



Übung zur Vorlesung

**Informations- und
Wissensmanagement
(Übung 3)**

Frank Eichinger



Geld für SQL-Anfragen?

- Aufruf zur Teilnahme an einem Strategiespiel
- Weitere Informationen:
 - <http://www.ipd.uka.de/~eichi/teaching/iwm0708/ServiceGame.pdf>
- Bei Fragen:
 - Christian von der Weth
 - weth@ira.uka.de





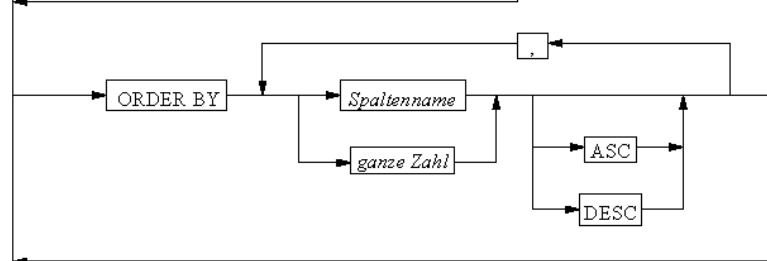
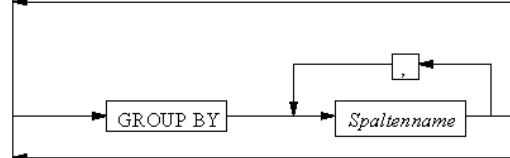
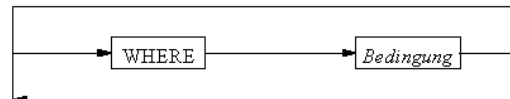
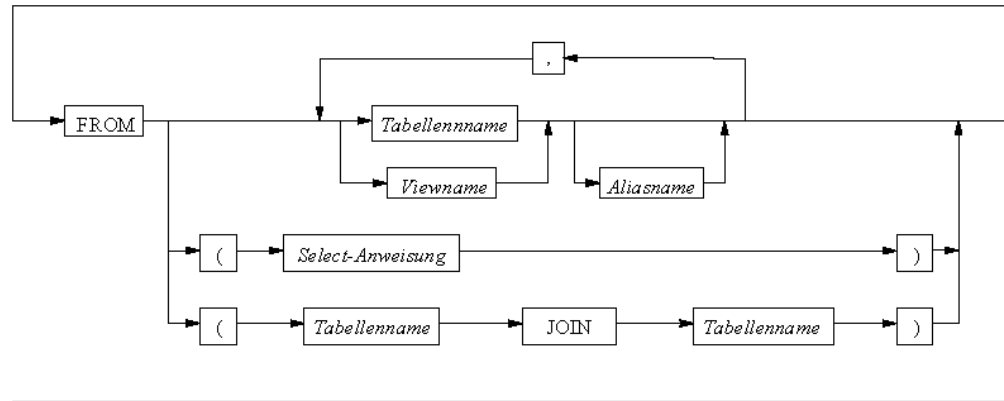
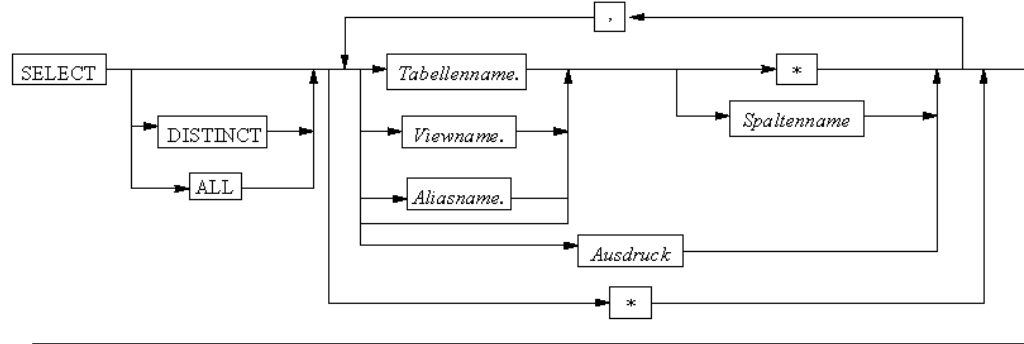
SQL-Anfragen

- Grundgerüst einer SQL-Anfrage

```
SELECT <attribute-list>  
FROM <table-name>, ...  
WHERE <predicate-list>  
GROUP BY <attribute-list>  
HAVING <predicate-list>  
ORDER BY <attribute-list>;
```

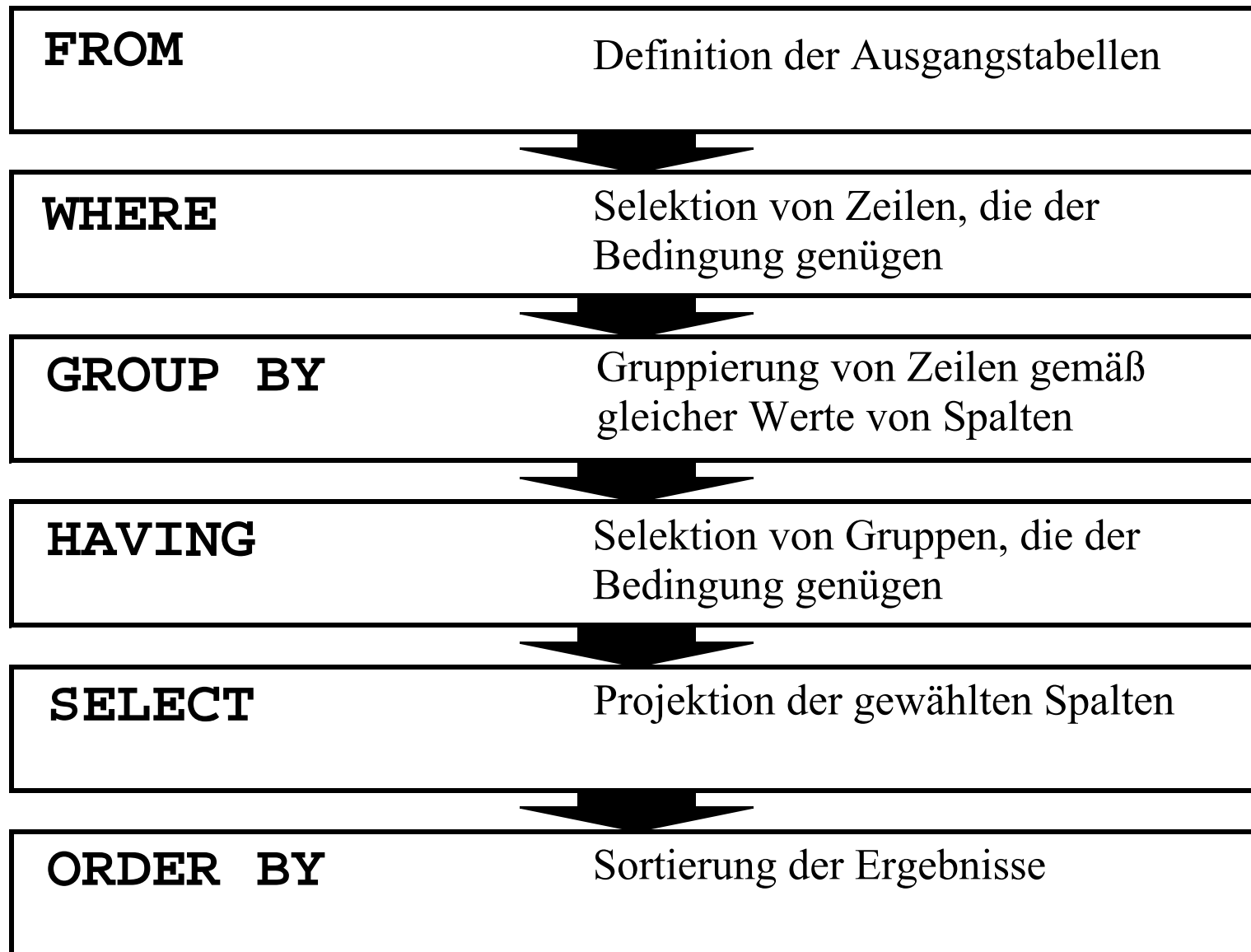


SELECT-Syntax



Graphik übernommen von
www.ifis.cs.tu-bs.de/html_d/skripte/handbuch.2.ps

Formulierung von Anfragen





Joins

- Bei N angegebenen Tabellen in der **FROM**-Klausel müssen im **WHERE**-Teil mindestens (N-1) Join-Bedingungen stehen.
- Anzahl der Join-Bedingungen erhöht sich bei zusammengesetzten Schlüsseln.





Gruppierung

- **HAVING** nie ohne **GROUP BY**!
- **GROUP BY**-Anfragen umfassen typischerweise auch immer Aggregate.
- Primärschlüssel als Gruppierungsattribut falsch bzw. sinnlos.
- Ein **GROUP BY** schränkt die Formulierung der **SELECT**-Klausel ein, auftreten dürfen:
 - Gruppierungsattribute
 - Aggregate über Nicht-Gruppierungsattribute

```
SELECT Stadt, Einwohner  
FROM Stadt  
GROUP BY Stadt;
```



Mengenoperationen

- Ergebnis der Teilanfragen müssen ein „vergleichbares“ Relationenschema besitzen:
 - Anzahl der Attribute
 - Grund-Datentypen der Attribute
 - Reihenfolge der Attribute (Aber: unwichtig wenn alle Attribute vom selben Datentyp)
 - Nicht: Namen der Attribute
- Beispiel:
 - *Wie lauten die Vor- und Nachnamen aller Professoren und Studenten?*
 - **SELECT name, vorname FROM student
UNION
SELECT name, vorname FROM professor;**





Verschachtelte Anfragen (1)

- (NOT) IN

```
SELECT * FROM Stadt
WHERE Stadt IN (SELECT Stadt FROM Hotel);
```
- (NOT) EXISTS

```
SELECT * FROM Stadt
WHERE EXISTS (SELECT * FROM Hotel
              WHERE Stadt.Stadt = Hotel.Stadt);
```
- Vergleichsoperator und Quantor (ANY/SOME, ALL)

```
SELECT * FROM Stadt
WHERE Stadt = ANY (SELECT Stadt FROM Hotel);
```
- Alle drei Anfragen werden in der Datenbank wie folgt als Equi-Join abgearbeitet:

```
SELECT Stadt.* FROM Stadt, Hotel
WHERE Stadt.Stadt = Hotel.Stadt;
```



Verschachtelte Anfragen (2)

- Relationenschema der Unteranfrage (außer **EXISTS**) muss mit den 'äußeren' Attributen übereinstimmen:
 - Anzahl der Attribute
 - Datentypen der Attribute
 - Reihenfolge der Attribute
- Beispiel:
 - **...WHERE telNr IN**
(SELECT phone FROM company);





Verschachtelte Anfragen (3)

- Nur wenn die Unteranfrage garantiert ein einzeliges Ergebnis liefert (außer **EXISTS**), darf ein „**=**“ statt einem „**IN**“ verwendet werden.
 - Aggregatwerte
 - Punktanfragen auf Primärschlüssel/UNIQUE-Attribute
- Beispiel (falsch):
 - **...WHERE ID = (SELECT ID FROM Personen);**
- Beispiel (richtig):
 - **...WHERE ID IN (SELECT ID FROM Personen);**
- Beispiel (richtig wenn Name UNIQUE):
 - **...WHERE ID = (SELECT ID FROM Pers
WHERE Name = 'Erik Buchmann');**
- Beispiel (auch richtig):
 - **...WHERE Gehalt =
(SELECT AVG(gehalt) FROM Personen);**



Verschachtelte Anfragen (4)

- Verschachtelung über mehrere Stufen möglich!
- Unteranfragen in der **FROM**-Zeile möglich.
 - Z.B. schreibt man zunächst eine komplizierte innere Anfrage und macht Gruppierungen, weitere **WHERE**-Klauseln oder Joins in einer äußeren Anfrage:

```
– SELECT      t2.attr_a, count(*)
FROM          tabelle_1 as t1,
              (SEHR KOMPL. ANFRAGE) as t2
WHERE         t1.attr_b = t2.attr_b
GROUP BY     t2.attr_a
HAVING       sum(t2.attr_c) > 50;
```
- Der Anfrageoptimierer macht automatisch aus der verschachtelten eine „flache“ Anfrage.



Verschachtelte Anfragen (5)

- Unkorrelierte Unteranfragen

```
SELECT Name FROM Stadt
WHERE Name IN (SELECT Name FROM Land);
```

- Korrelierte Unteranfragen

```
SELECT s.Name FROM Stadt s
WHERE EXISTS (SELECT * FROM Land l
              WHERE s.Name=l.Name);
```

- Unkorrelierte **EXISTS**-Unteranfragen sind falsch oder sinnlos.

Behandlung von Nullwerten

- **SELECT * FROM ANMELDUNG;**
- **SELECT * FROM ANMELDUNG
WHERE personen > 0;**
- **SELECT * FROM ANMELDUNG
WHERE personen <= 0;**
- **SELECT * FROM ANMELDUNG
WHERE personen <= 0
OR
personen IS NULL;**

NAME	PERSONEN
Lockemann	
Mülle	2
Böhm	1
Buchmann	0

NAME	PERSONEN
Mülle	2
Böhm	1

NAME	PERSONEN
Buchmann	0

NAME	PERSONEN
Lockemann	
Buchmann	0



Aufgabe 1 - Datenbankschema

- Stadt(stadt, land, einwohnerzahl)
- Entfernung(stadt1, land1, stadt2, stadt2,
entfernung)
- Hotel(hotel, stadt, land, klasse, adresse,
anzahlEZimmer, anzahlDZimmer,
preisEZimmer, preisDZimmer)
- Buchung(buchungNr, hotel, stadt, land,
anreiseDatum, abreiseDatum, kundenNr,
gebuchteEZimmer, gebuchteDZimmer)
- Kunde(kundenNr, name, vorname, adresse)



Aufgabe 1a+b)

- *Bestimmen Sie die Namen und Adressen aller Hotels in Österreich.*
- ```
SELECT hotel, adresse
FROM hotel h
WHERE h.land = 'Österreich';
```
- *Bestimmen Sie die Anzahl aller Kunden.*
- ```
SELECT count(*)
FROM   Kunde;
```



Aufgabe 1c)

- *Bestimmen Sie Name und Vornamen aller Kunden, die ein Hotel im Dezember gebucht haben.*
- ```
SELECT DISTINCT name, vorname
FROM Kunde k
WHERE k.kundenNr in (
 SELECT kundenNr
 FROM Buchung b
 WHERE b.anreiseDatum >= '2007-12-01'
 AND b.abreiseDatum <= '2007-12-31'
);
```



# Aufgabe 1d)

- *Bestimmen Sie das Hotel in Spanien mit der größten Bettenkapazität.*
- ```
SELECT h.hotel, h.stadt, h.land
FROM   hotel h
WHERE  h.land = 'Spanien'
AND    (h.anzahlEZimmer+2*h.anzahlDZimmer)=(
        SELECT max(h2.anzahlEZimmer +
                    2*h2.anzahlDZimmer)
        FROM   hotel h2
        WHERE  h2.land = 'Spanien'
        );
```



Aufgabe 1e)

- *Bestimmen Sie alle Hotels in deutschen Städten mit mehr als 500.000 Einwohnern.*
- ```
SELECT h.hotel
FROM Hotel h
WHERE (h.stadt, h.land) IN (
 SELECT s.stadt, s.land
 FROM Stadt s
 WHERE s.einwohnerzahl > 500000
 AND s.land = 'Deutschland'
);
```
- *Wie wird diese Anfrage agearbeitet?*



# Aufgabe 1f)

- *Bestimmen Sie die durchschnittliche Einwohnerzahl der Städte aller Länder, sortiert nach der durchschnittlichen Einwohnerzahl.*
- ```
SELECT    s.land, AVG(einwohnerzahl)
          AS DurchschnittlicheEinwohnerzahl
FROM      Stadt s
GROUP BY s.land
ORDER BY DurchschnittlicheEinwohnerzahl;
```



Aufgabe 1g)

- *Geben Sie Tage und Hotel aus, an denen in diesem Hotel mehr Einzel- oder Doppelzimmer gebucht wurden als verfügbar sind.*
- *Wenn es sich um mehr als einen Tag in Folge handelt, reicht es, den jeweils ersten Tag eines solchen Zeitraums auszugeben.*
- *Verwendung als Integritätsbedingung mit NOT EXISTS!*



Aufgabe 1g)

```
SELECT x.*
FROM hotel h,
(select d.datum, b.HOTEL, b.STADT, b.LAND,
sum(b.GEBUCHTEEZIMMER) ez,
sum(b.GEBUCHTEDZIMMER) dz
from buchung b, (select distinct ANREISEDATUM
as datum from buchung) d
where d.DATUM between b.ANREISEDATUM and
b.ABREISEDATUM
group by d.datum, b.HOTEL, b.STADT, b.LAND) x
WHERE x.HOTEL=h.HOTEL AND x.STADT=h.STADT AND
x.LAND=h.LAND
AND (x.ez>h.ANZAHLEZIMMER OR
x.dz>h.ANZAHLDZIMMER)
```





Aufgabe 1h)

- *Bestimmen Sie alle Städte die mehr als 10 Hotels besitzen und geben Sie auch die konkrete Anzahl aus.*
- ```
SELECT h.stadt, h.land, count(*)
FROM Hotel h
GROUP BY h.stadt, h.land
HAVING COUNT(*) > 10;
```



# Aufgabe 1i)

- *Bestimmen Sie die Namen aller Hotels in Paris, bei denen ein Einzelzimmer mindestens 10% weniger als der Durchschnitt aller Hotels in Paris kostet.*
- ```
SELECT h.hotel
FROM   Hotel h
WHERE  h.stadt = 'Paris'
AND    h.land = 'Frankreich'
AND    h.preisEZimmer <= 0.9 * (
    SELECT avg(preisEZimmer)
    FROM   Hotel k
    WHERE  k.stadt = 'Paris'
    AND    k.land = 'Frankreich'
);
```



Aufgabe 1j)

- *Bestimmen Sie alle Hotels in italienischen Städten mit mindestens 100.000 Einwohnern für die keine Buchung vorliegt. Geben Sie Name, Adresse und Klasse dieser Hotels aus, sortiert nach Name.*

```
• SELECT    h.hotel, h.adresse, h.klasse
FROM        Hotel h
WHERE       h.land = 'Italien'
AND         h.stadt IN (
            SELECT    s.stadt
            FROM        Stadt s
            WHERE       s.einwohnerzahl > 100000
            AND         (h.hotel, h.stadt, h.land) NOT IN (
                SELECT    b.hotel, b.stadt, b.land
                FROM        Buchung b ))
ORDER BY   h.hotel;
```



Aufgabe 1k)

- *Bestimmen Sie die Anzahl der Hotels im Umkreis von 1.000 km um Karlsruhe oder New York.*

```
• SELECT count(*)
  FROM Hotel h
 WHERE (h.stadt, h.land) IN (
     SELECT stadt1, land1
     FROM Entfernung
     WHERE entfernung < 1000
     AND ((stadt2 = 'Karlsruhe'
          and land2 = 'Deutschland') OR
          (stadt2 = 'New York' and land2 = 'USA'))

     UNION

     SELECT stadt2, land2
     FROM Entfernung
     WHERE entfernung < 1000
     AND ((stadt1 = 'Karlsruhe'
          and land1 = 'Deutschland') OR
          (stadt1 = 'New York' and land1 = 'USA'))
  );
```



Aufgabe 1l)

- *Geben Sie für jedes Land die Anzahl aller Paare von Städten an, die die gleiche Größe haben.*
- **SELECT** land, count(*)/2 as anz_paare
FROM (
 SELECT DISTINCT s1.stadt,
 s2.stadt, s1.land AS land
FROM stadt s1, stadt s2
WHERE s1.land = s2.land
AND s1.einwohnerzahl =
 s2.einwohnerzahl

) AS tmp
GROUP BY land





Heuristiken für Optimierung

- Auflösung von verschachtelten Anfragen
- Eliminierung von mehrfachen Projektionen
- Verschieben von Selektionen soweit wie möglich nach unten im Operatorbaum
- Zusammenfassen von Selektionen und Kreuzprodukten zu Joins
- Bestimmung der Reihenfolge der Joins in der Form, dass möglichst kleine Zwischenergebnisse entstehen
- Verschieben von Projektionen soweit wie möglich nach unten im Operatorbaum





Aufgabe 2a-c)

a) $\pi_{\text{Titel}} (\pi_{\text{Titel, Länge}} (\text{Lied}))$

$$\pi_{\text{Titel}} (\text{Lied})$$

b) $\sigma_{\text{Titel} = \text{'Jump'}} (\pi_{\text{Titel}} (\text{Lied}))$

$$\pi_{\text{Titel}} (\sigma_{\text{Titel} = \text{'Jump'}} (\text{Lied}))$$

c) $\sigma_{\text{Produktion.sfirma.Nam e} = \text{VideoClip, Produktion Name}} (\text{Filmproduktion} \times \text{VideoClip})$

$$\text{Filmproduktion} \bowtie \text{VideoClip}$$



Aufgabe 2d-f)

d) $\sigma_{ID=3312}(\sigma_{Format='PAL'}(VideoClip))$

$$\sigma_{Format='PAL'}(\sigma_{ID=3312}(VideoClip))$$

e) $\sigma_{Enthält.TrackNr=5}(Lied \bowtie Enthält)$

$$Lied \bowtie (\sigma_{TrackNr=5}(Enthält))$$

f) $\sigma_{Filmproduktion.Name='EMI' \wedge VideoClip.Format='PAL'}(VideoClip \bowtie Filmproduktion)$

$$((\sigma_{Name='EMI'}(Filmproduktion)) \bowtie (\sigma_{Format='PAL'}(VideoClip)))$$

Aufgabe 2g-i)

g) $\sigma_{\text{Lied.Titel} = \text{Enthält.Lied.Titel} \wedge \text{CD.Titel} = \text{Enthält.CD.Titel} \wedge \text{Enthält.Lied.Künstler} = \text{Lied.Künstler} \wedge \text{Lied.Künstler} = \text{'ABBA'}}$
(Enthält \times CD \times Lied)

$$\text{CD} \bowtie \text{Enthält} \bowtie (\sigma_{\text{Künstler} = \text{'ABBA'}}(\text{Lied}))$$

h) $\sigma_{\text{Länge} > 0}(\pi_{\text{Titel, Länge}}(\text{Lied}))$

keine Verbesserung

i) $\pi_{\text{VideoClip.Kosten}}(\text{VideoClip} \bowtie \text{Filmproduktion})$

keine Änderung (Verschiebung der Projektion nicht zulässig)



Aufgabe 2j+k)

j) $\pi_{\text{VideoClip. Produktion Name}}(\text{VideoClip} \bowtie \text{Filmproduktion})$

keine Änderung (Verschiebung der Projektion nicht zulässig)

k) $\sigma_{\text{TrackNr}=5}(\text{Lied} \bowtie (\sigma_{\text{CDTitel}='Best Of'}(\text{Enthält})))$

$\text{Lied} \bowtie (\sigma_{\text{TrackNr}=5}(\sigma_{\text{CDTitel}='Best Of'}(\text{Enthält})))$

