


CAS Information Engineering

Modul: Datenbanken & Data Warehousing

Dozent Manuel Kessler

DB-2 2024-09-09

Agenda

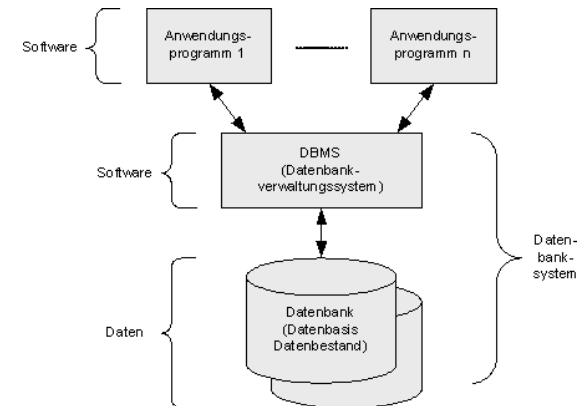
- 0900 – 0930 Rückblick, Repetition / Ergänzung konzeptioneller Entwurf
- 0930 – 1015 Grundbegriffe des relationalen Datenmodelles

- 1030 – 1100 Konzeptioneller Entwurf → Logischer Entwurf
- 1100 – 1200 Grundlagen der relationalen Algebra
- 1200 – 1230 Selbständige Übungen



Rückblick

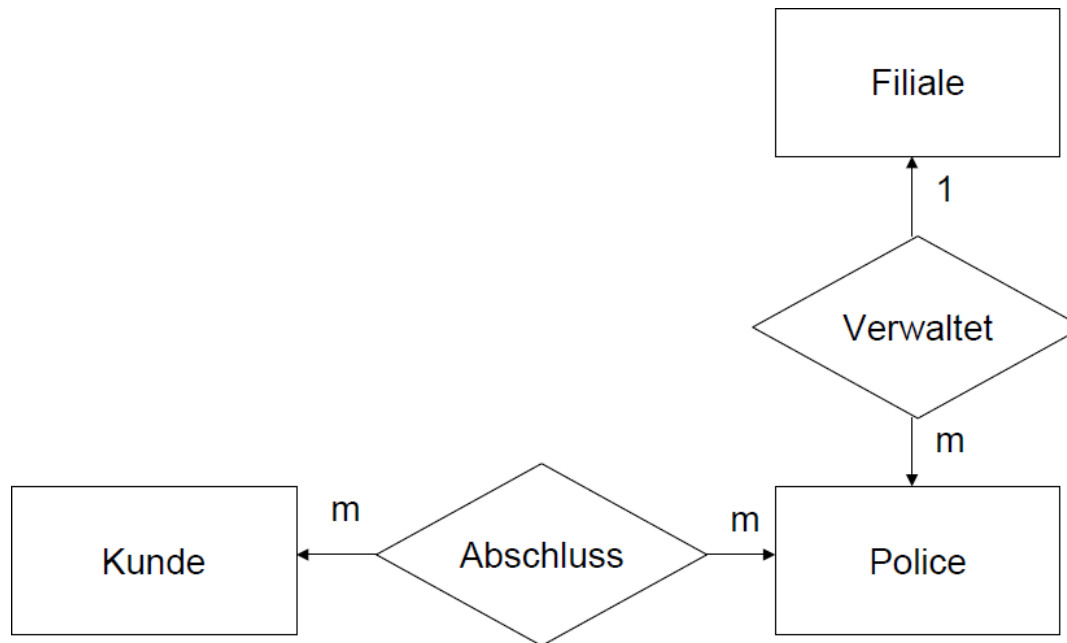
Was bisher geschah...

- Begriffshierarchie: Zeichen – **Daten** – **Information** – Wissen
- Datenarten:
 - **Strukturiert** → relationale Technologien
 - Semi-strukturiert → XML-Technologien
 - Unstrukturiert → proprietäre Technologien
- Architektur & Aufgaben eines **Datenbanksystems**
- Vorgehen beim Datenbankentwurf:
 - **Konzeptioneller** Entwurf (z.B. mit ERM)
 - **Logischer** Entwurf
 - Physischer Entwurf (im CAS INFE nicht behandelt)



Was bisher geschah...

- ERM:
 - Methode, um die Daten eines Anwendungsbereiches **grafisch** zu beschreiben.
 - Dient primär der **Kommunikation**, zeigt **Zusammenhänge** zwischen den Daten.
 - Positiv: Nur wenige Konzepte, **einfach zu verstehen**.
 - Negativ: Viele verschiedene Varianten in der Praxis, kann nicht alles Wichtige grafisch beschreiben, wird rasch unübersichtlich bei grossen Modellen.
- Hier eingeführt: Einfacher «Dialekt», mit günstigen Eigenschaften für die praktische Umsetzung in relationalen Datenbanksystemen:
 - Direkte Abbildung ohne Transformationen
 - Unterstützung von Mehrfachbeziehungen
 - Unterstützung von 1-1-Beziehungen
 - Realisierung der Kardinalitäten via Schlüssel (geprüft durch das RDBMS)
 - Siehe auch: DB-2-ER-Bachman-vs-Chen.pdf (MOODLE)



Repetition / Ergänzung konzeptioneller Entwurf

Repetition konzeptioneller Entwurf

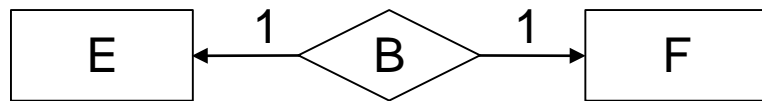
- Entwurfsmethode – wichtige Begriffe:
 - Entitätstypen: unabhängige, abhängige, ISA-/ID-abhängige, zusammengesetzte
 - Beziehungstypen
 - Attribute
 - Schlüssel: Schlüssel, Primärschlüssel, Fremdschlüssel
 - Kardinalitäten: 1, m, ID, ISA (führen zu Schlüsselbedingungen)

Repetition konzeptioneller Entwurf

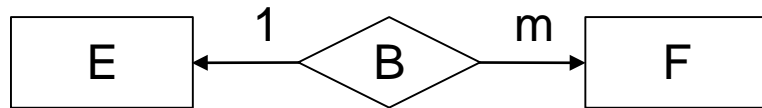
- Entwurfsmethode – Schlüssel:
 - Entitäts- und Beziehungstypen haben **Schlüssel** (mindestens einen, oft auch mehrere).
 - Schlüssel: **Attribut** oder **Attributkombination** die Entitäten bzw. Beziehungen **eindeutig identifizieren**.
 - Wenn Attribute eines anderen Entitäts- bzw. Beziehungstypen darauf verweisen («Fremdschlüssel») dann – und nur dann – sprechen wir von **Primärschlüssel** (und unterstreichen die entsprechenden Attribute im ER-Diagramm). Primär- und Fremdschlüssel sollten nach Möglichkeit gleich benannt werden. Sonstige Schlüssel werden **als Text** auf dem Diagramm notiert.
 - Jeder Entitäts- / zus. ges. Entitätstyp hat **höchstens einen Primärschlüssel**.
 - **Fremdschlüssel** (müssen nicht notwendigerweise selbst Schlüssel sein) **zeigen** via «Pfeile» **auf** die entsprechenden **Primärschlüssel**.
 - Die Schlüssel bei Beziehungstypen werden aus den Kardinalitäten abgeleitet (wobei bei Bedarf auch noch zusätzliche Schlüssel definiert werden dürfen).

Repetition konzeptioneller Entwurf

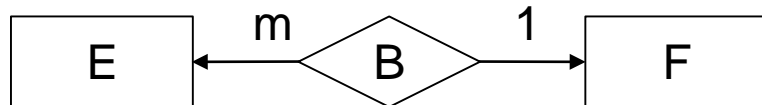
- Entwurfsmethode – Kardinalitäten, werden mit Hilfe von Schlüsseln durchgesetzt:



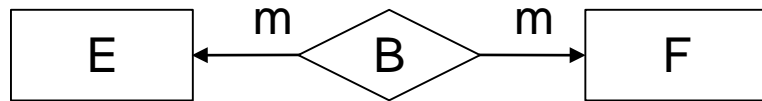
Schlüssel von E **und** F
(2 Schlüssel)



Schlüssel von F



Schlüssel von E



Schlüssel von E **komb.** F
(1 Schlüssel)

Korrekte ER-Diagramme

- Zwischenstand; wir kennen nun alle **Bausteine** zur Erstellung eines ER-Diagrammes:
 - Entitätstypen
 - Attribute
 - Beziehungstypen
 - Schlüssel
 - Kardinalitäten
 - ISA- und ID-Beziehungen
 - Zusammengesetzte Entitätstypen
- Wie setzt man sie **korrekt** zusammen?
- In der Praxis brauchen wir oft noch mehr (z.B. komplexere Integritätsbedingungen; was nicht gezeichnet werden kann wird textuell festgehalten).

Korrekte ER-Diagramme

- Man kann die Bausteine auf beliebig viele Arten «zusammensetzen». Nicht alles führt zu einem sinnvollen Ergebnis, man braucht also **Regeln**.
- Diese Regeln («Syntax») helfen beim:
 - **Erstellen** eines neuen Diagrammes
 - Beim **Überprüfen** eines bestehenden Diagrammes.
- Wenn die Regeln befolgt werden entstehen automatisch «korrekte» Diagramme (mit gewissen günstigen Eigenschaften).
- ACHTUNG: Korrektes Diagramm \neq richtiges Diagramm
- Man startet mit einem leeren Blatt...
(die Regeln können in beliebiger Reihenfolge angewandt werden)

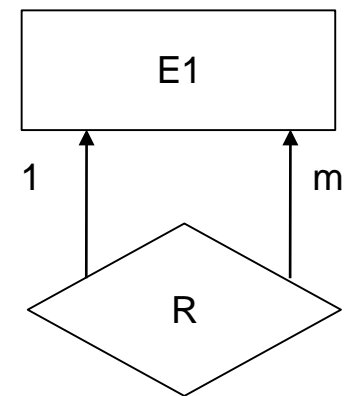
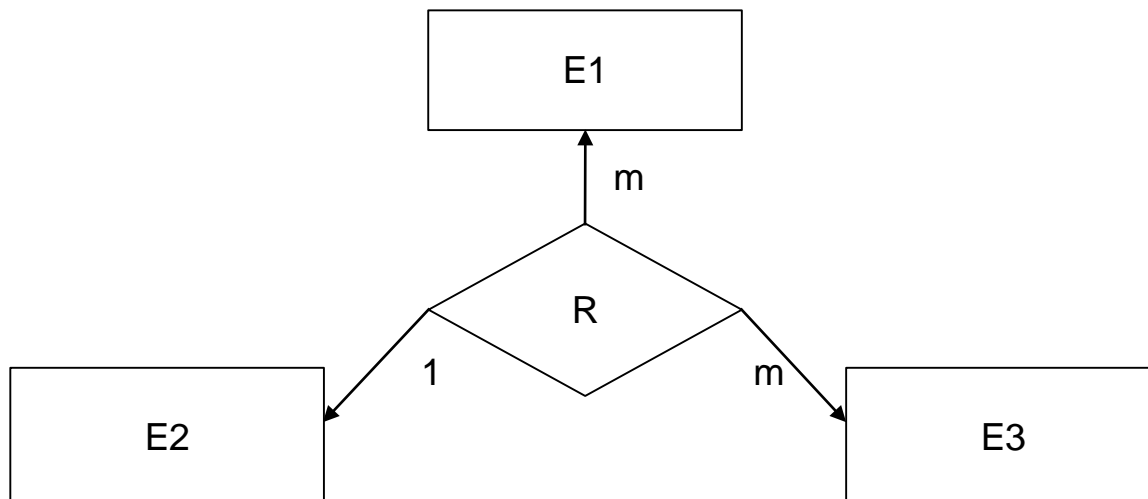
Korrekte ER-Diagramme: Regel 1

- **Definiere unabhängigen Entitätstyp**
- Voraussetzungen: Keine
- Ergebnis: Ein neues Rechteck (mit eindeutigem Namen)



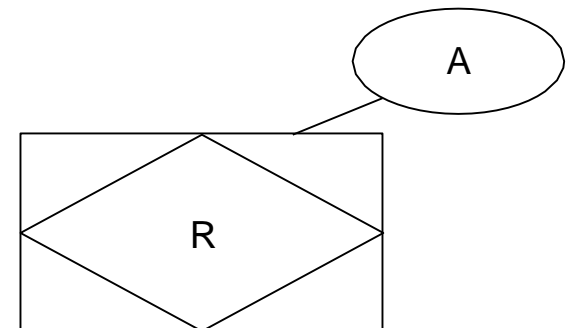
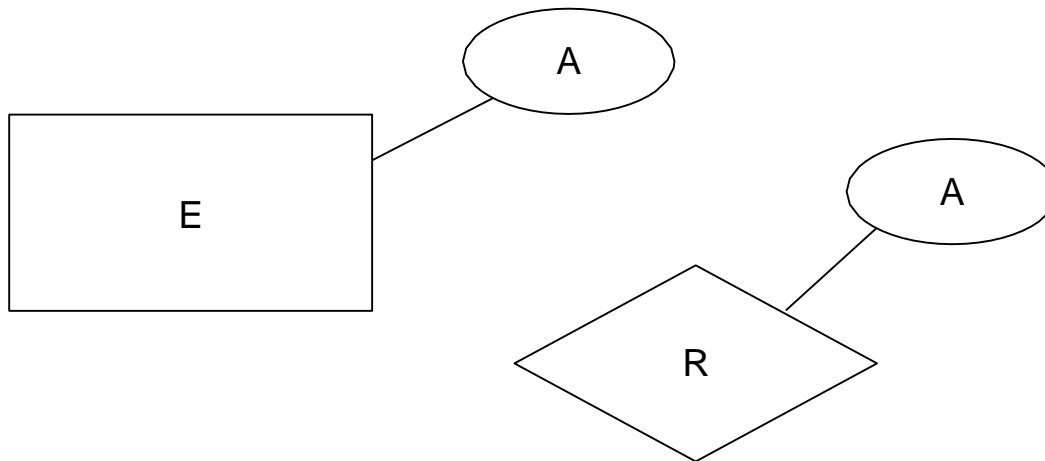
Korrekte ER-Diagramme: Regel 2

- **Definiere Beziehungstyp**
- Voraussetzungen: Mindestens zwei Entitätstypen (Rechtecke oder rechteckumschlossene Rhomben). Es kann auch zweimal derselbe sein.
- Ergebnis: Ein neuer Rhombus R mit '1' oder 'm' markierten Pfeilen zu den Entitätstypen



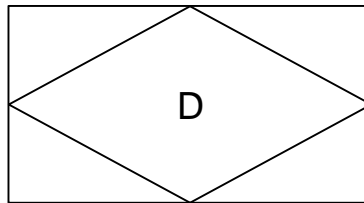
Korrekte ER-Diagramme: Regel 3

- **Definiere Attribut**
- Voraussetzung: Entitäts- oder Beziehungstyp (ein Rechteck, ein Rhombus oder ein rechteckumschlossener Rhombus)
- Ergebnis: Neues Oval strichverbunden mit E oder R und mit (innerhalb E bzw. R eindeutigem Namen)



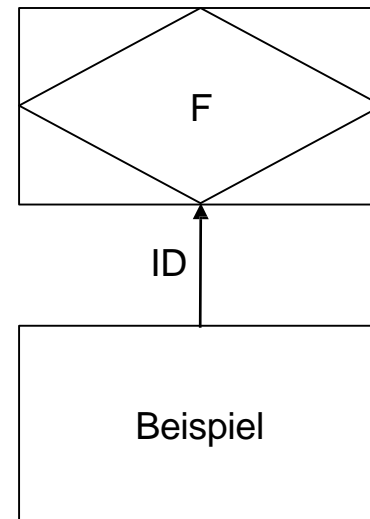
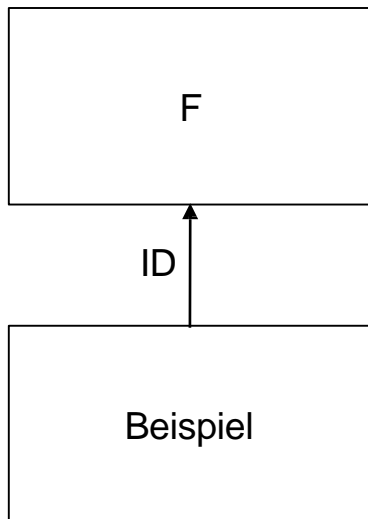
Korrekte ER-Diagramme: Regel 4

- Wandle Beziehungstyp in zusammengesetzten Entitätstyp um
- Voraussetzungen: Ein Rhombus D
- Ergebnis: Rhombus D durch ein Rechteck umschlossen



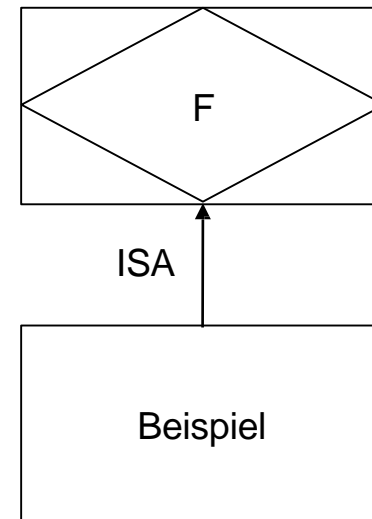
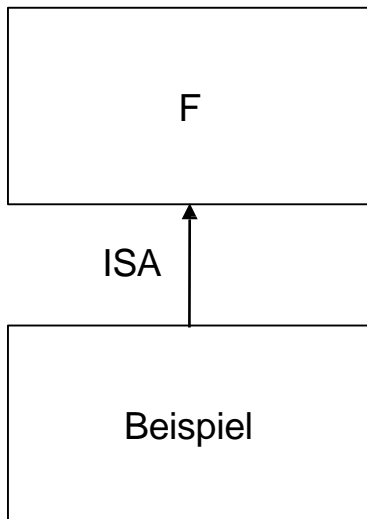
Korrekte ER-Diagramme: Regel 5

- **Definiere ID-abhängigen Entitätstyp**
- Voraussetzungen: Ein Rechteck oder rechteckumschlossener Rhombus F
- Ergebnis: Neues Rechteck mit 'ID' markiertem Pfeil zu F



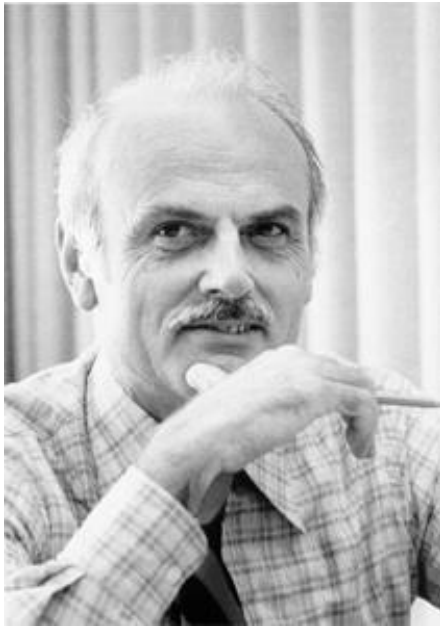
Korrekte ER-Diagramme: Regel 6

- **Definiere ISA-abhängigen Entitätstyp**
- Voraussetzungen: Ein Rechteck oder rechteckumschlossener Rhombus F
- Ergebnis: Neues Rechteck mit 'ISA' markiertem Pfeil zu F



Korrekte ER-Diagramme: Schlüssel

- Bevor ein Diagramm «weiterverarbeitet» werden kann, müssen die **Schlüssel** definiert sein. Ausgangslage: Korrektes Diagramm.
- Jeder unabhängige Entitätstyp erhält **einen oder mehrere** Schlüssel.
- Falls der Entitätstyp eingehende Pfeile hat, wählen wir einen **Primärschlüssel** (und unterstreichen diesen).
- Für Beziehungstypen: Wahl von **Fremdschlüssel(n)** und **Schlüssel(n)** (gemäss Kardinalitäten).
- Für Umwandlung in zusammengesetzte Entitätstypen: Primärschlüssel wählen (unterstreichen).
- Entitätstyp E ist ID- oder ISA-abhängig von F: **Primärschlüssel in F** wählen, **Fremdschlüssel und Schlüssel in E** wählen.
- → «**Angereichertes**» korrektes ER-Diagramm.



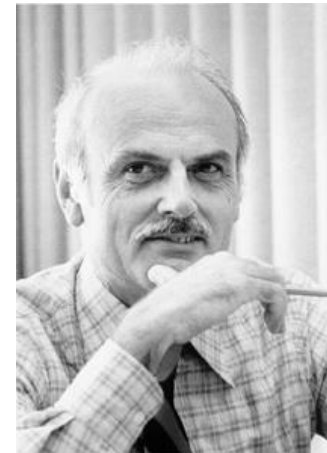
E.F. Codd

“There is nothing more practical than a good theory.”
James C. Maxwell

Grundbegriffe des relationalen Datenmodelles

Wer hats erfunden?

- E. F. Codd: A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 13(6): 377-387(1970)
- 1981 Turing-Award-Preisträger!
- In der Praxis (als formale Grundlage von RDBMS) von sehr grosser Bedeutung!



Gründe für den Erfolg des rel. Modelles

- Einfache Datenstruktur: **Relation** («mathematisches Objekt»).
- **Mengenorientierte** Verarbeitung der Daten, alle Operationen führen wieder zu Relationen. **Wenige Grundoperationen** zur Verarbeitung und dadurch eine klare Semantik (→ relationale Algebra).
- **Formale Theorie** zur Modellierung und Anfrageverarbeitung.
- Implementationen sind relativ einfach zu benutzen:
 - DDL (data definition language): → SQL
 - DML (data manipulation language): → SQL
 - DQL (data query language): → SQL
 - DCL (data control language): → SQL

Begriff: Wertebereich (Domäne)

- Menge einfacher bzw. «atomarer» Werte (entspricht im wesentlichen einem **Datentyp** einer höheren Programmiersprache).
- Beispiele:
 - Ganze Zahlen, Fixpunktzahlen (Dezimalbrüche), Gleitkommazahlen, ...
 - Menge aller Zeichenketten, evt. einer festen Länge
 - Aufzählungstypen, z.B. {red, green, blue}, {CHF, EUR, GBP, USD}
 - Unterbereichstypen, z.B. ganze Zahlen im Intervall [100 ... 999]
 - ...

Begriff: Attribut

- «Eigenschaften, die uns interessieren» (analog wie bei ERM)
- Ein Attribut besteht aus zwei Teilen:
 - **Bezeichnung/Name**
 - **Domäne/Wertebereich**, aus der die zugehörigen Werte stammen können.
- Ein Attribut nimmt **konkrete Werte** an (können sich im Laufe der Zeit ändern):
 - **Attributwerte** (Bsp. Umsatz / 200)
- Beispiel:
 - Anrede / {„Herr“, „Frau“}
 - Ort / string[30]
 - ...

Begriff: Tupel (n-Tupel)

- Sammlung von als zusammengehörig betrachteter Attribute:
 - Feste Zahl von Komponenten
 - Beliebige Anordnung (d.h. Reihenfolge ist erst wichtig, wenn einmal festgelegt)
 - Der Attributwert entstammt einer für jedes Attribut festgelegten Domäne.
- Wichtig: Die Attributwerte verändern sich über die Zeit, die Attribute selbst, also Name, Bedeutung und Wertebereich, bleiben konstant.
- Eine Menge von gleichartig strukturierten Tupeln bildet eine Relation:

Wert des Attributs A_i des Tupels t

R

A_1	...	A_i	...	A_n
a_{11}	A_{1n}
...
a_{k1}	...	$a_{ki} = t.A_i$...	a_{kn}
...

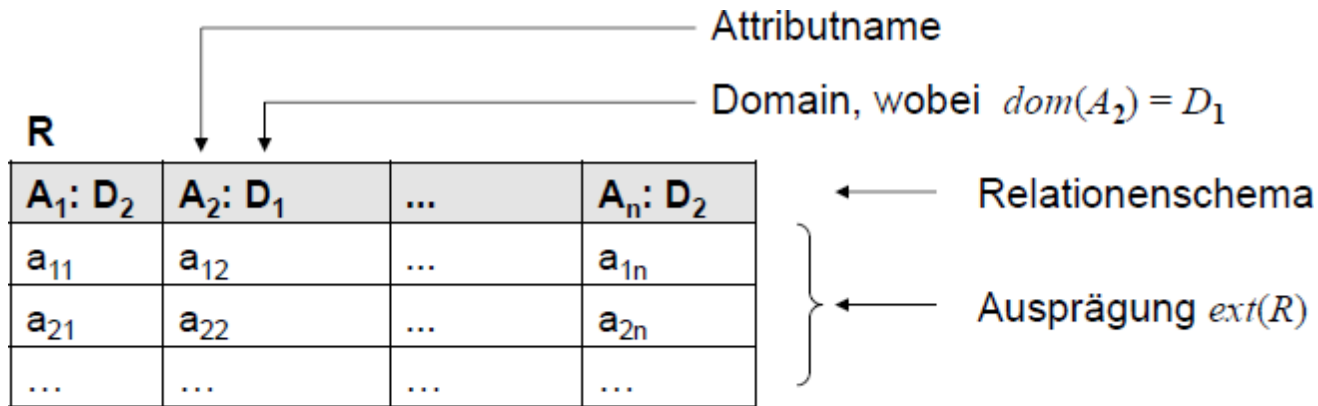
← Tupel t

Begriff: Relation

- Eine Relation R besteht aus zwei Elementen, einem:
 1. Einem sogenannten **Relationenschema** (auch **Relationsvariable** genannt): Menge von Namen von Attributen $\{A_1, \dots, A_n\}$, wobei jedem Attributnamen A_i ein Domain (Wertebereich) $\text{dom}(A_i) \in \{D_1, \dots, D_m\}$ zugeordnet wird

und
 2. Einer sogenannten **Ausprägung** (Menge von Tupeln, die dem Relationenschema entsprechen): $\text{ext}(R) \subseteq \text{dom}(A_1) \times \dots \times \text{dom}(A_n)$
- «Eine Relation ist eine Teilmenge des kartesischen Produktes von n endlichen Wertebereichen»
- Wichtig: Relationen sind **Mengen**, enthalten also (per Definition!) **keine doppelten Elemente und sind ungeordnet!**

Begriff: Relation (Notationen)



Gleiche Relation ohne Angabe der Domains:

R

A_1	A_2	...	A_n
a_{11}	a_{12}	...	a_{1n}
a_{21}	a_{22}	...	a_{2n}
...

Kurznotation für Relationenschema: $R(A_1, A_2, \dots, A_n)$

Beispiel: Relation

Employees

EmpNo: EmpNos	Name: PersonNames	AdrCity: CityNames	YrSalAmt: INTEGER	YrSalCur: Currencies	YrBonAmt: INTEGER	YrBonCur: Currencies
1	'Bechtolsheim'	'Palo Alto'	100000	'USD'	50000	'USD'
2	'Khosla'	'Altos Hills'	200000	'USD'	100000	'USD'
3	'McNealy'	'Atherton'	150000	'USD'	200000	'USD'
5678	'Saaner'	'Thalwil'	120000	'CHF'	100000	'CHF'

© R.Marti, 2004

- Domänen:
 - EmpNos = INTEGER
 - PersonNames = VARCHAR(30)
 - CityNames = VARCHAR(30)
 - Currencies = {'USD', 'GBP', 'EUR', 'CHF'}

Bemerkung zu Domänen

- Zweck von Domänen:
 - **Theorie:** Nur Attribute mit gleichen Domänen sind «kompatibel», z.B. bei Vergleichen, arithmetischen Operationen etc., Attribute mit verschiedenen Domänen sind nicht kompatibel.
 - **Praxis:** Nicht relevant, die meisten RDBMS unterstützen nur vordefinierte Domänen wie INTEGER, VARCHAR(n) etc. – und das ist auch gut so!
- Aber: Domänen dürfen nur sogenannte «atomare» Werte enthalten, d.h. Mengen von Werten sind nicht zulässig (sondern nur ein einzelner Wert).

UStudents

StudNo	Name	Faculty	ProgrammingSkills
'87-604-I'	'Meier'	'Comp Science'	{ 'Java', 'C', 'SQL' }
'91-872-I'	'Schmid'	'Comp Science'	{ 'Pascal', 'Java' }
'91-109-I'	'Anderegg'	'Comp Science'	{ 'Java' }
'94-555-L'	'Imboden'	'Chemistry'	{ }

Zwischenstand

- Mit den eingeführten Begriffen können wir nun **Datenstrukturen** (Relationen) **modellieren** und Daten (zu einem Zeitpunkt) darstellen:

Customers

CNo	Name	City	Balance	Discount
1	'Legrand'	'Genève'	0.00	0.10
2	'Studer'	'Zürich'	-800.00	0.20
...

Attributname

Relationenschema

Ausprägung

Tupel

Products

PNo	Descr	Weight	Price	Warehouse	Stock
1	'Paper'	2.000	20.00	'Zürich'	10000
2	'Disk Drive'	1.000	2500.00	'Bern'	400
...

Orders

OrdNo	CNo	PNo	Qty	Amount	Status	ValidDate
1	1	1	100	1800.00	'paid'	2010-07-16
2	1	1	100	1800.00	'paid'	2010-07-21
3	1	2	4	9000.00	'paid'	2010-09-30

Begriff: Schlüsselkandidat

- Gegeben: Eine Relation R mit dem Schema $R(A_1, \dots A_n)$
- Eine Teilmenge von Attributen $K \subseteq \{A_1, \dots A_n\}$ heisst **Schlüsselkandidat** wenn gilt:
 1. Für je zwei Tupel gilt zu **jedem Zeitpunkt**: Falls sie in den Attributwerten von K übereinstimmen, müssen sie gleich sein (d.h. es gibt nicht zwei Tupel mit denselben Schlüsselattribut**werten**).
 2. Man kann in K (= Menge der Schlüsselattribute) nichts weglassen, ohne diese Eigenschaft zu verlieren.
- Wenn mehrere Schlüsselkandidaten zur Verfügung stehen, muss eine **Auswahl** getroffen werden.
- Frage: Gibt es immer einen Schlüsselkandidaten?

Beispiel: Schlüsselkandidaten

Customers

CNo	Name	City	Balance	Discount
1	'Legrand'	'Genève'	0.00	0.10
2	'Studer'	'Zürich'	-800.00	0.20

Schlüsselkandidat:
{ CNo }

Products

PNo	Descr	Weight	Price	Warehouse	Stock
1	'Paper'	2.000	20.00	'Zürich'	10000
2	'Disk Drive'	1.000	2500.00	'Bern'	400

Schlüsselkandidaten:
{ PNo }
{ Descr }

Orders

OrdNo	CNo	PNo	Qty	Amount	Status	ValidDate
1	1	1	100	1800.00	'paid'	2010-07-16
2	1	1	100	1800.00	'paid'	2010-07-21
3	1	2	4	9000.00	'paid'	2010-09-30

Schlüsselkandidaten: { OrdNo }, evt. { CNo, PNo, ValidDate }

© R.Marti, 2004

Begriff: Primärschlüssel

- **Primärschlüssel** (primary key, PK): Ein ausgewählter Schlüsselkandidat, der explizit als Primärschlüssel bezeichnet wird.
- Kriterien für Auswahl des Primärschlüssels (allg. von Schlüsseln):
 - Attributwert(e) sollten sich möglichst wenig ändern (idealerweise nie).
 - Eindeutigkeit der Werte eines Primärschlüssels sollte über die Zeit gelten, d.h. ein einmal verwendeter Wert sollte später nicht wiederverwendet werden.
 - Attribut(e) sollten möglichst wenig Speicherplatz benötigen («kurze Schlüssel»)

→ Entwurfsentscheid
- Wenn nichts «passt»: Surrogatschlüssel («künstlicher Schlüssel») definieren.

Begriff: Fremdschlüssel

- **Fremdschlüssel** (foreign key, FK): Eine Menge von Attributen in einer Relation S zu der es eine Relation R gibt, deren Primärschlüssel von diesen Attributen in S referenziert werden. Ein Fremdschlüssel (in S) kann, muss aber nicht Schlüssel in S sein.
- Bemerkungen:
 - Die Attributnamen des Primärschlüssels von R und des Fremdschlüssels von S müssen nicht gleich sein.
 - Ein Attribut in S ist nicht notwendigerweise ein Fremdschlüssel nur weil ein Attribut gleichen Namens in einer anderen Relation R als Primärschlüssel auftritt. Primärschlüssel/Fremdschlüssel-Beziehungen müssen explizit deklariert werden.
 - Es ist empfehlenswert, für sich entsprechende Fremdschlüssel- und Primärschlüsselattribute die gleichen Namen zu verwenden (geht nicht immer).

Beispiel: Primär-/Fremdschlüssel

Primärschlüssel in **Customers**: { **CNo** }

Customers

CNo	Name	City	Balance	Discount
1	'Legrand'	'Genève'	0.00	0.10
2	'Studer'	'Zürich'	-800.00	0.20

Fremdschlüssel { CNo } in Orders
referenziert den **Primärschlüssel { CNo } in Customers**

Orders

OrdNo	CNo	PNo	Qty	Amount	Status	ValidDate
1	1	1	100	1800.00	'paid'	2010-07-16
2	1	1	100	1800.00	'paid'	2010-07-21
3	2	2	4	9000.00	'paid'	2010-09-30

Nebenbemerkung: **Primärschlüssel** in **Orders**: { **OrdNo** }

© R.Marti, 2004

Begriff: NULL, Integrität

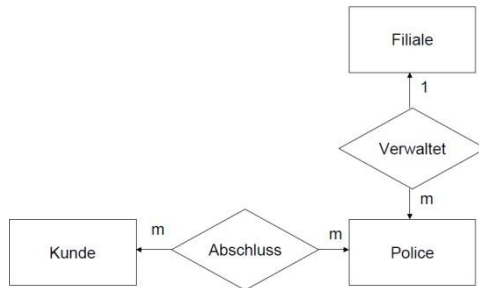
- NULL: Wenn der Wert eines Attributs A eines Tupels t **undefiniert** oder **unbekannt** ist, dann wird ein sogenannter Platzhalter (**NULL**) verwendet.
- Bemerkung: NULL's führen zu verschiedenen Problemen (z.B. bei Abfragen) und sollten deshalb **wenn immer möglich** vermieden werden (\rightarrow DB-4).
- Primärschlüsselbedingung (entity integrity): Die Attribute des Primärschlüssels K einer Relation R dürfen nie NULL sein.
- Fremdschlüsselbedingung (referential integrity): Für jeden Wert eines Fremdschlüssels F in einer Relation S muss in der referenzierten Relation R jeweils entweder ein Tupel mit demselben Wert als Primärschlüssel existieren **oder die Fremdschlüsselattributwerte müssen NULL sein.**

Zusammenfassung

- Im relationalen Modell werden Daten «tabellarisch» dargestellt:
- Eine Relation (Tabelle) besteht aus:
 - Einem **Schema** (Tabellenkopf, *unveränderlich*)
 - **Tupeln** («Zeilen», Inhalte *veränderlich*)
 - Tupel setzen sich zusammen aus n **Attributwerten**
 - Ein **Attribut** («Spalte») hat eine Bezeichnung und eine Domäne («Datentyp»)
 - Wichtig: Im relationalen Modell gibt es NIE zwei gleiche «Zeilen» (d.h. solche, die in allen Attributwerten übereinstimmen).

Kaffeepause



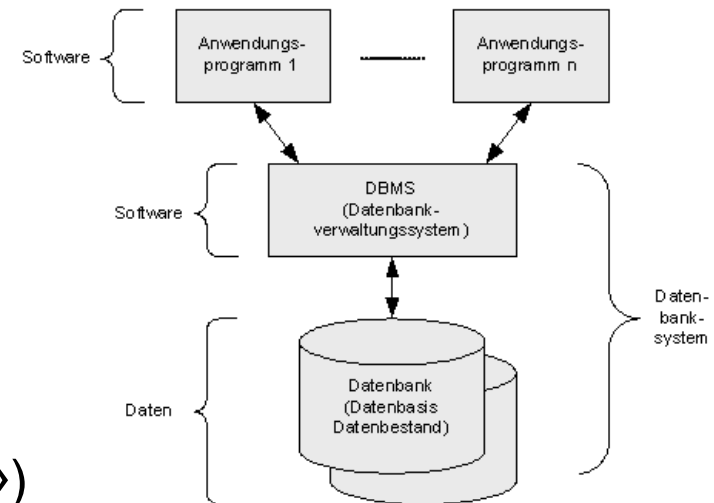


A_1	A_2	...	A_n
a_{11}	a_{12}	...	a_{1n}
a_{21}	a_{22}	...	a_{2n}
...

Konzeptioneller Entwurf → logischer Entwurf

Wie geht es weiter?

- Was können wir bisher?
 - Eine Problemstellung als **ER-Diagramm** beschreiben.
- Was wollen wir?
 - Eine **Datenbank**!
- ER-Diagramme können aber nicht direkt durch ein RDBMS ausgeführt werden.
- Das Diagramm muss abgebildet («übersetzt») werden in etwas, was ein RDBMS versteht.
- Das RDBMS «spricht» SQL (basiert auf dem Relationenmodell).



ER-Diagramm → relationales Modell

- Jedes **Kästchen** (Rechteck, Rhombus, rechteckumschlossener Rhombus) geht in eine **Relation** über, mit entsprechenden Attributen. **Wichtig**: Die **Domänen** sind **spätestens jetzt geeignet** zu wählen!
- Schlüssel, Fremdschlüssel, Primärschlüssel werden als solche übernommen.
- In der Praxis: Kein Umweg via relationales Modell sondern Abbildung direkt nach SQL (→ DB-3).
- Resultat: Implementierbare Datenbank! Aber Achtung: Da fehlt dann noch so manches....

Einschub: Warum ER-Diagramm?

- Es stellt sich die Frage, ob man bei einfachen Problemstellungen den Umweg via **ER-Diagramm** überhaupt gehen muss. Man könnte doch **direkt Relationen** definieren. Ja, könnte man, aber ...
 - Beispiel: Wir wollen folgenden Sachverhalt festhalten:
 - Studierende haben eine Studierendenummer, einen Namen und eine Adresse.
 - Sie studieren an genau einem Departement, das eine Nummer hat und einen Namen.
 - Sie belegen Kurse (die haben eine Bezeichnung) legen dort eine Prüfung ab, die mit einer Note bewertet wird.
- Relationsvariable: Student(SNo, SName, Adresse, DNo, DName, Kurs, Note)

Einschub: Warum ER-Diagramm?

- Das führt – beispielsweise – zu folgender Relation:

<u>SNo</u>	SName	Address	DNo	DName	<u>Course</u>	Grade
87-604-I	Meier	Basel	IIIC	Informatik	Informatik	6
87-604-I	Meier	Basel	IIIC	Informatik	Analysis	5
87-604-I	Meier	Basel	IIIC	Informatik	Physik	4
91-872-I	Schmid	Bern	IIIC	Informatik	Informatik	5
91-872-I	Schmid	Bern	IIIC	Informatik	Analysis	3
91-109-I	Anderegg	Zürich	IIIC	Informatik	Informatik	4
94-555-P	Imboden	Luzern	IX	Mathematik	Algebra	3

- Was ist daran schlecht?

Einschub: Warum ER-Diagramm?

- Wir haben sogenannte Anomalien:
 - Update-Anomalie: „Meier“ zieht nach Zürich um
→ Änderungen in 3 Tupeln nötig
 - Delete-Anomalie: „Imboden“ verlässt die Schule
→ Fakt, dass Dept. IX = Mathematik, geht verloren
 - Insert-Anomalie: „Kunz“ hat noch keine Prüfung abgelegt
→ Fakt, dass er in Aarau wohnt kann nicht eingefügt werden

<u>SNo</u>	SName	Address	DNo	DName	<u>Course</u>	Grade
87-604-I	Meier	Basel	IIIC	Informatik	Informatik	6
87-604-I	Meier	Basel	IIIC	Informatik	Analysis	5
87-604-I	Meier	Basel	IIIC	Informatik	Physik	4
91-872-I	Schmid	Bern	IIIC	Informatik	Informatik	5
91-872-I	Schmid	Bern	IIIC	Informatik	Analysis	3
91-109-I	Anderegg	Zürich	IIIC	Informatik	Informatik	4
94-555-P	Imboden	Luzern	IX	Mathematik	Algebra	3

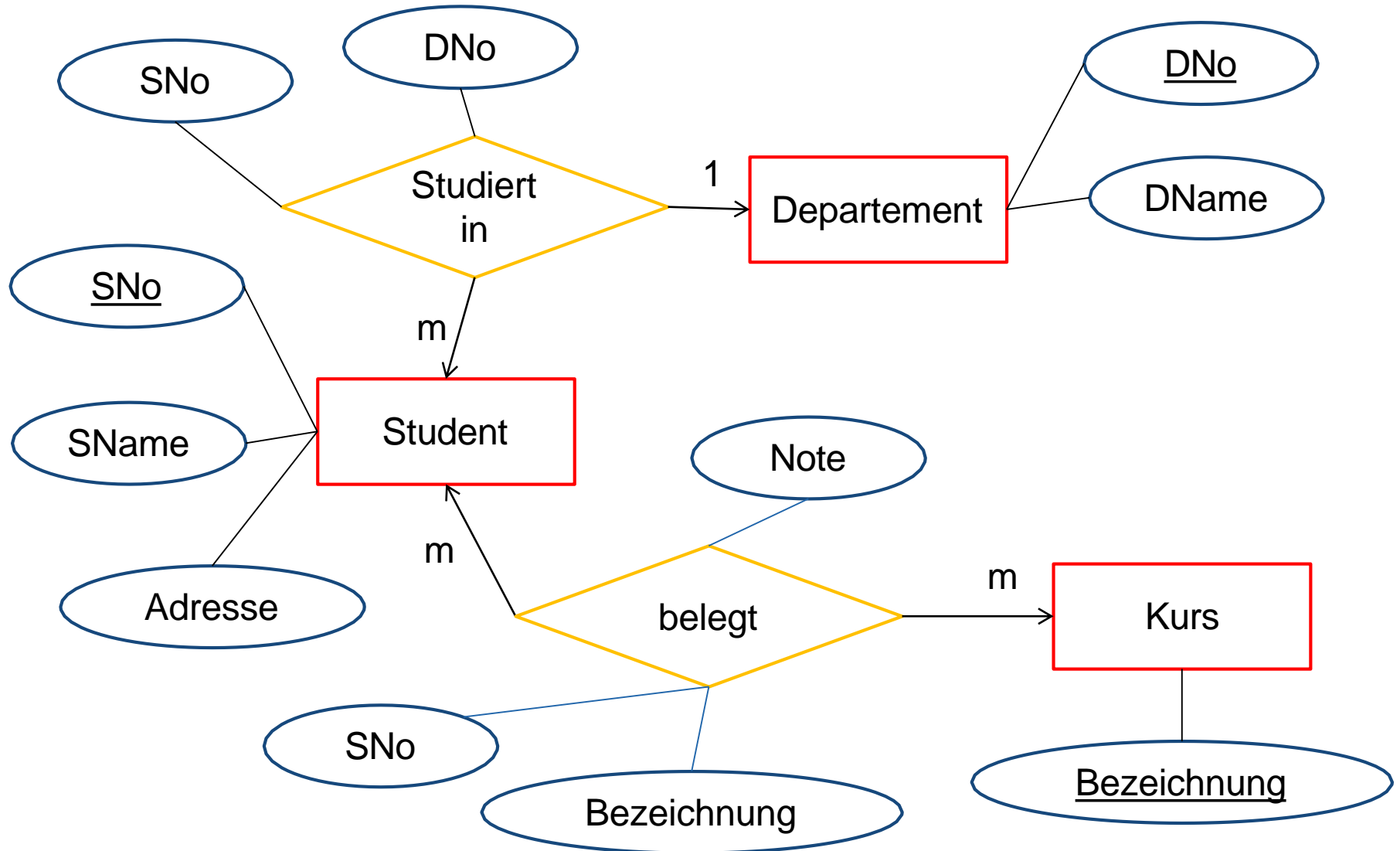
Einschub: Warum ER-Diagramm?

- Es gibt eine präzise (aber nicht ganz triviale) mathematische Vorgehensweise um diese Probleme zu «flicken»: **Normalisierung**.
- Aber: Was wäre passiert beim top-down-Entwurf?
 - Studierende haben eine Studierendenummer, einen Namen und eine Adresse.
 - Sie studieren an genau einem Departement, das eine Nummer hat und einen Namen.
 - Sie belegen Kurse (die haben eine Bezeichnung) legen dort eine Prüfung ab, die mit einer Note bewertet wird.
- Zeichnen Sie das resultierende ER-Diagramm.

Einschub: Warum ER-Diagramm?

- Studierende haben eine Studierendennummer, einen Namen und eine Adresse.
- Sie studieren an genau einem Departement, das eine Nummer hat und einen Namen.
- Sie belegen Kurse (die haben eine Bezeichnung) legen dort eine Prüfung ab, die mit einer Note bewertet wird.
- Entitätstypen, Attribute, Beziehungstypen, Primärschlüssel

Lösung: Keine Anomalien!



\div π \cup

σ \times ρ

\cap \bowtie \setminus

Relationale Algebra

Begriff: Relationale Algebra

- Operatoren und Regeln für das «Rechnen» mit Relationen (Analogie: Arithmetik mit Addition, Subtraktion... auf ganzen Zahlen).
- Bis jetzt: Relationenschemas mit Relationen (Attributwerten), die in einer Datenbank gespeichert sind (wie sie dahin kommen folgt später).
- Es fehlt: «Abgeleitete» Relationenschemas mit Relationen, die aus den «Basisrelationen» berechnet werden.
- Die Rechenvorschriften definieren eine «Abfragesprache», d.h. eine Sprache, mit der beliebige Abfragen an eine Datenbank gestellt werden können.

Begriff: Relationale Algebra

- Mathematik:
 - Algebra: Definiert durch Wertebereich und auf diesem definierte Operatoren.
 - Operand: Variablen oder Werte aus denen neue Werte konstruiert werden können.
 - Operator: Symbole, die Prozeduren repräsentieren, die aus gegebenen Werten neue Werte produzieren.
- Für Datenbankankfragen:
 - Inhalt der Datenbank (Relationen) sind **Operanden**.
 - **Operatoren** definieren **Funktionen zum Berechnen von Anfrageergebnissen**:
«Grundlegenden Dinge, die wir mit Relationen tun wollen».
 - Relationale Algebra (Relationenalgebra, RA).
 - Anfragesprache für das relationale Modell.

Kriterien für Abfragesprachen

- Ad-Hoc-Formulierung:
 - Benutzer sollen eine Anfrage formulieren können, ohne ein vollständiges Programm schreiben zu müssen.
- Deskriptivität / Deklarativität:
 - Benutzer sollen formulieren „Was will ich haben?“ und nicht „Wo finde ich das, was ich haben will?“
- Optimierbarkeit:
 - Sprache besteht aus wenigen Operationen.
 - Optimierungsregeln für die Operatorenmenge.

Kriterien für Abfragesprachen

- Abgeschlossenheit:
 - Anfragen erfolgen auf Relationen.
 - Anfrageergebnis ist wiederum **eine Relation** und kann als Eingabe für die nächste Anfrage verwendet werden → Schachtelung möglich.
- Mengenorientiertheit:
 - Operationen auf Mengen von Daten.
 - Nicht navigierend nur auf einzelnen Elementen („tuple-at-a-time“).
- Sicherheit:
 - Keine Anfrage, die syntaktisch korrekt ist, darf in eine Endlosschleife geraten oder ein unendliches Ergebnis liefern.
- ...

Klassifikation der rel. Operatoren

- Entfernende Operatoren:
 - Selektion, Projektion
- Mengenoperatoren:
 - Vereinigung, Schnittmenge, Differenz
- Kombinierende Operatoren:
 - Kartesisches Produkt, Join, Joinvarianten
- Umbenennung:
 - Verändert nicht Tupel, sondern Schema
- Ausdrücke der relationalen Algebra: „Anfragen“ (queries)

Entfernend: Selektion, σ

- Unärer Operator (d.h. nur ein Operand).
- Erzeugt neue Relation mit **gleichem Schema** aber einer Teilmenge der Tupel.
- Nur Tupel, die der sogenannten **Selektionsbedingung** entsprechen, werden übernommen.
- Selektionsbedingung: Kombination von logischen Ausdrücken bestehend aus Attributen und/oder Konstanten (das Ergebnis des Ausdrucks muss wahr oder falsch sein).
- Prüft Selektionsbedingung **für jedes Tupel** der Relation.

Entfernend: Selektion, σ

Filme					
Titel	Jahr	Länge	inFarbe	Studio	ProduzentID
Total Recall	1990	113	True	Fox	12345
Basic Instinct	1992	127	True	Disney	67890
Dead Man	1995	90	False	Paramount	99999

$\sigma_{\text{Länge} \geq 100}(\text{Filme})$

Titel	Jahr	Länge	inFarbe	Studio	ProduzentID
Total Recall	1990	113	True	Fox	12345
Basic Instinct	1992	127	True	Disney	67890

© F.Naumann, 2011

Entfernend: Selektion, σ

Customers

CNo	Name	City	Balance	Discount
1	'Legrand'	'Genève'	0.00	0.10
2	'Studer'	'Zürich'	-800.00	0.20
3	'Huber'	'Zürich'	-20.00	0.05

Finde alle Zürcher Kunden, die einen Rabatt von 15% oder mehr erhalten

$\sigma_{\text{City} = \text{'Zürich'} \wedge \text{Discount} \geq 0.15}$ Customers

CNo	Name	City	Balance	Discount
2	'Studer'	'Zürich'	-800.00	0.20

Entfernend: Projektion, π

- Unärer Operator (d.h. nur ein Operand).
- Erzeugt neue Relation mit einer **Teilmenge der ursprünglichen Attribute**
 - $\pi_{A1,A2,\dots,Ak}(R)$ ist eine Relation mit den Attributen $A1,A2,\dots,Ak$
- Achtung: Es können **Duplikate** entstehen, die **entfernt** werden müssen.
- Die Projektion wählt eine Menge von Spalten aus (und entfernt die Übrigen).
- Mit einer Projektion lässt sich auch die **Reihenfolge** der Spalten anpassen.

Entfernend: Projektion, π

Filme					
Titel	Jahr	Länge	inFarbe	Studio	ProduzentID
Total Recall	1990	113	True	Fox	12345
Basic Instinct	1992	127	True	Disney	67890
Dead Man	1995	121	False	Paramount	99999

$\pi_{\text{Titel,Jahr,Länge}}(\text{Filme})$

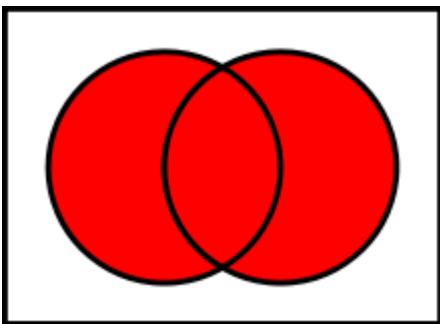
Titel	Jahr	Länge
Total Recall	1990	113
Basic Instinct	1992	127
Dead Man	1995	121

$\pi_{\text{inFarbe}}(\text{Filme})$

inFarbe
True
False

Mengenoperator: Vereinigung, \cup

- Sammelt Elemente (Tupel) zweier Relationen unter einem gemeinsamen Schema auf.
- Attributmengen (Schemas) beider Relationen müssen identisch sein.
 - Namen, Typen
 - Zur Not: Umbenennung
- Ein Element ist nur einmal in $(R \cup S)$ vertreten, auch wenn es jeweils einmal in R und S auftaucht: Duplikatentfernung.



Mengenoperator: Vereinigung, \cup

R			
Name	Adresse	Geschlecht	Geburt
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Mark Hamill	456 Oak Rd., Brentwood	M	8/8/88

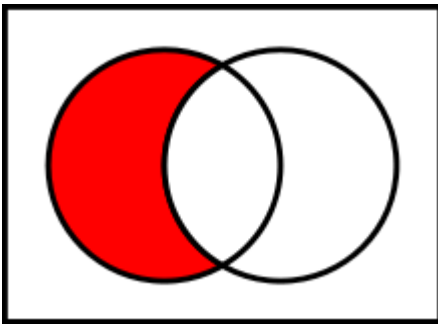
S			
Name	Adresse	Geschlecht	Geburt
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Harrison Ford	789 Palm Dr., Beverly Hills	M	7/7/77

$R \cup S$

Name	Adresse	Geschlecht	Geburt
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Mark Hamill	456 Oak Rd., Brentwood	M	8/8/88
Harrison Ford	789 Palm Dr., Beverly Hills	M	7/7/77

Mengenoperator: Differenz, \setminus oder $-$

- Differenz $R \setminus S$ (oder $R - S$) eliminiert die Tupel aus der ersten Relation, die auch in der zweiten Relation vorkommen.
- Attributmengen (Schemas) beider Relationen müssen identisch sein.
 - Namen, Typen
 - Zur Not: Umbenennung
- Achtung: $R \setminus S \neq S \setminus R$



Mengenoperator: Differenz, \ oder -

R			
Name	Adresse	Geschlecht	Geburt
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Mark Hamill	456 Oak Rd., Brentwood	M	8/8/88

S			
Name	Adresse	Geschlecht	Geburt
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Harrison Ford	789 Palm Dr., Beverly Hills	M	7/7/77

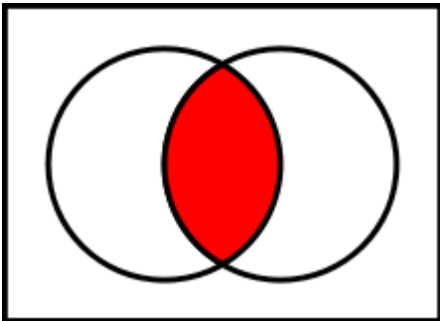
R - S

Name	Adresse	Geschlecht	Geburt
Mark Hamill	456 Oak Rd., Brentwood	M	8/8/88

© F.Naumann, 2011

Mengenoperator: Durchschnitt, \cap

- Durchschnitt $R \cap S$ ergibt die Tupel, die in beiden Relationen gemeinsam vorkommen.
- Attributmengen (Schemas) beider Relationen müssen identisch sein.
 - Namen, Typen
 - Zur Not: Umbenennung
- Ein Element ist nur einmal in $(R \cap S)$ vertreten, auch wenn es jeweils einmal in R und S auftaucht: Duplikatentfernung.



Mengenoperator: Durchschnitt, \cap

R			
Name	Adresse	Geschlecht	Geburt
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Mark Hamill	456 Oak Rd., Brentwood	M	8/8/88

S			
Name	Adresse	Geschlecht	Geburt
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Harrison Ford	789 Palm Dr., Beverly Hills	M	7/7/77

$R \cap S$

Name	Adresse	Geschlecht	Geburt
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99

© F.Naumann, 2011

Kombinierend: (kartesisches) Produkt, \times

- Kreuzprodukt zweier Relationen R und S ist die Menge aller Tupel, die man erhält, wenn man jedes Tupel aus R mit jedem Tupel aus S «kombiniert».
- Schema hat ein Attribut für jedes Attribut aus R und S
- Achtung: Bei Namensgleichheit wird kein Attribut ausgelassen, stattdessen: Umbenennen oder qualifizieren.

Kombinierend: (kartesisches) Produkt, \times

R

A	B
1	2
3	4

S

B	C	D
2	5	6
4	7	8
9	10	11

R \times S

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

Kombinierend: Verbund, Join, ⋈

- Motivation: Statt im Kreuzprodukt alle Paare zu bilden, sollen nur die Tupelpaare gebildet werden, deren Tupel irgendwie **übereinstimmen**.
- Idee, wie Tupel kombiniert werden:
 - «natural» join: **Übereinstimmung der Attributwerte in allen gemeinsamen Attributen**.
 - Gegebenenfalls Umbenennung
 - Schema: Vereinigung der beiden Attributmengen:



Kombinierend: Verbund, join, ⋈

R	A	B
	1	2
	3	4

S	B	C	D
	2	5	6
	4	7	8
	9	10	11

R ⋈ S		A	B	C	D
		1	2	5	6
		3	4	7	8

- Mehr als ein gemeinsames Attribut: Partner müssen **in allen** Attributwerten übereinstimmen!

R	A	B	C
	1	2	3
	6	7	8
	9	7	8

S	B	C	D
	2	5	6
	2	3	5
	7	8	10

R ⋈ S	A	B	C	D
	1	2	3	5
	6	7	8	10
	9	7	8	10

Kombinierend: Theta-join, \bowtie_P

- Verallgemeinerung des natürlichen Joins.
- Verknüpfungsbedingung P kann frei gestaltet werden (logischer Ausdruck).
- Konstruktion des Ergebnisses:
 - Bilde Kreuzprodukt
 - Selektiere mittels der Joinbedingung
 - Also: $R \bowtie_P S = \sigma_P(R \times S)$
 - $\theta \in \{=, <, >, \leq, \geq, \neq\}$
- Schema: Wie beim Kreuzprodukt.
- Natural Join ist ein Spezialfall des Theta-Joins.

Kombinierend: theta-join, \bowtie_{θ}

R

A	B	C
1	2	3
6	7	8
9	7	8

S

B	C	D
2	5	6
2	3	5
7	8	10

R $\bowtie_{A < D}$ S

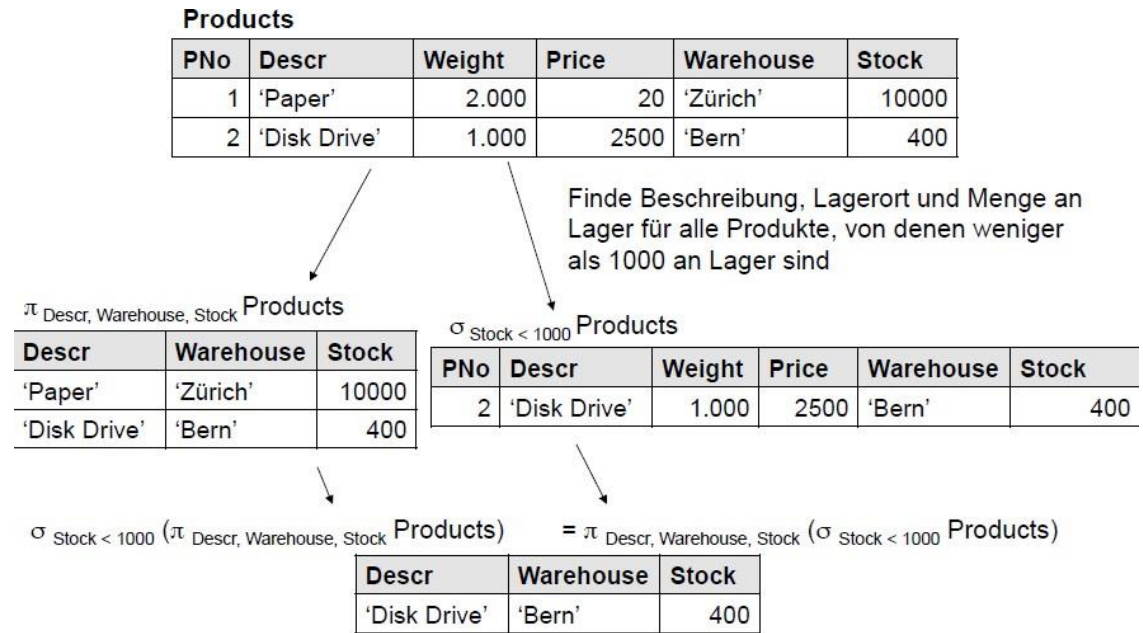
A	R.B	R.C	S.B	S.C	D
1	2	3	2	5	6
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

R $\bowtie_{A < D \text{ AND } R.B \neq S.B}$ S

A	R.B	R.C	S.B	S.C	D
1	2	3	7	8	10

Zwischenstand

- Was können wir bisher:
 - Komplexe Anfragen formulieren durch **Kombination** (Schachtelung) von **Ausdrücken**. Das Resultat eines (Teil)ausdruckes ist immer wieder eine Relation.
 - Darstellung: Als geschachtelter Ausdruck mittels Klammerung.
 - Einfaches Beispiel:



Zwischenstand

- Was fehlt noch?
 - «Zusammenfassen» von Attributwerten (aggregieren)
 - Beispiel: Gesamtumsatz, grösste PLZ, ...
 - «Zusammenfassen» von Tupeln (gruppieren)
 - Beispiel: Anzahl Verkäufe **pro** Postleitzahlengebiet, Umsatz **pro** Kunde, ...
 - Umbenennung, Berechnungen und Funktionsaufrufe in der Projektion
 - Sortierung
 - ...
- Machen wir dann direkt in SQL...

Outer Joins...

- Ausgangslage:

Customers

CNo	Name	City	Balance	Discount
1	'Legrand'	'Genève'	0.00	0.10
2	'Studer'	'Zürich'	-800.00	0.20

Orders

OrdNo	CNo	PNo	Qty	Amount	Status	ValidDate
1	1	1	100	1800.00	'paid'	2010-07-16
2	1	1	100	1800.00	'paid'	2010-07-21
3	1	2	4	9000.00	'paid'	2010-09-30

- Wir wollen wissen welche Kunden welche Produkte bestellt haben: \bowtie , \bowtie_p

Outer Joins...

- Wir wollen eine Liste aller Kunden mit bestellten Produkten (auch wenn sie nichts gekauft haben): ⋈, ⋈, ⋈ (outer joins)

Customers

CNo	Name	City	Balance	Discount
1	'Legrand'	'Genève'	0.00	0.10
2	'Studer'	'Zürich'	-800.00	0.20

Orders

OrdNo	CNo	PNo	Qty	Amount	Status	ValidDate
1	1	1	100	1800.00	'paid'	2010-07-16
2	1	1	100	1800.00	'paid'	2010-07-21
3	1	2	4	9000.00	'paid'	2010-09-30

Outer Joins – Varianten

- Outer Joins:
 - Left outer join: $L \bowtie R$
 - Right outer join: $L \ltimes R$
 - Full outer join: $L \bowtie\ltimes R$
- Left outer join: Alle Tupel von L mit NULL für Attribute von R wenn nicht verknüpfbar.
- Right outer join: Alle Tupel von R mit NULL für Attribute von L wenn nicht verknüpfbar.
- Full outer join: Alle Tupel von L und R mit NULL für Attribute von L bzw. R wenn nicht verknüpfbar.

Verbundvarianten

- Natürlicher join:

L					
A	B	C			
a ₁	b ₁	c ₁			
a ₂	b ₂	c ₂			

 \bowtie

R					
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

- Left outer join:

L					
A	B	C			
a ₁	b ₁	c ₁			
a ₂	b ₂	c ₂			

 \bowtie

R					
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
a ₂	b ₂	c ₂	NULL	NULL

Verbundvarianten

- Right outer join:

L			R			Resultat				
A	B	C	C	D	E	A	B	C	D	E
a ₁	b ₁	c ₁	c ₁	d ₁	e ₁	a ₁	b ₁	c ₁	d ₁	e ₁
a ₂	b ₂	c ₂	c ₃	d ₂	e ₂	NULL	NULL	c ₃	d ₂	e ₂

- Full outer join:

L			R			Resultat				
A	B	C	C	D	E	A	B	C	D	E
a ₁	b ₁	c ₁	c ₁	d ₁	e ₁	a ₁	b ₁	c ₁	d ₁	e ₁
a ₂	b ₂	c ₂	c ₃	d ₂	e ₂	a ₂	b ₂	c ₂	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	c ₃	d ₂	e ₂



Selbständige Übungen

Gruppenarbeit (2er Gruppen)

- Lösen der Aufgaben zum Thema Relationale Algebra.
- Wenn die Zeit nicht reicht, bitte bis nächster Woche fertig lösen.
- Musterlösung ab nächster Woche auf MOODLE.