Datenintegrität

Integritätsbedingungen

- Schlüssel
- Beziehungskardinalitäten
- Attributdomänen
- Inklusion bei Generalisierungen

statische Integritätsbedingungen

- Bedingungen an den Zustand der Datenbasis

dynamische Integritätsbedingungen

- Bedingungen an Zustandsübergänge

Referentielle Integrität

Fremdschlüssel

- verweisen auf Tupel einer Relation
- z.B. *gelesenVon* in *Vorlesungen* verweist auf Tupel in Professoren

referentielle Integrität

- Fremdschlüssel müssen auf existierende Tupel verweisen oder einen Nullwert enthalten

Referentielle Integrität in SQL

- Kandidatenschlüssel: **unique**
- Primärschlüssel: **primary key**
- Fremdschlüssel: **foreign key**

Beispiel:

create table R
(α **integer primary key**,
...);
create table S
(...,
κ **integer references** R);

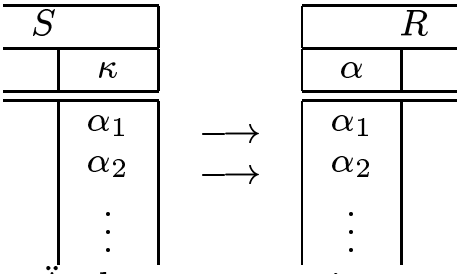
Einhaltung referentieller Integrität

Änderungen von referenzierten Daten

1. Default: Zurückweisen der Änderungsoperation
2. Propagieren der Änderungen: **cascade**
3. Verweise auf Nullwert setzen: **set null**

Einhaltung referentieller Integrität

Originalzustand

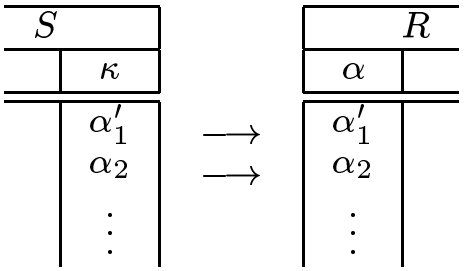


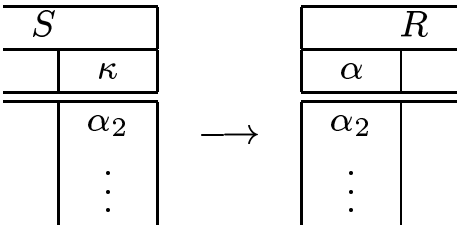
Änderungsoperationen

update R
set $\alpha = \alpha'_1$
where $\alpha = \alpha_1$;

delete from R
where $\alpha = \alpha_1$;

Kaskadieren

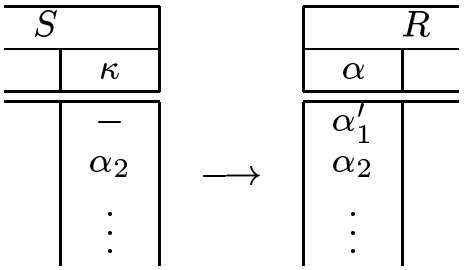


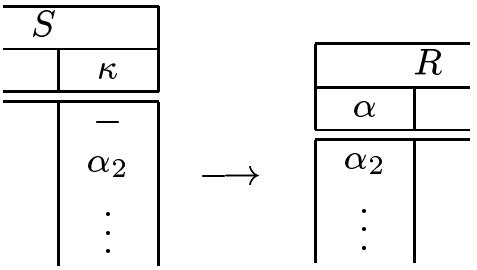


create table S
(...,
κ **integer references** R
on update cascade);

create table S
(...,
κ **integer references** R
on delete cascade);

Auf Null setzen

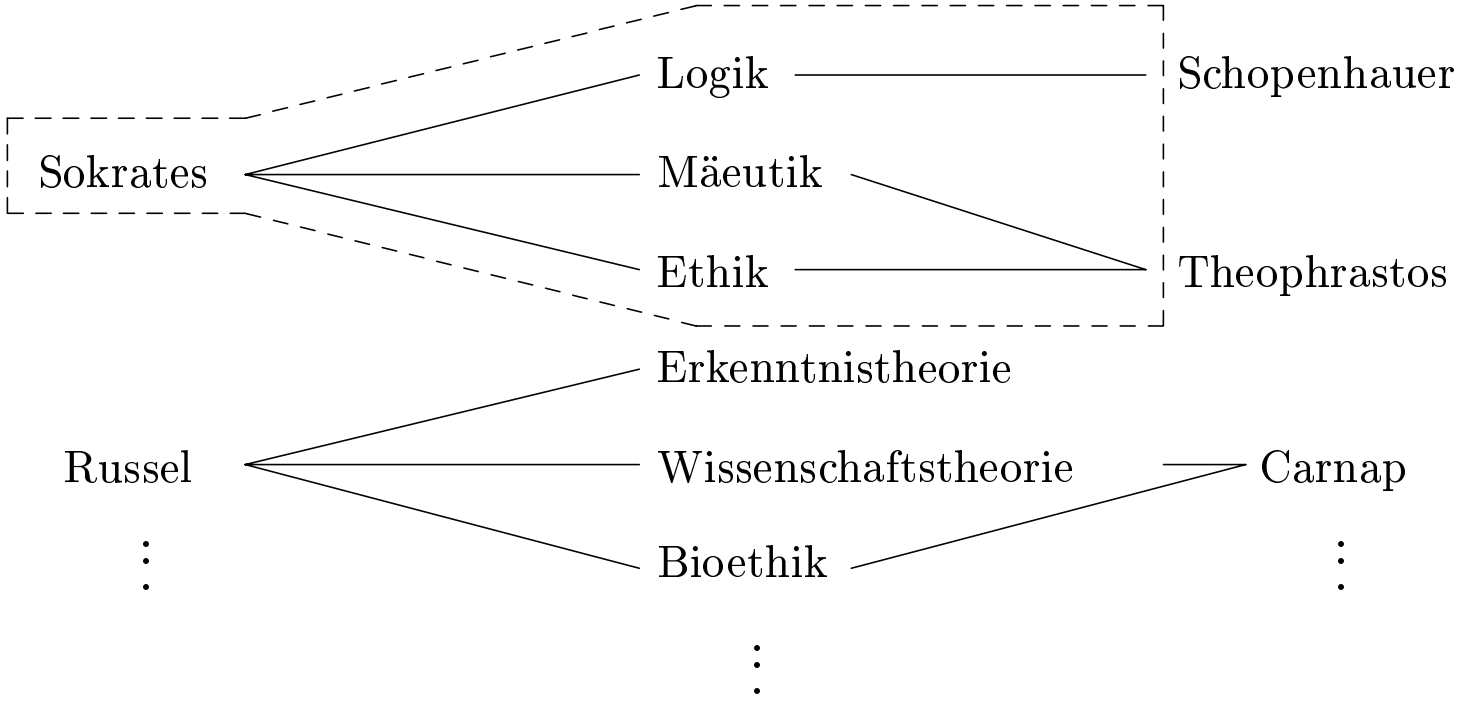




create table S
(...,
κ **integer references** R
on update set null);

create table S
(...,
κ **integer references** R
on delete set null);

Kaskadierendes Löschen



create table Vorlesungen
(. . . ,
gelesenVon **integer**
references Professoren
on delete cascade);

create table hören
(. . . ,
VorlNr **integer**
references Vorlesungen
on delete cascade);

Einfache statistische Integritätsbedingungen

- Wertebereichseinschränkungen
  ...**check** Semester **between** 1 **and** 13
- Aufzählungstypen
  ...**check** Rang **in** ('C2','C3','C4')...

Das Universitätsschema mit Integritätsbedingungen

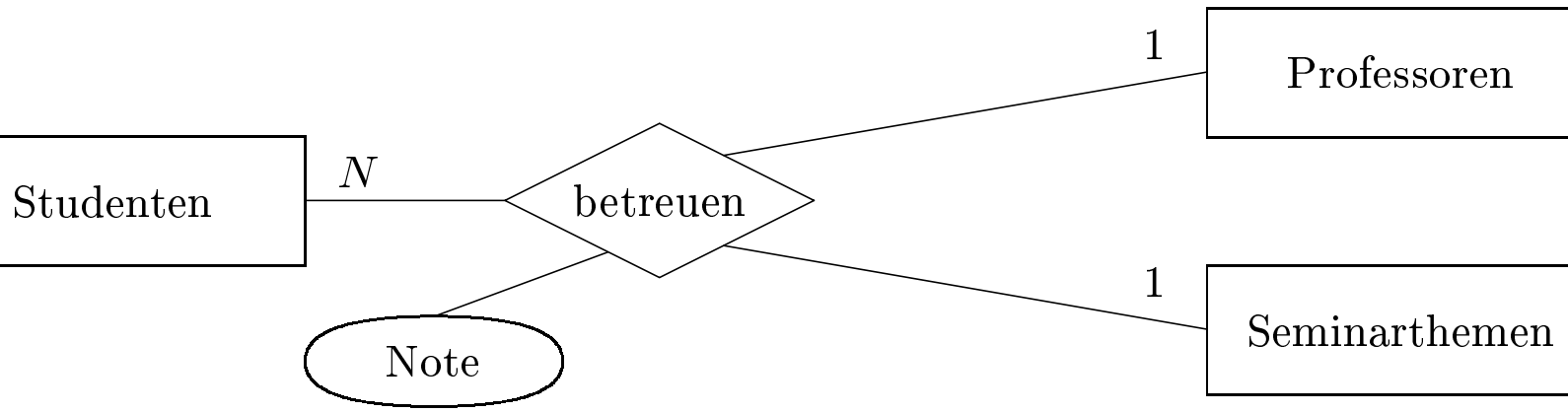
```
create table Studenten
    ( MatrNr        integer primary key,
      Name          varchar(30) not null,
      Semester      integer check Semester between 1 and 13);

create table Professoren
    ( PersNr        integer primary key,
      Name          varchar(30) not null,
      Rang          character(2) check (Rang in ('C2', 'C3', 'C4')),
      Raum          integer unique);

create table Assistenten
    ( PersNr        integer primary key,
      Name          varchar(30) not null,
      Fachgebiet    varchar(30),
      Boss          integer,
      foreign key   (Boss) references Professoren on delete set null);

create table Vorlesungen
    ( VorlNr        integer primary key,
      Titel         varchar(30),
      SWS           integer,
      gelesenVon    integer references Professoren on delete set null);
```

Das Universitätsschema mit Integritätsbedingungen

create table hören
(MatrNr **integer references** Studenten **on delete cascade**,
VorlNr **integer references** Vorlesungen **on delete cascade**,
primary key (MatrNr, VorlNr));

create table voraussetzen
(Vorgänger **integer references** Vorlesungen **on delete cascade**,
Nachfolger **integer references** Vorlesungen **on delete cascade**,
primary key (Vorgänger, Nachfolger));

create table prüfen
(MatrNr **integer references** Studenten **on delete cascade**,
VorlNr **integer references** Vorlesungen,
PersNr **integer references** Professoren **on delete set null**,
Note **numeric**(2,1) **check** (Note **between** 0.7 **and** 5.0),
primary key (MatrNr, VorlNr));




```
create table betreuen (
        MatrNr integer references Studenten
                        on delete cascade,
        PersNr integer references Professoren
                        on delete set null,
        Titel varchar(40) references Seminarthemen
                        on delete no action,
        Note numeric(2,1) check(Note between 0.7 and 5.0),
        constraint PrimKey primary key (MatrNr, Titel),
);

alter table betreuen
        add constraint NurEinsProProf unique (MatrNr, PersNr)
        exceptions into CliquenBildung;
```

```
create trigger keineDegradierung
before update on Professoren
for each row
when (old.Rang is not null)
begin
    if :old.Rang = 'C3' and :new.Rang = 'C2' then
        :new.Rang := 'C3';
    end if;
    if :old.Rang = 'C4' then
        :new.Rang := 'C4';
    end if;
    if :new.Rang is null then
        :new.Rang := :old.Rang;
    end if;
end
```

Sicherheit in DBMS

- Identifikation und Authentisierung
- Autorisierung und Zugriffskontrolle
- Auditing

Angriffsarten

- Mißbrauch von Autorität
- Inferenz und Aggregation
- Maskierung
- Umgehung der Zugriffskontrolle
- Browsing
- Trojanische Pferde
- Versteckte Kanäle

Discretionary Access Control

Zugriffsregeln (o, s, t, p, f) mit

- $o \in O$, der Menge der Objekte (z.B. Relationen, Tupel, Attribute),
- $s \in S$, der Menge der Subjekte (z.B. Benutzer, Prozesse),
- $t \in T$, der Menge der Zugriffsrechte (z.B. $T = \{\text{lesen}, \text{schreiben}, \text{löschen}\}$),
- p ein Prädikat (z.B. *Rang* = 'C4' für die Relation *Professoren*), und
- f ein Boolescher Wert, der angibt, ob s das Recht (o, t, p) an ein anderes Subjekt s' weitergeben darf.

Discretionary Access Control

Realisierung:

- Zugriffsmatrix
- Sichten
- „Query Modification“

Nachteile:

- Erzeuger der Daten= Verantwortlicher für deren Sicherheit

Zugriffskontrolle in SQL

Beispiel:

grant select
    on Professoren
    to eickler;

grant update (MatrNr, VorlNr, PersNr)
    on prüfen
    to eickler;

Zugriffskontrolle in SQL

Weitere Rechte:

- delete
- insert
- references

Weitergabe von Rechten:

- with grant option

Entzug von Rechten:

revoke update (MatrNr, VorlNr, PersNr)
on prüfen
from eickler **cascade**;

Realisierung des Zugriffsprädikats:

```
create view ErstSemestler as
    select *
    from   Studenten
    where Semester = 1;
grant select
    on     ErstSemestler
    to     tutor;
```

Schutz von Individualdaten durch Aggregation:

```
create view VorlesungsHärte (VorlNr, Härte) as
    select VorlNr, avg(Note)
    from prüfen
    group by VorlNr;
```

Auditing

Beispiele:

audit session by system
    whenever not successful;

audit insert, delete, update on Professoren;

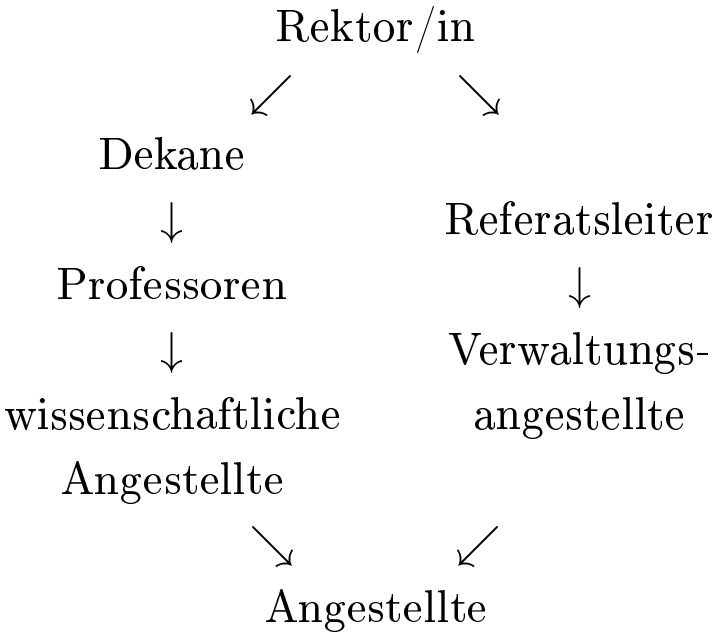
Verfeinerung des Autorisierungsmodells

- explizite/implizite Autorisierung
- positive/negative Autorisierung
- starke/schwache Autorisierung

Autorisierungsalgorithmus:

wenn es eine explizite oder implizite starke Autorisierung (o, s, t) gibt,
    dann erlaube die Operation
wenn es eine explizite oder implizite starke negative Autorisierung $(o, s, \neg t)$ gibt,
    dann verbiete die Operation
ansonsten
    wenn es eine explizite oder implizite schwache Autorisierung $[o, s, t]$ gibt,
        dann erlaube die Operation
    wenn es eine explizite oder implizite schwache Autorisierung $[o, s, \neg t]$ gibt,
        dann verbiete die Operation

Implizite Autorisierung von Subjekten



- explizite positive Autorisierung

  $\Rightarrow$ implizite positive Autorisierung auf allen *höheren* Stufen

- explizite negative Autorisierung

  $\Rightarrow$ implizite negative Autorisierung auf allen *niedrigeren* Stufen

Implizite Autorisierung von Operationen

schreiben
$\downarrow$
lesen

- explizite positive Autorisierung

  $\Rightarrow$ implizite positive Autorisierung auf allen *niedrigeren* Stufen

- explizite negative Autorisierung

  $\Rightarrow$ implizite negative Autorisierung auf allen *höheren* Stufen

Implizite Autorisierung von Objekten

Datenbank

↓

Schema

↓

Relation

↓

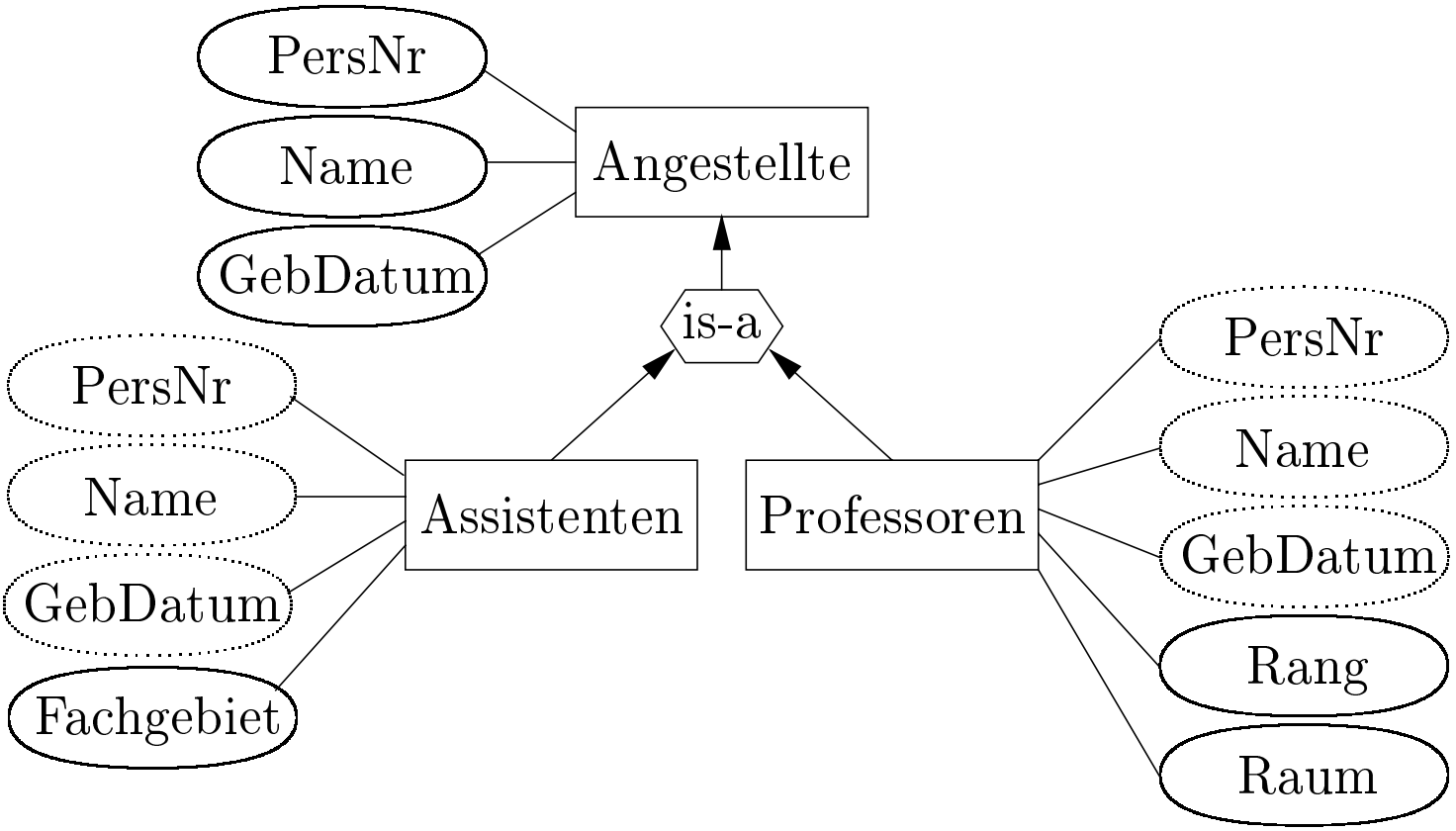
Tupel

↓

Attribut

- Implikationen abhängig von Operation

Implizite Autorisierung entlang einer Typhierarchie



Benutzergruppen:

- Verwaltungsangestellte dürfen die Namen aller Angestellten lesen
- wissenschaftliche Angestellte dürfen Namen und Rang aller Professoren lesen

Anfragen:

- lese die Namen aller Angestellten
- lese Namen und Rang aller Professoren

Implizite Autorisierung entlang einer Typhierarchie

Regeln:

- Benutzer mit einem Zugriffsrecht auf einen Objekttypen haben auf die geerbten Attribute in den Untertypen ein gleichartiges Zugriffsrecht.
- Ein Zugriffsrecht auf einen Objekttyp impliziert auch ein Zugriffsrecht auf alle von Obertypen geerbten Attribute in diesem Typ.
- Ein Attribut, das in einem Untertyp definiert wurde, ist nicht von einem Obertyp aus erreichbar.

Mandatory Access Control

- hierarchische Klassifikation von Vertrauenswürdigkeit und Sensitivität
- $clear(s)$, mit s Subjekt (clearance)
- $class(o)$, mit o Objekt (classification)
- Ein Subjekt s darf ein Objekt o nur lesen, wenn das Objekt eine geringere Sicherheitseinstufung besitzt ($class(o) \leq clear(s)$).
- Ein Objekt o muß mit mindestens der Einstufung des Subjektes s geschrieben werden ($clear(s) \leq class(o)$).

Kryptographie

- Gerade die Gefahr des Abhörens von Kommunikationskanälen ist in heutigen Datenbankarchitekturen und Anwendungen sehr groß.
- Die meisten Datenbankanwendungen werden in einer verteilten Umgebung betrieben – sei es als Client/Server-System oder als „echte“ verteilte Datenbank.
- In beiden Fällen ist die Gefahr des unlegitimierten Abhörens sowohl innerhalb eines LAN (local area network, z.B. Ethernet) als auch im WAN (wide area network, z.B. Internet) gegeben und kann technisch fast nicht ausgeschlossen werden.
- Deshalb kann nur die Verschlüsselung der gesendeten Information einen effektiven Datenschutz gewährleisten.

Ebenen des Datenschutzes

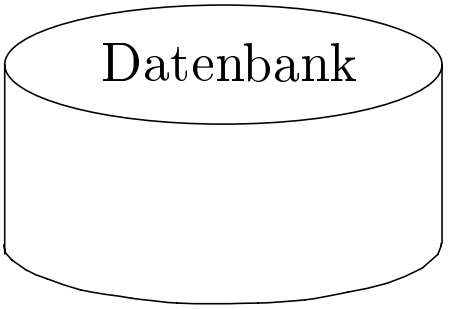
legislative Maßnahmen

organisatorische Maßnahmen

Authentisierung

Zugriffskontrolle

Kryptographie



- Benutzer soll sich der Existenz unzugänglicher Daten nicht bewußt sein

Beispiel (TC= Klassifizierung des gesamten Tupels):

Agenten						
TC	Kennung	KC	Name	NC	Spezialität	SC
g	007	g	Blond, James	g	meucheln	sg
sg	008	sg	Mata, Harry	sg	spitzeln	sg

Sichtweise eines „geheim“ eingestuften Benutzers:

Agenten						
TC	Kennung	KC	Name	NC	Spezialität	SC
g	007	g	Blond, James	g	–	g

Probleme:

- „geheimer“ Benutzer fügt Tupel mit Schlüssel „008“ ein
- „geheimer“ Benutzer modifiziert Spezialität von „007“

Multilevel-Relationen

Multilevel-Relation R mit Schema

$$\mathcal{R} = \{A_1, C_1, A_2, C_2, \ldots, A_n, C_n, TC\}$$

Relationeninstanzen R_c mit Tupeln

$$[a_1, c_1, a_2, c_2, \ldots, a_n, c_n, tc]$$

- $c \geq c_i$
- a_i ist sichtbar, wenn $class\ (s) \geq c_i$

Integritätsbedingungen

Sei κ *sichtbarer Schlüssel* der Multilevel-Relation R

Entity-Integrität. R erfüllt die Entity-Integrität genau dann, wenn für alle Instanzen R_c und $r \in R_c$ die folgenden Bedingungen gelten:

1. $A_i \in \kappa \Rightarrow r.A_i \neq \text{Null}$
2. $A_i, A_j \in \kappa \Rightarrow r.C_i = r.C_j$
3. $A_i \notin \kappa \Rightarrow r.C_i \geq r.C_\kappa$ (wobei C_κ die Zugriffsklasse des Schlüssels ist)

Null-Integrität. R erfüllt die Null-Integrität genau dann, wenn für jede Instanz R_c von R gilt:

1. $\forall r \in R_c, r.A_i = \text{Null} \Rightarrow r.C_i = r.C_\kappa$
2. R_c ist subsumierungsfrei, d.h. es existieren keine zwei Tupel r und s, bei denen für alle Attribute A_i entweder
   - $r.A_i = s.A_i$ und $r.C_i = s.C_i$ oder
   - $r.A_i \neq \text{Null}$ und $s.A_i = \text{Null}$ gilt.

Subsumptionsfreiheit von Relationen

a) R_{sg}

Agenten						
TC	Kennung	KC	Name	NC	Spezialität	SC
g	007	g	Blond, James	g	–	g

b) Änderung von R_{sg}

Agenten						
TC	Kennung	KC	Name	NC	Spezialität	SC
sg	007	g	Blond, James	g	meucheln	sg

c) Fehlende Subsumtionsfreiheit

Agenten						
TC	Kennung	KC	Name	NC	Spezialität	SC
g	007	g	Blond, James	g	–	g
sg	007	g	Blond, James	g	meucheln	sg

Interinstanz-Integrität. R erfüllt die Interinstanz-Integrität genau dann, wenn für alle Instanzen R_c und $R_{c'}$ von R mit $c' < c$

$$R_{c'} = f(R_c, c')$$

gilt. Die Filterfunktion f arbeitet wie folgt:

1. Für jedes $r \in R_c$ mit $r.C_\kappa \leq c'$ muß ein Tupel $s \in R_{c'}$ existieren, mit

$$s.A_i = \begin{cases} r.A_i & \text{wenn } r.C_i \leq c' \\ \text{Null} & \text{sonst} \end{cases}$$

$$s.C_i = \begin{cases} r.C_i & \text{wenn } r.C_i \leq c' \\ r.C_\kappa & \text{sonst} \end{cases}$$

2. $R_{c'}$ enthält außer diesen keine weiteren Tupel.

3. Subsumierte Tupel werden eliminiert.

Polyinstanziierungsintegrität. R erfüllt die Polyinstanziierungsintegrität genau dann, wenn für jede Instanz R_c für alle A_i die folgende funktionale Abhängigkeit gilt: $\{\kappa, C_\kappa, C_i\} \rightarrow A_i$.

Kryptographie

- Gerade die Gefahr des Abhörens von Kommunikationskanälen ist in heutigen Datenbankarchitekturen und Anwendungen sehr groß.
- Die meisten Datenbankanwendungen werden in einer verteilten Umgebung betrieben – sei es als Client/Server-System oder als „echte“ verteilte Datenbank.
- In beiden Fällen ist die Gefahr des unlegitimierten Abhörens sowohl innerhalb eines LAN (local area network, z.B. Ethernet) als auch im WAN (wide area network, z.B. Internet) gegeben und kann technisch fast nicht ausgeschlossen werden.
- Deshalb kann nur die Verschlüsselung der gesendeten Information einen effektiven Datenschutz gewährleisten.

Ebenen des Datenschutzes

legislative Maßnahmen

organisatorische Maßnahmen

Authentisierung

Zugriffskontrolle

Kryptographie