Verrassend bescheiden SQL-implementaties zijn mogelijk

Databases zonder poeha

Rick van Rein

We denken bij SQL-databases vaak aan indrukwekkende installaties op zware machines. Maar voor kleinere toepassingen, van boodschappenlijstjes tot MP3-databases, is dat overdreven. We bespreken hier een paar verrassend bescheiden implementaties van SQL die voor dergelijke toepassingen heel geschikt zijn.

De kleinste database die we ons kunnen voorstellen is voor de meesten MySQL. Eenvoudig te installeren, niet moeilijk te beheren en vrij krachtig als server. Maar MySQL vereist wel de rechten van een systeembeheerder, iets wat niet elke gebruiker overal heeft. En MySQL is toch wel echt een database, in de zin dat de backups afzonderlijk geregeld moeten worden en in de zin dat de data niet via alledaagse tools beheersbaar zijn.

> Voor scripting is het altijd handig om dynamische data te kunnen aanspreken

In mijn Unix-accounts op Universiteit Twente had ik altijd een heel stelsel van key/value opzoekscripts, waarmee ik allerlei zaken flexibel kon instellen. Voor een doorgewinterde shell-gebruiker zijn zulke zaken geweldig handig. Hoewel de kracht van SQL ontbrak, was het wel een database die ik zelf kon beheren vanuit mijn account, waarin ik kon wijzigen met een gewone editor en waarvan de data ook tegelijk met de rest van mijn home directory onder het backup-regime van de faculteit vielen.

Lichtgewicht

Voor scripting, zowel voor direct shell-gebruik als voor web scripts, is het altijd handig om dynamische data te kunnen aanspreken, maar omdat scripts vrijwel zonder state-informatie worden gedraaid is het vooral nuttig wanneer er niet te veel opstart-overhead is. Het opbouwen van megabytes aan data over caching en concurrency control is niet bepaald ideaal voor het snel afhandelen van een scriptje. En de mogelijkheid van een pool van open verbindingen brengt security-problemen met zich mee. Veel handiger is het dan om vanuit de scripts rechtstreeks bij de data te kunnen, zonder al te veel zorgen over de zwaarwichtige zaken die een door gebruikers gedeeld RDBMS met zich mee brengen.

SHQL, SHSQL en SQLite zijn SQL-implementaties die rechtstreeks vanuit scripts te gebruiken zijn. En dan bedoelen we niet alleen zoals in PHP, waar men programmeert op een API die ook voor C-programmeurs ter beschikking staat, maar we bedoelen ook het rechtstreeks intoetsen van SQL in een eenvoudig shell-script.

SHQL is zelfs een systeem dat geheel in shell-taal is geschreven. Er is een uitvoering voor ksh en de (tegenwoordig populairdere) bash shell-scripting taal. Behalve dat dit toont hoe krachtig deze commandline-talen voor Linux/Unix zijn, maakt het ook een hoop extra functionaliteit beschikbaar voor de gebruiker die geregeld gebruik maakt van de kracht van die shell.

SHQL in vijf minuten

Het downloaden van SHQL is snel gebeurd – de tarball is 26 KB groot – en ook de ingebruikname is triviaal. Na uitpakken van de tarball vinden we een README bestand van 800 woorden, en ook het commando shql geeft nette feedback.

We moeten een directory aanmaken voor de database, wat eenvoudig genoeg is met:

```
mkdir -p ~/.shql
```

Vervolgens is het eenvoudig om een database aan te maken (met het shql-commando in het zoekpad van de shell):

```
shql -c testdb
```

Vervolgens zijn er commando's (getoond na shql -h) voor het dumpen van de database als SQL-statements, van een tabel in CSV-formaat, het tonen van alle databases en het tonen van alle tabellen of views binnen een database. Maar het meest interessante is natuurlijk het opstarten van de SQL-prompt:

```
shql testdb
```

Hierna kan men SQL tikken.

```
testdb> create table person (id integere, naam
                                        varchar(255));
OK
testdb> insert into person values
(1,'Rick',2,'Hans');
( 2 rows )
testdb> select naam,id from person;
|-naam-   |-id-|
|Rick    |  1|
|Hans    |  2|
testdb> quit;
Goodbye
```

De database reageert razendsnel op al deze verzoeken. Ook worden bij syntaxfouten vrij uitgebreide commentaren gegeven, zodat er verbeterd kan worden. Wie overigens niet op een prompt wil werken kan de gewone shell-trucs toepassen. Men kan met een pipe commando's ingeven:

```
echo "select * from person;" | shql testdb
```

Let daarbij wel op de puntkomma die in SQL noodzakelijk is. Om verwarring te voorkomen met de shell (die * en ; zou interpreteren) staan er quotes om het statement.
Het is ook mogelijk om hierbij shell-variabelen te gebruiken, zoals in:

```
NAAM=Rick
echo "select * from person where naam='$NAAM';" |
                                        shql testdb
```

Een andere typische werkwijze voor de shell is het gebruik van de regels na een commando tot aan een eindmarker, zoals EOF in dit voorbeeld:

```
shql testdb << EOF
select *
from person
where naam="$NAAM";
EOF
```

De tabeluitvoer werkt met vaste kolombreedtes en is daardoor zeer geschikt voor verwerking met normale shell-commando's zoals cut. Zo kan de uitvoer geheel naar wens worden gebruikt in de lussen en dergelijke die een shell biedt. De kracht van dit alles is weergaloos doordat alles naadloos op elkaar aansluit.
Let wel: al deze commando's worden door shell-scripts geïnterpreteerd. Er is geen enkel specifiek C-programma nodig. Een indrukwekkende prestatie, zeker in broncode van minder dan 2000 programmaregels. Dat relativeert de complexiteit van SQL nogal!

Platte tekst

Het meest onderschatte formaat in computerland is vermoedelijk platte tekst. In een door Windows gedomineerde wereld zijn veel gebruikers gewend geraakt aan de afhankelijkheid van een applicatie om bij hun binaire bestanden te komen. Maar applicaties blijven niet eeuwig bruikbaar, omdat ze op een bepaalde OS-versie draaien, en dat weer op bepaalde hardware. De vooruitgang wil daardoor nog weleens oude data onbereikbaar maken. Overheden beginnen dat in te zien en sturen daarom aan op open source en open standaarden.

Het meest open formaat is uiteraard platte tekst, ook wel plain text of ASCII genoemd. Dit formaat is niet aan veranderingen onderhevig (behalve misschien de overgang op UTF-8 om een uitgebreidere karakterset aan te kunnen met bijvoorbeeld tildes en Japanse karakters). Het is dan ook prima leesbaar met elke editor, zonder enig probleem. Bovendien is er een scala aan command line tools die er van alles mee kunnen uithalen. Doorgewinterde shell-gebruikers kunnen toveren met tools als grep, cut, uniq en sed.
De informatie van shql is volgens deze doorzichtige formaten opgemaakt. Elke database is een subdirectory in het database-zoekpad. Bij het aanmaken van de testdb onder shql is gewoon een directory ~/.shql/testdb aangemaakt. Daaronder vinden we per tabel twee bestanden. In dit geval person@ met de type-informatie;

```
id    int4
naam  char255
```

en person~ met de tabelinhoud;

Samenvatting

In databaseland is MySQL bij lange na niet het kleinste dat mogelijk is. Bijvoorbeeld voor embedded systemen is MySQL veel te groot. SHQL is daar veel meer op zijn plaats. Deze database is verrassend krachtig, en kan net de missende link zijn bij het integreren van SQL in een bescheiden omgeving zoals een embedded systeem of een eenvoudige shell account.
Voor toepassing binnen een shell account (zonder systeembeheerdersrechten) is SQLite vermoedelijk de handigste keuze, tenminste als men bereid is te programmeren. Deze database is klein, snel en betrouwbaar. Hij is bovendien wijd verbreid. Het belangrijkste verschil met MySQL is dat er geen server daemon wordt gedraaid, maar dat een library met de DBMS-code wordt ingeladen in software.
In de vergelijking tussen SHQL en SHSQL wint de eerste vanwege de ultieme eenvoud van het systeem. In de vergelijking tussen SHSQL en SQLite wint SQLite het, omdat deze taal uitgebreider is. Tussen SHQL en SQLite willen we geen vergelijking maken, omdat die elk hun eigen bestaansreden lijken te hebben.

```
1     Rick
2     Hans
```

en tenslotte is er voor views nog een file met een naam als person@ met daarin de select statement die de inhoud van de view bepaalt.
De kolomscheiders in de person~ en person@ files zijn tabs; mocht een tab in de data voorkomen, dan wordt die met een escape ook juist afgehandeld.

SHSQL, werkend in een kwartier

SHSQL komt als onderdeel van een pakket Quisp, een taal die bedoeld is voor webscripts. Op zich is SHSQL afzonderlijk te gebruiken, maar men moet dus wel een wat groter pakket downloaden. Het gaat hier om een pakket van 725 KB, dat met een simpele 'make' te bouwen is op een Linux-systeem met de normale bouwomgeving aan boord. Daarna wordt in bin/ een vijftal commando's geïnstalleerd, die elk zo'n 116 KB groot zijn; samen 576 KB.
Voor SHSQL moet wat meer handwerk worden verricht om van start te gaan; we moeten de bin/ directory aan het zoekpad toevoegen, en het script newproject.sh draaien; dat script vraagt om een directory waarin de administratie van de database kan worden geplaatst. Daarna stellen we de environment variabele SHSQL_DB op die directory in, bijvoorbeeld onder de Bash met:

```
export SHSQL_DB=/path/to/testdb
```

Informatie op internet

- SHQL is een SQL-interpreter die geschreven is voor de Bash commandline, die op Linux standaard gebruikt wordt. (http://lorance.freeshell.org/shql)
- SHSQL is een SQL-interpreter in een binair commando, maar bedoeld om vanaf de commandline (dus vanuit een shell) aan te roepen. (http://shsql.sourceforge.net)
- SQLite is een SQL-interpreter die werkt als een library die aan elk programma kan worden gelinkt voor ingebedde afhandeling van SQL-query's. De website bevat uitvoerige documentatie. (www.sqlite.org)
- BusyBox is een basissysteem voor embedded systemen. Het is een executable die een enorme hoeveelheid commando's implementeert, waaronder ash. Optioneel kan er een simpel webservertje in worden opgenomen, dat volgens dezelfde visie als thttpd kiest voor dynamische delen in cgi-bin executables – zoals scripts. (www.busybox.net)
- THTTPD is een kleine, maar veilige en superefficiënte webserver waarachter een sterk ontwerp schuilgaat. Hij presteert beter onder druk dan Apache. (www.acme.com/software/thttpd)

Deze directory bevat al veel meer structuur dan de charmante, lege directory van SHQL, maar alles wat hierin wordt opgeslagen is nog altijd plaintext. Echter, de engine van SHSQL is geen script maar een binaire executable. Er zijn al wat testtabellen aangemaakt, die een record opslaan als een regel, met spaties als veldscheiding. De eerste dataregel vermeldt de naam van de kolommen, maar een type is nergens te vinden. SHSQL kan alleen met strings werken.

Een opvraag van informatie kan worden gedaan met aanroepen als

```
shsql "select * from examp_art"
```

of wanneer men HTML-geformatteerde tabelregels wenst met

```
shsql -html "select * from examp_art"
```

of op soortgelijke wijze met -xml voor een XML-formattering. Om van dat laatste een voorbeeld te geven:

```
<database_retrieval>
<row>
<id>206</id>
<artist>Persolglia</artist>
<title>Musical Romance</title>
<format>oil</format>
<size>31~ x 21.5~</size>
<frame>museum mounted, custom gold lear
                                        frame</frame>
...
```

De uitvoerformaten van SHSQL zijn een stuk moderner dan die van SHQL. Dat wreekt zich helaas wel in veel grotere executables. Maar zoals gezegd wordt SHSQL gebundeld met Quisp. Hoewel de DBMS afzonderlijk te gebruiken is, komt het hiermee in combinatie natuurlijk het best uit de verf. Quisp is een taaltje dat door een HTML-document heen geweven kan worden, een beetje zoals dat ook gebruikelijk is met server-side includes. Regels die met # beginnen worden beschouwd als een Quisp-directive. Onder deze directives zitten commando's om SQL te draaien, maar bijvoorbeeld ook om lussen en conditionele statements uit te voeren, of om I/O met files te doen.
Het is mogelijk om de uitvoer van een SQL-query te gebruiken in een Quisp-script, wat soms handiger is dan het gebruik van een directe dump van alle data, hoewel dat ook mogelijk is en soms zeer handig.
In vergelijking met PHP is Quisp kleiner, maar het is veel minder krachtig dan de algemene mogelijkheden van een script-taal, waarin ook vrij gemakkelijk tekstfragmenten (zoals panklare HTML) kunnen worden opgenomen. De vraag is een beetje of Quisp het leven echt veel beter maakt, met name omdat er een nieuwe scripttaal voor moet worden geleerd, in tegenstelling tot

PHP voor grote toepassingen en shell+SHQL voor kleine. Blijft dus SHSQL als afzonderlijke database over. Ook hier geldt een dergelijke laken- en servet-redenatie: het is te groot voor embedded toepassingen, en voor allerlei toepassingen vermoedelijk te klein om tegen MySQL en het nog te bespreken SQLite op te boksen. Het mist eigenlijk de charme van eenvoud die SHQL wel heeft.

SQLite in nul minuten

Als derde en laatste lichtgewicht database kijken we naar SQLite, die nog weer een stapje krachtiger is dan SHSQL, maar gek genoeg wel kleiner. SQLite is een library die aan C-programma's gelinkt kan worden, of via 'stubs' vanuit scripttalen zoals Python, PHP, Ruby, Perl en zelfs Tcl aanspreekbaar is. Deze library is 250 KB groot en vergt totaal geen configuratie. Het ontwerpdoel achter SQLite is eenvoud. Dat levert de nodige voordelen, zoals kleine afmetingen, betrouwbaarheid en snelheid. De belangrijkste keuze die van SQLite een eenvoudige SQL-implementatie maakt is dat het geen client/server model volgt. Er kunnen wel meerdere gebruikers zijn, dus concurrency control is wel nodig, al kan ook dat natuurlijk eenvoudig worden gehouden.

De SQL-variant die SQLite ondersteunt is vrij compleet, tot ACID-transacties aan toe. In 1 minuut is de voorgecompileerde library voor i386 te downloaden, unzippen en installeren. Daarna moet er natuurlijk nog wel omheen worden geprogrammeerd. Overigens is de library vrij gewild en is de kans dat hij in een distributie zit nog groter dan de kans dat hij al geïnstalleerd staat op een gemiddeld Linux-systeem. Net als de andere twee databases in dit artikel is ook SQLite een database die binnen iemands account draait.
De gegevens worden alleen niet in platte tekst opgeslagen, en ook de aanspraak gebeurt niet vanuit shell-scripts. SQLite is daarmee een tool voor programmeurs, maar omdat ook scriptprogrammeurs daaronder vallen, wordt het toch een behoorlijk toegankelijke oplossing. Voor veel toepassingen waarvoor nu gebruik wordt gemaakt van MySQL, kan SQLite daarom een interessant alternatief vormen.

Beperkingen

De beperkingen van SHQL en SHSQL vallen nogal mee; weliswaar is het een wat beperkte versie van SQL, maar ze zijn toch behoorlijk bruikbaar. SQLite is indrukwekkend, want het is vrijwel compleet.
Wie verwacht dat SHQL te simpel zou zijn voor geneste query's, ALTER TABLE of zelfs maar UPDATE, die vergist zich! Op sommige andere punten stelt SHQL wel teleur; zo kan dezelfde tabel niet meermalen worden gebruikt en zijn er niet veel functies op de data uit te voeren in SHQL. De documentatie is ook niet bijster helder over wat wel en niet ondersteund wordt. Een beetje onvoorspelbaar dus, maar met enig proberen is snel te zien of het een bepaalde toepassing aankan.

SHSQL ondersteunt geen rekenkundige functies, alleen wel COUNT, SUM, AVG, MIN en MAX. In de WHERE-clausule worden haakjes niet ondersteund om de verwerkingsvolgorde van AND en OR aan te passen, een rare misser die weliswaar is op te vangen maar die veel herschrijvingen van de booleaanse logica van query's nodig maakt. In tegenstelling tot SHQL kent SHSQL geen andere datatypes dan strings tot een maximum lengte van 255 karakters.

SQLite is vrijwel compleet. Geavanceerde faciliteiten zoals triggers en ALTER TABLE worden alleen niet volledig ondersteund. Van de 'alledaagse' functionaliteit van een database ontbreken met name RIGHT OUTER JOIN en FULL OUTER JOIN.

Miniscule webservers

Een mooie toepassing van deze shell-gebaseerde databases ligt voor de hand: een webservertje dat shell-scripts met SHQL of SHSQL gebruikt voor dynamische elementen. Zo'n webserver is nuttig in allerhande apparaatjes. Een opzet in deze richting is mogelijk met de kleine shell ash die veel commando's ingebouwd heeft, een webserver zoals thttpd en eenvoudige scripts als cgi-bin delen. Met AJAX-technieken is dit zelfs zeer interactief te maken. Een vlugge test deed vermoeden dat ash en shql prima samengaan.

> Het meest onderschatte formaat in computerland is vermoedelijk platte tekst

De thttpd webserver is erg eenvoudig, erg efficiënt en erg veilig, dus het past perfect in dit type toepassingen. Het werkingsprincipe is klein en slim; door te wachten op het eerstvolgende verzoek op één van de aangesloten webclients en dergelijke verzoeken stuk voor stuk af te handelen, voorkomt het de complexiteit van normale, multi-tasking webservers. Alleen voor cgi-bin scripts start de webserver een eigen proces op. In die processen draait een executable.
Een belangrijk ontwerpdoel van embedded systemen is het klein houden van de code; open source toolkits zijn succesvol door daarvoor gebruik te maken van scripts ter vervanging van anders veel complexere binaire programma's. De shell-tools zijn krachtig genoeg om dit mogelijk te maken. SHQL is hiervan een prima voorbeeld. SHSQL en SQLite zijn, omdat het afzonderlijke binaire executables zijn, wat minder geschikt voor deze insteek.

Rick van Rein
Dr. ir. H. van Rein (rick@openfortress.nl) is ontwikkelaar en beheerder bij OpenFortress Digital signatures.