Teil X

Sichten und Zugriffskontrolle

Sichten und Zugriffskontrolle

Lernziele für heute ...

- Verständnis des Sichtenkonzeptes von Datenbanken
- Kenntnisse zur Formulierung und Nutzung von Sichten in SQL
- Verständnis der Probleme bei Änderungen über Sichten
- Verständnis zu Datenschutzaspekten im Zusammenhang mit aggregierten/statistischen Daten

Sichten

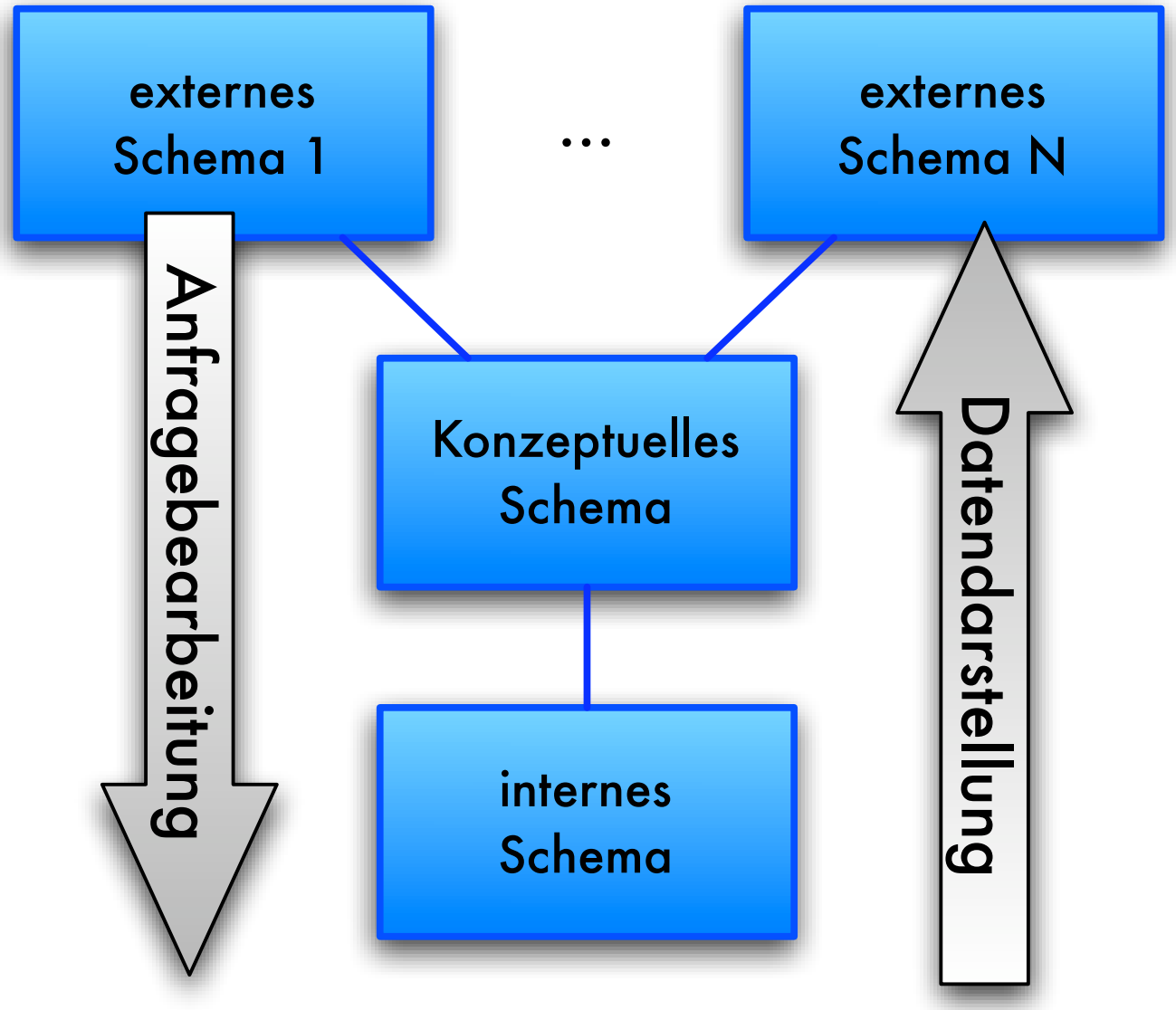
Sichten: virtuelle Relationen (bzw virtuelle Datenbankobjekte in anderen Datenmodellen) (englisch view)

- Sichten sind externe DB-Schemata folgend der 3-Ebenen-Schemaarchitektur
- Sichtdefinition
  - Relationenschema (implizit oder explizit)
  - Berechnungsvorschrift für virtuelle Relation, etwa SQL-Anfrage

Sichten /2

- Vorteile
  - Vereinfachung von Anfragen für den Benutzer der Datenbank, etwa indem oft benötigte Teilanfragen als Sicht realisiert werden
  - Möglichkeit der Strukturierung der Datenbankbeschreibung, zugeschnitten auf Benutzerklassen
  - logische Datenunabhängigkeit ermöglicht Stabilität der Schnittstelle für Anwendungen gegenüber Änderungen der Datenbankstruktur (entsprechend in umgekehrter Richtung)
  - Beschränkung von Zugriffen auf eine Datenbank im Zusammenhang mit der Zugriffskontrolle
- Probleme
  - automatische Anfragetransformation
  - Durchführung von Änderungen auf Sichten

Drei-Ebenen-Schema-Architektur



Definition von Sichten in SQL

```
create view SichtName [ SchemaDeklaration ]
as SQLAnfrage
[ with check option ]
```

Sichten - Beispiel

- alle Rotweine aus Bordeaux:

```
create view Rotweine as
   select Name, Jahrgang, WEINE.Weingut
   from WEINE natural join ERZEUGER
   where Farbe = 'Rot'
         and Region = 'Bordeaux'
```

Problembereiche bei Sichten

- Durchführung von Änderungen auf Sichten
- automatische Anfragetransformation

Kriterien für Änderungen auf Sichten

- **Effektkonformität**
  Benutzer sieht Effekt als wäre die Änderung auf der Sichtrelation direkt ausgeführt worden
- **Minimalität**
  Basisdatenbank sollte nur minimal geändert werden, um den erwähnten Effekt zu erhalten
- **Konsistenzerhaltung**
  Änderung einer Sicht darf zu keinen Integritätsverletzungen der Basisdatenbank führen
- **Respektierung des Datenschutzes**
  Wird die Sicht aus Datenschutzgründen eingeführt, darf der bewusst ausgeblendete Teil der Basisdatenbank von Änderungen der Sicht nicht betroffen werden

Projektionssicht

$$\texttt{WNW} := \pi_{\texttt{WeinID,Name,Weingut}}(\texttt{WEINE})$$

- In SQL mit **`create view`**-Anweisung:

```
create view WNW as
   select WeinID, Name, Weingut from WEINE
```

- Änderungsanweisung für die Sicht `WNW`:

```
insert into WNW values (3333, 'Dornfelder', 'Müller')
```

- Korrespondierende Anweisung auf der Basisrelation `WEINE`:

```
insert into WEINE
   values (3333, 'Dornfelder', null, null, 'Müller')
```

$\rightarrow$ Problem der Konsistenzerhaltung falls `Farbe` oder `Jahrgang` als **`not null`** deklariert!

Selektionssichten

$$\texttt{WJ} := \sigma_{\texttt{Jahrgang}>2000}(\pi_{\texttt{WeinID,Jahrgang}}(\texttt{WEINE}))$$

```
create view WJ as
select WeinID, Jahrgang
from WEINE
where Jahrgang > 2000
```

- Tupelmigration: Tupel WEINE(3456, 'Zinfandel', 'Rot', 2004, 'Helena'), wird aus der Sicht „herausbewegt“:

```
update WJ
set Jahrgang = 1998
where WeinID = 3456
```

Kontrolle der Tupelmigration

```
create view WJ as
select WeinID, Jahrgang
from WEINE
where Jahrgang > 2000
with check option
```

Verbundsichten

$$\texttt{WE} := \texttt{WEINE} \bowtie \texttt{ERZEUGER}$$

- In SQL:

```
create view WE as
select WeinID, Name, Farbe, Jahrgang, WEINE.Weingut,
   Anbaugebiet, Region
from WEINE, ERZEUGER
where WEINE.Weingut = ERZEUGER.Weingut
```

- Änderungsoperationen in der Regel nicht eindeutig übersetzbar:

```
insert into WE
values (3333, 'Dornfelder', 'Rot', 2002, 'Helena',
   'Barossa Valley', 'South Australia')
```

Verbundsichten /2

- Änderung wird transformiert zu

```
insert into WEINE
values (3333, 'Dornfelder', 'Rot', 2002, 'Helena')
```

- plus
  1. Einfügeanweisung auf ERZEUGER:

```
insert into ERZEUGER
values ('Helena', 'Barossa Valley', 'South Australia')
```

  2. oder alternativ:

```
update ERZEUGER
set Anbaugebiet = 'Barossa Valley', Region = 'South Australia'
where Weingut = 'Helena'
```

besser bzgl. Minimalitätsforderung, widerspricht aber Effektkonformität!

Aggregierungssichten

```
create view FM (Farbe, MinJahrgang) as
select Farbe, min(Jahrgang)
from WEINE
group by Farbe
```

- Folgende Änderung ist nicht eindeutig umsetzbar:

```
update FM
set MinJahrgang = 1993
where Farbe = 'Rot'
```

Klassifikation der Problembereiche

1. Verletzung der Schemadefinition (z.B. Einfügen von Nullwerten bei Projektionssichten)
2. Datenschutz: Seiteneffekte auf nicht-sichtbaren Teil der Datenbank vermeiden (Tupelmigration, Selektionssichten)
3. nicht immer eindeutige Transformation: Auswahlproblem
4. Aggregierungssichten (u.a.): keine sinnvolle Transformation möglich
5. elementare Sichtänderung soll genau einer atomaren Änderung auf Basisrelation entsprechen: 1:1-Beziehung zwischen Sichttupeln und Tupeln der Basisrelation (kein Herausprojizieren von Schlüsseln)

Behandlung von Sichten in SQL

- SQL-92-Standard
  - Integritätsverletzende Sichtänderungen nicht erlaubt
  - datenschutzverletzende Sichtänderungen: Benutzerkontrolle (**`with check option`**)
  - Sichten mit nicht-eindeutiger Transformation: Sicht nicht änderbar (SQL-92 restriktiver als notwendig)

Einschränkungen für Sichtänderungen

- änderbar nur Selektions- und Projektionssichten (Verbund und Mengenoperationen nicht erlaubt)
- 1:1-Zuordnung von Sichttupeln zu Basistupeln: kein **`distinct`** in Projektionssichten
- Arithmetik und Aggregatfunktionen im **`select`**-Teil sind verboten
- genau eine Referenz auf einen Relationsnamen im **`from`**-Teil erlaubt (auch kein Selbstverbund)
- keine Unteranfragen mit „Selbstbezug“ im **`where`**-Teil erlaubt (Relationsname im obersten SFW-Block nicht in **`from`**-Teilen von Unteranfragen verwenden)
- **`group by`** und **`having`** verboten

Sichtänderungen in SQL:2003

- seit SQL:2003 Aufhebung einiger Einschränkungen, insbesondere
  - Updates auf **`union all`**-Sichten (ohne Duplikateliminierung)
  - Inserts in Verbundsichten mit Primär-/Fremdschlüsselbeziehungen (mit einigen Einschränkungen)
  - Updates auf Verbundsichten mit Cursor (siehe folgendes Kapitel)

Sichtänderungen mit Instead-of-Triggern

- Definition von Triggern auf Sichten zur anwendungsspezifischen Propagierung der Änderungen auf die Basistabellen

```
create view V_WEINERZEUGER as
      select * from WEINE natural join ERZEUGER

create trigger V_WEINERZEUGER_Insert
      instead of insert on V_WEINERZEUGER
referencing new as N
for each row
begin
   insert into WEINE
      values (:N.WeinID, :N.Name, :N.Farbe,
         :N.Jahrgang, :N.Weingut);
end;
```

Auswertung von Anfragen an Sichten

- **`select`**: Sichtattribute evtl. umbenennen bzw. durch Berechnungsterm ersetzen
- **`from`**: Namen der Originalrelationen
- konjunktive Verknüpfung der **`where`**-Klauseln von Sichtdefinition und Anfrage (evtl. Umbenennungen)

Probleme bei Aggregierungssichten

```
create view FM (Farbe, MinJahrgang) as
select Farbe, min(Jahrgang)
from WEINE
group by Farbe
```

- Anfrage: *Weinfarben mit alten Jahrgängen*

```
select Farbe
from FM
where MinJahrgang < 1995
```

Probleme bei Aggregierungssichten /2

- Nach syntaktischer Transformation:

```
select Farbe
from WEINE
where min(Jahrgang) < 1995
group by Farbe
```

- keine syntaktische korrekte SQL-Anfrage – Korrekt wäre:

```
select Farbe
from WEINE
group by Farbe
having min(Jahrgang) < 1995
```

Probleme bei Aggregierungssichten /3

- Anfrage

```
select max (MinJahrgang)
from FM
```

- müsste wie folgt transformiert werden:

```
select max(min (Jahrgang))
from WEINE
group by Farbe
```

- Aber: Geschachtelte Aggregatfunktionen sind in SQL nicht erlaubt!

Rechtevergabe in Datenbanksystemen

- *Zugriffsrechte*

  `(AutorisierungsID, DB-Ausschnitt, Operation)`

- AutorisierungsID ist interne Kennung eines „Datenbankbenutzers“
- Datenbank-Ausschnitte: Relationen und Sichten
- DB-Operationen: Lesens, Einfügen, Ändern, Löschen

Rechtevergabe in SQL

```
grant <Rechte>
on <Tabelle>
to <BenutzerListe>
[with grant option]
```

Rechtevergabe in SQL /2

- Erläuterungen:
  - In <Rechte>-Liste: **all** bzw. Langform **all privileges** oder Liste aus **select**, **insert**, **update**, **delete**
  - Hinter **on**: Relationen- oder Sichtname
  - Hinter **to**: Autorisierungsidentifikatoren (auch **public**, **group**)
  - spezielles Recht: Recht auf die Weitergabe von Rechten (**with grant option**)

Autorisierung für **public**

```
create view MeineAufträge as
select *
from AUFTRAG
where KName = user;

grant select, insert
on MeineAufträge
to public;
```

„Jeder Benutzer kann seine Aufträge sehen und neue Aufträge einfügen (aber nicht löschen!).“

Zurücknahme von Rechten

```
revoke <Rechte>
on <Tabelle>
from <BenutzerListe>
[restrict | cascade ]
```

- **restrict**: Falls Recht bereits an Dritte weitergegeben: Abbruch von **revoke**
- **cascade**: Rücknahme des Rechts mittels **revoke** an alle Benutzer propagiert, die es von diesem Benutzer mit **grant** erhalten haben

Privacy: Begriff und Anwendungsgebiete

Privacy (Privatsphäre): das Recht jedes Einzelnen auf einen geschützten privaten Raum, der von anderen nur in definierten Ausnahmefällen verletzt werden darf

- elektronische Autobahn-Mautsysteme: Überwachung von Fahrzeugen
- Kreditkartenaktivitäten und diverse Payback- bzw. Rabattkarten: Kaufverhalten von Kunden
- Mobilfunksysteme: Bewegungsprofile der Nutzer
- RFID-Technologie: etwa im Einzelhandel Kaufverhalten, Warenflüsse, etc.

Statistische Datenbanken

- Datenbanken, in denen die Einzeleinträge dem Datenschutz unterliegen, aber statistische Informationen allen Benutzern zugänglich sind
- statistische Information = aggregierte Daten (Durchschnittseinkommen etc.)
- Problem: Gewinnung von Einzelinformationen durch indirekte Anfragen

Statistische Datenbanken: Beispiel

- Beispiel: Benutzer X darf Daten über Kontoinhaber sowie statistische Daten abfragen, jedoch keine einzelnen Kontostände
  1. Verfeinerung des Suchkriteriums (nur ein Kunde wird selektiert)

```
select count (*) from KONTO
where Ort = 'Manebach' and Alter = 24 and ...
```

  2. Name des Kontoinhabers

```
select Name from KONTO
where Ort = 'Manebach' and Alter = 24 and ...
```

  3. statistische Anfrage, die tatsächlich aber einen Einzeleintrag liefert

```
select sum(Kontostand) from KONTO
where Ort = 'Manebach' and Alter = 24 and ...
```

- Abhilfe: keine Anfragen, die weniger als n Tupel selektieren

Statistische Datenbanken: Beispiel /2

- X will Kontostand von Y herausfinden
- X weiss, dass Y nicht in Ilmenau lebt
- X hat abgefragt, dass in Ilmenau mehr als n Kontoinhaber leben
  1. Gesamtkontostand der Ilmenauer Kunden

```
select sum(Kontostand) from KONTO
where Ort = 'Ilmenau'
```

  2. Gesamtkontostand der Ilmenauer Kunden + Kunde Y

```
select sum(Kontostand) from KONTO
where Name = :Y or Ort = 'Ilmenau'
```

  3. Differenz der Ergebnisse liefert Kontostand von Y
- Abhilfe: statistische Anfragen nicht erlauben, die paarweise einen Durchschnitt von mehr als m vorgegebenen Tupeln betreffen

Statistische Datenbanken: Fazit

- kritische Parameter
  - Ergebnisgröße n
  - Größe der Überlappung der Ergebnismengen m

Sind nur Ergebnisse von Aggregatfunktionen erlaubt, dann benötigt eine Person $1 + (n - 2)/m$ Anfragen, um einen einzelnen Attributwert zu ermitteln

k-Anonymität

- für viele Zwecke (klinische Studien etc.) werden auch Detaildaten (Mikrodaten) benötigt

Name	Alter	PLZ	Geschlecht	FamStand	Krankheit
*****	38	98693	männl.	verh.	Schnupfen
*****	29	39114	weibl.	ledig	Fieber
*****	29	39114	weibl.	ledig	Anämie
*****	34	98693	männl.	verh.	Husten
*****	34	98693	männl.	verh.	Knochenbruch
*****	27	18055	weibl.	ledig	Fieber
*****	27	18055	weibl.	ledig	Schnupfen

k-Anonymität: Problem

- ist von einer Person aus dieser Relation bekannt, dass sie
  - männlich
  - 38 Jahre alt
  - verheiratet ist
  - in 98693 Ilmenau wohnt
- $\rightsquigarrow$ Schnupfen
- weitere Zuordnungen (Namen etc.) etwa durch Verbund mit anderen Daten
- Lösung: Data Swapping (??)

k-Anonymität

k-Anonymität: ein bestimmter Sachverhalt kann nicht zwischen einer vorgegebenen Anzahl k von Tupeln unterschieden werden

- eine Anfrage nach einer beliebigen Kombination von Alter, Geschlecht, Familienstand und Postleitzahl liefert entweder eine leere Relation oder mindestens k Tupel

k-Anonymität: Ansätze

- **Generalisierung**: Attributwerte durch allgemeinere Werte ersetzen, die einer Generalisierungshierarchie entnommen sind
  - die Verallgemeinerung des Alters einer Person zu Altersklassen: {35, 39} $\rightsquigarrow$ 30-40
  - Weglassen von Stellen bei Postleitzahlen: { 39106, 39114 } $\rightsquigarrow$ 39***
- **Unterdrücken von Tupeln**: Löschen von Tupeln, welche die k-Anonymität verletzen und damit identifizierbar sind

Kontrollfragen

- Was versteht man unter einer Datenbank-Sicht? Wie werden Sichten definiert?
- Sind Sichten änderbar? Unter welchen Bedingungen?
- Wie kann in Datenbanken der Datenschutz erreicht werden?

Zusammenfassung

- Sichten zur Strukturierung von Datenbanken
- Probleme bei Änderungen über Sichten
- Rechtesystem in SQL-DBS
- Privacy-Aspekte