AND Lname = 'Smith;
```

- Retrieve the name and address for all employees working in the Research Department

```
SELECT Fname, Lname, Address
FROM Employee, Department
WHERE Dname = 'Research' AND Dnumber = Dno;
```