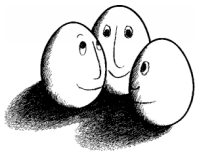
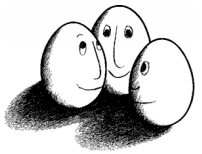


Relationale Datenbanken

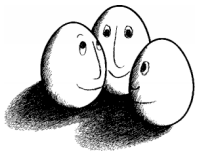
- Relationales Datenmodell
- Deklarationen
- Anfragen

Prof. Dr. Katharina Morik



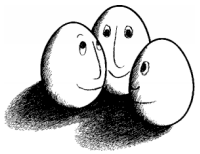
Datenbanksysteme

- Entwurf, z.B. mit Entity Relationship Model
- Deklaration
- Speichern der Daten
  - Hauptspeicher, Cache, virtueller Speicher, Platte
  - Indexierung, z.B. mit B-Bäumen
- Anfragen
  - Syntax in SQL, Semantik
  - Pläne zur Ausführung
  - Optimierung
- Änderungen (Transaktionen)



Literatur

- Joachim Biskup „Grundlagen von Informationssystemen“ vieweg Vg. 1995
- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom „Database Systems – The Complete Book“ Prentice Hall 2002 (1119 Seiten)



Relation

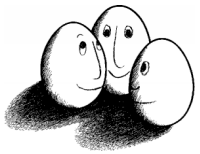
Attribute (Spaltennamen)
A = { titel, jahr, dauer, foto }

Tupel (Zeilen)
t: $A^m \rightarrow C^m$
C sind Konstante (Werte)

Relation r
Menge von Tupeln mit gleichem Definitionsbereich

Filme

titel	jahr	dauer	foto
Star Wars	1977	124	farbig
Mighty Ducks	1991	104	farbig
Wayne‘s World	1992	95	farbig



Schema

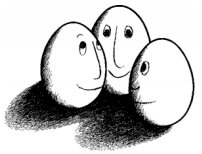
Name der Relation und Menge von Attributen mit Wertebereichen, hier:

Filme(
titel:string,
jahr:integer,
dauer:integer,
foto:{farbig, sw}
)

Wertebereiche sind einfache Datentypen: integer, string, date, Aufzählung

Filme

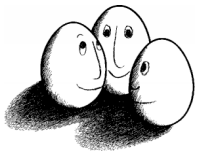
titel	jahr	dauer	foto
Star Wars	1977	124	farbig
Mighty Ducks	1991	104	farbig
Wayne‘s World	1992	95	farbig



DDL – Tabelle anlegen

- Data Definition Language, Teilmenge von SQL
- CREATE TABLE FilmeTest
  (titel VARCHAR2(18),
  jahr NUMBER);
- DESCRIBE FilmeTest liefert:

Name	Type
titel	VARCHAR2(18)
jahr	NUMBER



DDL – Relationenschema ändern

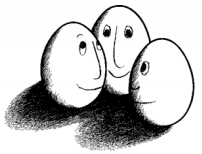
- ALTER TABLE FilmeTest
  ADD (star VARCHAR2(16));

- DESCRIBE FilmeTest liefert nun:

Name	Type
titel	VARCHAR2(18)
jahr	NUMBER
star	VARCHAR2(16)

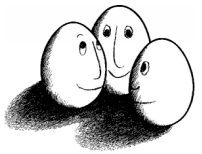
- ALTER TABLE FilmeTest
  MODIFY (titel VARCHAR2(20));

  ändert den Datentyp der Einträge des Attributes titel, so dass der Titel jetzt 20 Zeichen lang sein kann.



Instanzen eines Schemas

- Relationenschema der Form $R(A_1:D_1, ..., A_n:D_n)$
- (d,r) ist eine Instanz von $R(A_1:D_1, ..., A_n:D_n)$ gdw.
  - $d \subset C$,
  - $dom(r) = \{A_1, ..., A_n\}$
  - $t(A_i) \in D_i$
- Eine Instanz heißt auch „Zustand“. Das Schema ändert sich (fast) nie, der Zustand sehr häufig.

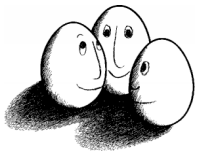


DML - Instanz erzeugen oder erweitern

- INSERT INTO FilmeTest (titel, jahr, star)
  VALUES (`Star Wars´, 1988, `C. Fisher´);

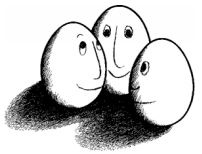
- Zeilenweise werden Tupel eingefügt

- Wenn man die Daten aus einer anderen Tabelle bekommen kann, darf statt VALUES ein SELECT-Ausdruck stehen, der Tupel aus der gegebenen Tabelle holt.



Datenbank und Datenbankschema

- Eine Menge von Relationenschemata ist ein Datenbankschema.
- Eine Menge von Instanzen von Relationenschemata ist ein Datenbankzustand (kurz: Datenbank).
- Metadaten beschreiben Daten. Ein Datenbankschema beschreibt eine Datenbank.



Funktionale Abhängigkeiten

Funktionale Abhängigkeiten (functional dependencies, FD) liegen vor, wenn einige der Attribute eindeutig die Werte anderer Attribute bestimmen, also:

FD: $A_1, ..., A_n \rightarrow B$

Wenn $t_1(A_1, ..., A_n) = t_2(A_1, ..., A_n)$

dann $t_1(B) = t_2(B)$ für beliebige r in R

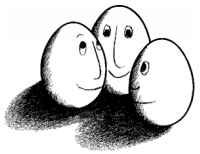
$A_1, ..., A_n \rightarrow B_1$

$A_1, ..., A_n \rightarrow B_2$ ...

$A_1, ..., A_n \rightarrow B_n$

wird geschrieben:

FD: $A_1, ..., A_n \rightarrow B_1, B_2, ..., B_n$



Beispiel

Filme

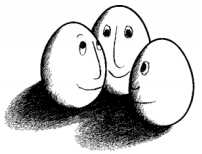
titel	jahr	dauer	foto	studio	star
Star Wars	1977	124	farbig	Fox	C.Fisher
Star Wars	1977	124	farbig	Fox	M.Hamill
Star Wars	1977	124	farbig	Fox	H.Ford
Mighty Ducks	1991	104	farbig	Disney	E.Estevez
Wayne‘s World	1992	95	farbig	Paramount	D.Carvey
Wayne‘s World	1992	95	farbig	Paramount	M.Meyers

FD:

titel,jahr

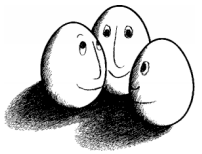
→

dauer,
foto,
studio



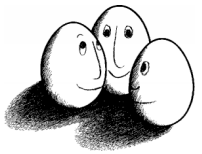
Schlüssel

- Eine Menge von Attributen $\{A_1, ..., A_n\}$ heißt Schlüssel für alle Relationen r zum Relationenschema R gdw.
  - Die Attribute funktional alle anderen Attribute der Relation determinieren.
  - Keine echte Teilmenge von $\{A_1, ..., A_n\}$ determiniert funktional alle anderen Attribute.



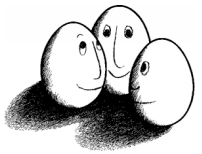
Schlüssel

- Eine Menge von Attributen $\{A_1, ..., A_n\}$ heißt Schlüssel für eine Relation r in R gdw.
  - Die Attribute funktional alle anderen Attribute der Relation determinieren.
    - Folglich können keine zwei Tupel dieselben Werte in $\{A_1, ..., A_n\}$ haben!
  - Keine echte Teilmenge von $\{A_1, ..., A_n\}$ determiniert funktional alle anderen Attribute.
    - Aber es kann verschiedene Schlüssel unterschiedlicher Länge geben! $\{A_1, ..., A_4\}$ $\{A_5, A_6\}$



Oberschlüssel

- Eine Menge von Attributen Y heißt Oberschlüssel, wenn eine echte Teilmenge $X \subset Y$ ein Schlüssel ist.
- Auch der Oberschlüssel determiniert alle anderen Attribute, ist aber nicht minimal.
- Trivialerweise ist die Menge aller Attribute in R ein Oberschlüssel für R.

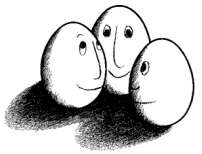


In anderen Worten...

- Funktionale Abhängigkeiten und Schlüssel sind Eigenschaften des Schemas, nicht des Zustands einer Datenbank.

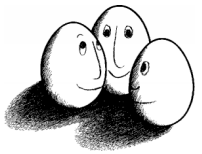
- Wir überlegen, welche Attribute wir als Schlüssel nehmen wollen.
  - Ein Film könnte mit demselben Titel in verschiedenen Jahren gedreht werden (remake).
  - Oft wird eigens ein Attribut eingeführt, das die Tupel durchnummeriert (ID).

- Das Speichern der Datenbank und Operationen nutzen Schlüssel aus!



Anomalien

- Redundanz
  - Warum muss für 1 Film n mal eingetragen werden, dass er in Farbe gedreht wurde?
- Eintraganomalie
  - Woran merken wir, dass wir bei n Tupeln desselben Films die Länge ändern müssen, wenn wir sie bei 1 Tupel dieses Films ändern?
- Löschanomalie
  - Woran merken wir, dass wir das letzte Tupel zu einem Film löschen und damit den Film insgesamt?

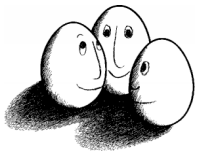


Dekomposition einer Relation

- Das Relationenschema $R(A_1:D_1, ..., A_n:D_n)$ wird aufgeteilt in $S(B_1:D_{i1}, ..., B_m:D_{im})$ und $T(C_1:D_{j1}, ..., C_k:D_{jk})$ so dass $\{B_1, ..., B_m\} \cup \{C_1, ..., C_k\} = \{A_1, ..., A_n\}$

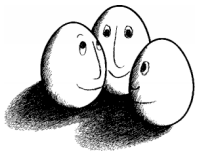
- Die Tupel in S sind Projektionen von R. Die Tupel von T sind Projektionen von R.

- Eine Projektion von R auf S besteht aus den Tupeln $t(B_1, ..., B_m)$.



Projektion

- Die Relation Filme(titel,jahr,dauer,foto,studio,star) wird projiziert auf Filme1(titel,jahr,dauer,foto,studio)
- Dabei ergeben die ersten drei Tupel dasselbe Ergebnis: Star Wars, 1977, 124, farbig, Fox
- Da eine Relation eine Menge von Tupeln ist, gibt es keine Doppelten, obwohl die Projektion von verschiedenen Tupeln dasselbe Ergebnis liefern kann!
- Aus 6 Tupeln in Filme erhalten wir 3 Tupel in Filme1.



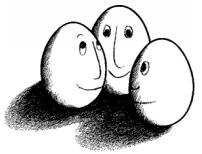
Beispiel

Filme2

titel	jahr	star
Star Wars	1977	C.Fisher
Star Wars	1977	M.Hamill
Star Wars	1977	H.Ford
Mighty Ducks	1991	E.Estevez
Wayne‘s World	1992	D.Carvey
Wayne‘s World	1992	M.Meyers

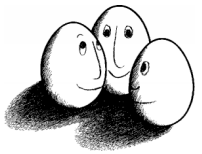
Filme1

titel	jahr	dauer	foto	studio
Star Wars	1977	124	farbig	Fox
Mighty Duck	1991	104	farbig	Disney
Wayne‘s World	1992	95	farbig	Paramount



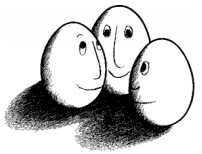
Anomalien entfernt

- Die Länge und die Fotografie wird nur noch für jeden Film einmal genannt, in Filme1.
  - Redundanz
  - Eingabeanomalie und
  - Löschanomalie sind nicht mehr vorhanden.



Was wissen Sie jetzt?

- Aus welchen Schritten besteht der Zyklus der Wissensentdeckung?
- Definieren Sie: Datenbank, Schema, Relation, Tupel!
- Und was ist eine funktionale Abhängigkeit?
- Was ist ein Schlüssel?



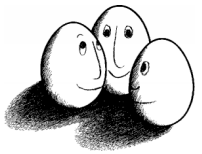
Boyce-Codd Normalform

- Für jede funktionale Abhängigkeit in einer Relation
  FD: $A_1, ..., A_n \rightarrow B$ gilt:

  $\{A_1, ..., A_n\}$ ist ein Oberschlüssel der Relation.

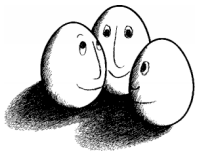
= Die linke Seite jeder funktionalen Abhängigkeit muss einen Schlüssel enthalten.

Ausschließlich funktionale Abhängigkeiten zwischen minimalen Schlüsseln und damit keine Redundanz!



Beispiel

- Filme ist nicht in Boyce-Codd-Normalform:
  - Schlüssel: titel, jahr, star
  - FD: titel, jahr →dauer, foto, studio
- Das Problem ist: {titel, jahr} determiniert nicht star.
- Deshalb ist durch die Dekomposition das Problem behoben: Filme1 und Filme2 sind in Boyce-Codd-Normalform.



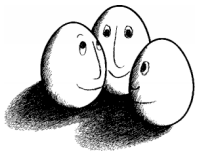
Normalformen

1. Jede Komponente eines Tupels, $t(A_i)$ hat als Wert einen einfachen (nicht zusammen-gesetzten, strukturierten) Datentyp.

2. Verboten sind nur funktionale Abhängigkeiten, deren linke Seite eine Untermenge eines Schlüssels sind.

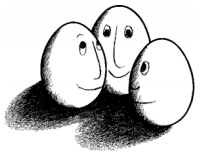
3. Für jede funktionale Abhängigkeit in einer Relation FD: $A_1, ..., A_n \rightarrow B$ gilt:

   $\{A_1, ..., A_n\}$ ist ein Oberschlüssel der Relation oder B ist Teil eines Schlüssels. (Schwächer als Boyce-Codd Normalform, Standard.)



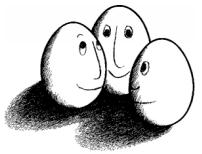
Was wissen wir jetzt?

- Wir wissen nicht, was ein Datenbanksystem ist.
- Wir wissen, dass eine Datenbank eine Menge von Relationen ist, die jeweils Mengen von Tupeln sind.
- Wir kennen die Metadaten (Relationenschema, Datenbankschema) und können ein Datenbankschema deklarieren (CREATE) und darstellen (DESCRIBE) lassen.
- Funktionale Abhängigkeiten und Schlüssel sind wichtige Eigenschaften des Schemas.
- Wir wissen, warum eine Datenbank in Boyce-Codd-Normalform sein sollte und haben eine Vorstellung, wie man die herstellt.



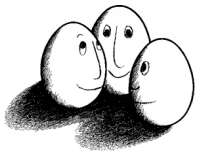
Anfragen

- **Algebra**
  Aus Operationen und Variablen oder Konstanten Ausdrücke gebildet,
  - z.B.
    Operationen:{+,-,*,/}, Variablen {x, y}, Konstante $\Re$
    Die Ausdrücke heißen (Un-)Gleichungen.
- **Relationenalgebra**
  - Variable und Konstante bezeichnen Relationen
  - Operationen bilden neue Relationen aus gegebenen Relationen.
  - Die Ausdrücke heißen Anfragen.



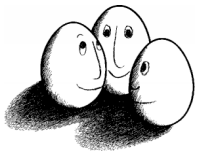
Operationen

- Vereinigung, Durchschnitt, Differenz
- Projektion
- Selektion
- Kartesisches Produkt
- Natürlicher Verbund



Vereinigung, Durchschnitt, Differenz

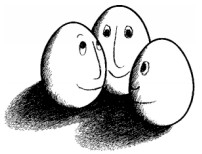
- Relationen R und S haben dasselbe Relationenschema und die Attribute werden in die gleiche Reihenfolge gebracht.
  - R ∪ S enthält alle Tupel von R und alle Tupel von S, aber keine Doppelten.
  - R ∩ S enthält alle Tupel, die in R und S vorkommen.
  - R\S enthält alle Tupel in R, die nicht in S vorkommen.



Projektion

projezieren eine Relation auf eine neue:

- π [A_1, ..., A_n](r) hat nur die Attribute A_1, ..., A_n von R.
  - π [titel,jahr, star] (Filme) ergibt genau Filme2.
- π_q (r) projiziert gemäß q, wobei q definiert werden muss.
  - π_q (Filme) mit q(star):=SchauspielerInnen ergibt Relation mit Attribut SchauspielerInnen statt star, Tupel bleiben.
  - π_q (Filme) mit q:=farbig ergibt die Relation:
    farbig
    true
    Es gibt nur ein Tupel, denn alle Filme waren Farbfilme.



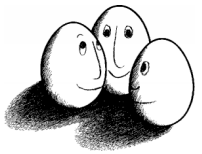
Selektion

σ[C](r):Die Selektion aus eine Relation r ergibt eine Relation mit der Teilmenge von Tupeln von r, die der Bedingung C genügen. Die Ergebnisrelation hat dasselbe Schema wie r.

C ist ein logischer Ausdruck, gebildet aus Gleichheit und Vergleichsoperatoren.

σ[dauer >=120](Filme1) ergibt

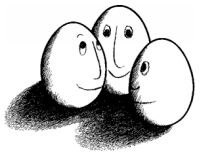
titel	jahr	dauer	foto	studio
Star Wars	1977	124	farbig	Fox



Structured Query Language -- Selektion

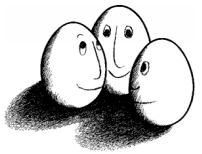
- $\pi[L](\sigma[C](r))$; L ist $A_1, ..., A_n$; C eine Bedingung und r eine Relation.

  In SQL: SELECT L FROM r WHERE C;
- SELECT * FROM Filme1; liefert Filme1
- SELECT studio FROM Filme1; liefert
  studio
  Fox
  Disney
  Paramount
- SELECT studio FROM Filme1 WHERE dauer>=120; liefert
  studio
  Fox



Bedingungen

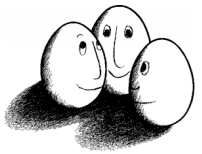
- Ergebnis der Anwendung einer Bedingung ist TRUE oder FALSE.
- Operatoren sind: NOT, AND, OR in der Reihenfolge ihrer Bindung. Alternativ: Klammerung.
- Vergleichsoperatoren sind: < , > , = , <= , >= , <> (ungleich)
- Arithmetische Operatoren, die auf Attributwerte vor dem Vergleich angewandt werden: + , * , - , /
- Konkatenation für Zeichenketten: `Disney`||`Studio´ ergibt `DisneyStudio´.



Beispiele SELECT

- SELECT titel
  FROM Filme
  WHERE jahr > 1970 AND NOT foto=`farbig´;

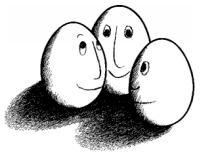
- SELECT titel
  FROM Filme
  WHERE ((jahr –1980)*(jahr – 1980) < 100 OR
  dauer < 90) AND
  studio=`Disney´;



Kartesisches Produkt

r X s ergibt eine Relation,

- deren Schema alle Attribute von r und alle Attribute von s enthält. Sollten Attribute in r und s gleich heißen, werden sie umbenannt.
- Alle Kombinationen von Tupeln in r und Tupeln in s bilden die Tupel in r X s.



Kartesisches Produkt -- Beispiel

r

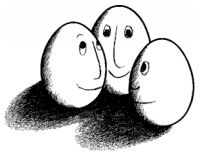
A	B
1	2
5	6

s

B	C	D
2	3	4
6	7	8

A	r.B	s.B	C	D
1	2	2	3	4
1	2	6	7	8
5	6	2	3	4
5	6	6	7	8

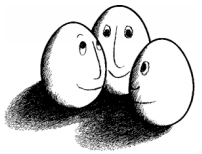
Furchtbar!



Natürlicher Verbund

$r \triangleright\triangleleft s$

Im Schema von r und dem von s kommen gleichermaßen die Attribute A_1, ..., A_n vor. Es sollen nur die Tupel kombiniert werden, die für diese Attribute in r und s denselben Wert haben. Alle anderen sind nicht in der Ergebnisrelation.



Natürlicher Verbund – Beispiel1

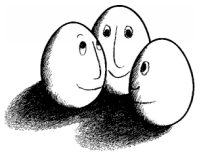
r

A	B
1	2
5	6

s

B	C	D
2	3	4
6	7	8

A	B	C	D
1	2	3	4
5	6	7	8

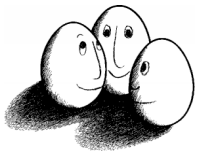


Natürlicher Verbund – Beispiel2

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

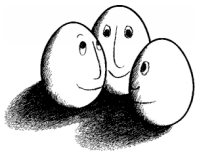
B	C	D
2	3	4
2	3	5
7	8	10

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10



Komplexität

- Falls r und s keine Attribute gemeinsam haben, so ist der natürliche Verbund gleich dem kartesischen Produkt. O(||r|| ||s||).
- Falls r und s alle Attribute gemeinsam haben, so ist der natürliche Verbund gleich dem Durchschnitt.



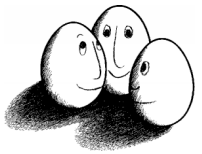
Multirelationale SQL-Anfragen

- FROM zeigt Liste von Relationen
  SELECT und WHERE beziehen sich auf angegebene Attribute von irgendeiner der Relationen.

- Ein Attribut A in r wird r.A geschrieben.

- ```
  SELECT Filme1.studio
  FROM Filme1, Filme2
  WHERE star=`H.Ford´ AND
            Filme1.titel=Filme2.titel AND
            Filme1.jahr=Filme2.jahr;
  ```

- π [Filme1.studio] (σ [Filme1.titel=Filme2.titel AND
  Filme1.jahr=Filme2.jahr] (Filme1 X Filme2))
  ```



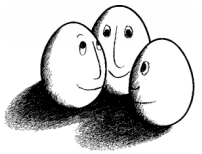
Vereinigung, Durchschnitt, Differenz in SQL

- (SELECT titel, jahr
  FROM Filme1
  WHERE studio=`Fox´)
  INTERSECT
  (SELECT titel, jahr
  FROM Filme2
  WHERE star=`H.Ford`);

  liefert alle Filme, die H. Ford bei Fox gedreht hat.

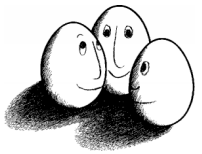
- Analog: **UNION**, **EXCEPT**

- Beide SELECT-Anfragen müssen Relationen mit demselben Schema ergeben!



Tupelvariablen

- Projektion eines Relationsnamen auf einen neuen Namen, die Tupelvariable (alias-Name).
- Damit Verbund oder Mengenoperation auf eine Tabelle angewandt werden kann, werden für diese Relation zwei Tupelvariablen eingeführt.
- Danach geht alles wie bei der multirelationalen Anfrage.

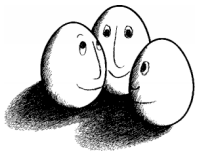


Beispiel

Stars

name	adresse
H.Ford	789 Palm Drive, Beverly Hills
C.Fisher	123 Maple Street, Hollywood

- SELECT Stars1.name, Star2.name
  FROM Stars Stars1, Stars Stars2
  WHERE Stars1.adresse=Stars2.adresse AND
  Stars1.name < Stars2.name;
- Liefert alle Tupel mit zwei Namen von zusammen wohnenden Stars. Ohne die zweite Bedingung würde derselbe Name zweimal in einem Tupel auftreten können.

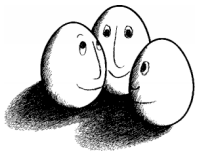


Ergebnis

Stars1.name	Stars2.name
Alec Baldwin	Kim Basinger
Calista Flockhart	Harrison Ford

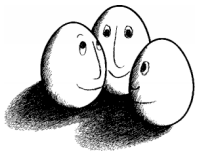
- Wäre <> statt < angegeben, wäre jedes Tupel zweimal, in anderer Reihenfolge vorgekommen.

Stars1.name	Stars2.name
Alec Baldwin	Kim Basinger
Kim Basinger	Alec Baldwin



Sichten (views)

- Relationen, deren Schema mit CREATE deklariert wurde, existieren physikalisch in der Datenbank.
- Relationen, deren Schema mit CREATE VIEW deklariert wurde, werden nicht gespeichert, sondern bei Bedarf berechnet.
- CREATE VIEW r‘ AS q
  wobei q eine Anfrage ist.

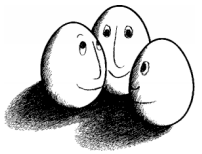


Beispiel für eine Sicht

- CREATE VIEW DisneyFilme AS
  SELECT titel, jahr
  FROM Filme1
  WHERE studio=`Disney´;

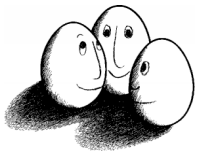
- Die **Sicht** DisneyFilme kann genauso abgefragt werden wie eine gespeicherte Relation.

- SELECT titel, jahr
  FROM DisneyFilme
  WHERE jahr>=1990;

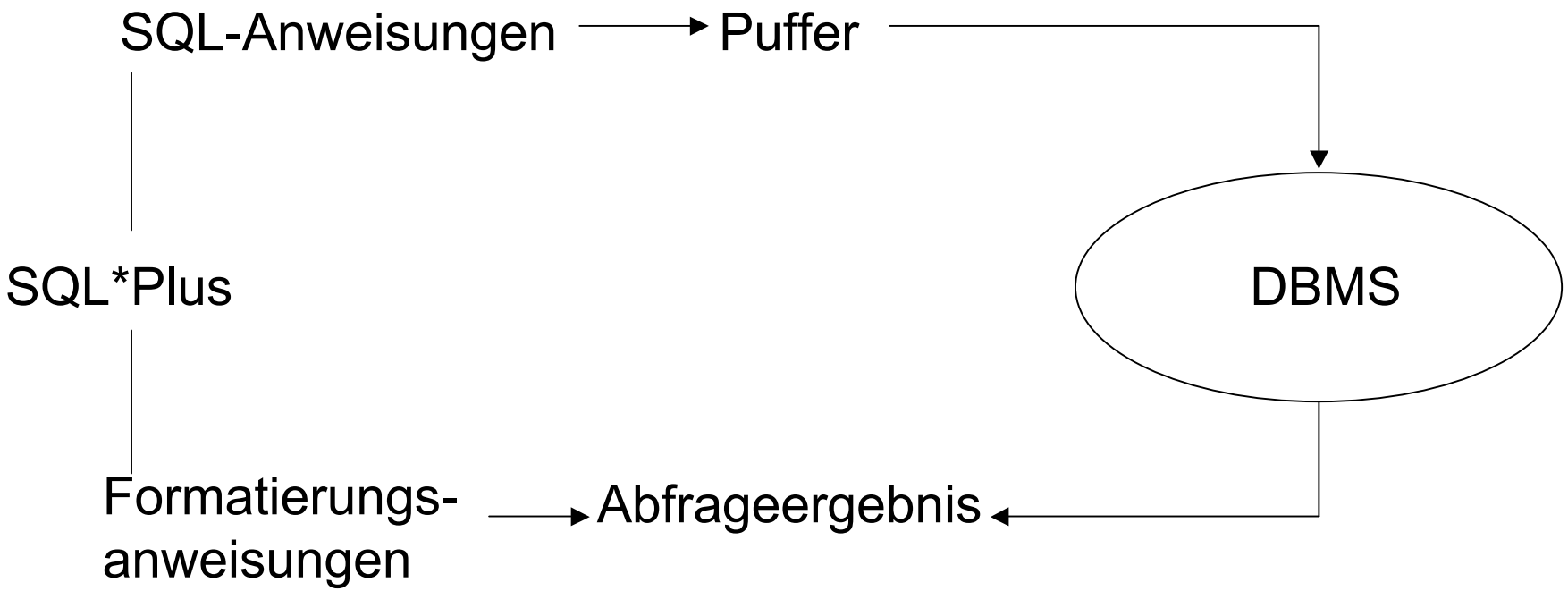


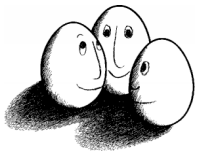
Praktisches

- SQL besteht aus Data Definition Language, Data Manipulation Language und SELECT.
- Zur Programmiersprache fehlen bedingte Anweisungen und Schleifen.
- Die Semantik von SQL ist durch die Relationenalgebra gegeben. Die physikalische und algorithmische Realisierung ist herstellerabhängig.
- Oracle hat eine Umgebung, in der man leicht SQL schreiben kann: SQL*Plus. Skript-Dateien zum Speichern von SQL-Anweisungen für wiederholtes Anwenden.



SQL und SQL*Plus





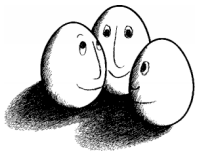
Was wissen wir jetzt?

- Wir haben die Relationenalgebra gesichtet und wissen, dass sie die Semantik von SQL-Anfragen angibt.

- Wir kennen SQL-Anfragen:

  SELECT
  FROM
  WHERE

- Wir kennen Operationen (Vereinigung, Durchschnitt, Differenz, Projektion, kartesisches Produkt, natürlicher Verbund) und Bedingungen.



SQL*Plus

- Sqlpls *Benutzername /Passwort @ Datenbank*
- Zeileneditor, d.h. ein Befehl besteht aus einer Zeile oder – für die Verlängerung auf die nächste Zeile.
- EDIT *Dateiname* ruft den Editor auf
- SAVE *Dateiname* speichert SQL-Puffer in Datei.sql
- GET *Dateiname* holt Datei in Puffer
- START *Dateiname* führt Befehle der Datei aus
- EXIT speichert Änderungen und beendet SQL*Plus.