

LE 7b – IBIS - Datenbanken

Advanced SQL – Verbundabfragen und Subselects etc

Angelehnt an Skript „Datenmodellierung und Datenmanagement“, THM
Mitwirkende Autoren: Prof. Dr. Guckert; Timo Péus, Dr. Thomas Farrenkopf,
Melanie Vanderpuye, Prof. Dr. Grüne (2017)

Prof. Dr. Markus Grüne, FB03, Wirtschaftsinformatik

Inhalt

Verbundabfragen

- INNER JOIN
- EQUI / THETA-JOIN
- OUTER JOIN
- SELF / AUTO JOIN

Lernziel / Fragen

Komplexe Verbundabfragen in der Relationenalgebra und mit SQL

- Wie bringe ich die normalisierten Daten wieder zusammen?
- Welche Eigenschaften haben die verschiedenen Operatoren?

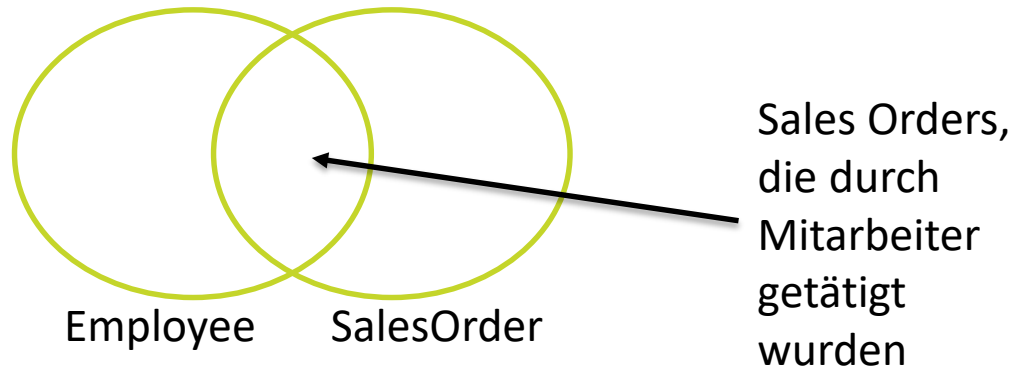
Filterung und Gebrauch von Prädikaten in WHERE-Bedingungen

Predicates and Operators	Description
= < >	Compares values for equality / non-equality.
IN	Determines whether a specified value matches any value in a subquery or a list.
BETWEEN	Specifies an inclusive range to test.
LIKE	Determines whether a specific character string matches a specified pattern, which can include wildcards.
AND	Combines two Boolean expressions and returns TRUE only when both are TRUE.
OR	Combines two Boolean expressions and returns TRUE if either is TRUE.
NOT	Reverses the result of a search condition.

Quelle: Microsoft 2017

Kombiniere Zeilen von mehreren Tabellen nach vorher definierten Vergleichskriterien.

- I.d.R. basierend auf einer Beziehung, die durch Primärschlüssel und Fremdschlüssel definiert ist.
- Beispiel: Suche alle Zeilen, die Daten aus den Tabellen **Employee** und **SalesOrder** kombinieren, wobei **Employee.EmployeeID** (Primärschlüssel) **SalesOrder.EmployeeID** (Fremdschlüssel) entspricht



Verbundabfragen

- ANSI SQL-92
 - Tabellen durch JOIN-Operator im FROM-Clause verbinden

```
SELECT ...  
FROM   Table1 JOIN Table2  
       ON <on_predicate>;
```

- ANSI SQL-89
 - Tabellen durch Kommata im FROM-Clause
 - Nicht empfohlen: fehleranfällig

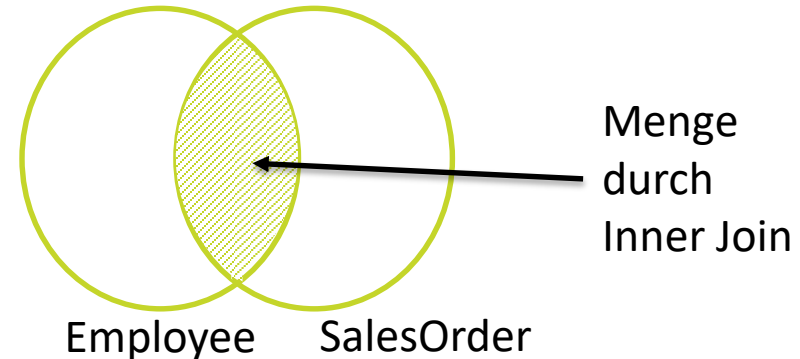
```
SELECT ...  
FROM   Table1, Table2  
WHERE  <where_predicate>;
```

Inner Joins (Standard-JOIN bei MS SQL Server)

Gibt nur Ergebnisse zurück, bei denen ein Match in beiden Eingangstabellen gefunden wird. Die zu matchenden Attribute werden im Prädikat definiert.

Falls das JOIN-Prädikat dem Gleichheitszeichen (=) entspricht, handelt es sich um einen Equi-Join.

```
SELECT emp.FirstName,  
       ord.Amount  
FROM HR.Employee AS emp  
      [INNER] JOIN  
Sales.SalesOrder AS ord  
ON emp.EmployeeID =  
   ord.EmployeeID
```



Natürlicher Verbund $R \bowtie S$

Beim **natürlichen Verbund** werden zwei Tabellen über ein (oder mehrere Attribute) per Gleichheit verknüpft. Vergleichsattribute sind alle Attribute mit gleichem Namen.

Alle Tupel mit gleichen Werten in den Vergleichsattributen werden verknüpft und kommen in die Ergebnismenge. Die Vergleichsattribute kommt nur einmal in die Ergebnisrelation.

Schreibweise/Symbol:

Klasse $\triangleright \triangleleft$ Schueler

	KNr	Klassenlehrer	Bezeichnung	SNr	Nachname	Vorname	GebDatum	SchulEintritt
▶	1	1	Klasse 5	8	Hingst	Ariane	1998-07-25	2008-08-01
	1	1	Klasse 5	12	Garefrekes	Kerstin	1998-09-04	2008-08-01
	1	1	Klasse 5	19	Hildebrand	Timo	1998-04-05	2008-08-01
	1	1	Klasse 5	20	Kahn	Oliver	1998-06-15	2008-08-01
	1	1	Klasse 5	21	Friedrich	Arne	1998-05-29	2008-08-01
	1	1	Klasse 5	26	Owomovela	Patrick	1998-11-05	2008-08-01

Die Join-Attribute müssen in beiden Relationen den gleichen Namen besitzen!

Theta-Join

Die Verbindung kann mit einer Vergleichsoperation Θ durchgeführt werden. Dann sprechen wir von **Theta-Join**.

Schreibweise/Symbol:

$$R \bowtie_{\text{attr_in_r } \Theta \text{ attr_in_s}, \dots} S$$

Alle Klassen, für die ein Lehrer nicht der Klassenlehrer ist!

```
SELECT *  
FROM lehrer l JOIN klasse k  
      ON k.klassenlehrer!=l.pnr
```

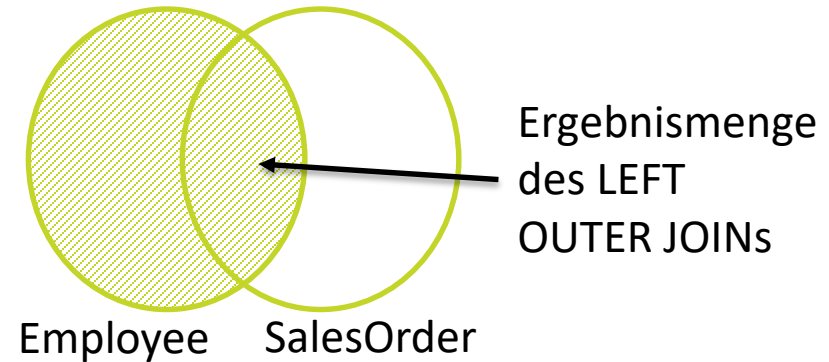
Outer Joins

Liefere alle Zeilen von einer Tabelle und alle dazugehörigen Zeilen der zweiten Tabelle. Liefere darüber hinaus die nicht zugehörigen Zeilen der zweiten Tabelle. Wo keine Ergebnisse in der ersten Tabelle gefunden werden, werden NULL-Werte hinzugefügt.

Varianten:

- LEFT, RIGHT, FULL OUTER JOIN

```
SELECT emp.FirstName,
       ord.Amount
FROM HR.Employee AS emp
LEFT [OUTER] JOIN Sales.SalesOrder AS ord
ON emp.EmployeeID =
   ord.EmployeeID;
```



Outer Join – anders formuliert

- Der innere Join selektiert alle Tupel, für die ein Treffer über die Join-Bedingung erzielt wird.
- Braucht man auch die Nicht-Treffer, so greift man zum Outer Join.
- Der FULL OUTER JOIN bringt alle Treffer sowie die Nicht-Treffer aus der linken und die Nicht-Treffer aus der rechten Tabelle.
- Schreibweise/Symbol:

$R \bowtie_{attr_in_r=attr_in_s, \dots} S$	(Left)
$R \Join_{attr_in_r=attr_in_s, \dots} S$	(Right)
$R \Join_{attr_in_r=attr_in_s, \dots} S$	(Full)

Self Joins

Vergleich von Zeilen in einer Tabelle.

Tabelle kommt 2mal in der FROM-Clause vor.

Beispiel: Gib alle Angestellten mit dem Namen ihres Managers zurück.

```
SELECT emp.FirstName AS  
Employee,
```

```
man.FirstName AS  
Manager  
FROM HR.Employee AS emp  
LEFT JOIN HR.Employee  
AS man  
ON emp.ManagerID =  
man.EmployeeID;
```

Employee		
EmployeeID	FirstName	ManagerID
1	Dan	NULL
2	Aisha	1
3	Rosie	1
4	Naomi	3

Ergebnis	
Employee	Manager
Dan	<i>NULL</i>
Aisha	Dan
Rosie	Dan
Naomi	Rosie

Lernziel / Fragen

Umgang mit JOINS in SQL

Wie bringe ich normalisierte Tabellen wieder in den Zusammenhang?

Inhalt

- Subselects
- Gruppierungen und Spaltenfunktionen

Lernziel / Fragen

Wie kann ich in der Where-Klausel auf andere Tabellen zugreifen?

Wie bilde ich Gruppierungen und Summen?

Weitere komplexe Abfragekonstrukte: Subselect und Gruppierungen

Subselects

- Die Where-Klausel eines Select-Statements kann wiederum Select-Statements enthalten. Diese können:
 - einen Wert zurückliefern, der mit Vergleichsoperator verglichen wird (sog. **skalare Subselects**)
 - eine Menge von Werten erzeugen (Test mit IN bzw. NOT IN)
 - mit **EXISTS** oder **NOT EXISTS** für eine Existenzprüfung dienen
- Gibt es eine Verbindung zum äußeren Select, so spricht man von einem **korrelierten Subselect**. Andernfalls heißt der Subselect **unkorreliert**.

Vorsicht: korrelierte Subselects müssen im Extremfall für jede Zeile der äußeren Treffermenge durchgeführt werden! Das kann lange dauern!

Anwendung von Subselects

- Klasse ohne Klausur

```
SELECT bezeichnung
FROM   klasse k
WHERE  NOT EXISTS
      (SELECT * FROM klausur kl WHERE k.knr=kl.knr)
```

- Unterricht beim Klassenlehrer

```
SELECT *
FROM   unterricht u
WHERE  pnr = (SELECT klassenlehrer
              FROM klasse k
              WHERE u.knr=k.knr)
```

Durchschnitt und Differenz

Beide Operatoren stehen in der Relationenalgebra zur Verfügung.

Allerdings gibt es dafür keine eigenen SQL-Konstrukte. Sie können aber mit Hilfe von Union und Subselects zusammengesetzt werden.

Schreibweise/Symbol:

$R - S$ Differenz

$R \cap S$ Durchschnitt

Die Symbole entsprechen den Symbolen der Mengenlehre.

Beide Operatoren gehören zwar zum SQL-Standard, sind aber in den gängigen Systemen (noch?) nicht implementiert.

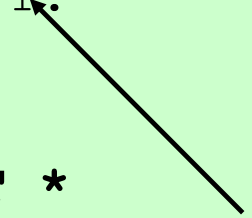
Durchschnitt

Es gibt kein SQL-Statement für den Durchschnitt zweier Tabellen.

Wird hier über eine Konstruktion eines Subselects mit EXISTS oder IN abgebildet.

Hinweis: dies ist nur zur Demonstration! Der Ansatz ist auch so fehlerhaft.

```
SELECT T.KEY, T.*  
FROM T  
WHERE EXISTS  
      (SELECT *  
        FROM U  
        WHERE U.KEY=T.KEY)
```



korreliert

```
SELECT T.KEY, T.*  
FROM T  
WHERE T.Key IN  
      (SELECT U.Key  
        FROM U)
```

unkorreliert

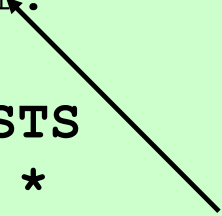
Differenz

Es gibt kein SQL-Statement für die Differenz zweier Tabellen.

Wird über eine Konstruktion eines Subselects mit NOT EXISTS oder IN abgebildet.

korreliert

```
SELECT T.KEY, T.*  
FROM T  
WHERE NOT EXISTS  
      (SELECT *  
        FROM U  
        WHERE U.KEY=T.KEY)
```



unkorreliert

```
SELECT T.KEY, T.*  
FROM T  
WHERE T.Key NOT IN  
      (SELECT U.Key  
        FROM U)
```

Übung

Benutzen Sie die auf den letzten beiden Folien vorgestellten Konstrukte für Durchschnitt und Differenz:

- Welche Vornamen kommen sowohl in der Tabelle Lehrer als auch Schüler vor?
- Welche Vornamen kommen in der Tabelle Schüler und nicht in der Tabelle Lehrer vor?

Übung

Welche Vornamen kommen sowohl in der Tabelle Lehrer als auch Schüler vor?

```
SELECT s.vorname
FROM schueler s
WHERE EXISTS
  (SELECT * FROM lehrer l WHERE s.vorname=l.vorname) ;

SELECT s.vorname
FROM schueler s
WHERE s.vorname IN
  (SELECT l.Vorname FROM lehrer l) ;
```

Übung

Welche Vornamen kommen in der Tabelle Schüler und nicht in der Tabelle Lehrer vor?

```
SELECT s.vorname
FROM schueler s
WHERE NOT EXISTS
  (SELECT * FROM lehrer l WHERE s.vorname=l.vorname) ;
SELECT s.vorname
FROM schueler s
WHERE s.vorname NOT IN
  (SELECT l.Vorname FROM lehrer l) ;
```

Some, Any, All

In der Where-Klausel eines Select-Statements können Werte gegen alle Werte eines Subselects geprüft werden.

Bei einem Vergleichsoperator kann dann festgelegt werden, ob

- alle Bedingungen (**ALL**=alle) oder
- einige die Bedingung erfüllen müssen (**SOME**=**ANY**=einige)

Die jüngste Kollegin

```
select pnr, nachname, gebdatum
from lehrer
where gebdatum >= all (select gebdatum from lehrer)
```


Übung

Suchen Sie alle Klassen, deren Klassenlehrer vor 1950 geboren wurde!

Finden Sie alle Lehrer, die nicht als Klassenlehrer auftreten!

Finden Sie die größte *Snr* innerhalb der Klassen (ohne Anwendung einer Spaltenfunktion)!

Suchen Sie alle Klassen, deren Klassenlehrer vor 1950 geboren wurde!

```
SELECT * FROM klasse k, lehrer l
WHERE k.klassenlehrer=l.pnr
AND YEAR(l.gebdatum)<=1950;
```

Finden Sie alle Lehrer, die nicht als Klassenlehrer auftreten!

```
SELECT * FROM lehrer l
WHERE
l.Pnr <> All (SELECT k.klassenlehrer FROM klasse k);
```

Finden Sie die größte *Snr* innerhalb der Klassen (ohne Anwendung einer Spaltenfunktion)!

```
SELECT * FROM klasse k, schueler s
WHERE k.knr=s.knr AND s.snr >= ALL
(SELECT s.snr FROM schueler s WHERE k.knr=s.Knr);
```

Statistik in der Projektion

Verschiedene Spaltenfunktion erlauben eine Verdichtung der Information in vertikaler Richtung:

sum, max, min, avg

Oder das Zählen der Zeilen:

count(*)

```
SELECT MAX(note)
FROM klausurnote;
```

Die schlechteste Note

```
SELECT COUNT(*)
FROM schueler;
```

Die Anzahl der Schüler

Zählen von Einträgen

Der Count-Operator nimmt auch einzelne Attribute als Argument entgegen. In diesem Fall ist das Ergebnis ebenfalls die Anzahl der Tupel.

Mit dem Zusatz DISTINCT werden die unterschiedlichen Werte in diesem Attribut gezählt.

```
SELECT COUNT(knr)  
FROM schueler;
```

```
SELECT COUNT(DISTINCT knr)  
FROM schueler;
```

Beispiel mit Subselect

Die Statistiken können in Subselects verwendet werden, um die Selektion einschränken zu können.

```
SELECT *  
FROM schueler s  
WHERE s.schuleintritt >  
      (SELECT  
         MIN(date_add(l.schuleintritt,Interval 30 year))  
        FROM lehrer l)
```

Alle Schüler deren Schuleintritt mehr als 30 Jahre nach dem Schuleintritt des dienstältesten Lehrers liegt.

Gruppierung mit Statistik

Häufig interessiert das Ergebnis eines Aggregats für bestimmte Gruppierungen der Tabelle. Dazu dient die **GROUP BY-Klausel**.

Die beste Note pro Klasse

```
SELECT knr, MIN(note)
FROM klausurnote
GROUP BY knr;
```

Durchschnittsnote über alle Klausuren pro Klasse

```
SELECT K.Bezeichnung, AVG(note)
FROM Klasse K JOIN Klausurnote KN ON (K.knr=KN.knr)
GROUP BY K.Bezeichnung;
```

Gruppierung über mehrere Attribute

Gruppierungen gehen auch über mehrere Attribute, die aber alle in der Projektion sein müssen.

```
SELECT knr, fnr, MIN(note)
FROM klausurnote
GROUP BY knr,fnr;
```

```
SELECT k.bezeichnung, f.bezeichnung,
MIN(note)
FROM klausurnote kn
      join fach f on (f.fnr=kn.fnr)
      join klasse k on (kn.knr=k.knr)
GROUP BY k.bezeichnung, f.bezeichnung;
```

- Die **HAVING-Klausel** erlaubt es, die Ergebnismenge einzuschränken.

```
SELECT knr, fnr, MIN(note)
FROM klausurnote
GROUP BY knr, fnr
HAVING MIN(note) < 2
```

Die beste Note pro Klasse und Fach, aber nur wenn sie besser als 2 ist

Beispiel

```
SELECT knr, COUNT (*)  
FROM schueler  
GROUP BY knr
```

Anzahl der Schüler pro Klasse

```
SELECT knr, COUNT (*)  
FROM schueler  
GROUP BY knr  
HAVING COUNT (*) > 5
```

Anzahl der Schüler pro Klasse
für Klassen mit mindestens 6
Schülern

Übung

Erstellen Sie eine Klassenliste mit den Anzahlen der Schüler!

Ermitteln Sie die Durchschnittsnote für alle Klausuren!

Ermitteln Sie nun die Durchschnittsnote für jede Klassen über alle Klausuren, die von den Klassen geschrieben wurden!

Ermitteln Sie nun die schlechteste Note für alle Klassen!

Erstellen Sie eine Liste aller Klassen, deren Gesamtdurchschnittsnote schlechter als 3 ist!

Erstellen Sie eine Klassenliste mit den Anzahlen der Schüler!

```
SELECT k.bezeichnung, s.knr, COUNT(*)  
FROM klasse k, schueler s  
WHERE k.knr=s.knr  
GROUP BY k.bezeichnung, s.knr;
```

Ermitteln Sie die Durchschnittsnote für alle Klausuren!

```
SELECT AVG(kn.note) FROM klausurnote kn;
```

Ermitteln Sie die Durchschnittsnote für jede Klasse über alle Klausuren, die von den Klassen geschrieben wurden!

```
SELECT kn.knr, AVG(kn.note) FROM klausurnote kn  
GROUP BY kn.knr;
```

Übung

Ermitteln Sie nun die schlechteste Note für alle Klassen!

```
SELECT kn.knr, MAX(kn.note) FROM klausurnote kn  
GROUP BY kn.knr;
```

Erstellen Sie eine Liste aller Klassen, deren Gesamtdurchschnittsnote schlechter als 3 ist!

```
SELECT kn.knr, AVG(kn.note) FROM klausurnote kn  
GROUP BY kn.knr HAVING AVG(kn.note)>3;
```

Lernziel / Fragen

Wie kann ich Gruppierungen und Summen realisieren?

Wie kann ich in der Where-Klausel auf andere Tabellen zugreifen?

Weitere komplexe Abfragekonstrukte: Subselect und Gruppierungen