Datenbanken II B: DBMS-Implementierung

Chapter 3: Database Security: Access Rights

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2021/22

http://www.informatik.uni-halle.de/~brass/dbi21/

Objectives

After completing this chapter, you should be able to:

- explain access rights in SQL and use the `GRANT` command.

  This was part of your basic database course and is only very quickly repeated.

- explain the purpose of system privileges in Oracle.

  As opposed to the standard "object privileges".

- get information about access rights from the system catalog.
- create users in Oracle.
- start and stop the Oracle server,
  enumerate different system states of the server.

Contents

Data Security: Motivation

- Information is a valuable asset: E.g. customer data must be kept safe from competitors.
- There are laws regulating the privacy of certain information: One might be sued for allowing private information (e.g. medical records, credit card numbers) to become public.
- The confidentiality of certain information (e.g. salaries) must be protected within a company.
- If an intruder deletes all data, a company will be out of business immediately.
- Unauthorized changes/falsification of data must be prohibited, e.g. employees should not be able to change their own salary.

Certain business rules — who may do what — must be enforced.

Security Requirements (1)

It should be possible to

- ensure that only legitimate users have access to the database (user identification/authentication).
- define which user can perform which operations on which database objects (authorization).
- log the actions of the users (auditing).
- ensure the privacy and integrity of the data also in a networked (client-server) environment.

Security Requirements (2)

It should be possible to

- limit the use of resources (disk space, CPU time) for specific users (quotas).

  Or else a single user can bring the complete database to a standstill.

- manage groups of users with the same rights.
- ensure that nobody has direct access to the data (circumventing the database access control).
- separate responsibilities of different administrators.

User Authentication (1)

- This is normally done with secret passwords.
- Some databases perform their own user identification, some rely on the operating system.

  Both have advantages and disadvantages: It is convenient not to have to enter passwords twice, and it is also good if application programs do not have to contain passwords. However, in a client-server environment, it might be not so clear that the user on the client is really the person he/she claims to be (and that the client is the computer it claims to be). Some client operating systems might also have a weaker security model than the server (think e.g. of a PC where anybody can enter a boot disk/CD).

- Choose a password which is not easy to guess.

  Even if direct decryption is not possible, a fast encryption routine can check many words for a match. If a hacker gets the encrypted password, he can try all words in a dictionary, names of persons/pop groups, all these words reversed, and all short words consisting only of a–z and 0–9.

User Authentication (2)

- It is possible to force users to change passwords periodically, e.g. every month.

  This might result in weaker passwords than if the user has time to think about it. Systems forcing the user to change his/her password generally also require that the old and the new password differ significantly (e.g. it is not enough to change only one letter), and that the user cannot switch back to the old password too soon.

- If a system discovers several unsuccessful login attempts, it should ring an alarm bell and possibly lock the account.

  In addition, there is often an artificial delay before the system tells the user that the password was incorrect. In this way a hacker program cannot try many words at full speed.

User Authentication (3)

- Many DBMS have certain default passwords for the DBA. Set new passwords immediately!

  E.g. in Oracle, `SYSTEM` has the password `MANAGER`, and the password for `SYS` is `CHANGE_ON_INSTALL`. A hacker will know this. In SQL Server, the most powerful account `sa` (system administrator) has by default no password.

- Remove guest accounts.

  Everybody knows that Oracle has an account `SCOTT` with password `TIGER`. SQL Server also has a `guest` account, to which Windows users otherwise unknown to the database are mapped. Check the accounts in the system periodically and make sure that each is really needed.

- Since most DBMS are client-server systems, the database server is automatically "on the net" (like a web server).

  So even if the hacker cannot get an operating system account on the server machine, he/she might be able to connect to the DBMS server. Oracle normally waits on port 1521 for connections, and the default SID is ORCL.

User Authentication (4)

- If a password is sent without encryption over the internet, other people may be able to see it.
  - It is possible to listen to all packages sent over the local Ethernet.

    Ethernet packages are broadcast to all connected computers, but normally a low software layer in the operating system ignores packages for other computers.
  - The route a package takes over the global internet is not predictable.

    Some gateway might be operated by a bad guy or might already be conquered by a hacker.

User Authentication (5)

- When entering the password, be sure that you are connected with the right program/computer.

  E.g. there were earlier programs which looked like a UNIX login prompt, but instead sent the password to a hacker. Insufficient care with the search path for commands might result in a getting a hacker program when calling "`sqlplus`". On the internet, it is possible that computers "pretend" to be some other computer.

- Don't use the same password for several systems.

  Don't ever create accounts in web shops with the same password as your Oracle account. The shop personnel might be able to read it.

- Never tell a password to official sounding strangers on the phone.

Permissions/Privileges (1)

- Often different users of a DB have different rights.
- Commands can be understood as consisting of
  - a "subject" (the user who executes it),
  - a "verb" (the operation, e.g. "`INSERT`), and
  - an "object" (e.g. the "`COURSES`" table).
- DBMS allow defining which operations a user can apply to which objects (see `GRANT` below).

  So the DBMS stores a set of triples (u, o, d) stating that user u can perform operation o on database object d. Operations are basically `SELECT`, `INSERT`, `UPDATE`, `DELETE`, objects are e.g. tables.

Permissions/Privileges (2)

- However, there are at least two problems with the user-operation-object model:
  - Commands like "`CREATE TABLE`" do not refer to existing objects, but it could be beneficial to restrict their use.
  - Large companies might need different administrators with different rights (not a single "root" user who can do everything).
- Every major DBMS has some solution to these problems, but each solution is somewhat different.

Permissions/Privileges (3)

- Views and stored procedures provide a way to encapsulate database objects:
  - Although a user cannot `SELECT` from a table directly, he/she might be able to access certain columns/rows or summary data via a view.

    Also data changes can be limited via views, see below.
  - Although a user cannot do a direct `INSERT` into a table, he/she might use a procedure which performs such an insertion after additional tests.

Auditing

- It should be possible to log user actions.

  Also unsuccessful attempts to execute commands (because of insufficient privileges) should be logged.

- In this way, it might at least afterwards be possible to find who is responsible for a problem.

  Or from which account a hacker has broken into the system.
  Of course, the auditing information itself must be sufficiently secure so that the hacker cannot delete it.

- The auditing must be done very selectively, or it will be difficult to find something interesting in too much data (and a lot of file space is needed).

Data Security

- Nobody should have direct access to the data (bypassing the database).

  At least nobody who has not also the DBA rights.

- The data is often stored non-encrypted in operating system files. Access to these files permits access to the data even without a DB account.
- Backup tapes must also be locked away.
- In Germany a PC containing an AIDS register was stolen. In the US, several notebooks from the CIA went missing.

Contents

GRANT Command (1)

- In SQL, access rights on database objects (tables, views, etc.) can be given to other users by means of the `GRANT` command.
- The `GRANT` command was already contained in the SQL-86 standard. It has the form

  `GRANT ⟨Rights⟩ ON ⟨Object⟩ TO ⟨Users⟩`

- E.g. give read and insert rights ("privileges") on the table `COURSES` to the users `BRASS` and `SPRING`.

```
GRANT SELECT, INSERT ON COURSES TO BRASS, SPRING
```

- This will give the users `BRASS` and `SPRING` read access to the table `COURSES` and the possibility to append data (add new rows).

GRANT Command (2)

- The users `BRASS` and `SPRING` will not be able to delete or modify rows in the table (unless they had these rights before).
- It is possible to later `GRANT` them the `UPDATE` and `DELETE` rights, too.

  Then `SELECT` and `INSERT` do not have to be repeated.
- The DBMS probably stores in a system table the user-command-object triples.

GRANT Command (3)

Possible Rights ("Privileges") in SQL-92:

- `SELECT`: Read access (use of the table in queries).

  In SQL Server, `SELECT` rights can apply to specified columns. This is not part of the SQL-92 standard and not supported in Oracle and DB2. But views give the same effect.

- `INSERT`: Appending new data (insertion of rows).

- `INSERT`$(A_1, \ldots, A_n)$: Only values of the A_i may be specified, other columns will be filled with default values (declared in the `CREATE TABLE` command).

  Allowing `INSERT` only for specific columns is part of the SQL-92 standard, but only supported in Oracle (not in SQL Server and DB2). But views serve the same purpose.

GRANT Command (4)

SQL-92 Rights, continued:

- `UPDATE`: Changing column values of existing rows.
- `UPDATE`$(A_1, \ldots, A_n)$: Only data in the columns A_i may be changed.
- `DELETE`: Deleting data (rows from the table).
- `REFERENCES`: Creating integrity constraints which reference this table.

  Referencing someone else's table in a foreign key constraint and not giving him/her `DELETE` rights on the new table can in effect limit his/her `DELETE` rights on his/her own table. Also `REFERENCES` allows checking whether a key value is there or not.

GRANT Command (5)

SQL-92 Rights, continued:

- `REFERENCES` $(A_1, \ldots, A_n)$: Only the A_i may be referenced.

  This is interesting if the table has two or more keys. It is supported in Oracle and DB2 (not in SQL Server).

- In addition, SQL-92 has the right "`USAGE`" on domains, character sets, etc. (not supported in any of the three DBMS).

GRANT Command (6)

Non-Standard Rights for Tables:

- `ALTER`: Right to change the table definition.

  Supported in Oracle and DB2, not SQL Server.

- `INDEX`: Right to create an index on this table.

  Supported in Oracle and DB2, not SQL Server.

Non-Standard Rights for Procedures/Packages:

- `EXECUTE`: Right to execute the procedure etc.

  This is supported in all three DBMS.

- `BIND`: Reoptimize SQL statements in a package.

  This exists in DB2 only.

GRANT Command (7)

Non-Standard Rights for Schema Objects (DB2):

- `ALTERIN`: Alter any object in the schema.
- `CREATEIN`: Create objects in the schema.
- `DROPIN`: Drop any object in the schema.

Non-Standard Rights for Directory Objects (Oracle):

- `READ`: Read files in the directory.

GRANT Command (8)

GRANT ALL PRIVILEGES:

- Instead of listing single rights it is possible to say

  GRANT ALL PRIVILEGES ON ⟨Object⟩ TO ⟨Users⟩

- This means all rights which the user executing the GRANT has on the object.

TO PUBLIC:

- Instead of listing all users in the system, the following is possible (this includes future users):

  GRANT SELECT ON COURSES TO PUBLIC

GRANT Command (9)

`WITH GRANT OPTION:`

- A user can be given a right with the possibility to pass on the right to other users.
- To do this, add the clause "`WITH GRANT OPTION`" to the `GRANT`-command:

```
GRANT SELECT ON COURSES TO BRASS
WITH GRANT OPTION
```

- This allows the user `BRASS` also to pass the grant option to other users (who then can also pass the right to other users, etc.).

GRANT Command (10)

Owner of an Object:

- The owner of a database object (table etc.), i.e. the user who created the object, has all rights on it including the grant option for these rights.
- By default, only the owner has rights on an object. Other users can get rights only by explicit `GRANT`s.

  Users with system administrator rights might be able to access the object without being granted rights explicitly.

- The owner of an object can also drop the object again (which is not included in the other rights).

GRANT Command (11)

CONTROL-Right in DB2:

- This gives all rights on the object with grant option. It also gives the right to drop the object.

  There is no CONTROL-right in Oracle and SQL Server. But it is similar to being the owner of the object.

- By default, the object creator has the CONTROL right.

  For views, the creator gets the CONTROL right only if he/she holds it also for the underlying base tables (and used views).

- But the CONTROL-right is always given without GRANT OPTION. It can be granted only by an administrator.

  Granting CONTROL needs the SYSADM or DBADM authority.

Revoking Access Rights (1)

- Previously granted access rights can be revoked (taken back) with a command very similar to `GRANT`:

  `REVOKE` ⟨`Rights`⟩ `ON` ⟨`Object`⟩ `FROM` ⟨`Users`⟩

- ⟨`Rights`⟩: comma-separated list of single privileges or "`ALL PRIVILEGES`".
- ⟨`Object`⟩: name of database object (e.g. table, view).
- ⟨`Users`⟩: comma-separated list of user names or "`PUBLIC`".

Revoking Access Rights (2)

- Example:

```
REVOKE INSERT ON COURSES FROM BRASS
```

  If `BRASS` had `SELECT` and `INSERT` rights on `COURSES` before this command, he will have only the `SELECT` right afterwards.

- Users can only revoke rights which they have granted earlier.

  Therefore, in its internal tables, the DBMS stores not only the triple user-right-object, but the quadruple (A, P, O, B): User A has granted privilege P on object O to user B.

Revoking Access Rights (3)

- It is not possible to grant a right to `PUBLIC` and revoke it from a specific user.

  Since `PUBLIC` also refers to future users, the database stores that it was granted to `PUBLIC` and not simply the right for every existing user.

- It is possible to grant "`ALL PRIVILEGES`" and to revoke them later selectively.

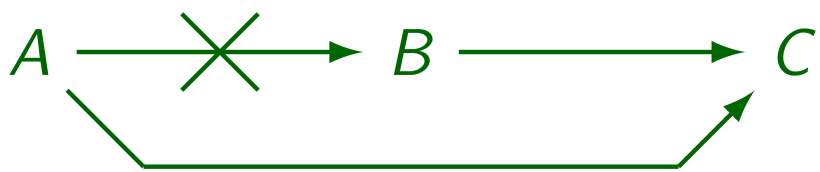
  "`ALL PRIVILEGES`" refers only to the rights the user currently has. They are stored one by one in the system table.

- When a table is deleted, all `GRANT`s for it are deleted, too. If it is later re-created, only the owner can access it.

Revoking Access Rights (4)

- If A granted a privilege "`WITH GRANT OPTION`" to B, and B granted it to C, and then A revokes the right from B, it will be recursively revoked from C.

- In SQL-92, this requires `CASCADE`, see below.
- However, if C got the right in addition on some other path (e.g. directly from A), he/she keeps it.



Revoking Access Rights (5)

- The `REVOKE` command restores the same situation as if B never had the right.
- But note that when B has used the right to change the tables, these changes are not magically undone.
- SQL-86 did not have a `REVOKE` command.
- In DB2, only the `CONTROL` right on a database object gives the possibility to `REVOKE` rights on it.

  It is a bit strange that the grant option allows users to grant the right, but not revoke it. In this way, DB2 does not have to keep track of the path a user got the right. If a user with `CONTROL` rights revokes it, it is gone (unless a group/public has the right).

Revoking Access Rights (6)

- In SQL-92 and SQL Server, `CASCADE` must be added if rights are to be revoked recursively, e.g.:

```
REVOKE INSERT ON Course FROM BRASS CASCADE
```

- In SQL Server, `CASCADE` is always required when revoking a right that was granted `WITH GRANT OPTION`.

  In SQL-92 this is only necessary if the user from which the right is revoked has used the grant option to grant the right to somebody else. Also, in SQL-92 `RESTRICT` can be specified instead of `CASCADE` to execute the `REVOKE` only if no other rights of other users depend on it. SQL Server does not understand `RESTRICT`.

- Oracle and DB2 do not support `CASCADE`/`RESTRICT`.

Revoking Access Rights (7)

- SQL-92 supports "`REVOKE GRANT OPTION FOR ...`".

  This is understood only in SQL Server, not in Oracle or DB2.

  SQL Server requires `CASCADE` is specified in addition.

- In Oracle, "`CASCADE CONSTRAINTS`" must be added to the `REVOKE` of the `REFERENCES` privilege if the privilege was used to create foreign keys. These foreign keys will then be dropped.

Contents

Object Privileges (1)

- Access rights in standard SQL are a set of triples:
  Who can execute which command on which table?

```
GRANT  SELECT ON EMP  TO   SMITH
REVOKE INSERT ON DEPT FROM MILLER
```

  Actually, they are quadruples: Who has given whom what right on which database object? This is important when rights are revoked.

- Rights can also be granted "`TO PUBLIC`".

  All users, including users created in future.

- Rights can be given "`WITH GRANT OPTION`".

  Then the grantee can grant the right to further users. The owner of a table (the user who created it) automatically holds all rights on it `WITH GRANT OPTION`. For views this is more complicated.

Object Privileges (2)

- USER_TAB_PRIVS: Grants on objects for which the current user is owner, grantor, or grantee.

USER_TAB_PRIVS					
GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE
PUBLIC	SCOTT	DEPT	SCOTT	SELECT	N
SMITH	SCOTT	EMP	SCOTT	SELECT	Y
MILLER	SCOTT	EMP	SMITH	SELECT	N
SMITH	SCOTT	EMP	SCOTT	INSERT	N

I.e. all users have read access to the table DEPT. SMITH got read access to EMP WITH GRANT OPTION, and has given the right to MILLER. In addition, SMITH can append rows to EMP.

In contrast to other data dictionary tables, the prefix USER here does not mean that only tables owned by the current user are listed.

Object Privileges (3)

- Columns of `USER_TAB_PRIVS`:
  - `OWNER` and `TABLE_NAME` identify the table.
  - `GRANTEE`: The user who got the privilege.
  - `GRANTOR`: The user who gave the privilege.

    Because of the grant option, not only the owner can be grantor.
  - `PRIVILEGE`: The right, e.g. `'SELECT'`.
  - `GRANTABLE`: `'YES'` if right includes grant option.
- In this way, the SQL `GRANT` commands are stored in the data dictionary.

Object Privileges (4)

- USER_TAB_PRIVS_MADE is the subset of USER_TAB_PRIVS with OWNER=USER.
- USER_TAB_PRIVS_RECD is the subset of USER_TAB_PRIVS with GRANTEE=USER.
- The user might also have access to database objects because of grants to PUBLIC, which are not listed in these tables (but see ALL_TAB_PRIVS).

  Unless, of course, they are made by the current user or refer to tables of the current user. Otherwise, the name of the current user is neither OWNER, nor GRANTOR, nor GRANTEE, therefore the grant is not shown.

Object Privileges (5)

- The `INSERT` and `UPDATE` right can be given selectively for certain columns.

  An insert right for only part of the columns means that the other columns cannot be explicitly specified, and thus get their declared default value (or null).

- `USER_COL_PRIVS`: Grants that refer to single columns.

  `USER_COL_PRIVS` looks like `USER_TAB_PRIVS`, but has the additional column `COLUMN_NAME`.

- Grants for whole tables are not repeated here.

- `USER_COL_PRIVS_MADE`, `USER_COL_PRIVS_RECD`: Subsets with current user as owner/grantee (as above).

Contents

System Privileges (1)

- Commands like "`CREATE TABLE`" cannot be restricted with this standard security model.

  A user who is only supposed to enter data does not need to create new tables. For a secure system, every user should only be able to execute the commands he/she is supposed to execute.

- Therefore, Oracle has also "system privileges".

  Every major DBMS vendor has a to the problem (all different).

- In contrast to "object privileges", these refer to the execution of specific commands, not to DB objects.

- E.g. one needs the system privilege "`CREATE TABLE`" in order to be able to execute this command.

System Privileges (2)

- In order to log into Oracle, one needs the system privilege "`CREATE SESSION`".

  An account can be locked by not granting (or revoking) this privilege. It is still possible to access tables, views, etc. under this account via synonyms or "⟨`User`⟩.⟨`Table`⟩" (if one has the necessary access rights).

- Many system privileges are only for DBAs, e.g.:
  - "`SELECT ANY TABLE`" (read access to all tables),
  - "`DROP ANY TABLE`" (delete data of arbitrary users),
  - "`CREATE USER`" (create a new user).

System Privileges (3)

- Since the usual privileges of a DBA are separated into different system privileges, it is possible to have several DBAs with different responsibilities.

  Of course, one can still have one DBA with all privileges.

- There are currently 157 different system privileges.

  Basically, every administration command corresponds to a system privilege. Different kinds of `CREATE` commands also correspond to system privileges (since these commands could not be restricted otherwise). Most commands also have an `ANY`-version as a system privilege (allows one to apply the command to objects of any user). `CREATE ANY TABLE`: create tables in any schema.

System Privileges (4)

- If a user has a system privilege "`WITH ADMIN OPTION`", he/she can give it to other users:

```
GRANT CREATE TABLE TO SCOTT
```

  Adding "`WITH ADMIN OPTION`" gives `SCOTT` the right to grant "`CREATE TABLE`", too.

- When a system privilege is revoked from a user A who had it "`WITH ADMIN OPTION`", privileges are not recursively revoked from users B who got it from A.

  This might be the reason why it was not called "`GRANT OPTION`". But it is very similar ("`GRANT OPTION`" can be used only for object privileges).

System Privileges (5)

- SYSTEM_PRIVILEGE_MAP: List of all system privileges.

SYSTEM_PRIVILEGE_MAP	
PRIVILEGE	NAME
⋮	⋮
-5	CREATE SESSION
⋮	⋮
-40	CREATE TABLE
⋮	⋮
-47	SELECT ANY TABLE
⋮	⋮

System Privileges (6)

- USER_SYS_PRIVS: System privileges granted to the current user or to PUBLIC.

  Columns are: USERNAME (always the name of the current user, not very useful), PRIVILEGE (name of the system privilege, no join with SYSTEM_PRIVILEGE_MAP necessary), ADMIN_OPTION (similar to grant option for object privileges).

- DBA_SYS_PRIVS: System privileges for each user.

  For DBA only. It has the columns GRANTEE, PRIVILEGE, ADMIN_OPTION.

- Only directly granted privileges are listed.

  Additional system privileges might have been granted via roles (see below). Therefore, USER_SYS_PRIVS is often empty, although the user actually has many system privileges.

Contents

Roles (1)

- It is difficult to grant privileges to many users one by one. In one way or another, all modern DBMS support groups of users with similar privileges.
- Oracle has the concept of "roles", which are sets of privileges that can be granted to users:

```
CREATE ROLE MANAGEMENT
```

This command requires DBA rights (system privilege "`CREATE ROLE`").

- Access rights are granted to a role in the same way as they are granted to a user:

```
GRANT SELECT ON EMP TO MANAGEMENT
```

Roles (2)

- Roles can be granted to users (by their owner or users who got them `WITH ADMIN OPTION`):

  ```
  GRANT MANAGEMENT TO JIM, MARY
  ```

- When a user A is granted a role R, A receives all privileges that were or will be granted to R.

  But if "`MANAGEMENT`" is not one of the default roles of these users, which are automatically activated when they log in, they must explicity execute "`SET ROLE MANAGEMENT`" in every session in which they want to use these privileges. (This is not enforced in Oracle 8.0.) Roles can be protected by passwords. Then `SET ROLE` requires a password.

- If role A is granted to role B, B includes all rights of A. Thus, B is more powerful than A.

Roles (3)

- Several roles are predefined in Oracle 8, e.g.
  - `CONNECT`: Basic usage rights.

    This corresponds to the system privileges: `CREATE SESSION`, `ALTER SESSION`, `CREATE DATABASE LINK`, `CREATE SYNONYM`, `CREATE TABLE`, `CREATE CLUSTER`, `CREATE VIEW`, `CREATE SEQUENCE`.
  - `RESOURCE`: Rights for advanced users.

    This includes e.g. `CREATE TABLE`, `CREATE PROCEDURE`, `CREATE TRIGGER`. Students in this course were granted `CONNECT` and `RESOURCE` (but `UNLIMITED TABLESPACE` was revoked).
  - `DBA`: Right to do everything.
- In older Oracle versions, users were classified into these three types.

Roles (4)

- DBA_ROLES: List of all roles defined in the system.

  It has the columns ROLE, PASSWORD_REQUIRED. Only the DBA can create roles, and only the DBA can see the list of all roles.

- USER_ROLE_PRIVS: Roles granted to the current user.

  Roles granted to PUBLIC are also listed: All users have the rights included in such roles. Columns are: USERNAME, GRANTED_ROLE, ADMIN_OPTION, DEFAULT_ROLE, OS_GRANTED.

- DBA_ROLE_PRIVS: Which roles are granted to which user? Also role-to-role grants are shown.

  Columns: GRANTEE, GRANTED_ROLE, ADMIN_OPTION, DEFAULT_ROLE.

  GRANTEE can be a user or another role.

Roles (5)

- The following tables/views list the access rights included in roles accessible to the current user:
  - `ROLE_ROLE_PRIVS`: Roles implied by a role.

    Columns are: `ROLE`, `GRANTED_ROLE`, `ADMIN_OPTION`.

    All rights in `GRANTED_ROLE` are included in `ROLE`.
  - `ROLE_SYS_PRIVS`: System privileges in a role.

    Columns are: `ROLE`, `PRIVILEGE`, `ADMIN_OPTION`.
  - `ROLE_TAB_PRIVS`: Table privileges granted to roles.

    Columns are: `ROLE`, `OWNER`, `TABLE_NAME`, `COLUMN_NAME` (null if right for entire table), `PRIVILEGE`, `GRANTABLE`.

Contents

Creating Users (1)

User Authentication:

- Oracle can perform the user authentication itself. One must specify a user name and a password:

```
CREATE USER BRASS IDENTIFIED BY ABC_78
```

  Passwords have the same syntax as table names: They are not case-sensitive and `"..."` is needed to include special characters.

- Oracle can also rely on the authentication done by the operating system or a network service:

```
CREATE USER OPS$BRASS IDENTIFIED EXTERNALLY
```

  So when the UNIX user `BRASS` logs into Oracle (with empty username/password), he becomes the Oracle user `OPS$BRASS`.

Creating Users (2)

- A user created as explained above has no rights, not even the system privilege to connect to the DB.
- The necessary privileges can be given e.g. with:

```
GRANT CONNECT, RESOURCE TO BRASS
```

- After the `GRANT`, these roles can be made default roles, so that they are automatically activated when the user logs in:

```
ALTER USER BRASS DEFAULT ROLE ALL
```

It seems that roles without a password automatically become default roles (?). So this command might not be necessary.

Tablespaces and Quotas (1)

- A tablespace is a database file or a collection of DB files (storage space, container for tables).
- All tablespaces are listed in the system catalog table DBA_TABLESPACES.

  E.g. use "SELECT TABLESPACE_NAME FROM DBA_TABLESPACES" to list all tablespaces. This query must be executed by a DBA. All users have read access to USER_TABLESPACES (tablespaces that are accessible by the current user). The files for each tablespace are listed in DBA_DATA_FILES. It has e.g. the columns FILE_NAME, FILE_ID, TABLESPACE_NAME, BYTES. See also DBA_FREE_SPACE/USER_FREE_SPACE and DBA_FREE_SPACE_COALESCED.
- The tablespace "SYSTEM" contains e.g. the data dictionary (collection of system tables).

Tablespaces and Quotas (2)

- ```
  CREATE USER BRASS IDENTIFIED BY MY_PASSWORD
                DEFAULT TABLESPACE USER_DATA
                TEMPORARY TABLESPACE TEMPORARY_DATA
                QUOTA 2M ON USER_DATA
  ```

  The quota size can also be `UNLIMITED`.

- A tablespace can be defined when a table is created.

  Otherwise it is stored in the user's `DEFAULT TABLESPACE` (which is `SYSTEM` if it is not set in the `CREATE USER`).

- Without quota (and "`UNLIMITED TABLESPACE`"), the user cannot create tables on the tablespace.

  Use: `REVOKE UNLIMITED TABLESPACE FROM BRASS`
  ```

Data Dictionary: Users (1)

- ALL_USERS: List of all users, accessible by all users:
  - USERNAME: Name of the Oracle account.
  - USER_ID: Internal number of the account.
  - CREATED: Date/time when account was created.

ALL_USERS		
USERNAME	USER_ID	CREATED
SYS	0	29-JAN-98
SYSTEM	5	29-JAN-98
SCOTT	20	29-JAN-98
BRASS	24	13-MAY-01
⋮	⋮	⋮

Data Dictionary: Users (2)

- DBA_USERS: Full information about all users.
  Only the DBA can look at this table.

  It has the following columns: USERNAME, USER_ID, PASSWORD (stored in encrypted form), DEFAULT_TABLESPACE, TEMPORARY_TABLESPACE, CREATED, PROFILE, ACCOUNT_STATUS (indicates whether account is locked, expired, or unlocked), LOCK_DATE, EXPIRY_DATE, INITIAL_RSRC_CONSUMER_GROUP, EXTERNAL_NAME.

- USER_USERS: Single row with information about the current user.

  It has the following columns: USERNAME, USER_ID, ACCOUNT_STATUS, LOCK_DATE, EXPIRY_DATE, DEFAULT_TABLESPACE, CREATED, EXTERNAL_NAME.

Data Dictionary: Quotas

- DBA_TS_QUOTAS: How many bytes/blocks on which tablespace are charged to which user, and what is the allowable maximum (quota)?

  Columns of this table are: TABLESPACE_NAME, USERNAME, BYTES, MAX_BYTES, BLOCKS, MAX_BLOCKS. The columns BYTES and MAX_BYTES are derived from the information in blocks.

- USER_TS_QUOTAS: The current and maximal file space usage of the current user.

- All table data is charged to the table owner (even if other users actually inserted the rows).

Some Predefined Users (1)

- `SYS`: Owner of the system tables (data dictionary).

  Most powerful account. Default password: `CHANGE_ON_INSTALL`.

- `SYSTEM`: The default database administrator.

  For most administration tasks. Default password: `MANAGER`.

- `SCOTT`: Guest and demonstration account.

  Default password: `TIGER`. Sometimes there are additional accounts used in tutorials: `ADAMS`, `BLAKE`, `CLARK`, `JONES`.

- `OUTLN`: Schema contains information for optimizer.

  Default password: `OUTLN`.

Some Predefined Users (2)

- DBSNMP: Information for the "intelligent agent".

  It is used for remote administration via the "enterprise manager". Default password: DBSNMP.

- One should check the list of users in the system table ALL_USERS and lock all users that are currently not needed (or change their passwords).

  There is also a table DBA_USERS with more information. The list of users created during the installation can change with new versions. Also, when one installs additional software (e.g. the Oracle application manager), more accounts are created.

- Hackers know all the default passwords!

Changing and Deleting Users

- If a user has forgotten his/her password:

```
ALTER USER BRASS IDENTIFIED BY NEW_PASSWORD
```

- A user without tables can be deleted in this way:

```
DROP USER BRASS
```

- To delete the user including all his/her data, use:

```
DROP USER BRASS CASCADE
```

- The following command ensures that the user can no longer log in, but leaves his/her data untouched:

```
ALTER USER BRASS ACCOUNT LOCK
```

External Password File

- Whereas the above passwords are stored in the database (encrypted), there usually is an additional file that contains passwords of administrators who need e.g. to start up the database.

  When the database is not running, passwords stored in the database cannot be accessed. If you use `CONNECT INTERNAL` in the server manager (`svrmgrl`) or `CONNECT SYS AS SYSDBA`, the default password is `ORACLE`. Actually, the `SYS` password in the password file and in the database can be different. The password file is generated by the `orapwd` utility program. Later, every user granted `SYSDBA`/`SYSOPER` rights is also stored in the password file. Instead of using a password file, you can use OS authentication. This depends on the parameter `REMOTE_LOGIN_PASSWORDFILE`

Other Security Features (1)

- The resource usage of DB users can be restricted by creating a "profile" for them. This defines e.g.
  - How many concurrent sessions the user can have (number of windows with DB applications).
  - After what idle time he/she is logged off.
  - How much CPU time and how many logical reads (disk accesses) are allowed per session/per call.
  - After what time a password must be changed.
  - Which function is used to check the password complexity.

Other Security Features (2)

- Oracle also has an `AUDIT` command for defining which user actions are logged in system tables, so that one can later find out who did what.
  - E.g. all insertions should be logged that were executed (not refused):

```
AUDIT INSERT ON SCOTT.EMP
BY SESSION WHENEVER SUCCESSFUL;
```

    "`BY SESSION`" means that only one record is written for an entire session that did this operation (default). Alternative: "`BY ACCESS`".

  - E.g. log all unsuccessful login attempts:

```
AUDIT CONNECT WHENEVER NOT SUCCESSFUL;
```