

DB I

Vorlesung 16. März 2004

Dipl.-Inf. T. Mättig

tmaettig@hs-zigr.de

Tel.: 03581 4828-269

Raum GR I 257

Änderung Übungs-Termin

- Hilfsassistent: Marcel Hermkes
- Alle zusammen zu einem Termin
- ~~Dienstag 14⁰⁰ Uhr~~
- Möglich sind:
 - Montag ab 18⁰⁰ Uhr
 - Dienstag ab 18⁰⁰ Uhr
 - Donnerstag 16³⁰ Uhr
 - Freitag 8⁰⁰-9³⁰ Uhr

Errata

- Diese Folien sind im Rahmen des ersten DB I Semesters von Klaus ten Hagen entstanden.
- „Bug reports“ an Klaus@ten-Hagen.org sind willkommen.
 - Folien nach der Vorlesung unter <http://maettig.com/db1>

Literatur

- „Relationale Datenbanken“ von Hermann Sauer, Addison-Wesley
- „Datenbanken & Java“ von Gunter Saake und Kai-Uwe Sattler
- „Java Servlet Programming“ von Jason Hunter und William Crawford
- „Theorie und Praxis relationaler Datenbanken“ von R. Steiner
- WWW

Vorlesungsinhalte DB I

- Einführung RDBMS
- Relationale Grundlagen
 - Begriffe
 - Relationale Operationen
- SQL
 - CREATE
 - INSERT
 - SELECT
 - UPDATE
 - DELETE
- DBA (Administration)
- Datenbank-Entwurf
 - Normalisierung
 - Normalformen
 - Entity-Relationship (ER) Modell
 - UML
- SQL for Experts
 - SQL Übung mit komplexeren Beispielen
- Performanz
- Projekt
 - Ziel: Was zum Vorführen
 - Idee: Geldautomat

Übungsinhalte DB I: „SQL for Beginners“

- JRE
 - Für RDBMS, IDE, SQL clients, HTTP Server, Servlet container
 - <http://java.sun.com/j2se/1.3>
- RDBMS: McKoi
 - Embedded RDBMS, das nur JRE 1.4 benötigt.
 - <http://mckoi.com/database>
- SQL clients
 - McKoi kommt mit einem rudimentären SQL Client „JDBC Query Tool“.
 - <http://www.mckoi.com/database/UseEmbeddedApp.html#2>
 - Squirrel-SQL
 - <http://squirrel-sql.sourceforge.net>

Übungsinhalte DB I: SQL as part of an IDE

- Alternative zu Standard-„SQL Clients“:
- IDE: Eclipse
 - www.Eclipse.org
 - Eine deutsche Seite: www.EclipseProject.de
 - JFaceDBc als „Plug-In“ der IDE „Eclipse“
- Wenn Sie wollen können Sie auch Postgres, MS Access, MySQL, SQLite (?) verwenden → müssen nur die selben SQL-Übungen vorführen

Inhalte / „Scope“

- Vorlesung:
 - Standardprogram
- Übung:
 - SQL mit „Clients“ ist ein MUSS

Vorlesungsplan

- **2004-03-15 – heute**
- **2004-03-22 – Vorlesung #2**
- **2004-03-29 – Vorlesung #3**
- **2004-04-05 – Vorlesung #4**
- 2004-04-12 – Ostern
- **2004-04-19 – Vorlesung #5**
- **2004-04-26 – Vorlesung #6**
- **2004-05-03 – Vorlesung #7**
- **2004-05-10 – Vorlesung #8**
- **2004-05-17 – Vorlesung #9**
- **2004-05-24 – Vorlesung #10**
- 2004-05-31 – Pfingsten
- **2004-06-07 – Vorlesung #11**
- **2004-06-14 – Vorlesung #12**
- **2004-06-21 – Übungsklausur**
- **2004-06-28 – Konsultation**
- **2004-07-05 – Zweite Prüfungswoche**
- **2004-07-12 – Dritte Prüfungswoche**

Edgar F. „Ted“ Codd

- Codd formed the concepts for organizing and accessing data that are embodied in the relational database, a widely-used approach to organizing business data. Critical of IBM's then current data management systems, Codd, as a young IBM programmer working in Santa Teresa, California, proposed that data be organized according to principles based on identified relations between various kinds of data. The data itself would be organized in two-dimensional (row and column) tables and specific items in a table could be related to data located in other tables. Codd saw the need to reduce or eliminate redundancy in data and to allow data to be accessed through logical rather than physical identification. One of Codd's key ideas was the process for organizing data into the right number of tables, a process known as normalization.
- To access data using this relational model, Codd envisioned a simple language based on relational set theory. Codd also believed that a database management system should provide a standard access approach so that an application program did not have to be aware of how the data was organized. As a result, IBM in 1982 came out with the first version of what later became the Structured Query Language (SQL). In 1977, Oracle became the first commercial relational database management system. IBM's DB2 followed in 1981.
- Retiring from IBM after a serious injury in the early 1980s, Codd had his own consulting group until 1999. He died on April 18, 2003 at his home on Williams Island, Florida.
- http://searchdatabase.techtarget.com/sDefinition/0,,sid13_gci895554,00.html

Related Terms

- Relational Database
 - http://searchdatabase.techtarget.com/sDefinition/0,,sid13_gci212885,00.html
- Normalization
 - http://searchdatabase.techtarget.com/sDefinition/0,,sid13_gci212669,00.html
- Set Theory
 - Http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci333100,00.html
- Database Management System
 - http://searchdatabase.techtarget.com/sDefinition/0,,sid13_gci213669,00.html
- Structured Query Language
 - http://searchdatabase.techtarget.com/sDefinition/0,,sid13_gci214230,00.html
- Oracle
 - http://searchdatabase.techtarget.com/sDefinition/0,,sid13_gci214534,00.html

Geschichte der Datenbanken

- http://ycmi.med.yale.edu/nadkarni/db_course/History_Contents.htm
- Historical Perspective and Modern Developments
- Databases were originally collections of flat files, manipulated by programs that were typically written in COBOL. Typically, multiple copies of the same data were maintained, each copy sorted in a different way. The invention of the index (a data structure on disk that enabled one to look at the data as though it were sorted on one or more fields) changed all that. A single flat file could have as many indexes (on as many fields) as necessary, making multiple copies unnecessary.
- Hierarchical Databases (mid 60s)
- The first attempt at a database was IBM's IMS, which followed what was termed the hierarchical model. This allowed one-to-many relationships (e.g., one manager, many subordinates) but had a problem with many-to-many relationships. (E.g., if a subordinate reported to more than one manager, you had to duplicate the subordinate's data.)
- Network Databases (late 60s)
- The Network model, developed in response to this shortcoming, allowed many-to-many relationships. They are fast and efficient (network databases are still used to this day in applications such as satellite communications and airline reservations, where very fast response times are needed). This is partly because they use "hard pointers": references to the contents of one table from another actually refer to the physical address of the relevant data on disk, bypassing the need for indexes.
- However, network databases require a lot of programming to use successfully- unlike relational databases, they cannot be used by non-programming power users out of the box. Also, they are inflexible: once the data is organized in a particular way, it is hard to change one's mind. Data reorganization involves the equivalent of major surgery.
- Relational Databases (late 60s to present)
- Today's de facto standard databases were conceived in late 60s by Edgar F. Codd at IBM. Codd was a mathematician, and the heavily mathematical tone of his papers was partly responsible for the delayed acceptance of his work. Codd's idea was to define a set of operations that would work on databases the way algebraic operators worked on numbers. The first pilot implementation (at IBM San Jose by a team led by Chris Date in the early 70s) demonstrated the viability of the concept.
- In keeping with the ideas of COBOL, relational database implementation was thought to be facilitated by creation of an English-like Query Language. The language created for this purpose was called SQL (Structured Query Language), though it eventually grew in scope to handle other tasks such as editing/modification of data, data definition (e.g., specifying the structure of individual tables) and database security (e.g., specifying the privileges of individual database users for different tables in the database).
- Object-Oriented Databases (OODBs) and Object-Relational Databases (late 80s to present)
- Object-orientation is a programmer's rather than an end-user's concept. To understand this term (which has been one of the buzzwords in database marketing during the 90's), you must first understand the concept of a class. A class is a "kind of thing" that we want to represent information about- a student, employee, product, etc. To use an over-simplification, it is roughly the equivalent of a table in a relational database. with an important enhancement: the permissible operations on a class ("methods") are also specified along with the

Geschichte der Datenbanken

- Erste Datenbanken
 - Sammlung flacher Dateien, programmiert mit COBOL
 - Mehrere Kopien, jeweils anders sortiert
 - Indexe wurden eingeführt
- Hierarchische Datenbanken (Mitte 60er Jahre)
 - Erlaubten one-to-many relationships
- Netzwerk-Datenbanken (späte 60er)
 - Erlaubte many-to-many relationships
 - Nicht von Nicht-Programmierern verwendbar
- Relationale Datenbanken (späte 60er bis heute)
 - Entwickelt von Edgar F. Codd (Mathematiker) bei IBM
 - SQL (Query, obwohl die Sprache mehr als nur „Fragen“ beherrscht)
- Objekt-Orientierte Datenbanken (OODBs) und Objekt-Relationale Datenbanken (späte 80er bis heute)
 - Klassen, Objekte → OOP

RDBMS

- Relational
- Data
- Base
- Management
- System

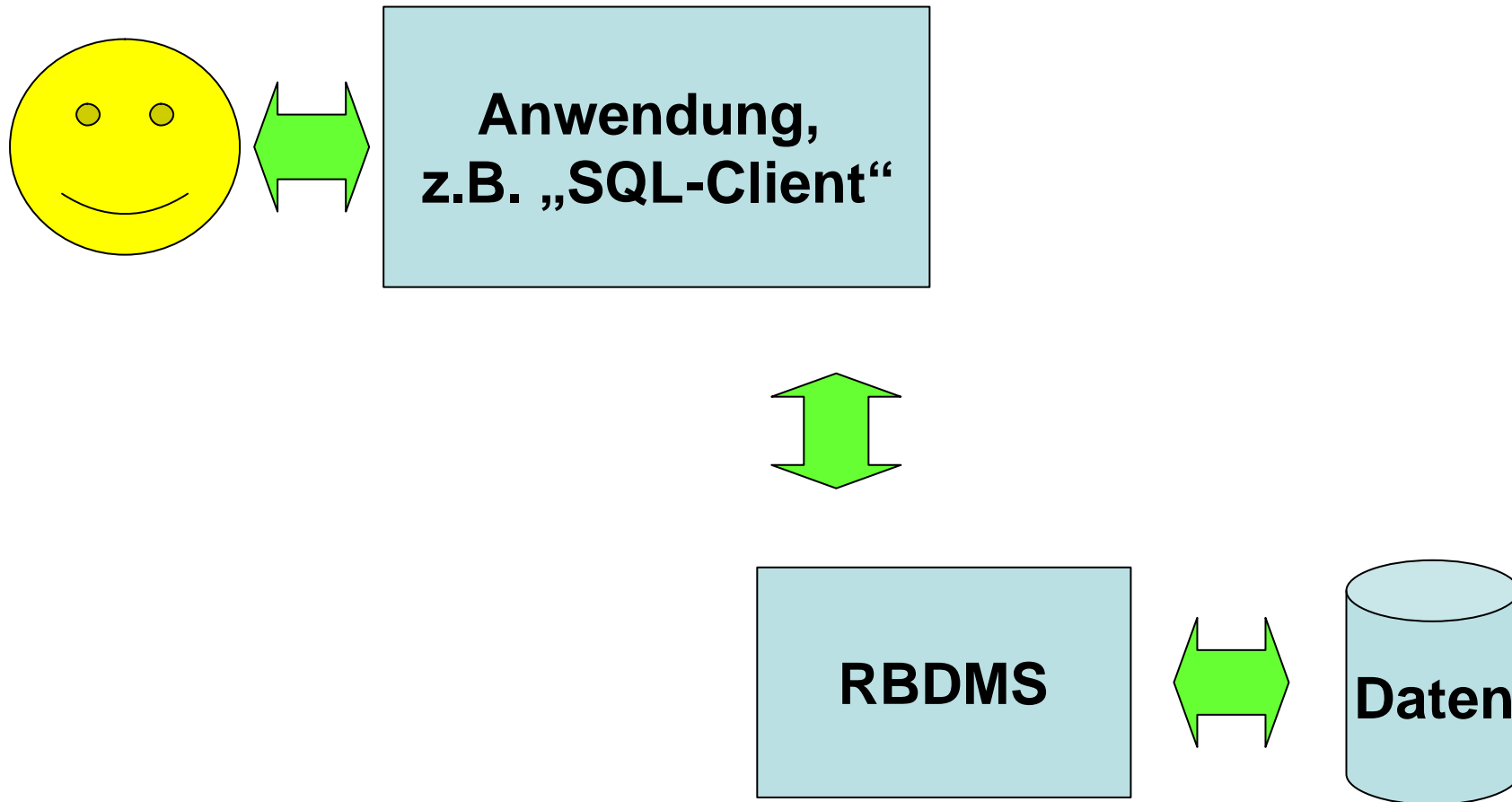
Warum RDBMS?

- Oder: Wann soll ich ein RDBMS einsetzen?
- Wenn Sie eine der folgenden Funktionen brauchen und nicht noch einmal entwerfen, implementieren und QA'en wollen / dürfen / können:
 - Dateiformate und Zugriff
 - Suche mit Indexierung
 - Mehrbenutzerbetrieb
 - Transaktionen
 - Schutz der Daten vor Systemfehlern → Integrität
 - Mehr Daten als in den RAM passen → Caching

Warum RDBMS: Ein Vergleich

- <http://www.jegsworks.com/Lessons/lesson1-2/lesson2-4database.htm>

RDBMS-Architektur



Vorteile der RDBMS Architektur

- Trennung:
 - Physikalische Sicht z.B. Dateien
 - Logische Sicht z.B. Tabellen
 - Anwendersicht z.B. Bestellung
- Integrität der Daten
- Transaktionen
 - Mehrbenutzer Betrieb "multi user"
 - Systemfehler
- In der Praxis: „N tier architecture“
 - Web-browser, load balancer, web-server, [GUI generation, business logic] @ servlet container <JDBC> RDBMS

RDBMS: Grundlagen

- Relationale Modell
- SQL
- Datenmodellierung
- Zugriff auf Daten von Programmiersprachen, z.B. Java via JDBC

Relationale Modell

- Relationale Begriffe
- Relationale Integrität
- Relationale Algebra / Operationen

Relationale Begriffe

- Relation Tabelle Rx
- Attribut Spalte/Feld AttrY
- Domäne Wertebereich
—Vgl. SQL-Datentypen
- Tupel Zeile/Datensatz Tn
- Grad Zahl der Spalten Deg(Rx)
- Kardinalität Zahl der Zeilen Card(Rx)
- NULL: Unbekannter/undefinierter Wert

Relation

| Art_Nr | Art_Bez | Art_Art | Lief_Nr |
|---------------|----------------|----------------|----------------|
| 1 | MultiSync I | Monitor | 1 |
| 2 | MultiSync II | Monitor | 2 |
| 3 | Wonder 3 | Grafikkarte | 1 |
| 4 | Flat 14" | Monitor | 3 |
| 5 | Deskjet 693 | Drucker | 4 |
| 6 | LaserJet Z | Drucker | 3 |

Domain, Relation, Attribut, Tupel, Degree, Cardinality

Warum „Relation“ statt Tabelle?

- Studi „fährt“ Auto
- Die Tabelle beschreibt die Relation „fährt“ zwischen „Studi“ und „Auto“.
- Studi „beteiligt“ sich an einer Lehrveranstaltung
- Studi und Lehrveranstaltung sind durch „beteiligt“ verknüpft.

| MNr | Studi | Auto |
|-----|---------|--------|
| 1 | Meyer | Kaefer |
| 2 | Mueller | 911 |
| 3 | Schmidt | Trabi |
| ... | ... | ... |

| MNr | Studi | Lehrveranst |
|-----|---------|-------------|
| 1 | Meyer | DB I |
| 2 | Mueller | C++ |
| 3 | Schmidt | SW Eng. |
| ... | ... | ... |

Schlüssel

- Schlüssel / „Key“
 - Kombination von Attributen die eine eindeutige Identifikation jeder Zeile ermöglichen.
- Minimaler Schlüssel / „Candidate Key“
 - Ein Schlüssel mit einer minimaler Zahl von Attributen, d.h. kein Attribut kann gelöscht werden ohne das die Eigenschaft Schlüssel zu sein verloren geht.
- Hauptschlüssel / Primary Key „PK“
 - Ein ausgewählter Candidate Key
- Fremdschlüssel / Foreign Key „FK“
 - Eine Attribute, welches einen PK einer anderen Tabelle enthält.

Relationale Operationen

- Restriction Zeilenselektion/Teilmenge
- Projection Spaltenselektion
- Union Vereinigung
- Intersection Schnittmenge/Durchschnitt
- Difference Differenz
- Product Kartesisches Produkt
- Join Verbindung
- Division Division

Eigenschaften v. Relationen

- Keine doppelten Tupel / Zeilen
 - Keine Tupel gleichen Wertes in derselben Tabelle.
- Tupelreihenfolge beliebig
 - Die Reihenfolge der Zeilen in einer Tabelle ist nicht definiert.
- Attributreihenfolge beliebig
- Attributewerte sind atomar
 - Es gibt keine Teile von Attributen, z.B. Name.Vorname
 - (Erste Normalform, später mehr!)

Relationale Integrität

- „Entity Integrity“
 - Ein Primary Key darf nicht den Wert NULL annehmen.
- „Referential Integrity“
 - Zu jedem Wert eines FK der Relation R1 muss ein PK in der Relation R2 existieren.
 - Der Wert des FK darf nicht NULL sein.
 - (Kurz: FK dürfen nicht ins „Leere“ zeigen.)
- (Mehr zur Sicherstellung der Relationalen Integrität durch das RDBMS nach der SQL Einführung.)

Relationale Operationen

- Motivation
 - Grundlagen
 - Begriffswelt
 - Vorbereitung auf SQL
- Das Ergebnis einer Operation auf Relationen ist wieder eine Relation.

Restriction

- Zeilenauswahl
- REST(Relation, Bedingung)
- REST(R1, Attr2 = 'A')

| Attr1 | Attr2 |
|-------|-------|
| 1 | A |
| 2 | B |
| 3 | C |

| Attr1 | Attr2 |
|-------|-------|
| 1 | A |
| | |
| | |

Projection

- Spaltenauswahl
- $\text{PROJ}(\text{Relation}, \langle \text{Attr1}, \text{Attr2}, \dots, \text{attrN} \rangle)$
- $\text{PROJ}(\text{R1}, \langle \text{Attr2} \rangle)$

| Attr1 | Attr2 |
|-------|-------|
| 1 | A |
| 2 | B |
| 3 | C |

| Attr2 |
|-------|
| A |
| B |
| C |

Union

- Vereinigung
- UNION(Relation1, Relation2)
- UNION(R1, R2)

| Attr1 |
|-------|
| A |
| B |
| C |

| Attr1 |
|-------|
| X |
| Y |



| Attr1 |
|-------|
| A |
| B |
| C |
| X |
| Y |

Union kompatibel

1. $\text{Deg}(R1) = \text{Deg}(R2)$
 2. Für alle N aus $[1, \text{Deg}(R1)]$ gilt:
 $\text{Domain}(R1.\text{Attr}N) = \text{Domain}(R2.\text{Attr}N)$
- Kurz: Die Relationen müssen denselben Aufbau haben.

Intersection

- Schnittmenge
- `INTERSECTION(Relation1, Relation2)`

| Attr1 |
|-------|
| A |
| B |
| C |

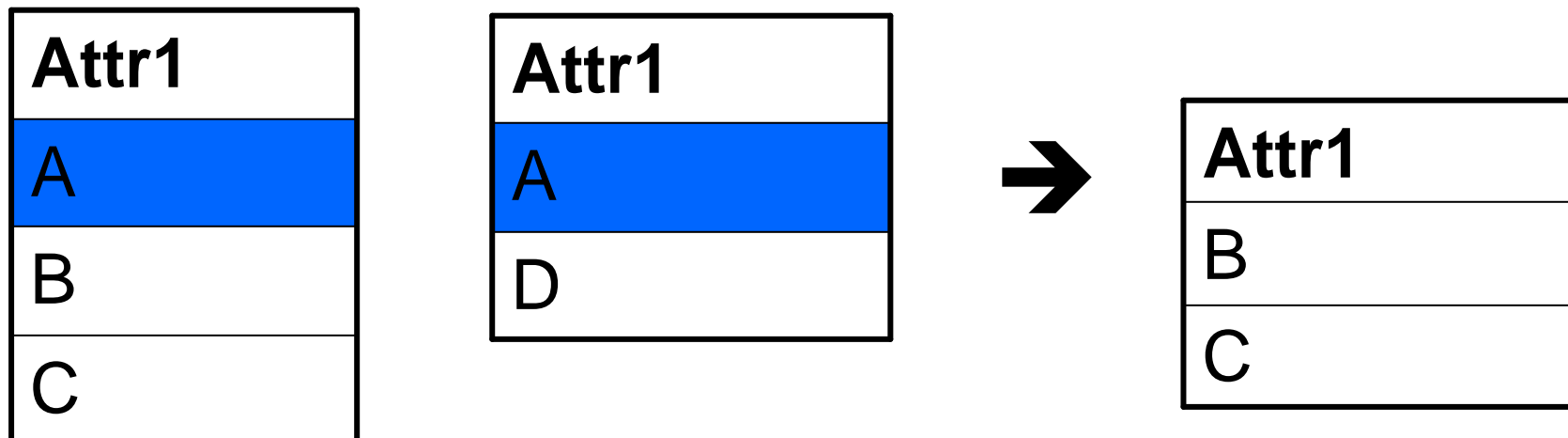
| Attr1 |
|-------|
| A |
| B |



| Attr1 |
|-------|
| A |
| B |

Difference

- Substraktion
- `DIFFERENCE(Relation1, Relation2)`



Product

- Kartesisches Product
- $\text{PRODUCT}(\text{Relation1}, \text{Relation2})$
- $\text{PRODUCT}(R1, R2)$

| Attr1 |
|-------|
| A |
| B |
| C |

X

| Attr1 |
|-------|
| X |
| Y |



| Attr1 | Attr2 |
|-------|-------|
| A | X |
| A | Y |
| B | X |
| B | Y |
| C | X |
| C | Y |

Join

- Natural JOIN
- JOIN(R1, A2 = A4, R2)

| A1 | A2 |
|----|----|
| A | W |
| B | X |
| C | Y |

| A3 | A4 |
|----|----|
| C | Y |
| D | Z |

| A1 | A2 | A3 |
|----|----|----|
| C | Y | C |

Join = Product + Rest

| A1 | A2 |
|----|----|
| A | W |
| B | X |
| C | Y |

| A3 | A4 |
|----|----|
| C | Y |
| D | Z |



| A1 | A2 | A3 | A4 |
|----|----|----|----|
| A | W | C | Y |
| A | W | D | Z |
| B | X | C | Y |
| B | X | D | Z |
| C | Y | C | Y |
| C | Y | D | Z |

JOIN(R1, A2 = A4, R2)

| A1 | A2 | A3 |
|----|----|----|
| C | Y | C |



Join = Product + Rest

JOIN(R1, A3 = A4, R2)

=

REST(PROD(R1,R2),
R1.A3 = R2.A4)

Join: Eigenschaften

- Attribute im „Restrictionsteil“
 - Keine Schlüssel
 - Gleiche Domänen
- Relationen können mit jeder anderen Relation "gejoint" werden.
- Relationen können mit sich selbst gejoint werden.

Joins

- THETA JOIN
 - Beliebige Vergleichsoperation im „Restrictionteil“
- EQUI JOIN
 - „=“ im Restrictionsteil
- Natural JOIN
 - Ergebnisrelation enthält die gleichen Attribute nur einmal (siehe Bsp.)
- Auto JOIN
 - Join einer Relation mit sich selber

Inner Join

| A1 | A2 |
|----|----|
| A | W |
| B | X |
| C | Y |

| A3 | A4 |
|----|----|
| | |
| C | Y |
| D | Z |

JOIN (R1, A2=A4, R2)

| A1 | A2 | A3 | A4 |
|----|----|----|----|
| C | Y | C | Y |

Left Outer Join

| A1 | A2 |
|----|----|
| A | W |
| B | X |
| C | Y |

| A3 | A4 |
|----|----|
| C | Y |
| D | Z |

| A1 | A2 | A3 | A4 |
|----|----|------|------|
| A | W | Null | Null |
| B | X | Null | Null |
| C | Y | C | Y |

JOIN (R1, A2 *= A4, R2)

Der Inhalt der linken “left” Relation ist Bestandteil der Ergebnisrelation.

Right Outer Join

| A1 | A2 |
|----|----|
| A | W |
| B | X |
| C | Y |

| A3 | A4 |
|----|----|
| C | Y |
| D | Z |

| A1 | A2 | A3 | A4 |
|------|------|----|----|
| C | Y | C | Y |
| Null | Null | D | Z |

JOIN (R1, A2 =* A4, R2)

Der Inhalt der rechten “right” Relation wird in das Ergebnis übernommen.

Division

- $\text{DIVISION}(\text{Relation1}, \text{Attr1}, \text{Relation2})$
- $\text{DIVISION}(R1, A1, A2/A3, R2)$

| A1 | A2 |
|----|----|
| A | X |
| A | Y |
| A | Z |
| B | X |
| C | Y |

| A3 |
|----|
| X |
| Z |



| A1 |
|----|
| A |

Division: Umkehrung

DIVISION(R1, A1, A2/A3, R2)

- $R1/R2 \Rightarrow R3$ $5/3 \Rightarrow 1$
- $R3 * R2 \Rightarrow R1$ $1 * 3 \Rightarrow 3$
- PROD(R3,R2)

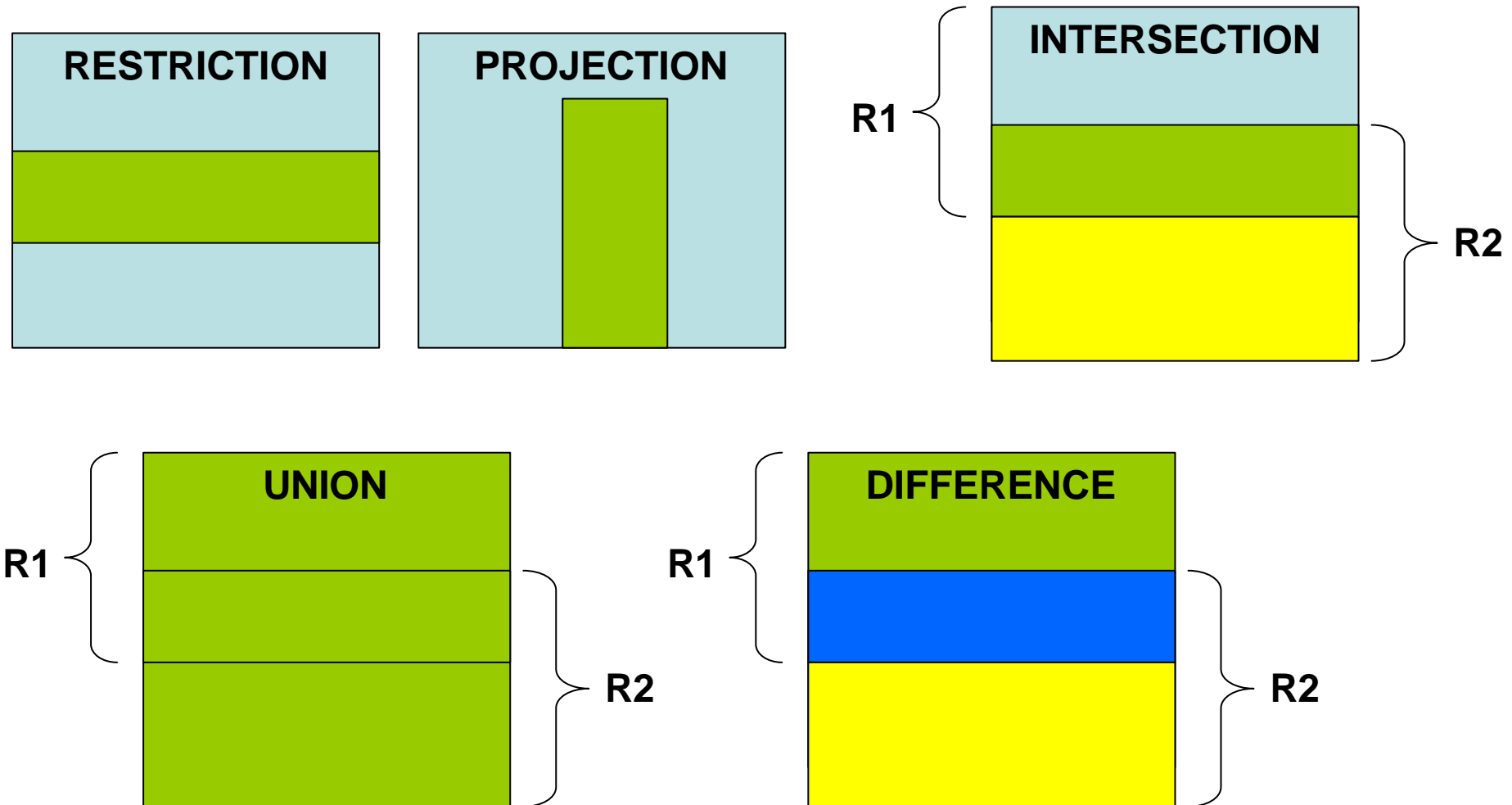
| A1 |
|----|
| A |

| A3 |
|----|
| X |
| Z |

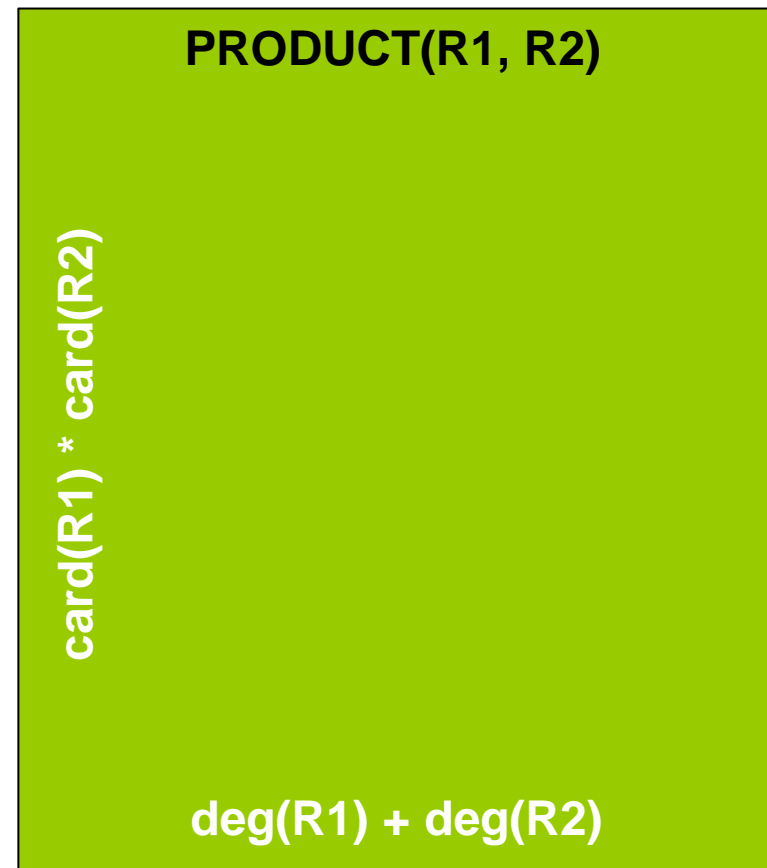
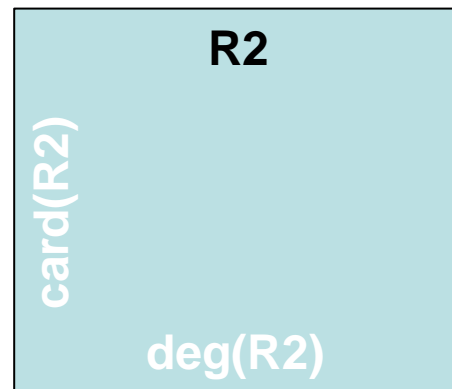
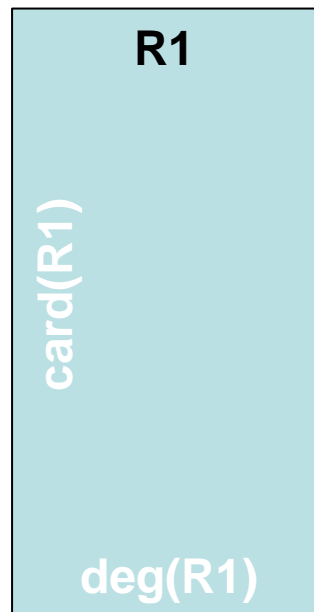


| A1 | A2 |
|----|----|
| A | X |
| A | Y |
| A | Z |
| B | X |
| C | Y |

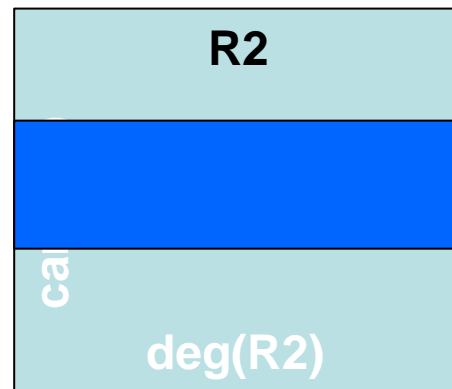
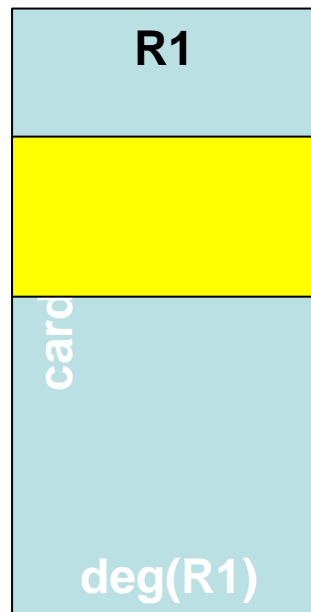
Visualisierung Restriction ... Difference



Visualisierung: Product



Visualisierung: Join



Relationale Operationen: Grad und Kardinalität

| | Grad | Kardinalität |
|---------------------|-------------------------------------|---|
| Restriction | $= \text{deg}(R1)$ | $\leq \text{card}(R1)$ |
| Projection | $\leq \text{deg}(R1)$ | $\leq \text{card}(R1)$ |
| Union | $= \text{deg}(R1) = \text{deg}(R2)$ | $\leq \text{card}(R1) + \text{card}(R2)$ |
| Intersection | $= \text{deg}(R1) = \text{deg}(R2)$ | $\leq \min\{\text{card}(R1), \text{card}(R2)\}$ |
| Difference | $= \text{deg}(R1) = \text{deg}(R2)$ | $\leq \text{card}(R1)$ |
| Product | $= \text{deg}(R1) + \text{deg}(R2)$ | $= \text{card}(R1) * \text{card}(R2)$ |
| Join | $= \text{deg}(R1) + \text{deg}(R2)$ | $\leq \text{card}(R1) * \text{card}(R2)$ |
| Division | $= \text{deg}(R1) - 1$ | $\leq \text{card}(R1)$ |

Beispiele Relationaler Operationen

Bsp. 1: REST(Artikel, Lief_Nr = 4)

| Art_Nr | Art_Bez | Art_Art | Lief_Nr |
|--------|--------------|-------------|---------|
| 1 | MultiSync I | Monitor | 1 |
| 2 | MultiSync II | Monitor | 2 |
| 3 | Wonder 3 | Grafikkarte | 1 |
| 4 | Flat 14" | Monitor | 3 |
| 5 | Deskjet 693 | Drucker | 4 |
| 6 | LaserJet Z | Drucker | 3 |
| | | | |

REST(Artikel, Lief_Nr = 4)

| Art_Nr | Art_Bez | Art_Art | Lief_Nr |
|---------------|----------------|----------------|----------------|
| 5 | Deskjet 693 | Drucker | 4 |

PROJ(REST(Artikel, Lief_Nr = 4), <Art_Bez>)

| Art_Nr | Art_Bez | Art_Art | Lief_Nr |
|--------|--------------|-------------|---------|
| 1 | MultiSync I | Monitor | 1 |
| 2 | MultiSync II | Monitor | 2 |
| 3 | Wonder 3 | Grafikkarte | 1 |
| 4 | Flat 14" | Monitor | 3 |
| 5 | Deskjet 693 | Drucker | 4 |
| 6 | LaserJet Z | Drucker | 3 |

PROJ(REST(Artikel, Lief_Nr = 4), <Art_Bez>)

| Art_Bez |
|----------------|
| Deskjet 693 |

Beispiel 2: Lieferanten

| Lief_Nr | Lief_Name |
|---------|-----------|
| 1 | NEC |
| 2 | Siemens |
| 3 | ST |
| 4 | Medion |

Beispiel 2

- Welche Lieferanten liefern einen Monitor?

JOIN(Artikel,
Lief_Nr = Lief_Nr, Lieferanten)

REST(PRODUCT(Artikel, Lieferanten),
Lief_Nr = Lief_Nr)

Relationen: Artikel und Lieferanten

| Art_Nr | Art_Bez | Art_Art | Lief_Nr |
|--------|--------------|-------------|---------|
| 1 | MultiSync I | Monitor | 1 |
| 2 | MultiSync II | Monitor | 1 |
| 3 | Wonder 3 | Grafikkarte | 2 |
| 4 | Flat 14" | Monitor | 3 |
| 5 | Deskjet 693 | Drucker | 4 |
| 6 | LaserJet Z | Drucker | 4 |

| Lief_Nr | Lief_Name |
|---------|-----------|
| 1 | Sony |
| 2 | ATI |
| 3 | Samsung |
| 4 | HP |

JOIN(Artikel, Lief_Nr = Lief_Nr, Lieferanten)

| Art_Nr | Art_Bez | Art_Art | Lief_Nr | Lief_Name |
|---------------|----------------|----------------|----------------|------------------|
| 1 | MultiSync I | Monitor | 1 | Sony |
| 2 | MultiSync II | Monitor | 1 | Sony |
| 3 | Wonder 3 | Grafikkarte | 2 | ATI |
| 4 | Flat 14" | Monitor | 3 | Samsung |
| 5 | Deskjet 693 | Drucker | 4 | HP |
| 6 | LaserJet Z | Drucker | 4 | HP |

**REST(JOIN(Artikel, Lief_Nr = Lief_Nr,
Lieferanten), Art_Art = 'Monitor')**

| Art_Nr | Art_Bez | Art_Art | Lief_Nr | Lief_Name |
|---------------|----------------|----------------|----------------|------------------|
| 1 | MultiSync I | Monitor | 1 | Sony |
| 2 | MultiSync II | Monitor | 1 | Sony |
| 4 | Flat 14" | Monitor | 3 | Samsung |

PROJ(REST(JOIN(...), Art_Art = 'Monitor'),
<Lief_NAME>)

- Welche Lieferanten liefern einen Monitor?

| Lief_Name |
|-----------|
| Sony |
| Sony |
| Samsung |

Beispiel 2: Lösung

Welche Lieferanten liefern einen Monitor?

```
PROJ(  
  REST(  
    JOIN(Artikel, Lief_Nr = Lief_Nr, Lieferanten),  
      Art_Art = 'Monitor'),  
  <Lief_NAME>)
```

Division: Beispiel

| Pilot | Skills |
|---------|--------|
| Celko | Cub |
| Higgins | B-52 |
| Higgins | F-14 |
| Higgins | Cub |
| Jones | B-52 |
| Jones | F-14 |
| Smith | B-1 |
| Smith | B-52 |
| Smith | F-14 |

| Airplane |
|----------|
| B-1 |
| B-52 |
| F-14 |

| DIVISION(R1,R2) |
|------------------------|
| Smith |

Domain (Wertebereich) → SQL-Datentypen

| | DB2 | Oracle |
|----------------|---------|---------|
| INTEGER 8 bit | | |
| INTEGER 16 bit | X | X |
| INTEGER 32 bit | X | X |
| DECIMAL | X | X |
| REAL 32 bit | | |
| REAL 64 bit | X | |
| REAL 128 bit | | |
| CHAR(n) | 254 max | 254 max |
| VARCHAR(n) | X | X |
| DATE | | X |

→ Konsultieren Sie das Manual des RDBMS, welches Sie verwenden wollen.

DROP TABLE

- DROP TABLE Lehrveranstaltung;