Authorization Types

Forms of authorization on parts of the database:

- **Read** - allows reading, but not modification of data.
- **Insert** - allows insertion of new data, but not modification of existing data.
- **Update** - allows modification, but not deletion of data.
- **Delete** - allows deletion of data.

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 list>

- <user list> is:
  - a user-id
  - **public**, which allows all valid users the privilege granted
  - A role (more on this later)
- 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
- **insert**: the ability to insert tuples
- **update**: the ability to update using the SQL update statement
- **delete**: the ability to delete tuples.
- **all privileges**: used as a short form for all the allowable privileges

Revoking Authorization in SQL

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

  revoke <privilege list>

  on <relation name or view name> **from** <user list>

- Example:

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

- <privilege-list> may be **all** to revoke all privileges the revokee may hold.
- If <revokee-list> includes **public,** all users lose the privilege except those granted it explicitly.
- If the same privilege was granted twice to the same user by different grantees, the user may retain the privilege after the revocation by one grantee.
- All privileges that depend on the privilege being revoked are also revoked.

Roles

- **create role** instructor;
- **grant** *instructor* **to Amit;**
- Privileges can be granted to roles:
  - **grant select on** *takes* **to** *instructor*;
- Roles can be granted to users, as well as to other roles
  - **create role** *teaching_assistant*
  - **grant** *teaching_assistant* **to** *instructor*;
    - *Instructor* inherits all privileges of *teaching_assistant*
- Chain of roles
  - **create role** *dean*;
  - **grant** *instructor* **to** *dean*;
  - **grant** *dean* **to** Satoshi;

Authorization on Views

- **create view** *geo_instructor* **as**
  (**select** *
  from *instructor*
  where *dept_name* = ’ Geology’);
- **grant select on** *geo_instructor* **to** *geo_staff*
- Suppose that a *geo_staff* member issues
  - **select** *
    from *geo_instructor*;
- What if
  - *geo_staff* does not have permissions on *instructor?*
  - creator of view did not have some permissions on *instructor?*

Other Authorization Features

- **references** privilege to create foreign key
  - **grant reference** (*dept_name*) **on** *department* **to** Mariano;
  - why is this required?
- transfer of privileges
  - **grant select on** *department* **to** Amit **with grant option**;
  - **revoke select on** *department* **from** Amit, Satoshi **cascade**;
  - **revoke select on** *department* **from** Amit, Satoshi **restrict**;
- Etc. read Section 4.6 for more details we have omitted here.

More Advanced SQL (from chp 5)

- Accessing SQL From a Programming Language
  - Dynamic SQL
    - JDBC and ODBC
  - Embedded SQL
- SQL Data Types and Schemas
- Functions and Procedural Constructs
- Triggers
- Advanced Aggregation Features
- OLAP

JDBC and ODBC

- API (application-program interface) for a program to interact with a database server
- Application makes calls to
  - Connect with the database server
  - Send SQL commands to the database server
  - Fetch tuples of result one-by-one into program variables
- ODBC (Open Database Connectivity) works with C, C++, C#, and Visual Basic
- JDBC (Java Database Connectivity) works with Java
- HDBC-ODBC and others

JDBC

- **JDBC** is a Java API for communicating with database systems supporting SQL.
- JDBC supports a variety of features for querying and updating data, and for retrieving query results. Also for accessing metadata.
- JDBC also supports metadata retrieval, such as querying about relations present in the database and the names and types of relation attributes.
- Model for communicating with the database:
  - Open a connection
  - Create a "statement" object
  - Execute queries using the Statement object to send queries and fetch results
  - Exception mechanism to handle errors

ODBC

- Open DataBase Connectivity(ODBC) standard
  - standard for application program to communicate with a database server.
  - application program interface (API) to
    - open a connection with a database,
    - send queries and updates,
    - get back results.
- Applications such as GUI, spreadsheets, etc. can use ODBC
- Was defined originally for Basic and C, versions available for many languages.

Embedded SQL

- The SQL standard defines embeddings of SQL in a variety of programming languages such as C, Java, and Cobol.
- A language to which SQL queries are embedded is referred to as a **host language**, and the SQL structures permitted in the host language comprise *embedded* SQL.
- The basic form of these languages follows that of the System R embedding of SQL into PL/I.
- **EXEC SQL** statement is used to identify embedded SQL request to the preprocessor

  EXEC SQL <embedded SQL statement > END_EXEC

  Note: this varies by language (for example, the Java embedding uses # SQL { …. };)

Example Query

- From within a host language, find the ID and name of students who have completed more than the number of credits stored in variable credit_amount.
- Specify the query in SQL and declare a *cursor* for it

```
EXEC SQL
  declare c cursor for
  select ID, name
  from student
  where tot_cred > :credit_amount
END_EXEC
```

Embedded SQL (Cont.)

- The **open** statement causes the query to be evaluated

  EXEC SQL **open** *c* END_EXEC

- The **fetch** statement causes the values of one tuple in the query result to be placed on host language variables.

  EXEC SQL **fetch** *c* **into** :*si, :sn* END_EXEC

  Repeated calls to **fetch** get successive tuples in the query result

- A variable called SQLSTATE in the SQL communication area (SQLCA) gets set to '02000' to indicate no more data is available

- The **close** statement causes the database system to delete the temporary relation that holds the result of the query.

  EXEC SQL **close** *c* END_EXEC

  Note: above details vary with language. For example, the Java embedding defines Java iterators to step through result tuples.

Updates Through Cursors

- Can update tuples fetched by cursor by declaring that the cursor is for update

```
declare c cursor for
  select *
  from instructor
  where dept_name = 'Music'
for update
```

- To update tuple at the current location of cursor *c*

```
update instructor
set salary = salary + 100
where current of c
```

Procedural Constructs in SQL

Procedural Extensions and Stored Procedures

- SQL provides a **module** language
  - Permits definition of procedures in SQL, with if-then-else statements, for and while loops, etc.
- Stored Procedures
  - Can store procedures in the database
  - then execute them using the **call** statement
  - permit external applications to operate on the database without knowing about internal details
- Object-oriented aspects of these features are covered in Chapter 22 (Object Based Databases)

Triggers

Triggers

- A **trigger** is a statement that is executed automatically by the system as a side effect of a modification to the database.
- To design a trigger mechanism, we must:
  - Specify the conditions under which the trigger is to be executed.
  - Specify the actions to be taken when the trigger executes.
- Triggers introduced to SQL standard in SQL:1999, but supported even earlier using non-standard syntax by most databases.
  - Syntax illustrated here may not work exactly on your database system; check the system manuals

When Not To Use Triggers

- Risk of unintended execution of triggers, for example, when
  - loading data from a backup copy
  - replicating updates at a remote site
  - Trigger execution can be disabled before such actions.
- Other risks with triggers:
  - Error leading to failure of critical transactions that set off the trigger
  - Cascading execution

Recursive Queries

The Power of Recursion

- Recursive queries are powerful and can deal with trees and graphs effectively.
- Recursive views are required to be **monotonic**. *Thus they cannot be defined using difference or set aggregates in recursive rules.*

OLAP**

Data Analysis and OLAP

- **Online Analytical Processing (OLAP)**
  - Interactive analysis of data, allowing data to be summarized and viewed in different ways in an online fashion (with negligible delay)

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*)
- 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