Einführung in Datenbanken

Kapitel 3: SQL: Geschichte und Lexikalische Syntax

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2018/19

http://www.informatik.uni-halle.de/~brass/db18/

Inhalt

1. SQL: Geschichte
2. Syntax-Graphen
3. Lexikalische Syntax

SQL: Bedeutung

- Heute ist SQL die einzige DB-Sprache für relationale DBMS (Industriestandard).

  Andere Sprachen wie QUEL sind ausgestorben. Die Sprache QBE hat zumindest einige graphische Schnittstellen zu Datenbanken inspiriert (z.B. in Access). Die Sprache Datalog hat SQL:1999 beeinflusst und wird noch in der Forschung studiert und weiterentwickelt, und hat eventuell eine Zukunft als Datenbank-Programmiersprache. Jedes kommerzielle RDBMS muss heute eine SQL-Schnittstelle haben. Es kann zusätzlich noch andere (z.B. graphische) Schnittstellen geben. Nicht alle Datenbanken sind relational und haben dann auch andere Anfragesprachen bzw. Programmier-Schnittstellen.

- SQL wird verwendet für:
  - Interaktive "Ad-hoc"-Befehle und
  - Anwendungsprogrammentwicklung (in andere Sprachen wie C, Java, HTML eingebettet).

Geschichte

- SEQUEL, eine frühere Version von SQL, wurde von Chamberlin, Boyce et al. bei IBM Research, San Jose (1974) entwickelt.

  SEQUEL steht für "Structured English Query Language" ("Anfragesprache in strukturiertem Englisch"). Manche Leute Sprechen SQL noch heute auf diese Art aus. Der Name wurde aus rechtlichen Gründen geändert (SEQUEL war bereits eine eingetragene Marke). Codd war auch in San Jose, als er das relationale Modell erfand.

- SQL war die Sprache des System/R (1976/77).

  System/R war ein sehr einflussreicher Forschungs-Prototyp.

- Ersten SQL-unterstützenden kommerziellen Systeme waren Oracle (1979) und IBM SQL/DS (1981).

Standards (1)

- Erster Standard 1986/87 (ANSI/ISO).

  Das war sehr spät, weil es schon viele SQL-Systeme auf dem Markt gab. Der Standard war der "kleinste gemeinsame Nenner". Er enthielt im wesentlichen nur den Schnitt der SQL-Implementierungen.

- Erweiterung für Fremdschüssel usw. '89 (SQL-89).

  Diese Version wird auch SQL-1 genannt. Alle kommerziellen Implementierungen unterstützen heute diesen Standard, aber jede hat viele Erweiterungen. Der Standard hatte 120 Seiten.

- Größere Erweiterung: SQL2 oder SQL-92 (1992).

  626 Seiten, aufwärts kompatibel zu SQL-1. Der Standard definiert drei Kompatibilitätsgrade: "Entry", "Intermediate", "Full" (+ später "Transitional" zwischen ersten beiden). Oracle 8.0 und SQL Server 7.0 sind "Entry Level"-kompatibel, höhere Konstrukte zum Teil.

Standards (2)

- Weitere große Erweiterung: SQL:1999.

  Anstelle von Levels definiert der Standard nun "Core SQL" und eine Vielzahl von Paketen von Sprach-Features.

- Wichtige neue Möglichkeiten in SQL:1999:
  - Nutzer-definierte Datentypen, strukturierte Typen.

    Man kann nun "distinct types" definieren, die Domains sehr ähnlich sind, aber ein Vergleich zwischen verschiedenen "distinct types" ist nun ein Fehler. Es gibt auch Typ-Konstruktoren "`ARRAY`" und "`ROW`" für strukturierte Attributwerte und "`REF`" für Zeiger auf Zeilen.
  - OO-Fähigkeiten, z.B. Vererbung/Untertabellen.
  - Rekursive Anfragen.
  - Trigger, persistent gespeicherte Module.

Standards (3)

- SQL:2003 führt XML- und OLAP-Unterstützung ein.

  Größte Neuerungen: Unterstützung von XML, Verwaltung von externen Daten. Außerdem gibt es nun Generatoren für eindeutige Nummern und einen "`MULTISET`" type constructor. Ansonsten ist es nur die 2. Auflage des SQL:1999-Standards. Auch die OLAP-Elemente (On-Line Analytical Processing), die als Zusatz zum SQL:1999-Standard veröffentlicht wurden, sind nun im SQL:2003-Standard integriert.

- 2006: Überarbeitung des XML-Teils.

Standards (4)

- Erweiterungen 2008 u.a.:
  - `TRUNCATE`
  - `ORDER BY` auch in Unteranfragen (für `FETCH FIRST`).
  - `INSTEAD OF` Trigger.
- Erweiterungen 2011 u.a.:
  - Zeitbezogene Daten (PERIOD FOR)
  - Erweiterungen der Window Functions.
- Erweiterungen 2016 u.a.:
  - JSON-Unterstützung
  - Row Pattern Matching.

Standards (5)

- Aktuelle Version des Standards ist SQL:2016.

ISO/IEC 9075-1:2016	Framework (SQL/Framework)
ISO/IEC 9075-2:2016	Foundation (SQL/Foundation)
ISO/IEC 9075-3:2016	Call-Level Interface (SQL/CLI)
ISO/IEC 9075-4:2016	Persistent Stored Modules (SQL/PSM)
ISO/IEC 9075-9:2016	Management of External Data (SQL/MED)
ISO/IEC 9075-10:2016	Object Language Bindings (SQL/OLB)
ISO/IEC 9075-11:2016	Information and Definition Schemas (SQL/Schemata)
ISO/IEC 9075-13:2016	SQL Routines and Types Using the Java TM Programming Language (SQL/JRT)
ISO/IEC 9075-14:2016	XML-Related Specifications (SQL/XML)

  Teil 1, 2, 11: minimale Anforderungen, andere Teile: Erweiterungen.

  Die Lücken in den Nummern der Bände sind nur historisch zu verstehen.

  Es gibt einen verwandten Standard ISO/IEC 13249:

  "SQL multimedia and application packages".

- Teil 2 definiert den Kern der Sprache und hat 1707 Seiten.

Inhalt

1 SQL: Geschichte

2 Syntax-Graphen

3 Lexikalische Syntax

Syntax-Formalismus

- In dieser Vorlesung wird die Syntax von SQL-Anfragen mit "Syntax-Graphen" definiert.

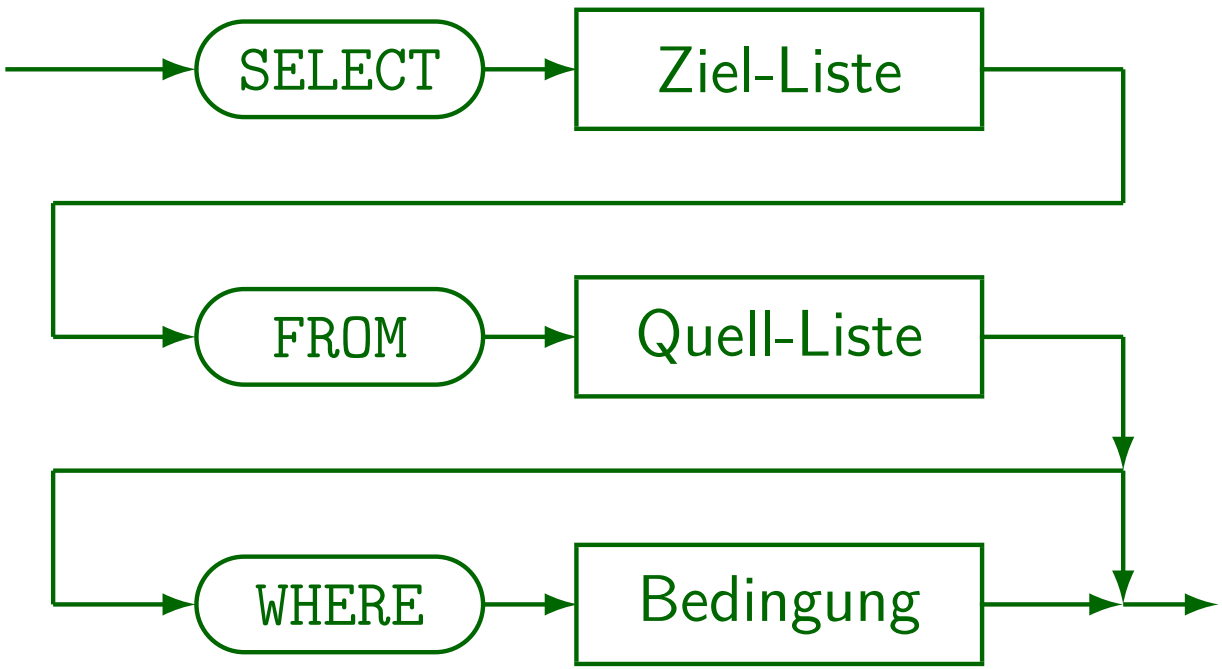
  Beispiel auf nächster Folie. Alternative zu kontextfreien Grammatiken (definiert dieselbe Klasse von Sprachen). In Anhang A genauer.

- Um eine Zeichenfolge syntaktisch richtig zu erstellen, muss man einen Pfad vom Start bis zum Ziel durch den Graphen verfolgen.

  Wörter in Ovalen werden direkt in die Ausgabe geschrieben, Kästen sind "Aufrufe" anderer Graphen: An diesem Punkt muss man einen Pfad durch den Graphen finden, dessen Name in dem Kasten steht. Anschließend kehrt man zu der Kante zurück, die den Kasten verlässt. Das Oracle-SQL-Referenz-Handbuch enthält auch Syntax-Graphen, aber dort ist die Bedeutung von Ovalen und Kästen umgekehrt.

Basis-Anfrage-Syntax (1)

SELECT-Ausdruck (vereinfacht):



Basis-Anfrage-Syntax (2)

- Jede SQL-Anfrage muss die Schlüsselwörter `SELECT` und `FROM` enthalten.

  Oracle stellt eine Relation "`DUAL`" zur Verfügung, die nur eine Zeile hat. Sie kann benutzt werden, wenn nur eine Berechnung ohne Zugriff auf die DB durchgeführt wird: "`SELECT TO_CHAR(SQRT(2)) FROM DUAL`" berechnet $\sqrt{2}$.

- In PostgreSQL, SQL Server, Access und MySQL kann die `FROM`-Klausel weggelassen werden, z.B. `SELECT 1+1`.

  In Oracle, DB2 und dem SQL-92-Standard ist dies ein Syntax-Fehler.

SQL-Syntax in der Vorlesung

- SQL:2016 ist zu groß, um es hier vollständig zu behandeln. Ein großer Teil des Standards ist auch noch nicht in derzeitigen DBMS implementiert.
- SQL/89 ($\sim$ Einstiegs-Level von SQL-92) wird vollständig behandelt und ein Teil von SQL:2016, der in fast allen DBMS implementiert ist.

  Manchmal werden Details der SQL-Syntax spezieller Systeme erklärt, meist im Kleingedruckten. Dies ist irrelevant für Klausuren. Sie sollen einen Eindruck von der Portabilität der Konstrukte geben (und helfen, wenn man mit diesem DBMS arbeiten muss). In Klausuren werden nur Punkte abgezogen, wenn man extrem nicht-portable Konstrukte verwendet (z.B. Anfragen, die nur in MySQL laufen).
- In der Vorlesung "Datenbank-Programmierung" werden weitere fortgeschrittenen SQL-Konstrukte behandelt.

Inhalt

1 SQL: Geschichte

2 Syntax-Graphen

3 Lexikalische Syntax

Lexikalische Syntax

- Die lexikalische Syntax einer Sprache definiert, wie Wortsymbole ("Token") aus einzelnen Zeichen zusammengesetzt werden.
- Z.B. definiert sie die genaue Syntax von
  - Bezeichnern (Namen für z.B. Tabellen, Spalten),
  - Literalen (Datentyp-Konstanten, z.B. Zahlen),
  - Schlüsselwörtern, Operatoren, Trennzeichen.
- Anschließend wird die Syntax von Anfragen u.s.w. basierend auf diesen Wortsymbolen definiert.

  D.h. eine Anfrage wird dann als Folge von Token definiert, nicht als Folge von Zeichen. Diese Zweiteilung bewirkt z.B., dass man auf der höheren Syntaxebene nicht mehr über eingestreuten Leerplatz nachdenken muss.

Leerzeichen und Kommentare

Leerplatz ist zwischen Wortsymbolen (Token) erlaubt:

- Leerzeichen (meist auch Tabulator-Zeichen)
- Zeilenumbrüche
- Kommentare:
  - Von "`--`" bis ⟨Zeilenende⟩

    Unterstützt in SQL-92, PostgreSQL, Oracle, SQL Server, IBM DB2, MySQL. MySQL benötigt ein Leerzeichen nach "`--`", SQL-92 nicht. Access unterstützt diesen Kommentar nicht und auch nicht `/* ...*/`.
  - Von "`/*`" bis "`*/`"

    Nur in PostgreSQL, Oracle, SQL Server und MySQL unterstützt: weniger portabel.

Formatfreie Sprache

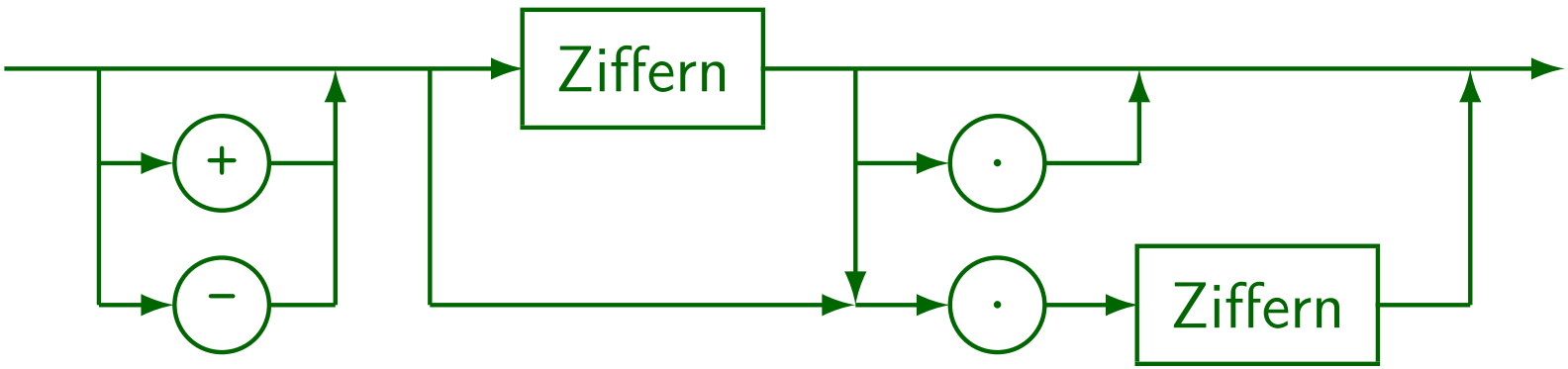
- Obige Regel (beliebigere Leerplatz zwischen Token) bedeutet, dass SQL eine formatfreie Sprache wie z.B. Java ist:
  - Es ist nicht nötig, dass "`SELECT`", "`FROM`", "`WHERE`" am Anfang neuer Zeilen stehen. Man kann auch die ganze Anfrage in eine Zeile schreiben.
  - Man kann z.B. komplexe Bedingungen auf mehrere Zeilen verteilen und Einrückungen verwenden, um die Struktur deutlich zu machen.

Zahlen (1)

- Numerische Literale sind Konstanten numerischer Datentypen (Fixpunkt- und Gleitkommazahlen).
- Z.B.: `1`, `+2.`, `-34.5`, `-.67E-8`
- Zahlen stehen nicht in Hochkommas!
- Numerisches Literal:

Zahlen (2)

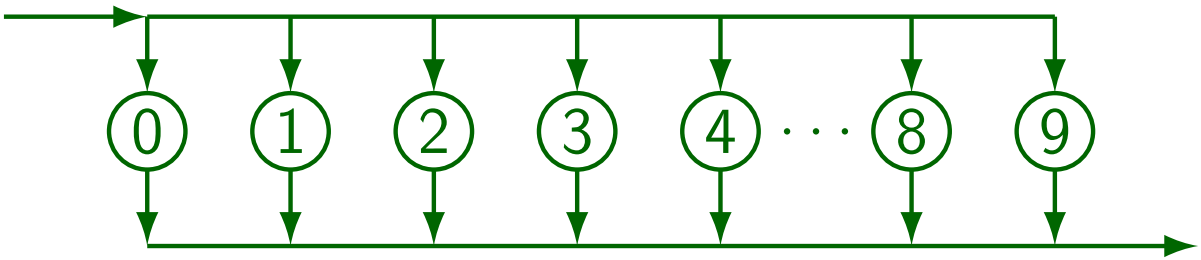
- Festkommazahl (“Exact Numeric Literal”)



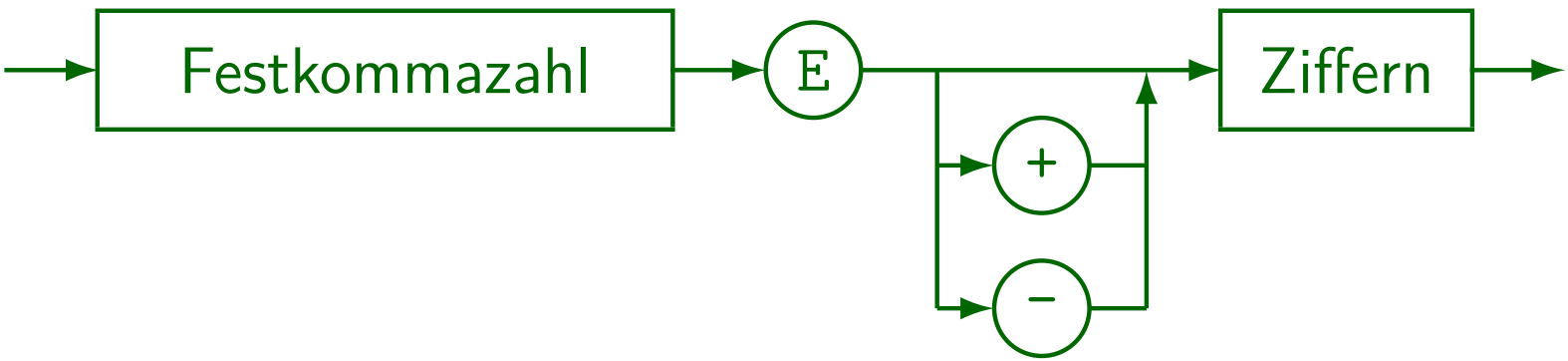
- Ziffern (vorzeichenlose ganze Zahl):

Zahlen (3)

- Ziffer:



- Fließkommazahl (“Approximate Numeric Literal”):



Zeichenketten (1)

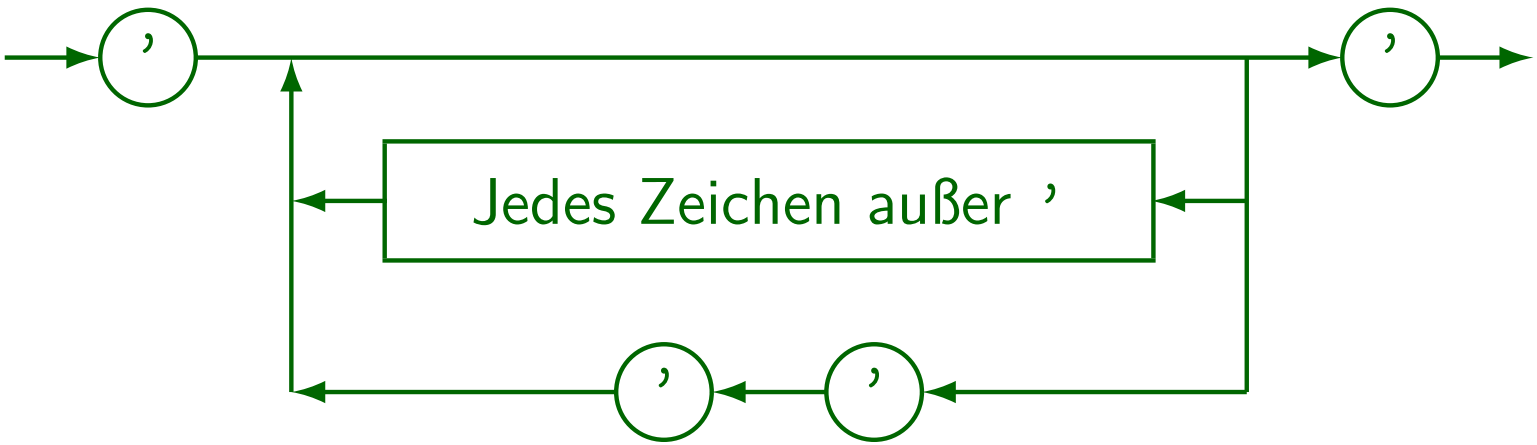
- Ein Zeichenketten-Literal ist eine Folge von Zeichen, eingeschlossen in Hochkommas, z.B.
  - `'abc'`
  - `'Dies ist ein String.'`
- Hochkommas in Zeichenketten müssen verdoppelt werden, z.B. `'John''s book'`.

  Der tatsächliche Wert der Zeichenkette ist `John's book` (mit einfachem Hochkomma). Das Verdoppeln ist nur eine Art, es einzugeben. Beachten Sie, dass die aus Java bekannten Escape-Sequenzen wie `\'` in SQL nicht vorgesehen sind (und z.B. in PostgreSQL auch tatsächlich nicht funktionieren).
- Natürlich ist auch die leere Zeichenkette erlaubt: `''`.

  In Oracle werden die leere Zeichenkette und der Nullwert identifiziert. Das entspricht nicht dem Standard.

Zeichenketten (2)

- String Literal:



Zeichenketten (3)

- SQL-92 erlaubt das Splitten von Zeichenketten (jedes Segment eingeschlossen in '...') zwischen Zeilen.

  PostgreSQL und MySQL unterstützen diese Syntax, Oracle, SQL Server und Access nicht. Zeichenketten können aber mit dem Konkatenations-Operator (|| in PostgreSQL und Oracle, + in SQL Server und Access) kombiniert werden.

- SQL-92 und alle sechs DBMS erlauben Zeilenumbrüche in Zeichenketten-Konstanten.

  D.h., das Hochkomma kann man auf einer folgenden Zeile schließen.

- Access und MySQL erlauben auch in Anführungszeichen " eingeschlossene String-Literale. Nicht konform zum Standard!

  Z.B. bei PostgreSQL und Oracle ist dies ein Syntaxfehler.
  Microsoft SQL Server hat die Option "`SET QUOTED_IDENTIFIER ON`" (inzwischen standardmäßig gesetzt), die das standard-konforme Verhalten liefert.

Andere Konstanten (1)

- Es gibt mehr Datentypen als nur Zahlen und Zeichenketten, z.B. (s. Kapitel 10):
  - Zeichenketten mit nationalem Zeichensatz
  - Datum, Zeit, Zeitstempel, Datum-/Zeit-Intervall
  - Bit-Strings, binäre Daten
  - Large Objects (Dateien als Tabelleneintrag)
- Die Syntax der Konstanten dieser Datentypen ist allgemein sehr systemabhängig.

  Oft gibt es keine Konstanten dieser Typen, aber es gibt eine automatische Typ-Konvertierung ("coercion") von Strings.

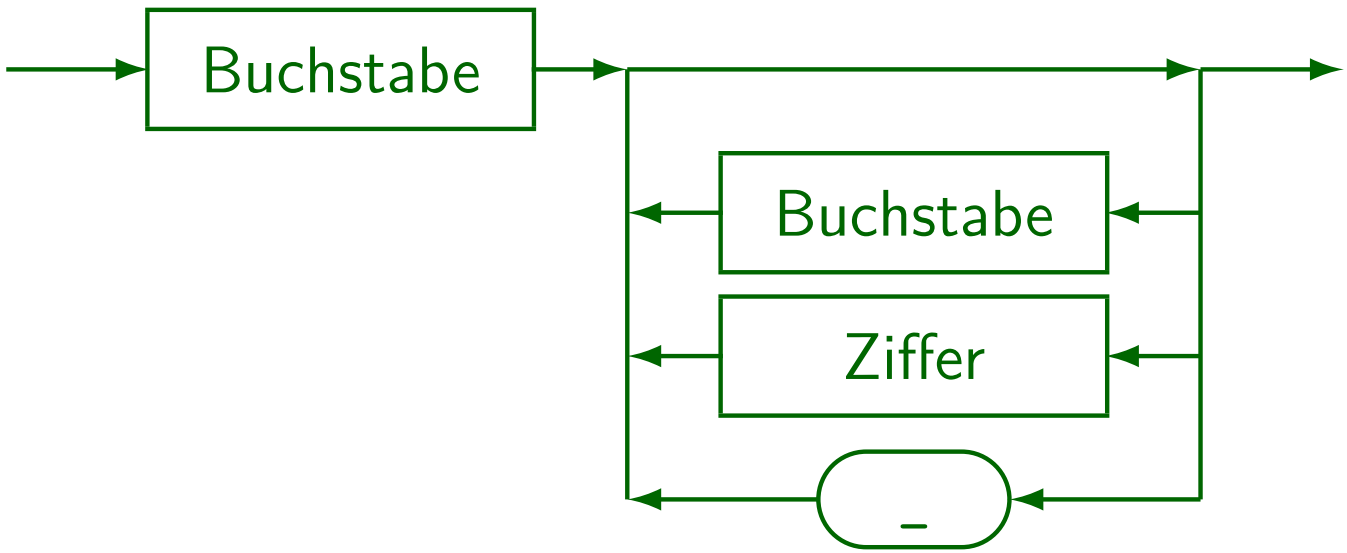
Andere Konstanten (2)

- Z.B. werden Datumswerte wie folgt geschrieben:
  - Oracle: `'31-OCT-02'` (US), `'31.10.2002'` (D).

    Das Default-Format (Teil der nationalen Sprach-Einstellungen) wird automatisch konvertiert, ansonsten:
    `TO_DATE('31.10.2002','DD.MM.YYYY')`.
  - SQL-92-Syntax: `DATE '2002-10-31'`.
  - PostgreSQL und MySQL verstehen diese Syntax (auch ohne "`DATE`"), zusätzlich noch weitere Varianten.
  - DB2: `'2002-10-31'`, `'10/31/2002'`, `'31.10.2002'`.
  - SQL Server: z.B. `'20021031'`, `'10/31/2002'`, `'October 31, 2002'` (abhängig von Sprache).
  - Access: `#10/31/2002#` (US), `#31.10.2002#` (D).

Bezeichner (1)

- Bezeichner:



- Z.B. `Dozenten_Name`, `X27`, aber nicht `_XYZ`, `12`, `2BE`.
- U.a. als Tabellen- und Spaltennamen verwendet.

Bezeichner (2)

- Bezeichner können bis 18 Zeichen haben (mind.).

System	Länge	Erstes Zeichen	Andere Zeichen
SQL-86	≤ 18	A-Z	A-Z,0-9
SQL-92	≤ 128	A-Z,a-z	A-Z,a-z,0-9,_
PostgreSQL	≤ 63	A-Z,a-z,_	A-Z,a-z,0-9,_,$
Oracle	≤ 30	A-Z,a-z	A-Z,a-z,0-9,_,#,$
MySQL	≤ 64	A-Z,a-z,0-9,_,$	A-Z,a-z,0-9,_,$
SQL Server	≤ 128	A-Z,a-z,_,(@,#)	A-Z,a-z,0-9,_,@,#,$
IBM DB2	≤ 18 (8)	A-Z,a-z	A-Z,a-z,0-9,_
Access	≤ 64	A-Z,a-z	A-Z,a-z,0-9,_

In MySQL müssen Bezeichner einen Buchstaben enthalten, ggf. aber nicht vorn.

- Müssen verschieden von reservierten Wörtern sein.

Es gibt viele reservierte Wörter, siehe unten. Einbettungen in Programmiersprachen (PL/SQL, Visual Basic) fügen noch mehr hinzu.

Bezeichner (3)

- Es ist möglich, nationale Zeichen zu verwenden.

  Das ist implementierungsabhängig. Z.B. wählt man in Oracle bei der Installation einen DB-Zeichensatz. Alphanumerische Zeichen von diesem Zeichensatz können in Bezeichnern verwendet werden.

- Bezeichner/Schlüsselwörter nicht case-sensitiv.

  Das scheint das zu sein, was der SQL-92-Standard sagt (das Buch von Date/Darwen über den Standard stellt es klar so dar). In PostgreSQL sind alle Bezeichner nicht case-sensitiv (werden in Kleinbuchstaben konvertiert). Oracle konvertiert alle Zeichen außerhalb von Hochkommas in Großbuchstaben. In SQL Server kann Case-Sensitivität bei der Installation gewählt werden. In MySQL hängt die Case-Sensitivität der Tabellennamen von der Case-Sensitivität von Dateinamen im zugrundeliegenden Betriebssystem ab (Tabellen als Dateien gespeichert). Innerhalb einer Anfrage muss man in MySQL konsistent bleiben. Schlüsselwörter und Spaltennamen sind in MySQL jedoch nie case-sensitiv.

Bezeichner (4)

- Betrachten Sie z.B. die Anfrage

```
SELECT X.VORNAME, X.NACHNAME
FROM   STUDENTEN X
```

- Folgende Anfrage ist vollkommen äquivalent:

```
select X . Vorname ,
       x.NachNaMe
  From Studenten X
```

(Dies zeigt auch die Formatfreiheit von SQL.)

Achtung: Zeichenketten-Vergleiche sind normalerweise case-sensitive:

```
select x.name from studenten x where x.name = 'lisa'
```

wird keine Antwort liefern (obwohl es `'Lisa'` gibt).

Reservierte Wörter - SQL (1)

[1] = Oracle 8.0
[2] = SQL-92
[3] = SQL Server 7

— **A** —
ABSOLUTE[2]
ACCESS[1]
ACTION[2]
ADD[1,2,3]
ALL[1,2,3]
ALLOCATE[2]
ALTER[1,2,3]
AND[1,2,3]
ANY[1,2,3]
ARE[2]
AS[1,2,3]
ASC[1,2,3]
ASSERTION[2]
AT[2]
AUTHORIZATION[2,3]
AUDIT[1]
AVG[2,3]

— **B** —
BACKUP[3]
BEGIN[2,3]
BETWEEN[1,2,3]
BIT[2]
BIT_LENGTH[2]
BOTH[2]
BREAK[3]
BROWSE[3]
BULK[3]
BY[1,2,3]

— **C** —
CASCADE[2,3]
CASCADED[2]
CASE[2,3]
CATALOG[2]
CHAR[1,2]
CHARACTER[2]
CHAR_LENGTH[2]
CHARACTER_LENGTH[2]
CHECK[1,2,3]
CHECKPOINT[3]
CLOSE[2,3]
CLUSTER[1]
CLUSTERED[3]
COALESCE[2,3]
COLLATE[2]
COLLATION[2]
COLUMN[1,3]
COMMENT[1]
COMMIT[2,3]
COMMITTED[3]
COMPRESS[1]
COMPUTE[3]

Reservierte Wörter - SQL (2)

CONFIRM[3]
CONNECT[1,2]
CONNECTION[2]
CONSTRAINT[2,3]
CONSTRAINTS[2]
CONTAINS[3]
CONTAINSTABLE[3]
CONTINUE[2,3]
CONTROLROW[3]
CONVERT[2,3]
CORRESPONDING[2]
COUNT[2,3]
CREATE[1,2,3]
CROSS[2,3]
CURRENT[1,2,3]
CURRENT_DATE[2,3]
CURRENT_TIME[2,3]
CURRENT_TIMESTAMP[2,3]
CURRENT_USER[2,3]
CURSOR[2,3]

— D —

DATABASE[3]
DATE[1,2]
DAY[2]
DBCC[3]
DEALLOCATE[2,3]
DEC[2]
DECIMAL[1,2]
DECLARE[2,3]
DEFAULT[1,2,3]
DEFERRABLE[2]
  DEFERRED[2]
DELETE[1,2,3]
DENY[3]
DESC[1,2]
DESCRIBE[2]
DESCRIPTOR[2]
DIAGNOSTICS[2]
DISCONNECT[2]
DISK[3]
DISTINCT[1,2,3]
DISTRIBUTED[3]
DOMAIN[2]
DOUBLE[2,3]
DROP[1,2,3]
DUMMY[3]
DUMP[3]

— E —

ELSE[1,2,3]
END[2,3]
END-EXEC[2]
ERRLVL[3]
ERROREXIT[3]
ESCAPE[2,3]
EXCEPT[2,3]
EXCEPTION[2]

Reservierte Wörter - SQL (3)

EXCLUSIVE[1]
EXEC[2,3]
EXECUTE[2,3]
EXISTS[1,2,3]
EXIT[3]
EXTERNAL[2]
EXTRACT[2]
— **F** —
FALSE[2]
FETCH[2,3]
FILE[1,3]
FILLFACTOR[3]
FIRST[2]
FLOAT[1,2]
FLOPPY[3]
FOR[1,2,3]
FOREIGN[2,3]
FOUND[2]
FREETEXT[3]
FREETEXTTABLE[3]
FROM[1,2,3]
FULL[2,3]
— **G** —
GET[2]
GLOBAL[2]
GO[2]
GOTO[2,3]
GRANT[1,2,3]
GROUP[1,2,3]
— **H** —
HAVING[1,2,3]
HOLDLOCK[3]
HOUR[2]
— **I** —
IDENTITY[2,3]
IDENTITY_INSERT[3]
IDENTITYCOL[3]
IDENTIFIED[1]
IF[3]
IMMEDIATE[1,2]
IN[1,2,3]
INCREMENT[1]
INDEX[1,3]
INDICATOR[2]
INITIAL[1]
INITIALLY[2]
INNER[2,3]
INPUT[2]
INSENSITIVE[2]
INSERT[1,2,3]
INT[2]
INTEGER[1,2]
INTERSECT[1,2,3]
INTERVAL[2]
INTO[1,2,3]
IS[1,2,3]

Reservierte Wörter - SQL (4)

ISOLATION[2,3]
— **J** —
JOIN[2,3]
— **K** —
KEY[2,3]
KILL[3]
— **L** —
LANGUAGE[2]
LAST[2]
LEADING[2]
LEFT[2,3]
LEVEL[1,2,3]
LIKE[1,2,3]
LINENO[3]
LOAD[3]
LOCAL[2]
LOCK[1]
LONG[1]
LOWER[2]
— **M** —
MATCH[2]
MAX[2,3]
MAXEXTENTS[1]
MIN[2,3]
MINUS[1]
MINUTE[2]
MIRROREXIT[3]
MODE[1]
MODIFY[1]
MODULE[2]
MONTH[2]
— **N** —
NAMES[2]
NATIONAL[2,3]
NATURAL[2]
NCHAR[2]
NETWORK[1]
NEXT[2]
NO[2]
NOAUDIT[1]
NOCHECK[3]
NOCOMPRESS[1]
NONCLUSTERED[3]
NOT[1,2,3]
NOWAIT[1]
NULL[1,2,3]
NULLIF[2,3]
NUMBER[1]
NUMERIC[2]
— **O** —
OCTET_LENGTH[2]
OF[1,2,3]
OFF[3]
OFFLINE[1]
OFFSETS[3]
ON[1,2,3]

Reservierte Wörter - SQL (5)

ONCE[3]
ONLINE[1]
ONLY[2,3]
OPEN[2,3]
OPENDATASOURCE[3]
OPENQUERY[3]
OPENROWSET[3]
OPTION[1,2,3]
OR[1,2,3]
ORDER[1,2,3]
OUTER[2,3]
OUTPUT[2]
OVER[3]
OVERLAPS[2]

— **P** —

PARTIAL[2]
PCTFREE[1]
PERCENT[3]
PERM[3]
PERMANENT[3]
PIPE[3]
PLAN[3]
POSITION[2]
PRECISION[2,3]
PREPARE[2,3]
PRESERVE[2]
PRIMARY[2,3]
PRINT[3]
PRIOR[1,2]
PRIVILEGES[1,2,3]
PROC[3]
PROCEDURE[2,3]
PROCESSEXIT[3]
PUBLIC[1,2,3]

— **R** —

RAISERROR[3]
RAW[1]
READ[2,3]
READTEXT[3]
REAL[2]
RECONFIGURE[3]
REFERENCES[2,3]
RELATIVE[2]
RENAME[1]
REPEATABLE[3]
REPLICATION[3]
RESOURCE[1]
RESTORE[3]
RESTRICT[2,3]
RETURN[3]
REVOKE[1,2,3]
RIGHT[2,3]
ROLLBACK[2,3]
ROW[1]
ROWCOUNT[3]
ROWGUIDCOL[3]

Reservierte Wörter - SQL (6)

ROWID[1]
ROWNUM[1]
ROWS[1,2]
RULE[3]
— **S** —
SAVE[3]
SCHEMA[2,3]
SCROLL[2]
SECOND[2]
SECTION[2]
SELECT[1,2,3]
SERIALIZABLE[3]
SESSION[1,2]
SESSION_USER[2,3]
SET[1,2,3]
SETUSER[3]
SHARE[1]
SHUTDOWN[3]
SIZE[1,2]
SMALLINT[1,2]
SOME[2,3]
SQL[2]
SQLCODE[2]
SQLERROR[2]
SQLSTATE[2]
START[1]
STATISTICS[3]
SUBSTRING[2]
SUCCESSFUL[1]
SUM[2,3]
SYNONYM[1]
SYSDATE[1]
SYSTEM_USER[2,3]
— **T** —
TABLE[1,2,3]
TAPE[3]
TEMP[3]
TEMPORARY[2,3]
TEXTSIZE[3]
THEN[1,2,3]
TIME[2]
TIMESTAMP[2]
TIMEZONE_HOUR[2]
TIMEZONE_MINUTE[2]
TO[1,2,3]
TOP[3]
TRAILING[2]
TRAN[3]
TRANSACTION[2,3]
TRANSLATE[2]
TRANSLATION[2]
TRIGGER[1,3]
TRIM[2]
TRUE[2]
TRUNCATE[3]
TSEQUAL[3]

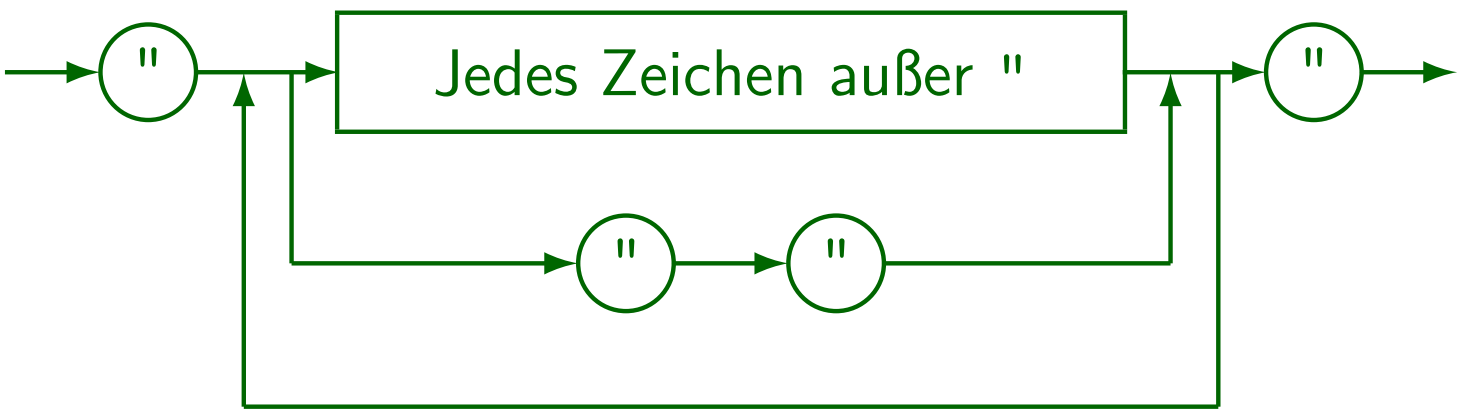
Reservierte Wörter - SQL (7)

— **U** —
UID[1]
UNCOMMITTED[3]
UNION[1,2,3]
UNIQUE[1,2,3]
UNKNOWN[2]
UPDATE[1,2,3]
UPDATETEXT[3]
UPPER[2]
USAGE[2]
USE[3]
USER[1,2,3]
USING[2]
— **V** —
VALIDATE[1]
VALUE[2]
VALUES[1,2,3]
VARCHAR[1,2]
VARCHAR2[1]
VARYING[2,3]
VIEW[1,2,3]
— **W** —
WAITFOR[3]
WHEN[2,3]
WHENEVER[1,2]
WHERE[1,2,3]
WHILE[3]
WITH[1,2,3]
WORK[2,3]
WRITE[2]
WRITETEXT[3]
— **Y** —
YEAR[2]
— **Z** —
ZONE[2]

Delimited Identifier (1)

- Es ist möglich, jede Zeichenfolge in Anführungszeichen als Bezeichner zu verwenden, z.B. "id, 2!".

  Solche Bezeichner sind case-sensitive und es gibt keine Konflikte mit reservierten Wörtern. SQL-86 enthält dies nicht. In Deutsch würde “Delimited Identifier” in etwa “Abgegrenzte Bezeichner” heißen.



Delimited Identifier (2)

- Delimited Identifier sind keine String-Konstanten! Solche haben die Form `'...'`.

  SQL Server akzeptiert ' und " für String-Konstanten und nimmt `[...]` für Delimited Identifier. "`SET QUOTED_IDENTIFIER ON`" schaltet auf den SQL-92-Standard um (aber Delimited Identifier sind nicht case-sensitive). Access versteht `[...]` und `` `...` `` für Delimited Identifier, schließt aber die Zeichen `!.` `` ` `` `[]"` und Leerzeichen am Anfang aus.

- Wenn man z.B. in Oracle schreibt:

  ```
  SELECT * FROM STUDENTEN WHERE VORNAME = "Lisa"
  ```

  Fehler: `"Lisa"` ist ein ungültiger Spaltenname.

  Delimited Identifier werden normalerweise nur verwendet, um ausgegebene Spaltennamen umzubenennen (oder wenn Spaltennamen in einer neuen DBMS-Version zu reservierten Wörtern werden).

Delimited Identifier (3)

- Delimited Identifier werden hauptsächlich verwendet, um Ausgabe-Spalten umzubenennen, z.B.

```
SELECT VORNAME AS "Vorname", NACHNAME "Name"
FROM   STUDENTEN
```

  "AS" ist optional (außer in MS Access).

- Ist aber der neue Spaltenname ein legaler Bezeichner, sind die Anführungszeichen unnötig:

```
SELECT VORNAME AS V_NAME, NACHNAME Name
FROM   STUDENTEN
```

- In PostgreSQL werden normale Bezeichner (ohne "...") in Kleinbuchstaben ausgegeben.

  In Oracle in Großbuchstaben.

Lexikalische Fehler

- Anführungszeichen, z.B. `"Lisa"`, für String-Literale verwenden (Delimited Identifier, kein String).

  Manche Systeme erlauben `"..."`, aber das verletzt den Standard.

- Hochkommas für Zahlen verwenden, z.B. `'123'`.

  Das sollte einen Typfehler geben. Das DBMS könnte jedoch einfach den Typ von einem der Operanden konvertieren. Da < usw. für Strings und Zahlen anders definiert ist, kann dies gefährlich sein und sollte vermieden werden. Z.B. `'12' < '3'`.

- Reservierte Wörter als Tabellen-, Spalten- oder Tupelvariablennamen verwenden.

  Die Fehlermeldung könnte seltsam sein (nicht verständlich). Daher sollte man diese Möglichkeit im Auge behalten.

SQL-Anfragen: Begrenzung

- In Oracle SQL*Plus muss jedes SQL-Statement mit einem Semikolon "`;`" abgeschlossen werden.

  Da SQL-Statements über mehrere Zeilen gehen können, ist dies notwendig, damit SQL*Plus weiß, wann das SQL-Statement beendet ist. Auch wenn SQL in C-Programme eingebettet ist, wird das Semikolon als Begrenzer verwendet.

- Aber eigentlich gehört das Semikolon nicht zum SQL-Statement.

  Z.B. ist in dem Anfrage-Analyse-Fenster von MS SQL Server kein Semikolon erforderlich. Es könnte sogar ein Fehler sein, wie im Kommandozeilen-Interface von DB2. Auch wenn SQL-Statements als Strings an Prozeduren übermittelt werden, wie z.B. in ODBC, ist kein Semikolon erforderlich.