

Kapitel 4: Der SQL-Standard

Ein *Anfrageausdruck* in SQL besteht aus einer SELECT-Klausel, gefolgt von einer FROM-Klausel, gefolgt von einer WHERE-Klausel.

Grundform eines SFW-Ausdruck

```
SELECT  $A_1, \dots, A_n$  (...Attribute der Ergebnisrelation)
FROM  $R_1, \dots, R_m$  (...benötigte Relationen)
WHERE  $F$  (...Auswahlbedingung)
```

äquivalenter Algebraausdruck

$$\pi[A_1, \dots, A_n](\sigma[F](R_1 \times \dots \times R_m))$$

wobei der Einfachheit halber disjunkte Formate für R_1, \dots, R_m angenommen sind.

äquivalenter Kalkülausdruck

$$\{(X_{i_1} : A_1, \dots, X_{i_n} : A_n) \mid \exists \vec{Y} R_1(\vec{X}_1) \wedge \dots \wedge R_m(\vec{X}_m) \wedge F\}$$

X_{i_1}, \dots, X_{i_n} sind gerade alle Variablen, die in $\vec{X}_1, \dots, \vec{X}_m$ auftreten und nicht in \vec{Y} .

SFW-Ausdruck im Allgemeinen

```
SELECT  $A_1, \dots, A_n$  (...Attribute der Ergebnisrelation, bzw.
    beliebige attributwertige SQL-Ausdrücke)
FROM  $R_1, \dots, R_m$  (...benötigte Relationen, bzw.
    beliebige relationenwertige SQL-Ausdrücke)
WHERE  $F$  (...Auswahlbedingung, bzw.
    beliebige boolesche SQL-Ausdrücke)
```

Mondial relationale Datenbank Teil 1

Land				Provinz		
<u>LName</u>	<u>LCode</u>	HStadt	Fläche	<u>PName</u>	<u>LCode</u>	Fläche
Austria	A	Vienna	84	Baden	D	15
Egypt	ET	Cairo	1001	Bavaria	D	70,5
France	F	Paris	547	Berlin	D	0,9
Germany	D	Berlin	357	Ile de France	F	12
Italy	I	Rome	301	Franken	D	null
Russia	RU	Moscow	17075	Lazio	I	17
Switzerland	CH	Bern	41			
Turkey	TR	Ankara	779			

Stadt					
<u>SName</u>	<u>LCode</u>	<u>PName</u>	Einwohner	LGrad	BGrad
Berlin	D	Berlin	3472	13,2	52,45
Freiburg	D	Baden	198	7,51	47,59
Karlsruhe	D	Baden	277	8,24	49,03
Munich	D	Bavaria	1244	11,56	48,15
Nuremberg	D	Franken	495	11,04	49,27
Paris	F	Ile de France	2125	2,48	48,81
Rome	I	Lazio	2546	12,6	41,8

Mondial relationale Datenbank Teil 2

Lage			Mitglied		
<u>LCode</u>	<u>Kontinent</u>	Prozent	<u>LCode</u>	<u>Organisation</u>	Art
D	Europe	100	A	EU	member
F	Europe	100	D	EU	member
TR	Asia	68	D	WEU	member
TR	Europe	32	ET	UN	member
ET	Africa	90	I	EU	member
ET	Asia	10	I	NAM	guest
RU	Asia	80	TR	UN	member
RU	Europe	20	TR	CERN	observer

4.1 Basis-Datentypen und Built-in Functions

Basis-Datentypen

SQL bietet eine Fülle von unterschiedlichen Datentypen an, mittels derer die Wertebereiche der Spalten einer Tabelle festgelegt werden können.

- ▶ INTEGER, SMALLINT
- ▶ NUMERIC, DECIMAL. Angabe Anzahl Ziffern insgesamt und Anzahl Kommastellen.
- ▶ REAL, DOUBLE PRECISION, FLOAT
- ▶ CHARACTER, CHARACTER VARYING, CHARACTER LARGE OBJECT
- ▶ BIT, BIT VARYING, BINARY LARGE OBJECT
- ▶ BOOLEAN
- ▶ DATE, TIME, TIMESTAMP, INTERVALL,

Built-in Functions

Zur Manipulation der Werte der einzelnen Datentypen können spezielle Funktionen verwendet werden. Während der SQL-Standard hier sehr sparsam ist, bieten die einzelnen Hersteller von Datenbanksystemen einen großen Vorrat an. Die folgenden Beispiele basieren auf *Oracle Database SQL Reference 10g*. Es handelt sich um eine Auswahl sowohl der Funktionstypen als auch der jeweiligen Funktionen des jeweiligen Typs.

- ▶ Numeric:
ABS, ACOS, ASIN, ATAN, ATAN2, BITAND, CEIL, COS, COSH, EXP, FLOOR, LN, LOG, MOD, POWER, REMAINDER, ROUND, SIGN, SIN, SINH, SQRT, TAN, TANH, TRUNC
- ▶ Character:
CHR, CONCAT, INITCAP, LOWER, LPAD, LTRIM, REGEXP_REPLACE, REGEXP_SUBSTR, REPLACE, RPAD, RTRIM, SUBSTR, TRANSLATE, TREAT, TRIM, UPPER
ASCII, INSTR, LENGTH, REGEXP_INSTR
- ▶ Aggregate:
AVG, COLLECT, CORR, COUNT, FIRST, LAST, MAX, MEDIAN, MIN, RANK, REGRESSION, STATS_BINOMIAL_TEST, STDDEV, SUM, VARIANCE

CAST und CASE

- ▶ CAST erlaubt eine *explizite* Typkonversionen zwischen unterschiedlichen Typen;
- ▶ CASE beschränkt die Konversionen im Wesentlichen auf Wertumwandlungen innerhalb eines Typs.

Erstelle eine Tabelle mit Spalten LName und LCode, die zu jedem Land den Namen auf die ersten zwei Zeichen reduziert und anstatt 'D' den Wert 'BRD' von LCode verwendet. Beide Werte sollen konkateniert werden wobei getrennt durch '>'.

```
SELECT CONCAT(CONCAT(CAST(LName AS VARCHAR(2)), '>'),
             CASE LCode
               WHEN 'D' THEN 'BRD'
               ELSE LCode
             END)
FROM Land
```

4.2 Nullwerte

Die Problematik

- ▶ Liegt zu einem Attribut kein Wert vor, so kann dies durch Verwendung des *Nullwerts* null ausgedrückt werden.
- ▶ Als mögliche Interpretationen eines Nullwertes können wir unterscheiden: *Wert existiert, jedoch zur Zeit unbekannt* - *Wert existiert erst in der Zukunft* - *Wert prinzipiell unbekannt* - oder auch *Attribut nicht anwendbar*.

Beispiel

Student					
MatrNr	Name	Adresse	Semester	Exmatrikulationsdatum	Mutterschutz
1223	Hans Eifrig	null	2	null	null
3434	Lisa Lustig	Bergstraße 11	4	null	ja
1234	Maria Gut	Am Bächle 1	null	null	nein

Nullwerte und SQL

- ▶ SQL bietet die Prädikate `IS NULL` und `IS NOT NULL` an, um auf Existenz von Nullwerten prüfen zu können.
- ▶ In Ausdrücken der Form `A+B`, `A+1`, etc. ist das Resultat `null`, wenn einer der Operanden `null` ist.
- ▶ Ausdrücke mit Vergleichsoperatoren der Form `A=B`, `A<>B`, `A<B`, etc. haben den Wahrheitswert `UNKNOWN`, wenn mindestens einer der beteiligten Operanden den Wert `null` besitzt.
- ▶ SQL liegt eine dreiwertige Logik zugrunde. (`t=TRUE`, `f=FALSE`, `u=UNKNOWN`).

Bestimme alle Provinzen, zu denen die Fläche bekannt ist.

```
SELECT * FROM Provinz
WHERE Fläche IS NOT NULL
```

Wahrheitswerte

AND	t	u	f	OR	t	u	f	NOT	
t	t	u	f	t	t	t	t	t	f
u	u	u	f	u	t	u	u	u	u
f	f	f	f	f	t	u	f	f	t

Welche Ausgabe liefert:

```
SELECT * FROM Provinz
WHERE NOT (Fläche > 0)
```

Vermeide Nullwerte wann immer es geht!

... denn sie verkomplizieren die Anfrageformulierung und ihre Semantik bzgl. der realen Welt ist nicht eindeutig.

.... eine sinnvolle Anwendung von Null-Werten: *äußerer Verbund* (engl. outer join).

Land				Stadt					
LName	LCode	HStadt	Fläche	SName	LCode	PName	Einwohner	LGrad	BGrad
Austria	A	Vienna	84	Berlin	D	Berlin	3472	13,2	52,45
Egypt	ET	Cairo	1001	Freiburg	D	Baden	198	7,51	47,59
France	F	Paris	547	Karlsruhe	D	Baden	277	8,24	49,03
Germany	D	Berlin	357	Munich	D	Bavaria	1244	11,56	45,15
Italy	I	Rome	301	Nuremberg	D	Franken	495	11,04	49,27
Russia	RU	Moscow	17075	Paris	F	Ile de France	2125	2,48	48,81
Switzerland	CH	Bern	41	Rome	I	Lazio	2546	12,6	41,8
Turkey	TR	Ankara	779						

Wieviel Einwohner haben die Hauptstädte der einzelnen Länder?

```
SELECT L.LName AS Land, L.HStadt, S.Einwohner
FROM Land L LEFT OUTER JOIN Stadt S
ON L.HStadt = S.SName
```

LName	HStadt	Einwohner
Austria	Vienna	null
Egypt	Cairo	null
France	Paris	2125
Germany	Berlin	3472
Italy	Rome	2546
Russia	Moscow	null
Switzerland	Bern	null
Turkey	Ankara	null

- ▶ Bei einem Left OUTER JOIN bleiben die Tupel der linken Relation erhalten; rechts werden bei fehlenden Verbundpartnern Nullwerte ergänzt,
- ▶ bei einem RIGHT OUTER JOIN werden analog gegebenenfalls links Nullwerte ergänzt,
- ▶ und ein FULL OUTER JOIN berechnet die Vereinigung des entsprechenden LEFT OUTER JOIN und RIGHT OUTER JOIN.

4.3 Anfragen mit Aggregierungsfunktionen

COUNT, MIN, MAX, SUM und AVG

Wieviele Länder gibt es in der Tabelle Land, wie groß ist die maximale, die minimale Fläche und die durchschnittliche Fläche aller Länder?

```
SELECT COUNT(LCode), MAX(Fläche), MIN(Fläche), AVG(Fläche)
FROM Land
```

Wieviele Länder haben eine Mitgliedschaft bzgl. der EU?

```
SELECT COUNT(*) AS AnzEU
FROM Mitglied
WHERE Organisation = 'EU'
```

Wieviele unterschiedliche Organisationen werden in Mitglied aufgeführt?

```
SELECT COUNT(DISTINCT Organisation) FROM Mitglied
```

Besonderheiten

- ▶ COUNT(*) liefert die Anzahl Zeilen der Tabelle, die sich nach Auswerten der FROM- und WHERE-Klausel ergeben hat.
- ▶ `SELECT LName, MAX(Fläche) FROM Land`
ist syntaktisch nicht zulässig, da ein Aggregierungsoperator eine Menge von Zeilen auf einen einzigen Wert reduziert. Zulässig wäre:
`SELECT MAX(Fläche) FROM Land`
- ▶ Aggregierungsfunktionen ignorieren für ihre Berechnungen Nullwerte.
- ▶ Eine Ausnahme ist COUNT(*); hier werden auch alle Zeilen, in denen alle Spalten null sind, mitgezählt.

4.4 Anfragen mit Gruppierungen

- ▶ Mittels einer *Gruppierung* können wir eine virtuelle Struktur über einer Tabelle definieren.
- ▶ Die Gruppierungsattribute fassen alle Zeilen der Tabelle jeweils zu einer Gruppe zusammen, die bezüglich aller Gruppierungsattribute gleiche Werte haben und zusätzlich die in einer optionalen HAVING-Klausel festgelegten Bedingungen erfüllt.
- ▶ Anfragen über einer gruppierten Tabelle betrachten die einzelnen Gruppen zusammen mit den Gruppierungsattributen analog zu einer Zeile einer Tabelle.

Konsequenterweise dürfen Attribute, die nicht als Gruppierungsattribute verwendet wurden, nur als Parameter für Aggregierungsfunktionen verwendet werden.

Wie groß ist die durchschnittliche Einwohnerzahl der Städte der jeweiligen Länder?

```
SELECT LCode, AVG(Einwohner) FROM Stadt
GROUP BY LCode
```

In welchen Ländern ist die durchschnittliche Einwohnerzahl kleiner 2 Mio.?

```
SELECT LCode, AVG(Einwohner) FROM Stadt
GROUP BY LCode
HAVING AVG(Einwohner) < 2000
```


Bestimme die Einwohnerzahlen der drei größten Städte.

```
SELECT DISTINCT COUNT(*) AS Rang, A.Einwohner
FROM Stadt A, Stadt B
WHERE (A.Einwohner <= B.Einwohner)
GROUP BY A.Einwohner
HAVING COUNT(*) <= 3
ORDER BY Rang
```

Bestimme die Einwohnerzahlen der drei größten Städte.

```
SELECT DISTINCT COUNT(*) AS Rang, A.Einwohner
FROM Stadt A, Stadt B
WHERE (A.Einwohner <= B.Einwohner)
GROUP BY A.Einwohner
HAVING COUNT(*) <= 3
ORDER BY Rang
```

Bestimme die Einwohnerzahlen der drei größten Städte.

```
SELECT MAX(A.Einwohner) AS Rang1,
       MAX(B.Einwohner) AS Rang2,
       MAX(C.Einwohner) AS Rang3
FROM Stadt A, Stadt B, Stadt C
WHERE (A.Einwohner > B.Einwohner)
      AND (B.Einwohner > C.Einwohner)
```

... oder direkt mit fortgeschrittenem SQL unter Verwendung der Funktion ROW_NUMBER():¹

```
SELECT myNUMBER, Einwohner FROM
( SELECT
  ROW_NUMBER() OVER (ORDER BY Einwohner DESC) AS myNUMBER,
  Einwohner
  FROM Stadt
) T
WHERE myNUMBER < 4;
```

Je nach Hersteller existieren weitere nicht standardkonforme Lösungen.

¹Nur als Hinweis, nicht klausurrelevant!

SFW-Ausdruck

SELECT A_1, \dots, A_n	Liste der Attribute
FROM R_1, \dots, R_m	Liste der Relationen
WHERE F	Bedingung
GROUP BY B_1, \dots, B_k	Liste der Gruppierungsattribute
HAVING G	Gruppierungsbedingung
ORDER BY H	Sortierordnung

Für die Auswertungsreihenfolge gilt: FROM-Klausel vor WHERE-Klausel vor GROUP-Klausel vor HAVING-Klausel vor ORDER-Klausel vor SELECT-Klausel.