UNIVERSITÄT DES SAARLANDES

Saarbrücken, 18.06.2015
Information Systems Group

Vorlesung „Informationssysteme“

Vertiefung Kapitel 7: Zugriffskontrolle

Erik Buchmann (buchmann@cs.uni-saarland.de)

Foto: M. Strauch

Aus den Videos wissen Sie...

- ...wie man Postgres installiert und einsetzt
  - Schema erzeugen, Daten manipulieren, Anfragen durchführen
- …wie SQL funktioniert
  - Deklarative Sprache, Umsetzung von Sprachkonstrukten auf Operatoren ist Sache des DBMS

- Vertiefung heute:
  - Zugriffskontrolle in SQL im Allgemeinen und Postgres im Besonderen
  - Rollenhierarchien zur strukturierten Rechtevergabe

Role-Based Access Control

Foto: M. Strauch

Motivation

- In einer DB werden häufig sensible Daten gespeichert
  - Betriebsgeheimnisse, persönliche Daten, Kundendaten
- DB ist kritische Infrastruktur
  - Totalausfall kann das Ende des Unternehmens bedeuten

- DBMS muss verhindern, dass Unberechtigte Einsicht bekommen
  - Anfragen auf die Daten selbst
- DBMS muss verhindern, dass Unberechtigte Änderungen vornehmen
  - Datenbanken und Datenbankschemata anlegen, ändern, löschen
  - Indexe, Sichten, Zugriffskontrollmechanismen anlegen, ändern, löschen

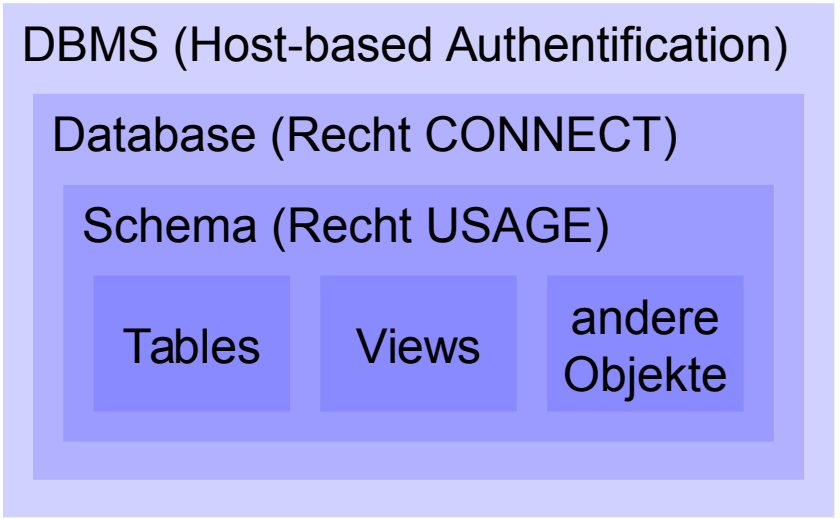
Die Frage der Granularität

- Je detaillierter die Zugriffsregeln, desto aufwändiger ist die Auswertung
  - Zugriff auf die gesamte DB?
  - Zugriff auf eine Datenbank in der DB?
  - Zugriff auf eine Relation in einer Datenbank?
  - Zugriff auf ein Tupel in einer Relation?
  - Zugriff auf ein Attribut in einem Tupel?

- Standard-SQL
  - Zugriffsregeln auf Relationenebene
  - Recht auf Systemzugriff, Schemaänderungen auf Datenbank- und Systemebene
  - Feingranulare Zugriffsregeln über Views
    - CREATE VIEW v AS (SELECT * FROM mitarbeiter WHERE gehalt < 1000);
      GRANT SELECT ON v TO rolle;

Zugriff auf Datenobjekte

- Damit ein Nutzer („Rolle“) auf Datenobjekt zugreifen kann:
  1) Authentifizierung im System erfolgreich (Host-based Auth.)
  2) Rolle hat CONNECT-Recht für die Datenbank
  3) Rolle hat USAGE-Recht auf das Schema, das das Datenobjekt enthält
  4) Rolle ist Besitzer vom Datenobjekt, oder hat Zugriffsrechte darauf

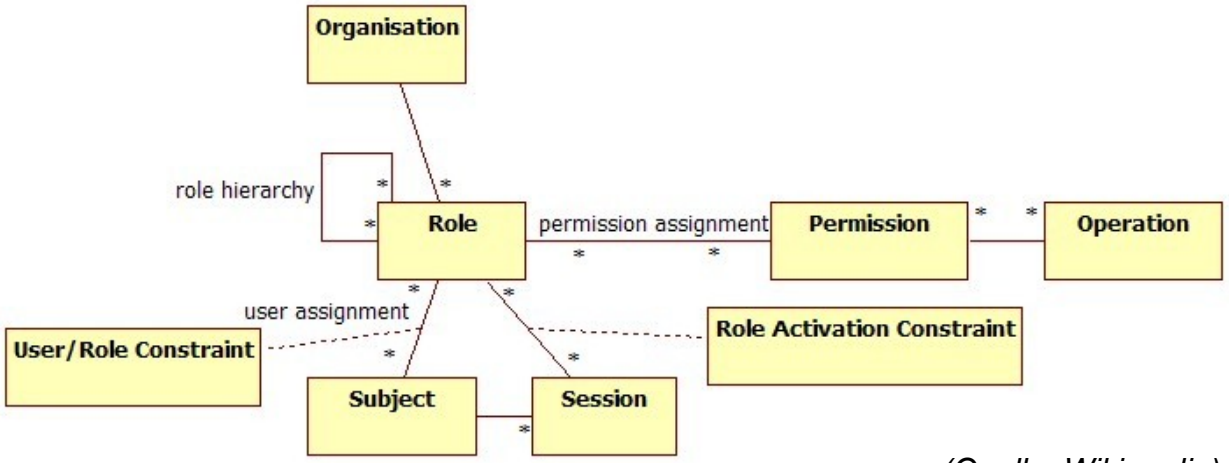


- Host-based Authentification
  - In Postgres pg_hba.conf: Welche Rolle darf sich von wo aus einloggen?

```
# TYPE   DATABASE    USER       ADDRESS                 METHOD
local    all         all                                trust
host     all         all        127.0.0.1/32            trust
local    db1,db2     all                                md5
host     postgres    all        192.168.12.10/32        md5
```

Role-Based Access Control

- „Rolle“ steht synonym für „User“!
- Eine Rolle kann in einer Rollen-Hierarchie angeordnet sein
- Eine Rolle hat bestimmte Rechte auf bestimmte Datenobjekte
- Eine Rolle darf bestimmte Rechte auf andere Rollen übertragen



(Quelle: Wikimedia)

Rollen in Postgres

- Rolle anlegen, ändern, löschen
  - CREATE ROLE name [[WITH] option ...]
  - ALTER ROLE name [[WITH] option ...]
  - DROP ROLE [IF EXISTS] name

- Optionen

(NO)SUPERUSER	Superuser darf alles, keine Rechteprüfung
(NO)CREATEDB	neue DB anlegen
(NO)CREATEROLE	neue Rolle anlegen
(NO)LOGIN	in das System einloggen
PASSWORD 'password'	Passwort für Rollen, die sich einloggen dürfen
VALID UNTIL 'timestamp'	Gültigkeitsdauer des Passworts

Beispiele

- CREATE ROLE verkauf WITH PASSWORD 'bla123' CREATEDB LOGIN;
  - Erzeuge Rolle „verkauf“
  - verkauf darf sich mit Passwort „bla123“ am System anmelden
  - verkauf hat das Recht, Datenbanken anzulegen
- CREATE ROLE einkauf NOLOGIN;
  - Erzeuge Rolle „einkauf“
  - Rolle darf sich nicht einloggen, kann aber Zugriffsrechte erhalten
  - Wichtig für Rollenhierarchien, wird noch erläutert
- ALTER ROLE verkauf WITH NOCREATEDB;
  - Ändere Rolle so, dass keine Datenbanken mehr angelegt werden dürfen

Spezielle Rollen

- SUPERUSER
  - Bei frisch installiertem Postgres ist die Rolle „postgres“ Superuser
  - Außerhalb der Rechteverwaltung, d.h., alles außer Systemtabellen löschen
  - Superuser-Rechte können mit CREATE ROLE, ALTER ROLE jeder Rolle zugewiesen/entzogen werden
    - z.B. ALTER ROLE dbadmin SUPERUSER;
- PUBLIC
  - Implizite Gruppenrolle, zu der jede andere Rolle gehört
  - Nützlich, um beispielsweise Relationen für alle sichtbar zu machen
    - z.B. GRANT SELECT ON TABLE katzenfutter TO PUBLIC;

GRANT und REVOKE

- Rechte zuweisen
  - GRANT (ALL PRIVILEGES | recht1, recht2, … recht n)
    ON obj TO (PUBLIC | rolle) [WITH GRANT OPTION]
    - WITH GRANT OPTION: Rolle darf Recht an andere weitergeben
- Rechte entziehen
  - REVOKE (ALL PRIVILEGES | recht1, recht2, … recht n)
    ON obj FROM (PUBLIC | rolle) [CASCADE | RESTRICT]
    - CASCADE, RESTRICT: Betrifft Rollenhierarchien, wird noch erläutert
- Rollen mit Rechte-Abhängigkeiten dürfen nicht gelöscht werden
  - Lösung
    REVOKE ALL PRIVILEGES FROM rolle;
    DROP ROLE rolle;

Rechte

- Rechte auf Datenobjekte
  - SELECT [spalte1, spalte2, … spalte n]
  - INSERT [spalte1, spalte2, … spalte n]
  - UPDATE [spalte1, spalte2, … spalte n]
  - DELETE und TRUNCATE
- Rechte auf Schema-Ebene
  - CREATE (Datenbanken, Schemata, Tabellen anlegen)
  - REFERENCES [spalte1, spalte2, … spalte n]
  - *Achtung, DROP, ALTER (Table, Database) darf nur Ersteller und SUPERUSER, Recht kann nicht verliehen werden!*
- Rechte auf Datenbank-Ebene *(vgl. Folie 6)*
  - CONNECT (Verbindung zur DB aufbauen)
  - USAGE (Auf das Relationenschema zugreifen)
  - EXECUTE (DB-Funktionen ausführen)

Beispiele

[Person]: { [Name] }
[Anschrift]: { [AID, Name, Straße] }
[Katzenfutter]: { [KID, Name, Preis] }
[Trockenfutter]: { [ID, Pelletgröße, KID] }
[Naßfutter]: { [ID, Wasseranteil, KID] }
[kauft]: {KID, Name}

Rollen: einkauf, verkauf

- GRANT ALL PRIVILEGES
  ON Person TO einkauf;
- GRANT DELETE ON Anschrift
  TO verkauf;
- GRANT DELETE, SELECT ON kauft TO verkauf WITH GRANT OPTION;
- GRANT SELECT (ID, Wasseranteil), UPDATE (Wasseranteil)
  ON Naßfutter TO einkauf, verkauf;
- REVOKE SELECT ON Naßfutter FROM einkauf;
- REVOKE SELECT (ID, Wasseranteil) ON Naßfutter FROM einkauf;

Automatisch zugewiesene Rechte

- Erzeuger einer Relation
  - Alle Operationen zur Datenmanipulation
    - INSERT, UPDATE, DELETE, SELECT, etc.
  - Alle Operationen zur Schemamanipulation
    - ALTER TABLE, DROP TABLE, etc.
  - Wenn Rolle mit WITH GRANT OPTION
    - Alle eigenen Rechte zur Datenmanipulation auf selbsterstellter Relation an andere Rollen weitergeben
    - Kein Recht auf ALTER TABLE, DROP TABLE etc. vergebbar
- Rolle, der der Erzeuger einer Relation angehört
  - Dieselben Rechte wie der Erzeuger
- SUPERUSER
  - Darf alles außer Systemtabellen löschen und ändern

Keine Unter-/Obermengenbeziehungen von Rechten

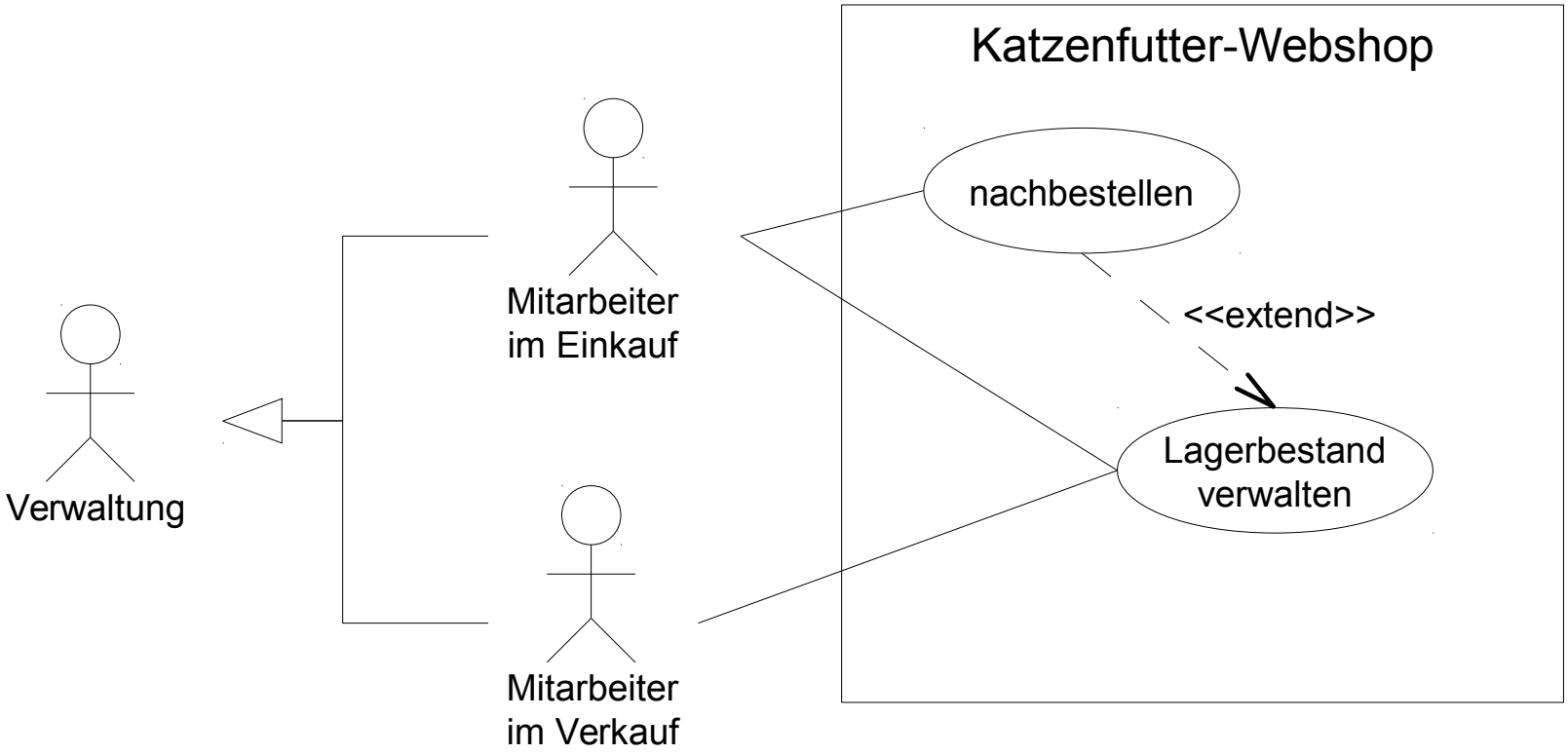
- Keine implizite Differenz von Rechten
  - Funktioniert nicht
    1. GRANT SELECT ON kundendaten TO schulze
    2. REVOKE SELECT(name) ON kundendaten FROM schulze;
    - keine Auswirkungen auf schulze, obwohl Recht SELECT(name) Teilmenge von SELECT
  - Funktioniert
    1. GRANT SELECT(name, str, plz, ort, tel) ON kundendaten TO schulze
    2. REVOKE SELECT(name) ON kundendaten FROM schulze;
    - Recht SELECT(name) erfolgreich entzogen
- Keine Vereinigung von Rechten
  - Das sind unterschiedliche Rechte
    1. GRANT SELECT ON kundendaten TO schulze;
    2. GRANT SELECT(name, str, plz, ort, tel) ON kundendaten TO schulze;
    3. REVOKE SELECT(name) ON kundendaten FROM schulze;
    - keine Auswirkungen auf schulze

Rollenhierarchien

Foto: M. Strauch

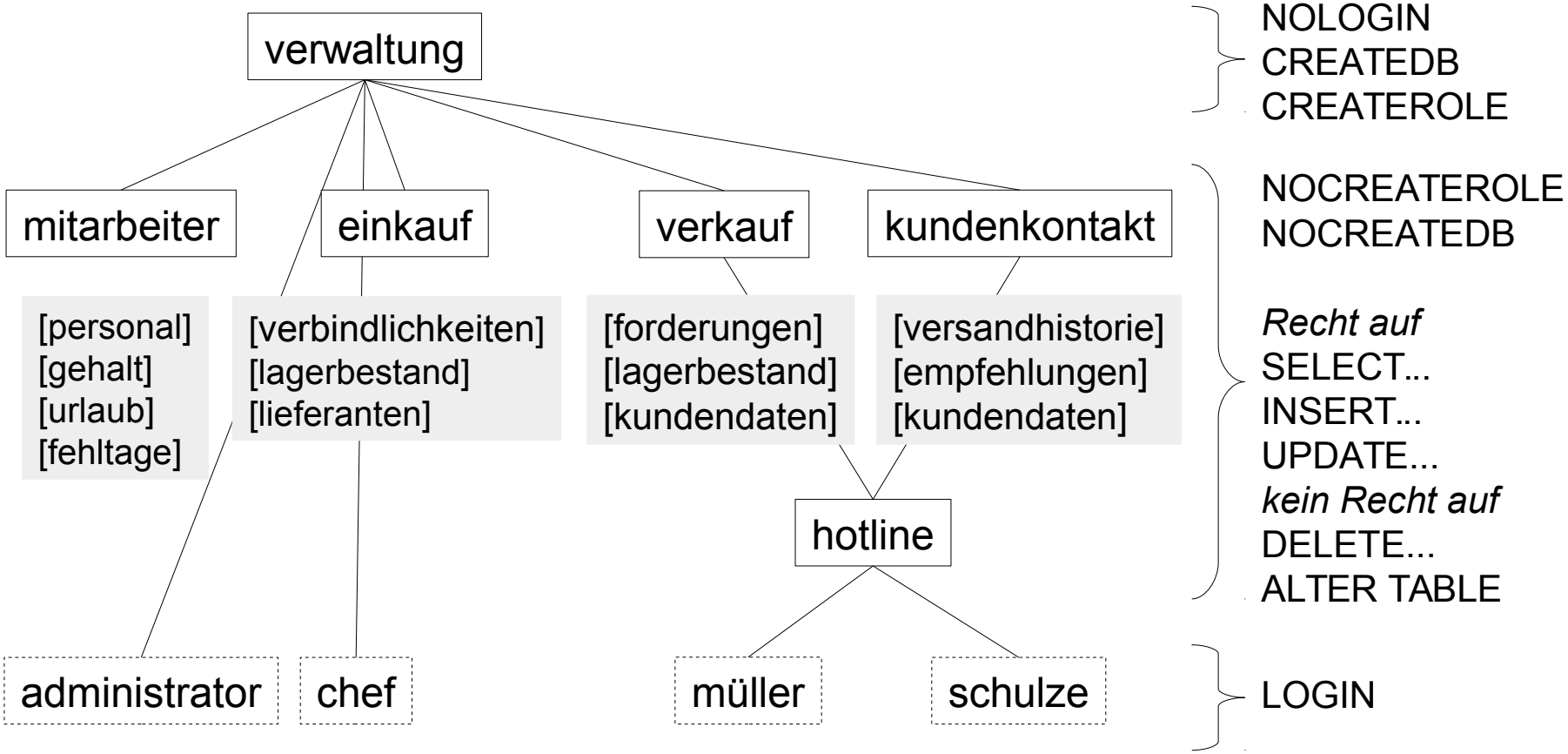
Schnittstelle zum Anwendungsfalldiagramm

- Generalisierungshierarchien zwischen Akteuren im Anwendungsfalldiagramm abbilden



Strukturierte Teilnehmerhierarchien

- statt fehleranfälliger Einzelvergabe von Rechten



Rollenhierarchien über Rechteweitergabe

- Wenn Rolle ein Recht WITH GRANT OPTION hat, darf dieses Recht an andere weitergegeben werden
  - Oder eine Teilmenge dieser Rechte

- Beispiel
  - **Nutzer a**
    CREATE TABLE r (…);
    GRANT ALL PRIVILEGES ON r TO **b** WITH GRANT OPTION;
  - **Nutzer b**
    GRANT SELECT, INSERT ON r TO **c** WITH GRANT OPTION;
  - **Nutzer c**
    GRANT SELECT ON r TO **d**;

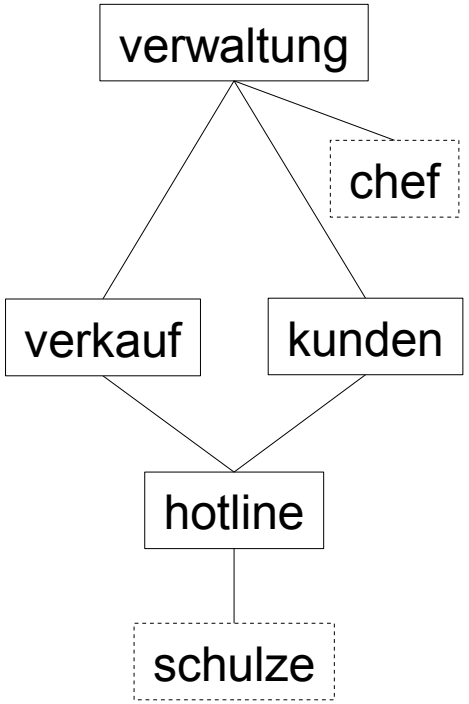
Rollenhierarchien über Vererbung

- Können beim Anlegen oder Ändern einer Rolle spezifiziert werden
  - CREATE ROLE name [[WITH] option ...]
  - ALTER ROLE name [[WITH] option ...]

- Option für Rollenhierarchien
  - (NO)INHERIT
    Rolle erbt alle Rechte von den Rollen, in denen sie Mitglied ist
  - IN ROLE existierende rolle 1, [existierende rolle 2....]
    neue/geänderte Rolle wird Mitglied der existierenden Rolle(n)

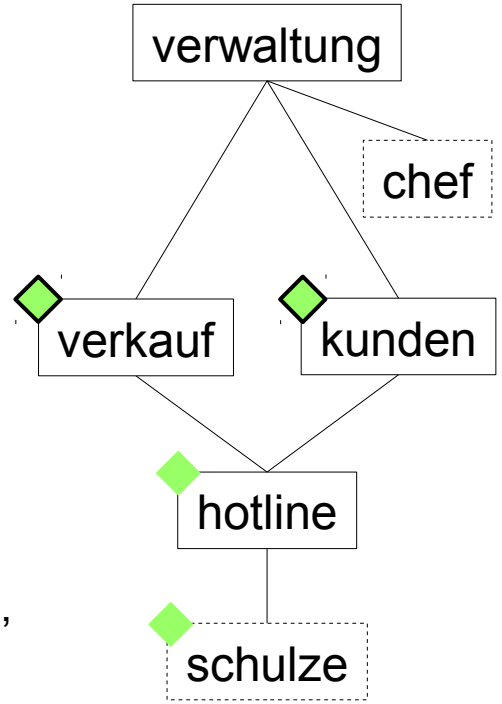
- Beispiel
  CREATE ROLE buchhaltung CREATEDB NOLOGIN;
  CREATE ROLE personal LOGIN INHERIT IN ROLE buchhaltung;
  - „buchhaltung“ ist eine Gruppenrolle, von der andere Rollen Rechte erben
  - „personal“ hat alle Rechte von „buchhaltung“ plus eigene Rechte

Laufendes Beispiel (1/2)



- Rechtevererbung
  - CREATE ROLE verwaltung NOLOGIN CREATEDB CREATEROLE;
  - CREATE ROLE verkauf INHERIT IN ROLE verwaltung NOCREATEDB NOCREATEROLE;
  - CREATE ROLE kunden INHERIT IN ROLE verwaltung NOCREATEDB NOCREATEROLE;
  - CREATE ROLE hotline INHERIT IN ROLE verkauf, kunden;
  - CREATE ROLE chef LOGIN PASSWORD '123' INHERIT IN ROLE verwaltung;
  - CREATE ROLE schulze LOGIN PASSWORD '321' INHERIT IN ROLE hotline;

Laufendes Beispiel (2/2)



Als Rolle chef:

- CREATE TABLE kundendaten (…);
  - kundendaten gehört chef,
    Alle Mitglieder von verwaltung dürfen
    kundendaten einsehen, verändern
- GRANT INSERT, UPDATE, SELECT
  ON kundendaten TO verkauf, kunden;
  - schulze erbt das Recht (◇) auf kundendaten zuzugreifen,
    darf sie aber nicht löschen

Experiment:

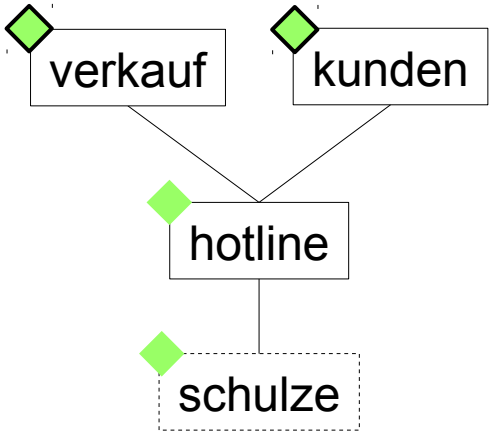
- REVOKE ALL PRIVILEGES ON kundendaten FROM verkauf;
  - schulze kann weiter auf kundendaten zugreifen
- REVOKE ALL PRIVILEGES ON kundendaten FROM hotline;
  - schulze kann noch auf kundendaten zugreifen → Warum?
- REVOKE ALL PRIVILEGES ON kundendaten FROM hotline CASCADE;
  - schulze kann immernoch zugreifen → Warum?

Entziehen von Zugriffsrechten

- REVOKE rechte ON obj FROM rolle [CASCADE | RESTRICT]
  ist wesentlich weniger einfach als es klingt!
  - Eine Rolle kann das Recht haben, Rechte weiterzugeben
    GRANT … WITH GRANT OPTION
  - Eine Rolle kann Rechte von anderen Rollen erben
    und an andere weitervererben
    CREATE ROLE … INHERIT … IN ROLE …

- Erwartung: Entzug eines Rechts führt unmittelbar zum Verlust der damit verbundenen Möglichkeiten
- Aber:
  - Es kann sein, dass Rechteinhaber das Recht ***auch*** aus anderer Quelle hat
  - CASCADE betrifft ***nur*** mit GRANT weitergegebene Rechte
  - Mit INHERIT ererbte Rechte können ***nicht*** „unterwegs“ gelöscht werden
  - Keine Mengenbeziehungen bei Entzug und Vergabe von Rechten

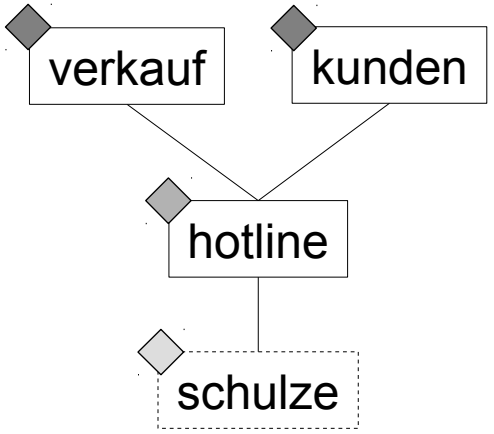
Vererbte und weitergebene Rechte



- Bei Vererbung über INHERIT
  - schulze verliert Zugriff auf kundendaten
    REVOKE ALL PRIVILEGES ON kundendaten
    FROM verkauf, kunden;
  - REVOKE … FROM hotline hat keine Auswirkungen

- Bei Weitergabe über WITH GRANT OPTION

  verkauf: GRANT SELECT ON kundendaten TO hotline WITH GRANT OPTION
  kunden: GRANT SELECT ON kundendaten TO hotline WITH GRANT OPTION
  hotline: GRANT SELECT ON kundendaten TO schulze
  - schulze verliert Zugriff auf kundendaten
    verkauf: REVOKE ALL PRIVILEGES ON kundendaten
    FROM hotline CASCADE;
    kunden: REVOKE ALL PRIVILEGES ON kundendaten
    FROM hotline CASCADE;
  - Ohne CASCADE und wenn nicht beide Rechtegeber Rechte zurückziehen ändert sich für schulze nichts



Zum Abschluss

Foto: M. Strauch

Wie geht es weiter?

- bis Montag, 22.06., 12 Uhr
  - Quiz: Schemaoperationen in SQL
- Dienstag, 23.06., GHH 12-14 Uhr: Tutoriumstermin
  - kurze Besprechung von Aufgabenblatt 7
  - nächstes Aufgabenblatt: komplexe Anfragen in SQL
- Donnerstag, 25.06.: Präsenztermin
  - Noch mehr interessante Problemstellungen in SQL