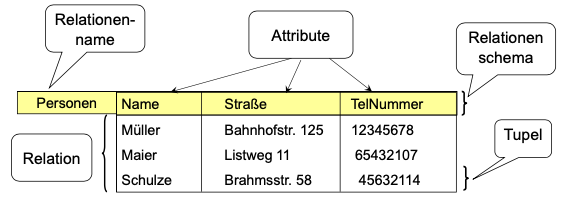
# Kapitel 3 Implementierungsmodell

## Relationen Modell

* Das relationale Modell definiert eine Datenstruktur
* Alle Informationen ausschließlich durch atomare Werte dargestellt
* Implementierung von Entitätstypen und Beziehungstypen durch Relationen
* Definiert Operationen auf (mehreren) Relationen
* Vereinigung, Durchschnitt, Differenz, Kartesisches Produkt
* Projektion, Selektion, Verbundoperationen

### Datenstrukturen



Relationenschema Personen(Name, Straße, TelNummer)

dom(Name) = dom(Strasse) = {Zeichenkette}

dom(TelNummer) = {8 - stellige Zahl}

* Sortierung der Zeilen ohne Bedeutung
* Sortierung der Spalten ohne Bedeutung (eindeutiger Attributname)
* Alle Werte sind atomar

### Relation und Relationen Schema

Gegeben seien n Wertebereiche (Domänen) D1, D2, ..., Dn, die nur atomare Werte enthalten

* atomare Werte: Zeichenketten (Strings), Zahlen

Eine **Relation** r ist definiert als Teilmenge des kartesischen Produktes (Kreuzprodukt) der Domänen

r ⊆ D1 x D2 x ... x Dn

**Daten der realen Welt werden als Sammlung von Relationen dargestellt:**

* Relation = Tabelle
* Spalte = Werte aus einem Wertebereich Di
* Zeile = Tupel (d1, d2, ..., dn)

**mathematisch**:

* n: Anzahl der Attribute (**Grad** der Relation)
* Di: Wertebereich des Attributes Ai ist eine Menge
* r ∈ D1 x D2 x ... x Dn: **n - stellige Relation** über den Wertebereichen D1, ..., Dn
* (d1, d2, ..., dn) ∈ r: **Tupel** der Relation R
* di ∈ Di: i - te Komponente des Tupels

**Zusammenfassung:**

* Relationenschema R (A1, A2, ..., An):
* R ist der Relationenname
* Ai ist der Name des i-ten Attributes
* Jedem Attribut Ai ist ein Wertebereich dom(Ai) = Di zugeordnet.
* Zu jedem Zeitpunkt existiert genau eine Relation r zum Relationenschema R: “r ist eine Instanz von R”.
* Die zu R gehörigen Relationen sind also sämtliche Relationen des Typs   
  r ∈ dom(A1) x dom(A2) x ... x dom(An) = D1 x ... x Dn
* Die Gesamtheit der Relationenschemata einer Datenbank   
  heißt Schema der relationalen Datenbank oder **Datenbankschema**
* Zusätzlich gehören zum Schema noch **Integritätsbedingungen**

**Beispiel**: Kunde( Kundennr, Name, Telefon)

dom(Kundennr) = int || dom(Name) = String || dom(Telefon) = Menge der Zeichenketten

Kunde = { (4711, „Mustermann“, „+49 511 47 114 711“),

(4712, „Musterfrau“, „+49 511 47 124 712“),... }

## Relationale Algebra

**Operationsarten:**

* Datenverwaltung (Datendefinition, Datenmanipulation, Zugriffs-, Integritäts- und Transaktionskontrolle)
* Anfragen (engl. queries)

**Relationenalgebra** prozedurale Sprache, in der spezifiziert wird, wie eine Anfrage auszuwerten ist.

### Selektion

Ein Bild, das Tisch enthält.

Automatisch generierte BeschreibungEs werden Zeilen einer Relation ausgewählt, für die **eine Bedingung** zutrifft.

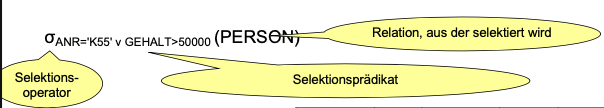
**Im Beispiel: Notation: Ergebnis** ist eine neue **Relation** mit

- Ort = 'Hannover' - σOrt='Hannover'(Kunde) **gleichem Schema**

* Gesucht sind alle Einträge mit bestimmten Eigenschaften
* Auswahl von Zeilen (Tupel) einer Relation über Prädikate.

Schreibweise: σP(R)

* **Ein Bild, das Text enthält.

  Automatisch generierte Beschreibung**Prädikate sind **boolesche Ausdrücke**
* Prädikate bestehen aus:
* Operanden (Konstanten oder Attributnamen)
* Vergleichs-Operatoren (<, =, >, ≤, ≠, ≥)
* Logische Operatoren (¬, ∧, ∨)
* **Definition:** σP(R) = { t | t ∈ R ∧ P(t) }

### Projektion

Es werden **bestimmte Spalten** einer Relation ausgewählt.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

**Im Beispiel: Notation: Ergebnis** ist eine neue **Relation** mit

- Name - πName(Kunde) **anderem Schema**

* Es interessieren oft nicht alle Spalten einer Relation
* Auswahl der Spalten (Attribute) A1, A2, ..., Ak aus einer Relation R mit Grad n ≥ k.

πA1, A2, ..., Ak(R)

* **Definition**: πA1, A2, ..., Ak(R) = { p| ∃ t ∈ R mit p=(t(A1), ..., t(Ak)) }

Bsp. σGehalt > 50000(πName, Gehalt, Alter(Person))

### Kreuzprodukt

**Kartesisches Produkt zweier Mengen:** alle möglichen Paare, deren erstes Element aus der einen Menge und deren zweites Element aus der anderen Menge stammt

* Beispiel 1:

- M1 = {1,2,3} M2 ={a,b}

- M1 × M2 = { (1,a), (1,b), (2,a), (2,b), (3,a), (3,b) }

Wenn in einer Menge bereits Tupel sind, dann werden sie im Bereich Datenbanken i.d.R. „flachgeklopft“

* Beispiel 2:

- M1 = { (1,2), (3,4) } M2 ={ (a,x), (b,y) }

- M1 × M2 = { ((1,2), (a,x)), ((3,4), (a,x)), ((1,2), (b,y)), ((3,4), (b,y)) }

= { (1,2,a,x), (3,4,a,x), (1,2,b,y), (3,4,b,y) }

**Kombinieren von Kunden und Telefonnummern Alle Kunden mit allen Telefonnummern**

Ein Bild, das Tisch enthält.

Automatisch generierte BeschreibungEin Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung



**Kombination mit Selektion und Projektion**

* Ein Bild, das Tisch enthält.

  Automatisch generierte BeschreibungπKunde.Kdnr, Name, Ort, Telefon (σKunde.Kdnr = Telefon.Kdnr(Kunde × Telefon))

Verbundoperation (engl. Join): Kunde ⨝Kunde.Kdnr = Telefon.Kdnr Telefon

Enspricht: σKunde.Kdnr = Telefon.Kdnr(Kunde × Telefon)

### Verbund

Kartesisches Produkt zwischen zwei Relationen R mit Grad r und Relation S mit Grad s, eingeschränkt durch eine Θ -Bedingung zwischen zwei Spalten A und B Vergleichsoperator Θ ∈ {<, =, >, ≤, ≠, ≥}

* Θ -Verbund zwischen R und S:

R ⨝A Θ B S

* Für Θ = '=' spricht man auch vom Gleichverbund (**Equi-Join**):

R ⨝A = B S

* Es sind auch kompliziertere Bedingungen möglich:

R ⨝A > B v A < C S

#### Natürlicher Verbund (engl. Natural Join)

Es werden diejenigen Tupel aus den Relationen R und S kombiniert, für die die Werte der Attribute gleichen Namens übereinstimmen. Im Ergebnis sind diese Attribute nur einmal vorhanden.

Gegeben:

• R (A1, A2, ..., Am, B1, B2, ..., Bn) • S (B1, B2, ..., Bn, C1, C2, ..., Ck)

natürlicher Verbund von R und S ist nun:

• πA1, A2, ..., Am, R.B1, ..., R.Bn, C1, ... Ck (σR.B1=S.B1 ∧ ... ∧ R.Bn=S.Bn(R × S))

und wird folgendermaßen aufgeschrieben:

**• R⨝S**

Es kann πKunde.Kdnr, Name, Ort, Telefon (σKunde.Kdnr = Telefon.Kdnr(Kunde × Telefon)) als **Kunde ⨝ Telefon** geschrieben werden.

### Umbenennungen

Manchmal müssen Relationen oder Attribute umbenannt werden, damit Namenskonflikte aufgelöst werden

Qualifizierte Attributnamen bestehen aus Relationennamen und Attributnamen, getrennt durch einen Punkt:

Ein Bild, das Tisch enthält.

Automatisch generierte BeschreibungEin Bild, das Tisch enthält.

Automatisch generierte BeschreibungBeispiele:

• StudentIn.Name

• ProfessorIn.Name

Frage: Was ist die Vor-Voraussetzung für das Java-Projekt?

**Umbenennung - Beispiel**

Idee: kartesisches Produkt aus Voraus × Voraus und Herausfiltern des   
passenden Eintrags

**Umbenennung – Definition und Schreibweise Umbenennungsoperator**

ρNeuerName(AlterName)

Beispiel

ρV1(Voraus) benennt die Tabelle Voraus in V1 um   
ρV2(Voraus) benennt die Tabelle Voraus in V2 um

Ergebnis:

σV2.Nachfolger="Java-Projekt" ∧ V1.Nachfolger=V2.Vorgänger(ρV1(Voraus) x ρV2(Voraus))

### Ein Bild, das Lampe, Halskettchen enthält. Automatisch generierte BeschreibungMengenoperationen

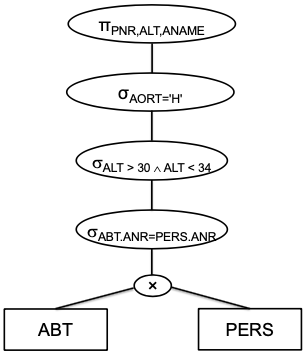
Vereinigungsverträglichkeit der beteiligten Relationen muss gegeben sein:

* Gleicher Grad der Relationen, d.h. dieselbe Anzahl Attribute
* Gleiche Bereiche, d.h. dieselben Domänen für die Attribute

**Typischen Mengenoperationen:**

|  |  |  |
| --- | --- | --- |
| **Vereinigung (UNION)**  R ∪ S = {t | t ∈ R ∨ t ∈ S} | **Differenz**  R - S = R \ S = {t | t ∈ R ∧ t ∉ S} | **Durchschnitt**  R ∩ S = R - (R - S) = {t | t ∈ R ∧ t ∈ S} |

### Anfrageoptimierung

Finde alle Angestellten (PNR, ALTER, ANAME), die in einer Abteilung in Hannover arbeiten und zwischen 30 und 34 Jahre alt sind.

πPNR,ALT,ANAME(σAORT='H'(σALT > 30 ∧ ALT < 34(σABT.ANR=PERS.ANR(ABT × PERS))))

πPNR, ALT, ANAME(σALT > 30 ∧ ALT < 34 ∧ AORT='H'(ABT ⨝ABT.ANR=PERS.ANR PERS))

πPNR, ALT, ANAME(σAORT='H'(ABT) ⨝ABT.ANR=PERS.ANR σALT > 30 ∧ ALT < 34(PERS))

**Operatorbaum**

In einem Operatorbaum kann man sich veranschaulichen, wie und in welcher Reihenfolge die Operatoren ausgeführt werden.

### Optimierungsregeln für Anfragen

* Führe Selektionen so früh wie möglich aus
* Führe Projektionen frühzeitig aus
* Fasse einfache Selektionen auf einer Relation zusammen
* Verknüpfe bestimmte Selektionen mit einem vorausgehenden kartesischen Produkt zu einem Verbund
* Bestimme die Verbundreihenfolge so, dass die Anzahl und die Größe der Zwischenobjekte minimiert wird

### Äußerer Verbund

Bisher konnten Joins unvollständig sein, d.h. Tupel aus einer Relation waren im Join nicht vertreten, wenn sie keinen Partner gefunden haben.

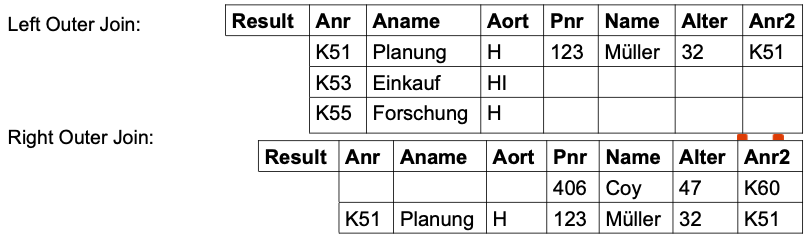
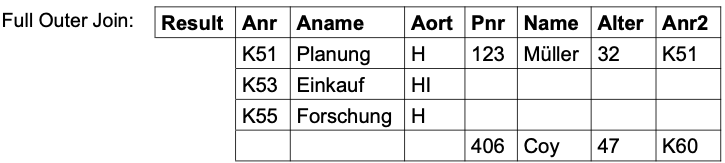
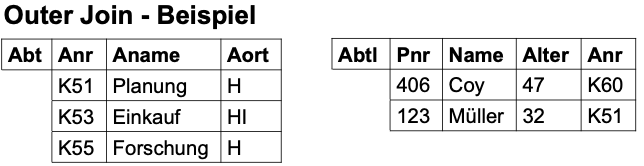
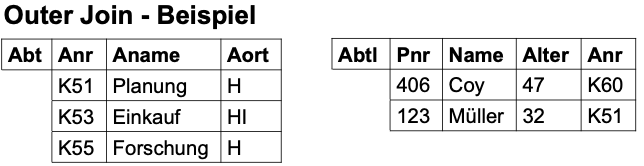
**Äußerer Verbund**

• Nehme auch partnerlose Tupel mit auf

**Äußerer Verbund: Versionen**

* linker äußerer Verbund (left outer join):   
  alle Tupel der linken Relation bleiben erhalten R ⟕Bedingung S
* rechter äußerer Verbund (right outer join):   
  alle Tupel der rechten Relation bleiben erhalten R ⟖Bedingung S
* (vollständiger) äußerer Verbund (full outer join):   
  alle Tupel bleiben erhalten R ⟗Bedingung S

Beispiel:



## Zusammenfassung Relationenalgebra

* Anfragen lassen sich in Relationenalgebra ausdrücken
* Relationale Operatoren arbeiten auf Relationen (Tabellen)
* Mit Hilfe der Projektion und der Selektion kann man Spalten und Zeilen einer Tabelle auswählen
* Mehrere Tabellen können miteinander verknüpft werden:
  + Basisverknüpfung ist das kartesische Produkt
  + Ein Join filtert aus dem kartesischen Produkt die relevanten Einträge
* Es gibt die aus der Mathematik bekannten Mengenoperationen
* Es gibt häufig verschiedene Ausdrücke für dieselbe Anfrage. Dann sollte man einen möglichst günstigen Ausdruck wählen.
* Der Optimizer der Datenbank übernimmt diese Aufgabe!

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Left Outer Join: R ⟕Bedingung S

• Alle R-Tupel plus ggf. passende von S

Right Outer Join: R ⟖Bedingung S

• Alle S-Tupel plus ggf. passende von R

Full Outer Join: R ⟗Bedingung S

• Alle von R und S

**Relationenmodell**:

* Relationen mit Attributen (Tabellen)
* jedes Attribut hat einen Datentyp

**Relationale Algebra:**

* Operationen, um Daten aus den Tabellen abzufragen.
* Zentrale Operationen:
* **Selektion, Projektion**
* Kreuzprodukt -> **Verbund**
* Zentrale Operationen:
* Schlüssel zur Identifikation von Tupeln

**Primärschlüssel**

* Integritätsbedingungen

**Fremdschlüssel**

## Schlüssel

Insgesamt: „Superschlüssel => Schlüssel => Primärschlüssel“

**Superschlüssel** sind Teilmengen von Attributen, die ein Objekt eindeutig identifizieren

Ein **Schlüssel** ist ein Superschlüssel, der nicht verkleinert werden kann.

Unter allen Schlüsseln wird ein Schlüssel ausgewählt, der zur Identifikation verwendet wird. Dieser wird **Primärschlüssel** genannt. - Ein Schlüssel wird als Primärschlüssel ausgezeichnet.

Ein Primärschlüssel (primary key, auch PK abgekürzt) muss zwei wichtige Eigenschaften erfüllen:

* **Dauerhaft eindeutig**: Es gibt jede Ausprägung nur einmal (Schlüsseleigenschaft), auch in Zukunft!
* **Unveränderlich**: Die Attribute des PK einer Zeile ändern sich nicht

## Integritätsbedingungen

Integritätsbedingungen sind Bestimmungen, die eingehalten werden müssen, um die Korrektheit und die logische Richtigkeit der Daten zu sichern.

**Wahrheitsanforderungen**:

* Können nur durch Vergleich mit der Realität überprüft werden

*Beispiel: - Wohnt Albert Einstein wirklich in der Raumstr. 2, wie wir in der Tabelle Person gespeichert haben?*

**Logische Integritätsanforderungen:**

* Betreffen die Gestalt der einzelnen Tabellen bzw. Relationen und die Beziehungen zwischen den verschiedenen Relationen

*Beispiel: - Gibt es in der Tabelle Kunde überhaupt den Kunden mit der Id 16?*

### Logische Integritätsanforderungen

**Lokale Integritätsbedingungen**

* Bedingungen innerhalb **einer Relation**
* Es gibt nicht zwei identische Tupel in einer Tabelle:
* Eine Relation ist eine Menge.
* Jeder Wert eines Attributs gehört zu einem definierten Wertebereich:
* Für jedes Attribut wird ein Datentyp festgelegt
* Jedes Tupel muss eindeutig identifizierbar sein:
* Festlegung eines Primärschlüssels

**Globale Integritätsbedingungen**

* Bedingungen, die über den Bereich einer **Relation hinaus reichen**
* Ein referenziertes Tupel einer anderen Relation muss existieren:
* Referentielle Integrität

## Referentielle Integrität

Referentielle Integritätsbedingung

* Ein Tupel einer Relation, auf das sich ein Tupel einer anderen Relation bezieht, muss vorhanden sein.

Beispiel: In die Tabelle ANG\_PRO dürfen nur Personalnummern von Angestellten eingetragen werden, die auch in der Firma arbeiten

Ein **Fremdschlüssel** ist eine Attributmenge einer Relation, die in einer anderen Relation (Primär-)Schlüssel ist.

* Ein Bild, das Tisch enthält.

  Automatisch generierte BeschreibungAlle Attributwerte eines Fremdschlüssels tauchen in einer anderen Relation als Werte des (Primär-)Schlüssels auf.