SQL Fundamentals

COURSE : 23CAT- Database Management System
UNIT I : Introduction
CLASS : I Semester / I MCA

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 Aspects

- Data Definition Language
- Data Manipulation Language
- Embedded and Dynamic SQL
- Triggers
- Transaction Management
- Security
- Client server Execution and remote database access

The **SQL data-definition language** (DDL) allows the specification of information about relations, including:

- The schema for each relation.
- The domain of values associated with each attribute.
- Integrity constraints
- And as we will see later, also other information such as
  - The set of indices to be maintained for each relations.
  - 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 maximum length n
- **Int** - Integer (a finite subset of the integers that is machine-dependent)
- **Smallint -** Small integer (a machine-dependent)
- **numeric(p,d)** - Fixed point number
- **Real -** double precision
- float(n) - Floating point number, with user-specified precision of at least n digits.

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

```
create table r (A1 D1, A2 D2, ..., An Dn,
                (integrity-constraint 1),  ...,
                (integrity-constraint k))
```

- R-> name of the relation, Ai an attribute name in the schema of relation r
- Di is the data type of values in the domain of attribute Ai

Example:

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

- **not null**
- **primary key (A1, ..., An)**
- **foreign key (Am, ..., An) references r**

primary key declaration on an attribute automatically ensures **not null**

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);
```

More Examples

```
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 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 in Table

- **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 in 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

  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
- **select** clause lists the attributes desired in the result of a query
- To force the elimination of duplicates, insert the keyword **distinct** after select
- The keyword **all** specifies that duplicates should not be removed

An asterisk in the select clause denotes “all attributes”

- **select** clause can contain arithmetic expressions
- Rename expression using the **as** clause

- **where** clause specifies conditions
- Conditions can be combined using the logical connectives **and, or,** and **not**

- **from** clause lists the relations

SQL allows renaming relations and attributes using the **as** clause:
old-name **as** *new-name*

Queries Example

select distinct *dept_name* **from** *instructor*

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

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

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

```
select name from instructor
where salary between 90000
and 100000
```

```
select name, course_id from instructor,
teaches where (instructor.ID, dept_name) =
(teaches.ID, 'Biology');
```

String Operations

- string-matching operator for comparisons on character strings
- operator **like** uses patterns that are described using two special characters:
  percent (%). The % character matches any substring
  underscore (_). The _ character matches any character
- Example

```
select name from instructor
where name like '%dar%'
```

- Intro%' matches any string beginning with "Intro"
- '_ _ _' matches any string of exactly three characters.
- '_ _ _ %' matches any string of at least three characters
- concatenation (using "||")

- List in alphabetic order the names of all instructors

```
select distinct name from instructor
order by name
```

- We may specify ***desc*** for descending order or ***asc*** for ascending order, for each attribute; ascending order is the default.

  Example: order by name desc

- Can sort on multiple attributes

  Example: order by dept_name, name

- Find courses that ran in Fall 2009 or in Spring 2010

```
(select course_id from section where sem = 'Fall' and year = 2009)
 union
(select course_id from section where sem = 'Spring' and year = 2010)
```

- Find courses that ran in Fall 2009 and in Spring 2010

```
(select course_id from section where sem = 'Fall' and year = 2009)
 intersect
(select course_id from section where sem = 'Spring' and year = 2010)
```

- Find courses that ran in Fall 2009 but not in Spring 2010

```
(select course_id from section where sem = 'Fall' and year = 2009)
 except
(select course_id from section where sem = 'Spring' and year = 2010)
```

Aggregate functions

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values
group by: group items

select avg (*salary*) **from** *instructor*
where *dept_name*= 'Comp. Sci.';c

select count (**distinct** *ID*) **from** *teaches*
where *semester* = 'Spring' **and** *year* = 2010

select count (*) **from** *course*;

select dept_name, avg (salary) as avg_salary
from instructor group by dept_name

More Examples

```
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 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 in Table

- **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 in 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.

Architectural Design Challenges