Chapter 20

Security and Integrity

Topics

- Designing security
  - Discretionary authorization
  - Mandatory authorization
  - Enterprise authorization

- Designing integrity
  - Null, default, domain and check constraints
  - Unique, primary and foreign keys
  - Triggers

© Pearson Education Limited 2005
Business rules

- **Business rules** – across all data (and all applications)
- From the software engineering perspective:
  - **security and integrity** are mechanisms to enforce business rules
  - **transactions and concurrency** are means of implementing application logic
- **Security** has a personal (individual) dimension
  - Authorization privileges and security clearances are assigned to individual users and roles that they perform
- **Integrity** has a multi-personal (user community) dimension

Security – objectives and tools

- Objectives for designing security in enterprise information systems:
  - **Theft and fraud** – to disallow opportunities to access data or programs for illegal aims.
  - **Secrecy** – to disallow revealing of information to unauthorized users and applications.
  - **Integrity** – to disallow modification of data by unauthorized users and applications.
  - **Availability** – to allow access at all times to authorized users and applications.
- To achieve these objectives:
  - **Authentication** – verifying identity of a user/application (user id and password combination).
  - Granting privileges to access database objects (discretionary authorization).
Security - overview

- Users/applications can be granted privileges on:
  - schema objects
  - data objects
  - SQL operations
  - execution of procedures/functions

- Discretionary authorization may not be sufficient:
  - Mandatory authorization - users/applications are assigned clearance levels and database objects are assigned corresponding security levels
  - Enterprise authorization – access rights of all users and applications are stored in an enterprise authorization database

Discretionary authorization

- Centered on the notion of a privilege
  - All access rights to objects must be explicitly granted as privileges
    - grant
    - revoke
    - deny (not a standard SQL)

- Privileges can be granted/revoked to users and/or roles
  - A role is normally related to a job function (e.g. the role of a teacher).
  - The user's privileges are determined by the union of privileges granted to all roles that the user is assigned to.
Granting privileges to users and roles

create user anne identified by annepsswd;
create user michael identified by michaelpsswd;
create role student;
create role teacher;
grant create session to anne;
grant create session to michael;
grant select on grades to student;
grant update on grades to teacher;
grant student to anne;
grant student to teacher;
grant teacher to michael;

System and object privileges

- **System privilege**
  - right to perform a particular action on a database system as the whole or an action with regard to the database schema (logical or physical structure), e.g.
    - create session
    - create, alter and drop tables

- **Object privilege**
  - how users/applications can work with the database content, e.g.
    - select, insert, update and delete on tables/views
    - execute – the right to execute a procedure/function.

- **Application users** require only object privileges (and a system privilege to connect to and log in to a database).

- **Privileges can be passed from a user/role to another user/role**
  - SQL with grant option (for users only), e.g.
    - grant delete on grades to diana with grant option
### Passing privileges – authorization graph

- **DBA**
  - (DBA, student, select on grades)
  - (DBA, teacher, update on grades)
  - (DBA, diana, delete on grades with grant)
- **teacher**
  - (teacher, michael, role)
- **student**
  - (student, anne, role)
  - (student, teacher, role)
- **diana**
  - (diana, teacher, delete on grades)

### Effective privileges

**Mary**
- Access Object X

[ Mary does not have direct user privilege on X ]

Obtain union of privileges in Mary's roles

Does Mary have privilege on X?
- Yes
- No

Is Mary denied privilege on X in any role that Mary belongs to?
- Yes
- No

Revoke the deny

Mary can access X
**Programmatic discretionary authorization**

- **Views**
  - A SQL query (select statement) that is given a name and it can be called by its name

- **Synonyms**
  - An alias (alternative name) for a table, view, stored procedure, function, or another synonym

  ```sql
  create synonym actor for MovieActor.actor;
  ```

- **Stored procedures**
  - Precompiled batches of code, which have names and can take parameters

---

**Views**

<table>
<thead>
<tr>
<th></th>
<th>actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>actor_code</td>
<td>NUMBER(5) &lt;pk&gt; not null</td>
</tr>
<tr>
<td>actor_name</td>
<td>VARCHAR2(25) not null</td>
</tr>
<tr>
<td>address</td>
<td>VARCHAR2(100) null</td>
</tr>
<tr>
<td>demanded_pay_per_movie</td>
<td>INTEGER null</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>actor_no_personal_info</th>
</tr>
</thead>
<tbody>
<tr>
<td>actor_code</td>
<td>&quot;actor&quot;.&quot;actor_code&quot; NUMBER(5)</td>
</tr>
<tr>
<td>actor_name</td>
<td>&quot;actor&quot;.&quot;actor_name&quot; VARCHAR2(25)</td>
</tr>
</tbody>
</table>

- Restrict access to a predetermined set of rows and/or columns of a table
- Most interesting views are not updatable
- Creating a view does not require a resource privilege
  - To create a view, a user must only have prior select privileges on all tables referred to by the view

```sql
create or replace view "actor_no_personal_info" as
select "actor"."actor_code", "actor"."actor_name"
from "actor"
```
### Synonyms

- Some database management systems unite the concepts of user and schema, i.e. a schema is created for each user account → security problem!
- Solutions:
  - Allow all users to share a single schema and, therefore, the same username and password → does not solve anything
  - Still allow all users to share a single schema but store usernames and passwords in the database (governed by the same schema) to authenticate the users of the system.
  - Use multiple local schemas (one schema per user), synonyms, and one global schema.
    - each user has his/her own schema but they do not contain any tables or data; they are only filled with synonyms
    - synonyms connect each schema to a central (global) schema
    - all requests performed on the user schemas are redirected to the global schema through the synonyms

### Stored procedures and functions

- They completely encapsulate actions performed on the database
- They can detect the access level of the connected client and can service only permitted activities
- The only authorization privilege assignable to a stored procedure or function is the execute privilege
  - if application accesses all data by means of stored procedures and functions, a database administrator can revoke from all application users and roles the privileges to select, insert, update and delete data
Mandatory authorization

- defines top-down clearance levels for users and security classes for accessed objects in order to eliminate some possible bypasses of security
  - useful in systems where secrecy is a great concern
- security classes and clearance go hand-in-hand; a user has to have the clearance at least as high as the object’s classification to gain access to it
- the scheme is called mandatory to emphasize that classifications and clearances are maintained centrally by security administrators and cannot be changed by users
- challenge-response approach involving encryption:
  - user/application tries to connect to database,
  - database sends a challenge string to user/application,
  - user/application uses a predefined password (encryption key) to encrypt the challenge string and sends encrypted string back to the database,
  - database decrypts the string with the same password and it allows connection only if it gets the initial challenge string.

Bell-LaPadula model

- Variant of mandatory authorization centered on four concepts:
  - objects (e.g. tables, views, columns, rows, stored procedures),
  - subjects (e.g. users, application programs),
  - classifications of objects,
  - clearances of subjects.
- Four classification/clearance levels:
  - top secret
  - secret
  - confidential
  - unclassified
- Once objects are given classification levels and subjects are assigned clearance levels, a subject with certain clearance can only access objects with the same or lower classification.
Enterprise authorization

- when faced with enterprise information systems with possibly hundreds or applications, thousands of users, and several corporate databases
- enterprise-wide authorization policies, that cannot be changed by individual users or even developers, must be adopted
- privileges on database objects and on application objects should be consistent and in sync
  - the authorization system must be able to determine privileges that apply to a user when that user is using a particular application program
  - the need to distinguish between an application role and a user role

Application vs user role

- In SQL Server:
  - application roles have no members
  - application roles are inactive by default (they are activated by the application when it starts)
  - all effective privileges of the current user are removed and only privileges assigned to the application role are applied
- In Oracle:
  - so-called advanced security features implement the notion of an enterprise user (or schema-independent user) → application role
  - special directory service to map a user (so called Distinguished Name (DN) of a user) to an enterprise user, when the user accesses the database via an application
  - the application is associated with so-called shared schema and a set of privileges for the application on the shared schema is defined
  - the privileges of an enterprise user is the union of the privileges associated with the shared schema and any additional privileges that are associated with the roles granted to the user in the directory service
**Authorization database**

- RoleToClientPermission
  - role_id
  - client_object_id
  - client_permission_id

- RoleToServerPermission
  - role_id
  - server_object_id
  - server_permission_id

- ApplicationRole
  - app_role_id
  - parent_app_role_id
  - app_role_name
  - app_login_id
  - app_password

- User
  - user_id
  - user_name
  - server_login_id
  - server_password

- UserRole
  - user_role_id
  - parent_user_role_id
  - user_role_name

- AppRoleToUserRole
  - user_role_id
  - app_role_id

- UserToAppRole
  - user_id
  - role_id

- UserToUserRole
  - user_id
  - user_role_id

**Designing integrity**

- Types of integrity constraints:
  - null
  - default
  - domain (data type)
  - check (rule, assertion)
  - unique (alternate key)
  - primary key
  - foreign key
  - trigger (active rule, assertion)
### Null and default constraints

```sql
create table "listed_as"  
("actor_code"         NUMBER(5)                        not null,  
"movie_code"         NUMBER(5)                        not null,  
"position"           NUMBER(2)                        default 1,  
constraint PK_LISTED_AS primary key ("actor_code", "movie_code")
)
```

### Domain and check constraints

```sql
create domain feature_type varchar2(20)  
constraint CKC_FEATURE_TYPE  
check (value in ("drama","comedy","horror"))
```
### Unique and primary keys

**Table: actor**

- `actor_code`: NUMBER(5) not null<br>
- `actor_name`: VARCHAR2(25) not null<br>
- `address`: VARCHAR2(100)<br>
- `demanded_pay_per_movie`: INTEGER default 50000<br>

**Constraints:**
- Primary key: `actor_code`
- Unique: `actor_name`, `address`

```
create table "actor"  
  "actor_code"              NUMBER(5)                    not null,  
  "actor_name"              VARCHAR2(25)                 not null,  
  "address"                 VARCHAR2(100),  
  "demanded_pay_per_movie"  INTEGER                      default 50000,  
  constraint PK_ACTOR primary key ("actor_code"),  
  constraint CK1_ACTOR unique ("actor_name", "address")
);
```

### Foreign keys

**Table: listed_as**

- `actor_code`: NUMBER(5) not null<br>
- `movie_code`: NUMBER(5) not null<br>
- `position`: NUMBER(2) null

**Constraints:**
- Primary key: `actor_code`, `movie_code`, `position`
- Foreign key:
  - Reference: `actor_code`
  - Action: on delete cascade

```
alter table "listed_as"  
  add constraint FK_LISTED_A_REFERENCE_ACTOR  
  foreign key ("actor_code")  
  references "actor" ("actor_code")  
  on delete cascade;
```
Triggers

-- After delete trigger "movie"
create trigger "tda_movie" after delete
on "movie" for each row
declare
    integrity_error exception;
    errno integer;
    errmsg char(200);
    dummy integer;
    found boolean;
begin
    -- Delete all children in "listed_as"
delete "listed_as"
where "movie_code" = :old."movie_code";
    -- Errors handling
    exception
        when integrity_error then ...

Summary

- **Security** has a personal (individual) dimension. **Integrity** has a multi-personal (user community) dimension.
- **Authentication** is the task of verifying identity of a user/application trying to connect to a database.
- **Discretionary authorization** is based on granting privileges to access database objects.
- **Programmatic authorization** uses views, synonyms, and stored procedures.
- In **mandatory authorization** users/applications are assigned clearance levels and database objects are assigned corresponding security levels.
- **Enterprise authorization** demands that access rights of all users and applications are stored in an enterprise authorization database, which every application must consult.
- **Integrity** is about correctness and consistency of data accessible to the program.
- **Triggers** constitute a procedural way of enforcing referential integrity constraints as well as implementing other business rules that the database must satisfy.