The picture can't be displayed.

Database Systems

Lecture 10: Intermediate SQL (part 2)

Dr. Momtazi
momtazi@aut.ac.ir

based on the slides of the course book

Outline

- Join Expressions
- Views
- Transactions
- Integrity Constraints
- **SQL Data Types and Schemas**
- Authorization

Built-in Data Types in SQL

- **date:** Dates, containing a (4 digit) year, month and date
  - Example: **date** ‘2005-7-27’
- **time:** Time of day, in hours, minutes and seconds.
  - Example: **time** ‘09:00:30’ **time** ‘09:00:30.75’
- **timestamp**: date plus time of day
  - Example: **timestamp** ‘2005-7-27 09:00:30.75’
- **interval:** period of time
  - Example: interval ‘1’ day
  - Subtracting a date/time/timestamp value from another gives an interval value
  - Interval values can be added to date/time/timestamp values

Default Values

- **create table** *student*
  (*ID* **varchar** (5),
  name **varchar** (20) **not null**,
  dept_name **varchar** (20),
  tot_cred **numeric** (3,0) **default** 0,
  primary key (*ID*))

insert into *student(ID,name,dept_name)*

values('12789', 'Newman', 'Comp. Sci.');

Index Creation

- **create table** *student*
  (*ID* **varchar** (5),
  name **varchar** (20) **not null**,
  dept_name **varchar** (20),
  tot_cred **numeric** (3,0) **default** 0,
  primary key (*ID*))

create index *studentID_index* **on** *student*(*ID*)

Index Creation

- Indices are data structures used to speed up access to records with specified values for index attributes
  - e.g. **select ***
    from *student*
    where *ID* = ‘12345’

  can be executed by using the index to find the required record, without looking at all records of *student*

  More on indices in Chapter 11

Large-Object Types

- Large objects (photos, videos, CAD files, etc.) are stored as a *large object*:
  - **blob**: binary large object -- object is a large collection of uninterpreted binary data (whose interpretation is left to an application outside of the database system)
  - **clob**: character large object -- object is a large collection of character data

book_review **clob**(10KB)

image **blob**(10MB)

movie **blob**(2GB)

Large-Object Types

- When a query returns a large object, a “locator” is returned rather than the large object itself.
- The locator can then be used to fetch the large object in small pieces, rather than all at once
- Much like reading data from an operating system file using a read function call

User-Defined Types

- SQL supports two forms of user-defined data types:
  - distinct types
  - structured data types
    - allows the creation of complex data types with nested record structures, arrays and multisets (Chapter 22)

User-Defined Types

- **create type** construct in SQL creates user-defined type

  create type *Dollars* **as numeric (12,2) final**

  - **create table** *department*
    (*dept_name* **varchar** (20),
    building **varchar** (15),
    budget Dollars);

- NOTE: The keyword final isn’t really meaningful in this context but is required by theSQL:1999 standard; some implementations allow the final keyword to be omitted.

User-Defined Types

- It is possible for several attributes to have the same data type.
  - e.g., the name attributes for student name and instructor (the set of all person names)
  - but not instructor name and dept_name (we would normally not consider the query "Find all instructors who have the same name as a department")
  - ⇒ assigning an instructor's name to a department name is probably a programming error
  - Similarly, comparing a monetary value expressed in dollars and pounds

create type *Dollars* **as numeric (12,2) final**
create type *Pounds* **as numeric (12,2) final**

User-Defined Types

- Declaring different types for different attributes results to strong type checking
  - e.g., (department.budget+20) would not be accepted
    - The attribute and the integer constant 20 have different types

- Solution:
  - Values of one type can be cast (converted) to another domain:

cast *(department.budget* **to** *numeric (12,2))*

Domains

- **create domain** construct in SQL-92 creates user-defined domain types

  create domain *person_name* **char**(20) **not null**

- Types and domains are similar. Domains can have constraints, such as **not null**, specified on them.

- **create domain** *degree_level* **varchar**(10)
  constraint *degree_level_test*
  check (**value in** (’Bachelors’, ’Masters’, ’Doctorate’));

Create Table Extensions

- Creating tables that have the same schema as an existing table.

create table *temp_instructor* **like** *instructor*

create table *t1* **as**

    (**select** *

    from *instructor*

    where *dept_name= 'Music'*)

with data

Create Table Extensions

- **create table … as** statement closely resembles the create view statement and both are defined by using queries.
- The main difference is that the contents of the table are set when the table is created, whereas the contents of a view always reflect the current query result

Outline

- Join Expressions
- Views
- Transactions
- Integrity Constraints
- SQL Data Types and Schemas
- **Authorization**

Authorization

- Forms of authorization on parts of the database:
  - **Read** - allows reading, but not modification of data.
  - **Insert** - allows insertion of new data, but not deletion or updating of existing data.
  - **Update** - allows updating, but not insertion or deletion of data.
  - **Delete** - allows deletion of data, but not insertion or updating.
- Each of these authorization types is called a **privilege**
- A user who creates a new relation is given all privileges on that relation automatically

Authorization

Forms of authorization to modify the database schema

- **Index** - allows creation and deletion of indices.
- **Resources** - allows creation of new relations.
- **Alteration** - allows addition or deletion of attributes in a relation.
- **Drop** - allows deletion of relations.

Authorization Specification in SQL

- The **grant** statement is used to confer authorization

  grant <privilege list>

  on <relation name or view name>

  to <user/role list>

- <user list> is:
  - a user-id
  - **public**, which allows all valid users the privilege granted
- Granting a privilege on a view does not imply granting any privileges on the underlying relations.
- The grantor of the privilege must already hold the privilege on the specified item (or be the database administrator).

Privileges in SQL

- **select:** allows read access to relation,or the ability to query using the view
  - Example: grant users U_1, U_2, and U_3 **select** authorization on the *instructor* relation:

    grant select on *instructor* **to** U_1, U_2, U_3

- **update**:

    grant update on *instructor* **to** *U*1, *U*2, *U*3

Privileges in SQL

- The authorization may be given either on all attributes of the relation or on only some, but not on specific tuples.
- If the list of attributes is omitted, the privilege will be granted on all attributes of the relation.

grant update on *instructor* **to** *U*1, *U*2, *U*3

grant update *(name)* **on** *instructor* **to** *U*1, *U*2, *U*3

Revoking Authorization in SQL

- The **revoke** statement is used to revoke authorization.

  revoke <privilege list>

  on <relation name or view name>

  from <user/role list>

- Example:

  revoke select on *department* **from** U_1, U_2, U_3

  revoke update *(budget)* **on** *department* **from** U_1, U_2, U_3

Roles

- Authorizations can be granted to roles, in exactly the same fashion as they are granted to individual users.
- Each database user is granted a set of roles that he/she is authorized to perform.

create role *lecturer*;

grant *lecturer* **to** U_1**;**

grant select on *takes* **to** *lecturer*;

Roles

- Roles can be granted to users, as well as to other roles

  create role *teaching_assistant*

  grant *teaching_assistant* **to** *lecturer*;

  - *lecturer* inherits all privileges of *teaching_assistant*

- Chain of roles

  create role *dean*;

  grant *instructor* **to** *dean*;

  grant *dean* **to** U_2;

- When a user logs in to the database system, the actions executed by the user during that session have
  - all the privileges granted directly to the user
  - all privileges granted to roles that are granted (directly or indirectly via other roles) to that user

Authorization on Views

- Authorization on view gives us the possibility to define authorization with respect to some specific tuples

create view *geo_instructor* **as**
(**select** *
from *instructor*
where *dept_name* = ’Geology’);

grant select on *geo_instructor* **to** *geo_staff*

- Then a *geo_staff* member can issue

  select *

  from *geo_instructor*;

Other Authorization Features

- **references** privilege to create foreign key
  - **grant reference** (*dept_name*) **on** *department* **to** U_1;
- transfer of privileges
  - **grant select on** *department* **to** U_1 **with grant option**;
  - **revoke select on** *department* **from** U_1, U_2 **cascade**;
  - **revoke select on** *department* **from** U_1, U_2 **restrict**;

Transfer of privileges

$$\begin{tikzcd}
 & U_1 \arrow[r] \arrow[dr] & U_4 \\
\text{DBA} \arrow[ur] \arrow[r] \arrow[dr] & U_2 \arrow[r] & U_5 \\
 & U_3 &
\end{tikzcd}$$

The picture can't be displayed.

Questions?