SQL: Data Definition Language

Anne Denton

Department of Computer Science
North Dakota State University

Outline

1. SQL Basics
   - Background
   - Schemas and Tables
2. Data Definition Statements
   - Domains
   - Constraints

Table of Contents

General Features

- Declarative (you say what you want, not how to get it)
- Acts as Data Definition Language and as Data Manipulation Language
- Can be embedded into general purpose programming languages
- It is maintained by ISO/IEC JTC 1, Information technology, Subcommittee SC 32, Data management and interchange
  - Different features beyond the standard are added for different DMBSs
  - Note that compliance level with standard for proprietary DBMSs is often lower than for open source DBMSs like PostGreSQL

History

- Evolved out of SEQUEL (Structured English QUEry Language) developed by IBM Research, and often pronounced as such
- First standardized in 1986
- For history check out `http://en.wikipedia.org/wiki/SQL`
- Has preserved properties of other old languages
  - Not case-sensitive
    - Causes more conflicts with reserved words
    - Upper case keywords conventional but not required
  - Equality is tested with single “=" sign
  - Boolean expressions use “AND", “OR", “NOT"

Terminologie

relational algebra	SQL	Databases
relation (set of tuples)	table (multiset)	file
tuple	row	record
attribute	column	attribute

Example

```
CREATE TABLE Department
( dept_no INT,
dept_name VARCHAR(50) NOT NULL,
building VARCHAR(50),
PRIMARY KEY (dept_no));

CREATE TABLE Student
( sid INT,
student_fname VARCHAR(50) NOT NULL,
student_lname VARCHAR(50) NOT NULL,
major_dept INT,
PRIMARY KEY (sid),
FOREIGN KEY (major_dept) REFERENCES
Department(dept_no));
```

Note the following

- Schema elements
  - Tables
  - Attributes
- Constraints
  - Domain constraints
  - Primary key constraints
  - Foreign key constraint
  - Constraint on null

Inserting records

- Use these insertions for later examples

```
INSERT INTO Department
VALUES (2740, 'Computer
Science', 'QBB');

INSERT INTO Department
VALUES (2755, 'Physics ',
'South Eng.');
```

```
INSERT INTO Student
VALUES (42, 'John', 'Doe',
2740);

INSERT INTO Student
VALUES (4711, 'Jane',
'Smith', 2740);

INSERT INTO Student
VALUES (815, 'Jack', 'Box',
NULL);
```

Listing Table Content

- Inspecting a table

  ```
  SELECT *
  FROM Department;
  ```

- Listing all tables

  Meta data is queried just as data

  ```
  SELECT *
  FROM pg_tables;
  ```

- In PostGreSQL, `\dt` is a shortcut for listing the tables in the current schema

Selection and Projection

- Selecting a row
  - Relational algebra: $\sigma_{sID=4711}$

    ```
    SELECT *
    FROM Student
    WHERE sid = 4711;
    ```
- Projecting to a column
  - Relational algebra: $\pi_{student\_lname}$

    ```
    SELECT student_lname
    FROM Student;
    ```

Joining two tables

- Relational algebra: *Student* $\bowtie$ *Department*
    - Conventional notation shows how join is constructed from Cartesian product, selection, and projection
      ```
      SELECT sid,
      student_fname, student_lname, dept_no,
      dept_name, building
      FROM Student, Department
      WHERE major_dept = dept_no;
      ```
    - Newer notation can be generalized to outer joins
      ```
      SELECT sid, student_fname, student_lname,
      dept_no, dept_name, building
      FROM Student INNER JOIN Department ON
      major_dept = dept_no;
      ```

Practical tips

- Change your password using \password
- Create a shortcut for the following after inserting the appropriate `dbname` and `user`

  ```
  mypsql=psql "dbname=[...]
  host=shinji.cs.ndsu.nodak.edu user=[...]
  port=5432 sslmode=require"
  ```

  in your `.profile` file, and use `source .profile`
- Quit psql using \q
- Don't forget the semicolon at the end of each statement (but if you do forget it, you can still add it on the next line)
- Make use of metadata with `\dt;`

More practical tips

- If you have a text file called `xyz.sql` in the directory from which you call `mypsql` you can use the `-f xyz.sql`
- Transfer `xyz.sql` to and from the lab using file transfer (sftp, scp, or WinSCP respectively)
- To avoid having a `CREATE TABLE` rejected because it already exists, drop table before recreating with same name, use `DROP TABLE MyTable CASCADE;`
- `CASCADE` deletes foreign key constraints that reference this table

Upper/lower Case and Comments

- Originally SQL required you to use upper case
    - Modern DBMSs convert all keywords and table, attribute, etc. names into upper case, so you can use whatever you are comfortable with
    - Note that you cannot use reserved words regardless of upper / lower case!

      `https://en.wikipedia.org/wiki/SQL_reserved_words`
    - The content of Character-String types is not converted
- Comments
    - ANSI-standard SQL supports double dash, –, as a single line comment identifier
    - Some extensions also support curly brackets or C style /* comments */ for multi-line comments.

Table of Contents

1. SQL Basics
   - Background
   - Schemas and Tables
2. Data Definition Statements
   - Domains
   - Constraints

Schemas

- Tables belong to schemas
- Not every user has the privilege to create a schema
- Schemas are sometimes tied to user accounts (as is the case in class)
- Schemas can also be set up to allow multiple users access (we have considered this for the project)

Tables

- Tables are created with
  `CREATE TABLE Department (...);`
- Tables are deleted with
  `DROP TABLE Department (...);`
- Foreign keys that reference the table are deleted using
  `DROP TABLE Department (...)  CASCADE;`

Question 1 Multiple answers can be correct

The keyword `CASCADE` after a DROP TABLE statement is necessary

1. When a foreign key that is defined in the table references an existing primary key in a different table
2. When the primary key that is defined in the table is referenced by a foreign key in a different table

Schema Evolution

- If requirements change the schema may have to change
- For example adding a column:

  ```
  ALTER TABLE Department
  ADD founded DATE;
  ```

- You can also drop columns, although you would have to be very careful to check that it isn't used

  ```
  ALTER TABLE Department
  DROP COLUMN founded;
  ```

- Some SQL implementations don't included dropping of columns
- Addition and removal of constraints is common, and will be discussed later in the notes

Question 2 Multiple answers can be correct

Adding a column to a table

1. Is often done as part of user transactions
2. Affects all previously created records
3. Is difficult to undo after others have built code to use the extra column

Table of Contents

1. SQL Basics
   - Background
   - Schemas and Tables
2. Data Definition Statements
   - Domains
   - Constraints

Numeric Types

- Integer numbers (INTEGER or INT, SMALLINT, etc.)
- Real numbers (FLOAT, DOUBLE PRECISION, etc.)
- DECIMAL(i,j) allows you to define the precision
  - precision (total number of decimal digits): i
  - scale (digits after the decimal point): j
  - 123.45 requires at least DECIMAL(5,2)
  - Doesn't use floating point notation
  - Keep in mind that for financial applications, correct rounding is very important

`http://www.cs.toronto.edu/~nn/csc309/guide/pointbase/docs/html/htmlfiles/dev_datatypesandconversionsFIN.html`

Question 3 (Multiple answers can be correct)

Assume that you are asked to create a financial application that is backed by a database table. You are asked to represent dollars and cents rounded to the nearest cent. What datatype to you pick?

1. DOUBLE PRECISON is best because the cents represent digits after the decimal point.
2. DECIMAL works well to represent dollars with two digits after the decimal point for the cents.
3. Whenever money is to be represented, numbers have to be converted to cents and INTEGER has to be used.

Character-string types: General considerations

- You have to use character types for numbers if they contain dashes, slashes, or letters
- You cannot use character types for numbers, if you want to use them in calculations
- Do not use character types for dates, times, time intervals, or anything that exists as a built-in type, because special functions and output formats are available and format is checked
- When entering character types, you use single quotes
- Double quotes are not normally used in DBMSs

Question 4 (Multiple answers can be correct)

Telephone numbers

1. Should be stored as INTEGER not a character-string type, because you should not include the slashes
2. Do not need to be stored as INTEGER because no mathematical operations are done on them, and INTEGER may not offer enough space
3. Once dashes and parentheses have been removed they could be stored in a character-string type that is defined to only allow digits

Character-string types: CHAR and VARCHAR

- `CHAR(n)`: fixed length of length `n`
  - If you enter '123'or 'ab'into a field defined as CHAR(4) it will be padded with blanks
  - The default of `n` is 1
  - CHAR provides space for a single character
- `VARCHAR(n)`: varying length with max. length `n`
  - `VARCHAR` is intended for variable character strings
- Some DBMSs also have a type `TEXT` for which you do not have to provide a maximum and/or allow `VARCHAR` to be used without maximum, which means arbitrary length
- The CREATE DOMAIN command allows more specific definitions
- Large quantities of text should be stored as Large Objects

Large Objects

- When data vary substantially in length or have undetermined content, large object types are most appropriate
- Binary Large OBjects: `BLOB`
- Can be used for video, audio, or any other type
- Some DBMSs have more specific terms, e.g. `CLOB` for large character objects, otherwise use `BLOB` for text as well

Boolean Data Type

- Standard SQL has BOOLEAN and, e.g. PostGreSQL implements it `http://www.postgresqltutorial.com/postgresql-boolean/`
- Not all other, especially proprietary DBMSs do, and the following is an interesting read `https://stackoverflow.com/questions/2426145/oracles-lack-of-a-bit-datatype-for-table-columns`
- This is because individual bits cannot be stored other than within a larger data type
- Note that, for processing, columns of bits can be combined into storage- and processing-efficient bitvectors, but not for record-level storage.

Question 5 (Multiple answers can be correct)

Representing 8 bits as one integer in a database

1. Would be less storage efficient than creating 8 Boolean attributes
2. Would be very difficult to maintain and cause confusion and is hence not recommended despite being more storage efficient than creating 8 Boolean attributes
3. Is not possible because integers and Booleans are different data types
4. Is highly recommended

Date and Time

- Standard SQL has types for `DATE`, `TIME` and `TIMESTAMP`
- Some proprietary systems require that even times be stored in the `DATE` type, and formatting be used to show the appropriate part
- `to_date()` and `to_char()` can be used `https://www.postgresql.org/docs/9.1/functions-formatting.html`

Question 6

Date types in relational databases

1. Are normally stored using two digits for the day, two digits for the month, and two digits for the year
2. Allow for using multiple formats that can be selected by using the `to_date()` and `to_char()` functions
3. Should be used for dates instead of `VARCHAR` if at all possible

Table of Contents

Inline Definition of Constraints

- Most constraints that apply to only a single attribute can be specified directly after the attribute
  - `NOT NULL`
  - `DEFAULT <value>`
  - `PRIMARY KEY`
  - `UNIQUE` for an alternate keys

```
CREATE TABLE Department
( ...
dept_name VARCHAR(50) NOT NULL DEFAULT 'Unassigned',
...)
```

- Multiple constraints are listed sequentially
- `NOT NULL` is automatically enforced when you designate `PRIMARY KEY`
  - No separating comma
- Some DBMSs allow this inline definitions for foreign key constraints

Defining constraints on separate line

- Defining keys on separate lines adds to clarity
- Composite keys have to be defined on a separate line

  ```
  PRIMARY KEY (attr1, attr2, ...)
  ```
- Foreign key definition
  - Table that defines the primary key must have been created
  - When attribute names match between tables, they do not have to be listed:

    ```
    FOREIGN KEY (attr1, attr2, ...)  REFERENCES
    OtherTable
    ```
  - When attribute names do not all match between tables, they have to be listed:

    ```
    FOREIGN KEY (attr1, attr2, ...)  REFERENCES
    OtherTable (otherAttr1, otherAttr2, ...)
    ```
- Using a separate line is also needed for naming constraints

Question 7

When a foreign key is defined in SQL

1. It is in the table that references another
2. It is in the table that is referenced by another
3. For composite keys, it has to be done separately for each attribute
4. The referenced primary key must already have been defined previously

Naming constraints

- To name constraints, it is usually best to define them on a separate line
- Foreign key constraints should normally be named so they can be dropped:

```
CREATE TABLE Student
( ...
majorDept INT,
...
CONSTRAINT major_dept_const
FOREIGN KEY (major_dept) REFERENCES
Department(dept_no));
```

Separate definition of constraints

- Some recommend adding foreign key constraints after all table definitions using `ALTER TABLE`, such that the ordering of table definitions does not matter:

  ```
  ALTER TABLE Student ADD CONSTRAINT major_dept_const
  FOREIGN KEY (major_dept) REFERENCES Department(dept_no);
  ```

- That allows dropping them later

  ```
  ALTER TABLE Student DROP CONSTRAINT major_dept_const;
  ```

Question 8 (Multiple answers may be correct)

Adding foreign keys after tables have been created

1. Causes the need for schema evolution early and is not recommended
2. Allows defining tables in any order

Referentially Triggered Actions: Basic concepts

- Only the referential integrity constraint can be violated through deletions, and such violations can be handled through automatically triggered actions
- In principle, referentially triggered actions can be defined `ON DELETE` and `ON UPDATE`, but they have relatively little practical relevance, since primary keys should be chosen, so they do not have to be changed
- Remember that referentially triggered actions happen when a record is deleted from the table that defines that primary key that is being referenced
- The default is to reject deletions that would violate referential integrity
  Nothing has to be specified

Question 9 (Multiple answers can be correct)

In order to disallow the deletion of a record that contains a primary key that is still being reference by a foreign key

1. Choose the referentially triggered action ON DELETE REJECT
2. You do not specify a referentially triggered action
3. (...) It does not matter how the sentence is completed since you should not disallow this. Specifying a referentially triggered action is preferable.

Referentially Triggered Actions: Specifics

- If records that reference a no longer existing primary key are to be deleted use `ON DELETE CASCADE`
- If the foreign values that reference a no longer existing primary key are to be set null use `ON DELETE SET NULL` or to a default `ON DELETE SET DEFAULT`

```
CREATE TABLE Course
( ...
deptNo VARCHAR(4),
...
FOREIGN KEY (dept_no) REFERENCES Department
ON DELETE CASCADE);
```

- Note that not all DBMSs offer all combinations of referentially triggered actions