

Database

Uit Wikipedia, de vrije encyclopedie

Een **database**, **gegevensbank** of **databank** is een digitaal opgeslagen archief, ingericht met het oog op flexibele raadpleging en gebruik. Databases spelen een belangrijke rol voor het archiveren en actueel houden van gegevens bij onder meer de overheid, financiële instellingen en bedrijven, in de wetenschap, en worden op kleinere schaal ook privé gebruikt.

Inhoud

- 1 Inleiding
- 2 Gegevensbank
- 3 Belang van databases
- 4 Koppeling van databases
- 5 Datamining
- 6 Privacy
- 7 Databasecorruptie
- 8 Geografisch Informatiesysteem
- 9 Zie ook

Inleiding

Het woord *database* wordt voor verschillende begrippen gebruikt:

1. de opgeslagen gegevens als zodanig.
2. de wijze waarop de gegevens zijn opgeslagen, zie datamodel.
3. de software waarmee databases kunnen worden aangemaakt en benaderd, zie databasemanagementsysteem (DBMS).

Dit artikel gaat over de eerstgenoemde betekenis van *database*.

Gegevensbank

Een database moet aan de volgende minimale (CRUD) voorwaarden voldoen om als database gezien te worden:

1. Gegevens moeten eenvoudig duurzaam kunnen worden opgeslagen (Create).
2. Gegevens moeten eenvoudig kunnen worden opgezocht en doorzocht (Read).
3. Gegevens moeten onderhouden kunnen worden (Update).
4. Gegevens moeten verwijderd kunnen worden zonder dat dat de werking van dat systeem nadelig beïnvloedt (Delete).

Om aan deze voorwaarden te kunnen voldoen is een essentiële regel belangrijk: De database moeten integer zijn:

1. Gegevens moeten consistent zijn en mogen bijvoorbeeld niet dubbel worden opgeslagen. Dit

betekent ook dat de samenhang of relatie met andere gegevens moeten (blijven) kloppen.

2. De gegevens moeten geautoriseerd toegevoegd, onderhouden of verwijderd worden.

Het opstellen van een verantwoorde manier om gegevens structureren is een vak op zich, het gaat in een database meer om de structuur die in bijvoorbeeld een adressenbestand niet voorkomt. (De relatie geadresseerde met adres wordt gemakshalve niet gezien. Meerdere geadresseerden per adres of meerdere adressen voor een geadresseerde?) Daarom is een adressenbestand op zichzelf een onjuist voorbeeld. Dit wordt vaak door spreadsheetgebruikers en zelfs programmeurs onderschat. De methoden zijn samen te vatten als normaliseren. Daarover zijn theoretische modellen ontwikkeld en vele boeken geschreven. Belangrijke namen op dit gebied zijn Charles Bachman en Ted Codd en Chris Date. Zie verder datamodel.

Een database is meer dan een gedigitaliseerd archief, een meerwaarde is dat de gegevens in een database zodanig zijn opgeslagen dat deze gegevens kwalitatief zijn, goed doorzoekbaar zijn, samenhang hebben en in relatie met andere items staan. In het relationele model worden onderdelen in een aparte *kolom* gezet, maar wel in dezelfde *rij*, zodat het duidelijk is dat deze onderdelen bij elkaar horen en eigenschappen van het onderhavige item zijn. Soms is het praktisch om gegevens uit te breiden om de doorzoekbaarheid te vergroten, bijvoorbeeld door beschrijvende teksten, één of meerdere categorieën te gebruiken of er trefwoorden aan toe te kennen.

Belang van databases

Databases zijn een essentieel onderdeel van de informatiemaatschappij, steeds meer gegevens worden in een database opgeslagen. Het functioneren van de overheid, bedrijven en wetenschap is tegenwoordig zonder databases ondenkbaar.

Steeds meer gegevens worden ook via internet bereikbaar gemaakt. Vanaf halverwege de jaren 1990 worden er speciale programmeertalen ontwikkeld juist om de communicatie tussen databases en de internetgebruiker mogelijk te maken. Ook zijn er componenten ontwikkeld die functioneren als intermedium tussen programma en database, onder andere ODBC en JDBC.

Ook zoekmachines maken gebruik van een database, door de pagina's op internet te *indexeren*. De gebruiker van een zoekmachine zoekt niet direct op internet, maar in de *index* die is aangemaakt.

In de wetenschap worden databases veel gebruikt om meetgegevens of experimentele gegevens in op te slaan. Om statistische conclusies uit deze gegevens te kunnen trekken schiet de software van veel DBMSen te kort. Voor statistische analyse van gegevens en relaties tussen gegevens is een digitaal rekenblad veel beter geschikt. Programma's als SAS en SPSS zijn daarentegen prima geschikt om statistische analyses te doen op grote groepen gegevens, die zelfs de capaciteit van programma's als MS SQL Server te boven gaan.

Koppeling van databases

Verschillende databases die gedeeltelijk overlappende gegevens bevatten kunnen worden gekoppeld. Technisch is dat niet altijd even gemakkelijk, maar het principe is eenvoudig: als er twee databases zijn, waarbij database X de belastinggegevens bevat van personen en database Y informatie over de banktegoeden van personen, leg dan een relatie tussen de personen die in beide databases staan, zodanig dat van de personen die in beide databases voorkomen, de belastinggegevens naast de gegevens over banktegoeden kunnen worden gelegd. Dit kan alleen als de personen in beide databases precies dezelfde

naam of hetzelfde nummer hebben. Het gebruik van een algemeen persoonsnummer zoals het Nederlandse Burgerservicenummer vereenvoudigt dan ook de koppeling van databases met persoonsgegevens.

Datamining

Datamining is een term die gebruikt wordt om extra informatie te halen uit bestaande databases. Het gaat daarbij vaak om statistische informatie. Een bedrijf dat een database heeft van klanten en hun bestellingen zou bijvoorbeeld kunnen nagaan in welke gebieden de klanten wonen die het meeste afnemen, en op basis daarvan strategische besluiten kunnen nemen.

Privacy

Het toenemend gebruik van databases (en de koppeling daarvan) heeft ook een negatieve kant: de privacy van personen komt in gevaar. Zeker als het gaat over het gebruik van elektronische communicatie en het koppelen daarvan aan persoonsgegevens is het mogelijk om bijzonder veel informatie over personen te verzamelen. Een voorbeeld hiervan is het internationale spionageproject ECHELON, dat gebouwd is om dagelijks 3 miljard afzonderlijke elektronische berichten op te vangen, te analyseren en op te slaan. Maar ook een marketingbedrijf als Doubleclick verzamelt dagelijks een grote hoeveelheid informatie over het gedrag van gebruikers op het Internet, informatie die in eerste instantie is gebonden aan een IP-adres of cookies. Als een cookie (of IP-adres) kan worden geassocieerd met een e-mailadres en vervolgens met een persoon en een adres, zijn dergelijke gegevens op de markt veel geld waard.

In een aantal landen (waaronder België, Nederland en de meeste EU-lidstaten) zijn er echter strenge wetten die de privacy van personen en hun gegevens moeten garanderen.^[1] In andere landen (zoals de Verenigde Staten) zijn de wetten minder strikt. Zo zijn in verschillende staten de databanken met informatie over criminelen, pedofielen, echtscheidingen, huwelijken, etc. publiek toegankelijk.^{[2][3][4]} Het feit dat niet in alle landen de privacy-wetgeving dezelfde is heeft ook gevolgen. Zo wordt in sommige gevallen informatie die in een bepaald land niet publiek gemaakt mag worden gewoon op servers in een ander land gezet waar dit niet illegaal is.

Databasecorruptie

Omdat databases vaak langdurig in gebruik zijn, en tal van mensen wijzigingen aanbrengen, ontstaan onvermijdelijk fouten: administratieve fouten, nalatigheid, onwilligheid of onmacht van klanten bij het invullen van formulieren of zelfs fraude en vandalisme. Hierdoor komt het voor dat de informatie die de database bevat niet correct of "vervuild" is. Dit verschijnsel wordt ook wel *databasecorruptie* genoemd en speelt onder andere een grote rol bij bevolkingsregisters, waar frauduleuze inschrijvingen een probleem zijn.

Geografisch Informatiesysteem

Een geografisch informatiesysteem (GIS) kan beschouwd worden als een database voor geografische informatie, dit geldt voor alle drie de betekenissen van een database.

Zie ook

- Database publishing
- Databasedump

Referenties

1. CBPL. "Nationale wetgeving." (<http://web.archive.org/web/20120315040802/http://www.privacycommission.be/nl/legislation/national>), 2009.
2. FamilyWatchDog. "familywatchdog.us" (<http://familywatchdog.us/>), 2009. Geraadpleegd op 2 april 2013.
3. SearchSystems. "Public Records Directory" (<http://publicrecords.searchsystems.net/>), 2008. Geraadpleegd op 2 april 2013.
4. National Alert Registry. "Registered Offenders List" (<http://www.registeredoffenderslist.org/>), 2009. Geraadpleegd op 2 april 2013.

Databases

Databasemanagementsysteem · Databasemodel · Databasenormalisatie · Referentiële integriteit · Relationele algebra · Relationele database · Relationeel model

Begrippen: Database · ACID · CRUD · Null · Vreemde sleutel · Primaire sleutel · Kandidaatsleutel · Axioma's van Armstrong

Objecten: Tabel · Relatie · View · Transactie · Trigger · Index · Opgeslagen procedure · Cursor · Partitie

SQL: Select · Insert · Update · Merge · Delete · From · Join · Union · Create · Drop · Commit · Rollback · Truncate · Alter · Where · DDL · DML · DCL

Producten: Relationele databases

Overgenomen van "<http://nl.wikipedia.org/w/index.php?title=Database&oldid=42824734>"

Categorieën: Database | Databaseserver | Dataopslag

-
- Deze pagina is het laatst bewerkt op 24 dec 2014 om 20:22.
 - De tekst is beschikbaar onder de licentie Creative Commons Naamsvermelding/Gelijk delen, er kunnen aanvullende voorwaarden van toepassing zijn. Zie de gebruiksvoorwaarden voor meer informatie.
Wikipedia® is een geregistreerd handelsmerk van de Wikimedia Foundation, Inc., een organisatie zonder winstoogmerk.

Databasenormalisatie

Uit Wikipedia, de vrije encyclopedie

Databasenormalisatie is een techniek bij het ontwerpen van databases. Ze dient twee doelen: het spaarzaam omgaan met opslagruimte en het vermijden van meervoudige vastlegging van dezelfde data, een potentieel bron van fouten.

Inhoud

- 1 Relationale databases
- 2 Normaalvormen
 - 2.1 Niet-genormaliseerde gegevens
 - 2.2 Eerste normaalvorm (1NF)
 - 2.2.1 Een voorbeeld
 - 2.3 Tweede normaalvorm (2NF)
 - 2.3.1 Een voorbeeld
 - 2.4 Derde normaalvorm (3NF)
 - 2.4.1 Een voorbeeld
 - 2.5 Boyce-Codd-normaalvorm (BCNF)
 - 2.5.1 Een voorbeeld
 - 2.6 Vierde normaalvorm (4NF)
 - 2.7 Vijfde normaalvorm (5NF)
- 3 Referenties
- 4 Zie ook

Relationale databases

De techniek van databasenormalisatie wordt in het bijzonder gebruikt in relationele databases. Het woord "relationeel" geeft aan dat de relatie tussen de gegevens deel uit maakt van de database. In computerdatabases worden de relaties tussen de gegevens bewaakt door een software-tussenlaag, het RDBMS.

Normaalvormen

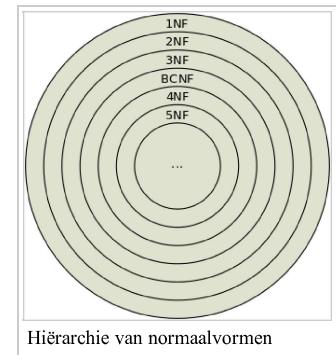
Er bestaan verschillende manieren om dubbelingen uit databases te verwijderen, dit proces wordt normaliseren genoemd, de oplossingen noemt men normaalvormen. De normalisatie leidt ertoe dat elke regel in elke tabel met behulp van een unieke identificatie, een sleutel, opgevraagd kan worden. Elke normaalvorm stelt bepaalde eisen aan de manier waarop de gegevens zijn opgeslagen (zoals eisen aan de geldende functionele afhankelijkheden).

Er zijn meerdere normaalvormen bekend, maar de meest gebruikte zijn de zogenaamde eerste, tweede, derde en vierde normaalvorm. De gegevens staan in een bepaalde normaalvorm wanneer aan een aantal voorgeschreven voorwaarden voldaan is. Gegevens staan bijvoorbeeld in de tweede normaalvorm als en slechts als ze voldoen aan de eerste normaalvorm en aan een aantal extra regels.

Ted Codd formuleerde het idee van normalisatie in *A Relational Model of Data for Large Shared Data Banks*^[1] in 1970.

There is, in fact, a very simple elimination procedure which we shall call normalization. Through decomposition nonsimple domains are replaced by "domains whose elements are atomic (nondecomposable) values."*

* Zijn term *elimineren* is enigszins misleidend: er gaat geen informatie verloren tijdens normalisatie; hij doelde waarschijnlijk op de wiskundige betekenis van het elimineren van complexiteit of redundantie



Hiërarchie van normaalvormen

De eerste drie normaalvormen (1NF, 2NF en 3NF) werden gedefinieerd door Codd in *Further normalization of the Data Base Relational Model*^[2]

Alle genormaliseerde gegevens staan minstens in 1NF. Sommige gegevens staan ook in 2NF, sommige zelfs in 3NF. Codd gaf aan dat gegevens in 2NF wenselijker waren dan deze in 1NF, 3NF was nog wenselijker. De ontwerper van de database zou dus moeten streven naar gegevens in 3NF.

Codds oorspronkelijke definitie van 3NF bleek later niet volmaakt. De definitie werd herbekeken en versterkt door Boyce en Codd in *Recent Investigations into Relational Data Base Systems*^[3]. Gegevens in 3NF in deze nieuwe definitie voldeden ook aan de oude definitie, maar gegevens die aan 3NF voldeden volgens de oude definitie voldeden niet noodzakelijk aan de nieuwe. De nieuwe definitie was dus sterker dan de oude en werd later de *Boyce/Codd normaalvorm* genoemd als een versterking van de voorwaarden van de oude 3NF.

Later introduceerde Ron Fagin nog enkele sterke normaalvormen. In *Multivalued Dependencies and a New Normal Form for Relational Databases*^[4] definieerde hij een nieuwe vierde normaalvorm (in die tijd werd de latere BCNF nog steeds de derde normaalvorm genoemd). In *Normal Forms and Relational Database Operators*^[5] definieerde hij nog een nieuwe normaalvorm, de *projection-join normal form* (PJ/NF) of vijfde normaalvorm.

Niet-genormaliseerde gegevens

Ieder niet gestructureerd gegevensbestand is in de *nulde normaalvorm* (0NF) of niet-genormaliseerd. Gegevens van verschillende soorten kunnen op elke regel voorkomen, bijgevolg kunnen deze niet in kolommen worden opgedeeld.

Een voorbeeld:

Verg. BRZZ, laptop mee!
di: combi naar garage
Jos jarig?

Eerste normaalvorm (1NV)

Elke tabel met gegevens die voldoet aan de definitie van een relatie is in de eerste normaalvorm (1NV). Wanneer gegevens aan een relatie voldoen zijn ze dus reeds genormaliseerd.

- elk attribuut is atomair, en bevat dus één enkele waarde (bijvoorbeeld een telefoonnummer-attribuut mag geen twee telefoonnummers bevatten); indien een attribuut meerdere waarden bevat zouden deze waarden in een andere tabel moeten worden ondergebracht.
- geen enkel attribuut wordt herhaald
- alle attributen blijven constant in de tijd

Een voorbeeld

Ter vergelijking: tabel uit het personeelsbestand van een fictief bedrijf in 0NF:

Naam	Adres	Salaris	Afdeling
P.Jansen	Stationsweg 14 Venlo	Schaal 2: € 1503,00	Boekhouding
W.de Vries	't Steegke 23 Epe	Schaal 1: € 1245,00	Kantine
T.Oud	Dorpsstraat 1 Ons Dorp	Schaal 3: € 1789,00	Boekhouding

Diezelfde tabel uit het personeelsbestand van een fictief bedrijf in 1NF:

Naam	Voorvoegsel	Initialen	Straat	Nr	Plaats	Afdeling	Schaal	Salaris
Jansen	NULL	P.	Stationsweg	14	Venlo	Boekhouding	2	1503
Vries	de	W.	't Steegke	23	Epe	Kantine	1	1245
Oud	NULL	T.	Dorpsstraat	1	Ons Dorp	Boekhouding	3	1789

Tweede normaalvorm (2NV)

Een relatie is in 2NF als alle attributen die niet in de sleutel zijn opgenomen, functioneel afhankelijk zijn van de gehele sleutel (geen gedeeltelijke afhankelijkheid). Een relatie met één attribuut als sleutel is automatisch in 2NF.

- voldoet aan de eerste normaalvorm
- alle niet-sleutelattributen zijn volledig functioneel afhankelijk van de primaire sleutel.

Een voorbeeld

Ter vergelijking: tabel uit het personeelsbestand van een fictief bedrijf in 1NF:

Naam	Voorvoegsel	Initialen	Straat	Nr	Plaats
Jansen	NULL	P.	Stationsweg	14	Venlo
Vries	de	W.	't Steegke	23	Epe
Vries - Zeilstra	de	A.	't Steegke	23	Epe
Oud	NULL	T.	Dorpsstraat	1	Ons Dorp

Diezelfde data uit het personeelsbestand van een fictief bedrijf in 2NF:

Werknemer					Adres			
Werknemer_ID	Naam	Voorvoegsel	Initialen	Adres_ID	Adres_ID	Straat	Nr	Plaats
77	Jansen	NULL	P.	34	34	Stationsweg	14	Venlo
78	Vries	de	W.	45	45	't Steegke	23	Epe
79	Vries - Zeilstra	de	A.	45	67	Dorpsstraat	1	Ons Dorp
80	Oud	NULL	T.	67				

Het attribuut *Adres_ID* is nu een vreemde sleutel die verwijst naar de primaire sleutel in de tabel *Adres*. Adres is ondergebracht in een nieuwe tabel, omdat een adres niet onlosmakelijk verbonden is aan een persoon. Er kunnen immers meerdere personen op een adres wonen en mensen kunnen verhuizen.

Derde normaalvorm (3NV)

Een relatie is in 3NF indien ze in 2NF is en geen transitieve afhankelijkheid kent.

- voldoet aan de tweede normaalvorm
- alle attributen die niet tot een sleutel behoren hangen niet af van een niet-sleutelattribuut

Een voorbeeld

Ter vergelijking: tabellen uit het personeelsbestand van een fictief bedrijf in 2NF:

Werknemer					Adres					
Werknemer_ID	Naam	Voorvoegsel	Initialen	Adres_ID	Adres_ID	Straat	Nr	Plaats	Provincie	Land
77	Jansen	NULL	P.	34	34	Stationsweg	14	Venlo	Limburg	Nederland
78	Vries	de	W.	45	45	't Steegke	23	Epe	Gelderland	Nederland
79	Vries - Zeilstra	de	A.	45	47	Straatweg	1	Apeldoorn	Gelderland	Nederland
80	Dijk	NULL	D.	47						

Niet-sleutel *Land* hangt hier af van niet-sleutelattribuut *Provincie* en is in het geval van Gelderland dus redundant opgeslagen.

Diezelfde data uit het personeelsbestand van een fictief bedrijf in 3NF:

Werknemer					Adres				Provincie			
Werknemer_ID	Naam	Voorvoegsel	Initialen	Adres_ID	Adres_ID	Straat	Nr	Plaats	Provincie_ID	Provincie_ID	Provincie	Land
77	Jansen	NULL	P.	34	34	Stationsweg	14	Venlo	12	12	Limburg	Nederland
78	Vries	de	W.	45	45	't Steegke	23	Epe	6	6	Gelderland	Nederland
79	Vries - Zeilstra	de	A.	45	47	Straatweg	1	Apeldoorn	6			
80	Dijk	NULL	D.	47								

In dit voorbeeld is geen enkel niet-sleutelattribuut (grijze cellen) afhankelijk van een niet-sleutelattribuut.

Boyce-Codd-normaalvorm (BCNV)

Een relatie is in BCNF (Boyce-Codd Normal Form) als elke determinant een kandidaatssleutel is.

- voldoet aan de derde normaalvorm
- er zijn geen transitieve afhankelijkheden, dus geen enkele sleutel bevat informatie over een andere sleutel binnen dezelfde tabel, behalve over de gehele primaire sleutel

Een voorbeeld

Een tabel uit een database met aannemers. In dit voorbeeld wordt aangenomen dat een werknemer niet bij twee aannemers hetzelfde werk mag doen, maar dat hij wel verschillende werkzaamheden mag uitvoeren bij verschillende bedrijven.

Aannemers

Naam	Werk	Aannemer
Janssen	Loodgieter	B.V. De Loden Gieter
Zeilstra	Loodgieter	B.V. De Loden Gieter
Zeilstra	Elektromonteur	Elektrobedrijf B.V.

In deze tabel zijn twee mogelijkheden voor primaire sleutels, namelijk de combinatie van *Naam* en *Aannemer* en de combinatie van *Naam* en *Werk*. In beide gevallen zal het overgebleven niet-sleutelattribuut informatie bevatten over een deel van de primaire sleutel en niet over de gehele primaire sleutel.

Diezelfde data uit een database met aannemers in BCNV:

Medewerkers		Aannemers	
Naam	Aannemer	Aannemer	Werk
Janssen	B.V. De Loden Gieter	B.V. De Loden Gieter	Loodgieter
Zeilstra	B.V. De Loden Gieter	Elektrobedrijf B.V.	Elektromonteur
Zeilstra	Elektrobedrijf B.V.		

In dit voorbeeld bevat ieder niet-sleutelattribuut enkel informatie over de gehele primaire sleutel (blauw).

Vierde normaalvorm (4NV)

Een relatie is in 4NF als ze in BCNF staat en geen meerwaardige afhankelijkheden kent.

- voldoet aan de Boyce-Codd-normaalvorm
- bevat geen enkele meervoudige functionele afhankelijkheid

Vijfde normaalvorm (5NV)

- voldoet aan de vierde normaalvorm

- elke afhankelijkheid bevat een sleutel voor de relatie

Referenties

1. CODD, E.F., *A Relational Model of Data for Large Shared Data Banks* (<http://www.acm.org/classics/nov95/toc.html>), in *Communications of the ACM* **13** (6), pp 377-387, juni 1970
2. CODD E.F., *Further normalization of the Data Base Relational Model* in RUSTIN, RANDALL J. (ed.), *Data Base Systems, Courant Computer Science Symposia Series 6*. Englewood Cliffs, N.J., Prentice-Hall, 1972
3. CODD, E.F., *Recent Investigations into Relational Data Base Systems*, Proc. IFIP Congress, Stockholm, 1974
4. FAGIN, R., *Multivalued Dependencies and a New Normal Form for Relational Databases*, ACM Transactions on Database Systems 2 (3), sept 1977
5. FAGIN, R., *Normal Forms and Relational Database Operators*, ACM SIGMOD International Conference on Management of Data, May 31-June 1, 1979, Boston, Mass.

Zie ook

- Denormalisatie

Databases

Databasemanagementsysteem · Databasemodel · **Databasenormalisatie** · Referentiële integriteit · Relationele algebra · Relationele database · Relationeel model

Begrippen: Database · ACID · CRUD · Null · Vreemde sleutel · Primaire sleutel · Kandidaatsleutel · Axioma's van Armstrong

Objecten: Tabel · Relatie · View · Transactie · Trigger · Index · Opgeslagen procedure · Cursor · Partitie

SQL: Select · Insert · Update · Merge · Delete · From · Join · Union · Create · Drop · Commit · Rollback · Truncate · Alter · Where · DDL · DML · DCL

Producten: Relationele databases

Overgenomen van "<http://nl.wikipedia.org/w/index.php?title=Databasenormalisatie&oldid=42824796>"

Categorieën: Database | Dataopslag | Modelvorming

-
- Deze pagina is het laatst bewerkt op 24 dec 2014 om 20:38.
 - De tekst is beschikbaar onder de licentie Creative Commons Naamsvermelding/Gelijk delen, er kunnen aanvullende voorwaarden van toepassing zijn. Zie de gebruiksvoorwaarden voor meer informatie.
- Wikipedia® is een geregistreerd handelsmerk van de Wikimedia Foundation, Inc., een organisatie zonder winstoogmerk.

SQL

Uit Wikipedia, de vrije encyclopedie

SQL (Structured Query Language) is een ANSI/ISO-standaardtaal voor een relationeel databasemanagementsysteem (DBMS). Het is een gestandaardiseerde taal die gebruikt kan worden voor taken zoals het bevragen en het aanpassen van gegevens in een relationele database. SQL kan met vrijwel alle moderne relationele databaseproducten worden gebruikt.

SQL is een vierde-generatie-taal (G4-taal) omdat ze niet imperatief maar declaratief is, zoals Prolog.

Inhoud

- 1 Beschrijving
- 2 Werking
 - 2.1 Een tabel maken
 - 2.2 Een tabel verwijderen
 - 2.3 Een tabel vullen
 - 2.4 Een tabel lezen
 - 2.5 Gegevens bijwerken
- 3 SQL-uitdrukkingen
- 4 Transacties
- 5 Zie ook
- 6 Externe links

Beschrijving

SQL is gebaseerd op de relationele algebra en werd in de loop van de jaren zeventig ontwikkeld door IBM (San José). Sinds het ontstaan van SQL hebben reeds vele verschillende SQL-versies het levenslicht gezien. Pas in de loop van de jaren 80 werd SQL gestandaardiseerd. Tegenwoordig gebruiken de meeste RDBMS'en ten minste SQL-92.

Bij het beschouwen van de verschillende SQL-implementaties moeten we vaststellen dat bijna elk DBMS zijn eigen extra functies heeft toegevoegd aan SQL-92. Dit maakt dat computerprogramma's waarbij de databaseinterface werd geschreven met behulp van SQL niet noodzakelijk zonder problemen kunnen worden gemigreerd van de ene naar de andere SQL-compatibele database. In vele gevallen werd door de ontwikkelaar van de software wel een of andere SQL-functie gebruikt die enkel maar bestaat in de SQL-implementatie van één specifiek DBMS.

In eerste instantie werd SQL ontwikkeld als een vraagtaal voor de eindgebruiker. Het idee was dat businessmanagers SQL zouden gebruiken om bedrijfsgeschiedenis te analyseren. Achteraf is gebleken dat SQL te complex is om door eindgebruikers toegepast te worden. Het gebruik van SQL impliceert immers een volledige kennis van de structuur van de te ondervraagde database. Tegenwoordig wordt SQL vrijwel uitsluitend door tussenkomst van een applicatie gebruikt. De programmeur van de applicatie benadert de database met SQL via een application programming interface (API), zoals ODBC of ADO (Windows), JDBC (Java) of een productspecifieke API. SQL is dus in essentie omgevormd van een taal voor eindgebruikers tot een brug tussen applicaties en databases.

SQL kan worden opgedeeld in drie onderdelen: de Data Manipulation Language (DML), de Data Control Language (DCL) en de Data Definition Language (DDL).

Werking

SQL maakt voor de communicatie met het DBMS gebruik van zogenaamde query's. Een query is een ASCII-tekenreeks en is een opdracht die naar het DBMS wordt verstuurd. Het DBMS zal de opdracht interpreteren en uitvoeren en stuurt eventueel gegevens terug naar het opdrachtgevende programma.

Een SQL-query ziet er bijvoorbeeld als volgt uit:

```
SELECT *
FROM tblKlanten
WHERE tblKlanten.krediet < 0;
```

De betekenis van bovenstaande query is als volgt:

- **SELECT**: hierachter wordt geplaatst welke velden (kolommen) worden geselecteerd; * betekent 'alle velden'.
- **FROM**: hierachter komt de naam van de tabel, in dit geval **tblKlanten**.
- **WHERE**: hierachter komen veldnamen met waarden waaraan de velden moeten voldoen.
In dit geval: alle records waarvan het veld **krediet** in de tabel **tblKlanten** kleiner is dan 0.

Met SQL is het mogelijk om tabellen aan te maken, te wijzigen, te vullen en te verwijderen.

Een tabel maken

```
CREATE TABLE tabelnaam (
    veldNaam1 veldtype1 [NOT NULL] [PRIMARY KEY | UNIQUE]
    ,veldNaam2 veldtype2 [NOT NULL] [PRIMARY KEY | UNIQUE]
    [, ...]);
```

Een tabel verwijderen

```
DROP TABLE tabelnaam;
```

Een tabel vullen

(zie ook: Insert (SQL))

```
INSERT INTO tabelnaam [veldnaam1 [,veldnaam2 [, ...,]]]
VALUES (waarde1[,waarde2[,...]]);
```

Een tabel lezen

(zie ook: Select (SQL))

```
SELECT veldna(a)m(en)
FROM Tabelnaam
[WHERE conditie]
[GROUP BY veldnaam [, veldnaam ...]]
[HAVING conditie2]
[ORDER BY veldnaam [ASC | DESC] [, veldnaam [ASC | DESC] ...]];
```

Gegevens bijwerken

(zie ook: Update (SQL))

```
UPDATE <tabel-naam> SET veldnaam=waarde, veldnaam2=waarde2 WHERE conditie;
```

SQL-sleutelwoorden staan in HOOFDletters, niet verplichte code tussen [en]

SQL-uitdrukkingen

uitdrukking	beschrijving	query
CREATE DATABASE	nieuwe database	CREATE DATABASE database_naam;
CREATE TABLE	nieuwe tabel	CREATE TABLE tabel_naam(kolom1 text, kolom2 integer [, ...]);
INSERT	nieuwe gegevens in de tabel toevoegen	INSERT INTO tabelnaam (kolom1, kolom2 [, ...]) VALUES (waarde1, waarde2 [, ...]);
SELECT	gegevens uit een tabel selecteren	SELECT kolom1, kolom2 [, ...] FROM tabelnaam WHERE conditie;
UPDATE	bestaande records wijzigen	UPDATE tabelnaam SET kolom1 = waarde1 [, ...] WHERE conditie;
DELETE	records wissen uit de database	DELETE FROM tabelnaam WHERE conditie;
TRUNCATE	ALLE records wissen uit een tabel	TRUNCATE TABLE tabelnaam;
DROP	tabel geheel wissen uit de database	DROP TABLE tabelnaam;

Transacties

Een transactie bestaat uit een reeks van een of meer databasewijzigingen. De transactie maakt mogelijk dat zo'n reeks wijzigingen ofwel allemaal, ofwel geen van allen worden doorgevoerd.

Neem als voorbeeld de database van een bank, waarbij bij een overboeking eerst het saldo van rekening A met een SQL-opdracht wordt verlaagd, en daarna het saldo van rekening B met een SQL-opdracht wordt verhoogd. Als er een fout plaatsvindt nadat het saldo op rekening A is verlaagd, maar voordat het saldo op rekening B is verhoogd, 'verdwijnt' het bedrag. De database is dan "inconsistent". Deze ongewenste toestand kan worden voorkomen door de twee SQL-opdrachten binnen een transactie af te werken.

Een transactie kan expliciet worden gestart (`BEGIN WORK` of `BEGIN TRANSACTION`) of impliciet bij de eerste lees- of schrijfopdracht sinds de voorgaande transactie (of sinds het begin van de sessie).

Er zijn twee commando's om een transactie af te sluiten:

- **COMMIT:** De transactie wordt doorgevoerd. Als het commit-commando door de database succesvol wordt uitgevoerd dan zijn de bewerkingen definitief in de database doorgevoerd. Sommige statements bevatten een impliciete commit. Ook is het mogelijk een impliciete commit te implementeren.
- **ROLLBACK of ABORT:** De transactie wordt niet doorgevoerd. De gegevens houden de waarde die ze hadden vóór de transactie. Bij sommige databases wordt impliciet een rollback uitgevoerd als er een fout optreedt gedurende een transactie.

De syntaxis en manier van werken verschilt enigszins per database systeem zoals Oracle, DB2, PostgreSQL en andere.

Zo gaat in Oracle het uitvoeren van een DDL-commando impliciet gepaard aan een commit, en zo'n commando kan dus niet worden teruggedraaid (geROLLBACKed). In PostgreSQL daarentegen kunnen DDL-commando's in een transactie worden uitgevoerd en vervolgens ongedaan gemaakt door ROLLBACK.

```
BEGIN WORK;
UPDATE tabel SET kolom1 = waarde1, kolom2 = waarde2 WHERE kolomID = eenID;
COMMIT;
```

Zie ook

- **INSERT-commando**
- **SELECT-commando**
- **UPDATE-commando**
- **DELETE-commando**
- **Transactie**
- **SQL-injectie**

Externe links

- **(en) SQLzoo.net** (<http://sqlzoo.net/>): SQL leren met behulp van doe-opdrachten
- **(en) SQL Fiddle** (<http://www.sqlfiddle.com/>): SQL uitproberen met keuze uit verschillende DBMS
- **(nl) 1Keydata.com** (<http://www.1keydata.com/nl/sql/>): Deze SQL-tutorial wil beginners de bouwstenen van de databasetaal SQL aanleren

Begrippen: Database · ACID · CRUD · Null · Vreemde sleutel · Primaire sleutel · Kandidaatsleutel · Axioma's van Armstrong

Objecten: Tabel · Relatie · View · Transactie · Trigger · Index · Opgeslagen procedure · Cursor · Partitie

SQL: Select · Insert · Update · Merge · Delete · From · Join · Union · Create · Drop · Commit · Rollback · Truncate · Alter · Where · DDL · DML · DCL

Producten: Relationale databases

Overgenomen van "<http://nl.wikipedia.org/w/index.php?title=SQL&oldid=42824848>"

Categorieën: SQL | Relationale database

- Deze pagina is het laatst bewerkt op 24 dec 2014 om 20:45.
- De tekst is beschikbaar onder de licentie Creative Commons Naamsvermelding/Gelijk delen, er kunnen aanvullende voorwaarden van toepassing zijn. Zie de gebruiksvoorwaarden voor meer informatie.
Wikipedia® is een geregistreerd handelsmerk van de Wikimedia Foundation, Inc., een organisatie zonder winstoogmerk.