Kapitel 3: Einstieg in SQL

- SQL (Structured Query Language) ist die in der Praxis am weitesten verbreitete Datenbanksprache für relationale Datenbanken.
- Die Historie von SQL geht zurück bis 1974, die Anfangszeit der Entwicklung relationaler Datenbanken.
- Alles begann mit SEQUEL, der *Structured English Query Language*.
- Der Sprachumfang von SQL ist einer permanenten Weiterentwicklung und Standardisierung unterworfen. Derzeit relevant sind der Stand von 1992, 1999, 2003, 2008, 2011 und 2016 entsprechend bezeichnet mit SQL-92, SQL:1999, SQL:2003, SQL:2008, SQL:2011 und SQL:2016.

3.1 Beispiels-Datenbank

Mondial-Datenbank Teil 1

Land

LName	LCode	HStadt	Fläche
Austria	A	Vienna	84
Egypt	ET	Cairo	1001
France	F	Paris	547
Germany	D	Berlin	357
Italy	I	Rome	301
Russia	RU	Moscow	17075
Switzerland	CH	Bern	41
Turkey	TR	Ankara	779

Provinz

PName	LCode	Fläche
Baden	D	15
Bavaria	D	70,5
Berlin	D	0,9
Ile de France	F	12
Franken	D	null
Lazio	I	17

Stadt

SName	LCode	PName	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-Datenbank Teil 2

Lage

LCode	Kontinent	Prozent
D	Europe	100
F	Europe	100
TR	Asia	68
TR	Europe	32
ET	Africa	90
ET	Asia	10
RU	Asia	80
RU	Europe	20

Mitglied

LCode	Organisation	Art
A	EU	member
D	EU	member
D	WEU	member
ET	UN	member
I	EU	member
I	NAM	guest
TR	UN	member
TR	CERN	observer

3.2 Einfache Anfragen

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

SFW-Ausdruck

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

Anfragen über einer Relation

(a) *gesamter Inhalt*

Gib den vollständigen Inhalt der Tabelle Stadt.

```
SELECT * FROM Stadt
```

Stadt

SName	LCode	PName	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

(b) *einzelne Spalten* (Projektion)

In welchen Provinzen liegen die Städte der Tabelle Stadt?

```
SELECT PName FROM Stadt
```

PName
Berlin
Baden
Baden
Bavaria
Franken
Ile de France
Lazio

In welchen *unterschiedlichen* Provinzen liegen die Städte der Tabelle Stadt?

```
SELECT DISTINCT PName FROM Stadt
```

PName
Berlin
Baden
Bavaria
Franken
Ile de France
Lazio

(c) *einzelne Zeilen* (Selektion)

Stadt

SName	LCode	PName	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

Wie heißen die Städte, die mehr als 1 Mio. Einwohner haben?

```
SELECT * FROM Stadt
   WHERE Einwohner > 1000
```

Stadt

SName	LCode	PName	Einwohner	LGrad	BGrad
Berlin	D	Berlin	3472	13,2	52,45
Munich	D	Bavaria	1244	11,56	48,15
Paris	F	Ile de France	2125	2,48	48,81
Rome	I	Lazio	2546	12,6	41,8

(d) *geänderte Spaltenbezeichnungen und neue Spalten*

Stadt

SName	LCode	PName	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

Kennzeichne die Tatsache, dass eine Stadt mehr als 1 Mio. Einwohner hat, durch den Wert `'Großstadt'` einer neuen Spalte mit Namen `StadtKategorie`; die Spalte `SName` soll die Bezeichnung `Stadt` erhalten.

```
SELECT SName AS Stadt, 'Großstadt' AS StadtKategorie FROM Stadt
   WHERE Einwohner > 1000
```

Stadt	StadtKategorie
Berlin	Großstadt
Munich	Großstadt
Paris	Großstadt
Rome	Großstadt

(e) *Pattern Matching* (1)

Land

LName	LCode	HStadt	Fläche
Austria	A	Vienna	84
Egypt	ET	Cairo	1001
France	F	Paris	547
Germany	D	Berlin	357
Italy	I	Rome	301
Russia	RU	Moscow	17075
Switzerland	CH	Bern	41
Turkey	TR	Ankara	779

Erstelle eine Namensliste der Länder, deren Namen mit 'G' anfängt oder mit 'y' aufhört?

```
SELECT LName FROM Land WHERE LName LIKE 'G%' OR LName LIKE '%y'
```

LName
Germany
Italy
Turkey

(e) *Pattern Matching* (2)

Land

LName	LCode	HStadt	Fläche
Austria	A	Vienna	84
Egypt	ET	Cairo	1001
France	F	Paris	547
Germany	D	Berlin	357
Italy	I	Rome	301
Russia	RU	Moscow	17075
Switzerland	CH	Bern	41
Turkey	TR	Ankara	779

Erstelle eine Namensliste der Länder, deren dritter Buchstabe des Namen 'y' ist?

```
SELECT LName FROM Land WHERE LName LIKE '__y%'
```

LName
Egypt

Anfragen über mehreren Relationen

Anfragen mit mehreren Relationen in der FROM-Klausel sind sogenannte *Verbund*-Anfragen (engl. join-queries).

(a) *einfacher Verbund*

Land

LName	LCode	HStadt	Fläche
Austria	A	Vienna	84
Egypt	ET	Cairo	1001
France	F	Paris	547
Germany	D	Berlin	357
Italy	I	Rome	301
Russia	RU	Moscow	17075
Switzerland	CH	Bern	41
Turkey	TR	Ankara	779

Stadt

SName	LCode	PName	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

$\Longrightarrow$

Land	Stadt
Germany	Berlin
Germany	Freiburg
Germany	Karlsruhe
Germany	Munich
Germany	Nuremberg
France	Paris
Italy	Rome

Gib zu jedem Land die zugehörigen Städte an.

```
SELECT Land.LName AS Land, Stadt.SName AS Stadt
    FROM Land, Stadt
    WHERE Land.LCode = Stadt.LCode
```

Was passiert, wenn man die Auswahlbedingung (WHERE) entfernt?

```
SELECT Land.LName AS Land, Stadt.SName AS Stadt
   FROM Land, Stadt
```

?

... man berechnet das kartesische Produkt der angegebenen Relationen!

(b) *Korrelationsnamen*

... Verwende optionale Korrelationsnamen.

```
SELECT S.SName, L.LName
   FROM Stadt S, Land L
   WHERE S.LCode = L.LCode
```

(c) *Verbund mit Auswahlbedingung*

Land

LName	LCode	HStadt	Fläche
Austria	A	Vienna	84
Egypt	ET	Cairo	1001
France	F	Paris	547
Germany	D	Berlin	357
Italy	I	Rome	301
Russia	RU	Moscow	17075
Switzerland	CH	Bern	41
Turkey	TR	Ankara	779

Stadt

SName	LCode	PName	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

Gib zu jedem Land die zugehörigen Städte mit mehr als 1 Mio Einwohner an.

```
SELECT L.LName AS Land, S.SName AS Stadt
   FROM Land L, Stadt S
   WHERE L.LCode = S.LCode AND S.Einwohner > 1000
```

Land	Stadt
Germany	Berlin
Germany	Munich
France	Paris
Italy	Rome

(d) *Verbund einer Relation mit sich selbst*

Lage

LCode	Kontinent	Prozent
D	Europe	100
F	Europe	100
TR	Asia	68
TR	Europe	32
ET	Africa	90
ET	Asia	10
RU	Asia	80
RU	Europe	20

Bestimme alle Paare von Ländern, die in einem gemeinsamen Kontinent liegen.

```
SELECT L1.LCode AS Land1, L2.LCode AS Land2
   FROM Lage L1, Lage L2
   WHERE L1.Kontinent = L2.Kontinent
   AND L1.LCode < L2.LCode
```

(1) FROM Lage L1, Lage L2

Lage L1 (8 Tupel)

LCode	Kontinent	Prozent
D	Europe	100
F	Europe	100
TR	Asia	68
TR	Europe	32
ET	Africa	90
ET	Asia	10
RU	Asia	80
RU	Europe	20

$\times$

Lage L2 (8 Tupel)

LCode	Kontinent	Prozent
D	Europe	100
F	Europe	100
TR	Asia	68
TR	Europe	32
ET	Africa	90
ET	Asia	10
RU	Asia	80
RU	Europe	20

$=$

$L_1 \times L_2$ (64 Tupel)

L1.LCode	L1.Kontinent	L1.Prozent	L2.LCode	L2.Kontinent	L2.Prozent
D	Europe	100	D	Europe	100
D	Europe	100	F	Europe	100
...	...	...	...	...	...
D	Europe	100	RU	Asia	80
...	...	...	...	...	...
RU	Asia	20	D	Europe	100
...	...	...	...	...	...
RU	Europe	20	RU	Europe	20

(2) `FROM Lage L1, Lage L2`
`WHERE L1.Kontinent = L2.Kontinent`

$L_1 \times L_2$ (**64** Tupel)

L1.LCode	L1.Kontinent	L1.Prozent	L2.LCode	L2.Kontinent	L2.Prozent
D	Europe	100	D	Europe	100
D	Europe	100	F	Europe	100
...	...	...	...	...	...
D	Europe	100	RU	Asia	80
...	...	...	...	...	...
RU	Asia	20	D	Europe	100
...	...	...	...	...	...
RU	Europe	20	RU	Europe	20

$\Downarrow$

(**26** Tupel)

L1.LCode	L1.Kontinent	L1.Prozent	L2.LCode	L2.Kontinent	L2.Prozent
D	Europe	100	D	Europe	100
D	Europe	100	F	Europe	100
...	...	...	...	...	...
D	Europe	100	RU	Europe	20
...	...	...	...	...	...
RU	Europe	20	D	Europe	100
...	...	...	...	...	...
RU	Europe	20	RU	Europe	20

(3)
```
FROM Lage L1, Lage L2
WHERE L1.Kontinent = L2.Kontinent
AND L1.LCode < L2.LCode
```

(**9** Tupel)

L1.LCode	L1.Kontinent	L1.Prozent	L2.LCode	L2.Kontinent	L2.Prozent
D	Europe	100	F	Europe	100
ET	Asia	10	TR	Asia	68
RU	Asia	80	TR	Asia	68
D	Europe	100	TR	Europe	32
F	Europe	100	TR	Europe	32
RU	Europe	20	TR	Europe	32
ET	Asia	10	RU	Asia	80
D	Europe	100	RU	Europe	20
F	Europe	100	RU	Europe	20

(4)
```
SELECT DISTINCT L1.LCode AS Land1,
                L2.LCode AS Land2
  FROM Lage L1, Lage L2
  WHERE L1.Kontinent = L2.Kontinent
  AND L1.LCode < L2.LCode
```

(8 Tupel)

Land1	Land2
D	F
ET	TR
RU	TR
D	TR
F	TR
ET	RU
D	RU
F	RU

(e) *impliziter und expliziter Verbund*

Implizit:

Gib zu jedem Land die zugehörigen Städte an.

```
SELECT S.SName, L.LName
   FROM Stadt S, Land L
   WHERE S.LCode = L.LCode
```

Explizit:

```
SELECT S.SName, L.LName
   FROM Stadt S JOIN Land L
   ON S.LCode = L.LCode
```

Wird Gleichheit über Attributen mit identischen Bezeichnern gefordert redet man von einem *natürlichen Verbund* (engl. natural join) und schreibt kürzer:

```
SELECT S.SName, L.LName
   FROM Stadt S NATURAL JOIN Land L
```

Spezialfall *kartesisches Produkt*:

```
SELECT S.SName, L.LName
   FROM Stadt S CROSS JOIN Land L
```

Auswertung einfacher SQL-Anfragen

SFW-Ausdruck

```
SELECT A1,...,An   (...Attribute der Ergebnisrelation)
   FROM R1,...,Rm  (...benötigte Relationen)
   WHERE F         (...Auswahlbedingung)
```

intuitive deklarative Semantik

Das Ergebnis besteht aus den Tupeln des kartesischen Produkts $R_1 \times ... \times R_m$, die die Bedingung der WHERE-Klausel F erfüllen, wobei nur die Werte der Attribute $A_1, ..., A_n$ ausgegeben werden.

Nested-Loop-Semantik

FOR each Tupel t_1 in Relation R_1 DO
  FOR each Tupel t_2 in Relation R_2 DO

$\vdots$

FOR each Tupel t_m in Relation R_m DO
  IF WHERE-Klausel ist erfüllt nach Ersetzen der Attributnamen in F
    durch die entsprechenden Werte der gerade betrachteten Tupel $t_1, \ldots, t_m$.
  THEN Bilde ein Antwort-Tupel aus den Werten der in der
    SELECT-Klausel angegebenen Attributen $A_1, \ldots, A_n$
    bezüglich der gerade betrachteten Tupel $t_1, \ldots, t_m$.

Sortierung

Stadt

SName	LCode	PName	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

Sortiere die Zeilen der Tabelle Stadt aufsteigend nach LCode und für gemeinsame Werte zu LCode absteigend nach dem Breitengrad.

```
SELECT * FROM Stadt
   ORDER BY LCode DESC, BGrad ASC
```

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

3.3 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 11g*. 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

Beispiele

- ```
  SELECT UPPER(LName)
  FROM Land
  ```
- ```
  SELECT CONCAT(LName,LCode) AS LNameAndLCode
  FROM Land
  ```
- ```
  SELECT CONCAT(CONCAT(LName, '>'),LCode) AS LNameAndLCode
  FROM Land
  ```
- ```
  SELECT LName || '>' || LCode) AS LNameAndLCode
  FROM Land
  ```
- ```
  SELECT Flaeche, LN(EXP(Flaeche)) AS Interessant
  FROM Land WHERE Flaeche < 100
  ```
- ```
  SELECT A.SName. B.SName,
   SQRT(power((A.BGrad-B.BGrad)*111.13,2)+
        power((A.LGrad-B.LGrad)*71.44,2)) AS Luli
  FROM Stadt A, Stadt B
  WHERE A.SName = 'Freiburg' AND B.SName = 'Karlsruhe'
  ```

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 einer Spalte LNameUndLCode. Zu jedem Land wird der Name auf die ersten zwei Zeichen reduziert und anstatt 'D' wird der Wert 'BRD' von LCode verwendet. Beide Werte sollen konkateniert werden getrennt durch '>'.

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

3.4 empfohlene Lektüre

SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

ABSTRACT: In this paper we present the data manipulation facility for a structured English query language (SEQUEL) which can be used for accessing data in an integrated relational data base. Without resorting to the concepts of bound variables and quantifiers SEQUEL identifies a set of simple operations on tabular structures, which can be shown to be of equivalent power to the first order predicate calculus. A SEQUEL user is presented with a consistent set of keyword English templates which reflect how people use tables to obtain information. Moreover, the SEQUEL user is able to compose these basic templates in a structured manner in order to form more complex queries. SEQUEL is intended as a data base sublanguage for both the professional programmer and the more infrequent data base user.

1

[1] In: Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control. Kann aus dem Institutsnetz heraus vom ACM-Portal heruntergeladan werden.