Introduction to SQL

Dr. Shrutilipi Bhattacharjee, Assistant Professor, Dept. of IT, NIT Karnataka, India

History

- IBM Sequel language developed as part of System R project at the IBM San Jose Research Laboratory
- Renamed Structured Query Language (SQL)
- ANSI and ISO standard SQL:
  - SQL-86
  - SQL-89
  - SQL-92
  - SQL:1999 (language name became Y2K compliant!)
  - SQL:2003
- Commercial systems offer most, if not all, SQL-92 features, plus varying feature sets from later standards and special proprietary features
  - Not all examples here may work on your particular system

SQL Parts

- **DML:** Provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database
- **Integrity:** The DDL includes commands for specifying integrity constraints
- **View definition:** The DDL includes commands for defining views
- **Transaction control:** Includes commands for specifying the beginning and ending of transactions
- **Embedded SQL and dynamic SQL:** Define how SQL statements can be embedded within general-purpose programming languages
- **Authorization:** Includes commands for specifying access rights to relations and views

Data Definition Language

- The SQL **data-definition language (DDL)** allows the specification of information about relations, including:
  - The schema for each relation
  - The type of values associated with each attribute
  - The Integrity constraints
  - The set of indices to be maintained for each relation
  - Security and authorization information for each relation
  - The physical storage structure of each relation on disk

Domain Types in SQL

- **char(n):** Fixed length character string, with user-specified length *n*
- **varchar(n):** Variable length character strings, with user-specified maximum length *n*
- **int:** Integer (a finite subset of the integers that is machine-dependent)
- **smallint:** Small integer (a machine-dependent subset of the integer domain type)
- **numeric(p, d):** Fixed point number, with user-specified precision of *p* digits, with *d* digits to the right of decimal point (ex., **numeric**(3,1), allows 44.5 to be stores exactly, but not 444.5 or 0.32)
- **real, double precision:** Floating point and double-precision floating point numbers, with machine-dependent precision
- **float(n):** Floating point number, with user-specified precision of at least *n* digits

Create Table Construct

- An SQL relation is defined using the **create table** command:

create table *r*
(*A*$_1$ *D*$_1$, *A*$_2$ *D*$_2$, ..., *A*$_n$ *D*$_n$,
(integrity-constraint$_1$),
...,
(integrity-constraint$_k$))

- ***r*** is the name of the relation
- each A_i is an attribute name in the schema of relation ***r***
- D_i is the data type of values in the domain of attribute A_i
- Example:

create table *instructor* (
ID **char**(5),
name **varchar**(20),
dept_name **varchar**(20),
salary **numeric**(8, 2))

Integrity Constraints in Create Table

- Types of integrity constraints
  - **primary key** (A_1, ..., A_n)
  - **foreign key** (A_m, ..., A_n) **references** *r*
  - **not null**
- SQL prevents any update to the database that violates an integrity constraint
- Example:

create table *instructor* (
ID **char**(5),
name **varchar**(20) **not null**,
dept_name **varchar**(20),
salary **numeric**(8, 2),
primary key (*ID*),
foreign key *(dept_name*) **references** *department);*

Relation Definitions

```
create table student (
        ID                  varchar(5),
        name                varchar(20) not null,
        dept_name           varchar(20),
        tot_cred            numeric(3,0),
        primary key (ID),
        foreign key (dept_name) references department);
```

```
create table course (
        course_id           varchar(8),
        title               varchar(50),
        dept_name           varchar(20),
        credits             numeric(2,0),
        primary key (course_id),
        foreign key (dept_name)
        references department);
```

```
create table takes (
        ID                  varchar(5),
        course_id           varchar(8),
        sec_id              varchar(8),
        semester            varchar(6),
        year                numeric(4,0),
        grade               varchar(2),
        primary key (ID, course_id, sec_id, semester, year),
        foreign key (ID) references  student,
        foreign key (course_id, sec_id, semester, year) references section);
```

Updates to Tables

- **Insert**
  - **insert into** *instructor* **values** ('10211', 'Smith', 'Biology', 66000);
- **Delete**
  - Remove all tuples from the *student* relation
    - **delete from** *student*
- **Drop Table**
  - **drop table** *r*
- **Alter**
  - **alter table** *r* **add** *A D*
    - Where *A* is the name of the attribute to be added to relation *r* and *D* is the domain of *A*
    - All exiting tuples in the relation are assigned *null* as the value for the new attribute
  - **alter table** *r* **drop** *A*
    - Where *A* is the name of an attribute of relation *r*
    - Dropping of attributes not supported by many databases

Basic Query Structure

- A typical SQL query has the form:

select A_1, A_2, ..., A_n
from r_1, r_2, ..., r_m
where P

  - A_i represents an attribute
  - R_i represents a relation
  - P is a predicate

- The result of an SQL query is a relation

The *Select* Clause

- The **select** clause lists the attributes desired in the result of a query
  - Corresponds to the projection operation of the relational algebra

- Example: Find the names of all ***instructors***:

  select *name* **from** *instructor*

- **Note:** SQL names are case insensitive (i.e., you may use upper- or lower-case letters)
  - E.g., *Name* ≡ *NAME* ≡ *name*
  - Some people use upper case wherever we use bold font

The *Select* Clause

- SQL allows duplicates in relations as well as in query results
- To force the elimination of duplicates, insert the keyword **distinct** after select
- Find the department names of all ***instructors***, and remove duplicates

  select distinct *dept_name* **from** *instructor*
- The keyword **all** specifies that duplicates should not be removed

  select all *dept_name* **from** *instructor*

dept_name
Comp. Sci.
Finance
Music
Physics
History
Physics
Comp. Sci.
History
Finance
Biology
Comp. Sci.
Elec. Eng.

The *Select* Clause

- An asterisk in the select clause denotes “all attributes”

  select * **from** *instructor*

- An attribute can be a literal with no **from** clause

  select '437'

  – Results is a table with one column and a single row with value “437”

  – Can give the column a name using:

  select '437' **as** *AAA*

- An attribute can be a literal with **from** clause

  select ‘A’ **from** *instructor*

- Result is a table with one column and *N* rows (number of tuples in the *instructors* table), each row with value “A”

The *Select* Clause

- The **select** clause can contain arithmetic expressions involving the operation, +, –, *, and /, and operating on constants or attributes of tuples

- The following query would return a relation that is the same as the *instructor* relation, except that the value of the attribute *salary* is divided by 1:

select *ID, name, salary/12* **from** *instructor*

- Can rename "s*alary/12"* using the **as** clause:

select *ID, name, salary/12* **as** *monthly_salary*

The *Where* Clause

- The **where** clause specifies conditions that the result must satisfy
  - Corresponds to the selection predicate of the relational algebra

- To find all **instructors** in Comp. Sci. dept

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.'

- SQL allows the use of the logical connectives **and, or,** and **not**

- The operands of the logical connectives can be expressions involving the comparison operators <, <=, >, >=, =, and <>

- Comparisons can be applied to results of arithmetic expressions

- To find all **instructors** in Comp. Sci. dept with salary > 70000

  select *name*
  from *instructor*
  where *dept_name = 'Comp. Sci.'* ***and*** *salary > 70000*

name
Katz
Brandt

The *From* Clause

- The **from** clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation of the relational algebra

- Find the Cartesian product ***instructor* × *teaches*** generates every possible instructor – teaches pair, with all attributes from both relations

  select * **from** *instructor, teaches*

  - For common attributes (e.g., *ID*), the attributes in the resulting table are renamed using the relation name (e.g., *instructor.ID*)

- Cartesian product not very useful directly, but useful when combined with where-clause condition (selection operation in relational algebra)

Examples

- Find the names of all instructors who have taught some course and the *course_id*

  select *name, course_id*
  from *instructor, teaches*
  where *instructor.ID = teaches.ID*

- Find the names of all instructors in the Art ***department*** who have taught some course and the *course_id*

  select *name, course_id*
  from *instructor, teaches*
  where *instructor.ID = teaches.ID* ***and***
  instructor.dept_name = 'Art'

name	*course_id*
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

The *Rename* Operation

- The SQL allows renaming relations and attributes using the **as** clause:

  old-name **as** *new-name*

- Find the names of all ***instructors*** who have a higher salary than some instructor in 'Comp. Sci'.

  select distinct *T.name*
  from *instructor* **as** *T, instructor* **as** *S*
  where *T.salary > S.salary* **and** *S.dept_name = 'Comp. Sci.'*

- Keyword **as** is optional and may be omitted

  instructor **as** *T* ≡ *instructor T*

Next Lecture

Introduction to SQL

Thank you for your attention...

Any question?

Contact:
Department of Information Technology, NITK Surathkal, India
6th Floor, Room: 13
Phone: +91-9477678768
E-mail: shrutilipi@nitk.edu.in

Self Join Example

- Relation *emp-super*

person	*supervisor*
Bob	Alice
Mary	Susan
Alice	David
David	Mary

- Find the supervisor of “Bob”
- Find the supervisor of the supervisor of “Bob”
- Can you find ALL the supervisors (direct and indirect) of “Bob”?