



**Folien basierend auf  
Thorsten Papenbrock**

- Übung Datenbanksysteme I  
SQL

G-3.1.14, Campus III / F-E.06 Campus II

Leon Bornemann  
Hasso Plattner Institut

- JDBC Übung
  - Morgen vermutlich normale Vorlesung!
- Nächstes Übungsblatt
  - Vollständig Lesen!
  - Nicht die komplette Ergebnismenge hinschreiben!
  - Status Installation und Datenimport?
  - Aufgaben getrennt hochladen!

**DBSI - Übung**

SQL

Leon Bornemann  
Chart **2**

# Merklblatt SQL

## Grundbausteine:

**SELECT** <Attributliste> (= Schema der Ergebnisrelation; \* für alle Attribute)  
**FROM** <Relationenliste> (Relationen aus denen die **Tupel** stammen)  
**WHERE** <Bedingungen>; (Bedingungen an die Daten; Verknüpfung über Schlüsselwort **AND**)

### SELECT:

- Entspricht Projektion  $\pi$  der relationalen Algebra
- Umbenennung: `SELECT Titel AS Filmtitel`
- Arithmetik: `SELECT Länge * 3.1415 AS LängeMalPi`
- Konstanten: `SELECT ‚Herr‘ AS Titel`
- **Duplikateliminierung**  $\delta$ : `SELECT DISTINCT Titel`

### FROM:

- Entspricht Kreuzprodukt  $\times$  der relationalen Algebra (falls mehrere Relationen gewählt)

### WHERE:

- Entspricht Selektion  $\sigma$  der relationalen Algebra
- WHERE-Teil der Anfrage ist optional
- **Operatoren**: =, <, >, <=, >=, LIKE, NOT, ANY, ALL, EXISTS, IN, ...
- Kann Kreuzprodukt des FROM-Teils zum **Join** machen:
  - WHERE Person.ID = Mensch.ID (Natürlicher **Join**  $\bowtie$ )
  - WHERE Person.Name = Mensch.Vorname (Theta **Join**  $\bowtie_{\theta}$ )

## Komplexe Anfragen:

**SELECT** <Attributliste>  
**FROM** <Relationenliste>  
**WHERE** <Bedingungen>  
**GROUP BY** <Gruppierungsattribute>  
**HAVING** <Bedingungen auf Gruppierungsattribute>  
**ORDER BY** <Attributliste>;

### GROUP BY ... HAVING:

- Entspricht Gruppierung  $\gamma$  der relationalen Algebra
- **Aggregationsoperatoren** für SELECT-Statement:
  - AVG(<Attribut>)
  - COUNT(<Attribut>)
  - SUM(<Attribut>)
- HAVING entspricht einer Selektion nach der Gruppierung

### ORDER BY:

- Entspricht Sortierung  $\tau$  der relationalen Algebra
- Sortiert das Ergebnis der Anfrage entsprechend der Attributliste
- Aufsteigend: ORDER BY Vorname, Nachname ASC
- Absteigend: ORDER BY Vorname, Nachname DESC

## Datendefinition: Data Definition Language (DDL)

**CREATE TABLE** <Tabellenname>(<Attributliste mit Datentypen>;

- Aufgabe: neue Tabelle erstellen
- Datentypen mit Länge n bzw. m: CHAR(n), VARCHAR(n), BIT(n), DECIMAL(n,m)
- Datentypen mit impliziter Länge: INT, FLOAT
- Datentypen für Objekte: CLOB, BLOB
- Datentypen für Zeiten: TIME, DATE, TIMESTAMP
- Bsp.: `CREATE TABLE Schauspieler (`
  - Name CHAR(30),
  - Adresse VARCHAR(255),
  - Geschlecht CHAR(1),
  - Geburtsdag DATE);
- Nebenbedingungen:
  - Primärschlüssel: PRIMARY KEY
  - Eindeutigkeit: UNIQUE
  - Default-Werte: DEFAULT <Defaultwert>
  - Nicht-Null: NOT NULL
  - Fremdschlüssel: FOREIGN KEY (<Attributliste>) REFERENCES <Tabellenname>(<Attributliste>)
  - Weitere: CHECK, CREATE TRIGGER, CREATE ASSERTION, ...

Bsp.: `CREATE TABLE Schauspieler (`

- SchauspielerNummer INT PRIMARY KEY,
- Name CHAR(30) NOT NULL,
- Adresse VARCHAR(255) NOT NULL UNIQUE,
- Geschlecht CHAR(1),
- Geburtsdag DATE DEFAULT DATE ‚0000-00-00‘,
- FOREIGN KEY (Adresse) REFERENCES Haus(Adresse));

**DROP TABLE** <Tabellenname>;

- Aufgabe: bestehende Tabelle löschen
- Bsp.: `DROP TABLE Schauspieler;`

**ALTER TABLE** <Tabellenname> <Aktion>;

- Aufgabe: bestehende Tabelle ändern
- ADD: - Attribut hinzufügen
  - Bsp.: `ALTER TABLE Schauspieler ADD Telefon CHAR(16)`
- DROP: - Attribut löschen
  - Bsp.: `ALTER TABLE Schauspieler DROP Geburtsdag`

**CREATE INDEX** <Indexname> **ON** <Tabellenname> (<Attributliste des neuen Index>;

- Aufgabe: Index erstellen

**DROP INDEX** <Indexname>;

- Aufgabe: Index löschen

## Datenbearbeitung: Data Modelling Language (DML)

**INSERT INTO** <Tabellenname>(<Attributliste>) **VALUES** (<Attributliste>;

- Aufgabe: **Tupel** einfügen
- Bsp.: `INSERT INTO Studio(Name, Nummer) VALUES (‚Pixa‘, 34);`
- **Ergebnis einer Anfrage für Einfügen nutzen**:
  - Bsp.: `INSERT INTO Studios(Name)`
    - `SELECT DISTINCT StudioName`
    - FROM Film
    - WHERE StudioName NOT IN
    - (SELECT Name
    - FROM Studios);

**DELETE FROM** <Tabellenname> **WHERE** <Bedingung>;

- Aufgabe: **Tupel** löschen
- Bsp.: `DELETE FROM Studio WHERE Name=‚Pixa‘;`

**UPDATE** <Tabellenname> **SET** <Zuweisung> **WHERE** <Bedingung>;

- Aufgabe: Attributwerte ändern
- Bsp.: `UPDATE Studio SET Name=‚Pixa‘ WHERE Name=‚Pi‘;`

**Bulk insert**: IMPORT, LOAD, ... (→ DBMS spezifisch)

## Mengenoperationen

<Anfrage> **UNION** (<Anfrage>) (Liefert Vereinigung „ $\cup$ “ der beiden Ergebnismengen)

<Anfrage> **EXCEPT** (<Anfrage>) (Liefert Differenz „ $-$ “ der beiden Ergebnismengen)

<Anfrage> **INTERSECT** (<Anfrage>) (Liefert Schnittmenge „ $\cap$ “ der beiden Ergebnismengen)

- UNION, EXCEPT und INTERSECT nutzen Mengensemantik (→ eliminieren Duplikate)
- UNION ALL, EXCEPT ALL und INTERSECT ALL nutzen Multimenssemantik (→ erhalten Duplikate)

## Join-Varianten

1.) **Kreuzprodukt mit Bedingung**:

```
SELECT *
FROM <Join-Relation1>, <Join-Relation2>
WHERE <Join-Attribut1> = <Join-Attribut2>;
```

2.) **Schlüsselwort**:

```
<Tabellenname> CROSS JOIN <Tabellenname>
<Tabellenname> NATURAL JOIN <Tabellenname>
<Tabellenname> NATURAL INNER JOIN <Tabellenname>
<Tabellenname> NATURAL LEFT OUTER JOIN <Tabellenname>
<Tabellenname> NATURAL RIGHT OUTER JOIN <Tabellenname>
<Tabellenname> NATURAL FULL OUTER JOIN <Tabellenname>
```

## Sichten

**CREATE VIEW** <Sichtname> **AS** <Anfrage>;

- Aufgabe: Erstelle eine Sicht für die gegebene SQL-Anfrage

<b>SELECT</b>	<Attributliste>
<b>FROM</b>	<Relationenliste>
<b>WHERE</b>	<Bedingungen>
<b>GROUP BY</b>	<Gruppierungsattribute>
<b>HAVING</b>	<Bedingungen auf Gruppierungsattribute>
<b>ORDER BY</b>	<Attributliste>;

Weitere Schlüsselwörter:

**DISTINCT, AS, JOIN**

**AND, OR**

**MIN, MAX, AVG, SUM, COUNT**

**NOT, IN, LIKE, ANY, ALL, EXISTS**

**UNION, EXCEPT, INTERSECT**

...

Übersicht:

# Relationale Übungsschemata

Modellnummer

Prozessorgeschwindigkeit [MHz]

**Product(maker, model, type)**

- Beispieletupel: ('Lenovo', 1005, 'pc')

Festplattengröße [GB]

**PC(model, speed, ram, hd, rd, price)**

- Beispieletupel: (1005, 1000, 128, 20, '12xDVD', 1499)

Geschwindigkeit und Typ des Laufwerks

**Laptop(model, speed, ram, hd, screen, price)**

- Beispieletupel: (2008, 650, 64, 10, 12.1, 1249)

Bildschirm-  
auflösung

**Printer(model, color, type, price)**

- Beispieletupel: (3005, 'bubble', 99)

Typ (laser, ink-jet, bubble)

Preis [Euro]

DBSI - Übung  
SQL

Leon Bornemann  
Chart 5

## Product(**maker**, model, type)

- Beispeltupel: ('Lenovo', 1005, 'pc')

## PC(model, speed, ram, hd, rd, price)

- Beispeltupel: (1005, 1000, 128, 20, '12xDVD', 1499)

## Laptop(model, speed, ram, hd, screen, price)

- Beispeltupel: (2008, 650, 64, 10, 12.1, 1249)

## Printer(model, color, type, price)

- Beispeltupel: (3005, true, 'bubble', 200)

Anfrage: „Welche PC-Modelle haben eine Geschwindigkeit von mindestens 1000 MHz?“

$$\pi_{\text{model}}(\sigma_{\text{speed} \geq 1000}(\text{PC}))$$

Lösung:  
Deutsch → SQL

```
SELECT model  
FROM PC  
WHERE speed >= 1000;
```

Beispieltapel: (3000, 1700, 10000, 200)

Anfrage: „Welche PC-Modelle haben eine Geschwindigkeit von mindestens 1000 MHz?“

$\pi_{\text{model}}(\sigma_{\text{speed} \geq 1000}(\text{PC}))$

## Product(**maker**, model, type)

- Beispieltupel: ('Lenovo', 1005, 'pc')

## PC(model, speed, ram, hd, rd, price)

- Beispieltupel: (1005, 1000, 128, 20, '12xDVD', 1499)

## Laptop(model, speed, ram, hd, screen, price)

- Beispieltupel: (2008, 650, 64, 10, 12.1, 1249)

## Printer(model, color, type, price)

- Beispieltupel: (3005, true, 'bubble', 200)

Anfrage: „Welche Hersteller bauen Laptops mit einer Harddisk von mindestens 10 GB Größe?“

$$\pi_{\text{maker}}(\sigma_{\text{hd} \geq 10}(\text{Product} \bowtie \text{Laptop}))$$



- a) **SELECT** maker  
**FROM** Product, Laptop  
**WHERE** Product.model = Laptop.model  
**AND** hd >= 10;
- b) **SELECT** maker  
**FROM** Product **NATURAL JOIN** Laptop  
**WHERE** hd >= 10;
- c) **SELECT** maker  
**FROM** Product **JOIN** Laptop **ON** Product.model = Laptop.model  
**WHERE** hd >= 10;

## Product(**maker**, model, type)

- Beispeltupel: ('Lenovo', 1005, 'pc')

## PC(model, speed, ram, hd, rd, price)

- Beispeltupel: (1005, 1000, 128, 20, '12xDVD', 1499)

## Laptop(model, speed, ram, hd, screen, price)

- Beispeltupel: (2008, 650, 64, 10, 12.1, 1249)

## Printer(model, color, type, price)

- Beispeltupel: (3005, true, 'bubble', 200)

Anfrage: „Finde die Modellnummern aller Farblaserdrucker.“

$\pi_{\text{model}}(\sigma_{\text{color}=\text{true} \wedge \text{type}=\text{'laser'}}(\text{Printer}))$

Lösung:  
Deutsch → SQL

```
SELECT model  
FROM Printer  
WHERE color=true  
AND type='laser';
```

Beispielwerte: (3000, true, 'laser', 200)

Anfrage: „Finde die Modellnummern aller Farblaserdrucker.“

$\pi_{\text{model}}(\sigma_{\text{color=true} \wedge \text{type='laser'}}(\text{Printer}))$

## Product(**maker**, model, type)

- Beispieltupel: ('Lenovo', 1005, 'pc')

## PC(model, speed, ram, hd, rd, price)

- Beispieltupel: (1005, 1000, 128, 20, '12xDVD', 1499)

## Laptop(model, speed, ram, hd, screen, price)

- Beispieltupel: (2008, 650, 64, 10, 12.1, 1249)

## Printer(model, color, type, price)

- Beispieltupel: (3005, true, 'bubble', 200)

*Anfrage:* „Finde die Modellnummer und den Preis aller Produkte (jeden Typs), die von Hersteller 'Apple' gebaut werden.“

```

$$\pi_{\text{model,price}}(\sigma_{\text{maker}='Apple'}(\text{Product} \bowtie (\pi_{\text{model,price}}(\text{PC}) \cup \pi_{\text{model,price}}(\text{Laptop}) \cup \pi_{\text{model,price}}(\text{Printer}))))$$

```

# Lösung: Deutsch → SQL

```
(SELECT Product.model, price
FROM Product JOIN PC ON Product.model = PC.model
WHERE maker = 'Apple')
UNION
(SELECT Product.model, price
FROM Product JOIN Laptop ON Product.model = Laptop.model
WHERE maker = 'Apple')
UNION
(SELECT Product.model, price
FROM Product JOIN Printer ON Product.model = Printer.model
WHERE maker = 'Apple');
```

Beispielantwort: (0000, 'Mac', '00000', 200)

Anfrage: „Finde die Modellnummer und den Preis aller Produkte (jeden Typs), die von Hersteller 'Apple' gebaut werden.“

$$\pi_{\text{model,price}}(\sigma_{\text{maker}='Apple'}(\text{Product} \bowtie (\pi_{\text{model,price}}(\text{PC}) \cup \pi_{\text{model,price}}(\text{Laptop}) \cup \pi_{\text{model,price}}(\text{Printer}))))$$

Lösung:

# Deutsch → SQL

```
WITH Price AS (  
  (SELECT model, price FROM PC) UNION  
  (SELECT model, price FROM Laptop) UNION  
  (SELECT model, price FROM Printer))  
SELECT Price.*  
FROM Product NATURAL JOIN Price  
WHERE maker = 'Apple';
```

Beispiel: (0000, 'Mac', 'iMac', 200)

Anfrage: „Finde die Modellnummer und den Preis aller Produkte (jeden Typs), die von Hersteller 'Apple' gebaut werden.“

$$\pi_{\text{model,price}}(\sigma_{\text{maker}='Apple'}(\text{Product} \bowtie (\pi_{\text{model,price}}(\text{PC}) \cup \pi_{\text{model,price}}(\text{Laptop}) \cup \pi_{\text{model,price}}(\text{Printer}))))$$

DBSI - Übung  
SQL

Leon Bornemann  
Chart 14

## Product(**maker**, model, type)

- Beispeltupel: ('Lenovo', 1005, 'pc')

## PC(model, speed, ram, hd, rd, price)

- Beispeltupel: (1005, 1000, 128, 20, '12xDVD', 1499)

## Laptop(model, speed, ram, hd, screen, price)

- Beispeltupel: (2008, 650, 64, 10, 12.1, 1249)

## Printer(model, color, type, price)

- Beispeltupel: (3005, true, 'bubble', 200)

Anfrage: „Finde alle Hersteller, die Laptops, aber keine PCs herstellen.“

DBSI - Übung  
SQL

Leon Bornemann  
Chart 15

$$\delta(\pi_{\text{maker}}(\text{Product} \bowtie \text{Laptop})) - \delta(\pi_{\text{maker}}(\text{Product} \bowtie \text{PC}))$$

Lösung:  
Deutsch → SQL

```
(SELECT DISTINCT maker  
FROM Product, Laptop  
WHERE Product.model = Laptop.model)  
EXCEPT  
(SELECT DISTINCT maker  
FROM Product, PC  
WHERE Product.model = PC.model);
```

**DISTINCTs** sind  
nicht nötig wegen  
Mengensemantik  
von **EXCEPT!**  
Für Multimengen:  
**EXCEPT ALL**

Anfrage: „Finde alle Hersteller, die Laptops, aber keine PCs herstellen.“

**DBSI - Übung**  
SQL

$$\delta(\pi_{\text{maker}}(\text{Product} \bowtie \text{Laptop})) - \delta(\pi_{\text{maker}}(\text{Product} \bowtie \text{PC}))$$

Leon Bornemann  
Chart **16**



## Product(**maker**, model, type)

- Beispieltupel: ('Lenovo', 1005, 'pc')

## PC(model, speed, ram, hd, rd, price)

- Beispieltupel: (1005, 1000, 128, 20, '12xDVD', 1499)

## Laptop(model, speed, ram, hd, screen, price)

- Beispieltupel: (2008, 650, 64, 10, 12.1, 1249)

## Printer(model, color, type, price)

- Beispieltupel: (3005, true, 'bubble', 200)

*Anfrage:* „Finde alle Harddisk-Größen, die in mehr als zwei PCs vorkommen.“

$$\pi_{hd}(\sigma_{\text{Anzahl} > 2}(\gamma_{hd, \text{count}(\text{model}) \rightarrow \text{Anzahl}}(\text{PC})))$$

Lösung:  
Deutsch → SQL

```
SELECT hd  
FROM PC  
GROUP BY hd  
HAVING COUNT(model) > 2;
```

Beispielwerte: (0000, 1234, 56789, 200)

Anfrage: „Finde alle Harddisk-Größen, die in mehr als zwei PCs vorkommen.“

$\pi_{hd}(\sigma_{\text{Anzahl} > 2}(\gamma_{hd, \text{count}(\text{model}) \rightarrow \text{Anzahl}}(\text{PC})))$

## Product(**maker**, model, type)

- Beispeltupel: ('Lenovo', 1005, 'pc')

## PC(model, speed, ram, hd, rd, price)

- Beispeltupel: (1005, 1000, 128, 20, '12xDVD', 1499)

## Laptop(model, speed, ram, hd, screen, price)

- Beispeltupel: (2008, 650, 64, 10, 12.1, 1249)

## Printer(model, color, type, price)

- Beispeltupel: (3005, true, 'bubble', 200)

Anfrage: „Finde alle Paare von PCs mit gleicher Festplatten- und Hauptspeichergröße. Vermeide dabei doppelte Paare.“

DBSI - Übung

SQL

Leon Bornemann

Chart 19

$(\rho_{PC1}(PC)) \bowtie_{PC1.HD=PC2.HD \wedge PC1.RAM=PC2.RAM \wedge PC1.model < PC2.model} (\rho_{PC2}(PC))$

Lösung:  
Deutsch → SQL

```
SELECT *  
FROM PC PC1, PC PC2  
WHERE PC1.hd = PC2.hd  
AND PC1.ram = PC2.ram  
AND PC1.model < PC2.model;
```

Beispielpaar: (6000, 4146, 80000, 200)

Anfrage: „Finde alle Paare von PCs mit gleicher Festplatten- und Hauptspeichergröße. Vermeide dabei doppelte Paare.“

**DBSI - Übung**  
SQL

Leon Bornemann  
Chart **20**

$(\rho_{PC1}(PC)) \bowtie_{PC1.HD=PC2.HD \wedge PC1.RAM=PC2.RAM \wedge PC1.model < PC2.model} (\rho_{PC2}(PC))$

## Product(**maker**, model, type)

- Beispieltupel: ('Lenovo', 1005, 'pc')

## PC(model, speed, ram, hd, rd, price)

- Beispieltupel: (1005, 1000, 128, 20, '12xDVD', 1499)

## Laptop(model, speed, ram, hd, screen, price)

- Beispieltupel: (2008, 650, 64, 10, 12.1, 1249)

## Printer(model, color, type, price)

- Beispieltupel: (3005, true, 'bubble', 200)

*Anfrage:* „Finde alle Hersteller, die mindestens einen Laptop und einen PC mit derselben Prozessorgeschwindigkeit anbieten.“

**DBSI - Übung**  
SQL

$$\pi_{D.a}(\sigma_{A.b=B.b \wedge A.a=C.g \wedge B.a=D.g \wedge C.a=D.a}(\rho_{A(a,b,c,d,e,f)}(\text{Laptop}) \times \rho_{B(a,b,c,d,e,f)}(\text{PC}) \times \rho_{C(a,g,h)}(\text{Product}) \times \rho_{D(a,g,h)}(\text{Product})))$$

Leon Bornemann  
Chart **21**

Lösung:

# Deutsch → SQL

```
SELECT D.a
FROM
  (SELECT model AS a, speed AS b, ram AS c, hd AS d, screen AS e, price AS f FROM Laptop) A,
  (SELECT model AS a, speed AS b, ram AS c, hd AS d, rd AS e, price AS f FROM PC) B,
  (SELECT maker AS a, model AS g, type AS h FROM Product) C,
  (SELECT maker AS a, model AS g, type AS h FROM Product) D
WHERE A.b = B.b
AND A.a = C.g
AND B.a = D.g
AND C.a = D.a;
```

Beispielantwort: (0000, 1000, 80000, 200)

Anfrage: „Finde alle Hersteller, die mindestens einen Laptop und einen PC mit derselben Prozessorgeschwindigkeit anbieten.“

**DBSI - Übung**  
SQL

$$\pi_{D.a}(\sigma_{A.b=B.b \wedge A.a=C.g \wedge B.a=D.g \wedge C.a=D.a}(\rho_{A(a,b,c,d,e,f)}(\text{Laptop}) \times \rho_{B(a,b,c,d,e,f)}(\text{PC}) \times \rho_{C(a,g,h)}(\text{Product}) \times \rho_{D(a,g,h)}(\text{Product})))$$

Leon Bornemann  
Chart 22

Lösung:

Deutsch → SQL

```
SELECT D.maker  
FROM Laptop A, PC B, Product C, Product D  
WHERE A.speed = B.speed  
AND A.model = C.model  
AND B.model = D.model  
AND C.maker = D.maker;
```

Beispielwerte: (0000, 1234, 56789, 200)

Anfrage: „Finde alle Hersteller, die mindestens einen Laptop und einen PC mit derselben Prozessorgeschwindigkeit anbieten.“

$$\pi_{D.a}(\sigma_{A.b=B.b \wedge A.a=C.g \wedge B.a=D.g \wedge C.a=D.a}(\rho_{A(a,b,c,d,e,f)}(\text{Laptop}) \times \rho_{B(a,b,c,d,e,f)}(\text{PC}) \times \rho_{C(a,g,h)}(\text{Product}) \times \rho_{D(a,g,h)}(\text{Product})))$$

**DBSI - Übung**  
SQL

Leon Bornemann  
Chart **23**

Aufgabe:

Deutsch → SQL

```
WITH Rechner AS (  
    (SELECT model FROM PC) UNION  
    (SELECT model FROM Laptop))  
SELECT Product.model,screen  
FROM Product LEFT OUTER JOIN laptop ON Product.model = Laptop.model  
WHERE maker = 'A'  
AND Product.model IN (Select * FROM Rechner)
```

Beispielantwort: (0000, 'Mac', '1330', 200)

Anfrage: „Finde die Modellnummern aller Laptops und PCs,  
die von 'Apple' hergestellt wurden. Gib für Laptops zusätzlich die  
Bildschirmgröße aus.“

**DBSI - Übung**  
SQL

Leon Bornemann  
Chart **24**



Aufgabe:

Deutsch → SQL

```
SELECT model,screen FROM (  
    (SELECT model,null AS screen FROM PC ) UNION  
    (SELECT model,screen FROM LAPTOP)) AS devices  
WHERE model IN  
    (SELECT model FROM Product WHERE maker = 'A')
```

Beispielantwort: (0000, 15.4, 1330, 200)

Anfrage: „Finde die Modellnummern aller Laptops und PCs, die von ‘Apple’ hergestellt wurden. Gib für Laptops zusätzlich die Bildschirmgröße aus.“

**DBSI - Übung**  
SQL

Leon Bornemann  
Chart **25**

Aufgabe:

## Deutsch → SQL

```
WITH max_Speeds AS (  
    SELECT maker, MAX(speed) as max_speed  
    FROM product NATURAL JOIN pc  
    GROUP BY maker)  
Select p.maker, p.model, ms.max_speed, pc.price  
FROM max_Speeds ms, product p, pc  
WHERE ms.maker = p.maker  
AND pc.model = p.model  
AND pc.speed = ms.max_speed  
ORDER BY max_speed DESC, price ASC
```

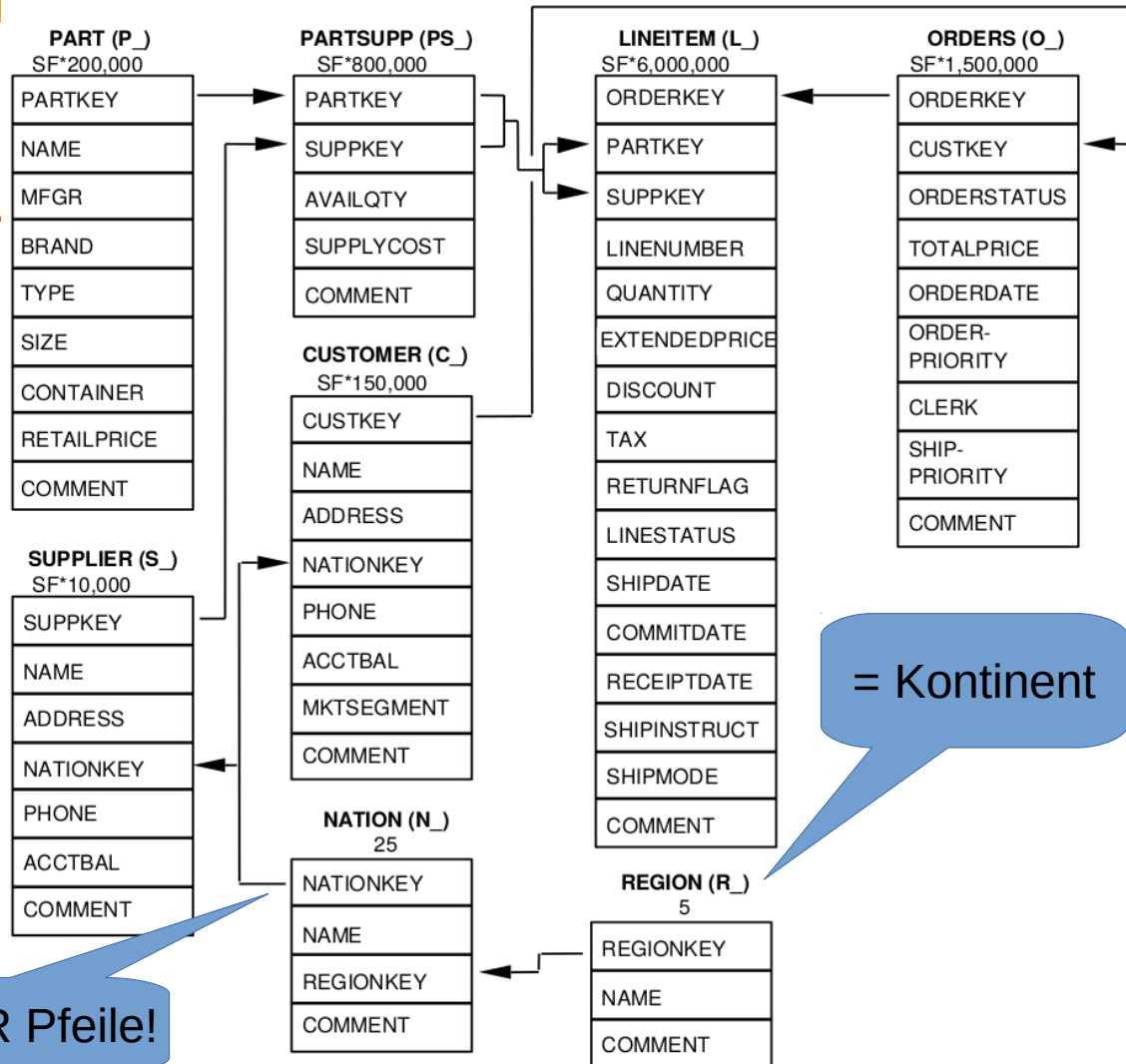
*Anfrage:* „Gib für alle Hersteller die Modellnummern der PCs mit der maximale Prozessorgeschwindigkeit, sowie deren Preis aus. Sortiere das Ergebnis absteigend nach Prozessorgeschwindigkeit und Aufsteigend nach Preis.“


**DBSI - Übung**  
SQL

Leon Bornemann  
Chart **26**

# Sales Schema

- Fragen zum Schema?
- Fragen zum Datenimport?
- Was für Queries (Business-Fragen) könnte man hier stellen?





Übung Datenbanksysteme I  
SQL

G-3.1.14, Campus III / F-E.06 Campus II

Leon Bornemann  
Hasso Plattner Institut