Structured Card Query Language 'aan de tand gevoeld'

Database in je portemonnee

Rick van Rein

In een smart card kun je in principe allerhande diensten kwijt. Tot onze grote verrassing kwamen we zelfs een standaard tegen voor een binaire variant op SQL, speciaal bedoeld voor smart cards.

Smart cards zijn de plastic kaartjes die we allemaal krijgen van onze favoriete winkelketen, benzinepomp en telefoonboer. De vorm en maten daarvan zijn gestandaardiseerd door de ISO, evenals de mogelijke chips die er in zitten. De desbetreffende standaardenreeks heet ISO-7816, en in deel 7 van die standaard is een mogelijkheid gestandaardiseerd om een variant op SQL in een smart card te ondersteunen.
In deel 1 van ISO-7816 worden fysieke zaken vastgelegd, zoals waar de grens tussen buigen en barsten ligt. In deel 2 spreekt men over de plek waar de contacten van een eventuele smart card te vinden zijn. In deel 3 worden de elektrische laag en een paar protocollen op het laagste niveau uit de doeken gedaan.
Bij deel 4 wordt het interessant. Dat deel legt een soort file-systeem vast dat in een smart card kan worden opgenomen. Denk daarbij aan files met daarin records die op nummer geselecteerd kunnen worden, denk aan voor- en achteruitspoelen in zo'n file, denk aan inloggen en dergelijke. Voor dat deel is een reeks commando's vastgelegd die daarvoor gebruikt kunnen worden. Die commando's hebben een bepaald binair formaat, en de standaard geeft aan hoe die door een conformerende kaart dienen te worden geïnterpreteerd. De standaard leeft heel erg op het niveau dat elke bit telt, en er zijn dan ook vrij veel tabellen waarin de hoogste twee bits aangeven welke tabel de interpretatie van de resterende bits vastlegt. Allemaal erg vervelend om mee te werken, maar embedded systemen zijn dan ook specialistische omgevingen.
In deel 7 over de Structured Card Query Language (SCQL) is men wat meer ontspannen. Weliswaar stuurt men niet 'SELECT NAME,SALARY FROM PERSONS' naar een database op een kaart die aan dat deel voldoet, maar de vertaling in een binair formaat is opvallend *rechttoe-rechtaan*. Om een SQL-parser onnodig te maken in de smart card is een eenvoudiger te parsen formaat gekozen, maar de informatie die in de query naar zo'n kaart gezonden wordt, is qua inhoud wel degelijk vergelijkbaar met SQL. Dat maakt het mogelijk deze standaard eens als SQL-variant aan de tand te voelen.

Architectuur

Het idee van een database op een smart card is vooral het meeneembaar maken van gevoelige data. Het kan best zijn dat koppeling nodig is met een grotere database, die de smart card slechts als plugin beschouwt. Of wat meestal handiger werkt, is het aanspreken van de smart card als zelfstandige DBMS via een driver die een SQL-achtige syntax omzet in de codes die de smart card verwacht.

Dankzij de standaard aansluitingen van een smart card kan deze in elke standaard interface device (of smart card reader) worden gestoken waarvoor de computer drivers heeft. Op alle operating systemen is dan een API te vinden op het door Window gedomineerde PC/SC niveau, waar commando-codes naar een smart card kunnen worden gestuurd, zoals bijvoorbeeld de binaire codes uit het kader. Een driver kan een ODBC-service aanbieden door alle aangeboden query's te vertalen naar zulke commando-codes.
Op die manier kan elke toepassing die bij de smart card wil, gebruikmaken van gewoon SQL of een deel daarvan. Zodra de smart card verwijderd is uit de reader, zijn de data niet langer beschikbaar.

Toegangscontrole

Het idee van een smart card is dat van een rekeneenheid die aan iemand mee te geven is zonder dat diegene er volledige controle over kan krijgen. Een smart card van goede kwaliteit is uitgerust met een fysiek beveiligde chip, waarin gevoelige informatie kan worden opgeslagen, zonder dat die er zomaar uit te onttrekken is. Het is op het eerste gezicht dus niet zo'n raar idee om bedrijfskritieke gegevens op te slaan op een smart card.
De toegang tot een smart card wordt meestal geregeld met een PIN. Zie bankpas. De redenatie achter deze betrekkelijk korte codes is dat ze alleen over lokale draadjes reizen en dus niet aftapbaar zijn. Bovendien zal de chip blokkeren als te vaak een foute PIN is geprobeerd – zodat de kans op raden bij vier cijfers maar 1:333 is, of beter bij ingewikkelder PIN-codes.

Gegeven de PIN is, binnen dezelfde sessie, toegang mogelijk tot afgeschermde functionaliteit – bijvoorbeeld het aanspreken van de database die op de kaart staat. Dat kan, afhankelijk van de PIN waarmee ingelogd is, als er diverse gebruikers zijn. In SCQL wordt onderscheid gemaakt tussen de eigenaar van de hele

database, de eigenaren van bepaalde objecten, en alledaagse gebruikers. Met een binaire variant op GRANT en REVOKE kunnen aan zulke gebruikers diverse rechten worden toegekend of onttrokken, zoals in een normale database. Doordat ook het VIEW-mechanisme bestaat, is het mogelijk om zelfs delen van tabellen (bepaalde kolommen en/of bepaalde rijen) af te schermen van bepaalde gebruikers.

Dit geheel timmert het gebruik van de database dicht op de gebruikelijke manier, en met de hardwarebescherming van de smart card. Het genoemde PIN-mechanisme is overigens slechts een voorbeeld, want de precieze manier van inloggen wordt niet in de SCQL-standaard vastgelegd. Maar er wordt wel verwezen naar de toegangsmechanismen uit ISO-7816-4, en die worden doorgaans uitgewerkt in de vorm van een PIN. Het linken aan die standaard is ook erg handig; er zijn bijvoorbeeld kaartlezers met toetsenbordje die daar gebruik van maken om in te loggen zonder tussenkomst van een workstation. Waardoor de PIN niet te stelen is met spyware.

Datatypes

Een grote beperking van SCQL is dat het alleen VARCHAR-data kan opslaan, desnoods met de randvoorwaarde dat de waarde uniek is binnen de kolom van een tabel. De maximumlengte wordt in een byte opgeslagen en kan dus hooguit 255 zijn.
Voor een kleine omgeving zijn grotere databrokken wellicht ook niet zo'n goed idee. In het hoogst nodige geval is in 255 ook een (hele lange) codeersleutel op te slaan waarmee gecodeerde informatie ontsleuteld kan worden, met dezelfde veiligheids-implicaties als wanneer de gegevens rechtstreeks vanaf de kaart te 'gedownload' zouden worden.
Opvallend is vooral het ontbreken van de waarde NULL. Dit scheelt veel complexiteit en uitzonderingen in de SCQL-code, maar het betekent dat aan uitzonderingen expliciet een waarde moet worden gegeven, en dat is dus extra werk voor de programmeur. Dat hoort een beetje bij een embedded systeem, maar het is voor een database-ontwerper wel even slikken.

Stukje bij beetje

In deel 4 van de smart card standaard wordt uitgelegd hoe een file in kleine records tegelijk te downloaden is. Dit scheelt flink in de communicatietijd, omdat die gebeurt met snelheden die vergelijkbaar zijn met een modem – 9600 baud en 115200 baud zijn bijvoorbeeld gangbare snelheden, zelfs al loopt de kaart op 3,57 MHz.
Op dezelfde manier wordt ook een database per record geleegd. Een SELECT is dan ook niet mogelijk, want dat zou een hele grote dump kunnen leveren over een heel dun lijntje. In plaats daarvan is er een DECLARE CURSOR statement waarin een SELECT query wordt opgenomen; met behulp van OPEN, FETCH, NEXT en FETCH NEXT is hiermee door de lijst van records te navigeren. Overigens maken ook de UPDATE en DELETE gebruik van de cursor. Wederom krijgt de programmeur die de kaart gebruikt een beetje meer werk dan bij een normaal DBMS.
Er mag op elk moment slechts één cursor actief zijn. Dat wil dus zeggen dat SCQL geen noodzaak tot *concurrency management* introduceert. Gezien de afmetingen van een smart card, en het persoonsgebonden karakter van gebruik, is dat een redelijke vereenvoudiging – het bespaart werk waarvan het voordeel zelden of nooit duidelijk zal worden. De enige opmerking in dit opzicht is dat het wel mogelijk is, op ondersteunende kaarten, om meerdere sessies tegelijk te draaien op een enkele kaart. Dat wordt onder andere uitgedrukt in de protocol-byte genaamd CLA (zie kader).

Voorbeeld Binaire INSERT

Als indicatie van het binaire formaat dat de smart card accepteert in plaats van volledig SQL volgt hier een voorbeeld, namelijk:

```
INSERT INTO EMPLOYEE VALUES
                ('Jan', 'Janssen', 'Sales');
```

Dit statement kan worden verzonden in een ongecodeerde of een gecodeerde verbinding, de zogenaamde CLAss van de verbinding. Voor normaal verkeer is dat de hexadecimale waarde 0x00. Vervolgens volgt de INStructiebyte, in dit geval voor PERFORM SCQL OPERATION, ofwel de code 0x10. Dan is er nog ruimte voor 2 parameterbytes. P1 staat altijd op 0x00, zodat toekomstige uitbreidingen mogelijk zijn, P2 wordt 0x8C om INSERT aan te duiden. Dit is de header van het commando.
De body bestaat uit een lengte, gevolgd door de tabelnaam, het aantal kolommen en dan dat aantal velden. Alle strings worden voorafgegaan door een byte met de lengte die volgt.
Het totale protocol-pakket voor deze SCQL-instructie is dan:

```
CLA  0x00            // Gewone verbinding
INS  0x10            // PERFORM SCQL
                        OPERATION
P1   0x00            // Gereserveerd voor
                        later
P2   0x8C            // INSERT
Lc   0x1C            // Lengte van data
Data 0x08,"EMPLOYEE" // Tabelnaam
     0x03            // 3 Kolommen
     0x03,"Jan"      // Value #1, 'Jan'
     0x07,"Janssen"  // Value #2, 'Janssen'
     0x05,"Sales"    // Value #3, 'Sales'
Le   (leeg)          // Lengte van antwoord
```

De kaart reageert hierop met een code die O.K. of een foutmelding aangeeft. Andere instructies vertalen naar een binair formaat via een soortgelijk stramien. De vertaling is dus zo triviaal dat hier zonder meer een SQL-interpreterende driver voor te bouwen is.

Communiceren met een smart card

De aansluitingen op een smart card zijn gestandaardiseerd. Zowel de plek op de kaart, als de signalen die erop worden aangesloten:

Vcc		GND
RST		Vpp
CLK		I/O
N/C		N/C

De aansluitingen Vcc en GND zijn de voeding en het nulniveau dat als referentie voor alle signalen dient; Vpp is soms een programmeerspanning voor flash of EEPROM. Op CLK staat een kloksignaal van een aantal MHz. Met RST wordt de interne chip gereset en via I/O wordt een bidirectionele, half-duplex seriële verbinding opgezet. Al die signalen worden door een kaartlezer bediend.

Van de seriële verbinding weten de smart card en de kaartlezer op elk moment wie aan de beurt is om te zenden. In principe transporteert de kaartlezer, meestal op verzoek van een aangesloten computer, een opdracht naar de kaart, waarop de kaart dan een reactie terugstuurt. De kaart neemt geen initiatieven.

Het transporteren van de opdrachten en antwoorden gebeurt met ingebouwde controles. Hiervoor is een paar inpak-protocollen gedefinieerd. Voor kaarten met een contact-interface zijn dat T=0 en T=1. De eerste doet controles per byte, de tweede per datablok.

SIM-kaarten hebben een andere kaartgrootte, maar ze voldoen ook aan deze aansluitingen. Er zijn verder nog draadloze kaarten en kaartlezers die op dit lage niveau anders, maar op hogere niveaus eender werken.

Strikt genomen is het denkbaar dat daar interpretatieverschillen tussen kaarten ontstaan, en het is niet handig dat de standaard dit niet voorkomt.

Verschillen in SQL-model

Behalve verschillen op het allerlaagste niveau zijn er ook grote verschillen tussen SCQL en SQL zoals een normale DBMS dat implementeert. In een normale SQL-database zijn aannames over volgorde van records uit den boze, tenzij er expliciet gesorteerd wordt. De oorzaak daarvan is dat SQL geënt is op verzamelingenleer. In SCQL geldt dat een INSERT altijd toevoegingen doet aan het eind van de tabel, dus daar is wel degelijk iets aan te nemen over de opslagwijze. Dat is eigenlijk heel vreemd, behalve dat het nuttig kan zijn voor een driver die er bovenop gebouwd wordt.

Verder is het spijtig dat de standaard vastlegt dat tabellen niet gecombineerd kunnen worden. De condities die in een SELECT kunnen worden gebruikt vergelijken altijd een kolom met een waarde, en dus nooit een kolom met een kolom. Een simpele join is dus onbegonnen werk. Maar ook gebeurt een SELECT of een VIEW altijd op basis van een enkele tabel.

Wie denkt dat dit alles te vergoelijken is op grond van de eenvoud van de processortjes op een smart card die vergist zich lelijk. Deze processoren kunnen allemaal wel op zo'n 4 MHz lopen, en hoewel ze vaak met maar een paar KB RAM zijn uitgerust, hebben ze wel tientallen KB flash memory of EEPROM 'aan boord'. Ze zijn dus vergelijkbaar met een Commodore 64 of een ZX Spectrum 48K – en die konden dit soort werk echt wel verzetten. Maar doordat de standaard zichzelf beperkingen oplegt, zijn JOIN's en dergelijke alleen beschikbaar te maken als *proprietary* uitbreidingen op de standaardprotocollen.

Transacties

Smart cards ondersteunen transacties. Mocht dat raar klinken, bedenk dan dat het gaat om devices die op elk moment door een onnadenkende gebruiker uit de kaartlezer gerukt kunnen worden. Dit schakelt de processor op de kaart uit, om de volgende keer een *koude reset* door te maken. Bij zo'n reset is het erg prettig als de laatste stabiele toestand hersteld kan worden. En aldus geschiede.

De transacties die ook in de SCQL-standaard besloten zitten zijn eenvoudige BEGIN/COMMIT/ROLLBACK transacties, zonder de mogelijkheden van two-phase commit. Dat is ook wel logisch, want na een PREPARE kan de kaart zich opmaken voor een COMMIT, maar als dan de kaart uit de lezer wordt getrokken kan alsnog verwarring ontstaan.

Merk op dat het in een gedistribueerde database mogelijk is om een enkele node zonder two-phase commit te laten werken; daartoe zendt een transaction manager eerst PREPARE naar alle andere nodes, bij succes vervolgens COMMIT naar de kaart, en als dat slaagt krijgen alle andere nodes ook een COMMIT en anders een ROLLBACK. Er is alleen geen mechanisme om bijvoorbeeld twee kaarten te laten samenwerken in een transactie. Aangezien dat fundamenteel onmogelijk is, kan daar ook moeilijk iets tegenin worden gebracht.

Toepassingen?

Het grote probleem van deze standaard is wellicht de toepasbaarheid. Er lijkt geen commerciële activiteit te bestaan rond dit deel van ISO-7816, in scherp contrast met de overige delen. Wellicht is dit eruit te verklaren dat een smart card juist floreert als de kaart code bevat die de data intern verwerkt. Een database lepelt slechts data op zonder er iets mee te doen; maar SCQL kan er niet eens aan rekenen.

Maar misschien zien we allerlei zinnige toepassingen over het hoofd. Als dat zo is dan vernemen we dat graag, op onderstaand e-mail-adres. Wie weet is er een toepassing die zat te wachten op dit mechanisme!

Rick van Rein

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