SQL Overview
Defining a Schema

CPSC 315 – Programming Studio

Project 1, Lecture 3

Slides adapted from those used by Jeffrey Ullman, via Jennifer Welch

Generations of Programming Languages

• 1st generation
  – Machine code
• 2nd generation
  – Human-readable but directly related to processor
  – Assembly language, C (sort of)
• 3rd generation
  – Abstraction from processor, easier for humans
  – Fortran, C/C++, Java, etc.
• 4th generation
  – Programming Language for specific task
  – e.g. SQL, Matlab
• 5th generation
  – Give constraints (goal), and result follows logically
  – e.g. Prolog

SQL

• Structured Query Language
• Database language used to manage and query relational databases
• A well-known, commonly used standard
  – Regularly updated
• Many extensions, variations
  – Platform-specific versions, etc.

SQL Elements

• Data Definition Language (DDL)
  – Supports creation of database schema
• Data Manipulation Language (DML)
  – Supports entering/removing data
• Querying Language
  – Supports query operations (don’t change data itself)
• Others:
  – Transaction control, Data control
Our Discussion of SQL

- Will highlight some of the structures and features of SQL
- Give you an idea of the basics of how it works
  - Reflects how relational databases work
  - Not meant to make you SQL programmers
- You will need to implement equivalent functions for parts of what we discuss

Database Schema

- The set of relations (tables) in the database.
- Create, delete, change tables

CREATE

- Define a relation
CREATE TABLE <name> ( 
    <element list>
);

Element Types

- INT, INTEGER
  - Integers
- FLOAT, REAL
  - Floating-Point numbers
- CHAR(n)
  - Fixed-length string of n characters
- VARCHAR(n)
  - Variable-length string of up to n characters
- DATE
  - yyyy-mm-dd
- TIME
  - hh:mm:ss
Example

CREATE TABLE HouseRep (  
    Name VARCHAR(80),  
    Party CHAR(10),  
    Birthdate DATE,  
    YearsInCongress INT,  
    Salary REAL  
);

Declaring Keys

- Keys declared within CREATE statement
- Key attributes functionally determine all other attributes in the relation
- List under PRIMARY KEY
  - Elements of primary key can not be NULL

Example

CREATE TABLE HouseRep (  
    Name VARCHAR(80),  
    Party CHAR(10),  
    Birthdate DATE,  
    YearsInCongress INT,  
    Salary REAL,  
    PRIMARY KEY (Name)  
);

Example

CREATE TABLE HouseRep (  
    Name VARCHAR(80),  
    Party CHAR(10),  
    Birthdate DATE,  
    YearsInCongress INT,  
    Salary REAL,  
    PRIMARY KEY (Name, Birthdate)  
);
Other Element Modifiers

- **UNIQUE**
  - Placed after type
  - Only one tuple in that relation for each value (except NULL)
  - Can imply key if no primary key given
  - Can be NULL

- **NOT NULL**
  - Cannot take value NULL

- **DEFAULT**
  - Default value specified

Example

```sql
CREATE TABLE HouseRep (
    Name VARCHAR(80) UNIQUE,
    Party CHAR(10),
    Birthdate DATE NOT NULL,
    YearsInCongress INT
            DEFAULT 0,
    Salary REAL
            DEFAULT 120000.00
);
```

Other Table Modifications

- **DROP <name>**
  - Deletes that table

- **ALTER TABLE <name> ADD <attribute>**
  - Adds a new column to table

- **ALTER TABLE <name> DROP <attribute>**
  - Removes the column from the table

Views

- **Views are a sort of “virtual table”, usually created as the result of a query**
  - We'll discuss queries later

- **Format:**
  ```sql
  CREATE VIEW <name> AS <query>
  ```