4.2 SQL-Anfragen 1

- 4.1 SQL – DDL und DML
- 4.2 SQL-Anfragen 1
- 4.3 SQL-Anfragen 2
- 4.4 Programmiersprachen-Anbindung

Datenbank-Anfragen (DQL)

- Formulierung von Anfragen durch SELECT-Statement
- Aufbau
  - SELECT *Attributliste*
  - FROM *Relationen*
  - WHERE *Selektionsbedingung*

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Finde die Namen aller Angestellten.

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Finde Namen und Gehalt der Angestellten in Abteilung K55.

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Liste die verschiedenen Vorgesetzten-Nummern auf.

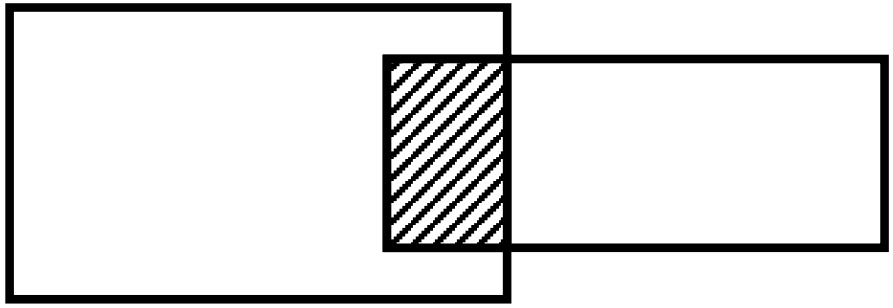
Operatoren in SELECT-Anfragen

- Logische Operatoren
  - AND — Und-Verknüpfung
  - OR — Oder-Verknüpfung
  - NOT — Verneinung
- Vergleichsoperatoren
  - =, !=, <, >, <=, >= — Vergleich
  - IN, NOT IN — Mengenvorgabe
  - BETWEEN x AND y — Intervallvorgabe
  - LIKE p — Maskierung
'%': beliebige Anzahl Zeichen
'_': ein einziges Zeichen
  - IS NULL — leerer Wert

Mengen-Operatoren in SELECT-Anfragen

- Mengen-Operatoren
  - UNION — Vereinigungsmenge
  - INTERSECT — Schnittmenge
  - MINUS — „OHNE“-Menge

A INTERSECT B



```
SELECT ...
FROM ...
WHERE...

INTERSECT

SELECT ...
FROM ...
WHERE...
```

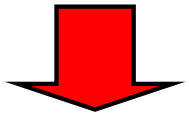
Abfragen über mehrere Tabellen

- Verknüpfung zweier Tabellen durch Auflistung der Tabellen in der FROM-Klausel
- Verbindung der Tabellen in der WHERE-Klausel
- Hinzufügen des Tabellennamens, falls Attributname nicht mehr eindeutig
- Beispiel: Ermittle alle Mitarbeiter, die in der Abteilung 'Personal' arbeiten
  - ```
    SELECT pers.name
    FROM pers, abt
    WHERE pers.anr = abt.anr
      AND abt.aname = 'Personal';
    ```
- Reihenfolge der aufgelisteten Tabellen
  - *Keine* Auswirkung auf Performance
  - Reihenfolge sollte sich an Lesbarkeit orientieren
    ```

Abfragen über mehrere Tabellen

Pers	pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
	406	Coy	1950	01.03.86	60.000	Kaufmann	K55	123
	123	Mueller	1958	01.09.80	48.000	Programmierer	K51	
	829	Schmidt	1960	01.06.90	54.000	Kaufmann	K53	123
	874	Abel		01.05.94	42.000	Softw.Entwickler	K55	829
	503	Junghans	1975		33.000	Programmierer	K51	123
	...	...	...	...	...	...	...	...

Abt	anr	aname	ort
	K51	Entwick	Erlangen
	K53	Buchh	Nürnberg
	K55	Personal	Nürnberg
	...	...	...



$A_{pers.anr = abt.anr}(pers, abt)$

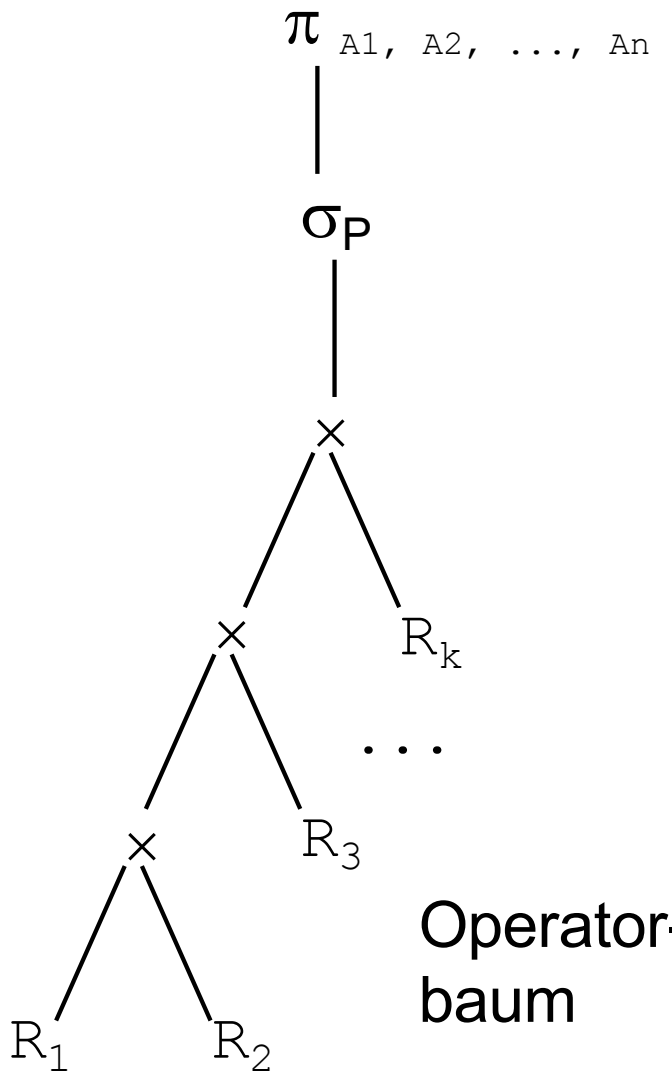
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr	anr	aname	ort
406	Coy	1950	01.03.86	60.000	Kaufmann	K55	123	K55	Personal	Nürnberg
123	Mueller	1958	01.09.80	48.000	Programmierer	K51		K51	Entwick	Erlangen
829	Schmidt	1960	01.06.90	54.000	Kaufmann	K53	123	K53	Buchh	Nürnberg
874	Abel		01.05.94	42.000	Softw.Entwickler	K55	829	K55	Personal	Nürnberg
503	Junghans	1975		33.000	Programmierer	K51	123	K51	Entwick	Erlangen
...	...	...	...	...	...	...	...	...	...	...

Übersetzung SQL-Anfragen in Relationenalgebra

```
SELECT A1, A2, ..., An
FROM R1, ..., Rk
WHERE P;
```

entspricht

$$\pi_{A1,\ldots,An}(\sigma_P(R_1 \times \ldots \times R_k))$$



Operator-baum

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Finde die Namen der Angestellten aus den Abteilungen K51, K54, K55.

Datenbank-Updates
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Versetze alle Angestellten aus den Abteilungen K52 bis K54 nach Abteilung K51, wenn sie vor 1974 geboren sind.

Semantische Datenintegrität
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Programmierer arbeiten in Erlangen.

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Finde die Namen der Angestellten und die Namen der Abteilungen, denen sie angehören.

Geschachtelte Anfragen (nested queries)

- Eine SELECT-Anfrage kann eingeschlossene SELECT-Anfragen enthalten
- Anwendung, wenn eine Anfrage vom Suchergebnis einer anderen Anfrage abhängt
- Beispiel für eine einfache Unterabfragen in der WHERE-Klausel
  - Ermittle alle Mitarbeiter, die mehr als Herr Mueller verdienen
- Probleme
  - Was passiert, wenn es mehrere Müller gibt?
  - Was passiert, wenn es keinen Müller gibt?

```
SELECT pers.name
FROM pers
WHERE gehalt > (SELECT gehalt FROM pers
                WHERE name = 'Mueller');
```

Geschachtelte Anfragen mit ALL, ANY

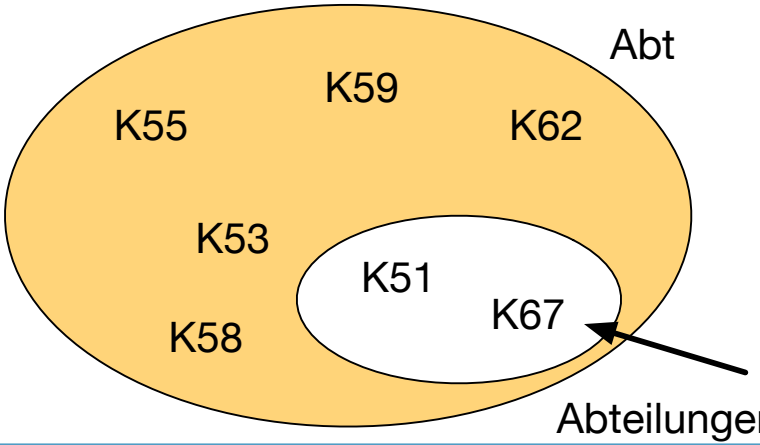
- Operator ALL bzw. ANY
  - Unterabfrage mit ALL bedeutet, dass Bedingung für alle Elemente der Menge gelten muss
  - Unterabfrage mit ANY bedeutet, dass Bedingung für mindestens ein Elemente der Menge gelten muss
- Beispiel: Ermittle alle Mitarbeiter, die mehr als alle Mitarbeiter der Abteilung 'K51' verdienen

```
SELECT name
FROM pers
WHERE gehalt > ALL (SELECT gehalt
                    FROM pers
                    WHERE anr = 'K51');
```

- Anmerkung: Viele Anfragen lassen sich auf verschiedene Weise formulieren (z.B. sowohl über Joins als Unterabfragen)

Geschachtelte Anfragen mit IN und NOT IN

- Operator IN bzw. NOT IN
  - IN überprüft, ob ein Wert in einer Untertabelle enthalten ist
  - NOT IN überprüft, ob ein Wert nicht in Untertabelle enthalten ist
- Beispiel: Ermittle alle Abteilungen, in denen keine Programmierer arbeiten
  - ```
    SELECT aname
    FROM abt
    WHERE anr NOT IN (SELECT anr
                      FROM pers
                      WHERE beruf = 'Programmierer');
    ```



Differenzmenge

Abteilungen mit Programmierer
    ```

Geschachtelte Anfragen mit EXISTS

- Operator EXISTS
  - Operator EXISTS überprüft, ob ein aktueller Wert einer Abfrage existiert
- Beispiel: Ermittle alle Mitarbeiter, die nicht das Maximum verdienen
  - ```
    SELECT pers.name
    FROM pers p1
    WHERE EXISTS (SELECT *
                  FROM pers p2
                  WHERE p1.gehalt < p2.gehalt);
    ```
- Verwendung von Aliase
  - Damit in oberer Anfrage das Gehalt der betrachteten Person in die Subquery übergeben werden kann, muss ein Alias definiert werden
  - Aliase werden in der FROM-Klausel definiert
    ```

Vermeidung korrelierter Unteranfragen

- Korrelierte Unterabfragen führen zu stark unperformanten Abfragen
- Beispiel zur Vermeidung korrelierter Unteranfragen
  - Anfrage: Welche Abteilungen sind nicht leer?

```
SELECT anr
FROM abt a
WHERE EXISTS (SELECT NULL FROM pers p
              WHERE a.anr = p.anr);
```

Korrelation zur Hauptanfrage

```
SELECT DISTINCT p.anr
FROM pers p
```

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Welche Abteilungen sind leer?

Unterabfragen in INSERT und UPDATE

- Unterabfragen in INSERT und UPDATE
  - Insert- und Update-Funktionen können ebenfalls Unterabfragen enthalten
  - Die Anweisung wird für alle ermittelten Werte durchgeführt
- Beispiel
  - Setze das Gehalt aller Mitarbeiter, die weniger verdienen als Herr Mueller auf das durchschnittliche Gehalt der Abteilung 'K53'

```
UPDATE pers
SET gehalt = (SELECT AVG(gehalt)
              FROM pers
              WHERE anr = 'K53')
WHERE gehalt < (SELECT gehalt
                FROM pers
                WHERE name = 'Mueller');
```

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Finde die Namen der Angestellten, die in Abteilungen in Erlangen arbeiten.

Datenbank-Updates
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Lösche alle Angestellten, die für Abteilungen in Erlangen arbeiten.

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Finde alle Angaben über diejenigen Angestellten, die im gleichen Jahr geboren sind und das gleiche Gehalt beziehen wie der Mitarbeiter Coy. Der Name Coy sei eindeutig.

Funktionsaufrufe in SELECT-Anfragen

- Funktionsaufrufe in SQL-Statements

```
- SELECT UPPER(name), LOWER(beruf)
  FROM pers;

- SELECT TO_CHAR(eindat, 'DD-MON-YY')
  FROM pers

- SELECT name, eindat
  FROM pers
  WHERE MONTHS_BETWEEN(SYSDATE, eindat) > 24
```

Funktionen in SELECT-Anfragen

- Aggregatfunktionen
  - `MIN(spaltenname)` Minimalwert
  - `MAX(spaltenname)` Maximalwert
  - `COUNT(*)` Anzahl der vorhandenen Zeilen
  - `COUNT(spname)` Anzahl der Zeilen ungleich NULL
  - `COUNT(DISTINCT spname)` Anzahl Zeilen mit untersch. Werten
  - `SUM(spaltenname)` Summe
  - `AVG(spaltenname)` Mittelwert

- Rechenoperationen
  - `+, -, *, /` Addition, Subtraktion, Multiplikation, Division

- Ausgabe-Reihenfolge
  - `ORDER BY spname ASC/DESC` Sortierung

Weitere wichtige Funktionen in SQL

- COALESCE
  - Ersatzwert für Null-Werte, verhindert damit die Ausgabe NULL
  - `COALESCE(gehalt, 0)`
  - `COALESCE(Beruf, 'unbekannt')`
  - Short Circuit:
    - Zweiter Parameter wird nur berechnet, wenn notwendig
- ROUND
  - Runden von Zahlen
  - Zweiter Parameter gibt Anzahl Nachkommastellen an
  - Beispiel Berechnung MwSt:
    - `ROUND(20*0.19, 2)`

Datum und Zeit in Oracle

- Komplexe Behandlung, da kulturspezifische Belange berücksichtigt werden müssen, Zeitzonen, etc.
- Datum und Datumsarithmetik in Oracle
  - DATE speichert Tag, Monat und Jahr, aber aus historischen Gründen auch Stunden, Minuten, Sekunden
  - Addition eines Datums mit 1 ergibt Datum mit dem nächsten Tag
  - Subtraktion eines Datums von einem anderen ergibt Anzahl Tage
  - SYSDATE liefert aktuelles Datum

    ```
    SELECT SYSDATE FROM DUAL;
    ```
- Timestamp
  - Für feinere Zeitintervalle als 1 Sekunde
  - `SELECT SysTimeStamp FROM DUAL;`
  - Weiterer Datentyp: TIMESTAMP WITH TIME ZONE

Datum und Zeit in Oracle
Formatierungsmöglichkeiten

- Ein Datum wird nie so ausgegeben, wie es intern gespeichert wird
  - Umwandlung in eine von Menschen lesbare Form
- Ein- und Ausgabe mit Hilfe von Umwandlungsfunktionen und Datumsformaten
  - `MM` Monat
    `MON` Dreistellige Angabe für Monatsname: AUG
    `DD` Tag
    `YYYY` Jahr
    `YY` Jahr zweistellig
    `HH` Stunden
    `MI` Minuten
    `SS` Sekunden

Datum und Zeit in Oracle
Formatierungsmöglichkeiten

- Umwandlungsfunktionen
  - Umwandlung Datum → String: TO_CHAR
  - Umwandlung String → Datum: TO_DATE
- TO_CHAR-Beispiel: Ausgabe der Uhrzeit
  - Format: `TO_CHAR(string[,'format'])`
  - `SELECT TO_CHAR(SysDate, 'HH:MI:SS') FROM DUAL`
  - Vorsicht: MM enthält Monat !
- TO_DATE-Beispiel: Umrechnung Benutzereingabe in Datum
  - Format: `TO_DATE(string[,'format'])`
  - Beispiel: `TO_DATE('11.05.2021', 'MM.DD.YYYY')`

Datum und Zeit in Oracle
Formatierungsmöglichkeiten

- Festlegen des Datumsformats für eine komplette Connection
  - `ALTER SESSION SET nls_date_format = 'DD.MM.YYYY';`
  - Vorsicht: Das Datumformat ist damit nur innerhalb einer Session festgelegt
  - Nach disconnect oder exit von sqldeveloper oder Neustart eines Programms muss der Befehl wiederholt werden

- Abfrage des Jahrs in einem Datum
  - `... WHERE TO_CHAR(eintrittsdatum, 'YY') = '19' ;`
  - Oder noch einfacher (ähnlich: day, month, hour, minute, second):
  - `... where extract(year from eindat) = 1980;`

Rechnen mit Datum Oracle

- Rechnen mit Anzahl Tagen
  - Addition/Subtraktion einer Anzahl Tagen zu einem Datum
  - `SELECT SYSDATE + 1 FROM DUAL;`
  - Berechnung der Anzahl Tage zwischen zwei Datumsangaben
  - `SELECT SYSDATE - EINDAT FROM PERS;`
- Berechnung Zeitintervallen (INTERVAL)
  - `SELECT MONTHS_BETWEEN(SYSDATE, EINDAT) FROM PERS;`
  - `SELECT SYSDATE + INTERVAL '1' YEAR FROM DUAL;`
  - `SELECT SYSDATE + INTERVAL '2' MONTH FROM DUAL;`
  - Für zahlreiche weitere Funktionen siehe Handbuch

Rechnen mit Datum Oracle

- Rundungsfunktionen
  - Date beinhaltet auch die Zeit
  - Zwei Datumsangaben von aufeinanderfolgenden Tagen könnten sich daher um nur eine Minute unterscheiden
  - „Abschneiden“ der Zeit
  - `SELECT trunc(SYSDATE) FROM DUAL;`

String-Funktionen in Oracle

Eine Auswahl

||	verkettet zwei Strings
ASCII	liefert für den ersten Buchstaben des Strings die dezimale Darstellung in Datenbankzeichensatz
CONCAT	verkettet zwei Strings (genau wie ||)
INSTR	findet den Standort eines Zeichens in einem String
LENGTH	gibt die Länge des Strings aus
LOWER	wandelt alle Buchstaben in einem String in Kleinbuchstaben aus
LPAD	bringt einen String auf eine best. Länge, indem auf der linken Seite eine Zeichenfolge hinzugefügt wird
LTRIM	schneidet auf der linken Seite eines Strings Zeichen ab
RPAD	bringt einen String auf eine best. Länge, indem auf der rechten Seite eine Zeichenfolge hinzugefügt wird
RTRIM	schneidet auf der rechten Seite eines Strings Zeichen ab
SUBSTR	schneidet aus einer Zeichenkette einen bestimmten Ausschnitt aus
TRIM	schneidet alle Vorkommen eines oder mehrerer Zeichen auf der linken und rechten Seite eines Strings ab
UPPER	wandelt alle Buchstaben in einer Zeichenkette in Großbuchstaben um

Numerische Funktionen in Oracle

Eine Auswahl

ABS(value)	Absoluter Wert
CEIL(value)	kleinste Integer, der größer oder gleich value ist
EXP(value)	Exponent e von value
FLOOR(value)	Größter Integer, der kleiner oder gleich value ist
LN(value)	natürlicher Logarithmus von value
LOG(value)	Logarithmus von value zur Basis 10
MOD(value, divisor)	Modulo
COALESCE(value, substitute)	Ersatz für value, wenn value gleich NULL ist
POWER(value, exp)	Exponentialfunktion
ROUND(value, prec)	Rundung von value auf prec (Nachkommastellen)
SIGN	1 wenn value positiv, -1 wenn value negativ, 0 wenn value NULL ist
AVG(value)	Durchschnittsfunktion
COUNT(value)	Anzahl Zeilen in einer Spalte
MAX(value)	Maximum
MIN(value)	Minimum
SUM(value)	Summe aller Werte
VARIANCE(value)	Varianz aller Werte

Datums-Funktionen in Oracle

Eine Auswahl

ADD_MONTHS	addiert eine Anzahl Monate zu einem Datum
CURRENT_DATE	liefert das aktuelle Datum für die Zeitzone der Sitzung
CURRENT_TIMESTAMP	liefert den aktuellen Zeitstempel mit der aktiven Zeitzoneninformation
EXTRACT(timeunit FROM datetime)	extrahiert einen Teil eines Datums aus einem Datumswert
GREATEST(date1, date2,...)	sucht aus einer Datumsliste das letzte (aktuellste) Datum heraus
LEAST(date1, date2, ...)	sucht aus einer Datumsliste das erste Datum heraus
LAST_DAY(date)	liefert das Datum des letzten Tags des Monats zurück, in den das Datum fällt
LOCALTIMESTAMP	liefert den lokalen Zeitstempel in der aktiven Zeitzone
MONTHS_BETWEEN(date2, date1)	gibt das Ergebnis von date2-date1 als Monatsangabe zurück
NEW_TIME(date,'this','other')	gibt das Datum und die Zeit in einer anderen Zeitzone zurück
ROUND(date, 'format')	rundet ein Datum
SYSTIMESTAMP	liefert das Systemdatum, inklusive der Sekundenbruchteile und Zeitzone
SYSDATE	liefert das aktuelle Datum und die aktuelle Zeit
TO_CHAR(date,'format')	formatiert das Datum nach den Vorgaben in format

Konvertierungs-Funktionen in Oracle

Eine Auswahl

BIN_TO_NUM	konvertiert einen Binärwert in sein numerisches Äquivalent
CAST	Castet einen Typ in einen anderen
CONVERT	übersetzt eine Zeichenkette von einem nationalem Zeichensatz in einen anderen
DECODE	decodiert Datentyp CHAR, VARCHAR2 oder NUMBER auf Basis der Werte verschiedener Zeichenketten
TO_CHAR	konvertiert den Datentyp NUMBER oder DATE, so dass er sich wie ein String verhält
TO_DATE	konvertiert die Datentypen NUMBER, CHAR oder VARCHAR2, so dass sie sich wie ein Datum verhalten
TO_NUMBER	konvertiert den Datentyp CHAR oder VARCHAR2, so dass er sich wie eine Zahl verhält
TRANSLATE	übersetzt die Zeichen in einer Zeichenkette in verschiedene Zeichen

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Finde die Namen der Angestellten, die mehr verdienen als alle Mitarbeiter in Abteilung 'K55'.

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Finde die Namen der Angestellten, die mehr als ihre direkten Vorgesetzten verdienen. Gib auch die zugehörigen Namen der Vorgesetzten an.

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Finde die Namen aller Chefs.

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Liste alle Angestellten auf, deren Name mit 'A' beginnen.

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Liste die Angestellten aus Abteilung K55 nach ihren Gehältern aufsteigend und ihren Namen absteigend sortiert auf.

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

a) Liste alle Personen auf, die mehr als der Durchschnitt in der Firma verdienen.

b) Liste alle Personen auf, die mehr als der Durchschnitt in ihrer jeweiligen Abteilung verdienen.

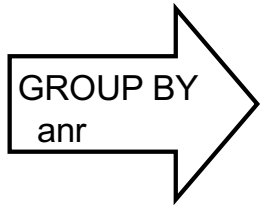
Gruppenverarbeitung

- Gruppenverarbeiten (GROUP BY)
  - Zusammenfassen von Tupeln mit gleicher Eigenschaft
  - Beispiel: Welche Gehälter haben die Abteilungen zu zahlen?

```
- SELECT anr, SUM(gehalt) as Summe
  FROM Pers
  GROUP BY anr;
```

Pers

pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...



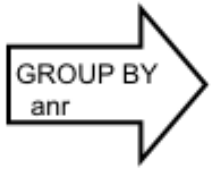
anr	Summe
K51	123.000
K53	74.000
K55	142.000
...	...

Erlaubte Anfragen mit GROUP BY

- In SELECT-Klausel erlaubte Elemente
  - Konstante oder Funktion ohne Parameter (z.B. SysDate)
  - Gruppenfunktion, wie SUM, AVG, MIN, MAX, COUNT
  - Ein mit einem Ausdruck in der GROUP BY-Klausel übereinstimmender Wert
- Im Beispiel nicht erlaubt
  - ```
    SELECT anr, name, SUM(gehalt) as Summe
    FROM Pers
    GROUP BY anr;
    ```

Pers

pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...



anr	Summe
K51	123.000
K53	74.000
K55	142.000
...	...
    ```

Gruppenverarbeitung
Auswahl von Gruppen

- Auswahl von Gruppen (HAVING)
  - Definition von Bedingungen, die Gruppen erfüllen sollen
  - Beisp.: Welche Abteilungen zahlen mehr als 100.000 Euro Gehalt?
  -
```
SELECT anr, SUM(gehalt) as Summe
FROM Pers
GROUP BY anr
HAVING SUM(gehalt) > 100000;
```

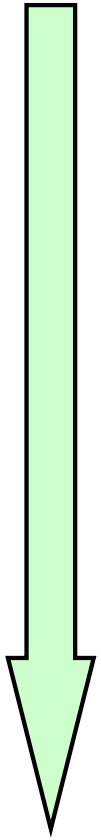
Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

GROUP BY
HAVING

anr	Summe
K51	123.000
K55	142.000
...	...

Auswertungsreihenfolge SELECT

FROM	Berechnung Ausgangstabelle
WHERE	Selektion von Tupel
GROUP BY	Gruppierung
HAVING	Selektion von Gruppen
SELECT	Projektion, Berechnung Aggregatsfunktionen
ORDER BY	Sortierung der Tupel



SELECT Syntax
Übersicht

- Syntax einer SELECT-Anweisung

```
Select-expression ::=
     SELECT [ALL|DISTINCT] list-of-select-items
     FROM list-of-table-references
     [WHERE condition]
     [GROUP BY expr [, expr ]...
         [HAVING condition]
     [ORDER BY expr [ASC, DESC...];
```

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Liste alle Abteilungsnummern und das Durchschnittsgehalt ihrer Angestellten auf (Monatsgehalt).

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Gruppiere die Angestellten nach Abteilungen und innerhalb der Abteilungen nach Berufen. Gib die Anzahl der Mitarbeiter und das durchschnittliche Monatsgehalt in jeder Gruppe an.

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Liste alle Abteilungsnummern auf, die mindestens zwei Programmierer beschäftigen.

Datenbank-Anfragen
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Wie viele verschiedene Berufe gibt es in Abteilung K55.

NULL-Werte

- Nullwerte
  - NULL kann für Attribute beliebigen Datentyps auftreten
  - NULL ist nicht identisch mit 0 oder ' '
- Mögliche Bedeutungen von Nullwerten
  - Unbekannter Wert
  - Nicht existierender Wert
  - Bedeutungsloser Wert
- Anfrage mit unklarer Bedeutung
  - ```
    SELECT COUNT(*)
    FROM pers
    WHERE vnr IS NULL;
    ```
    ```

Berechnungen mit NULL-Werten

- Arithmetische Ausdrücke (z.B. Addition, Multiplikation)
  - Resultat ist NULL, wenn eine der Operanden NULL ist
  - Beispiele: A + NULL = NULL, 0 * NULL = NULL
- Aggregatfunktionen (z.B. SUM, AVG)
  - Aggregatfunktionen ignorieren NULL in ihren Argumenten
- Vergleichsoperationen
  - Resultat ist NULL, wenn eine der Operanden NULL ist
  - Beispiele: A > NULL → NULL
- Logische Ausdrücke ("dreiwertige Logik")

OR	true	NULL	false
true	true	true	true
NULL	true	NULL	NULL
false	true	NULL	false

AND	true	NULL	false
true	true	NULL	false
NULL	NULL	NULL	false
false	false	false	false

NOT	
true	false
NULL	NULL
false	true

SELECT-Anfragen mit NULL-Werten

- Abfrage von Nullwerten
  - `IS NULL`: Suche nach Elementen mit Nullwerten
  - `IS NOT NULL`: Suche nach Elementen, bei denen irgendwelche Daten (ungleich NULL) vorhanden sind
- In WHERE-Bedingung werden nur Tupel weitergereicht, die true sind
- In Gruppierungen wird NULL als eigener Wert erfasst und als eigene Gruppe gebildet
  - Resultat ist NULL, wenn eine der Operanden NULL ist
  - Gruppierung ohne NULL:
  - ```
    SELECT jahrg, AVG(gehalt)
    FROM pers
    WHERE jahrg IS NOT NULL
    GROUP BY jahrg;
    ```
    ```

SELECT-Anfragen mit NULL-Werten

- VORSICHT: SELECT-Anfragen können in Verbindung mit Nullwerten unerwartete Ergebnisse liefern !
  - "You can never trust the answers you get from a database with nulls" (C.J. Date: SQL and Relational Theory)

```
SELECT count(*)
FROM Pers
WHERE gehalt >  100000
   OR gehalt <= 100000;
```

```
SELECT SUM(gehalt)
FROM Pers;
```

- Mögliche Maßnahmen
  - Möglichst viele Attribute mit NOT NULL definieren
  - Outer-Joins vorsichtig anwenden

Datenbank-Anfragen
Wiederholung

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

1: Liste alle Personen auf, die zwischen 1960 und 1970 geboren sind
2: Ermittle die Menge an Gehälter, die Mitarbeiter in Nürnberg beziehen
3: An welchen Orten sind Programmierer beschäftigt? An welchen Orten keine?
4: Erhöhe das Gehalt von allen Programmierer, die seit 30 Jahren in der Firma sind, um 10%
5: Liste alle Personen auf, die einen Chef haben, der jünger ist als sie selbst
6: Wie viel Mitarbeiter hat Herr Mueller?
7: Stelle alle Abteilungen zusammen, die mehr als zehn Programmierer haben.

Datenbank-Anfragen
Wiederholung

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

```
-- Welche Mitarbeiter sind aus Nürnberg oder Erlangen?
SELECT name FROM pers, abt
WHERE pers.anr = abt.anr
AND ort = 'Nuernberg'  OR  ort = 'Erlangen';
```

Ergebnis:

	NAME
1	Mueller
2	Coy
3	Schmidt
4	Abel
5	Junghans
6	Schmidt
7	Coy
8	Abel

Was ist denn hier falsch???