Introduction to SQL

Lecture 3

Dr. Reda M. Hussien

Data Definition Language

- 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 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.
- `real, double` 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 (A1 D1, A2 D2,..., An Dn,
(integrity_constraint1),..., (integrity_constraintk) );
```

- r is the name of the relation, each Ai is 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) );
```

Integrity Constraints in Create Table

- not null
- primary key $(A_1, A_2, ..., A_n)$
- foreign key $(A_m, ..., A_n)$ references R

```
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);
```

- primary key declaration on an attribute automatically ensures not null

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;
```

Updates to tables

- Alter

$$\texttt{alter table } R \texttt{ add } A\ D;$$

  - 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.

$$\texttt{alter table } R \texttt{ drop } A;$$

  - 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

$$\texttt{select}\ A_1 A_2,\ A_3, \ldots, A_n$$

$$\texttt{from}\ R_1,\ R_2, \ldots,\ R_m$$

$$\texttt{where}\ P$$

  - 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 (Cont.)

- 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;
```

The select Clause (Cont.)

- 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”

The select Clause (Cont.)

- Can give the column a name using:

```
select '437' as FOO
```

- 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 (Cont.)

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

```
select ID, name, salary / 12
from instructor;
```

  - would return a relation that is the same as the instructor relation, except that the value of the attribute salary is divided by 12.
  - Can rename “salary/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.';
```

The where Clause

- Comparison results can be combined using the logical connectives and, or, and not
  - To find all instructors in Comp. Sci. dept with salary > 80000

```
select name from instructor
where dept_name = 'Comp.Sci.'  and salary > 80000;
```

- Comparisons can be applied to results of arithmetic expressions.

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 X teaches

```
select * from instructor, teaches;
```

  - generates every possible instructor – teaches pair, with all attributes from both relations.
  - 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 combined with where-clause condition (selection operation in relational algebra).

Cartesian Product

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000

teaches

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009

Inst.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2009
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
12121	Wu	Pinance	90000	10101	CS-347	1	Fall	2009
12121	Wu	Pinance	90000	12121	FIN-201	1	Spring	2010
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2010
12121	Wu	Pinance	90000	22222	PHY-101	1	Fall	2009
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…

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';
```

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

End of Lecture 3