SQL - basis

Inhoudsopgave

Introductie: SQL

SQL

- Structured Query Language.

Een database-subtaal om gegevens in een relationele database op te vragen en bij te werken, en om de database te beheren. Hoewel het geen programmeertaal als C en Pascal is, kan SQL worden gebruikt om interactieve opvraagverzoeken (queries) te formuleren; SQL kan ook in een applicatie zijn opgenomen als instructie om gegevens te manipuleren. De SQL-standaard bevat ook onderdelen om gegevens te definiëren, wijzigen, besturen en beveiligen.

SQL geschiedenis

SQL komt oorspronkelijk uit de stal van IBM waar in de jaren zeventig onderzoek werd gedaan naar de mogelijkheden van relationele databasesystemen. Er werd een experimenteel database managementsysteem ontwikkeld met de naam System R. Als databasetaal werd gekozen voor Sequel, ooit uitgedacht door R.F. Boyce en D.D. Chamberlin. De taal werd hernoemd naar SQL, maar veel mensen spreken dit nog uit als 'siekwul'. Het System R-project bleek een succes en de bevindingen werden gebruikt om SQL/DS te bouwen. Dit was IBM's eerste relationele database managementsysteem dat werd verkocht. Vervolgens bracht IBM DB2 uit wat een enorm succes was. Daarna bouwden andere fabrikanten ook relationele systemen en sommige daarvan implementeerden SQL als databasetaal. In de jaren tachtig kwamen bedrijven als Informix, Sybase en Oracle met soortgelijke oplossingen. Ook deze oplossingen zijn zeer succesvol gebleken en worden op grote schaal toegepast. Om te zorgen voor standaardisatie van SQL zijn door een aantal instanties standaarden ontwikkeld. De meest bekende is ISO in samenwerking met ANSI. SQL is een databasetaal die dus al een tijdje meegaat en de verwachting is dat dit ook nog wel een tijd zo zal blijven. De interessantste ontwikkelingen bij gegevenshantering vinden plaats op het gebied van LDAP.

SQL dialecten

Hoewel er een ANSI SQL bestaat, een standaard die gedefinieerd is door het ANSI-instituut, wordt door de fabrikanten van databasemanagementsystemen hier niet altijd aan gehouden. Elke database heeft zijn eigen SQL-definities. In dit document zijn de basis SQL-statements opgenomen.

Basis SQL-commando's

CREATE TABLE

Beschrijving:
Maakt een nieuwe lege tabel aan in de geselecteerde database.

Syntaxis:
CREATE TABLE Tabelnaam (
Veldnaam datatype[grootte] veldtype[..],
);

Datatype
Datatype is het type waarde dat door het systeem wordt herkend wat in de betreffende kolom moet worden ingevoerd.

Soort	Datatype	Beschrijving
Tekst	CHARACTER(3)	Tekstveld (aantal karakters)
	CHAR	Tekstveld (aantal karakters)
Numeriek (exact)	DECIMAL	Een cijfer met decimalen
	DEC	Hetzelfde als DECIMAL
	NUMERIC(1,2)	Hetzelfde als decimal, het aantal decimalen is hierbij gespecificeerd (1 cijfer met 2 decimalen)
	INTEGER	Nummer zonder decimalen. Hierbij wordt de grootte van het nummer gespecificeerd door het databasesysteem.
	INT	Hetzelfde als INTEGER.
	SMALLINT	Nummer zonder decimalen. Hierbij wordt de grootte van het nummer gespecificeerd door het databasesysteem, welke kleiner zal zijn dan INTEGER.
Numeriek (globaal)	FLOAT	Nummers in exponentiële vorm.
	REAL	Gelijk aan FLOAT. Hierbij wordt de grootte van het nummer gespecificeerd door het databasesysteem.
	DOUBLE PRECISION	Gelijk aan REAL, maar met een dubbele precisie.
	DOUBLE	Hetzelfde als DOUBLE PRECISION.
Andere datatypes	DATE	Datum. In Europese implementaties: dd.mm.yyyy (dd = dag, mm = maand, yyyy = jaar)
	TIME	Tijd. In Europese implementaties: hh-mm-ss (hh = uur, mm = minuten, ss = secondes)
	VARCHAR	String met een variabele lengte, welke afhankelijk is van het databasesysteem (maximaal 254 tot 2048 karakters).
	LONG VARCHAR	String met een variabele lengte, welke afhankelijk is van het databasesysteem (maximaal 16 kilobytes).

Tabel 1: Datatype

Grootte
Geeft aan uit hoeveel eenheden de waarde mag bestaan, bijvoorbeeld CHAR(3) betekent dat de kolom alleen waardes mag hebben met een maximale grootte van 3 karakters. Bij bijvoorbeeld NUMERIC(1,2) wordt bedoeld dat de kolom alleen waardes mag hebben met 1 cijfer voor de komma en 2 cijfers achter de komma (decimalen).

Veldtype
Geeft aan aan welke voorwaarde de kolom nog meer moet voldoen:

Veldtype	Beschrijving
NOT NULL	In de kolom mag geen lege velden voorkomen
NULL	In de kolom mag lege velden voorkomen
UNIQUE	In de kolom moeten alle waarden uniek zijn, er mogen geen twee dezelfde waarden voorkomen.
CHECK(Logische construct)	Definieert dat de inhoud van de velden aan een bepaalde voorwaarde moet voldoen, bijvoorbeeld M/V.
DEFAULT=waarde	Wanneer er geen waarde wordt opgegeven voor een veld, zal deze de standaardwaarde krijgen.
PRIMARY KEY	Definieert dat de kolom(men) de primaire sleutel bevat, de waarden in de kolom moeten uniek zijn.

Tabel 2: Veldtype

DELETE FROM

Beschrijving:
Verwijderd records uit een database tabel.

Syntaxis:
DELETE FROM Tabelnaam [WHERE logische construct];

Bijvoorbeeld:
```
DELETE FROM Test
        WHERE testwaarde=0;
```

Resultaat:
Alle records, waarbij de velden van kolom testwaarde die de waarde 0 hebben worden verwijderd.

DROP TABLE

Beschrijving:
Verwijderd een tabel uit een database.

Syntaxis:
DROP TABLE Tabelnaam;

INSERT

Beschrijving:
Voegt nieuwe records toe aan een gespecificeerde tabel.

Syntaxis:
INSERT INTO Tabelnaam [(Veldnaam,...)] {VALUES (Waarde, ...)} ;

VALUES (Waarde, ...)
Met VALUES worden de waarden bepaald die in de nieuwe record moet worden opgenomen.

Bijvoorbeeld:
INSERT INTO Test (testobject, resultaat, omgevingstemperatuur)
VALUES (fiets, positief, 25);

Resultaat:
Aan de tabel test wordt een record toegevoegd:

Tabel: Test		
Testobject	**Resultaat**	**Omgevingstemperatuur**
Fiets	Positief	25

Indien een recordrij nog meer velden heeft dan de gespecificeerde, dan krijgen de overige velden de waarde NULL. Indien er een van de overige kolommen de specificatie NOT NULL heeft gekregen, dan geeft SQL een foutmelding.

SELECT

Beschrijving:
Haalt data uit een database.

Syntaxis:
SELECT * | {[DISTINCT | ALL] {veldnaam | CASE}[,..]}

```
FROM {Tabelnaam [Alias]} [,...]
[WHERE logische construct]
[GROUP BY {veldnaam | Integer} [,...]]
[HAVING logische construct]
[{UNION | INTERSECT | EXEPT [ALL] SELECT-statement}]
[...]
[ORDER BY {veldnaam | Integer} [,...]]
[INTO TEMP Tabelnaam];
```

*

Het sterretje vraagt om alle kolommen die voldoen aan de gestelde eisen.

DISTINCT

DISTINCT vraagt alleen maar om unieke records. Redundante data wordt niet meegenomen, alleen het eerste record wordt getoond, de volgende record(s) met dezelfde naam wordt overgeslagen.

Bijvoorbeeld:

Tabel: Test		
Testobject	**Resultaat**	**Omgevingstemperatuur**
Fiets	Positief	25
Brommer	Negatief	30
Quad	Positief	15
Auto	Positief	20
Auto	Negatief	23

SELECT DISTINCT Testobject FROM Test;

Testobject
Fiets
Brommer
Quad
Auto

FROM {Tabelnaam [Alias]}[,..]
FROM vertelt het databasemanagementsysteem uit welke tabellen de data moet worden gehaald.

WHERE logische construct
Met WHERE wordt de voorwaarden gedefinieerd waaraan de selectie moet voldoen, tevens wordt hiermee bepaald op welke manier meerdere tabellen moeten worden samengevoegd (JOIN).

GROUP BY {veldnaam | integer}
GROUP BY wordt gebruikt om gespecificeerde records te groeperen.

HAVING logische construct
HAVING wordt alleen gebruikt met een GROUP BY commando, om informatie van de groep te specificeren.

ORDER BY {veldnaam | integer} [,..]
ORDER BY sorteert de records van het resultaat. Wanneer een ORDER BY wordt gebruikt in een GROUP BY statement, dan worden de records van de geselecteerde groepen ook gesorteerd. In plaats van een kolomnaam te selecteren, kan ook het kolomnummer worden gebruikt.

UNION [ALL] SELECT-commando
Met UNION kan gebruikt worden om een tweede SELECT-statement op te nemen in de resultaten. Met UNION ALL worden alle records toegevoegd aan het resultaat.

INTERSECT [ALL] SELECT-commando
INTERSECT wordt gebruikt om resultaten van twee of meer SELECT-commando's samen te voegen tot een resultaattabel. De structuur van alle tabellen (de velden in de records) moeten compatibel met elkaar zijn. Wanneer de optie ALL niet wordt gebruikt, worden alleen unieke records getoond in het resultaat.

EXEPT [ALL] SELECT-commando
Afhankelijk SQL-dialect wordt ook wel MINUS of DIFFERENCE gebruikt.
EXCEPT wordt gebruikt voor het samenvoegen van resultaten van twee SELECT-commando's in een resultaattabel. De structuur van alle tabellen (de velden in de records) moeten compatibel met elkaar zijn. Met EXCEPT worden alleen de records toegevoegd die komen uit de eerste SELECT-commando en niet uit de tweede SELECT.

INTO TEMP Tabelnaam
INTO TEMP wordt gebruikt om een tijdelijke database tabel te maken, welke wordt verwijderd nadat de SQL-sessie is uitgevoerd. Deze commando wordt vaak uitgevoerd om de performance van de SQL-query te verbeteren.

CASE

Beschrijving:
Met CASE kan data worden vergeleken met andere data.

Syntaxis:
CASE {WHEN logische construct THEN {veldnaam | formule | CASE }} [...]

Voorbeeld:

```
SELECT klantennummer, naam,
CASE woonplaats <> 'Amsterdam'
THEN LEFT ('Geen Amsterdammer')
ELSE woonplaats
FROM Klanten
WHERE rating >1;
```

Hier worden alle klanten geselecteerd die een hogere rating hebben dan 1 en bovendien wonen in Amsterdam.

UPDATE

Beschrijving:
Verandert waarden van de records in een database tabel.

Syntaxis:
UPDATE Tabelnaam SET {veldnaam=waarde} [,...]
[{WHERE logische construct} | WHERE CURRENT }];

SET {veldnaam=waarde} [,...]
Met SET geef je aan welke velden worden vervangen met de gespecificeerde waarden.

SQL commando's voor ondersteuning data extractie

CREATE VIEW

Beschrijving:
CREATE VIEW maakt een speciale view (virtuele tabel) op een of meer SQL-tabellen in de database alsof het een nieuwe tabel is, alleen deze wordt dan virtueel aangemaakt (in het geheugen). Met een door CREATE VIEW gecreëerde tabel kan alleen met een beperkt deel van de SELECT-commando's gewerkt worden.

Syntaxis:
CREATE VIEW Viewnaam [(veldnaam [,...]] SELECT-commando;

DROP VIEW

Beschrijving:
Verwijderd een view uit de database (virtuele tabel).

Syntaxis:
DROP VIEW Viewnaam;

CREATE PROCEDURE

Beschrijving:
Met CREATE PROCEDURE wordt een functie/programma opgeslagen in de database, welke gebruikt kan worden in SQL-opdrachten of andere programma's/procedures.

Syntaxis:
```
CREATE [DBA] PROCEDURE procedurenaam
        ([{Parameternaam {Veldtype | REFERENCES {BYTE | TEXT}}
        [DEFAULT {waarde | NULL}]} [,...]])
        [RETURNING {Veldtype | REFERENCES {BYTE | TEXT}};]
        [Opdrachten;] [...]
        END PROCEDURE;
```

DBA
DBA geeft aan de database de opdracht om een voor de database administrator procedure.

REFERENCES {BYTE | TEXT}
Geeft aan aan welke gegevens de procedure wordt gekoppeld.

DEFAULT {waarde | NULL}
Geeft een standaardwaarde welke wordt aangeroepen wanneer de procedure wordt aangeroepen zonder parameters.

RETURNING {Veldtype | REFERENCES {BYTE | TEXT}}
RETURNING geeft aan de database de informatie hoeveel waardes en welke type waardes de procedure zal teruggeven.

Opdrachten
Een SQL-opdracht of een andere opdracht (DEFINE, RETURN, IF, FOR, FOREACH, WHILE, SWITCH, etc).

DROP PROCEDURE

Beschrijving:
Verwijderd een opgeslagen procedure.

Syntaxis:
DROP PROCEDURE Procedurenaam;

EXECUTE PROCEDURE

Beschrijving:
Start een procedure.

Syntaxis:
EXECUTE PROCEDURE Procedurenaam;

Overige SQL-commando's

SET ISOLATION TO

Beschrijving:
SET ISOLATION TO DIRTY READ zet de multiuser (meerdere gebruikers) isolatie zo, dat een SQL-commando geen rekening houdt met locked records (geblokkeerde records), ook al zijn de records gelocked in het kader van een UPDATE opdracht.

Syntaxis:
SET ISOLATION TO {DIRTY READ | COMMITED READ | CURSOR STABILITY | REPEATABLE READ};

SET PDQPRIORITY

Beschrijving:
Plaatst de uitvoeringsniveau tot nn. Standaardwaarde is 0 (zonder voorrang), wanneer er geen andere speciale instellingen zijn uitgevoerd.

Syntaxis:
SET PDQPRIORITY nn;

SET LOCK MODE TO

Beschrijving:
Stelt de timeout (tijd welke de databasemanagementsysteem wacht tot de records zal worden geunlocked) tot nn seconden.

Syntaxis:
SET LOCK MODE TO WAIT nn;

LOAD

Beschrijving:
Wordt gebruikt om een plat bestand in een database te laden. Standaard wordt als scheidingsteken | gebruikt.

Syntaxis:
LOAD FROM bestandsnaam [DELIMITER '*']
INSERT TO Tabelnaam

UNLOAD

Beschrijving:
Slaat het resultaat van een SELECT-opdracht op in een plat bestand. Elke record wordt op een nieuwe regel opgeslagen en de velden worden uit elkaar gehouden door een scheidingsteken. Standaardteken binnen een databasemanagementsysteem is |.

Syntaxis:
UNLOAD TO bestandsnaam [DELIMTER '*'] SELECT-commando.

BEGIN WORK

Beschrijving:
Markeert het begin van een transactie, een combinatie SQL-commando's, die teruggedraaid kunnen worden wanneer er problemen optreden.

Syntaxis:
BEGIN WORK;

COMMIT WORK

Beschrijving:
Markeert het einde van een transactie, een combinatie SQL-commando's, die teruggedraaid kunnen worden wanneer er problemen optreden.

Syntaxis:
COMMIT WORK;

ROLLBACK WORK

Beschrijving:
Draait alle SQL-commando's terug van een transactie.

Syntaxis:
ROLLBACK WORK;

SQL operatoren

Normale operatoren

; (Puntkomma, semicon)
De puntkomma wordt gebruikt om twee SQL-commando's te scheiden.

*** (Sterretje)**
Een sterretje is een alias voor alle velden van het geselecteerde tabel.

ALL
ALL wordt gebruikt om alle records van een SELECT-opdracht te selecteren.

ANY
ANY en SOME zijn hetzelfde. Wordt gebruikt om een logische construct op alle dataregels rechts van de operator te gebruiken.

BETWEEN
Controleert of een veld past tussen twee waarden.

Syntaxis:
BETWEEN waarde1 AND waarde2

EXISTS
Controleert het bestaan van een resultaat van een sub-SELECT opdracht.

Syntaxis:
EXIST (SELECT-commando)

IN
Controleert of de uitkomst voldoet aan de gestelde waarden.

Syntaxis:
IN ({Waarde [,...]} | SELECT-commando)

IS NULL
Geeft de waarde TRUE aan op het moment dat een veld leeg is.

Syntaxis:
Veldnaam IS [NOT] NULL

LIKE
LIKE wordt alleen gebruikt in combinatie met CHAR en VARCHAR. Stelt een voorwaarde aan de zoekopdracht.

Syntaxis:
LIKE string

Bijvoorbeeld:
SELECT * FROM Test WHERE testobject LIKE 'T%';
Selecteert alle records van de tabel Test waarbij de velden in kolom testobject begint met de letter T.

Jokertekens:
- _ (underscore) = alle karakters zijn toegestaan op deze plek.
- % (percentageteken) = alle karaktercombinaties zijn toegestaan op deze plek (een of meer karakters).

MATCHES
MATCHES wordt alleen in combinatie met CHAR en VARCHAR gebruikt.

Jokertekens:
- ? (vraagteken) = alle karakters zijn toegestaan op deze plek.
- * (sterretje) = alle karaktercombinaties zijn toegestaan op deze plek (een of meer karakters).
- [karakterset] = alleen de tussen vierkante haken opgenomen karakters toegestaan.

SOME
ANY en SOME zijn hetzelfde. Wordt gebruikt om een logische construct op alle dataregels rechts van de operator te gebruiken.

Logische operatoren

AND
De waarden aan beide zijden van deze operator moeten TRUE (waar) zijn, anders geeft de operator de waarde FALSE (niet waar) terug.

Syntaxis:
logische construct AND logische construct

NOT
Maakt FALSE van True en andersom.

Syntaxis:
NOT logische construct

OR
Betekend dat in ieder geval een van beide waarden aan beide kanten TRUE moet zijn, anders geeft de operator een FALSE terug.

Syntaxis:
logische construct OR logische construct

= (gelijk aan)
Is waar waneer de waardes aan beide zijden van de operator gelijk zijn.

Syntaxis:
formule1 = formule2

> (groter dan)
Is waar wanneer de waarde links van de operator groter is dan de waarde rechts van de operator.

Syntaxis:
formule1 > formule2

>= (groter dan of gelijk aan)
Is waar wanneer de waarde links van de operator groter of gelijk is dan de waarde rechts van de operator.

< (kleiner dan)
Is waar wanneer de waarde links van de operator kleiner is dan de waarde rechts van de operator.

<= (kleiner dan of gelijk aan)
Is waar wanneer de waarde links van de operator kleiner of gelijk is dan de waarde rechts van de operator.

<> (niet gelijk aan)
Is waar wanneer de waarden links en rechts niet gelijk zijn.

Rekenkundige operatoren

Binnen formules zijn alle normale rekenkundige operatoren mogelijk:

- + voor optellen
- - voor aftrekken
- * voor vermenigvuldigen
- / voor delen

Operatoren tabel

Operator	Betekenis	Prioriteit
+	Optellen	0
-	Aftrekken	0
*	Vermenigvuldigen	1
/	Delen	1
+	Toevoegen	2
-	Weghalen	2
=	Gelijk aan	3
<>	Ongelijk aan	3
<	Kleiner dan	3
>	Groter dan	3
<=	Kleiner dan en gelijk aan	3
>=	Groter dan en gelijk aan	3
[NOT] BETWEEN	[niet] tussen	3
IS [NOT] NULL	Heeft wel/geen waarde	3
[NOT] IN	Wel/niet in de lijst	3
NOT	Niet	4
AND	Logische en	5
OR	Logische of	6

Data samenstelling veranderen

AVG

Beschrijving:
Berekend het gemiddelde van alle waarden in de gespecificeerde kolom.

Syntaxis:
AVG (Veldnaam)

COUNT

Beschrijving:
Telt het aantal records van een gespecificeerde kolom. Indien één van de records NULL is, dan wordt deze niet meegeteld, tenzij gebruikt gemaakt wordt van COUNT(*). Bij COUNT(*) worden alle records geteld.

Syntaxis:
COUNT({[DISTINCT] Veldnaam}| *)

MAX

Beschrijving:
Geeft de maximale waarde terug van de gespecificeerde kolom.

Syntaxis:
MAX(Veldnaam)

MIN

Beschrijving:
Geeft de minimale waarde terug van de gespecificeerde kolom.

Syntaxis:
MIN(Veldnaam)

RANGE

Beschrijving:
Berekent het verschil tussen de minimale en maximale waarde van de gespecificeerde kolom.

Syntaxis:
RANGE(Veldnaam)

SUM

Beschrijving:
Telt alle waarden binnen de gespecificeerde kolom op.

Syntaxis:
SUM(Veldnaam)

STDDEV

Beschrijving:
Berekent de standaard afwijking van de waarden binnen de gespecificeerde kolom.

Syntaxis:
STDEV(Veldnaam)

VARIANCE

Beschrijving:
Berekent de verschillen van de waarden binnen de gespecificeerde kolom.

Syntaxis:
VARIANCE(Veldnaam)

Rekenkundige functies in SQL

ABS

Beschrijving:
Berekent de absolute waarde van het argument: if x<0 then (-1)*x else x.

Syntaxis:
ABS(Formule)

CEIL

Beschrijving:
Rond de waarde af naar boven.

Syntaxis:
CEIL(Formule)

FLOOR

Beschrijving:
Rond de waarde af naar beneden.

Syntaxis:
FLOOR(Formule)

HEX

Beschrijving:
Geeft de uitkomst terug in hexadecimale waarde.

Syntaxis:
HEX(Formule)

MOD

Beschrijving:
Geeft de restwaarde terug bij deling van twee argumenten.

Syntaxis:
MOD(Formule1, Formule2)

ROUND

Beschrijving:
Rond het getal af naar de dichtstbijzijnde drijvende komma getal, afhankelijk wat er is gespecificeerd (aantal decimalen). Indien er geen aantal decimalen is gedefinieerd, wordt het getal afgerond op hele getallen.

Syntaxis:
ROUND(Formule [,Integer])

SQRT

Beschrijving:
Berekend de vierkantswortel uit van het argument.

Syntaxis:
SQRT(Formule)

TAN

Beschrijving:
Geeft de tangens waarde weer van een argument.

Syntaxis:
TAN(Formule)

TRUNC

Beschrijving:
Rondt het getal af in de dichtstbijzijnde integer (hele getal).

Syntaxis:

TRUNC(Formule)

String functies in SQL

LENGTH

Beschrijving:
Geeft het aantal karakters in de string terug.

Syntaxis:
LENGTH(Formule)

LEFT

Beschrijving:
Geeft het aantal karakters terug vanaf de linkerkant (begin) van de string van formule1 terug tot aan de waarde dat gespecificeerd is in formule2.

Syntaxis:
LEFT(Formule1, Formule2)

RIGHT

Beschrijving:
Geeft het aantal karakters terug vanaf de rechterkant (einde) van de string van formule1 terug tot aan de waarde dat gespecificeerd is in formule2.

Syntaxis:
RIGHT(Formule1, Formule2)

SUBSTR

Beschrijving:
Geeft de karakters terug uit Formule1, welke begint op positie van Formule1 die gespecificeerd is in Formule2 en het aantal wordt bepaald in de waarde van Formule3.

Syntaxis:
SUBSTR(Formule1, Formule2, Formule3)

TRIM

Beschrijving:
TRIM verwijdert alle spaties aan het begin en eind van de string.

Syntaxis:
TRIM(Formule)

VALUE

Beschrijving:
Zet de string om in een getal.

Syntaxis:
VALUE(Formule)

Datum- en tijd-functies van SQL

CURRENT

Beschrijving:
Geeft de huidige datum en tijd weer.

Syntaxis:
CURRENT

DATE

Beschrijving:
Zet de waarde van de string om in een datum

Syntaxis:
DATE(Formule)

DAY

Beschrijving:
Geeft de dag aan uit een datum waarde.

Syntaxis:
DAY(DATE)

EXTEND

Beschrijving:
Plaatst de beschrijving van de datum- en tijdwaarden, waarbij het eerste argument de datum is die omgezet moet worden, het tweede argument definieert de beschrijving in de vorm van de beginwaarde en de eindwaarde.

Syntaxis:
EXTEND(Date [,beginwaarde TO eindwaarde])

Voorbeeld:
SELECT EXTEND(Date, MONTH TO YEAR) AS TIJD

MDY

Beschrijving:
Definieert een datum bestaande uit de componenten: maand, dag, jaar

Syntaxis:
MDY(Formule1, Formule2, Formule3)

MONTH

Beschrijving:
Haalt uit de datum de maand.

Syntaxis:
MONTH(DATE)

WEEKDAY

Beschrijving:
Berekent en geeft de weekdag terug van de gespecificeerde datum.

Syntaxis:
WEEKDAY(Date)

YEAR

Beschrijving:
Haalt het jaar uit de datum.

Syntaxis:
YEAR(Date)

Overige functies binnen SQL

DBSERVERNAME

Beschrijving:
Geeft de naam weer van de database server.

Syntaxis:
DBSERVERNAME

NVL

Beschrijving:
Vervangt alle waarden die NULL zijn met de aangegeven waarde.

Syntaxis:
NVL(Veldnaam [, Waarde])

SITENAME

Beschrijving:
Geeft de naam van de database server die op dat moment door de database wordt gebruikt.

Syntaxis:
SITENAME

USER

Beschrijving:
Geeft de naam van de aangemelde gebruiker (op de database).

Syntaxis:
USER