Datenbank-Programmierung

Kapitel 7: Zugriffsrechte in SQL

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Sommersemester 2024

http://www.informatik.uni-halle.de/~brass/dbp24/

Lernziele

Nach diesem Kapitel sollten Sie Folgendes können:

- Sicherheitsrelevante DBMS Funktionen erklären.
- Die SQL Befehle `GRANT` und `REVOKE` benutzen.

Inhalt

Zugriffsrechte (1)

- Oft haben verschiedene Nutzer einer Datenbank auch verschiedene Rechte.
- Kommandos bestehen aus
  - „Subject“ (der Benutzer, der es ausführt),
  - „Verb“ (die Operation, z.B. „`INSERT`), und
  - „Objekt“ (typischerweise eine Tabelle).
- In SQL kann man mit dem `GRANT`-Kommando (s.u.) definieren, welche dieser Tripel erlaubt sind.

  Das DBMS speichert also eine Menge von Tripeln (u, o, r), aus Benutzer u, Operation o (im wesentlichen `SELECT`, `INSERT`, `UPDATE`, `DELETE`), Objekt/Relation r. Eigentlich Quadrupel: Wer hat das Recht erteilt?

Zugriffsrechte (2)

- Das Subjekt-Verb-Objekt Modell für die Zugriffsrechte hat aber Einschränkungen:
  - Kommandos wie „`CREATE TABLE`“ beziehen sich nicht auf existierende Objekte, aber man muss ihre Benutzung einschränken können.

    Prinzip: Jeder sollte nur das tun können, was er tun muss.
  - Für DB-Administratoren sollen die normalen Zugriffsbeschränkungen nicht gelten. Es soll aber auch nicht jeder Administrator alles können.

    Große Firmen haben mehrere Administratoren.

Zugriffsrechte (3)

- Während das Subjekt-Verb-Objekt Modell sehr portabel ist (schon im SQL-86 Standard enthalten), sind die Erweiterungen zur Lösung der obigen Probleme meist vorhanden, aber sehr systemabhängig:
  - Oracle hat neben Objektrechten nach dem obigen Modell noch eine große Anzahl von Systemrechten (wer darf welches Kommando/Verb benutzen, ggf. auch für beliebige Objekte).
  - Andere Systeme unterscheiden oft nur wenige Benutzerklassen, die verschieden privilegiert sind.

Zugriffsrechte (4)

- Sichten und serverseitige Prozeduren („stored procedures“) erlauben es, Objekte zu verkapseln:
  - Z.B. ist es möglich, dass ein Benutzer für eine Tabelle selbst kein `SELECT`-Recht hat, aber über eine Sicht bestimmte Zeilen/Spalten oder aggregierte Daten sehen darf.
  - Es ist auch möglich, dass ein Benutzer kein direktes `INSERT`-Recht für eine Tabelle hat, aber eine Prozedur benutzen kann, die Daten in diese Tabelle nach zusätzlichen Prüfungen einfügt.

Inhalt

GRANT Kommando (1)

- In SQL können Zugriffsrechte für Datenbankobjekte (Tabellen, Sichten, etc.) an andere Benutzer mit dem `GRANT` Kommando gegeben werden.
- `GRANT` war schon im SQL-86 Standard enthalten.
  Es hat die Form

  `GRANT` ⟨Rechte⟩ `ON` ⟨Objekt⟩ `TO` ⟨Nutzer⟩
- Das DBMS speichert diese Tripel im Systemkatalog ab, und prüft bei jedem Zugriff, ob der aktuelle Nutzer das entsprechende Recht hat.

  Es wird noch zusätzlich vermerkt, wer das Recht vergeben hat.

GRANT Kommando (2)

- Angenommen, die Tabelle `AUFGABEN` gehört dem Benutzer `BRASS`.
- Er kann Lese- und Einfüge-Rechte ("privileges") für diese Tabelle den Benutzern `MEIER` und `JUNG` mit folgendem Kommando geben:

```
GRANT SELECT, INSERT ON AUFGABEN TO MEIER, JUNG
```

- Dann können `MEIER` und `JUNG` z.B. Folgendes tun:
  - `SELECT * FROM BRASS.AUFGABEN`
  - `INSERT INTO BRASS.AUFGABEN VALUES (...)`

    Die Beispiele sind noch aus Oracle, wo Schemata 1:1 den Benutzern entsprechen.

GRANT Kommando (3)

- `MEIER` und `JUNG` können die einmal eingegebenen Tabellenzeilen aber nicht ändern oder löschen.

  Auch nicht die Zeilen, die sie selbst eingegeben haben. Die Zeilen werden Bestandteil der Tabelle, die `BRASS` gehört. Wenn man nicht selbst etwas programmiert (z.B. eine Spalte mit `DEFAULT USER` (in Oracle), für die nicht explizit ein Wert angegeben werden kann: s.u.), ist nicht nachzuvollziehen, wer eine bestimmte Zeile eingegeben hat. Manche DBMS haben Erweiterungen für diesen Zweck.

- Man kann später z.B. dem Benutzer `MEIER` noch zusätzlich die fehlenden Rechte geben:

```
GRANT UPDATE, DELETE ON AUFGABEN TO MEIER
```

  Die bereits vorher vergebenen Rechte bleiben dabei bestehen.

GRANT Kommando (4)

Rechte für Tabellen/Sichten in SQL-92:

- `SELECT`: Lesezugriff (Anfragen an die Tabelle).

  In SQL Server kann man `SELECT` für einzelne Spalten vergeben. Das ist nicht im SQL-92 Standard und geht nicht in Oracle und DB2. Aber Sichten haben den gleichen Effekt.

- `INSERT`: Einfügen neuer Zeilen.

- `INSERT(`$A_1, \ldots, A_n$`)`: Nur für die Spalten A_i können Werte angegeben werden, die übrigen werden mit dem jeweiligen Defaultwert gefüllt ($\rightarrow$ `CREATE TABLE`).

  `INSERT` nur für bestimmte Spalten ist Teil des SQL-92 Standards, aber nur in Oracle möglich (nicht in SQL Server und DB2). Mit Sichten kann man den gleichen Effekt erreichen.

GRANT Kommando (5)

Rechte für Tabellen/Sichten in SQL-92, Forts.:

- `UPDATE`: Änderung von Tabelleneinträgen.
- `UPDATE`$(A_1, \ldots, A_n)$: Nur Daten in den Spalten A_i können geändert werden.
- `DELETE`: Löschung von Tabellenzeilen.
- `REFERENCES`: Anlegen von Fremdschlüsseln, die auf diese Tabelle verweisen.

  Wenn Benutzer A in seiner Tabelle R auf die Tabelle S des Benutzers B verweist, aber B keine `DELETE`-Rechte an R bekommt, kann B referenzierte Zeilen aus seiner eigenen Tabelle S nicht mehr löschen. A kann auch prüfen, ob ein Schlüsselwert in S existiert.

GRANT Kommando (6)

Rechte für Tabellen/Sichten in SQL-92, Forts.:

- `REFERENCES` $(A_1, \ldots, A_n)$: Nur dieser Schlüssel darf in Fremdschlüsseln referenziert werden.

  Das ist nur wichtig, wenn eine Tabelle mehrere Schlüssel hat. Unterstützt in Oracle und DB2 (nicht in SQL Server).

Rechte für Domains/Zeichensätze, etc. in SQL-92:

- `USAGE`: Verwendung z.B. der Domain (benutzerdefinierter Datentyp) in `CREATE TABLE` Anweisungen.

  In keinem der drei DBMS (Oracle, DB2, SQL Server) unterstützt.

GRANT Kommando (7)

Weitere Rechte für Tabellen (nicht im Standard):

- `ALTER`: Recht, die Tabellendefinition zu ändern.

  Unterstützt in Oracle und DB2, nicht in SQL Server.

- `INDEX`: Recht, einen Index für de Tabelle anzulegen.

  Unterstützt in Oracle und DB2, nicht in SQL Server.

Rechte für Prozeduren/Packages (nicht im Standard):

- `EXECUTE`: Recht, die Prozedur auszuführen.

  Unterstützt in allen drei DBMS.

- `BIND`: Neuoptimierung von SQL-Anweisungen.

  Nur in DB2 für "Packages" (SQL-Anweisungen eines Programms).

GRANT Kommando (8)

Rechte für Schema Objekte (nur DB2):

- `ALTERIN`: Recht, jedes Objekt (z.B. Tabelle) des Schemas zu ändern.
- `CREATEIN`: Recht, Objekte im Schema anzulegen.
- `DROPIN`: Recht, Objekte im Schema zu löschen.

Rechte für Directory Objekte (nur Oracle):

- `READ`: Recht, Dateien im Directory zu lesen.

Alle Rechte auf dieser Folie sind nicht im SQL Standard enthalten.

GRANT Kommando (9)

GRANT ALL PRIVILEGES:

- Anstatt einzelne Rechte aufzulisten, geht auch:

  GRANT ALL PRIVILEGES ON ⟨Object⟩ TO ⟨Users⟩

- Dies sind alle Rechte, die der Benutzer, der das GRANT-Kommando ausführt, selber für ⟨Object⟩ hat.

TO PUBLIC:

- Um Rechte an alle Benutzer der Datenbank zu geben (auch an zukünftige Nutzer), schreibt man:

  GRANT SELECT ON COURSES TO PUBLIC

GRANT Kommando (10)

WITH GRANT OPTION:

- Man kann einem Benutzer ein Recht auch mit der Möglichkeit geben, das Recht weiterzugeben.
- Dazu hängt man die Klausel "WITH GRANT OPTION" an das GRANT-Kommando an:

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

- Dies erlaubt dem Benutzer BRASS auch, die GRANT-Option an andere Benutzer weiterzugeben (die es dann selbst weitergeben können, u.s.w.).

GRANT Kommando (11)

Besitzer eines Objektes:

- Der Besitzer eines Objektes (z.B. einer Tabelle), d.h. der Benutzer, der die Tabelle angelegt hat, hat alle Rechte an dem Objekt inklusive `GRANT OPTION`.

  Direkt nach Anlegen der Tabelle hat nur der Besitzer Rechte an ihr. Andere Benutzer bekommen diese Rechte nur durch explizite `GRANT`s. Allerdings können Benutzer mit Administrator-Rechten auf Tabellen oft auch ohne die mit `GRANT` vergebenen Rechte zugreifen.

- Der Besitzer eines Objektes kann es wieder löschen (`DROP TABLE`).

  Dieses Recht ist nicht in den anderen Rechten enthalten. Mit `DELETE` kann man nur den Inhalt der Tabelle löschen.

GRANT Kommando (12)

CONTROL-Recht in DB2:

- Dies gibt alle Rechte am Objekt mit GRANT OPTION, und erlaubt auch, das Objekt zu löschen.

  In Oracle und SQL Server gibt es kein CONTROL-Recht. Es entspricht den Rechten des Besitzers einer Tabelle.

- Wenn ein Benutzer eine Tabelle neu anlegt, hat er/sie das CONTROL-Recht für diese Tabelle.

  Bei Sichten bekommt der Benutzer das CONTROL-Recht nur, wenn er/sie es auch für die verwendeten Basistabellen und Sichten hat. Das CONTROL-Recht wird immer ohne GRANT OPTION vergeben. Es kann nur durch Administratoren vergeben werden. Um ein CONTROL-Recht vergeben zu können, muss man die SYSADM oder DBADM-Stufe ("Authority") haben. Das CONTROL-Recht hat Bedeutung für REVOKE (s.u.).

Inhalt

Zugriffsrechte entziehen (1)

- Vergebene Zugriffsrechte können wieder entzogen (zurückgenommen) werden mit einem Kommando, das dem `GRANT`-Befehl sehr ähnlich ist:

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

- `⟨Rights⟩`: Liste von einzelnen Rechten (z.B. `SELECT`), durch Kommata getrennt, oder "`ALL PRIVILEGES`".
- `⟨Object⟩`: Name eines DB-Objektes (z.B. Tabelle).
- `⟨Users⟩`: Liste von Nutzernamen (durch Komma getrennt) oder "`PUBLIC`".

Zugriffsrechte entziehen (2)

- Beispiel:

```
REVOKE INSERT ON AUFGABEN FROM MEIER
```

  Wenn der Benutzer `MEIER` vor diesem Kommando `SELECT` und `INSERT`-Rechte an `AUFGABEN` hatte, hat er hinterher nur noch `SELECT`-Rechte.

- Benutzer können nur Rechte entziehen, die sie vorher vergeben haben.

  Deswegen speichert das DBMS in seinen internen Tabellen nicht nur das Tripel Benutzer-Recht-Objekt sondern das Quadrupel (A, P, O, B):
  Benutzer A hat das Recht P für Objekt O an Benutzer B vergeben.

Zugriffsrechte entziehen (3)

- Man kann ein an `PUBLIC` vergebenes Recht nicht selektiv einzelnen Nutzern entziehen.

  Man kann es natürlich zurücknehmen, aber dann verlieren es alle Nutzer. Da `PUBLIC` auch zukünftige Nutzer enthält, speichert die DB nur, dass es an `PUBLIC` vergeben wurde (nicht einzeln für jeden Nutzer).

- Man kann aber "`ALL PRIVILEGES`" vergeben, und die Rechte später selektiv zurücknehmen.

  "`ALL PRIVILEGES`" sind die Rechte, die der Benutzer aktuell hat. Sie werden jeweils einzeln in der Systemtabelle gespeichert.

- Wenn eine Tabelle gelöscht wird, werden auch alle dafür vergebenen Zugriffsrechte gelöscht.

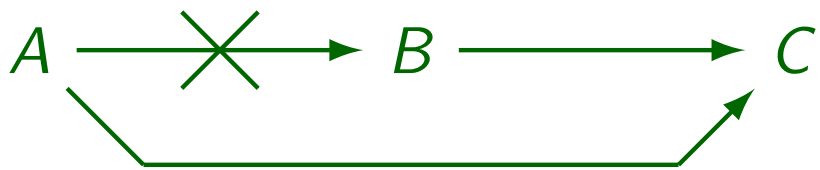
  Wird sie dann neu angelegt, kann nur der Besitzer darauf zugreifen.

Zugriffsrechte entziehen (4)

- Hat A ein Recht "`WITH GRANT OPTION`" an B gegeben, und B es an C weitergegeben, und A entzieht das Recht dann B, so verliert es rekursiv auch C:

In SQL-92 benötigt dies allerdings `CASCADE`, siehe unten.

- Falls C das Recht allerdings zusätzlich auf einem anderen Pfad bekommen hat (z.B. direkt von A), behält C das Recht:



Zugriffsrechte entziehen (5)

- Das `REVOKE` Kommando soll die gleiche Situation wieder herstellen, als hätte B das Recht nie gehabt.

  Nur bezüglich der Zugriffsrechte: Wenn B das Recht genutzt hat, um Tabellen zu ändern, bleiben diese Änderungen natürlich bestehen.

- SQL-86 hatte kein `REVOKE`-Kommando.

  Das rekursive Entziehen ist nicht ganz einfach: Man muss Pfade im Graphen der einzelnen Weitergaben verfolgen. Insbesondere geht das nicht mit einer (klassischen) SQL-Anfrage. Man braucht eine deduktive Datenbank oder rekursive Sichten in SQL.

Zugriffsrechte entziehen (6)

Systemabhängige Details:

- In DB2 kann man nur mit `CONTROL`-Recht für ein Objekt X Rechte für X entziehen.

  Es ist etwas komisch, dass man mit der `GRANT OPTION` das Recht weitergeben kann, aber dann nicht mehr zurücknehmen.

- Auf diese Art braucht DB2 nicht darüber Buch zu führen, auf welchem Weg ein Nutzer das Recht bekommen hat.

  Wenn ein Nutzer mit `CONTROL`-Recht das Recht entzieht, ist es weg (außer ggf. über Gruppen/`PUBLIC`).

Zugriffsrechte entziehen (7)

Systemabhängige Details, Forts.:

- In SQL-92 und SQL Server muss man `CASCADE` zum Kommando hinzufügen, um Rechte rekursiv zu entziehen, z.B.

```
REVOKE INSERT ON AUFGABEN FROM MEIER CASCADE
```

In SQL Server, muss `CASCADE` immer angegeben werden, wenn man ein Recht entziehen will, das `WITH GRANT OPTION` verliehen wurde. In SQL-92 muss man dies nur angeben, wenn der Nutzer das Recht tatsächlich weitergegeben hat. In SQL-92 kann man alternativ `RESTRICT` statt `CASCADE` angeben: Dann wird das `REVOKE` nur ausgeführt, wenn keine Rechte von anderen Nutzern davon abhängen. SQL Server versteht `RESTRICT` nicht. Oracle und DB2 kennen gar kein `CASCADE`/`RESTRICT`.

Zugriffsrechte entziehen (8)

Systemabhängige Details, Forts.:

- SQL-92 hat auch "`REVOKE GRANT OPTION FOR ...`".

  Dies wird nur von SQL Server verstanden, nicht von Oracle oder DB2.

  SQL Server verlangt, dass dann auch `CASCADE` angegeben wird.

- In Oracle muss man "`CASCADE CONSTRAINTS`" hinzufügen, wenn man ein `REFERENCES`-Recht zurücknehmen will, das genutzt wurde, um Fremdschlüssel anzulegen.

  - Die Fremdschlüssel-Integritätsbedingungen werden dann gelöscht.

Literatur/Quellen

- 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.
- K. Nagel: Informationsbroschüre zum Bundesdatenschutzgesetz, 10. Auflage, Oldenbourg, 2001.