Leibniz Universität IT Services

Leibniz Universität Hannover

S(tructured)Q(uery)L(anguage) Benutzerverwaltung in Datenbanksystemen

Benutzerverwaltung

- Beachtung des Datenschutzes. Schutz sensibler Informationen vor Verbreitung.
- Schutz vor Angriffen von außen.
- In Abhängigkeit von Rollen werden abgestuft Rechte an einer Datenbank vergeben.

Benutzer

- Unabhängig von den Benutzer des Betriebssystem, auf dem das Datenbanksystem installiert ist.
- Benutzer werden für die einzelnen Datenbanken auf einem Server nur einmal angelegt.
- Benutzer identifizieren sich mit Hilfe eines Benutzernamens und -passwort.

Anlegen eines Benutzers

```
postgres =# CREATE USER BenuterA
postgres =# LOGIN PASSWORD 'benutzerA';
```

- Der Befehl `CREATE USER` legt einen Benutzer unter einen eindeutigen Namen an.
- Der Benutzer muss sich beim Anmelden an die Datenbank mit einem Passwort identifizieren.
- Hinweis: Unter PostgreSQL ist ein Benutzer nichts anderes als eine Rolle, die sich mit einem Benutzernamen und -passwort an eine Datenbank anmeldet.

Löschen eines Benutzers

```
postgres =# DROP USER BenuterA;
```

- Der Befehl `DROP USER` löscht einen vorhandenen Benutzer mit all seinen Rechten.

Rollen

- Zusammenfassung von Benutzer zu Gruppen.
- Rollen haben Rechte. Benutzer, denen einen Rolle zugeordnet wurden, übernehmen automatisch die Rechte der Rolle.
- Ein Benutzer kann mehrere Rollen annehmen. Der Benutzer ist mehreren Benutzergruppen zugeordnet.

Administrator (Superuser)

- Anlegen von Datenbanken.
- Lesende und schreibende Zugriffe auf alle Objekte in einer Datenbank.
- Verteilung von Rechten an Benutzern.
- Anlegen von Rollen.
- Bei der Nutzung von PostgreSQL: `postgres`. Automatische Anlage bei Ausführung des Befehls `CREATE DATABASE`.

Rolle anlegen

```
postgres =# CREATE ROLE admin SUPERUSER;
```

- Mit Hilfe des Befehls `CREATE ROLE` wird eine Benutzergruppe angelegt.
- Mit Hilfe des Attributs `SUPERUSER` wird dieser Rolle Administrator-Rechte übertragen.

Rolle „Darf Datenbank anlegen“

```
postgres =# CREATE ROLE datenbankAnlegen CREATEDB;
```

- Mit Hilfe des Befehls `CREATE ROLE` wird eine Benutzergruppe angelegt.
- Mit Hilfe des Attributs `CREATEDB` wird dieser Rolle Recht Datenbanken zu erstellen übertragen.
- Weitere Attribute: https://www.postgresql.org/docs/9.6/static/sql-createrole.html.

Rollen ohne Attribute

```
postgres =# CREATE ROLE vertrieb;
```

- Mit Hilfe des Befehls `CREATE ROLE` wird eine Benutzergruppe angelegt.
- Die angelegte Benutzergruppe hat keinen besonderen Rechte vom Administrator übertragen bekommen.
- Der Name der Rolle sollte den Zweck der Rolle widerspiegeln. In diesem Beispiel wird eine Rolle für die Aufgaben eines Vertriebs erstellt.

Löschen von Rollen

```
postgres =# DROP ROLE personal;
```

- Mit Hilfe des Befehls `DROP ROLE` wird eine vorhandene Benutzergruppe gelöscht.

… in PostgreSQL anzeigen

```
postgres =# \du
```

- Mit Hilfe der Option `\du` werden alle Rollen und Benutzer angezeigt.

Rollen und deren Attribute ändern

```
Postgres =# ALTER ROLE datenbankneu NOCREATEROLE;
```

- Mit Hilfe des Befehls `ALTER ROLE` können Attribute der Rolle verändert werden.

Umbenennung von Rollen / Benutzern

```
Postgres =# ALTER ROLE salesmanagement
Postgres =# RENAME TO verkauf;
```

- Dem Befehl `ALTER ROLE` folgt der Name der zu ändernden Rolle oder des Benutzers.
- Dem Befehl `RENAME TO` folgt die neue Bezeichnung für die Rolle.

Zuweisung von Rollen

```
Postgres =# GRANT buchhaltung TO meier, mueller;
```

- Dem Befehl `GRANT` folgt der Name der Rolle, der Benutzer zugewiesen werden sollen.
- Dem Schlüsselwort `TO` folgen die Namen der Benutzer, die der Rolle zugewiesen werden. Die Namen der Benutzer werden durch ein Komma getrennt.
- Hinweis: Die Benutzernamen müssen vor der Zuweisung angelegt werden.

Zugriffsrechte vergeben

```
Postgres =# GRANT ALL ON kunde_firma TO vertrieb;
```

- Dem Befehl `GRANT` folgt eine Beschreibung der Zugriffsrechte. `ALL` beschränkt den Zugriff nicht.
- Der Befehl `ON` folgt der Name der Tabelle, für die die Zugriffsrechte vergeben werden sollen.
- Dem Schlüsselwort `TO` folgen die Namen der Rollen, für die Zugriffsrechte gelten sollen.

Auflistung von Zugriffsrechten

```
Postgres =# GRANT SELECT, UPDATE ON
Postgres =# kunde_firma TO buchhaltung;
```

- Dem Befehl `GRANT` folgt eine Beschreibung der Zugriffsrechte.
- `SELECT` erlaubt nur einen lesenden Zugriff.
- UPDATE erlaubt einen schreibenden Zugriff auf bestehende Datensätze
- Die verschiedenen Zugriffsrechte werden durch ein Komma getrennt.

Möglichkeiten

- `ALL`. Alle Zugriffsrechte auf eine Tabelle.
- `SELECT`. Leserechte. Ausführen von Auswahlabfragen.
- `INSERT`. Das Einfügen neuer Datensätze in einer Tabelle ist erlaubt.
- `UPDATE`. Schreibrechte. Vorhandene Datensätze können geändert werden.
- `DELETE`. Recht zum Löschen von Datensätzen.
- Weitere Zugriffsrechte in PostgreSQL: https://www.postgresql.org/docs/9.0/static/sql-grant.html

Anzeige von Zugriffsrechten in PostgreSQL

```
Postgres =# \dp
```

- Tabellen und die Zugriffsrechte auf die Tabellen werden aufgelistet.

Datenbankschemata

- Alle Strukturinformationen zu Tabellen, Beziehungen und Indizes.
- Kapselung von Datenbankobjekten in Abhängigkeit einer bestimmten Benutzergruppe.
- Eine Datenbank kann mehrere Schemata enthalten.

Anlegen eines Datenbankschemas

```
CREATE SCHEMA rechnungen;
```

- Der Befehl `CREATE SCHEMA` legt ein Schema in einer Datenbank an.
- Das Schema hat einen eindeutigen Namen. Der Name sollte die Art der Sammlung widerspiegeln.

Hinzufügen einer Tabelle

```
CREATE SCHEMA rechnungen;

CREATE TABLE rechnung.zahlungsart(
      kuerzel  VARCHAR(3) PRIMARY KEY,
      zahlungsart VARCHAR(30)
);
```

- Der Befehl `CREATE TABLE` legt eine neue Tabelle an.
- Die Tabelle hat den Namen `zahlungsart`.
- Die Tabelle wird in dem Schema `vertrieb` angelegt.
- Der Name des Schemas und der Tabelle werden mit einem Punkt verbunden.

Anzeige von Zugriffsrechten in PostgreSQL

```
Postgres =# \dp
Postgres =# \dp vertrieb.*
```

- Die Option `\dp` zeigt alle Tabellen und die Zugriffsrechte darauf im öffentlichen Bereich an.
- Mit Hilfe von `\dp schema.*` werden alle Tabellen und deren Zugriffsrechte in einem bestimmten Schema angezeigt.

Schema erstellen und autorisieren

```
CREATE SCHEMA AUTHORIZATION vertrieb

CREATE TABLE Land(
      landkuerzel VARCHAR(3) PRIMARY KEY,
      landname VARCHAR(150)
)
CREATE TABLE Kunde_Firma(
      id SERIAL PRIMARY KEY,
      firmenname VARCHAR(100),
      strasse VARCHAR(100),
      postleitzahl VARCHAR(6),
      ort VARCHAR(100),
      idLand VARCHAR(3) REFERENCES Land(landkuerzel),
      aufgenommenAm TIMESTAMP,
      darfBestellen BOOLEAN
);
```

Erläuterung

- Der Befehl `CREATE SCHEMA AUTHORIZATION vertrieb` erstellt ein Schema `vertrieb`, welches für die Rolle `vertrieb` autorisiert wird.
- Das Schema und der Besitzer des Schemas werden gleichzeitig definiert.
- Hinweis: Die Erstellung des Schemas endet mit dem ersten Semikolon. In dem Beispiel auf der vorherigen Seite werden die Tabelle `Land` und `Kunde_Firma` automatisiert dem Schema `vertrieb` zugeordnet.

Besitzer von Objekten

- Jeder, der ein Objekt in einer Datenbank mit `CREATE` anlegt, ist Besitzer des Objekts.
- Jeder, der für ein Schema autorisiert ist, hat alle Rechte für alle Objekte in dem Schema.
- Der Administrator und der Besitzer eines Objekts haben automatisch auf das Objekt Zugriff. Allen anderen Benutzer muss der Administrator die entsprechenden Rechte geben.

Zugriffsrechte für alle Tabellen im Schema

```
Postgres =# GRANT SELECT
            ON ALL TABLES IN SCHEMA vertrieb
            TO buchhaltung;
```

- Dem Befehl `GRANT` folgt eine Beschreibung der Zugriffsrechte. `SELECT` erlaubt einen lesenden Zugriff auf Tabellen.
- Der Befehl `ON` folgt das Schlüsselwort `ALL TABLES IN SCHEMA`. Die Zugriffsrechte werden für alle Tabellen in einem bestimmten Schema vergeben.
- Dem Schlüsselwort `TO` folgen die Namen der Rollen, für die Zugriffsrechte gelten sollen.

Zugriffsrechte für Tabellen im Schema

```
Postgres =# GRANT UPDATE
            ON vertrieb.Kunde_Firma TO buchhaltung;
```

- Der Befehl `GRANT` vergibt Zugriffsrechte für Tabellen.
- Das Zugriffsrecht `UPDATE` erlaubt eine Änderung der Datensätze in der Tabelle `Kunde_Firma` in dem Schema `vertrieb`.
- Dem Schlüsselwort `TO` folgen die Namen der Rollen, für die Zugriffsrechte gelten sollen.

Zugriffsrechte für Datenfelder in Tabellen

```
Postgres =# GRANT UPDATE(darfbestellen)
            ON vertrieb.Kunde_Firma TO buchhaltung;
```

- Der Befehl `GRANT` vergibt Zugriffsrechte für Tabellen.
- Das Zugriffsrecht `UPDATE` erlaubt eine Änderung des Datenfeldes `darfbestellen` in der Tabelle `Kunde_Firma` in dem Schema `vertrieb`.
- Dem Schlüsselwort `TO` folgen die Namen der Rollen, für die Zugriffsrechte gelten sollen.

Entziehung von allen Zugriffsrechten

```
Postgres =# REVOKE ALL ON vertrieb.bestellung
            FROM buchhaltung;
```

- Der Befehl `REVOKE` entzieht Zugriffsrechte für Tabellen.
- In diesem Beispiel werden alle (`ALL`) Zugriffsrechte für die Benutzergruppe `buchhaltung` für die Tabelle `Kunde_Firma` in dem Schema `vertrieb` entzogen.

Entziehung von bestimmten Zugriffsrechten

```
Postgres =# REVOKE SELECT ON vertrieb.Land
            FROM buchhaltung;
```

- Dem Befehl `REVOKE` kann eine Liste von Zugriffsrechten folgen, die der Benutzergruppe entzogen werden sollen.