Introduction to SQL and C Programming API

ECE 650
Systems Programming & Engineering
Duke University, Spring 2016

SQL

- Structured Query Language
- Major reason for commercial success of relational DBs
  - Became a standard for relational DBs
  - Used by many database management systems (DBMS)
  - Makes it easier to move DB apps from one DBMS to another
    - If DB apps use only features that are part of the standard
  - Also lets DB apps access data stored in multiple DBMS's

Relational Algebra vs. SQL Queries

- Relational algebra written as a sequence of operations
  - Requires specifying the *order* to execute query operations
  - This is complex and restrictive for users
- SQL language provides high-level declarative language
  - User specifies only *what* the result should be
  - DBMS optimizes and decides about how to execute query

SQL Terminology

- Table = Relation
- Row = Tuple
- Column = Attribute
- Commands for data definition are
  - CREATE, ALTER, DROP
- One basic command for retrieving (querying) information
  - SELECT

Tables

- 'CREATE TABLE' command creates a new relation
  - Give table a name, specify its attributes and constraints
  - For each attribute in the table:
    - Attribute name, data type (domain of values), constraints
  - Key, entity and referential integrity constraints for the table specified after the list of attributes
- 'DROP TABLE' command removes a table

CREATE TABLE Example

```
CREATE TABLE Employee
  ( FNAME     VARCHAR(15)     NOT NULL,
    MINIT     CHAR,
    LNAME     VARCHAR(15)     NOT NULL,
    SSN       CHAR(9)         NOT NULL,
    BATE      DATE,
    ADDRESS   VARCHAR(30),
    SALARY    DECIMAL(10,2),
    SUPERSSN  CHAR(9),
    DNO       INT             NOT NULL,
  PRIMARY KEY (SSN),
  FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN)
  FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNUMBER));
```

SQL Domains (Data Types)

- Numeric types
  - INT, SMALLINT
  - FLOAT, REAL, DOUBLE PRECISION
  - Formats: DECIMAL(i,j) (i=precision, j=scale)
- Character string
  - Fixed length: CHAR(n) or CHARACTER(n)
  - Variable length: VARCHAR(n) or CHAR VARYING(n)
    - n=max # of chars
  - Bit string: BIT(n) or BIT VARYING(n)
- Date and Time
  - DATE=YYYY-MM-DD, TIME=HH:MM:SS
  - TIMESTAMP includes both date and time
- Can also create a domain (like a typedef)
  - CREATE DOMAIN SSN_TYPE AS CHAR(9)

Default Values

- Can define a default value for an attribute
- Use DEFAULT <value> notation
  - If not specified, default is Null
- E.g.:

```
CREATE TABLE Employee
  ( FNAME     VARCHAR(15)     NOT NULL,
    <snip>
    DNO       INT             NOT NULL DEFAULT 1,
  PRIMARY KEY (SSN),
  FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN)
  FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNUMBER));
```


Referential Integrity Actions

- What should happen if referential integrity is violated
  - Recall this can happen as tuples are inserted or deleted
- Can specify a *referential triggered action* on foreign key
  - Options are:
    - SET NULL – Set foreign key attribute to NULL
    - CASCADE – Set foreign key attribute to updated value
    - SET DEFAULT – Set foreign key to default value
  - Must be qualified with one of:
    - ON DELETE – If tuple referenced by foreign key is deleted
    - ON UPDATE – If tuple referenced by foreign key is updated

Referential Integrity Actions (2)

- Examples
  - SET NULL ON DELETE: If tuple referenced by a foreign key is deleted, set the foreign key field to NULL in referencing tuples
  - CASCADE ON UPDATE: If tuple referenced by a foreign key is updated, update the foreign key value in referencing tuples
  - CASCADE ON DELETE: If a tuple referenced by a foreign key is deleted, delete referencing tuples
- Can also give a constraint a name (optional)

```
CREATE TABLE Employee
  ( <snip>
  CONSTRAINT EMPPK PRIMARY KEY (SSN),
  CONSTRAINT EMPSUPERFK FOREIGN KEY (SUPERSSN) REFERENCES
EMPLOYEE(SSN) ON DELETE SET NULL ON UPDATE CASCADE,
  <SNIP>);
```


Modify a Table

- ALTER TABLE
  - Add or drop columns (attributes)
  - Change column definitions
  - Add or drop table constraints
- Add attribute to a table:
  - ALTER TABLE Employee ADD JOB VARCHAR(12);
- Drop attribute from a table
  - ALTER TABLE Employee DROP ADRESS CASCADE;
    - Must choose either CASCADE or RESTRICT
    - CASCADE: constraints referencing this column are also dropped
    - RESTRICT: operation only succeeds if no constraints refer to column

Basic Queries

- SELECT statement
  - For retrieving database information
- Distinction between SQL and formal relational model
  - SQL allows a tale to have 2 or more tuples identical in all values
  - SQL table is thus not a *set* of tuples
    - It is a *multiset*
  - Some SQL relations are constrained to be sets
    - Due to key constraint
  - Something to be aware of as we discuss queries

Example Relational Database Tables

- COMPANY = {EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS, PROJECT, WORKS_ON, DEPENDENT}

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEP_NAME	SEX	BDATE	RELATIONSHIP

SELECT-FROM-WHERE

- Basic SELECT statement form:
  - SELECT <attribute list> // list of attribute names to return
  - FROM <table list> // list of table names to process the query
  - WHERE <condition>; // conditional expression to identify tuples
- Example:
  - SELECT BDATE, ADDRESS FROM EMPLOYEE WHERE FNAME='John' AND MINIT='B' AND LNAME='Smith';
  - Similar to the relational algebra expression:
    - $\pi_{BDATE, ADDRESS}(\sigma_{FNAME='John' \text{ AND } MINIT='B' \text{ AND } LNAME='Smith'}(EMPLOYEE))$
  - SELECT-clause specifies projection attributes
  - WHERE-clause specifies selection condition

Multiple Tables

- SELECT FNAME, LNAME, ADDRESS
  FROM EMPLOYEE, DEPARTMENT
  WHERE DNAME='Research' AND DNUMBER=DNO
  - Like a SELECT-PROJECT-JOIN sequence of relational algebra ops
  - DNAME='Research' is a *selection condition*
  - DNUMBER=DNO is a *join condition*
- SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE
  FROM PROJECT, DEPARTMENT, EMPLOYEE
  WHERE DNUM=DNUMBER AND MGRSSN=SSN AND PLOCATION='Stafford'
  - Two join conditions here
  - DNUM=DNUMBER relates a project to its controlling department
  - MGRSSN=SSN relates the controlling department to the employee managing it

Dealing with Ambiguous Attribute Names

- Same name may be used by different attributes in different tables (relations)
- In that case, must qualify the attribute name with relation name
  - Prefix relation name to attribute name
  - Separate two by a period
- For example, if both EMPLOYEE and DEPARTMENT tables used fields named NAME AND DNUMBER (instead of DNAME AND DNO)
- SELECT FNAME, LNAME, ADDRESS
  FROM EMPLOYEE, DEPARTMENT
  WHERE DEPARTMENT.NAME='Research' AND
  DEPARTMENT.DNUMBER=EMPLOYEE.DNUMBER

Aliasing

- Can declare alternative relation names
  - And even attribute names for the relation
- SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
  FROM EMPLOYEE AS E, DEPARTMENT AS S
  WHERE E.SUPERSSN=S.SSN;
- Think of E and S as two copies of same table
  - Allows us to join the two copies of the same table
  - Shows manager name for each employee name
- Can also alias the attribute names
  - EMPLOYEE AS E(FN, MI, LN, SSN, BD, ADDR, SEX, SAL, SSSN, DNO)

Unspecified WHERE-Clause

- SELECT SSN FROM EMPLOYEE;
  - Select all EMPLOYEE SSNs
- SELECT SSN, DNAME FROM EMPLOYEE, DEPARTMENT;
  - Select all combinations of EMPLOYEE SSN and DEPARTMENT DNAME
- Important to specify every selection and join condition in the WHERE Clause
  - Otherwise may end up w/ very large result relations (cross product)

Retrieving All Attributes

- What if we want all attributes of a high-degree table
  - Do not need to list them all in SELECT Clause
  - Can use the asterisk (*)
- SELECT * FROM EMPLOYEE WHERE DNO=5;
  - Retrieve all attributes of EMPLOYEE tuples who work in department number 5
- SELECT * FROM EMPLOYEE, DEPARTMENT WHERE DNAME='Research' AND DNO=DNUMBER
  - Retrieve all attributes of an EMPLOYEE and all attributes of their DEPARTMENT for every employee of 'Research' department

SQL vs. MySQL

- SQL
  - Declarative computer language
  - Intended for querying relational databases
- MySQL
  - A relational database
  - A piece of software optimized for data storage and retrieval
  - Can submit SQL queries to retrieve info from a database
  - There are other example databases:
    - Oracle, Microsoft SQL Server, SQLite
- In our programming assignment:
  - Will use MySQL to create database
  - C API allows programs to send SQL queries

MySQL

- Install a MySQL database server
  - sudo apt-get install mysql-server
  - Set a root DB password on installation
    - Root DB server access needed to create new databases
- Start and stop the database server
  - sudo service mysql start
  - sudo service mysql stop
- Connect to database server
  - mysql -u root -p //or create a new non-root user
  - help
  - Ctrl+L //clear screen
  - Ctrl+D //quit
  - system <cmd>

MySQL

- Create a database
  - CREATE DATABASE mydb;
  - SHOW DATABASES;
  - DROP DATABASE mydb;
- Create a new non-root user
  - When connected to database server as root
  - mysql> CREATE USER uname@localhost IDENTIFIED BY '34klq*';
  - mysql> GRANT ALL ON mydb.* to uname@localhost;

MySQL C API

- Install the API
  - sudo apt-get install libmysqlclient-dev
- Let's look at some example code
  - Posted to class web page as well