SQL Server

Einfache Abfragen

Stephan Karrer

Namensauflösung

Objekte der Datenbank werden wie folgt referenziert:

[[[server.] [database] .] [schema] .] object

server	Name des DB-Servers
database	Name der DB-Instanz
schema	Name des Schemas
object	Name des Objekts

```
SELECT EmployeeID, LoginID, ManagerID
  FROM AdventureWorks.HumanResources.Employee;

SELECT EmployeeID, LoginID, ManagerID
  FROM HumanResources.Employee;

SELECT EmployeeID, LoginID, ManagerID
  FROM Employee;
```

Abfragen mit SQL: SELECT-Anweisung

```
SELECT [ALL|DISTINCT] Auswahlliste
    FROM Quelle
    WHERE Where-Klausel
    [GROUP BY (Group-by-Attribut)
        [HAVING Having-Klausel]]
    [ORDER BY (Sortierungsattribut)[ASC|DESC]]
```

```
SELECT * FROM employees;

SELECT last_name, job_id, salary, department_id
    FROM employees;

SELECT first_name AS first, last_name [NACH-NAME], Gehalt = salary
    FROM employees;

SELECT last_name, job_id AS "Job-Kennung"
    FROM employees
    ORDER BY department_id, last_name DESC;

SELECT DISTINCT job_id
    FROM employees;
```

Abfragen mit SQL: SELECT-Anweisung mit Ausdrücken

```
SELECT last_name, salary, 12*(salary+100) AS new_annual_sal
    from employees;

SELECT first_name + last_name + ' is a ' + job_id
                AS "Employee Details"
    FROM employees;

SELECT last_name "employees in department 50"
    FROM employees
    WHERE department_id = 50
    ORDER BY last_name DESC;

SELECT DISTINCT job_id
    FROM employees
    WHERE salary <= 5000;
```

Numerische Datentypen

Datentyp	Bereich	Kommentar
bigint	-2^{63} bis $2^{63}-1$	8 Byte Ganzzahl
int	-2^{31} bis $2^{31}-1$	4 Byte Ganzzahl
smallint	-2^{15} bis $2^{15}-1$	2 Byte Ganzzahl
tinyint	0 bis 255	1 Byte Ganzzahl
decimal(p,s) dec(p,s) numeric(p,s)	$-10^{38}+1$ bis $10^{38}-1$	Feste Anzahl Dezimalstellen: p Anzahl Stellen gesamt (Standard: 18, Max.: 38) s Anzahl Nachkommastellen (Standard: 0) Ab SQL Server 2005: vardecimal-Speicherformat
money	-922.337.203.685.477,5808 bis 922.337.203.685.477,5807	Verwendet 8 Byte
smallmoney	-214.748,3648 bis 214.748,3647	Verwendet 4 Byte
float(n)	$-1{,}79e^{308}$ bis $1{,}79e^{308}$	Gleitkommazahl in einfacher Genauigkeit p Speicherplatz in Bits (Standard: 53)

Arithmetische Operatoren

Operator	Kommentar
+, -	Addition, Subtraktion
*, /	Multiplikation, Division
%	Modulo: nur für Ganzzahl- und Währungswerte

```
SELECT employee_id, salary * 1.1 + salary * commission_pct
    FROM employees;

SELECT employee_id, salary * ( 1.1 + commission_pct) "New Salary"
    FROM employees;

SELECT 800+900;   SELECT 20%2;    SELECT 2337.123/100.5;

SELECT $123.45+$55;
/*Währungssymbole stehen zur Verfügung, Dezimalpunkt ist Pflicht */
```

Binäre Datentypen

Datentyp	Bereich	Kommentar
bit	0 oder 1	Dient auch als Ersatz für Wahrheitswerte (Boolean): 0 false 1 true
binary(n)	max. 8000 Byte (n=8000)	Binärdaten fester Länge der Größe n, Standard ist 1
varbinary(n) varbinary(max)	max. 8000 Byte (n=8000) max. 2^{31}-1 Byte	Binärdaten variabler Länge der Größe n, Standard ist 1
image	max. 2^{31}-1 Byte	Binärdaten variabler Länge, veraltet !

```
CREATE TABLE MyCustomerTable (
 user_login   varbinary(85) DEFAULT SUSER_SID(),
 data_value   varbinary(1) );

INSERT MyCustomerTable (data_value) VALUES (0x4F);

DROP TABLE MyCustomerTable;
```

Binäre Operatoren

Operator	Kommentar
&	Bitweises Und
|	Bitweises Oder
~	Bitweises Negieren
^	Bitweises Exklusives Oder

- Anwendbar für int, smallint, tinyint, und bit
- Anwendbar für varbinary und binary, sofern der andere Operand ein Ganzzahl-Typ ist
- Ergebnis ist stets vom Typ Ganzzahl oder bit, d.h. es finden gegebenenfalls Typkonvertierungen statt

```
/*Beispiel  für (A & B):
* 0000 0000 1010 1010 -- Dezimal 170 (Datentyp int)
* 0000 0000 0100 1011 -- Dezimal 75  (Datentyp int)
* -------------------
* 0000 0000 0000 1010 -- Dezimal 10
*/

SELECT 170 & 75 ;
SELECT 0x00AA & 75; -- erster Operand hexadezimal
```

Zeichenketten

Datentyp	Bereich	Kommentar
char(n) nchar(n)	max. 8000 Zeichen (Byte) max. 4000 Zeichen (2 Byte Unicode)	Zeichenketten der festen Länge n, Standard ist n=1
varchar(n) nvarchar(n)	max. 8000 Zeichen (Byte) max. 4000 Zeichen (2 Byte Unicode)	Zeichenketten variabler Länge mit Max.-Wert n
varchar(max) nvarchar(max)	max. 2^{31} Byte max. 2^{30} Byte	Zeichenketten variabler Länge
text ntext	max. 2^{31} Byte max. 2^{30} Byte	Zeichenketten variabler Länge, veraltet !

```
CREATE TABLE tab1 (col1 varchar(6), col2 char(6),
                   col3 varchar(6), col4 char(6) );

INSERT INTO tab1 (col1, col2, col3, col4)
        VALUES ('abc', 'def ', 'ghi ', 'jkl');

SELECT col1 + col2 + col3 + col4 Concatenation
        FROM tab1;
```

Concatenation
----
abcdef ghi jkl

Datumstypen

Datentyp	Bereich	Kommentar
datetime	1. 1.1753 bis 31.12.9999	Die Genauigkeit ist auf 3,33 Millisekunden beschränkt
smalldatetime	1.1.1900 bis 6. 6.2079	Die Genauigkeit ist auf 1 Minute beschränkt
date	1.1.0001 bis 31.12.9999	Nur Datum
time(length)	hh:mm:ss:[0-7]	Nur Zeit, Länge 0 - 7 (max. Genauigkeit 100 Nanosekunden und ist default)
datetime2(length)	1.1.0001 bis 31.12.9999	Kombination von date und time
datetimeoffset (length)	1.1.0001 bis 31.12.9999	datetime2 mit zusätzlicher Zeitzone

Datumsformate

ISO 8601-Format: yyyy-mm-ddThh:mm:ss[.mmm]

```
Beispiel:
'2004-05-23T14:25:10'
'2004-05-23T14:25:10.487'
```

oder als nationalsprachliche Zeichenkette (abhängig vom nationalen Kontext)

Alphabetische Datumsangabe, `z.B. 'April 15, 1998'`

Numerische Datumsformate, `z.B. '4/15/1998 14:30:20:100'`

Unstrukturierte Zeichenfolge, `z.B. '19981207'`

Die nationale Anpassung des Formats erfolgt mit:

SET DATEFORMAT ()
SET LANGUAGE ()

```
SET DATEFORMAT mdy;
SELECT '12/31/1998' AS DateVar;

-- Result: 1998-12-31 00:00:00.000
```

Operatoren für Datumstypen

Operator	Kommentar
+, -	Addition, Subtraktion
=, <, >, >=, <=, <>, !=	Vergleichsoperatoren

```
SELECT GETDATE() - 15;
SELECT GETDATE() - '19940101';  --geht auch, aber unsinnig

SELECT CAST('12.01.2004' AS DATETIME) +15 ;

SELECT first_name, last_name
    FROM employees
    WHERE hire_date < 'Mai 1, 2006';  -- sofern deutscher Kontext
```

Sonstige Datentypen

Datentyp	Kommentar
cursor	Kann nur für Laufvariablen bei programmierten Abfragen verwendet werden
sql_variant	Kann beliebigen Typ, ausser die LOBs, bezeichnen
table	Datentyp für temporäre Tabellen, kann nicht für Tabellenspalten benutzt werden
timestamp	8 Byte Binärwert, der üblicherweise als Versionskennung benutzt wird
uniqueidentifier	16 Byte Binärwert zur Speicherung von GUIDs
xml	Zur Speicherung von XML-Daten
benutzerdefiniert	Alias-Datentypen, die mit CREATE TYPE generiert werden oder CLR-Typen aus einer .NET-Sprache

ANSI-SQL92 Datentypen und ihre Entsprechung

Synonym	SQL Server
binary varying	varbinary
char varying	varchar
character(n)	char(n)
character varying(n)	varchar(n)
dec	decimal
double precision	float
float(n)	real, für n=1-7 bzw. float für n=8-15
integer	int
national character(n)	nchar(n)
national character varying(n)	nvarchar(n)
national text	ntext
rowversion	timestamp

Logik-Operatoren

Vergleichsoperatoren können mit Zeichendaten, numerischen Daten oder Datumsdaten verwendet werden.

Operator	Kommentar
=, <, >, >=, <=, !<, !>, <>, !=	Vergleichsoperatoren

Logische Verküpfungsoperatoren können auf logische Ausdrücke und Werte angewendet werden.

Operator	Kommentar
AND, OR, NOT	Basisoperatoren
IS NULL	Prüft, ob der Operand ein NULL-Wert ist
IS NOT NULL	Prüft, ob der Operand kein NULL-Wert ist
BETWEEN, IN, EXISTS, ALL, SOME, ANY, LIKE	

Vergleichs-Operatoren: Einfache Beispiele

```
SELECT * FROM employees
    WHERE salary = 2500;

SELECT * FROM employees
    WHERE salary != 2500;

SELECT * FROM employees
    WHERE salary > 2500;

SELECT * FROM employees
    WHERE salary BETWEEN 5000 AND 10000;

SELECT * FROM employees
    WHERE job_id IN ('SA_MAN', 'SA_REP');

SELECT * FROM employees
    WHERE commission_pct IS NULL AND salary = 2100;
```

LIKE-Operator für Vergleiche

x [NOT] LIKE y [ESCAPE 'z']

Zur Bildung von Mustern können verwendet werden:

- % beliebig viele Zeichen (auch keines)
- _ genau ein Zeichen
- [ ] beliebiges Zeichen aus dem angegeben Bereich, z.B. [abc], [0-9]
- [^] beliebiges Zeichen ausser aus dem angegeben Bereich, z.B. [^0-9]

```
SELECT last_name, first_name
   FROM employees
   WHERE last_name LIKE 'Bai%';

SELECT Name
   FROM sys.system_views
   WHERE Name LIKE 'dm%';

SELECT employee_id, last_name, first_name
   FROM employees
   WHERE first_name LIKE '[KG]%[yn]';
```

TOP-N

SELECT [ALL | DISTINCT] [TOP expression [PERCENT] [WITH TIES]]	
ALL | DISTINCT	Mit oder ohne potentielle Duplikate
TOP (expression)	Nur die ersten n Zeilen (expression = n)
TOP (expression) PERCENT	Nur die ersten n-% Zeilen
TOP ... WITH TIES	Zusätzlich werden Zeilen mit gleichen geordneten Werten wie die TOP-Zeilen ausgegeben

```
SELECT TOP(10) WITH TIES last_name, salary
    FROM employees
    ORDER BY salary DESC;

SELECT TOP(50) PERCENT last_name, salary
    FROM employees
    ORDER BY salary DESC

SELECT TOP(2) WITH TIES last_name, salary
    FROM employees ORDER BY salary DESC;
```

Spezielle Anzeige-Klauseln: TABLESAMPLE

TABLESAMPLE [SYSTEM] (sample_number [PERCENT | ROWS])	
TABLESAMPLE	Gibt stichprobenartig ca. den gewünschten Prozentsatz oder die Anzahl Zeilen zurück. Ist nur geeignet für große Tabellen, die viele 8KB-Seiten benutzen, da die Auswahl anhand der Seiten erfolgt.

```
SELECT first_name, last_name
    FROM employees
    TABLESAMPLE (10 PERCENT) ;
```

Fallunterscheidungen mit CASE

```
/* Einfacher CASE-Ausdruck */
SELECT last_name,
    CASE salary
        WHEN 2500 THEN 'Low'
        WHEN 5000 THEN 'High'
        ELSE 'Medium' END
    FROM employees;


/* Komplexer CASE-Ausdruck */
SELECT last_name, salary,
       CASE
           WHEN salary < 2500 THEN 'Low'
           WHEN salary > 5000 THEN 'High'
           ELSE 'Medium' END AS range
    FROM employees
    ORDER BY last_name;
```

Nullwerte (Null Values)

Nullwerte stehen für nicht verfügbare bzw. unbekannte Werte und können in Tabellen als Werte von Zeilen vorkommen

Werte können explizit auf NULL gesetzt bzw. daraufhin überprüft werden

Ist ein Operand in arithmetischen Ausdrücken ein Nullwert, so ergibt die Auswertung stets NULL.

Bei der Konkatenation von Zeichenketten wird ein Nullwert ignoriert (d.h. wie eine leere Zeichenkette behandelt)

Bei der Auswertung logischer Ausdrücke wird durch Nullwerte die Prädikatenlogik erweitert.

Logik der Wahrheitswerte

x	y	x AND y	x OR y	NOT x
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	NULL	NULL	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	NULL	FALSE	NULL	TRUE
NULL	TRUE	NULL	TRUE	NULL
NULL	FALSE	FALSE	NULL	NULL
NULL	NULL	NULL	NULL	NULL

Behandeln von Null-Werten

```
SELECT last_name
    FROM employees
    WHERE commission_pct IS NULL
    ORDER BY last_name;
```

oder via SQL-Funktionen

Funktion	Kommentar
COALESCE (expression [,...n])	Gibt den ersten Ausdruck ungleich NULL zurück.
ISNULL (expression , replacement)	Ersetzt NULL durch den angegebenen Ersatzwert.
NULLIF (expression , expression)	Gibt einen NULL-Wert zurück, wenn die beiden angegebenen Ausdrücke gleich sind.

```
SELECT last_name, ISNULL(CAST(commission_pct AS varchar),'Not Applicable')
                  "COMMISSION" FROM employees
  WHERE last_name LIKE 'B%'
  ORDER BY last_name;
```

Konvertierung der Datentypen

Funktion
CAST (expression AS data_type [(length)])
CONVERT (data_type [(length)] , expression [, style])

```
SELECT 'The salary is ' + CAST(salary AS varchar(12))
                                AS ListPrice
   FROM employees;

SELECT GETDATE() AS UnconvertedDateTime,
       CAST(GETDATE() AS nvarchar(30)) AS UsingCast,
       CONVERT(nvarchar(30), GETDATE(), 126)
                                   AS UsingConvertTo_ISO8601;
```

Weitere SQL-Funktionen

Kategorie	Kommentar
Konfigurationsfunktionen	Informationen zu Konfigurationseinstellungen
Systemfunktionen	Informationen zur SQL Server-Instanz
Metadatenfunktionen	Informationen zu Attributen von Datenbanken und Datenbankobjekten
Kryptografiefunktionen	Verschlüsselung und digitale Signaturen
Sicherheitsfunktionen	Informationen zu Benutzern und Rollen
Datums- und Zeitfunktionen	Operationen für Datums- und Zeitwerte
Mathematische Funktionen	Durchführung numerischer Berechnungen
Zeichenfolgenfunktionen	Zeichenkettenverarbeitung
...	
Benutzerdefinierte Funktionen	