Datenbank-Programmierung

Anhang E: Mehr zu Zugriffsrechten in konkreten DBMS

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Sommersemester 2019

http://www.informatik.uni-halle.de/~brass/dbp19/

Inhalt

Nutzer und DB-Schemata (1)

- In Oracle muss man "⟨Benutzer⟩.⟨Objekt⟩" schreiben, um auf DB Objekte (Tabellen, Sichten, etc.) von anderen Nutzern zuzugreifen, z.B.:

```
SELECT * FROM BRASS.AUFGABE
```

- Natürlich geht das nur, wenn der Nutzer `BRASS` das `SELECT`-Recht an der Tabelle `AUFGABE` vergeben hat.

  Entweder direkt an Sie (den Nutzer, der den Befehl eingibt) oder an `PUBLIC` (eine weitere Alternative wären Rollen/Gruppen, s.u.). Wenn ein Nutzer, der nicht einmal das `SELECT`-Recht für eine Tabelle hat, versucht, auf diese zuzugreifen, gibt Oracle die gleiche Fehlermeldung, wie wenn die Tabelle nicht existiert. So ist auch die Information über die Existenz der Tabelle geschützt.

Nutzer und DB-Schemata (2)

- In Oracle ist der Nutzer-Name gleichzeitig Schema-Name. D.h. jeder Nutzer hat genau ein Schema, und jedes Schema gehört genau einem Nutzer.
- Eine eindeutige Identifikation eines Objektes (Tabelle etc.) innerhalb der Datenbank besteht also aus Nutzer-Name und Objekt-Name.
- Wenn man in Oracle zwei Mengen von Tabellen trennen will (also zwei Schemata braucht), muss man zwei Benutzer-Accounts verwenden.

Oracle unterstützt Schemata nicht unabhängig von Benutzerkonten.

Synonyme (1)

- Da die zweiteiligen Namen etwas umständlich sind, kann man in Oracle Abkürzungen definieren:

  ```
  CREATE SYNONYM AUFGABE FOR BRASS.AUFGABE
  ```

  Schreibt man nun "`AUFGABE`" (ohne vorangestellten Nutzer), wird dies intern durch "`BRASS.AUFGABE`" ersetzt.

- Dieses Synonym (Abkürzung, Macro) gilt nur für den Nutzer, der es definiert hat.

- Ein DBA kann aber auch "`PUBLIC`" Synonyme definieren. Dann sieht es so aus, als würde die Tabelle "`AUFGABE`" unter jedem Benutzer-Account existieren.

  Obwohl es tatsächlich nur eine Tabelle ist.

Synonyme (2)

- Solche "PUBLIC" Synonyme werden für den Systemkatalog ("Data Dictionary") genutzt.

  Z.B. ist `CAT` (Liste aller eigenen Tabellen) ein Synonym für die Sicht `SYS.USER_CATALOG` (dem Benutzer `SYS` gehört der Systemkatalog).

- Selbst wenn ein "PUBLIC" Synonym X definiert ist, kann ein Nutzer seine eigene Tabelle X anlegen.

  Dann ist X für diesen Nutzer seine eigene Tabelle. Er kann auf das Objekt aber noch mit "⟨User⟩.⟨Table⟩" zugreifen.

- Löschen von Synonymen: `DROP SYNONYM AUFGABE`

- Synonyme sind nicht Teil des SQL-92 Standards.

System-Rechte (1)

- Neben den oben beschriebenen Zugriffsrechten für Tabellen etc. (“object privileges”) hat Oracle auch System-Rechte (“system privileges”).
- Diese beziehen sich auf die Ausführung bestimmter Kommandos, nicht auf Datenbank-Objekte.
- Z.B. ist das System-Recht “`CREATE TABLE`” nötig, um dieses Kommando ausführen zu können.

  Ein Nutzer, der nur Daten eingeben soll, braucht keine Tabellen anlegen zu können. Für ein sicheres System sollte jeder Nutzer nur die Kommandos ausführen konnen, die er für seine Aufgabe sinnvoll benötigen könnte. Mit Objektrechten alleine könnte die Nutzung von `CREATE TABLE` nicht eingeschränkt werden.

System-Rechte (2)

- Um sich bei Oracle anmelden zu können, braucht man das System-Recht "`CREATE SESSION`".

  Ein Benutzerkonto kann gesperrt werden, indem dieses Recht entzogen wird. Die Tabellen etc. des entsprechenden Schema bleiben erhalten, und können von anderen Nutzern (mit entsprechenden Zugriffsrechten) genutzt werden.

- Viele System-Rechte sind nur für DBAs gedacht, z.B. gibt "`SELECT ANY TABLE`" Lesezugriff für alle Tabellen in der Datenbank.

  Auch wenn man das `SELECT`-Objekt-Recht für die Tabelle nicht hat.

System-Rechte (3)

- Da die DBA-Rechte in viele verschiedene System-Rechte aufgeteilt sind, kann man mehrere DBAs mit unterschiedlichen Aufgabenbereichen haben.

  Z.B. muss ein Mitarbeiter, der für die Backups zuständig ist, nicht unbedingt das "`CREATE USER`" Recht haben, um auch neue Benutzer anlegen zu können. Selbstverständlich erlaubt das Modell aber auch, einen DBA mit allen System-Rechten zu haben.

- Es gibt mehr als 90 verschiedene System-Rechte.

  Im wesentlichen gibt es für jedes Administrations-Kommando ein entsprechendes System-Recht. Verschiedene Arten von `CREATE`-Befehlen sind auch System-Rechte (da sie sonst nicht eingeschränkt werden könnten). Die meisten Befehle haben eine `ANY`-Version als System-Recht (erlaubt das Kommando auf Objekte aller Nutzer anzuwenden).

System-Rechte (4)

- Besitzt ein Nutzer ein System-Recht "`WITH ADMIN OPTION`" hat, kann er/sie es weitergeben:

```
GRANT CREATE TABLE TO SCOTT
```

  Fügt man "`WITH ADMIN OPTION`" hinzu, kann anschließend auch `SCOTT` das "`CREATE TABLE`"-Recht vergeben.

- Wenn ein Systemrecht einem Nutzer A entzogen wird, der es "`WITH ADMIN OPTION`" hatte, wird das Recht nicht rekursiv Nutzern B entzogen, die es von A bekommen haben.

  Vermutlich heißt es deswegen nicht "`GRANT OPTION`". Aber es ist sehr ähnlich ("`GRANT OPTION`" gibt es nur für Objekt-Rechte).

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` ⟨`Name`⟩

- Only those with the system privilege "`CREATE ROLE`" can execute this command (e.g. only the DBA).

Roles (2)

- Access rights are granted to a role (e.g. "STAFF") in the same way as they are granted to a user:

```
GRANT SELECT ON COURSES 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 can be granted to users:

```
GRANT STAFF TO JIM, MARY
```

Roles (3)

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

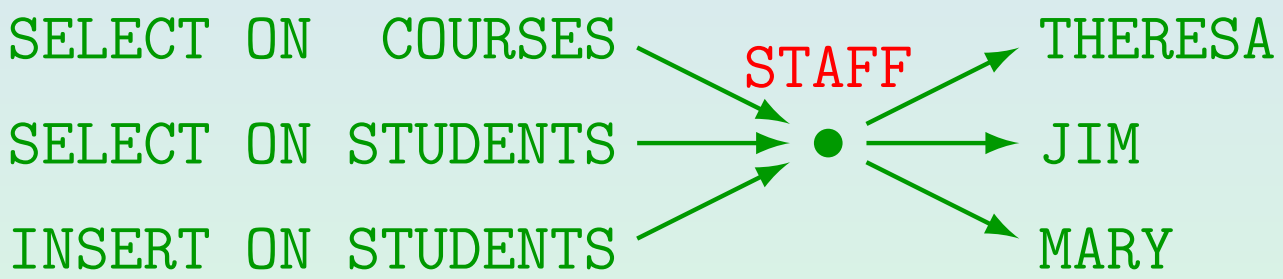
  But if "`STAFF`" is not one of the default roles of these users, which are automatically activated when they log in, they must explicity execute "`SET ROLE STAFF`" in every session in which they want to use these privileges. (It seems that this is not enforced in Oracle 8.0.)

- Only the owner of the role or a user who has received it with admin option can grant the role to another user.

```
GRANT STAFF TO THERESA WITH ADMIN OPTION
```

Roles (4)

- Roles are a level of indirection between 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 (5)

- 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.

- Roles can be protected by passwords. Then the SET ROLE command requires a password.

Roles (6)

- 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.

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
         QUOTA UNLIMITED ON TEMPORARY_DATA
  ```
- 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`
  ```

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
```

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!

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) is 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;
```

Inhalt

Users in DB2 (1)

- 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" and "Instance-Level Authorities" in DB2.
- A DB2 instance can manage several databases.

  Instance: server process. Database: Collection of DB schemas (and their table data). E.g. use "`CONNECT TO SAMPLE`" (a specific DB) after logging into DB2.

Users in DB2 (2)

- 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 BRASS
```

- This right can also be granted to "PUBLIC", in which case every OS user can connect to the database.

Database Authorities in DB2

- `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`: Create tables in new schemas.
- `CREATE_NOT_FENCED`: Extend DBMS (add functions).
- `DBADM`: Access and modify all DB objects, 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.

  Privileges granted to groups are not used when binding a package.

- 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

- 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.

Inhalt

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.
- One SQL Server Instance can manage several databases. The databases can have different users.

Creating New Users (2)

- 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 (graphical interface).

  Alternatively, one can use the system procedures `sp_addlogin` (SQL Server authentication) and `sp_grantlogin` (Windows NT authentication) to create a login and `sp_grantdbaccess` to allow the login to access a specific database (also needed for the default DB of the login).

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

- 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 allow administration of the server. The most powerful is "`sysadmin`" (e.g. "`sa`").
- Fixed database roles allow administration of a DB. The most powerful one is "`db_owner`" (e.g. "`dbo`").

System Privileges

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

Fixed Server Roles

- `sysadmin`: Perform any activity in SQL Server.
- `securityadmin`: Manage logins.
- `serveradmin`: Configure server-wide settings.
- `setupadmin`: Execute some system stored procedures, e.g. `sp_serveroption`.
- `processadmin`: Kill processes of other users.
- `diskadmin`: Manage the disk files.
- `dbcreator`: Create and alter databases.

Fixed Database Roles

- `db_owner`: Perform any activity in the database.
- `db_accessadmin`: Manage users of the database.
- `db_securityadmin`: Manage roles and permissions.
- `db_ddladmin`: Create, modify, drop any DB object.
- `db_backupoperator`: Make a backup copy of the DB.
- `db_datareader`: Read all tables in the database.
- `db_datawriter`: Modify all tables in the database.
- `denydatareader`, `denydatawriter`: These roles forbids to read/modify any table in the database.

DENY Statement

- SQL Server has a statement which is the opposite of `GRANT`, e.g.:

```
DENY SELECT ON COURSES TO BRASS
```

- The idea is that `BRASS` might be member of a group or role, which has the right, and you might want to declare `BRASS` as an exception.

  In an older version of SQL Server, `REVOKE` worked like `DENY` now. When an SQL92-conforming `REVOKE` was introduced, the old was called `DENY`.

- So the `DENY` on the user level takes precedence over the `GRANT` on the group/role/public level.

Literatur/Quellen

- 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.