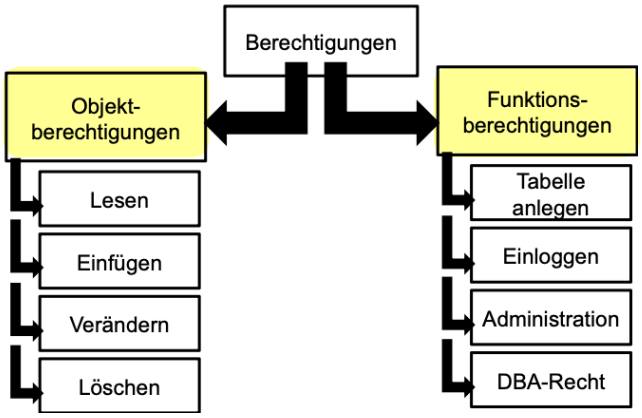
Kap 6: Sicherheit in DBMS:

Autorisierung: Wer darf was?

Motivation



Daten vor unberechtigtem Zugriff schützen

- Firmengeheimnisse schützen
- Kundendaten sichern
- Innerhalb der Firma, z.B. Gehälter

Daten vor Löschen und Verfälschungen schützen

- Manipulationen / Sabotage
- Löschung / „Daten-Gau“

Rahmenbedingungen

- Gesetzliche Regeln zum Datenschutz
- Compliance-Regeln, bspw. PCI-DSS (Payment Card Industry Data Security Standard)

Objektberechtigungen

Tabellen \ Personen	Müller (Angestellter)		Maier (Chefin)		Schulze (Kunde)	
Mitarbeiter (u. Gehälter)	☒ Lesen	☐ Einfügen	☒ Lesen	☒ Einfügen	☐ Lesen	☐ Einfügen
	☐ Ändern	☐ Löschen	☒ Ändern	☒ Löschen	☐ Ändern	☐ Löschen
Kunden	☒ Lesen	☒ Einfügen	☒ Lesen	☒ Einfügen	☒ Lesen	☐ Einfügen
	☒ Ändern	☒ Löschen	☒ Ändern	☒ Löschen	☒ Ändern	☐ Löschen
Bestellungen	☒ Lesen	☒ Einfügen	☒ Lesen	☒ Einfügen	☒ Lesen	☒ Einfügen
	☒ Ändern	☒ Löschen	☒ Ändern	☒ Löschen	☒ Ändern	☒ Löschen

Lesen — SELECT
delete / ändern — UPDATE
einfügen — INSERT
EXECUTE

Vergabe von Objektberechtigungen

`GRANT <Right> ON <Object> TO <User>;`

- `GRANT SELECT ON MITARBEITER TO MAIER;`
- `GRANT INSERT ON MITARBEITER TO MAIER;`
- `GRANT SELECT, UPDATE, INSERT, DELETE ON MITARBEITER TO MAIER;`
- `GRANT UPDATE ON MITARBEITER TO SCHULZE;`

  `REVOKE <Right> ON <Object> FROM <User>;`
- `REVOKE UPDATE ON MITARBEITER FROM SCHULZE;`

Rechte durch Views einschränken

ID	Name	Abteilung	Gehalt
1	Müller	Verkauf	30.000 €
2	Huber	Einkauf	45.000 €
3	Schmidt	Marketing	70.000 €
4	Fischer	Verkauf	50.000 €
5	Weber	Marketing	65.000 €

```
CREATE VIEW
V_MITARBEITER AS
SELECT
ID, NAME, ABTEILUNG
FROM MITARBEITER
```

View V_MITARBEITER

ID	Name	Abteilung
1	Müller	Verkauf
2	Huber	Einkauf
3	Schmidt	Marketing
4	Fischer	Verkauf
5	Weber	Marketing

```
REVOKE SELECT
ON MITARBEITER FROM MUELLER

GRANT SELECT
ON V_MITARBEITER TO MUELLER
```

Lesecechte einschränken?

Rechte einschränken: Stored Procedures

ID	Kunde	Artikel	Status
1	Lange	Schuhe	Ausgeliefert
2	Bauer	Hose	Bestellt
3	Krause	Hemd	Bestellt
4	Lange	Jacke	Bestellt
5	Weber	Hut	Ausgeliefert

```
CREATE PROCEDURE
LOESCHE_BESTELLUNG(p_id)
IS
BEGIN
  DELETE
    FROM BESTELLUNGEN
    WHERE ID = p_id
    AND Kunde =
    <Aktueller Benutzer>
    AND Status =
    'Bestellt';
END;
```

```
REVOKE DELETE
ON BESTELLUNGEN FROM SCHULZE

GRANT EXECUTE
ON LOESCHE_BESTELLUNG
TO SCHULZE
```

Mit welchen Rechten wird eine Stored Procedure ausgeführt?

Default-Einstellung:

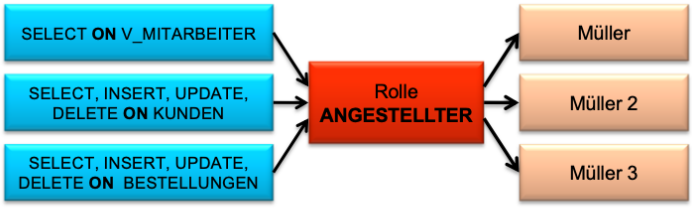
- Rechte der Benutzerin, die die SP angelegt hat

Mögliche Definition:

- "Invoker Rights"
- Dann läuft die SP mit den Rechten des aufrufenden Benutzers

Rollen

- Die Berechtigungen eines Nutzers leiten sich aus seiner Rolle ab.
- Typischerweise haben alle Nutzer, die die gleiche Rolle ausfüllen, auch die gleichen Rechte.
- Daher macht es Sinn, diese Rollen explizit anzulegen.
- Eine Person kann auch mehrere Rollen haben.



Zuweisung von Rechten über Rollen

Schritt 1: Rolle erzeugen

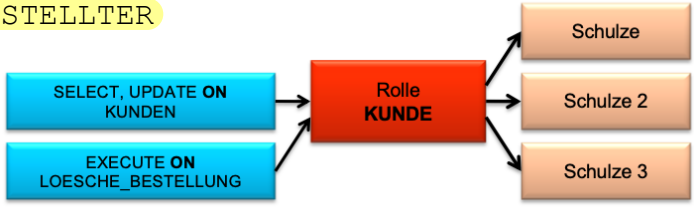
- `CREATE ROLE ANGESTELLTER`

Schritt 2: Recht an Rolle vergeben

- `GRANT SELECT ON V_MITARBEITER TO ANGESTELLTER`

Schritt 3: Rolle an Mitarbeiter vergeben

- `GRANT ANGESTELLTER TO MUELLER`
- `GRANT ANGESTELLTER TO MUELLER2`



Weitergabe von Rechten

Darf ich ein Recht, welches ich erhalten habe, an andere Nutzer weitergeben?
Dies bestimmt die sog. "Grant Option"
Administrator:

- `GRANT SELECTON KUNDEN TO MEIER WITH GRANT OPTION;`

Meier:

- `GRANT SELECTON KUNDEN TO MUELLER;`

rechte weitergeben?

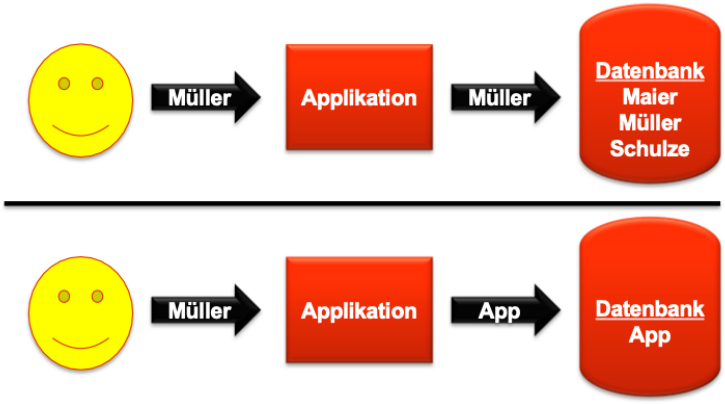
Authentifizierung: Wer bin ich?

Thema Authentifikation: Wie identifiziere ich mich?

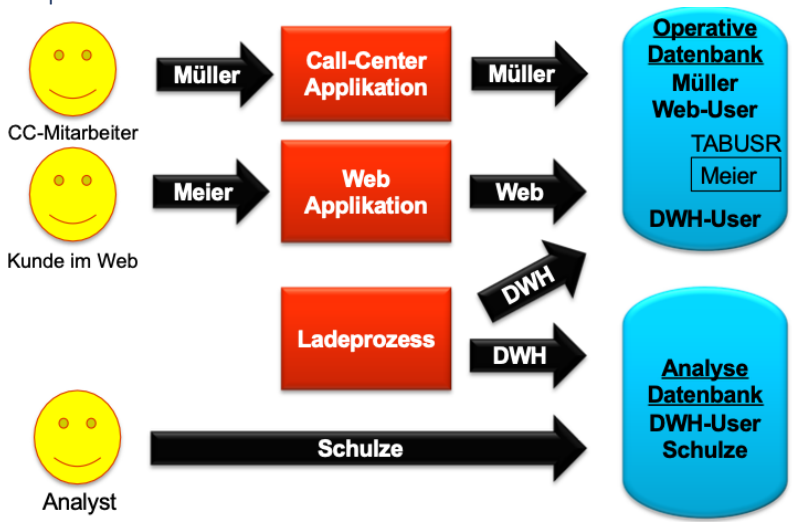
Klassisch: Username / Passwort schwache Authentifikation

Alternativen: z.B. digitale Zertifikate starke Authentifikation

Architektur



Beispiel-Architektur



Auditing: Wer hat was gemacht?

Bestimmte Änderungen müssen nachvollziehbar sein

- Wer hat das Gehalt eines Mitarbeiters geändert?

Bestimmte Zugriffe müssen protokolliert werden

- Wer hat personenbezogene Daten abgefragt?
- Dies ist z.T. gesetzlich vorgeschrieben

Technisch mit verschiedenen Ansätzen:

- Logik in der Anwendung:
  - Selbst programmiert
- Logik in der Datenbank:
  - Selbst programmiert über Datenbank-Prozeduren / Trigger
  - Vollautomatisch durch Datenbank-Auditing
  - Problem bei technischen DB-Nutzern
- **Organisatorisch: Log-Buch führen**
- Vollständigkeit? Missbrauch?

Sicherheitslücken vermeiden: Wie stelle ich das sicher?

- Niemand außer DBAs darf direkten Datenzugriff haben
  - Datenfiles schützen über System-Zugriffsrechte
- Backup-Dateien und Tapes sichern
- Netzwerkzugriff auf die Datenbank absichern
  - Eine Datenbank ist normalerweise nur von innen erreichbar
- System aktuell halten: Updates

Sicherheitsprobleme (Beispiele)

Beispiel: Bekannte Default-Passwörter

- Bei Oracle z.B. sys/change_on_install

SQL Injection

Applikation testet Login selbst über Benutzer-Tabelle

- ```
  query =
  "select name from benutzer where " +
  "name = '" + username + "' and
  "passwort = '" + password + "'";
  ```
- username ist "musterfrau"
- passwort ist "pwmusterfrau"

query enthält dann

- ```
  select name from benutzer where name = 'musterfrau'
  and passwort = 'pwmusterfrau'
  ```

Mit welche Eingabe für "username" und "passwort" knackt man dieses Programm?

Zusammenfassung

- Sicherheit wichtiges Thema bei Datenbanken
- Anforderungen an die Sicherheit ermitteln
- Entsprechendes Konzept erarbeiten
- Passende Architektur erarbeiten
- Zusammenarbeit mit anderen Experten
  - Netzwerksicherheit
  - Rechnersicherheit
  - Datenbankadministratoren
- Sicherheit in der Anwendung:
  - Beispiel SQL Injection