Part 12: Security

References:

- Elmasri/Navathe:Fundamentals of Database Systems, 3rd Edition, 1999.
  Chap. 22, "Database Security and Authorization"
- Silberschatz/Korth/Sudarshan: Database System Concepts, 3rd Edition, McGraw-Hill, 1999.
  Section 19.1, "Security and Integrity"
- Kemper/Eickler: Datenbanksysteme (in German), Ch. 12, Oldenbourg, 1997.
- Lipeck: Skript zur Vorlesung Datenbanksysteme (in German), Univ. Hannover, 1996.
- Date/Darwen: A Guide to the SQL Standard, Fourth Edition, Addison-Wesley, 1997.
- van der Lans: SQL, Der ISO-Standard (in German, there is an English version), Hanser, 1990.
- Oracle8 SQL Reference, Oracle Corporation, 1997, Part No. A58225-01.
- Oracle8 Concepts, Release 8.0, Oracle Corporation, 1997, Part No. A58227-01.
- Don Chamberlin: A Complete Guide to DB2 Universal Database. Morgan Kaufmann, 1998.
- Microsoft SQL Server Books Online: Accessing and Changing Data, Administering SQL Server.

Objectives

After completing this chapter, you should be able to:

- Explain some possible attacks.
- Explain some DBMS features related to security.
- Use the `GRANT` and `REVOKE` commands of SQL.
- Enumerate some other security-related features of database management systems.

  In this text, Oracle, DB2, and SQL Server are discussed.

Overview

DB 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.

DB Security: Requirements

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.
- limit the use of resources (disk space, CPU time) for specific users (quotas).
- manage groups of users with the same rights.
- ensure that nobody has direct access to the data.
- 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).

User Authentication (2)

- It is potentially dangerous that the same password is used again and again.

  If an intruder somehow observes the password, he/she can also use it. Some banks send their customers lists of passwords each of which can be used only once.
- 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.

User Authentication (3)

- Never store a password for another computer (server) on a disk. (Also think about cookies containing passwords.)

  It has happened that a public domain utility program emailed the password to the hacker. Even if the password is somehow encrypted, the computer of the hacker can be made looking like the computer on which the utility was run.
- 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/she can try all words in a dictionary, names of persons/pop groups/etc., all these words also reversed, and all short words consisting only of lowercase letters or only of digits (phone numbers).

User Authentication (4)

- 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 (unless the hacker gets the encrypted password).

- 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.

User Authentication (5)

- Remove guest accounts.

  Everybody knows that Oracle has an account `SCOTT` with password `TIGER`. SQL Server also has a `guest` account, to which NT 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 (6)

- 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.

User Authentication (7)

- When entering the password, be sure that the connection is 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.

- It is not safe to use the same password for several systems.

  Only people who do not care about security create accounts in web shops with the same password as they use for Oracle. Nobody knows what happens with such passwords.

Permissions/Privileges (1)

- Often different users of the same 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 "`Course`" table).
- DBMS allow defining which operations a user can apply to which objects (see the `GRANT` command 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.
- The `GRANT` command was already part of the SQL-86 standard and is very portable.

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.

  E.g. Oracle has "system privileges" besides the standard "object privileges".

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 are passed.
- Even though a user might in principle be allowed to execute a command, there may be limits on disk space, CPU time, etc.

  Or a single user can bring the complete DB to a standstill.

Permissions/Privileges (4)

- Often there are groups of users with similar rights.

  Every major DBMS has some mechanism to simplify the administration of such groups.
- Normal DBMS implement "Discretionary Access Control":
  - "Discretionary Access Control" gives the administrators the ability to grant privileges to users at their discretion. (I.e. they can do whatever they think is right.)
  - "Mandatory Access Control" supports a classification of users and data, and allows a user to read data only at the same or lower security level as his/her clearance.

    "Trusted Oracle" offers both.

Auditing

- It should be possible to log user actions.
- 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 a lot of file space is required (and it will be difficult find something interesting in too much data).
- It might be useful to audit also unsuccessful attempts to execute commands (because of insufficient privileges).

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.

  Encryption would slow down the DBMS.
- Backup tapes must also be locked away.
- In Germany a PC containing an AIDS register was stolen.

Overview

1. Requirements
2. GRANT Command
3. Oracle
4. DB2
5. SQL Server

GRANT Command in SQL (1)

- Access rights on database objects (tables, views, etc.) can be given to other users by means of the `GRANT` command.
- It has the form

  `GRANT` ⟨`Rights`⟩ `ON` ⟨`Object`⟩ `TO` ⟨`Users`⟩
- E.g. give read and insert rights ("privileges") on the table `Course` to the users `sbrass` and `spring`.

```
GRANT SELECT, INSERT ON Course TO sbrass, spring
```

- This does not allow them to delete or modify tuples in the table `Course` (unless they had these rights before).

  It is possible to later `GRANT` them these rights, too.

  Then `SELECT` and `INSERT` do not have to be repeated.

GRANT Command in SQL (2)

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

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

  For default values, see the `CREATE TABLE` command above. 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 in SQL (3)

- `UPDATE`: Changing existing data (updating table entries).
- `UPDATE`($A_1, \ldots, A_n$): Only data in the given columns A_i may be changed.

  This is supported in all three systems.
- `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 in SQL (4)

Rights in SQL-92, 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).

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.

GRANT Command in SQL (5)

Non-Standard Rights for Procedures/Packages:

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

  This is supported in all three DBMS.
- `BIND` (DB2 only): Reoptimize SQL statements in a package.

Non-Standard Rights for Schema Objects (DB2 only):

- `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 only):

- `READ`: Read files in the directory.

GRANT Command in SQL (6)

`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:

  `GRANT SELECT ON Course TO PUBLIC`

- This includes future users.

GRANT Command in SQL (7)

`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 Course TO sbrass
WITH GRANT OPTION
```

- This also allows the user `sbrass` to pass the grant option to other users. So rights can be passed recursively.

GRANT Command in SQL (8)

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 in SQL (9)

`CONTROL`-Right in DB2:

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

- There is no `CONTROL`-right in Oracle and SQL Server.

  But it is similar to being the owner of the object.

Revoking Access Rights (1)

- Previously granted access rights can be revoked with a command very similar to the GRANT command:

  REVOKE ⟨Rights⟩ ON ⟨Object⟩ FROM ⟨Users⟩
- ⟨Rights⟩ can be a comma-separated list of single privileges or "ALL PRIVILEGES".
- ⟨Object⟩ is a database object (table, view, etc.).
- ⟨Users⟩ is a comma-separated list of DB users or "PUBLIC".
- E.g. REVOKE INSERT ON Course FROM sbrass.
- Users can only revoke rights which they have granted earlier.
- SQL-86 did not have a REVOKE command.

Revoking Access Rights (2)

- 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.

  However, if C got the right in addition on some other path (e.g. directly from A), he/she keeps it.
- This 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.
- If a table is deleted, all GRANTs for it are deleted, too. If it is re-created, only the owner can access it.

Revoking Access Rights (3)

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

  REVOKE INSERT ON Course FROM sbrass CASCADE
- In SQL Server, CASCADE is always required when revoking a right which 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.
- Neither Oracle nor DB2 understand CASCADE/RESTRICT.

Revoking Access Rights (4)

- SQL-92 allows "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 DB2, only the CONTROL right on a database object gives the possibility to REVOKE rights on it.

  So 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).
- 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.

Overview

1. Requirements
2. GRANT Command
3. Oracle
4. DB2
5. SQL Server

Accessing Other User's Tables (1)

- In Oracle, one must normally write "⟨User⟩.⟨Name⟩" to access database objects (tables, views, etc.) of other users.
- E.g.: `SELECT * FROM sbrass.President`
- In Oracle, the user name is a DB schema identifier, so a database-wide unique identification of the object always consists of these two components.
- The guest account "SCOTT" with password "TIGER" can be used to check other user's rights.
- If a user tries to access a table, but does not even have the SELECT right on it, Oracle gives the same error message as if the table did not exist.

Accessing Other User's Tables (2)

- Since it is a bit difficult to write two-part names, Oracle allows user-defined shorthands:

```
CREATE SYNONYM President FOR sbrass.President
```

- A user with the system privilege "CREATE PUBLIC SYNONYM" (usually a DBA) can also use

```
CREATE PUBLIC SYNONYM President
                 FOR sbrass.President
```

  Then the table will look as if it exists under every account.

  But there is only a single copy. Such public synonyms are used for the data dictionary tables.

- A shorthand can be deleted with: `DROP SYNONYM President`.

System Privileges in Oracle (1)

- Besides access rights to tables etc., which are called "object privileges", Oracle also has "system privileges".
- These refer to the execution of specific commands, not to database objects.
- E.g. one needs the system privilege "CREATE TABLE" in order to be able to execute this command.

  A user who is only supposed to enter data does not need to create new tables. For a secure system, every user should have only the privileges he/she needs.

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

  So there can be locked accounts.

System Privileges in Oracle (2)

- Many of these privileges are only for DBAs, e.g.:
  - “SELECT ANY TABLE” (look at tables of other users),
  - “DROP ANY TABLE” (delete data of arbitrary users),
  - “CREATE USER” (create a new user).
- This allows different DBAs to have different responsibilities.
- There are currently more than 90 different system privileges.

  Every administration command corresponds to a system privilege. `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 any object). `CREATE ANY TABLE`: create tables in any schema.

System Privileges in Oracle (3)

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

  ```
  GRANT CREATE TABLE TO scott
  ```

  If “WITH ADMIN OPTION” is added, `scott` can pass it on.
- If a system privilege is revoked from a user who had it “WITH ADMIN OPTION”, no privileges are recursively revoked.

  This difference to the “GRANT OPTION” might be the reason why a different name was chosen. However, it is a bit confusing that the grant option can be used only for object privileges and the admin option only for system privileges.

Roles in Oracle (1)

- It is difficult to grant privileges to many users one by one.
- Oracle has the concept of “roles”, which are sets of privileges:

  CREATE ROLE ⟨Name⟩
- Only those with the system privilege “CREATE ROLE” can execute this command (normally only the DBA).
- Access rights are granted to a role (e.g. “staff”) in the same way as they are granted to a user:

  ```
  GRANT SELECT ON Course TO staff;
  ```
- Granting access rights to a role is treated in the same way as granting it to specific users (DBA rights are not needed).

Roles in Oracle (2)

- Roles can be granted to users:

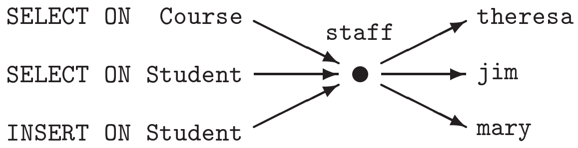
  ```
  GRANT staff TO jim, mary
  ```
- Then these users get all the privileges granted to the role.

  But if “staff” is not one of the default roles of these users, which are automatically activated when they log in, they must explicity execute “SET ROLE staff” in every session in which they want to use these privileges.
- Only the owner of the role or a user who has received it with admin option can grant the role to a user:

  ```
  GRANT staff TO theresa WITH ADMIN OPTION
  ```

Roles in Oracle (3)

- So roles are a level of indirection between the single privileges and users, intended to simplify the administration of a group of users with the same privileges:



- When further privileges are granted to "staff", these become automatically available to `theresa`, `jim`, and `mary`.

Roles in Oracle (4)

- A role can be granted to another role, e.g.

  `GRANT staff TO reg_office`

- Then users with the role "`reg_office`" also have all the privileges granted to "`staff`".

  I.e. "`reg_office`" is more powerful, it implies `staff`.

- It is possible to protect roles by passwords. Then this password must be entered in the `SET ROLE` command.
- Several roles are predefined in Oracle8, e.g. `CONNECT`, `RESOURCE`, and `DBA` (mainly for backward compatibility).

  These bundle system privileges necessary to work with Oracle. Students in this course have `CONNECT` and `RESOURCE` (but `UNLIMITED TABLESPACE` was revoked).

Creating Users in Oracle (1)

User Authentication:

- Oracle can perform the user authentication itself, then users have to enter a user name and a password:

  `CREATE USER sbrass IDENTIFIED BY my_password`

  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 OS:

  `CREATE USER ops$sbrass IDENTIFIED EXTERNALLY`

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

- An external network authentication service can also be used.

Creating Users in Oracle (2)

Giving System Privileges to New Users:

- A user created as explained above has no rights, not even the system privilege to connect to the DB.
- So the necessary privileges must be given explicitly, e.g.

  `GRANT connect, resource TO sbrass`

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

  `ALTER USER sbrass DEFAULT ROLE ALL`

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

Creating Users in Oracle (3)

Tablespaces:

- A tablespace is a database file or a collection of DB files.
- All tablespaces are listed in

  `DBA_TABLESPACES(TABLESPACE_NAME, ...)`

  Tablespaces accessible by the current user: `USER_TABLESPACES`.
- The actual files for each tablespace are listed in:

```
DBA_DATA_FILES(FILE_NAME, FILE_ID,
               TABLESPACE_NAME, BYTES, ...)
```

  See also `DBA_FREE_SPACE`/`DBA_FREE_SPACE_COALESCED`.
- The tablespace "`SYSTEM`" contains e.g. the data dictionary.

Creating Users in Oracle (4)

Defining Tablespaces and Quotas:

- ```
  CREATE USER sbrass IDENTIFIED BY my_password
              DEFAULT TABLESPACE user_data
              TEMPORARY TABLESPACE temporary_data
              QUOTA 2M ON user_data
              QUOTA UNLIMITED ON temporary_data
  ```
- `DEFAULT TABLESPACE` is used for tables created by the user.

  If he/she does not specify a tablespace. Default is `SYSTEM`.
- Without `QUOTA` clauses, the user has no right to access the tablespaces (unless he/she has "`UNLIMITED TABLESPACE`").

  Use: `REVOKE UNLIMITED TABLESPACE FROM sbrass`

Creating Users in Oracle (5)

Changing and Deleting Users:

- If a user has forgotten his/her password:

  `ALTER USER sbrass IDENTIFIED BY new_password`
- A user without tables can be deleted in this way:

  `DROP USER sbrass`
- To delete the user including all his/her data, use:

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

  `ALTER USER sbrass ACCOUNT LOCK`

Some Predefined Users

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

  This is the most powerful account. The default password is: `CHANGE_ON_INSTALL`.
- `SYSTEM`: The default database administrator.

  For most administration tasks. Default password: `MANAGER`.
- `SCOTT`: Guest and demonstration account.

  Default password: `TIGER`.
- `OUTLN`: Schema contains information for optimizer.

  Default password: `OUTLN`.
- `DBSNMP`: Information for "intelligent agent" which allows remote administration via the "enterprise manager".

  Default password: `DBSNMP`.
  ```

System Administrator Accounts

- Whereas the above passwords are stored in the database (encrypted), there usually is another password file for 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 in Oracle

- Limit the resource usage of database users can be restricted by creating a "profile" for them. This defines e.g.
  - How many concurrent sessions the user can have.
  - After what idle time he/she is logged off.
  - How much CPU time is allowed per session/per call.
  - How many logical reads (disk accesses) are allowed per session/per call.
  - After what time a password must be changed.
  - Which function should be used to check the password complexity.
- Oracle also has an `AUDIT` command (see manual).

Overview

1. Requirements
2. GRANT Command
3. Oracle
4. DB2
5. SQL Server

Users in DB2

- DB2 uses the authentication mechanism of the underlying operating system (DB2 itself does not manage passwords).
- DB2 has something similar to the "system privileges" of Oracle, they are called "Database Authorities" in DB2.
- An operating system user can connect to a database if he/she has the "`CONNECT`" authority.
- E.g. an administrator can "create" a user in the database by issuing this command:

  `GRANT CONNECT ON DATABASE TO sbrass`
- This right can also be granted to "`PUBLIC`", in which case every OS user can connect to the database.

Database Authorities in DB2

DB2 has only six "system privileges":

- CONNECT: Right to log into the database.
- CREATETAB: Right to create tables.
  No right is required for view creation.
- BINDADD: Right to create packages.
  I.e. to compile application programs with embedded SQL.
- IMPLICIT_SCHEMA: Right to create tables in new schemas.
- CREATE_NOT_FENCED: Right to link user-defined functions to the database code.
- DBADM: Access and modify all objects in the DB, grant any object privilege or DB authority except DBADM.

Groups in DB2

- DB2 has no roles, but it understands the concept of groups of the underlying operating system.
  E.g. UNIX and Windows NT have groups of users.
- Privileges can be granted not only to users, but also to groups.
  However, privileges granted to groups are not used when binding a package. The reason is that the package might exist for a long time in the database, but the OS does not inform DB2 when a user is removed from a group.
- So there are three ways a user might get a privilege:
  - It can be granted to this user personally,
  - to a group in which he/she is member,
  - or to PUBLIC.

Instance-Level Authorities

- A DB2 instance can manage several databases.
  Instance: server process. E.g. use "CONNECT TO SAMPLE" (a specific DB) after logging into DB2.
- The most powerful right in DB2 is the SYSADM authority.
  The account under which DB2 is installed plus all members of its group get SYSADM authority. It cannot be granted or revoked, but the group membership can be changed in the OS.
- There are also two other groups with fewer rights:
  - SYSCTRL: Can e.g. create or delete databases and tablespaces (plus all SYSMAINT privileges).
  - SYSMAINT: Can e.g. start or stop the database, do backups and restore the database after a crash.

Overview

1. Requirements
2. GRANT Command
3. Oracle
4. DB2
5. SQL Server

Creating New Users (1)

- SQL Server has two authentication mechanisms:
  - Windows NT Users or Groups can be mapped to SQL server logins.

    Then Windows NT does the authentication, there is no need to enter again a password. Users from trusted clients (which must run Windows NT) can also be mapped to SQL Server logins.
  - SQL Server Authentication (with password)

    In this case, SQL Server requires login name and password. This is the only possibility if SQL Server runs on Windows 95/98.

Creating New Users (2)

- One SQL Server Instance can manage several databases. The databases can have different users.
- It is necessary to distinguish between a login into the server, and the user in a specific database.

  Each login has a default DB to which it will be connected. The username in a DB may be different from the server login, and in different DBs different usernames can be used.
- Logins / users can be created with the Enterprise Manager.

  You can also use the procedures `sp_addlogin` (SQL Server auth.) and `sp_grantlogin` (Windows NT auth.) to create a login and `sp_grantdbaccess` to allow the login access to a specific database (also needed for default DB).

Special Users

- There is a login "`sa`" (system/server administrator). Everything can be done under this login.

  It uses SQL Server authentication (no password by default!).
- Every database has a user "`dbo`" (DB owner).

  Any member of the `sysadmin` sever role (e.g. "`sa`") is mapped to this user when he/she connects to a database.

  If a table is referenced without specifying a user, SQL Server first tries to find it in the account/schema of the current user, and then in the account/schema of "`dbo`".

  This eliminates the need for synonyms as used in Oracle.
- Some databases may have a "`guest`" user.

  A server login not mapped to a db user is mapped to `guest`.

Roles and System Privileges (1)

- SQL Server has the concept of roles (user groups) which seems to be basically the same as in Oracle.

  In addition SQL Server also uses Windows NT groups.
- However, some roles are special and contain administration rights which cannot explicitly be granted.
- Fixed server roles, listed below, allow administration of the server. The most powerful is "`sysadmin`".

  The login "`sa`" belongs to the role "`sysadmin`".
- Fixed database roles, listed below, allow administration of a database. The most powerful one is "`db_owner`".

  The user "`dbo`" belongs to the role "`db_owner`".

Roles and System Privileges (2)

- In addition, SQL Server has system privileges basically as in Oracle (can be managed via GRANT/REVOKE):
  - CREATE DATABASE
  - CREATE DEFAULT
  - CREATE PROCEDURE
  - CREATE RULE
  - CREATE TABLE
  - CREATE VIEW
  - BACKUP DATABASE
  - BACKUP LOG

Fixed Server Roles

- `sysadmin`: Right to perform any activity in SQL Server.
- `securityadmin`: Right to manage logins.
- `serveradmin`: Right to configure server-wide settings.
- `setupadmin`: Right to execute some system stored procedures, e.g. `sp_serveroption`.
- `processadmin`: Right to kill processes of other users.
- `diskadmin`: Right to manage the disk files.
- `dbcreator`: Right to create and alter databases.

Fixed Database Roles

- `db_owner`: Right to perform any activity in the database.
- `db_accessadmin`: Right to manage users of the database.
- `db_securityadmin`: Right to manage roles and permissions.
- `db_ddladmin`: Right to create, modify, drop any DB object.
- `db_backupoperator`: Right to make a backup copy of the DB.
- `db_datareader`: Right to read all tables in the database.
- `db_datawriter`: Right to modify all tables in the database.
- `denydatareader`: Forbids to read any table in the database.
- `denydatawriter`: Forbids to modify any table in the database.

DENY Statement

- SQL Server has also a DENY statement, which is the opposite of GRANT, e.g.: `DENY SELECT ON Course TO sbrass`
- The idea is that `sbrass` might be member of a group or role, which has the right, and you might want to declare `sbrass` as an exception.
- So the DENY on the user level takes precedence over the GRANT on the group/role/public level.

  The real motivation probably was that in an older version of SQL Server, a REVOKE really removed the right, whereas now the user can still have it granted via another path or granted via a group/role. So the developers introduced a statement which had the old REVOKE semantics.