Sichten

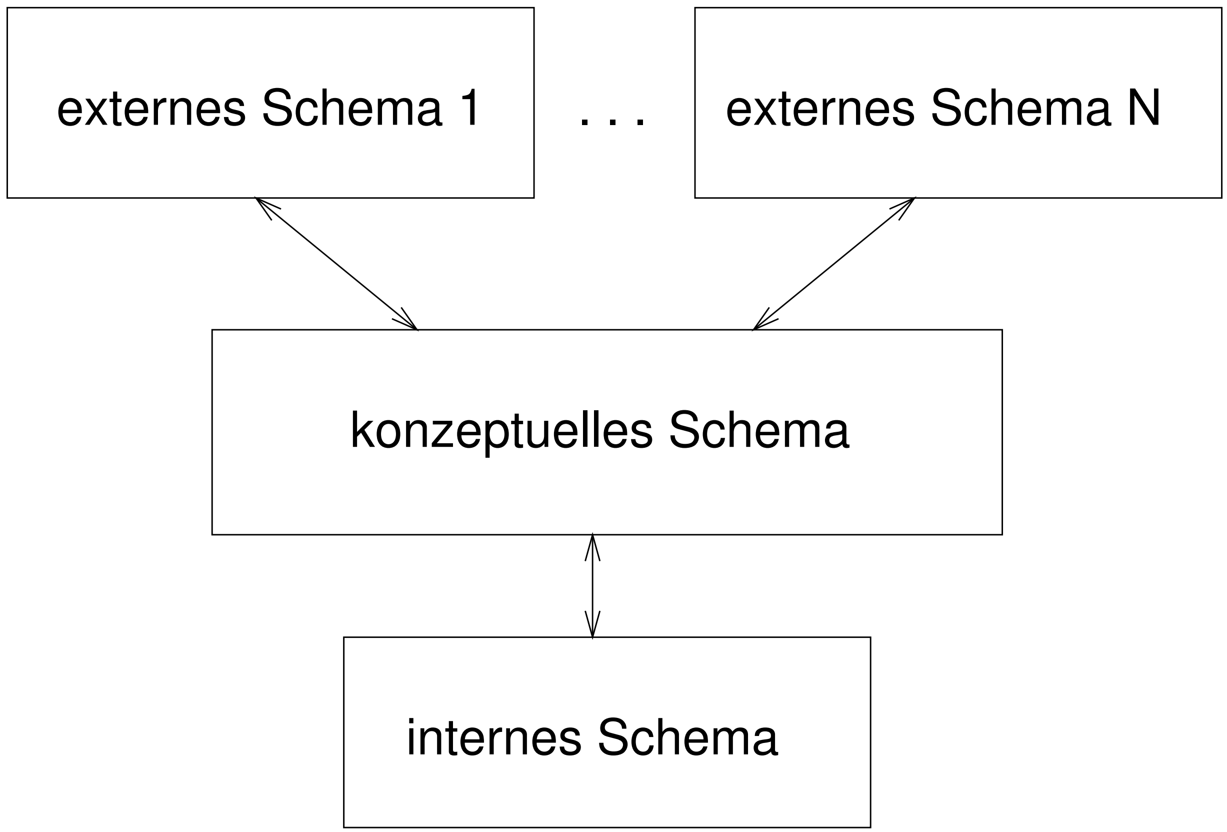
Sichten: *virtuelle Relationen* (bzw virtuelle Datenbankobjekte in anderen Datenmodellen) (englisch *view*)

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

Sichten II

- Vorteile
  - Vereinfachung von Anfragen
  - Strukturierung der Datenbank
  - logische Datenunabhängigkeit (Sichten stabil bei Änderungen der Datenbankstruktur)
  - Beschränkung von Zugriffen (Datenschutz)
- Probleme
  - automatische Anfragetransformation
  - Durchführung von Änderungen auf Sichten

Drei-Ebenen-Schema-Architektur



Definition einer Sicht

Angegebenwerden muß

- *Relationenschema* (explizit, implizit aus Ergebnistyp der Anfrage)
- *Berechnungsvorschrift* (Anfrage) für virtuelle Relation

Definition von Sichten in SQL

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

Vorteile von Sichten

- *Vereinfachung von Anfragen*
- *Strukturierung der Datenbankbeschreibung*
- *logische Datenunabhängigkeit*:

  Stabilität der Anwenderschnittstelle
- *Datensicherheit* / *Datenschutz*

Einsatz von Sichten am Beispiel

Beispielrelation:

```
Prüf( Studienfach, Fach, Student,
      Prüfer, Datum, Note)
```

1. Die Fakultät für Informatik sieht nur die Daten der Informatikstudenten.
2. Das Prüfungsamt sieht alle Daten.
3. Jeder Student darf seine eigenen Daten sehen (aber nicht ändern).
4. Die Kommission für die Vergabe von Promotionsstipendien darf von Studenten die Durchschnittsnote sehen.
5. Der Dekan darf statistische Daten über die Absolventen des letzten Jahrgangs lesen.
6. Die Sekretariate dürfen die Prüfungsdaten der zugehörigen Professoren einsehen.

Problembereiche bei Sichten

- *automatische Anfragetransformation*
- *Durchführung von Änderungen auf Sichten*

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

Kriterien für Änderungen auf Sichten II

- **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*

Beispielszenario im Relationenmodell

```
MGA(Mitarbeiter, Gehalt, Abteilung)
AL(Abteilung, Leiter)
```

`MGA` speichert Daten über Zugehörigkeit von Mitarbeitern zu Abteilungen und deren jeweiliges Gehalt

`AL` gibt für jede Abteilung den Abteilungsleiter an

Projektionssicht

$$\texttt{MA} := \pi_{\texttt{Mitarbeiter},\texttt{Abteilung}}(\texttt{MGA})$$

In SQL mit ‘**create view**’-Anweisung:

create view `MA` **as**
select `Mitarbeiter, Abteilung` **from** `MGA`

Änderungsanweisung für die Sicht `MA`:

insert into `MA` **values** `('Zuse', 'Info')`

Korrespondierende Anweisung auf der Basisrelation `MGA`:

insert into `MGA` **values** `('Zuse',` **null**`, 'Info')`

→ Problem der *Konsistenzerhaltung* falls `Gehalt` als **not null** deklariert!

Selektionssichten

$$\mathtt{MG} := \sigma_{\mathtt{Gehalt}>20}(\pi_{\mathtt{Mitarbeiter},\mathtt{Gehalt}}(\mathtt{MGA}))$$

create view `MG` **as**
select `Mitarbeiter, Gehalt`
from `MGA`
where `Gehalt > 20`

Tupelmigration: Ein Tupel $\mathtt{MGA}('\mathtt{Zuse}', 25, '\mathtt{Info}')$, wird aus der Sicht ‘heraus-bewegt’:

update `MG` **set** `Gehalt = 15` **where** `Mitarbeiter = 'Zuse'`

Kontrolle der Tupelmigration im SQL-Standard

```
create view MG as
select Mitarbeiter, Gehalt
from MGA
where Gehalt > 20
with check option
```

Verbundsichten

$$\texttt{MGAL} := \texttt{MGA} \bowtie \texttt{AL}$$

In SQL:

```
create view MGAL as
select Mitarbeiter, Gehalt,
       MGA.Abteilung, Leiter
from MGA, AL
where MGA.Abteilung = AL.Abteilung
```

Änderungsoperationen in der Regel nicht eindeutig übersetzbar:

```
insert into MGAL values ('Turing',30,'Info','Zuse')
```

Verbundsichten II

Änderung wird transformiert zu

insert into `MGA` **values** `('Turing', 30, 'Info')`

plus

1. Einfügeanweisung auf `AL`:

   insert into `AL` **values** `('Info','Zuse')`

2. oder alternativ:

   update `AL` **set** `Abteilung = 'Info'` **where** `Leiter = 'Zuse'`

   - besser bzgl. *Minimalitätsforderung*
   - widerspricht aber *Effektkonformität!*

Aggregierungssichten

```
create view AS (Abteilung, SummeGehalt)
as
select Abteilung, sum(Gehalt)
from MGA
group by Abteilung
```

Folgende Änderung ist nicht eindeutig umsetzbar:

```
update AS
set SummeGehalt = SummeGehalt + 1000
where Abteilung = 'Info'
```

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)

Besonderheiten der Behandlung von Sichten in SQL

Aktueller Standard SQL-92

- 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 in SQL

- 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

Auswertung von Anfragen an Sichten in SQL

- **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 `DS (Abteilung, GehaltsSumme)` **as**
select `Abteilung,` **sum**`(Gehalt)`
from `MGA`
group by `Abteilung`

Anfrage: *Abteilungen mit hohen Gehaltsausgaben*

select `Abteilung`
from `DS`
where `GehaltsSumme > 500`

Probleme bei Aggregierungssichten II

Nach syntaktischer Transformation:

```
select Abteilung
from MGA
where sum(Gehalt) > 500
group by Abteilung
```

Keine syntaktische korrekte SQL-Anfrage! Korrekt wäre:

```
select Abteilung
from MGA
group by Abteilung
having sum(Gehalt) > 500
```

Probleme bei Aggregierungssichten III

select avg `(GehaltsSumme)`
from `DS`

Diese Anfrage müßte wie folgt transformiert werden:

select avg`(`**sum** `(Gehalt))`
from `MGA`
group by `Abteilung`

Aber: Geschachtelte Aggregatfunktionen sind in SQL nicht erlaubt!