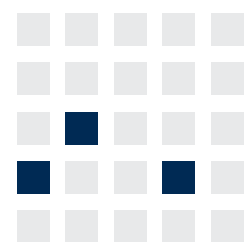


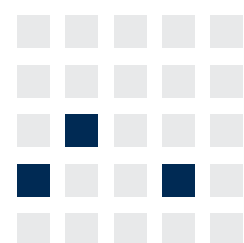
Einführung in die Wirtschaftsinformatik

Teil 12 – Abfragen und Joins

Wintersemester 2022/2023



**Lehrstuhl für Wirtschaftsinformatik
Prozesse und Systeme**
Universität Potsdam



**Chair of Business Informatics
Processes and Systems**
University of Potsdam

Univ.-Prof. Dr.-Ing. habil. Norbert Gronau
Lehrstuhlinhaber | Chairholder

Mail August-Bebel-Str. 89 | 14482 Potsdam | Germany
Visitors Digitalvilla am Hedy-Lamarr-Platz, 14482 Potsdam
Tel +49 331 977 3322

E-Mail ngronau@lswi.de
Web lswi.de



Alternative Ausdrücke

Abfragen über mehrere Tabellen

Bedingte Ausdrücke

CASE-Ausdrücke

```
CASE ausdruck WHEN comp_ausdruck1 THEN return_ausdruck_1
              [WHEN comp_ausdruck_2 THEN return_ausdruck_2
               WHEN comp_ausdruck_n THEN return_ausdruck_n
               ELSE else_ausdruck]
END
```

DECODE

```
DECODE (col|ausdruck, such1, ergebnis_1
        [, such2, ergebnis_2,...,]
        [, default]);
```

...stellen eine IF-THEN-ELSE-Logik innerhalb der SQL-Anweisung bereit.

Anwendung von CASE-Ausdrücken

- Bedingte Abfragen in der Form von IF-THEN-ELSE-Anweisung

```
SELECT name, position, gehalt,  
       CASE position WHEN 'Konstrukteur' THEN gehalt*1.10  
                     WHEN 'Konstrukteurin' THEN gehalt*1.10  
                     WHEN 'Monteur' THEN gehalt*1.15  
                     WHEN 'Meister' THEN gehalt*1.05  
       ELSE gehalt END Gehaltserhöhung  
FROM mitarbeiter ORDER BY position;
```

NAME	POSITION	GEHALT	GEHALTSERHÖHUNG
...	...	...	...
Klein	Konstrukteur	4100	4510
Adam	Konstrukteurin	4700	5170
...	...	...	...
Ascheid	Meister	3990	4189,5
Krause	Meister	3680	3864
...	...	...	...
Thal	Monteur	2400	2760
Möhlmann	Monteur	2740	3151
...	...	...	...

Hinweis: CASE-Klausel schließt mit END-Ausdruck.
WHEN-Klausel bezieht sich nur auf den Vergleich mit jeweils einem konkreten Wert.

Anwendung der Funktion DECODE

- Wirkung der Anweisung ähnlich dem CASE-Ausdruck

```
SELECT name, position, gehalt,  
       DECODE(position, 'Konstrukteur', gehalt*1.10,  
                    'Konstrukteurin', gehalt*1.10,  
                    'Monteur', gehalt*1.15,  
                    'Meister', gehalt*1.05, gehalt) "Neues  
       Gehalt" FROM mitarbeiter ORDER BY position;
```

NAME	POSITION	GEHALT	NEUES GEHALT
...	...	...	...
Klein	Konstrukteur	4100	4510
Adam	Konstrukteurin	4100	4510
...	...	...	...
Ascheid	Meister	3990	4189,5
Krause	Meister	3680	3864
...	...	...	...
Thal	Monteur	2400	2760
Möhlmann	Monteur	2740	3151
...	...	...	...

Hinweis: DECODE-Klausel schließt mit default-Wert ab. Wird dieser weggelassen, dann wird bei fehlender Übereinstimmung in allen Argumenten NULL-Wert ausgegeben

Abfragereihen mit der Funktion DECODE

- Anzeige des Steuersatzes aller Angestellten der Abteilung 310T

```
SELECT name, position, abt_nr, gehalt,  
       DECODE (TRUNC (gehalt/1000, 0),  
               0, '00%',  
               1, '20%',  
               2, '25%',  
               3, '30%',  
               4, '35%',  
               5, '40%',  
               '45%') AS "Steuersatz"  
FROM mitarbeiter WHERE abt_nr = '310T';
```

TRUNC(x,y) schneidet von einem Wert **x** Nachkommastellen ab, sodass nur noch **y** Nachkommastellen verbleiben

NAME	POSITION	ABT_NR	GEHALT	STEUERSATZ
Berg	Sekretärin	310T	2250	25%
Kettler	Abteilungsleiter	310T	8080	45%
Klein	Konstrukteur	310T	4100	35%
Plenk	Konstrukteur	310T	4000	35%



Alternative Ausdrücke

Abfragen über mehrere Tabellen

Joins - Abfragen über mehrere Tabellen

Projektion

A	B	C	D	E

Selektion

A	B	C	D	E

Join - Verbinden von Tabellen

A	B	C	D	E

F	G	H

D	F

Daten, die in unterschiedlichen Tabellen gespeichert sind, werden zusammengeführt. Die Auswahl von Daten erfolgt über eine Verknüpfung mehrerer Tabellen

Verknüpfung von Tabellen - Grundlagen, Syntax

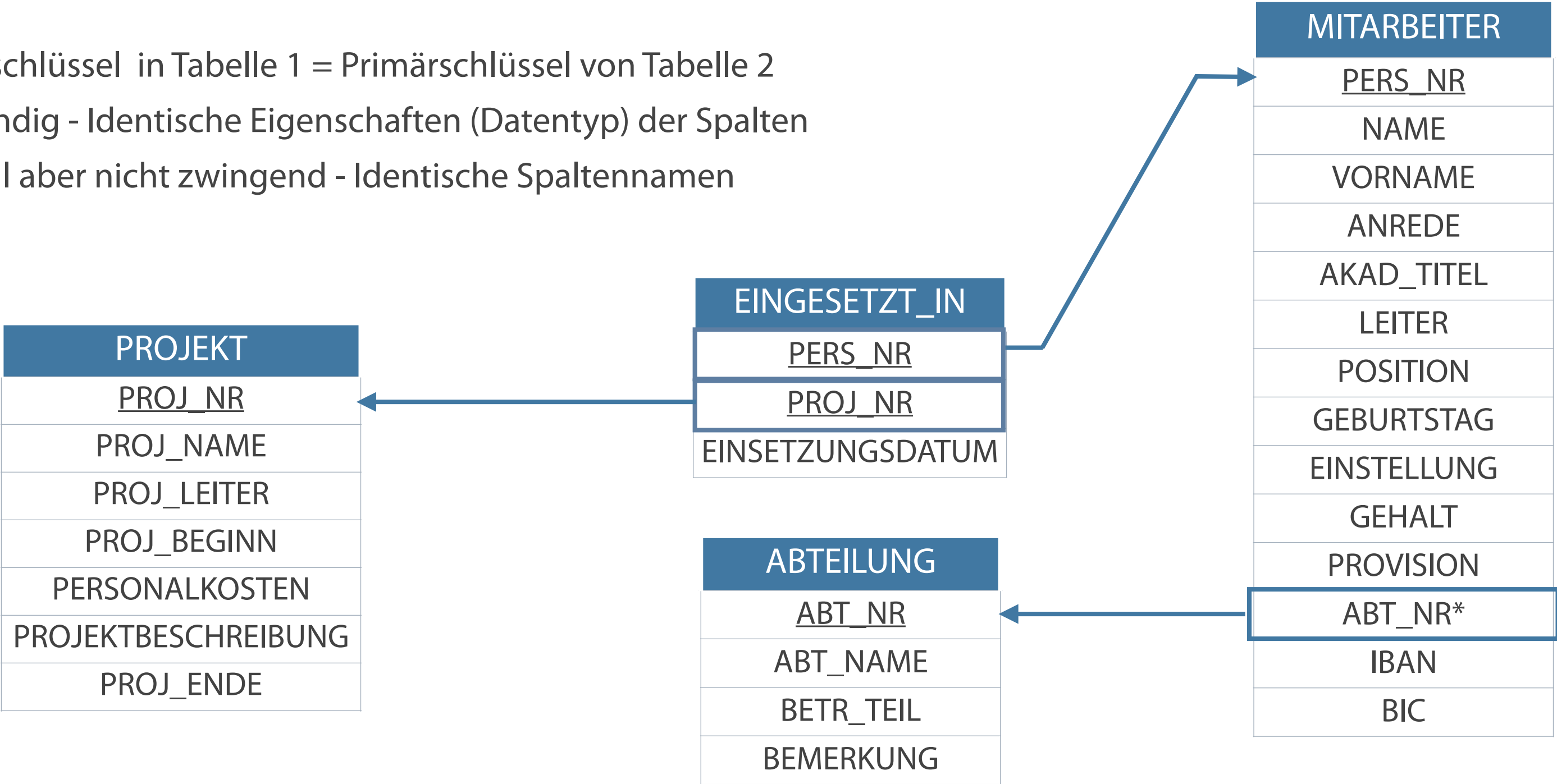
- Abfrage von Daten aus mehreren Tabellen mit JOIN
- Join-Bedingung in der WHERE-Klausel
- Tabellename steht vor Spaltennamen durch Punkt getrennt
- Syntax: tabellename.spaltenname

*Kann - bei unterschiedlichen Spaltennamen in den Tabellen
Muss - bei denselben Spaltennamen in verschiedenen Tabellen*

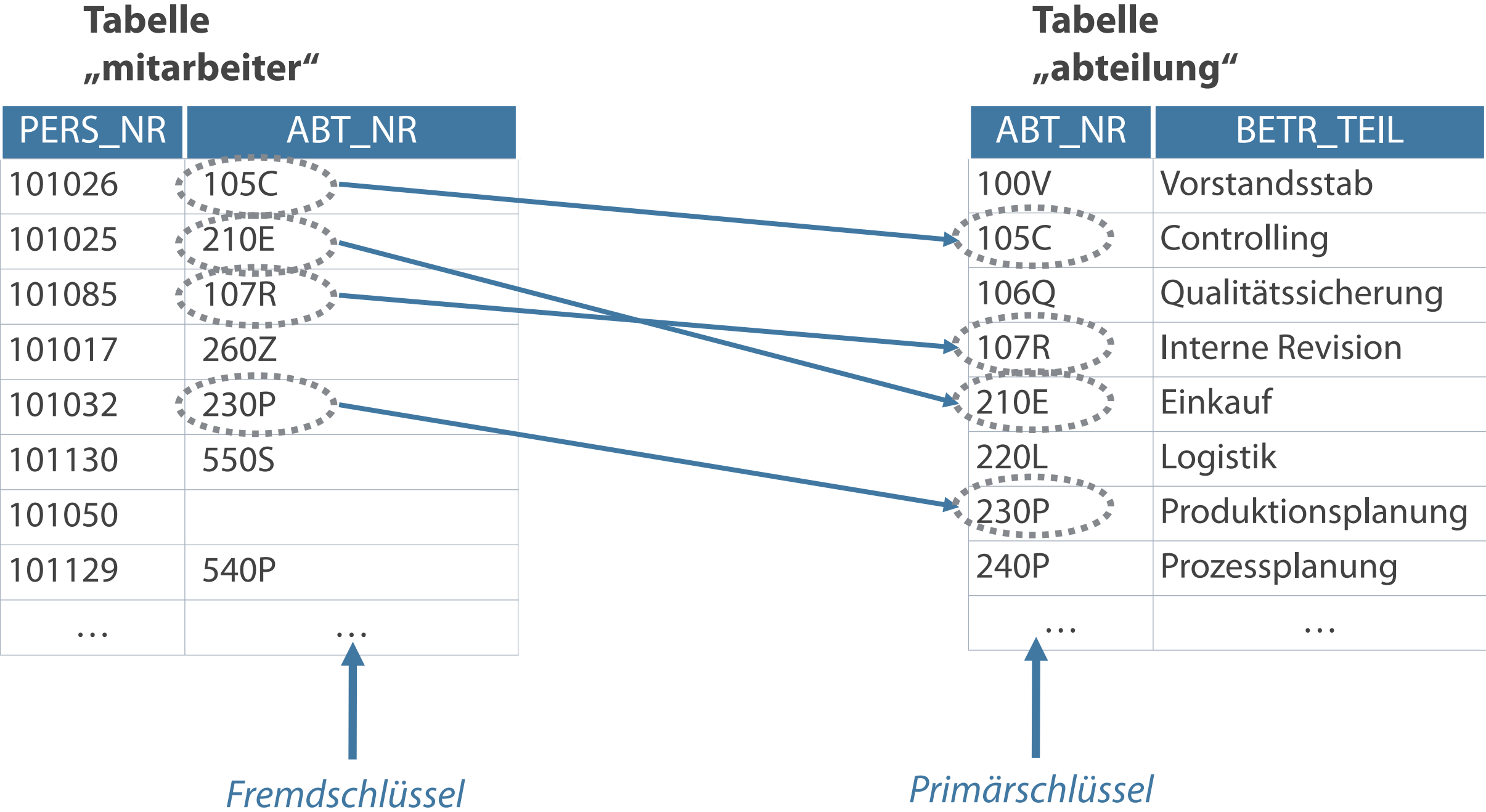
```
SELECT tabelle1.spalte_n, tabelle2.spalte_m  
FROM tabelle1, tabelle2  
WHERE tabelle1.spalte1 = tabelle2.spalte2;
```

Voraussetzungen für die Verknüpfung von Tabellen

- Fremdschlüssel in Tabelle 1 = Primärschlüssel von Tabelle 2
- Notwendig - Identische Eigenschaften (Datentyp) der Spalten
- Sinnvoll aber nicht zwingend - Identische Spaltennamen

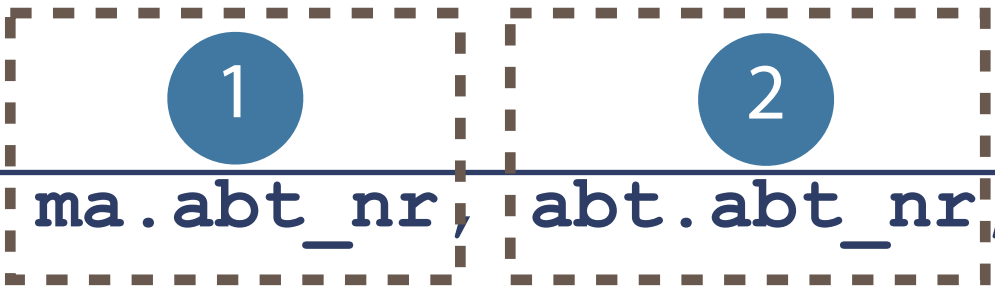


Equi-Joins



Equi-Joins verbinden zusammengehörende Datensätze mehrerer Tabellen.

Aufruf von Datensätzen über Equi-Joins



```
SELECT ma.pers_nr, ma.name, ma.abt_nr, abt.abt_nr,  
       abt.abt_name  
FROM mitarbeiter ma, abteilung abt  
WHERE ma.abt_nr = abt.abt_nr;
```

PERS_NR	NAME	1 ABT_NR	2 ABT_NR	ABT_NAME
...	...	...	...	...
101001	Büchner	260F	260F	Produktion-Fertigung
101002	Martens	260D	260D	Werkstatt
101003	Dost	210E	210E	Einkauf
...	...	...	...	...
101044	Behrens	220L	220L	Logistik
101045	Krause	640S	640S	Technik Service, Support
...	...	...	...	...

Umgang mit gleichen Spaltennamen in unterschiedlichen Tabellen

```
tab_name_a.spalte_x
tab_name_b.spalte_x
```

```
Name AS
Mitarbeitername
```

Kennzeichnung	Geschwindigkeit	Zuordnung
■ Eindeutige Kennzeichnung von Spaltennamen bei Vorkommen in mehreren Tabellen --> Einsatz von Präfixen	■ Performance- Verbesserung durch Angabe von Präfixen	■ Eindeutigkeit bei Namensgleichheit ■ Unterscheidung bei identischem Namen in unterschiedlichen Tabellen über Spalten-Alias

Spaltennamen werden, wenn sie in mehreren Tabellen vorkommen, durch Präfixe gekennzeichnet.

Beispiel - Problem der Namensgleichheit

Mehrdeutigkeit bei Namensgleichheit von Spalten

- Zuordnung Spaltenname zu Tabellennamen in Abfrage notwendig

A	SELECT pers_nr, name, abt_nr , abt_name FROM mitarbeiter, abteilung;	Fehler: Spalte "abt_nr" nicht eindeutig definiert
B	SELECT pers_nr, name, m.abt_nr , abt_name FROM mitarbeiter m , abteilung a;	Zuordnung der Spalte "abt_nr" erfolgt aus Tabelle "mitarbeiter"
C	SELECT pers_nr, name, a.abt_nr , abt_name FROM mitarbeiter m, abteilung a ;	Zuordnung der Spalte "abt_nr" erfolgt aus Tabelle "abteilung"

Werte der
Ausgabetabelle für
Beispiele B und C
sind identisch

Bei JOIN-Anweisungen muss jede Spalte einer der Tabellen eineindeutig zuordenbar sein, um eine korrekte Ausgabetabelle zu generieren.

Vereinfachung von Anfragen - Aliasnamen

Kürzen

- Verkürzen bzw. Abkürzen langer oder zusammengesetzter bzw. komplexer Tabellennamen

Beispiele

- ... FROM artikelstammdaten artikel
- ... FROM abteilung a, mitarbeiter m

Verlängern

- Lesbarmachen kryptischer oder missverständlicher Tabellenkürzel

Beispiele

- ... FROM abt abteilung, ma Mitarbeiter
- ... FROM btl betriebsteil

Zusätzliche Selektionskriterien in der Abfrage

Weitere Einschränkung der Anzeige von Datensätzen in der Abfrage

- Verwendung des Operators AND

```
SELECT ma.pers_nr, ma.name, ma.abt_nr, ma.position,  
       abt.abt_name  
FROM mitarbeiter ma, abteilung abt  
WHERE ma.abt_nr = abt.abt_nr  
      AND position = 'Sekretärin';
```

PERS NR	NAME	ABT NR	POSITION	ABT NAME
...	...	...	...	...
102050	Gutsche	110V	Sekretärin	Vertriebsteam Amerika
101004	Fuchs	260D	Sekretärin	Werkstatt
101022	Brauer	210E	Sekretärin	Einkauf
101031	Groß	230P	Sekretärin	Produktionsplanung
...	...	...	...	...
101114	Listig	330E	Sekretärin	Konstruktion Elektrik
101118	Groß	340R	Sekretärin	Konstruktion Research
102058	Tuncpinar	410V	Sekretärin	Vertriebsteam West

Natural Join

Verknüpfung über NATURAL JOIN-Klausel

- Basiert auf allen Spalten mit demselben Spaltentitel in beiden Tabellen
- Automatisches Einbinden aller Spalten mit gleichen Namen und gleichem Datentyp
- Auswahl der Zeilen mit übereinstimmenden Werten in allen gemeinsamen Spalten
- Wirkungsweise analog des Joins mittels WHERE-Klausel

Problemfelder

- Rückgabe einer Fehlermeldung bei Namensgleichheit aber unterschiedlichen Datentypen der Spalten
- Einsatz eines Präfixes vor verknüpften Spaltennamen ist nicht zulässig - Fehlermeldung

Datensatzaufruf über Natural Join

Formulierung über Equi-Join

```
SELECT abt_nr, abt_name, a.betr_teil, betriebsteilname
FROM abteilung a, betriebsteil b
WHERE a.betr_teil = b.betr_teil;
```

Verkürzung über Natural Join

```
SELECT abt_nr, abt_name, betr_teil, betriebsteilname
FROM abteilung NATURAL JOIN betriebsteil;
```

ABT_NR	ABT_NAME	BETR_TEIL	BETRIEBSTEILNAME
520G	Personal Gehalt	FR	Bereich Finanzen
510L	Personal Lohn	FR	Bereich Finanzen
420F	Finanzbuchhaltung	FR	Bereich Finanzen
...	...	...	...
260K	Werkstatt kubische	PO	Produktion, Organisation
108A	Marketing	V	Vertriebs GmbH
...	...	...	...

Durch den NATURAL JOIN kann in diesem Fall auf die WHERE-Klausel und die Alias verzichtet werden.

On-Klausel zur Angabe der Join-Bedingung

Erhöhung der Anweisungsverständlichkeit

- Wirkung der Join-Bedingung für den Natural-Join analog einem Equi-Join aller Spalten mit gleichem Namen
- On-Klausel dient der Festlegung von beliebigen Bedingungen oder der Angabe zu verknüpfender Spalten

```
SELECT pers_nr, name, vorname, abt_name
FROM mitarbeiter m JOIN abteilung a
ON m.abt_nr = a.abt_nr AND Gehalt < 900;
```

PERS_NR	NAME	VORNAME	ABT_NAME
101077	Cho	Melanie	Qualitätssicherung
101079	Schuster	Jens	Produktion-Fertigung
101088	Assmann	Niels	Produktion-Fertigung
101089	Schuster	Anika	Rechnungswesen
101080	Junge	Willi	Technik Endgeräte
101090	Kohl	Melanie	Technik Service, Support

Verknüpfung mehrerer Tabellen

- Zusätzliche Suchkriterien in WHERE-Klausel
- Verknüpfung von n Tabellen - mindestens n-1 Join-Bedingungen

mitarbeiter		abteilung		betriebsteil	
NAME	ABT_NR	ABT_NR	BETR_TEI	BETR_TEIL	BEREICHSNAME
Melzer	100V	100V	GL	GL	Zentrale
Bormann	340R	340R	KE	KE	Konstruktion, Entwicklung
Hofmann	100V	100V	GL	GL	Zentrale
Würz	610A	610A	IT	IT	Technik
Joseph	260Z	260Z	PO	PO	Produktion, Organisation
Cho	106Q	106Q	GL	GL	Zentrale
...	...	...	...	...	...

```
SELECT name, m.abt_nr, a.abt_nr, a.betr_teil, b.betr_teil,
betriebssteilname AS bereichsname
FROM mitarbeiter m, abteilung a, Betriebsteil b
WHERE m.abt_nr = a.abt_nr AND a.betr_teil = b.betr_teil;
```

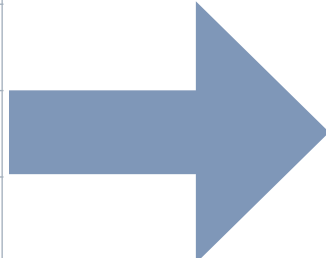

Non-Equi-Joins

Verbundbedingung ohne Gleichheitsprüfung

- Join-Bedingung, die nicht den Gleich-Operator verwendet
- Beispiel: Ungleichheitszeichen (<> oder !=)

mitarbeiter

NAME	GEHALT
Reinhard	56000
Johansson	35000
...	...
Riekhoff	3480
Fuchs	3600
...	...
Assmann	500
Kohl	500
Schuster	500



gratifikation

MINGEHALT	MAXGEHALT	GRATIF
0	2500	65
2501	3500	55
3501	5000	45
5001	7500	35
7501	1999999	25

Prüfung der Werte aus "GEHALT" mit den Spalten "MINGEHALT" und "MAXGEHALT" über WHERE-Klausel auf Ungleichheit
Zuordnung des jeweiligen Werte aus Spalte "GRATIF", wenn Ungleichheitsbedingung erfüllt ist

Beispiel für einen Non-Equi-Join

Verbundbedingung mit BETWEEN ... AND

```
SELECT m.name, CONCAT(m.gehalt, '€')gehalt, CONCAT(g.gratif, ', %')gratif
FROM mitarbeiter m, gratifikation g
WHERE m.gehalt
BETWEEN g.mingehalt AND g.maxgehalt;
```

NAME	GEHALT	GRATIF
Assmann	500€	65%
Schuster	500€	65%
...	...	...
Riekhoff	3480€	55%
Fuchs	3600€	45%
...	...	...
Michalke	8440€	25%
Schulz-Niemeyer	8900€	25%
Büchner	10430€	25%

Anzeige von Datensätzen nur bei erfüllter Bedingung

Zum Vergleich: Tabelle Gratifikation

MINGEHALT	MAXGEHALT	GRATIF
0	2500	65
2501	3500	55
3501	5000	45
5001	7500	35
7501	1999999	25

Die Verwendung der Tabellen-Aliasnamen erfolgt oft aus Performance-Gründen.

Outer-Joins

Besondere Form einer Verbundbedingung

- Anzeige der in direkter Beziehung stehenden Datensätze **und**
- Anzeige von Datensätzen, ohne direkten Bezug zu Datensätzen anderer Tabellen

abteilung

ABT_NR	ABT_NAME
100V	Vorstandsstab
105C	Controlling
106Q	Qualitätssicherung
...	...
530A	Personal Leiharbeit
540P	Personalentwicklung
550S	Arbeitssicherheit
...	...

mitarbeiter

PERS_NR	NAME	ABT_NR
...	...	...
101029	Roth	105C
101030	Riekhoff	-
101031	Groß	230P
101032	Eckert	230P
101033	Schulze	250A
101034	Grothe	260F
101035	John	105C
...	...	...

Abteilung 550S ist noch
ohne Mitarbeiter

Ausgangssituation

Mitarbeiter 101030 ist in
keiner Abteilung

Outer-Joins - Syntax

RIGHT: Aufruf aller Zeilen der rechten Tabelle nach
ON

```
SELECT tabelle1.spalte, tabelle2.spalte, ...  
FROM tabelle1  
RIGHT OUTER JOIN tabelle2  
ON tabelle1.spalte = tabelle2.spalte;
```

LEFT: Aufruf aller Zeilen der linken Tabelle nach ON

```
SELECT tabelle1.spalte, tabelle2.spalte, ...  
FROM tabelle1,  
LEFT OUTER JOIN tabelle2  
ON tabelle1.spalte = tabelle2.spalte;
```

Mit einem Right- bzw. Left-Outer-Join wird eine sogenannte rechte bzw. linke Inklusionsverknüpfung erstellt.

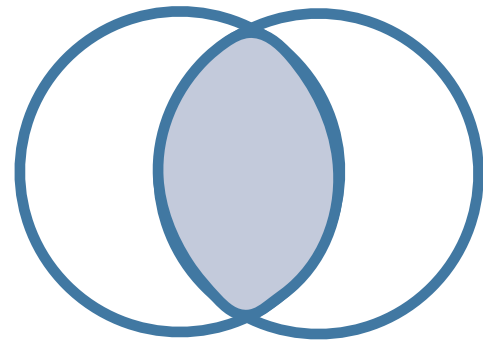
Aufruf eines Outer-Join

Auswahl auch von nicht verknüpften Datensätzen

```
SELECT m.name, m.abt_nr, a.abt_name
FROM mitarbeiter m
RIGHT OUTER JOIN abteilung a
ON m.abt_nr = a.abt_nr;
```

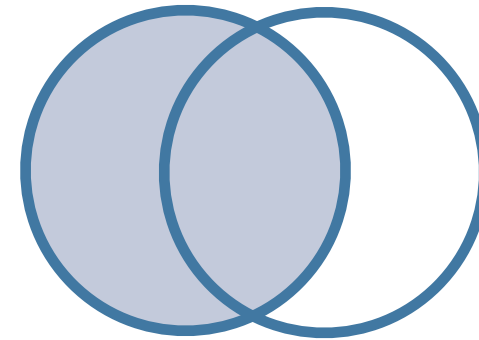
NAME	ABT_NR	ABT_NAME	
...	...	...	
Seljukow	500V	Vertriebsteam Deutschland	
Büchner	260F	Produktion-Fertigung	
Martens	260D	Werkstatt Drehteilefertigung	
Dost	210E	Einkauf	
...	...	...	
Mehmedovic	10VL	Leitung Vertrieb	
Winter	110V	Vertriebsteam Amerika	
Probst	410V	Vertriebsteam West	
-	-	Marketing	keine Daten in der Tabelle "mitarbeiter"
-	-	Elektro-Werkstatt	

Übersicht: Inner- und Outer-Joins



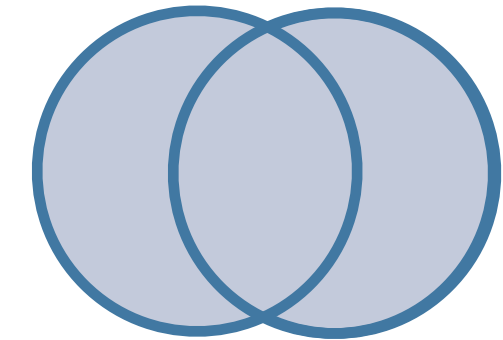
Inner-Join (Equi-Join)

- Verknüpfung zwischen zwei Tabellen
- Rückgabe nur von übereinstimmenden Zeilen



Left-Outer-Join

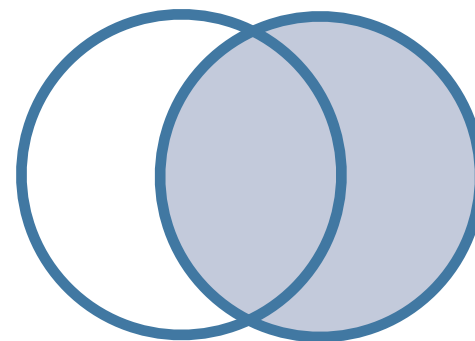
- Ergebnissrückgabe des Inner-Joins sowie Zeilen ohne Übereinstimmungen in der linken Tabelle



Full-Outer-Join

- Ergebnissrückgabe des Inner-Joins sowie eines Left-Outer-Joins **und** Right-Outer-Joins

Right-Outer-Join



- Ergebnissrückgabe des Inner-Joins sowie Zeilen ohne Übereinstimmungen in der rechten Tabelle

Full-Outer-Join

Kombination von Left-Outer-Join und Right-Outer-Join

- Anzeige aller in den verknüpften Basistabellen vorhandenen Zeilen

```
SELECT m.name, m.abt_nr, a.abt_name
FROM mitarbeiter m
FULL OUTER JOIN abteilung a
ON m.abt_nr = a.abt_nr;
```

NAME	ABT_NR	ABT_NAME
Büchner	260F	Produktion-Fertigung
Martens	260D	Werkstatt Drehteilefertigung
Dost	210E	Einkauf
...	...	...
Krajcsir		
Walker		
...	...	...
Brauer	210E	Einkauf
		Elektro-Werkstatt

Self-Join

Verknüpfung einer Tabelle mit sich selbst

- Zusätzliche Bezeichner/Alias-Namen in der FROM-Klausel zur Unterscheidung

```
SELECT a.pers_nr AS "Pers_Nr (Angestellte)", a.name AS
"Angestelltenname", a.anrede AS "Anrede", a.leiter AS
"Leiter(Pers_nr)", b.name AS "Name des Leiters", b.anrede
AS "Anrede"
FROM mitarbeiter a, mitarbeiter b
WHERE a.leiter = b.pers_nr
ORDER BY a.pers_nr;
```

Pers_Nr (Angestellte)	Angestelltenname	Anrede	Leiter (Pers_Nr)	Name des Leiters	Anrede
101001	Büchner	Herr	101060	Köhler	Herr
101002	Martens	Herr	101060	Köhler	Herr
101003	Dost	Herr	101059	Ernst	Herr
101004	Fuchs	Frau	101060	Köhler	Herr
101005	Rösch	Herr	101047	Klemm	Frau
...	...	...	...	...	...

Ein Self-Join ermöglicht die Herstellung von Verbindungen innerhalb einer Tabelle mit einer einzigen Abfrage.

Syntax für Self-Joins

Formale Betrachtung auf zwei Seiten einer Tabelle

```
SELECT tab1.spalte, ..., tab2.spalte
FROM tabelle1 tab1, tabelle1 tab2
WHERE tab1.spalte1 = tab2.spalte2;
```

Wer ist der/die Vorgesetzte jedes Mitarbeiters?

```
SELECT leiter.anrede || ' ' || leiter.name || ' ist
Vorgesetzte/r von ' || angestellte.anrede || ' ' ||
angestellte.name
FROM mitarbeiter angestellte, mitarbeiter leiter
WHERE angestellte.leiter = leiter.pers_nr;
```

LEITER.NAME 'ISTVORGESETZE/RVON' ANGESTELLTE.NAME
...
Herr Schmiedel ist Vorgesetzte/r von Herr Beyerke
...

Erzeugen eines kartesischen Produktes aus zwei Tabellen

Konstruktion zur Erzeugung einer neuen Menge

- Verbinden mehrerer Tabellen ohne Verknüpfungskriterium

```
SELECT spalte1, spalte2, ...  
FROM tabelle1  
CROSS JOIN tabelle2;
```

- Erzeugt dasselbe Ergebnis wie die simple Syntax:

```
SELECT spalte1, spalte2, ...  
FROM tabelle1, tabelle2;
```

- ...und liefert meist sinnlose Ergebnisse

Aber:

- Vorteil für Test von Datenbanken - Einfaches Erzeugen großer Datenmengen

Kartesisches Produkt

Bildung eines kartesischen Produktes

- Verknüpfung aller Zeilen aus der ersten Tabelle mit allen Zeilen aus der zweiten Tabelle
- Fehlende Join-Bedingung
- Ungültige Join-Bedingung

Grundprinzip

- Verknüpfung jeder Zeile einer Tabelle mit jeder Zeile der anderen Tabelle(n)
- Beispiel auf den folgenden Seiten - kartesisches Produkt aus Tabellen

mitarbeiter x abteilung —> 199 Zeilen x 37 Zeilen —>
—> 14 Spalten + 5 Spalten —>

7363 Zeilen
19 Spalten
Ergebnistabelle

Vermeidung eines kartesischen Produktes

- Gültige Join-Bedingung in der WHERE-Klausel
- Voraussetzung - Beziehung über Fremd- und Primärschlüssel bzw. Felder mit identischen Datentypen

Das formale Vorgehen beim Kartesischen Produkt entspricht dem beim Vektorprodukt in der Mathematik.

Bildung eines Kartesischen Produktes

mitarbeiter
PERS_NR
NAME
VORNAME
ANREDE
AKAD_TITEL
LEITER
POSITION
GEBURTSTAG
EINSTELLUNG
GEHALT
PROVISION
ABT_NR
IBAN
BIC

X

abteilung
ABT_NR
ABT_NAME
BETR_TEIL
BEMERKUNG

=

Ergebnis der Abfrage
PERS_NR
NAME
VORNAME
ANREDE
AKAD_TITEL
LEITER
POSITION
GEBURTSTAG
EINSTELLUNG
GEHALT
PROVISION
ABT_NR
IBAN
BIC
ABT_NR
ABT_NAME
BETR_TEIL
BEMERKUNG

Bei einem Kartesischen Produkt werden alle Spalten der betreffenden Tabellen miteinander "gekreuzt".

Kartesisches Produkt - Beispiel

7164 Zeilen

PERS_NR	NAME	VORNAME	...	ABT_NR	...	ABT_NR	ABT_NAME	BETR_TEIL	...
101001	Büchner	Edgar	...	260F	...	100V	Vorstandsstab	GL	...
101001	Büchner	Edgar	...	260F	...	105C	Controlling	GL	...
101001	Büchner	Edgar	...	260F	...	106Q	Qualitätssicherung	GL	...
...	...	...	...	...	...	...	...	...	...
101001	Büchner	Edgar	...	260F	...	260E	Elektro-Werksatt	PO	...
101001	Büchner	Edgar	...	260F	...	260F	Produktions-Fertigung	PO	...
...	...	...	...	...	...	...	...	...	...
101002	Martens	Eugen	...	260F	...	100V	Vorstandsstab	GL	...
101002	Martens	Eugen	...	260F	...	105C	Controlling	GL	...
...	...	...	...	...	...	...	...	...	...
101002	Martens	Eugen	...	260F	...	260F	Produktions-Fertigung	PO	...
...	...	...	...	...	...	...	...	...	...

21 Spalten

Die umrahmten Bereiche beinhalten Ergebnisse des Kartesischen Produktes mit einem logischen Zusammenhang.

Ein kartesisches Produkt mehrerer Tabellen erzeugt (in der Regel) keine sinnvolle Ergebnistabelle.

Zusammenfassung - Sprachelemente (SELECT)

Vollständige Form (Grundgerüst) einer SELECT-Anweisung

```
SELECT [ DISTINCT ] attribut_1, ..., attribut_n
FROM tabelle ...
[ WHERE bedingung ]
[ GROUP BY attributname ]
[ ORDER BY attributname ]
[ INTO TEMP tabname ];
```

SELECT..... **Was?**(Auswahl verwendeter Attribute)

DISTINCT..... Ignoriert Duplikate von Attributwerten

FROM.....**WOHER?** (aus den tabellen)

WHERE..... **Wobei?**(mit der Bedingung/welche Zeilen)

GROUP BY..... Gruppenbildung von Daten in einer Spalte

ORDER BY..... **Wie?**(Sortierung der Inhalte der Attribute)

INTO TEMPTEMPORÄR (Name der neuen temporären Tabelle)

Selektion, Projektion und Join - Fazit

Zusammenfassung

SELECT mitarbeiter.name, abteilung.name

Projektion

FROM mitarbeiter, abteilung

Join*

WHERE mitarbeiter.abt_nr = abteilung.abt_nr
AND abteilung.abt_nr = XXX**

Selektion

* Verbund (Kartesisches Produkt)

** Konkreter Wert

Kontrollfragen

- Wann werden CASE-Ausdrücke in SQL-Anweisungen verwendet?
- Nach der Normalisierung sind ursprüngliche Tabellen oftmals in mehrere neue Tabellen aufgeteilt. Mit welcher Funktion können die Daten daraus wieder miteinander verbunden werden?
- Unter welchen Bedingungen wird ein kartesisches Produkt erzeugt?
- Was bewirkt die Verwendung eines LEFT OUTER JOIN in einer Anweisung?

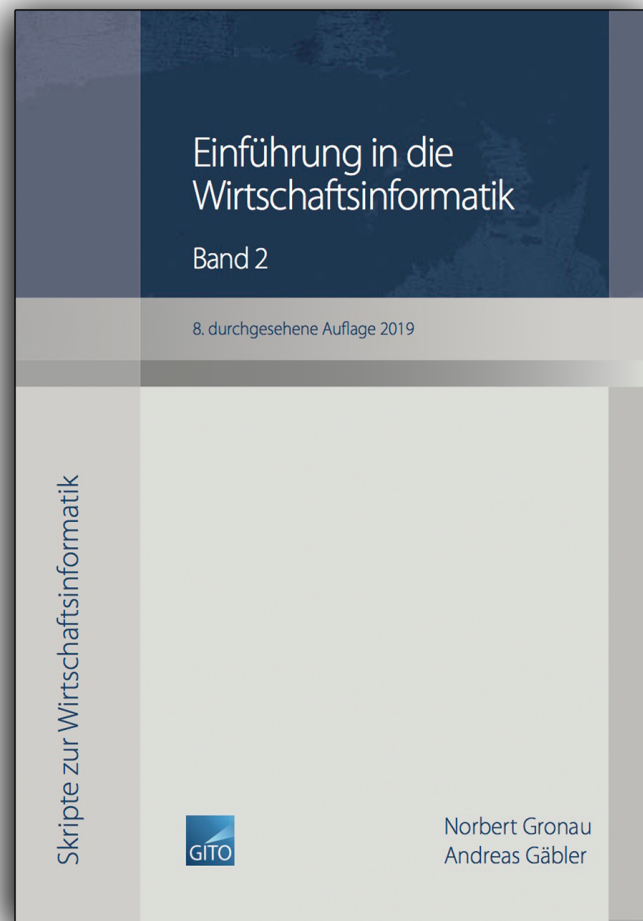
Literatur

Kemper, A./Eickler, A.: Datenbanksysteme; 10. Auflage, 2015, Oldenbourg Verlag

Elmazri, R./Navathe, S. B.: Grundlagen von Datenbanksystemen; 3. Auflage, 2002, Addison-Wesley

Greenberg, N./Nathan, P: Professioneller Einstieg in Oracle9i SQL - Band 1; 2002, Oracle

Kähler, W.-M.: SQL mit Oracle; 3. Auflage, 2008, Vieweg



Kontakt

Univ.-Prof. Dr.-Ing. Norbert Gronau

Universität Potsdam
Karl-Marx-Str. 67 | 14482 Potsdam
Germany

Tel. +49 331 977 3322
E-Mail ngronau@lswi.de

Gronau, N., Gäbler, A.:
Einführung in die Wirtschaftsinformatik, Band 2
8. überarbeitete Auflage
GITO Verlag Berlin 2019, ISBN 978-3-95545-285-8