

3. Grundlagen des Relationalen Datenmodells

- Grundkonzepte
- Relationale Invarianten
 - Primärschlüsselbedingung
 - Fremdschlüsselbedingung (referentielle Integrität)
 - Wartung der referentiellen Integrität
- Abbildung ERM / UML → RM
- Nachbildung von Generalisierung und Aggregation im RM
- Relationenalgebra
 - Mengenoperationen
 - relationale Operatoren: Selektion, Projektion, Join

Kapitel 4: Die Standard-Anfragesprache SQL

Kapitel 5: Logischer DB-Entwurf (Normalformenlehre)

Kapitel 6: Datendefinition und -kontrolle

DB-Anwendungsprogrammierung: ->DBS2



Lernziele

- Grundbegriffe des Relationenmodells
- Relationale Invarianten, insbesondere Vorkehrungen zur Wahrung der referentiellen Integrität
- Abbildung von ER/UML-Diagrammen in Relationenschema (und umgekehrt)
- Operationen der Relationenalgebra: Definition und praktische Anwendung



Relationenmodell - Übersicht

■ Entwicklungsetappen

- Vorschlag von E.F. Codd (Communications of the ACM 1970)
- 1975: Prototypen: System R (IBM Research), Ingres (Berkeley Univ.)
- seit 1980: kommerzielle relationale DBS

■ Datenstruktur: Relation (Tabelle)

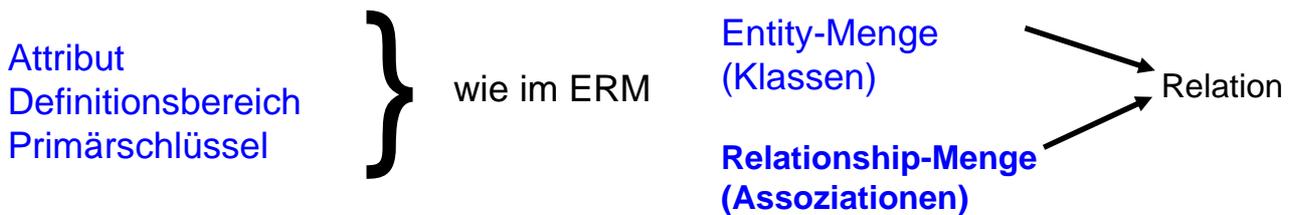
- einzige Datenstruktur (neben atomaren Werten)
- alle Informationen ausschließlich durch Werte dargestellt
- Integritätsbedingungen auf/zwischen Relationen: *relationale Invarianten*

■ Operatoren auf (mehreren) Relationen

- Vereinigung, Differenz
- Kartesisches Produkt
- Projektion
- Selektion
- zusätzlich: Änderungsoperationen (Einfügen, Löschen, Ändern)



Relationenmodell - Grundkonzepte



■ normalisierte Relation

$$R(A_1, A_2, \dots, A_n) \subseteq W(A_1) \times W(A_2) \times \dots \times W(A_n)$$

\downarrow \downarrow \downarrow
 D_i D_j D_k

- Relation = Untermenge des kartesischen Produktes der Attributwertebereiche
- nur einfache Attribute (atomare Werte) !

■ Darstellungsmöglichkeit für R:

n-spaltige Tabelle (*Grad* der Relation: n)

- *Kardinalität*: Anzahl der Sätze (Tupel)

■ Relation ist eine Menge: Garantie der Eindeutigkeit der Zeilen/Tupel über Primärschlüssel (ggf. mehrere Schlüsselkandidaten)



Normalisierte Relationen in Tabellendarstellung

FAK

<u>FNR</u>	FNAME	...
WI	Wirtschaftswiss.	...
MI	Math./Informatik	...

STUDENT

<u>MATNR</u>	SNAME	FNR	W-ORT
123 766	Coy	MI	Halle
654 711	Abel	WI	Leipzig
196 481	Maier	MI	Delitzsch
226 302	Schulz	MI	Leipzig

■ Grundregeln:

- Jede Zeile (Tupel) ist eindeutig und beschreibt ein Objekt (Entity) der Miniwelt
- Die Ordnung der Zeilen ist ohne Bedeutung
- Die Ordnung der Spalten ist ohne Bedeutung, da sie eindeutigen Namen (Attributnamen) tragen
- Jeder Datenwert innerhalb einer Relation ist ein atomares Datenelement
- Alle für Benutzer relevanten Informationen sind ausschließlich durch Datenwerte ausgedrückt

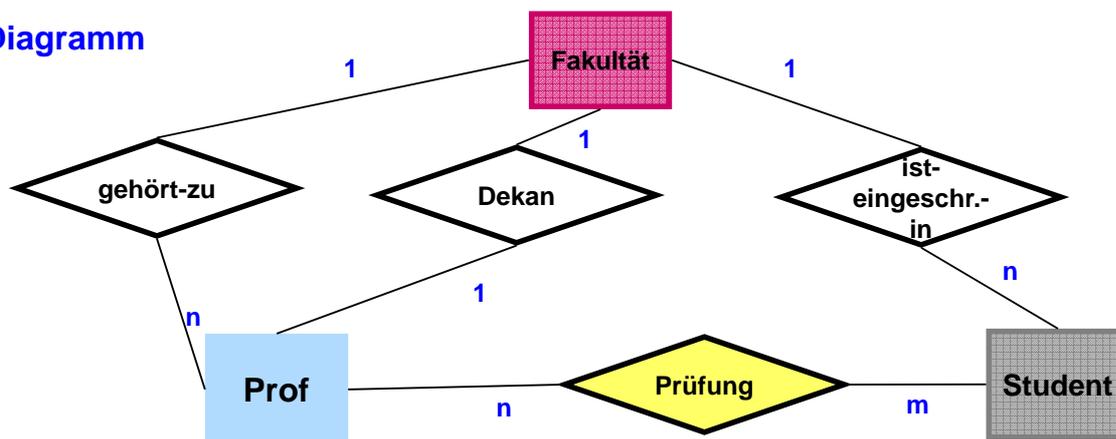
■ Darstellung von Beziehungen durch Fremdschlüssel (foreign key)

- Attribut, das in Bezug auf den Primärschlüssel einer anderen (oder derselben) Relation definiert ist (gleicher Definitionsbereich)

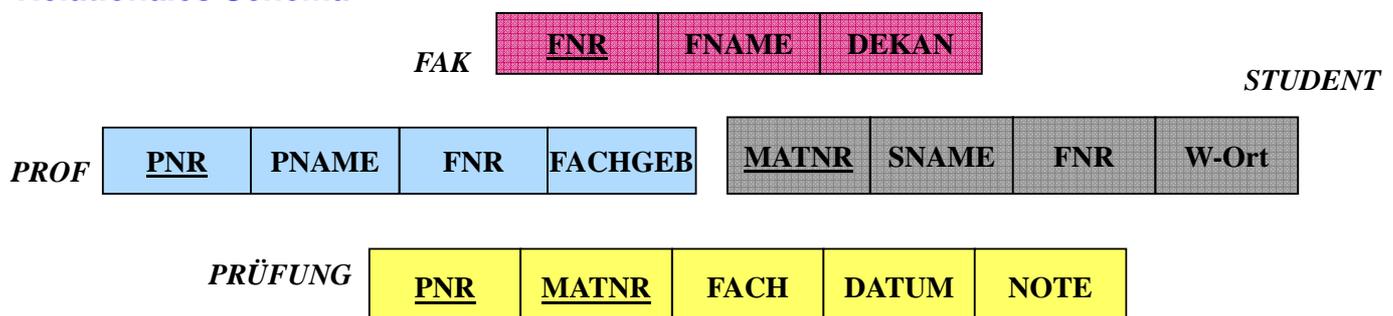


Anwendungsbeispiel

ER-Diagramm



Relationales Schema



Relationale Invarianten

- inhärente Integritätsbedingungen des Relationenmodells (Modellbedingungen)

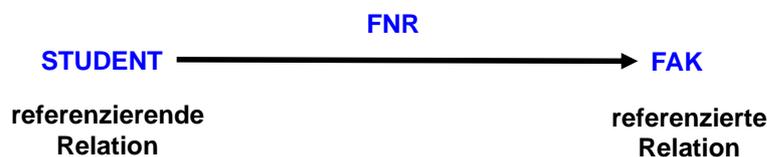
1. Primärschlüsselbedingung (Entity-Integrität)

- Eindeutigkeit des Primärschlüssels
- keine Nullwerte!

2. Fremdschlüsselbedingung (referentielle Integrität):

- zugehöriger Primärschlüssel muss existieren
- d.h. zu jedem Wert (ungleich Null) eines Fremdschlüsselattributs einer Relation R2 muss ein gleicher Wert des Primärschlüssels in irgendeinem Tupel von Relation R1 vorhanden sein

- Graphische Notation:



Relationale Invarianten (2)

- Fremdschlüssel und zugehöriger Primärschlüssel tragen wichtige interrelationale (bzw. intrarelationale) Informationen
 - gleicher Wertebereich
 - gestatten Verknüpfung von Relationen
- Fremdschlüssel
 - können Nullwerte aufweisen, wenn sie nicht Teil eines Primärschlüssels sind.
 - Fremdschlüssel ist „zusammengesetzt“, wenn zugehöriger Primärschlüssel „zusammengesetzt“ ist
- eine Relation kann mehrere Fremdschlüssel besitzen, die die gleiche oder verschiedene Relationen referenzieren
- Zyklen sind möglich (*geschlossener referentieller Pfad*)
- eine Relation kann zugleich referenzierende und referenzierte Relation sein (selbstreferenzierende Tabelle)

Relationale Invarianten (3)

■ DDL-Spezifikation in SQL bei CREATE TABLE

- Primary KEY- und FOREIGN KEY-Klauseln
- Angaben für Attributdefinition :
 - Attributname sowie Datentyp
 - Default-Werte
 - Eindeutigkeit (UNIQUE bzw. PRIMARY KEY)
 - FOREIGN-KEY-Klausel
 - Verbot von Nullwerten (NOT NULL)

```
CREATE TABLE STUDENT
(MATNR      INT,
 SNAME     VARCHAR (50)  NOT NULL,
 FNR       INT,

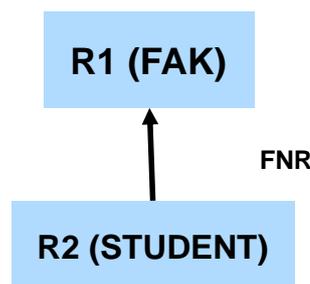
 PRIMARY KEY (MATNR),
 FOREIGN KEY (FNR) REFERENCES FAK )
```

```
CREATE TABLE FAK
(FNR      INT      PRIMARY KEY,
 FNAME   VARCHAR(50) NOT NULL,
 DEKAN   INT      REFERENCES PROF ... )
```



Wartung der referentiellen Integrität

■ Gefährdung bei INSERT, UPDATE, DELETE



- Fall 0: INSERT auf R1, DELETE auf R2
- Fall 1: INSERT bzw. UPDATE auf FS der referenzierenden (abhängigen) Relation R2: Ablehnung falls kein zugehöriger PS-Wert in referenzierter Relation R1 besteht
- Fall 2: DELETE auf referenzierter Relation R1 bzw. UPDATE auf PS von R1. Unterschiedliche Folgeaktionen auf referenzierender Relation R2 möglich, um referentielle Integrität zu wahren



Wartung der referentiellen Integrität (2)

- SQL-Standard erlaubt Spezifikation der referentiellen Aktionen für jeden Fremdschlüssel
- Sind Nullwerte verboten?
 - **NOT NULL**
- Löschregel für Zielrelation (referenzierte Relation R1):
 - ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT }**
- Änderungsregel für Ziel-Primärschlüssel (Primärschlüssel oder Schlüsselkandidat):
 - ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT }**
- Dabei bedeuten:
 - **NO ACTION** (Voreinstellung): Operation wird nur zugelassen, wenn keine zugehörigen Sätze (Fremdschlüsselwerte) vorhanden sind. Es sind folglich keine referentiellen Aktionen auszuführen
 - **CASCADE**: Operation „kaskadiert“ zu allen zugehörigen Sätzen
 - **SET NULL**: Fremdschlüssel wird in zugehörigen Sätzen zu “Null” gesetzt
 - **SET DEFAULT**: Fremdschlüssel wird auf einen benutzerdefinierten Default-Wert gesetzt



Anwendungsbeispiel

```
CREATE TABLE TEIL ( TNR INT PRIMARY KEY,
                   BEZEICHNUNG ... )

CREATE TABLE LIEFERANT (LNR INT PRIMARY KEY,
                       LNAME ... )

CREATE TABLE LIEFERUNG (TNR INT, LNR INT, DATUM ...
                        PRIMARY KEY (TNR, LNR),
                        FOREIGN KEY (TNR) REFERENCES TEIL, NOT NULL,
                        ON DELETE OF TEIL NO ACTION
                        ON UPDATE OF TEIL.TNR CASCADE,
                        FOREIGN KEY (LNR) REFERENCES LIEFERANT, NOT NULL,
                        ON DELETE OF LIEFERANT NO ACTION,
                        ON UPDATE OF LIEFERANT.LNR CASCADE )
```

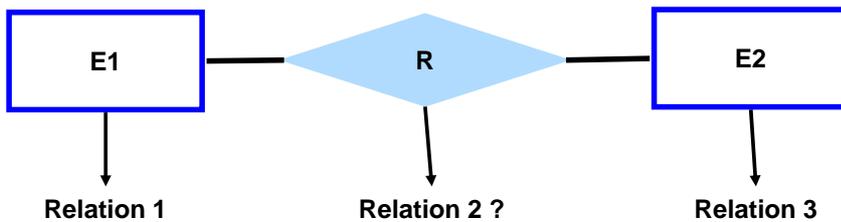
TEIL

LIEFERANT

LIEFERUNG



Abbildung ERM / UML -> RM



■ Kriterien

- Informationserhaltung
- Minimierung der Redundanz
- Minimierung des Verknüpfungsaufwandes
- Natürlichkeit der Abbildung
- keine Vermischung von Objekten
- Verständlichkeit

■ Regeln:

- Jeder Entity-Typ *muss* als eigenständige Relation (Tabelle) mit einem eindeutigen Primärschlüssel definiert werden.
- Relationship-Typen können als eigene Relationen definiert werden, wobei die Primärschlüssel der zugehörigen Entity-Typen als Fremdschlüssel zu verwenden sind.

2 Entitymengen mit n:1 - Verknüpfung



1.) Verwendung von drei Relationen

ABT (ANR, ANAME, ...)
PERS (PNR, PNAME, ...)
ABT-ZUGEH (PNR, ANR,)

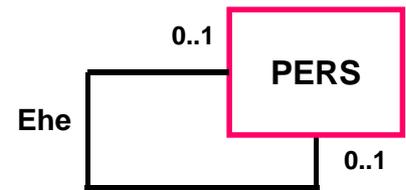
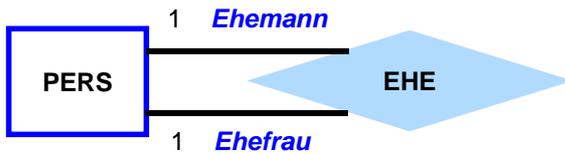
2.) Besser: Verwendung von zwei Relationen

ABT (ANR, ANAME, ...)
PERS (PNR, PNAME, ..., ANR)

■ Regel: n:1-Beziehungen lassen sich ohne eigene Relation darstellen.

- Hierzu wird in der Relation, der pro Tupel maximal 1 Tupel der anderen Relation zugeordnet ist, der Primärschlüssel der referenzierten Relation als Fremdschlüssel verwendet

1 Entitymenge mit 1:1 Verknüpfung



1.) Verwendung von zwei Relationen

PERS (PNR, PNAME, ...)
 EHE (PNR, GATTE, ...)

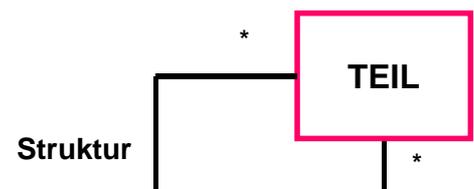
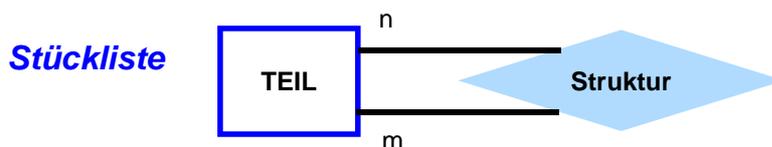
2.) Verwendung von einer Relation

PERS (PNR, PNAME, ..., GATTE)

■ Unterscheidung zu n:1 ?

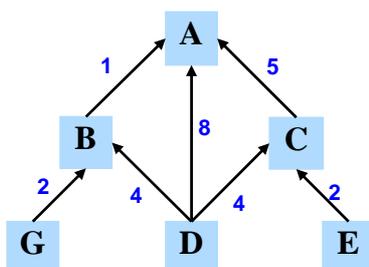


1 Entitymenge mit m:n-Verknüpfung



Darstellungsmöglichkeit im RM:

TEIL (TNR, BEZ, MAT, BESTAND)
STRUKTUR (OTNR, UTNR, ANZAHL)



Teil			
TNR	BEZ	MAT	BESTAND
A	Getriebe	-	10
B	Gehäuse	Alu	0
C	Welle	Stahl	100
D	Schraube	Stahl	200
E	Kugellager	Stahl	50
F	Scheibe	Blei	0
G	Schraube	Chrom	100

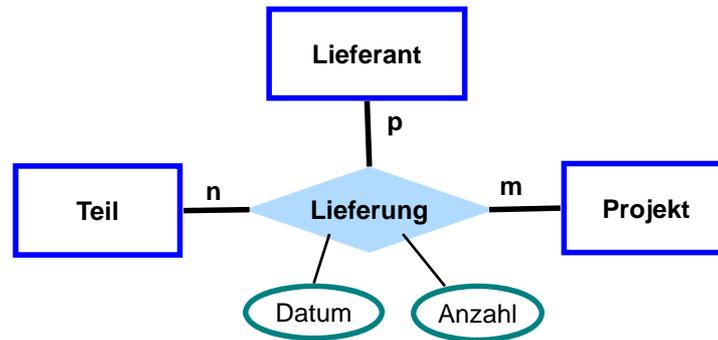
Struktur

OTNR	UTNR	Anzahl
A	B	1
A	C	5
A	D	8
B	D	4
B	G	2
C	D	4
C	E	2

- **Regel:** Ein n:m-Relationship-Typ muss durch eine eigene Relation dargestellt werden. Die Primärschlüssel der zugehörigen Entitymengen treten als Fremdschlüssel auf.

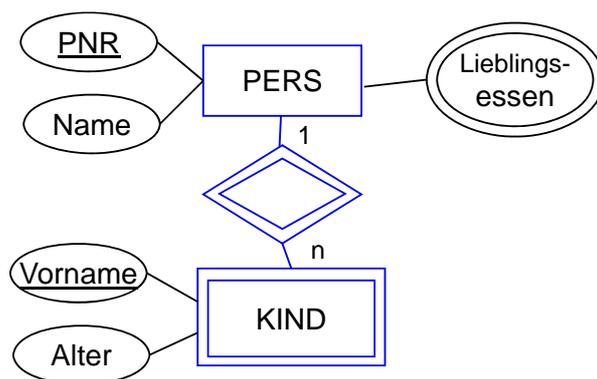


3 Entitätsmengen mit m:n-Verknüpfung



LIEFERANT (LNR, LNAME, LORT, ...)
 PROJEKT (PRONR, PRONAME, PORT, ...)
 TEIL (TNR, TBEZ, GEWICHT, ...)
 LIEFERUNG (

Abbildung mehrwertiger Attribute bzw. schwacher Entitätsmengen



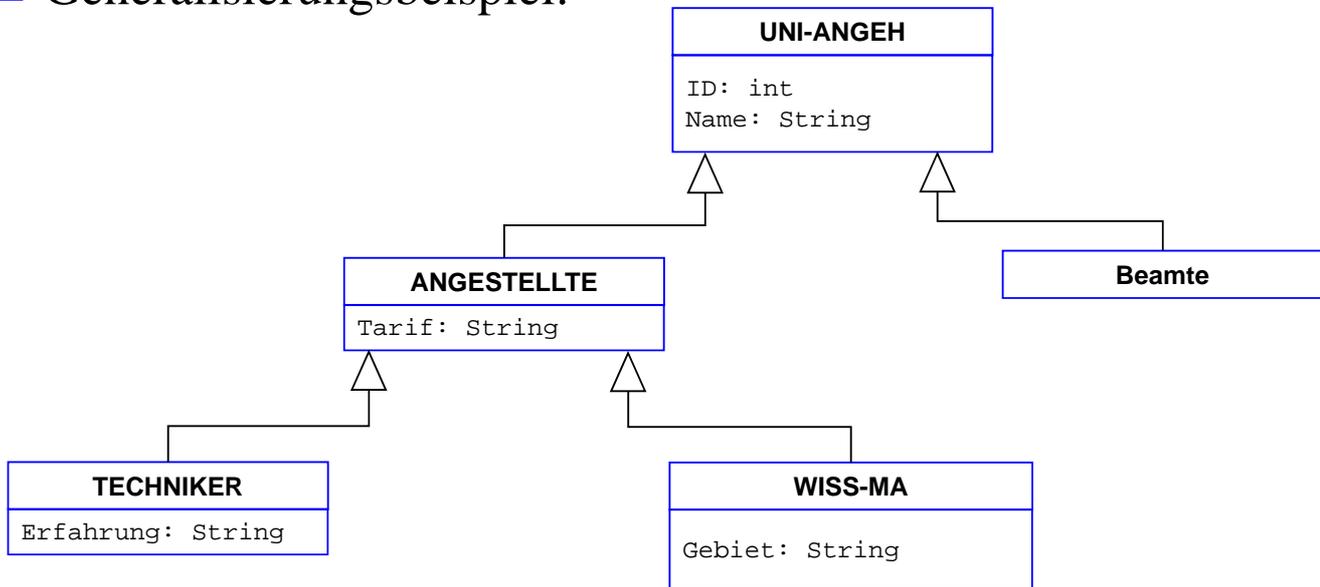
Darstellungsmöglichkeit im RM

PERS (PNR, NAME ...)

Abbildungen von Generalisierung und Aggregation im RM

- RM sieht keine Unterstützung der Abstraktionskonzepte vor
 - keine Maßnahmen zur Vererbung (von Struktur, Integritätsbedingungen, Operationen)
 - „Simulation“ der Generalisierung und Aggregation eingeschränkt möglich

■ Generalisierungsbeispiel:



Generalisierung – relationale Sicht

- pro Klasse 1 Tabelle (3 Varianten) oder insgesamt nur 1 Tabelle
- Lösungsmöglichkeit 1: vertikale Partitionierung
 - jede Instanz wird entsprechend der Klassenattribute in der IS-A-Hierarchie zerlegt und in Teilen in den zugehörigen Klassen (Relationen) gespeichert.
 - nur das ID-Attribut wird dupliziert

UNI-ANGEH	
ID	Name
007	Garfield
123	Donald
333	Daisy
765	Grouch
111	Ernie

ANGESTELLTE	
ID	Tarif
007	E14
123	E11
333	E9
765	E13

WISS-MA	
ID	Gebiet
007	XML
765	Cloud

TECHNIKER	
ID	Erfahrung
123	Linux

■ Eigenschaften

- geringfügig erhöhte Speicherkosten, aber hohe Aufsuch- und Aktualisierungskosten
- Integritätsbedingungen: $TECHNIKER.ID \subseteq ANGESTELLTE.ID$, usw.
- Instanzenzugriff erfordert implizite oder explizite Verbundoperationen
- Beispiel: Finde alle TECHNIKER-Daten



Generalisierung – relationale Sicht (2)

■ Lösungsmöglichkeit 2: horizontale Partitionierung

- jede Instanz ist genau einmal und vollständig in ihrer „Hausklasse“ gespeichert.
- keinerlei Redundanz

UNI-ANGEH

<u>ID</u>	Name
111	Ernie

WISS-MA

<u>ID</u>	Gebiet	Name	Tarif
007	XML	Garfield	E14
765	Cloud	Grouch	E13

ANGESTELLTE

<u>ID</u>	Name	Tarif
333	Daisy	E9

TECHNIKER

<u>ID</u>	Erfahrung	Name	Tarif
123	Linux	Donald	E11

■ Eigenschaften

- niedrige Speicherkosten und keine Änderungsanomalien
- Eindeutigkeit von ID zwischen Relationen aufwändiger zu überwachen
- Retrieval kann rekursives Suchen in Unterklassen erfordern.
- explizite Rekonstruktion durch Relationenoperationen (π , \cup)

=> Beispiel: Finde alle ANGESTELLTE



Generalisierung – relationale Sicht (3)

■ Lösungsmöglichkeit 3: volle Redundanz

- eine Instanz wird wiederholt in jeder Klasse, zu der sie gehört, gespeichert.
- sie besitzt dabei die Werte der Attribute, die sie geerbt hat, zusammen mit den Werten der Attribute der Klasse

UNI-ANGEH

<u>ID</u>	Name
007	Garfield
123	Donald
333	Daisy
765	Grouch
111	Ernie

ANGESTELLTE

<u>ID</u>	Name	Tarif
007	Garfield	E14
123	Donald	E11
333	Daisy	E9
765	Grouch	E13

WISS-MA

<u>ID</u>	Name	Tarif	Gebiet
007	Garfield	E14	XML
765	Grouch	E13	Cloud

TECHNIKER

<u>ID</u>	Name	Tarif	Erfahrung
123	Donald	E11	Linux

■ Eigenschaften

- hoher Speicherplatzbedarf und Auftreten von Änderungsanomalien.
- einfaches Retrieval, da nur die Zielklasse (z. B. ANGESTELLTE) aufgesucht werden muss



Generalisierung: Verfahrensvergleich

	Vertikale Partitionierung	Horizontale Partitionierung	Volle Redundanz
Änderungen			
Lesen			



„Wide Table“-Realisierung

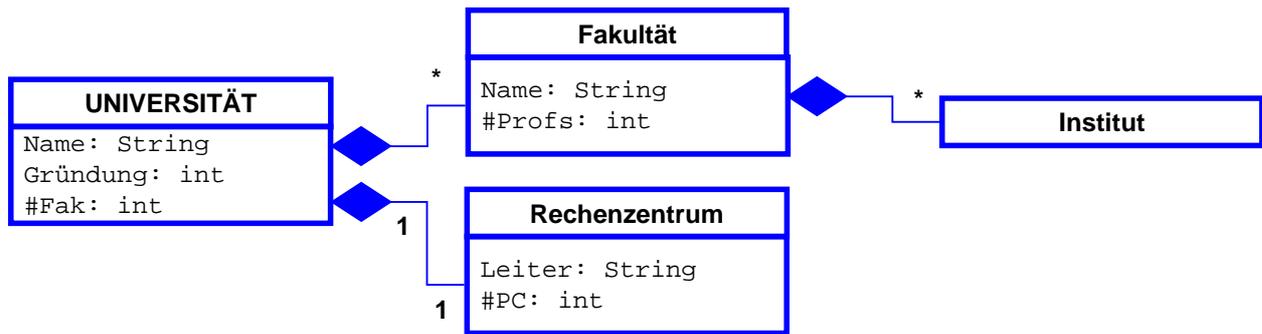
- Verwendung einer gemeinsamen Tabelle pro Klassenhierarchie
 - Verwendung eines Diskriminatorattributs zur Angabe der Klasse
- Minimale Redundanz ermöglicht effiziente Verwaltung
- Keine explizite Modellierung der Klassenhierarchie und Is-A-Semantik
 - Nutzer müssen Abhängigkeiten bei Lese- und Änderungsoperationen selbst beachten

UNI-ANGEH

<u>ID</u>	Name	Typ	Tarif	Gebiet	Erfahrung	...
007	Garfield	WISS-MA	E14	XML	NULL	
123	Donald	TECHNIKER	E11	NULL	Linux	
333	Daisy	ANGESTELLTE	E9	NULL	NULL	
765	Grouch	WISS-MA	E13	Cloud	NULL	
111	Ernie	UNI-ANGEH	NULL	NULL	NULL	



Aggregation – relationale Sicht



Universität

ID	Name	Gründung	#Fak
UL	Univ. Leipzig	1409	14
TUD	TU Dresden	1828	14

Fakultät

FID	Uni	Name	#Profs
123	UL	Theologie	14
132	UL	Mathe/Informatik	28

Institut

ID	FID	Name
1234	123	Neutestamentliche Wissenschaft
1235	123	Alttestamentliche Wissenschaft
1236	123	Praktische Theologie
1322	132	Informatik

- Komplexe Objekte erfordern Zerlegung über mehrere Tabellen

Sprachen für das Relationenmodell

- Datenmodell = Datenobjekte + Operatoren
- im RM wird vereinheitlichte Sprache angestrebt für:
 - Anfragen (Queries) im 'Stand-Alone'-Modus
 - Datenmanipulation und Anfragen eingebettet in eine Wirtssprache
 - Datendefinition
 - Zugriffs- und Integritätskontrolle
 - Unterstützung verschiedener Benutzerklassen:
Anwendungsprogrammierer, DBA, gelegentliche Benutzer
- Verschiedene Grundtypen von Sprachen
 - Formale Ansätze: Relationenalgebra und Relationenkalkül
 - Abbildungsorientierte Sprachen (z. B. SQL)
 - Graphik-orientierte Sprachen (z. B. Query-by-Example)

Relationenalgebra

- **Algebra**: ein System, das aus einer nichtleeren Menge und einer Familie von Operationen besteht

- Relationen sind Mengen
- Operationen auf Relationen arbeiten auf einer oder mehreren Relationen als Eingabe und erzeugen eine Relation als Ausgabe (Abgeschlossenheitseigenschaft)
=> mengenorientierte Operationen

- Operationen:

Klassische Mengenoperationen:

- Vereinigung
- Differenz
- kartesisches Produkt
- Durchschnitt (ableitbar)

Relationenoperationen:

- Restriktion (Selektion)
- Projektion
- Verbund (Join) (ableitbar)
- Division (ableitbar)



Selektion (Restriktion)

- Auswahl von Zeilen einer Relation über Prädikate, abgekürzt σ_P

$$\sigma_P(R) = \{ t \mid t \in R \wedge P(t) \}$$

P = log. Formel (ohne Quantoren !) zusammengestellt aus:

- Operanden: Attributnamen oder Konstanten
- Vergleichsoperatoren $\theta \in \{ <, =, >, \leq, \neq, \geq \}$
- logische Operatoren: \vee, \wedge, \neg

- Beispiele:

$\sigma_{\text{GEHALT} < \text{PROVISION}}(\text{PERS})$

$\sigma_{\text{BERUF} = \text{'Programmierer'} \wedge \text{ALTER} < 50}(\text{PERS})$

- Eigenschaften

- $\text{grad}(\sigma_P(R)) = \text{grad}(R)$
- $\text{card}(\sigma_P(R)) \leq \text{card}(R)$



Projektion

- Auswahl der Spalten (Attribute) A_1, A_2, \dots, A_k aus einer Relation R (Grad $n \geq k$)

$$\pi_{A_1, A_2, \dots, A_k}(R) = \{ p \mid \exists t \in R : p = \langle t[A_1], \dots, t[A_k] \rangle \}$$

- Beispiel:

$$\pi_{\text{NAME, GEHALT}}(\text{PERS})$$

- Eigenschaften:

- Wichtig: Duplikate werden entfernt ! (Mengeeigenschaft)
- $\text{grad}(\pi_A(R)) \leq \text{grad}(R)$
- $\text{card}(\pi_A(R)) \leq \text{card}(R)$



Relationenalgebra: Beispiel-DB

<u>ANR</u>	ANAME	ORT
K51	Planung	Leipzig
K53	Einkauf	Frankfurt
K55	Vertrieb	Frankfurt

<u>PNR</u>	Name	Alter	Gehalt	ANR	MNR
406	Abel	47	50700	K55	123
123	Schulz	32	43500	K51	-
829	Müller	36	40200	K53	406
574	Schmid	28	36000	K55	123

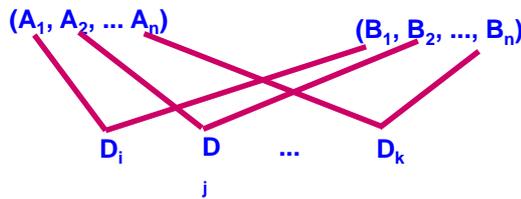
- Finde alle Angestellten aus Abteilung K55, die mehr als 40.000 verdienen
- Finde alle Abteilungsorte
- Finde den Abteilungsnamen von Abteilung K53



Klassische Mengenoperationen

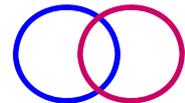
- Voraussetzung: *Vereinigungsverträglichkeit* der beteiligten Relationen:

Gleicher Grad - Gleiche Bereiche: $\Rightarrow W(A_i) = W(B_i) \quad : \quad i = 1, n$



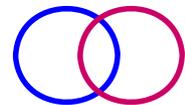
- Vereinigung: $R \cup S = \{ t \mid t \in R \vee t \in S \}$

- $\text{card}(R \cup S) \leq \text{card}(R) + \text{card}(S)$



- Differenz: $R - S = \{ t \mid t \in R \wedge t \notin S \}$

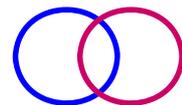
- $\text{card}(R - S) \leq \text{card}(R)$



- Durchschnitt:

$$R \cap S = R - (R - S) = \{ t \mid t \in R \wedge t \in S \}$$

- $\text{card}(R \cap S) \leq \min(\text{card}(R), \text{card}(S))$



(Erweitertes) Kartesisches Produkt

R (Grad r) und S (Grad s) beliebig

$$R \times S = \{ k \mid \exists x \in R, y \in S : k = x \mid y \}$$

Beachte: $k = x \mid y = \langle x_1, \dots, x_r, y_1, \dots, y_s \rangle$

nicht $\langle \langle x_1, \dots, x_r \rangle, \langle y_1, \dots, y_s \rangle \rangle$ wie übliches kart. Produkt

- $\text{grad}(R \times S) = \text{grad}(R) + \text{grad}(S)$; $\text{card}(R \times S) = \text{card}(R) * \text{card}(S)$

Beispiel

R			S		
A	B	C	D	E	F
a	γ	1	b	γ	3
d	α	2	d	α	2
b	β	3			

R x S					

Allgemeiner Verbund (Theta-Join)

■ grob:

- Kartesisches Produkt zwischen zwei Relationen R (Grad r) und S (Grad s).
- eingeschränkt durch Θ -Bedingungen zwischen Attribut A von R und Attribut B von S.

■ Θ -Verbund zwischen R und S:

$$R \bowtie_{A \Theta B} S = \sigma_{A \Theta B} (R \times S)$$

mit arithm. Vergleichsoperator $\Theta \in \{<, =, >, \leq, \neq, \geq\}$

■ Bemerkungen:

- **Gleichverbund (Equijoin):** $\Theta = '='$:



Natürlicher Verbund (Natural Join)

■ grob: Gleichverbund über alle gleichen Attribute und Projektion über die verschiedenen Attribute

■ Natürlicher Verbund zwischen R und S:

gegeben: $R(A_1, A_2, \dots, A_{r-j+1}, \dots, A_r), S(B_1, B_2, \dots, B_j, \dots, B_s)$

o.B.d.A.: (sonst. Umsortierung: $B_1 = A_{r-j+1}, B_2 = A_{r-j+2} \dots B_j = A_r$)

$$R \bowtie S = \pi_{A_1, \dots, A_r, B_{j+1}, \dots, B_s} \sigma_{(R.A_{r-j+1} = S.B_1) \wedge \dots \wedge (R.A_r = S.B_j)} (R \times S)$$

\bowtie Zeichen für Natural Join $\Rightarrow \Theta = '='$

- Join-Attribute sind durch Übereinstimmungsbedingung gegeben

R		
A	B	C
a ₁	b ₁	c ₁
a ₂	b ₂	c ₂

A

S		
C	D	E
c ₁	d ₁	e ₁
c ₃	d ₂	e ₂

=

Resultat				
A	B	C	D	E
a ₁	b ₁	c ₁	d ₁	e ₁



Join-Beispiel

ANR	ANAME	ORT
K51	Planung	Leipzig
K53	Einkauf	Frankfurt
K55	Vertrieb	Frankfurt

PNR	Name	Alter	Gehalt	ANR	MNR
406	Abel	47	50700	K55	123
123	Schulz	32	43500	K51	-
829	Müller	36	40200	K53	406
574	Schmid	28	36000	K55	123

- Finde alle Angestellten (PNR, ALTER, ANAME), die in einer Abteilung in Frankfurt arbeiten und älter als 30 sind.



Semi-Join

- Ergebnisbeschränkung des Gleichverbundes auf eine der beiden Eingaberelationen

$$S \bowtie_R R = \pi_{S\text{-Attribute}} (S \bowtie R)$$

$$S \bowtie_S R = \pi_{R\text{-Attribute}} (S \bowtie R)$$

S			E	R			=	Resultat		
A	B	C		C	D	E		A	B	C
a ₁	b ₁	c ₁		c ₁	d ₁	e ₁		a ₁	b ₁	c ₁
a ₂	b ₂	c ₂		c ₃	d ₂	e ₂				

S			F	R			=	Resultat		
A	B	C		C	D	E		C	D	E
a ₁	b ₁	c ₁		c ₁	d ₁	e ₁		c ₁	d ₁	e ₁
a ₂	b ₂	c ₂		c ₃	d ₂	e ₂				



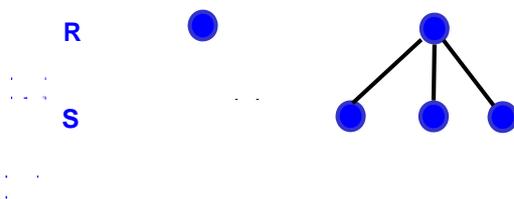
Äußerer Verbund (Outer Join)

■ Ziel: Verlustfreier Verbund soll erzwungen werden

- Ein Gleichverbund zwischen R und S heißt *verlustfrei*, wenn alle Tupel von R und S am Verbund teilnehmen. Die inverse Operation Projektion erzeugt dann wieder R und S (**lossless join**).

■ Bisher: $R \bowtie S$ liefert nur „vollständige Objekte“

- Es sollen aber auch Teilobjekte als Ergebnis geliefert werden (z. B. komplexe Objekte)



- Trick: Einfügen künstlicher Verbundpartner

Outer Join (2)

- Def.: Seien A die Verbundattribute, $\{\equiv\}$ der undefinierte Wert und

$$R' := R \cup ((\pi_A(S) - \pi_A(R)) \times \{\equiv\} \times \dots \times \{\equiv\})$$

$$S' := S \cup ((\pi_A(R) - \pi_A(S)) \times \{\equiv\} \times \dots \times \{\equiv\})$$

Äußerer Gleichverbund

$$R \bowtie_{R.A=S.A} S := R' \bowtie_{R'.A=S'.A} S'$$

Äußerer natürlicher Verbund

$$R \bowtie_{R.A=S.A} S := R' \bowtie_{R'.A=S'.A} S'$$

■ Linker und rechter äußerer Gleichverbund

- Nur die linke bzw. rechte Argumentrelation bleibt verlustfrei (Einfügen künstlicher Verbundpartner in rechter bzw. linker Eingaberelation)

$$R \bowtie_{R.A=S.A} S := R \bowtie_{R.A=S'.A} S'$$

Linker äußerer Gleichverbund

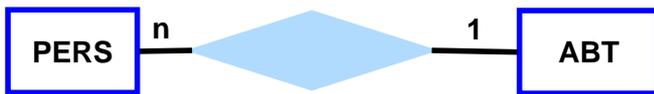
$$R \bowtie_{R.A=S.A} S := R' \bowtie_{R'.A=S.A} S$$

Rechter äußerer Gleichverbund

■ Verallgemeinerung auf 2 (oder mehr) Joins, z.B. $R \bowtie S \bowtie T$

- selbst isolierte Tupel können zu einem vollständigen Pfad expandiert werden

Outer Join - Beispiel



PERS

PNR	ANR ...
P1	A1
P2	A1
P3	A2
P4	-
P5	-

ABT

ANR	ANAME ...
A1	A
A2	B
A3	C

PERS ⋈ ABT

PNR	ANR	ANAME ...

PERS ⋈_{ABT} ABT

PNR	ANR	ANAME ...

PERS ⋈_{ABT} ABT

PNR	ANR	ANAME ...

PNR	ANR	ANAME ...

PERS ⋈_{ABT} ABT

3 - 39



Division

- Beantwortung von Fragen, bei denen eine „ganze Relation“ zur Qualifikation herangezogen wird
- Simulation des Allquantors \Rightarrow ein Tupel aus R steht mit allen Tupeln aus S in einer bestimmten Beziehung

Definition

Voraussetzung: S-Attribute \subset R-Attribute

Sei R vom Grad r und S vom Grad s, $r > s$

t sei (r-s)-Tupel, u sei s-Tupel;

Dann gilt:

$$R \div S = \{ t \mid \forall u \in S : tu \in R \}$$

grad (R \div S) =

card (R \div S) =



Division (2)

■ Beispiel

LPT		
LNR	PNR	TNR
L1	P1	T1
L1	P2	T1
L2	P1	T1
L2	P1	T2
L2	P2	T1

÷

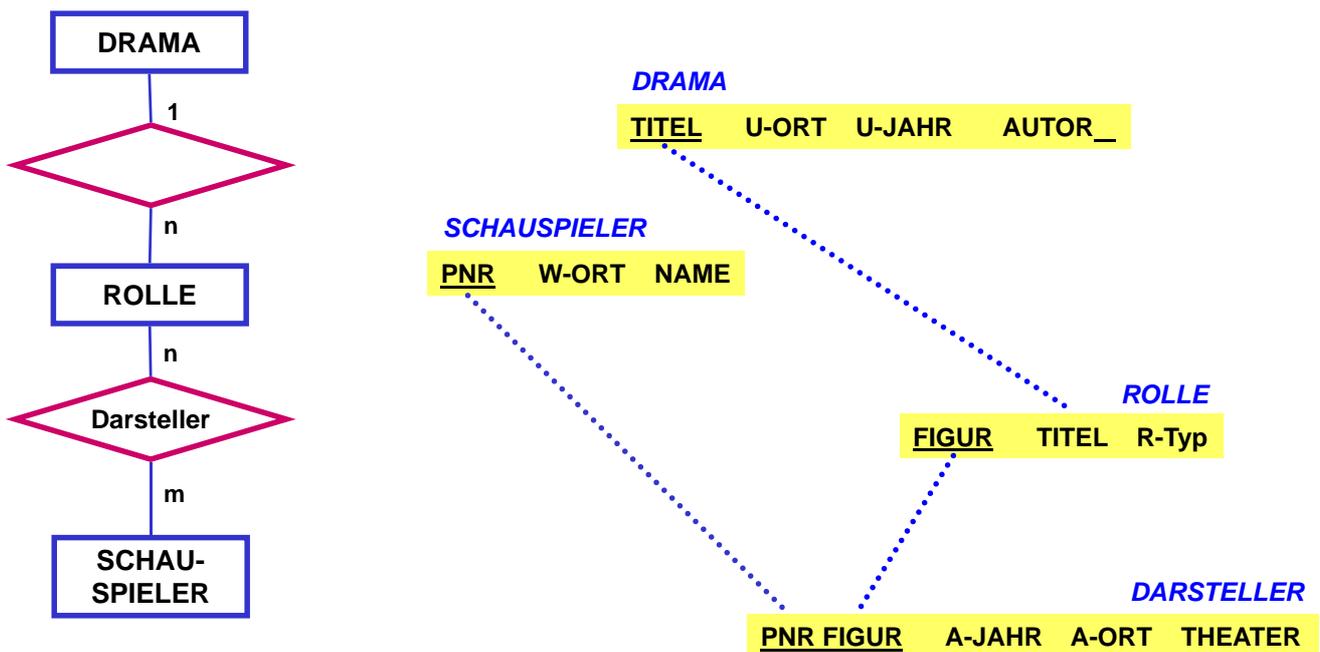
PT	
PNR	TNR
P1	T1
P1	T2
P2	T1

- Welche Lieferanten beliefern alle Projekte?
- Welche Lieferanten liefern alle Teile?

■ Zusammenhang zwischen Division und kartesischem Produkt:
 $(R \times S) \div S = R$



Beispiel-DB: Bühne



Beispielanfragen

- Welche Darsteller (PNR) haben im Schauspielhaus gespielt?
- Finde alle Schauspieler (NAME, W-ORT), die einmal im 'Faust' mitgespielt haben.
- Finde alle Schauspieler (NAME), die bei in Weimar uraufgeführten Dramen an ihrem Wohnort als 'Held' mitgespielt haben
- Finde die Schauspieler (PNR), die nie gespielt haben
- Finde alle Schauspieler (NAME), die alle Rollen gespielt haben.



Zusammenfassung Relationenalgebra

- saubere mathematische Definition
- mengenorientierte Operationen
- keine Änderungsoperationen!
- für Laien nicht leicht verständlich

