Mehrtabellenanfragen

Equi-Join: Join, bei dem der Link durch eine Gleichheitsforderung etabliert wird.

Verallgemeinerung (1): Link kann durch eine beliebige Relation etabliert werden.

z.B. "Größer als", ...

Verallgemeinerung (2): SELECT-Klausel kann mehr als zwei Tabellen verknüpfen.

Mehrtabellenanfragen

Aliases: Vergabe von Pseudonymen für Tabellen

Referenz auf Tabellen (z.B in Selektionsbedingungen) unter Verwendung der Pseudonyme

`select ... from` ⟨Tabellenname1⟩ ⟨Alias1⟩, ⟨Tabellenname2⟩ ⟨Alias2⟩
`where` ⟨Alias1⟩.⟨Feldname1⟩=⟨Alias2⟩.⟨Feldname2⟩

Wozu Aliases?

1.**Reduzierter Schreibaufwand** in Joins bei langen Tabellennamen

```
select
        sonderabschreibungen.nr,
        sonderabschreibungen.bewilligung,
        sonderabschreibungen.art,
        invest_objekte.nr,
        invest_objekte.typ,
        invest_objekte.preis
    from sonderabschreibungen, invest_objekte
    where sonderabschreibungen.objekt=invest_objekte.id
      and invest_objekte.preis>100000

select X.nr,X.bewilligung,X.art,Y.nr,Y.typ,Y.preis
    from sonderabschreibungen X, invest_objekte Y
    where X.objekt=Y.id and Y.preis>100000
```

Mehrtabellenanfragen

Wozu Aliases?

2. Self-Join: Verknüpfung einer Tabelle mit sich selbst

Beispiel: Weiterverkauf einer Immobilie

Verknüpfung der Tabelle `bew` mit sich selbst

1. Verkäufe mit identischen Objektidentifikatoren (`obj_id`)
2. Der Käufer im ersten Tupel ist der Verkäufer im zweiten.

```
select X.kauf,X.preis,Y.preis from bew X, bew Y
    where X.obj_id=Y.obj_id and X.kauf=Y.verk
```

kauf	preis	preis
Schneider	560000	615000

```
select X.kauf,X.preis,Y.preis,X.datum,Y.datum
    from bew X, bew Y
    where X.obj_id=Y.obj_id and X.kauf=Y.verk
```

kauf	preis	preis	datum	datum
Schneider	560000	615000	88.12.13	96.12.01

Datenaktualisierung

UPDATE: Änderung von Feldinhalten

`update` ⟨Tabellenname⟩ `set` ⟨Feldname⟩ = ⟨Wert⟩ `where` ⟨Bedingung⟩

```
select * from obj where str like 'B%' and no=27

+----+------------+----------------------+----+------+
| id | type       | str                  | no | jahr |
+----+------------+----------------------+----+------+
| 2  | EFH        | Bahnhofsstr          | 27 | 1943 |
+----+------------+----------------------+----+------+

update obj set jahr=1997 where id=2

select * from obj where str like 'B%' and no=27

+----+------------+----------------------+----+------+
| id | type       | str                  | no | jahr |
+----+------------+----------------------+----+------+
| 2  | EFH        | Bahnhofsstr          | 27 | 1997 |
+----+------------+----------------------+----+------+
```

Auch einfache arithmetische Ausdrücke in der SET-Klausel zulässig (von mSQL nicht unterstützt):

```
update table
  set value = value + 1
  where value < 27
```

Datenaktualisierung

DELETE: Löschen von Tupeln

`delete from` ⟨Tabellenname⟩ `where` ⟨Bedingung⟩

DROP: Löschen einer ganzen Tabelle

`drop table` ⟨Tabellenname⟩

ALTER: Ändern der Tabellenstruktur (von mSQL nicht unterstützt)

`alter table` ⟨Tabellenname⟩ `add` (⟨Feldname⟩ ⟨Typ⟩, ...)

Datenmodellierung

Datenmodellierung: Strukturierung einer Domäne mit dem Ziel einer optimalen Anpassung an die Möglichkeiten des relationalen Modells

1. Das Relationale Modell beschränkt die Möglichkeiten zur Modellierung einer Domäne.
   - ☐ zeitliche Aspekte
2. Eine effiziente Arbeit mit der Datenbank erfordert die Einhaltung zusätzlicher Integritätsbedingungen.
   - ☐ Eindeutige Identifizierung von Datenbankeintragungen.
   - ☐ Vermeiden kombinatorischer Probleme beim Join.

Datenmodellierung

Entity-Relationship-Diagramme

Entities: Objekte einer Domäne, über die Informationen verwaltet werden sollen.

z.B. Immobilien, Besitzer, Verkäufe, ...

z.B. Bücher, Leser, Ausleihen, Vorbestellungen, ...

Immobilien

Properties: Eigenschaften (Attribute) von Entitäten

z.B. Adresse, Straßenname, Hausnummer, Preis, Besitzername, ...

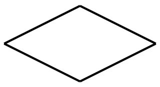
z.B. Autor, Titel, Ausleihdatum, Rückgabedatum, ...

Autor

Relationships: Beziehungen zwischen Entitäten

z.B. Besitzer verkauft Immobilie

z.B. Leser leiht ein Buch aus



Datenmodellierung

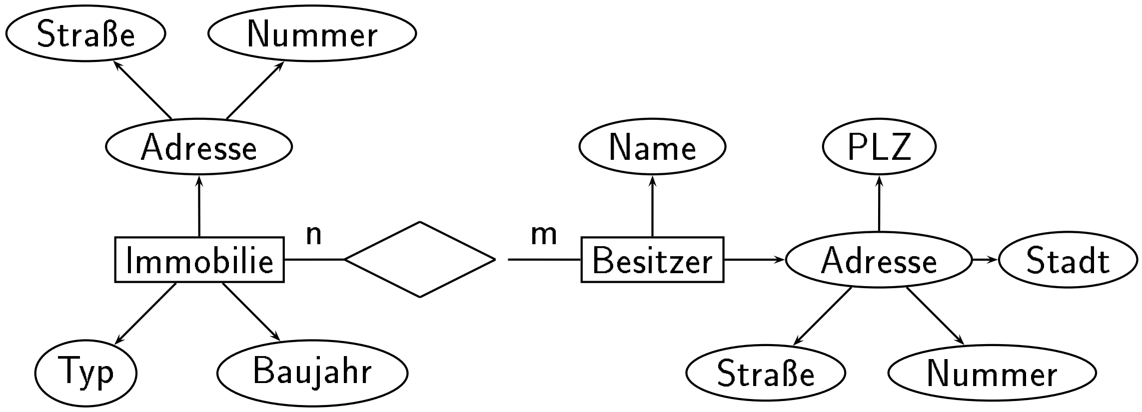
Entwurfskriterien:

Regel 1: Beschränkung auf binäre Beziehungen (join)

Regel 2: Keine Dopplung von Information (Aktualisierungsprobleme)

Regel 3: Keine m:n-Beziehungen.

Beispiel: Immobilien-Markt

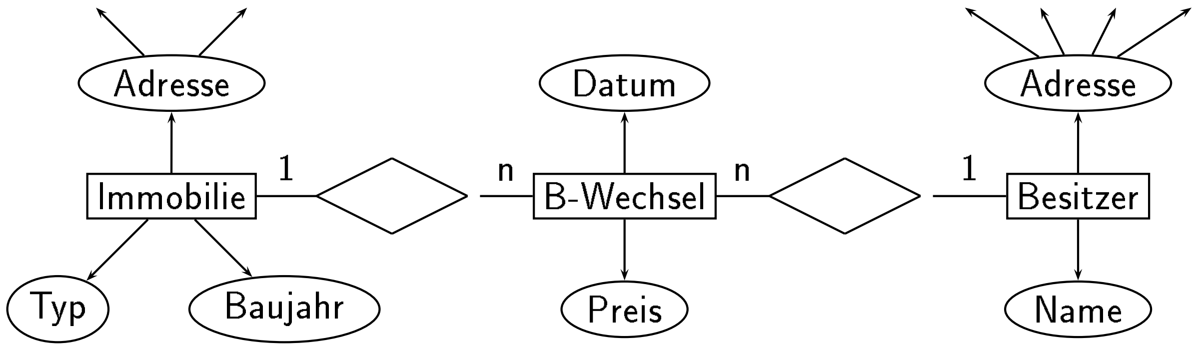


Probleme:

- m:n-Beziehung zwischen Immobilien und Besitzern
- Besitzer-Immobilien-Beziehung ist zeitabhängig
  - Beschränkung auf den atemporalen Momentanzustand
  - aber: erlaubt keine Analysen der Marktentwicklung
- Kann man die Adressen zusammenfassen? (Beseitigt Informationsdopplung!)

Datenmodellierung

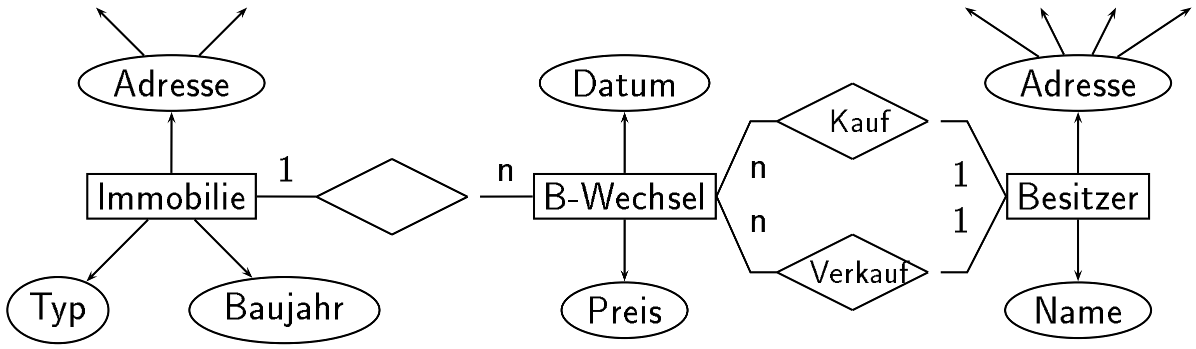
Lösung: Explizite Repräsentation von Besitzwechseln.



Probleme:

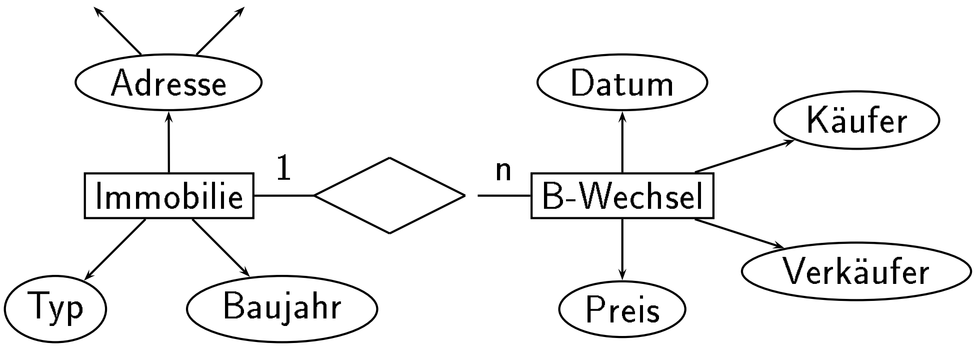
- Zwei unterschiedliche Beziehungen zwischen Besitzern und Besitzwechseln: Käufer/Verkäufer.

Lösung: Modellierung von zwei getrennten Beziehungen.



Datenmodellierung

Vereinfachung: Käufer und Verkäufer sind eindeutig identifizierbar.



id	type	str	no	jahr

id	obj_id	verk	kauf	preis	datum

Datenmodellierung

Abbildung von E/R-Diagrammen auf Datenbanktabellen:

Regel 4: Jede Entität wird zu einer separaten Tabelle.

Regel 5: Jede Eigenschaft wird zu einer Spalte in der Tabelle der zugehörigen Entität.

Regel 6: Bei kompositionalen Eigenschaften werden nur die eingebetteten Komponenten als Tabellenspalte repräsentiert.

Regel 7: Jede Tabelle benötigt einen Primärschlüssel (primary key).

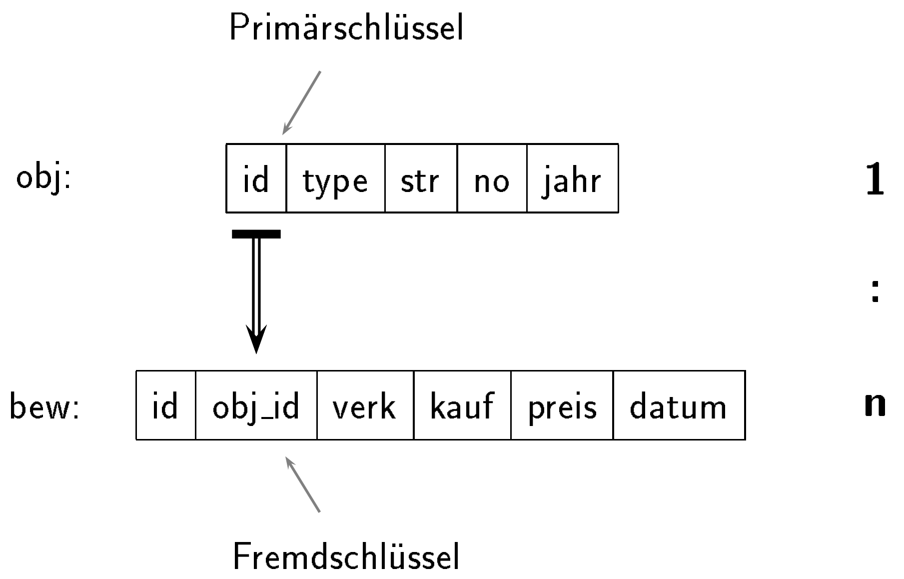
Primärschlüssel: Feldinhalt (bzw. Kombination von Feldinhalten), die eine *eindeutige* Identifizierung einer Tabellenzeile ermöglicht.

Einfachste Realisierung: Vergabe einer eindeutigen Objektkennzeichnung (Objektnummer, Patientennummer, Rechnungsnummer, ...)

Regel 8: Jede 1:n-Beziehung verbindet einen Primärschlüssel mit einer Referenzspalte (Fremdschlüssel, foreign key).

- Der Fremdschlüssel ist eine separate Spalte der betreffenden Tabelle.
- Die Domäne des Fremdschlüssels ist die des zugehörigen Primärschlüssels.
- m:n-Beziehungen sind nicht erlaubt!

Datenmodellierung

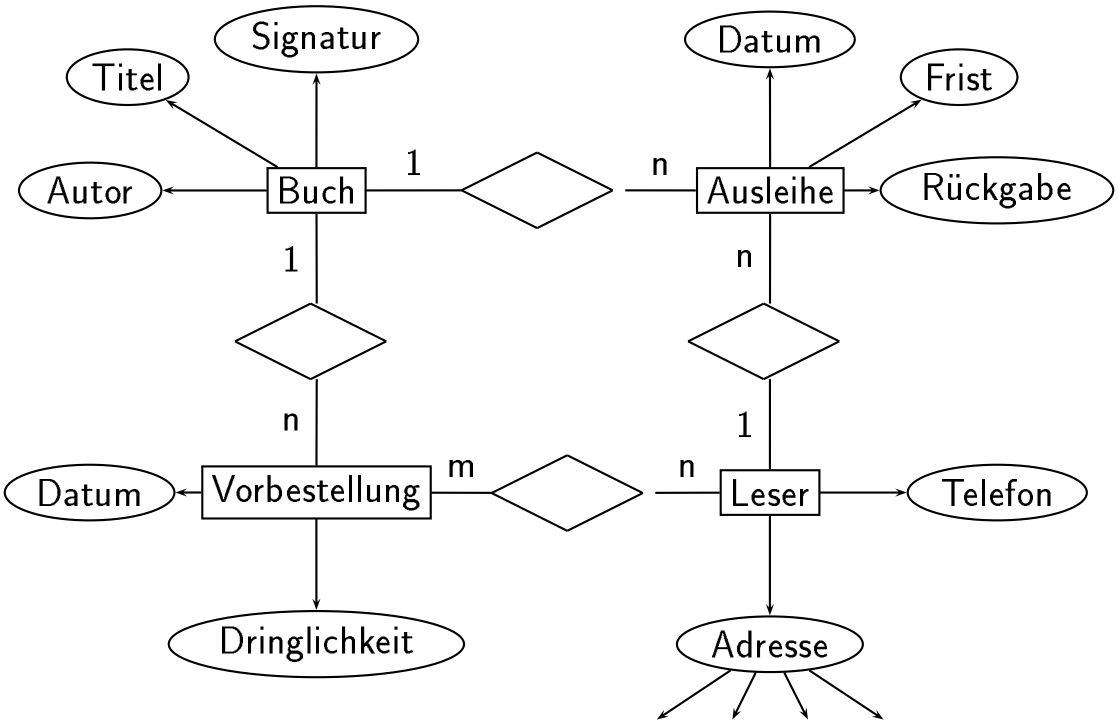


Referentielle Integrität: Regel 7 und 8 garantieren die Auflösbarkeit aller tabellenübergreifenden Referenzen und ermöglichen die effiziente Arbeit mit der Datenbank.

Datenmodellierung

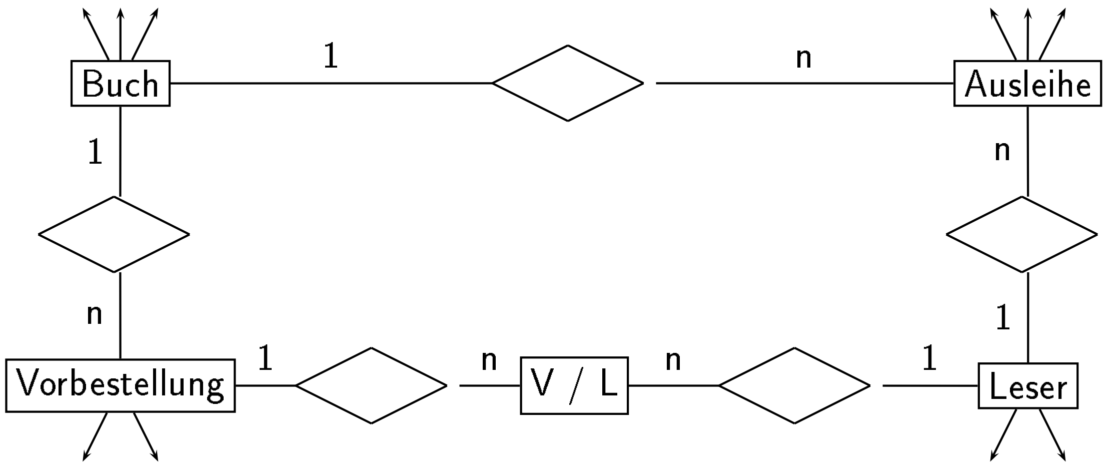
m:n-Beziehungen

Beispiel: Bibliotheksverwaltung



Datenmodellierung

Lösung: Aufspaltung in zwei 1:n-Beziehungen.



Relationenalgebra

Funktionale Abbildung: SELECT-Klauseln erzeugen immer Tabellen aus Tabellen

$\boldsymbol{T} \rightarrow \boldsymbol{T}$

$\boldsymbol{T} \times \boldsymbol{T} \rightarrow \boldsymbol{T}$

$\rightarrow$ seiteneffektfrei

Relationenalgebra: Beliebige Operationen lassen sich durch eine geeignete Kombination von Elementaroperationen beschreiben:

Restrict: Auswahl von Tupeln einer Relation

Project: Auswahl von Argumenten einer Relation

Product: Cartesisches Produkt zweier Relationen

(natural) Join: Verknüpfung zweier Relationen über einen Link

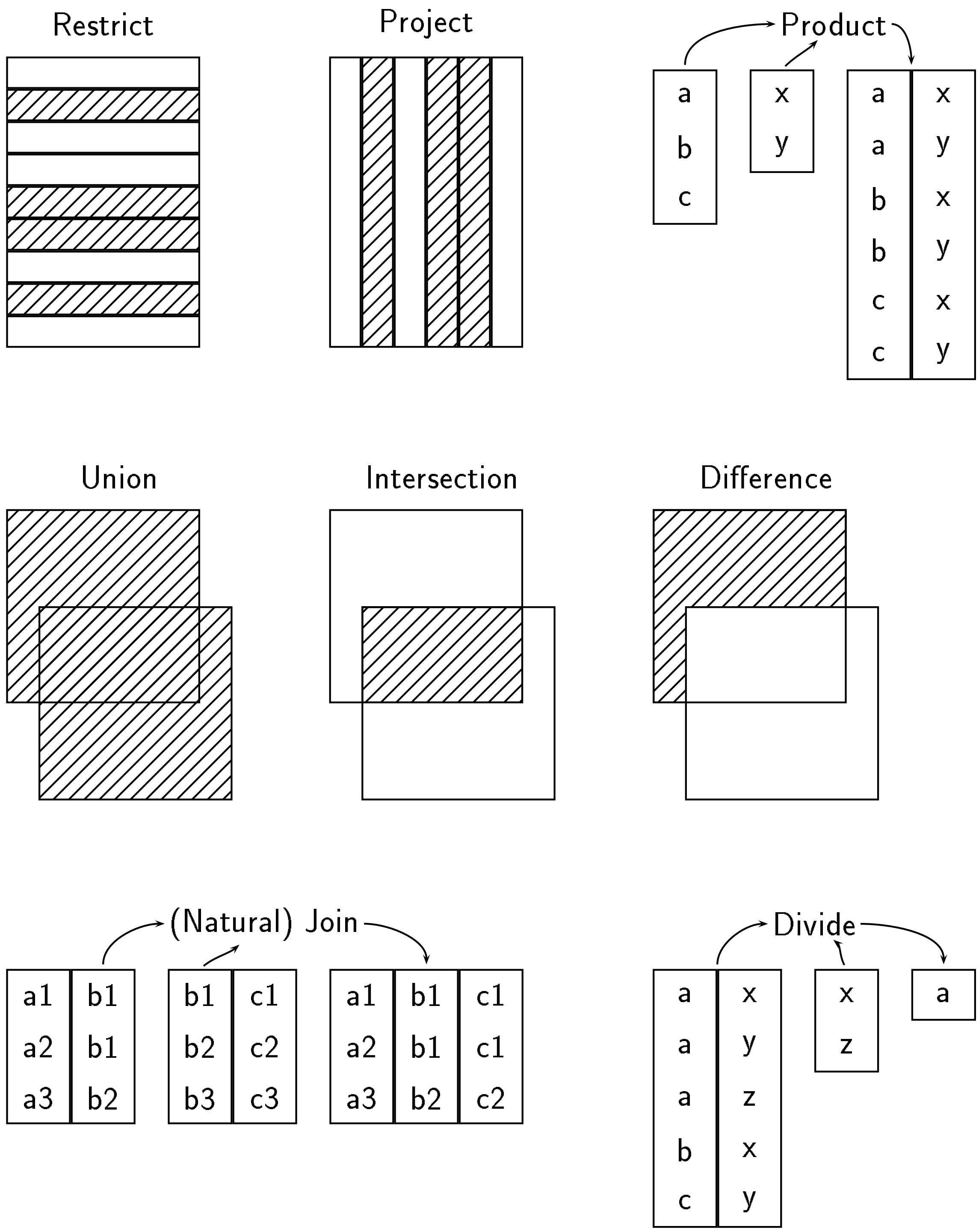
Union: Mengenvereinigung der Tupel zweier strukturell gleicher Relationen

Intersection: Mengendurchschnitt der Tupel zweier strukturell gleicher Relationen

Difference: Mengendifferenz der Tupel zweier strukturell gleicher Relationen

Divide: Verknüpfung einer binären Relation mit einer unären, so daß die resultierende unäre Relation alle Werte eines Arguments der binären Relation enthält, deren korrespondierendes Argument mit einem Werte der unären Relation übereinstimmt.

Relationenalgebra



Relationenalgebra

- SELECT-Klausel ist Kombination aus Projektion und Restriktion, sowie ggf. einem oder mehreren natürlichen Joins.

  PROJECT(RESTRICT(T))

  RESTRICT(PROJECT(T))

  PROJECT(RESTRICT(JOIN(T_1, T_2)))

- Join kann auch mehr als zwei Tabellen verknüpfen.

  JOIN(T_1, T_2)

  JOIN(T_1,JOIN(T_3, T_4))

  JOIN(T_1,JOIN(T_3,JOIN(T_5, T_6)))

  ...

- mehrere SELECT-Klauseln können ineinander eingebettet werden (wird von mSQL nicht unterstützt):

```
select ... from ⟨Tabelle1⟩
  where ⟨Feldname⟩ in
    (select ⟨Feldname⟩ from ⟨Tabelle2⟩
      where ... )
```

  → Verwendung des Suchergebnisses für eine Tabelle zur Suche in einer anderen.

Relationenalgebra

Grenzen relationaler Datenbankkalküle:

1. Begrenzter Einsatz arithmetischer Operationen
   - ◻ UPDATE-Klausel
   - ◻ Auswertungsfunktionen (sum, avg, count, ...)
2. Anzahl der Tupel in einer Relation ist endlich.
3. Zahl der in einem Join verknüpften Relationen ist durch die Suchanfrage vorgegeben.
4. Komplex strukturierte Domänen lassen sich nur schwer mit einem relationalen Kalkül modellieren (→ objektorientierte Datenbanken).
5. Implizites Wissen über bestimmte Entitäten kann nicht direkt (als generelle Aussage) repräsentiert werden, sondern muß explizit (d.h. individuell) kodiert werden (→ deduktive Datenbanken).

Erweiterungen:

- ◻ Einbettung in eine (imperative) Programmiersprache (Embedded SQL). SQL-Kommandos können von einer Einbettungssprache aus aktiviert werden.
- ◻ Persistente Erweiterung einer Programmiersprache (Modula, Prolog). Datentypen der Wirtssprache können durch ein (relationales) Datenbanksystem verwaltet werden.