

Tabel (database)

Uit Wikipedia, de vrije encyclopedie

Een **tabel** is een entiteit binnen een relationele database. Een tabel heeft een naam en bestaat verder uit:

- 'Kolommen'. Deze vormen de definitie van de velden. Zo'n beschrijving luidt bijvoorbeeld:
 - Numeriek 10.2, alleen lezen, verplicht in te vullen veld
 - Alfanumeriek 20, lezen/schrijven, optioneel veld
- 'Rijen' of *records*. Deze bevatten de gegevens.
- **Sleutel** of *index*. Deze dienen om de goede rij snel op kunnen te zoeken. Vaak is er een primaire sleutel (primary key) en een aantal secundaire sleutels. De term zoeksleutel wordt ook vaak gebruikt.
- *Triggersoftware*. Deze wordt aangeroepen bij mutatie van een record. Dit in tegenstelling tot *remote procedures* die expliciet worden aangeroepen middels een remote procedure call (RPC) vanuit clientsoftware en dus geen enkele binding hebben met een tabel.

Een record bestaat weer uit velden. Elk veld komt overeen met een kolom die zijn metagegevens bevat. De gegevens in een tabel wordt vaak gepresenteerd als een matrix van velden waarvan de eerste regel de kolomnamen vormen.

In een goede database komt een gegeven maar 1 keer voor. Om gegevens te koppelen kunnen relaties worden gebruikt tussen de tabellen.

Inhoud

- 1 Voorbeeld
- 2 Relaties
- 3 Index (of sleutel)
- 4 Trigger
 - 4.1 Uitleg
 - 4.2 Voorbeelden

Voorbeeld

Hieronder een voorbeeld van de tabellen *Klanten* en *Facturen*. In de tabel *Klanten* is de kolom *KlantID* de primaire sleutel. In de tabel *Facturen* is *FactuurID* de primaire sleutel. Dit zijn gebruikelijke namen voor sleutels. De getallen zelf betekenen niets, ze moeten echter wel uniek binnen de tabel zijn.

Een andere zoeksleutel voor *Klanten* zou kunnen zijn de combinatie van kolommen *Naam* en *Achternaam*. Deze hoeft dan niet uniek te zijn. De primaire sleutel is altijd uniek.

Klanten:		
KlantID	Naam	Achternaam
1	David	Copperfield
2	Hans	Kazan

Facturen:		
FactuurID	KlantID	bedrag
1	2	10
2	1	20
3	2	20
4	2	30
5	1	40

De nummers in de kolom *Klanten.KlantID* komen overeen met die in *Facturen.KlantID*. Dit vormt een 1-op-n-relatie waarbij 1 klant n facturen kan hebben.

Relaties

De relationele database (RDB) heeft zijn naam te danken aan het feit dat een tabel relaties legt tussen velden: hetzij velden van één rij van één tabel, hetzij velden van verschillende tabellen, wanneer deze met elkaar verbonden zijn via een sleutelkolom. Binnen de tabel Facturen heeft het veld *KlantID* een *foreign key*. Dit omdat deze kolom in feite de sleutel is van een andere tabel.

Een tabel beschrijft in het algemeen meerdere n-op-m-relaties, omdat een kolom k1 meerdere keren dezelfde waarde kan bevatten, telkens horend bij een verschillende waarde van kolom k2, terwijl tezelfdertijd verschillende waarden van k1 met dezelfde waarde van k2 kunnen overeenkomen. Behoren kolommen k1 en k2 tot twee verschillende tabellen, dan heeft men dus een n-op-m-relatie tussen die twee tabellen. Meestal echter zal men in dat geval een extra tabel aanmaken die de koppeling implementeert. Zo'n tabel bevat dan twee foreign keys, één naar elk van de twee basistabellen. In die basistabellen komt elke waarde in kolom k1 resp. k2 slechts één keer voor, terwijl in de koppelabel de oorspronkelijke n-op-m-relatie terug te vinden is. Meer hierover in het artikel Datamodel.

Index (of sleutel)

Een index (of zoeksleutel) biedt de mogelijkheid om zeer snel de juiste rij in een tabel te vinden, ook als er miljoenen rijen zijn. De technieken die hiervoor gebruikt worden lijken op technieken die voor associatieve arrays gebruikt worden. Deze technieken zijn hashtables en binaire bomen. Als een rij aan de tabel toegevoegd of verwijderd wordt, of er worden velden gewijzigd, dan moeten de indices bijgewerkt worden. Omdat indices soms in een aparte file bijgehouden worden spreekt men ook wel over het bijwerken van de index file.

Het bovenstaande impliceert dat hoe meer indices er over een tabel heen liggen, hoe langer het wijzigen van een rij duurt. Het opzoeken gaat echter sneller. Het opzoeken van een rij via kolommen waarvoor geen index bestaat duurt enorm lang vergeleken bij geïndexeerd zoeken. De ontwerper van de database moet het maken van indices dus goed kiezen naar aanleiding van het te verwachten gebruik. Bij een tabel die niet vaak wijzigt zal men geneigd zijn meer indices aan te leggen.

Verwar een index of sleutel niet met het begrip "primaire sleutel" of "foreign key" zoals hierboven bij "relaties" besproken!

Trigger

Uitleg

Het is bij de meeste relationele databases mogelijk om triggers aan te brengen. Zo'n trigger zorgt ervoor dat op het moment dat er in een tabel een wijziging optreedt (record wijzigen, record verwijderen, record toevoegen) er een script (bijvoorbeeld SQL) of computerprogramma (bijvoorbeeld RPG) aangeroepen wordt dat controles uitvoert of meer wijzigingen in de database aanbrengt. Als zo'n controle faalt kan de wijziging niet uitgevoerd worden.

Voorbeelden

- Bij het *verwijderen* van een klant-record moeten ook de bijbehorende facturen verwijderd worden.
- Bij het *aanmaken* van een factuur-record wordt gecontroleerd of er wel een bijbehorende klant is. Zo nee, dan faalt het aanmaken van een factuur-record.
- Bij het *wijzigen* van een factuur-record wordt gecontroleerd of de klant-id niet verandert.
- Bij het aanmaken van een factuur wordt een teller in de klanten-tabel met één verhoogd.

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=Tabel_\(database\)&oldid=42824733](http://nl.wikipedia.org/w/index.php?title=Tabel_(database)&oldid=42824733)"

Categorieën: 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.

Select (SQL)

Uit Wikipedia, de vrije encyclopedie

Een **SELECT**-opdracht in SQL geeft een verzameling rijen of records terug uit één of meer tabellen van een database. Het resultaat van een **SELECT**-opdracht is dus een nieuwe (zij het virtuele, dat wil zeggen niet fysiek opgeslagen) tabel.

De opdracht wordt gebruikt om een query uit te voeren die nul of meer rijen ophaalt door data uit één of meer tabellen van een database te raadplegen en mogelijk te converteren of zelfs samen te vatten. In de meeste toepassingen is **SELECT** het meest gebruikte Data Manipulation Language-commando (DML) van SQL.

Bij **SELECT** kan je meestal de volgende sleutelwoorden gebruiken:

- **FROM** – het enige syntactisch verplichte sleutelwoord; om aan te geven welke tabel of tabellen ondervraagd wordt/worden.
- **WHERE** – gebruikt om op te geven welke rijen moeten worden opgehaald door middel van één of meerdere criteria.
- **GROUP BY** – gebruikt om rijen (met dezelfde waarde in de opgegeven expressie) samen te brengen in een kleinere groep rijen.
- **HAVING** – gebruikt samen met **GROUP BY** om te bepalen welke groepen opgehaald moeten worden.
- **ORDER BY** – gebruikt om de expressie(s) op te geven op basis waarvan de resultaatgegevens gesorteerd moeten worden.
- **INTO** – gebruikt om de rijen te kopiëren naar een nieuw aan te maken tabel of naar één of meerdere variabelen.

Inhoud

- 1 SELECT Syntax
- 2 Voorbeelden
- 3 Join
- 4 Zie ook

SELECT Syntax

```
SELECT column_name(s)  
FROM TABLE_NAME
```

en

```
SELECT * FROM TABLE_NAME
```

De asterisk (*) wordt gebruikt om snel alle kolommen te selecteren.^[1]

Voorbeelden

Voorbeeld 1:

Gegeven een tabel T zal de query `SELECT * FROM T` zal alle kolommen, rijen en velden ophalen uit de tabel.

Voorbeeld 2:

De query `SELECT K1 FROM T` zal alleen de kolom K1 ophalen en daarvan alle rijen tonen — in relationele algebra noemt men dit een *projectie*.

Voorbeeld 3:

De query `SELECT * FROM T WHERE K1 = '1'` zal alle kolommen en alle rijen ophalen waarvan de waarde van de kolom K1 gelijk is aan '1' — in relationele algebra heet dit een *selectie*.

Voorbeeld 4:

```
SELECT * INTO nieuwetabel FROM T
```

zal een nieuwe tabel "nieuwetabel" aanmaken en alle rijen van T daarin plaatsen.

Join

In een `SELECT`-opdracht kunnen ook twee tabellen samengevoegd worden. Dit gaat door middel van een `JOIN`. Er zijn twee soorten joins, een inner-join en een outer-join. Een outer-join kan op zijn beurt weer left, right of full zijn.

Voorbeeld:



```
SELECT * FROM tabel1 INNER JOIN tabel2 ON tabel1.veld = tabel2.veld
```

Zie ook

- SQL
- DELETE
- UPDATE
- INSERT
- COPY

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

Bronnen, noten en/of referenties

1. **W3 Schools** (http://www.w3schools.com/sql/sql_select.asp) (20-03-2011)

Overgenomen van "[http://nl.wikipedia.org/w/index.php?title=Select_\(SQL\)&oldid=42824944](http://nl.wikipedia.org/w/index.php?title>Select_(SQL)&oldid=42824944)"

Categorieën: SQL | Relationele database

-
- Deze pagina is het laatst bewerkt op 24 dec 2014 om 21:00.
 - 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.

Join (SQL)

Uit Wikipedia, de vrije encyclopedie

Een **JOIN**-clause is een onderdeel van een SQL-query, waardoor records van twee of meer tabellen uit een database gecombineerd kunnen worden.

Er zijn twee soorten joins in SQL volgens de ANSI-standaard, een inner-join en een outer-join. Een outer-join kan op zijn beurt weer left, right of full zijn.

Een left outer join doet een query op één tabel en zoekt dan bij elk resultaatrecord extra velden van de tweede tabel. De syntax is als volgt:

```
SELECT * FROM tabel1 LEFT OUTER JOIN tabel2 ON tabel1.veld = tabel2.veld
```

Een left outer join garandeert dus de aanwezigheid van alle rijen van de linker tabel (hier tabel1). Een inner join doet dat niet, omdat rijen van tabel1 waarvoor er in tabel2 geen enkele rij gevonden wordt waarvoor tabel1.veld = tabel2.veld niet getoond worden.

Dit kan aangevuld worden met de **WHERE**-clause en de andere clauses. In de **ON**-clause wordt gespecificeerd welke velden van de beide tabellen overeen moeten komen. Dit wordt gebruikt om de goede records van de tweede tabel bij de eerste te vinden. Indien geen record in tabel2 gevonden wordt is er toch een resultaatrecord, weliswaar met lege velden (NULL's) in het deel dat uit tabel2 afkomstig is. De sleutelwoordcombinatie **RIGHT OUTER JOIN** kan ook gebruikt worden. Verwissel in dat geval tabel1 en tabel2 in de uitleg hierboven. Een **FULL OUTER JOIN** ten slotte garandeert de aanwezigheid van alle rijen van zowel tabel1 als tabel2: het resultaat is dus de **UNION** van de **LEFT OUTER JOIN** en de **RIGHT OUTER JOIN**.

Een inner-join is in eerste instantie gelijk aan wat men in de verzamelingenleer en in de relationele algebra het cartesisch product noemt. Dit wil zeggen dat het resultaat alle combinaties van records van de eerste tabel met alle records van de tweede tabel bevat. In tweede instantie —na het **ON**-keyword— worden deze records gefilterd zodat enkel de rijen overblijven waarvan de velden, die in de **ON**-clause gespecificeerd zijn, overeenkomen. De syntax is als volgt:

```
SELECT * FROM tabel1 INNER JOIN tabel2 ON tabel1.veld = tabel2.veld
```

Het aantal resultaatrecords van een inner-join kan maximaal oplopen tot het aantal records in het cartesisch product. Records van tabel1 waarvoor geen overeenkomstige records in tabel2 worden gevonden, worden niet getoond. Daar hebben we de eerder vermelde **OUTER JOIN** voor. Anderzijds worden records van tabel1 waarvoor meerdere overeenkomstige records van tabel2 worden gevonden, ook even zoveel keer getoond.

Voorheen werd de inner-join anders geformuleerd, namelijk zonder de **ON**-clause; de join-informatie stond gewoon in de **WHERE**-clause. Toen was een vergissing snel gemaakt. Een voorbeeld van de oude notatie:

```
SELECT * FROM tabel1, tabel2 WHERE tabel1.veld = tabel2.veld
```

Deze notatie is nog steeds syntactisch correct, en logisch equivalent aan de inner-join.

Zie ook

- [SQL](#)
- [SELECT](#)

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

Bronnen, noten en/of referenties

Overgenomen van "[http://nl.wikipedia.org/w/index.php?title=Join_\(SQL\)&oldid=42824955](http://nl.wikipedia.org/w/index.php?title=Join_(SQL)&oldid=42824955)"

Categorieën: SQL | Relationele database

- Deze pagina is het laatst bewerkt op 24 dec 2014 om 21:02.
- De tekst is beschikbaar onder de licentie Creative Commons Naamsvermelding/Gelijk delen, er kunnen aanvullende voorwaarden van toepassing zijn. Zie de gebruiksovereenkomst voor meer informatie.

Wikipedia® is een geregistreerd handelsmerk van de Wikimedia Foundation, Inc., een organisatie zonder winstoogmerk.

Insert (SQL)

Uit Wikipedia, de vrije encyclopedie

Een **INSERT**-opdracht in SQL voegt één of meer rijen toe aan een tabel in een database.

Basisvorm

INSERT-opdrachten hebben de volgende vorm:

- **INSERT INTO** *tabel* (*kolom1*, [*kolom2*, ...]) **VALUES** (*waarde1*, [*waarde2*, ...])

Het aantal kolommen en waarden moet hetzelfde zijn. Als een kolom niet opgegeven wordt, dan wordt de standaardwaarde gebruikt voor die kolom. De waarden die bij het **INSERT**-commando opgegeven (of verondersteld) worden moeten aan alle constraints voldoen. Wanneer één of meer constraints worden geschonden, treedt een syntaxfout op en wordt de nieuwe rij niet toegevoegd.

Voorbeeld:

```
INSERT INTO telefoonboek (naam, nummer) VALUES ('Piet Janssens', '555-1212')
```

Als er waarden gegeven worden voor alle kolommen in de tabel, dan mag een kortere notatie gebruikt worden waarbij de volgorde van de kolommen in de tabel moet worden aangehouden:

- **INSERT INTO** *tabel* **VALUES** (*waarde1*, [*waarde2*, ...])

Voorbeeld (veronderstellende dat 'naam' en 'nummer' de enige kolommen zijn in de tabel 'telefoonboek'):

```
INSERT INTO telefoonboek VALUES ('Piet Janssens', '555-1212')
```

- **INSERT INTO** *tabel* **SELECT** ...

Deze constructie laat toe om velden te selecteren (via de **Select (SQL)**) en deze geselecteerde waarden als invoegwaarden te gebruiken in de **INSERT**-operatie. Op deze manier kan een aantal records/rijen van de ene tabel in een andere tabel worden gekopieerd. Voorbeeld (veronderstellende dat 'naam' en 'nummer' twee kolommen zijn in de tabel 'telefoonboek'):

```
INSERT INTO TabelTwee (nummer, naam) SELECT nummer, naam FROM telefoonboek
```

Zie ook

- SQL
- SELECT
- DELETE
- UPDATE

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=Insert_\(SQL\)&oldid=42824951](http://nl.wikipedia.org/w/index.php?title=Insert_(SQL)&oldid=42824951)"

Categorieën: SQL | Relationele database

-
- Deze pagina is het laatst bewerkt op 24 dec 2014 om 21:01.
 - 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.

Delete (SQL)

Uit Wikipedia, de vrije encyclopedie

Een **DELETE**-opdracht in SQL verwijdert één of meerdere rijen van een relationele database.

DELETE wordt gebruikt in de volgende vorm:

```
DELETE FROM tabelnaam WHERE [criteria]
```

Opdat de **DELETE**-opdracht zou lukken moet de gebruiker de nodige machtigingen hebben voor het wijzigen van data, en de verwijderde rij mag geen primaire sleutel zijn in een geldende "foreign key" constraint (tenzij die gedefinieerd is met een "SET NULL"- of "CASCADE"-conditie).

Voorbeelden

Verwijder alle rijen uit tabel *T*. Na afloop is tabel "*T*" dus leeg.

```
DELETE FROM T
```

Verwijder alleen die rijen uit tabel *T* waarvoor de waarde van *C2* in de rij exact gelijk is aan "a".

```
DELETE FROM T WHERE C2 = 'a'
```

Verwijder rijen uit tabel "*T1*" indien de waarde van *C2* (in tabel *T1*) bestaat in het veld *C3* tabel *T2*

```
DELETE FROM T1 WHERE C2 IN (SELECT C3 FROM T2)
```

Zie ook

- SQL
- SELECT
- UPDATE
- INSERT
- TRUNCATE

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=Delete_\(SQL\)&oldid=42824791](http://nl.wikipedia.org/w/index.php?title=Delete_(SQL)&oldid=42824791)"

Categorieën: SQL | Relationele database

- Deze pagina is het laatst bewerkt op 24 dec 2014 om 20:36.
 - 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.

Update (SQL)

Uit Wikipedia, de vrije encyclopedie

Een **UPDATE**-opdracht in SQL wijzigt gegevens in één of meerdere records van een database.

UPDATE wordt gebruikt in de volgende vorm:

```
UPDATE tabelnaam SET kolomnaam1 = waarde1 [, kolomnaam2 = waarde2 ...] [WHERE criteria]
```

Opdat de UPDATE-opdracht zou lukken moet de gebruiker de nodige machtigingen hebben voor het wijzigen van data, de nieuwe waarde mag geen conflict geven met de geldende constraints (zoals primaire sleutels, unieke indexen en constraints).

Voorbeelden

Zet de waarde van kolom $C1$ in tabel T op 1, indien de waarde van $C2$ in de rij "a" is.

```
UPDATE T SET C1 = 1 WHERE C2 = 'a'
```

Verhoog de waarde van de kolom $C1$ met 1 indien $C2$ "a" is.

```
UPDATE T SET C1 = C1 + 1 WHERE C2 = 'a'
```

Verhoog de waarde van de kolom $C1$ (in tabel $T1$) met 2 indien de waarde van $C2$ (in tabel $T1$) bestaat in het veld $C3$ (in tabel $T2$)

```
UPDATE T1 SET C1 = C1 + 2 WHERE C2 IN (SELECT C3 FROM T2)
```

Complexe voorbeeld

Dit voorbeeld wijzigt meerdere kolommen in tabel $T1$ door deze te scheiden met een komma (,) indien de waarde van $C0$ (in tabel $T1$) bestaat in het veld $C1$ in de tabel $T2$:

- $C1$ wordt de waarde van de drie kolommen bij elkaar opgeteld;
- $C2$ wordt verhoogd met 1;
- $C3$ wordt vermenigvuldigd met 2.

```
UPDATE T1 SET C1 = C1 + C2 + C3, C2 = C2 + 1, C3 = C3 * 2 WHERE C0 IN (SELECT C1 FROM T2)
```

Zie ook

- SQL
- SELECT
- INSERT
- DELETE

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=Update_\(SQL\)&oldid=42824810](http://nl.wikipedia.org/w/index.php?title=Update_(SQL)&oldid=42824810)"

Categorie: SQL

-
- Deze pagina is het laatst bewerkt op 24 dec 2014 om 20:40.
 - 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-injectie

Uit Wikipedia, de vrije encyclopedie

De term **SQL-injectie** (Engels: *SQL injection*) wordt gebruikt voor een type kwetsbaarheid van computerapplicaties, meestal webapplicaties. Applicaties die informatie in een database opslaan maken vaak gebruik van SQL om met de database te communiceren. SQL-injectie kan gebeuren als invoer van gebruikers op onvoldoende gecontroleerde wijze wordt verwerkt in een SQL-statement. Om de precieze werking van SQL-injectie te begrijpen is het belangrijk om te weten hoe SQL werkt.

Inhoud

- 1 Rol van de apostrof in SQL
- 2 SQL-injectie
- 3 Preventie
 - 3.1 Afwijzen van verkeerde invoer
 - 3.2 Backslash
 - 3.3 Statement
 - 3.4 Databasepermissies
- 4 Externe links

Rol van de apostrof in SQL

In SQL heeft de apostrof een belangrijke functie, namelijk het afbakenen van niet-numerieke gegevens. Om bijvoorbeeld alle personen met de naam "Jansen" te selecteren uit een tabel wordt het volgende statement gebruikt

```
SELECT * FROM persoon WHERE achternaam = 'Jansen'
```

In een applicatie waar gezocht kan worden naar personen, zal de gebruiker in het zoekveld uitsluitend "Jansen" invullen. In de applicatie wordt op basis van deze invoer bovenstaande code naar de database gestuurd.

Interessant wordt het als de gebruiker een apostrof in het zoekveld invult, bijvoorbeeld "'t Hart". In een correct statement moet dan namelijk de apostrof worden verdubbeld, of voorzien van een backslash ()).

```
SELECT * FROM persoon WHERE achternaam = '''t Hart'  
SELECT * FROM persoon WHERE achternaam = '\\'t Hart'
```

Als dat niet gebeurt, dan levert het een incorrect SQL-statement op, en volgt een foutmelding van de database. Maar het betekent ook dat de applicatie niet beschermd is tegen SQL-injectie.

SQL-injectie

SQL-injectie bestaat er uit dat een gebruiker in het invoerveld tekens invoert die er voor zorgen dat een ongewenste SQL-query wordt uitgevoerd. Daarbij wordt vaak gebruikgemaakt van de apostrof. Dit kan alleen als bij het genereren van de SQL-code op basis van gebruikersinvoer de apostrof niet goed wordt afgevangen.

De gebruiker typt bijvoorbeeld "Jansen' OR 'a' = 'a" in het zoekveld. Het resulterende statement is dan

```
SELECT * FROM persoon WHERE achternaam = 'Jansen' OR 'a' = 'a'
```

Omdat "'a' = 'a'" altijd waar is, voldoet nu elk record aan de gestelde voorwaarde.

Met bovenstaand voorbeeld kan de hacker extra informatie ophalen uit de database. Dezelfde methode levert ook de mogelijkheden om nieuwe informatie aan de database toe te voegen, bestaande informatie aan te passen en informatie te verwijderen. Daarvoor is informatie nodig over de structuur (namen van tabellen en kolommen) van de database. De naamgeving van tabellen en kolommen is meestal logisch om werken met de database voor een reguliere gebruiker eenvoudig te houden en daardoor voorspelbaar. Daarnaast kan de hacker ook diverse zaken uitproberen om bijvoorbeeld een gebruikersaccount met beheerders-rechten aan te maken. Lukt dit, dan kan de hacker de totale controle over de computer overnemen, met alle gevolgen van dien.

Preventie

Door middel van het geven van de minimaal noodzakelijke rechten van de gebruiker en de SQL-server kan het ongewenst aanpassen van gegevens en het uitvoeren van ongewenste commando's op het systeem moeilijker gemaakt worden.

Afwijzen van verkeerde invoer

Men kan alleen bepaalde tekens en strings toestaan in de invoer en alles wat een betekenis heeft in een SQL-commando afwijzen (bijvoorbeeld de tekens en strings: insert drop '-- ; enzovoorts, maar ook char zodat niet via een omweg een ' in de invoer kan voorkomen). Zuiver numerieke gebruikersnamen en wachtwoorden moeten afgewezen worden, omdat die automatisch omgezet kunnen worden in strings met ongewenste betekenis.

Backslash

De injectie met SQL-code kan eenvoudig tegengegaan worden door het juist verwerken van informatie die door een gebruiker wordt aangeleverd. In de programmeertaal PHP kan dat bijvoorbeeld via `mysql_real_escape_string()` (http://www.php.net/mysql_real_escape_string). Deze functie vangt (my)SQL-specifieke karakters af door er een backslash (\) voor te plaatsen. Hierdoor weet het systeem dat enkel het letterteken bedoeld wordt, en niet meer de scheidende functie van het afbakenen van gegevens. Een stukje voorbeeld-programmeertaal in PHP kan er als volgt uitzien:

```
<?php  
$result = mysql_query("SELECT * FROM persoon WHERE achternaam = '" . mysql_real_escape_string($_POST['achternaam']) . "'");  
?>
```

Statement

Een andere methode om injectie tegen te gaan is door middel van een voorgedefinieerd statement. Hierbij wordt in het aanroepende programma het statement opgebouwd met een variabele. De inhoud van de variabele wordt dan gekoppeld aan de gebruikersinvoer.

Bijvoorbeeld (in de taal Java):

In plaats van

1. Connection con = (maak verbinding met de database)
2. Statement stmt = con.createStatement();
3. ResultSet rset = stmt.executeQuery("SELECT * FROM persoon WHERE achternaam = " + invoer + "");;

is het beter om het volgende te gebruiken

1. Connection con = (maak verbinding met de database)
2. PreparedStatement pstmt = con.prepareStatement("SELECT * FROM persoon WHERE achternaam = ?");
3. pstmt.setString(1, invoer);
4. ResultSet rset = pstmt.executeQuery();

Databasepermissies

Schade kan beperkt worden door alleen de strikt nodige permissies te verlenen. Bijvoorbeeld kan men het op SQL-server onmogelijk maken bepaalde tabellen te lezen:

```
deny SELECT ON sys.sysobjects TO webdatabaseLogon;
deny SELECT ON sys.objects TO webdatabaseLogon;
deny SELECT ON sys.tables TO webdatabaseLogon;
deny SELECT ON sys.views TO webdatabaseLogon;
```

Externe links

- (en) Advanced SQL injection
(http://web.archive.org/web/20070928163708/http://www.ngssoftware.com/papers/advanced_sql_injection.pdf)

Overgenomen van "<http://nl.wikipedia.org/w/index.php?title=SQL-injectie&oldid=42005084>"

Categorie: SQL

-
- Deze pagina is het laatst bewerkt op 2 sep 2014 om 00:11.
 - 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.