Relationele databases en SQL: achtergronden en didactiek

Een SQL-boek met interactieve 'Boekverkenner'

Leo Wiegerink
Jeanot Bijpost
Marco de Groot

Samenvatting
Een elektronische leeromgeving die eenzijdig voortkomt uit de 'push' van de technologie zal vaak teleurstellen. De SQL-cursus die in dit artikel centraal staat, is ontwikkeld vanuit de erkenning dat een boek in veel opzichten een superieur medium is. Een papieren cursusboek, met een didactisch hoogwaardige inhoud, was daarom het uitgangspunt. De auteurs hebben er echter een elektronische versie aan toegevoegd, die de student alle routinewerk uit handen neemt: de interactieve 'Boekverkenner'.
Dit artikel gaat in op de inhoud en de didactiek van het papieren boek, behandelt de meerwaarde van de Boekverkenner, en gaat in op de gebruikte technologie. Tot slot wordt even vooruit gekeken naar de in het najaar te verschijnen nieuwe cursus 'Databases' van de Open Universiteit Nederland, die volgens hetzelfde concept wordt ontwikkeld.

1. Zoeken naar de ideale elektronische leeromgeving

We wilden het ideale boek maken over 'relationele databases en SQL': een hoogwaardige inhoud en didactiek in een behuizing van ouderwetse kwaliteit die méérdere zintuigen aanspreekt. Het moest daarom een fraai uiterlijk hebben, lekker aanvoelen en zelfs lekker ruiken. Je moest er mee onderuit kunnen zakken en in kunnen schrijven, zoals bij alle boeken die je tot je geestelijk eigendom maakt door ze letterlijk stuk te lezen. De technologie lag dus vast: papier! Of toch niet? Want we wilden nog zoveel méér: met de achterkant van je potlood moest je een SQL-statement kunnen aantikken, waarna de tekst plaats maakte voor ... Om kort te gaan, het moest een boek worden dat het beste in zich verenigt van papieren en elektronische media. Hoewel we misschien beter nog twintig jaar hadden kunnen wachten, zijn we toch maar gewoon begonnen.

Het uitgangspunt werd een papieren boek, met zijn vooralsnog onvervangbare kwaliteiten. En we hebben ons de vraag gesteld: hoe creëer je met de huidige stand van de technologie een zo zinvol mogelijke elektronische aanvulling? Het woord 'technologie' lijkt misschien het toverwoord in deze vraag, maar is het juist niet. Een ontwerp dat te sterk vanuit de technologie is ontwikkeld, resulteert vaak in een trukendoos met maar weinig échte meerwaarde. Liever kiezen we voor een benadering vanuit het boek zélf. Je pakt een boek, gaat zitten en vraagt je tijdens het lezen af "wat zou ik nu willen kunnen doen?".

De antwoorden bleken niet zo moeilijk: de student moet kunnen experimenteren, en opgaven kunnen maken zonder de drempel van vervelend routinewerk. Zonder angst een voorbeelddatabase te ruïneren, want die moet in een handomdraai te herstellen zijn. De student moet snel alle informatie over alle voorbeelddatabases bij de hand hebben, zoals diagrammen en scripts. Van de SQL-programmeeromgeving mag je verwachten dat deze de student/programmeur verwent met anderhande gemakken. En zo kwamen we van het een op het ander: tenminste één voorbeelddatabase van realistische grootte, extra SQL-interfaces voor illustratie van multi-user problematiek, enzovoort. Vanuit deze meerwaardebenadering is de 'Boekverkenner' ontworpen. Maar eerst was er het papier ...

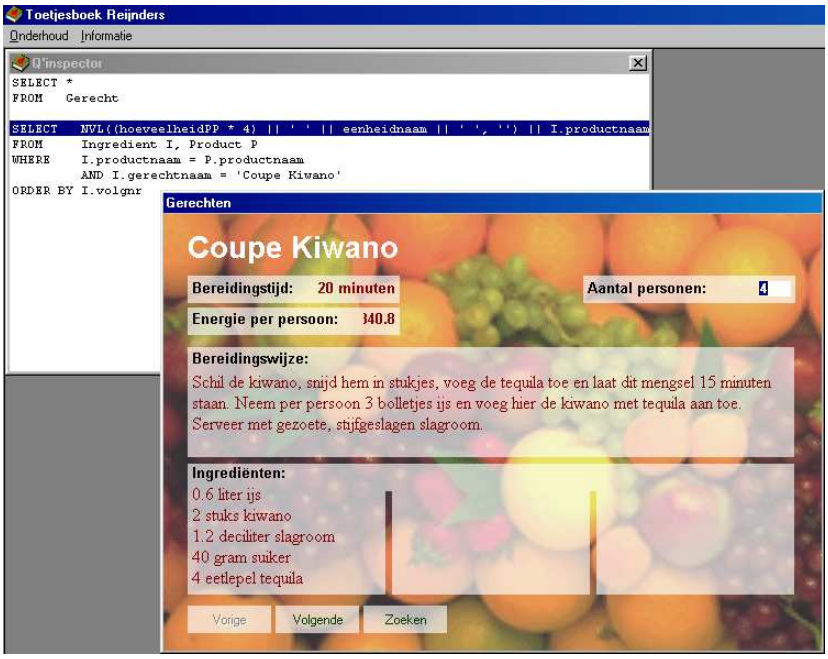
2. Het ‘papier’: inhoud en didactiek

Al schrijvende (en vooral herschrijvende) hebben de ideeën voor de inhoud en de didactiek van het papieren boek vorm gekregen. Hoewel beide nauw samenhangen, geven we eerst een schets van de inhoud, waarna we ingaan op enkele didactische uitgangspunten.

Inhoud

Het boek bestaat uit vijf delen.
Deel A ‘Relationele databases’ begint met een praktische introductie in de ‘achterkant’ van een relationeel systeem: structuur en regels van een relationele database. Hoewel normalisatie als ontwerpmethode is achterhaald, zijn ‘normalisatie en standaardisatie’ als uitgangspunt genomen om voor een eenvoudig voorbeeld (een kookboek voor lekkere toetjes) verschillende ‘soorten van dingen’ naar hun eigen tabel toe te praten. Dit blijkt nog steeds een zeer geschikte methode om fundamentele begrippen zoals redundantie en (in)consistentie te introduceren. De volgende hoofdstukken gaan over de communicatie met een relationele database: twee ‘voorkant’-programma’s (een SQL-tool en de grafische applicatie ‘Toetjesboek Reijnders’, zie figuur 1) en SQL-in-vogelvlucht. Dan volgt een hoofdstuk gewijd aan problemen en beperkingen van relationele structuren, waarin ook normalisatie weer terugkomt. Netjes, maar niet formeel.
De delen B ‘Relationele databases bevragen en wijzigen’ en C ‘Relationele databases beheren’ bevatten een gedegen SQL-cursus. Een bijzonder hoofdstuk hierin is het hoofdstuk over query-optimalisatie, met een voorbeelddatabase van realistische grootte: 115.000 records.
Deel D ‘Verdieping’ sluit het boek af met erg vier mooie en nuttige hoofdstukken, over resp. probleemoplossen, transacties en multi-usergebruik, triggers en stored procedures, en de data dictionary.
Deel E tenslotte bevat bijlagen, waarvan die met diagrammen van voorbeelddatabases de belangrijkste is.

Voor de SQL-voorbeelden en -opgaven wordt gebruik gemaakt van het rdbms (relationeel databasemanegementsysteem) Firebird. De SQL van dit product is grotendeels conform de SQL2-standaard.



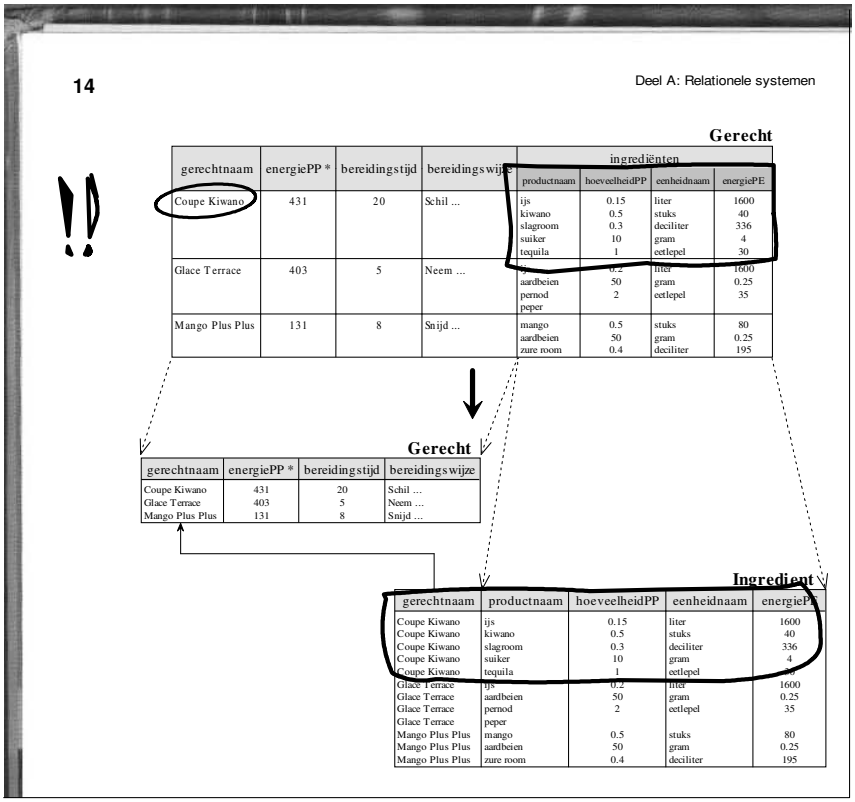
Figuur 1 Grafische ‘Toetjesboek’-applicatie, met de naar het rdbms verzonden SQL-opdrachten zichtbaar gemaakt in het ‘Q-inspector’- venster

'Single point of ...'

Het databaseprincipe 'single point of definition' (SPOD) vervult een belangrijke rol in de cursus. Niet alleen in verband met gegevensopslag, maar overal waar je consistentie kunt bevorderen door dingen op het juiste niveau te definiëren. Denk hierbij ook aan privilegeverstrekking door middel van rollen, en het vastleggen van datatypen via domeinen. Steeds wordt dit voor de student nadrukkelijk ook zo benoemd.

Maar SPOD kent nog meer gedaanten. Zoals: 'single point of programming style', wat niets anders betekent dan: programmeren volgens een strakke *huisstijl*. We pleiten voor het durven stellen van eisen: houd je aan 'de' stijl van de cursus, of aan je eigen *gedocumenteerde* programmeerstijl.

Een andere, didactische variant: 'single point of explication'. Ofwel: hang een cursus niet op aan de veelheid van theorieën, methoden en technieken die de praktijk te zien geeft (een fenomenologische aanpak). En evenmin aan een historische aanpak, die daar nauw aan verwant is. Zelf hebben we geprobeerd een strakke lijn uit te zetten, en in onze uitleg de grootst mogelijke consistentie te betrachten. Een voorbeeld: een 'herhalende groep' is van meet af aan één – gestructureerde – kolom, zie figuur 2. Later kan daar dan 'vanzelf' op worden aangesloten bij het behandelen van gestructureerde datatypen. En de 'moderne' discussie over de niet-vanzelfsprekendheid van de eerste normaalvorm kan glashelder worden uitgelegd.



Figuur 2 Een bijzondere technologie: papier! Voorts: normalisatie gevisualiseerd, met een herhalende groep als één, gestructureerde kolom

Conceptuele benadering

De ‘single point of explication’-benadering kan ook als volgt worden geformuleerd: ‘conceptueel correct is didactisch meestal nog het eenvoudigst’. Zo’n strakke lijn blijkt uitstekende mogelijkheden te bieden om de fenomenen van de praktijk en van de historie van het juiste referentiekader te voorzien. De theorie ís niet zo moeilijk, als we de historie achter ons durven laten en beseffen dat ‘het’ relationele model helemaal niet bestaat.
Zo’n conceptuele benadering biedt talloze mogelijkheden om de theorie levend te houden. Bijvoorbeeld: de relationele join- en groeperingsoperaties komen neer op verschillende vormen van de-normalisatie. Ook taalkritiek hoort tot zo’n benadering: de tekorten van SQL blootleggen en nadenken over wat een ‘relationeel complete’ taal eigenlijk zou moeten bieden.

Visualisatie en schemavorming

Bij de ‘nette maar niet formele’ aanpak die we hebben nagestreefd, vervullen de vele plaatjes een essentiële rol. Ze dragen bij aan abstractie, in de zin van het naar voren halen van de essentie. En aan *schema*-vorming, in de betekenis die Richard Skemp daaraan heeft gegeven in zijn ‘Psychology of learning mathematics’[1]: het vormen van mentale modellen, met sterk visuele en sensomotorische aspecten. Door hun consistentie en eenvoud vormen deze een solide kapstok voor verdergaande kennis- en inzichtverwerving.
Basis voor schemavorming was in ons geval een consequente weergave van (niet-recursieve) ouder/kind-relaties in databasediagrammen: ouder boven, kind beneden. Anders gezegd: de richting boven-beneden correspondeert *bijna* altijd met één-veel. Slechts een enkele keer (bij een cyclische verwijzingsstructuur) hoeft dit principe te worden doorbroken[2]. Zo worden spaghettidiagrammen vermeden en belangrijker: er is optimale ondersteuning voor de idee van *databasenavigatie*, in onze visie de rode draad bij SQL-programmeren.

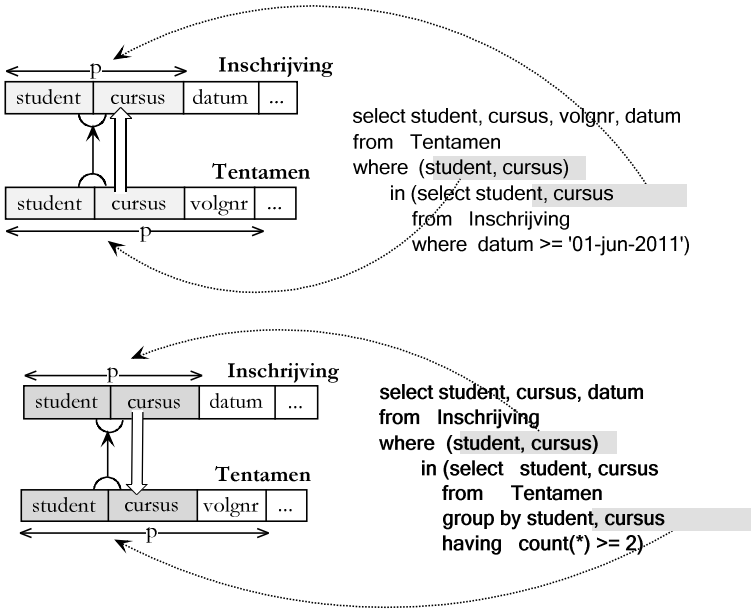
Databasenavigatie

Het oplossen van een queryprobleem zou moeten beginnen bij de vraag: over welke ‘soorten van dingen’ willen we informatie? Dit geeft de starttabel van het *navigatiepad* (eigenlijk is ‘boom’ een betere term). Verdere details van de vraagstelling (welke gegevens willen we zien? aan welke voorwaarden moeten die nog voldoen?) bepalen hoe we van daaruit door de database moeten navigeren.
Essentieel is dat je vanuit een rij ‘naar boven’ navigerend steeds bijbehorende *waarden* vindt in ouder- of grootoudertabellen. Je gaat immers de één-kant op. En dat je ‘naar beneden’ navigerend altijd een *herhalende groep* vindt. Je gaat immers de veel-kant op. Veelal is daarvan een statistisch kental nodig, in welk geval de navigatie technisch wordt gerealiseerd door een join gecombineerd met groeperen. Gaat het alleen om een conditie dan is ook (en soms alléén) subselectnavigatie een optie, zie figuur 3 voor een illustratie.

---

[1] Skemp R. (1971), *The psychology of learning mathematics*, Penguin Books

[2] Maar zelfs dan gaat het principe ‘eigenlijk’ op: de cyclische verwijzing is slechts *pseudo*-cyclisch. Het gaat immers om een potentieel meervoudig gebruikte tabel, waarvan in wezen twee (of meer) exemplaren onafhankelijk van elkaar worden gebruikt. Overigens is dat iets dat voor elke tabel geldt: kom je via een navigatiepad weer ‘dezelfde’ tabel tegen, dan gaat het in feite om een nieuw exemplaar van die tabel. Langs zo’n navigatiepad hebben we daardoor uitsluitend gewone kind-ouder- of ouder-kindassociaties. We kennen dat natuurlijk van SQL-statements, waarin we meerdere tabelexemplaren daadwerkelijk van elkaar onderscheiden door het gebruik van tabel-aliasssen.



Figuur 3 Kind-ouder- en ouder-kindnavigatie, technisch gerealiseerd via een subselect, en didactisch ondersteund door een strakke tekenconventie (ouder boven, kind beneden)

Hier is ware schemavorming opgetreden, in de zin van Skemp: 'naar boven' en 'naar beneden' staan voor een heel complex aan achterliggende concepten. Zo staat 'naar beneden' voor: 'van één naar veel', 'van rij naar bijbehorende herhalende groep', 'van primaire sleutel naar verwijssleutel', 'van rij naar corresponderende statistische kentallen', enzovoort. Er is eenheid in verscheidenheid ontstaan, met name mentaal. Ofwel: inzicht. Tevens hebben we een sleutel in handen tot een *leerbare* probleemaanpak en een heldere programmeerstijl. Goed begrip van de join is daarbij essentieel.

De join

De join (inner of outer) wordt geïntroduceerd als een 'tabelverbreding' van de *linker* tabeloperand, hetzij met waarden opgehaald via navigatie 'naar boven' (zie Boekverkenner-window van figuur 5), hetzij met een herhalende groep van waarden opgehaald door navigatie 'naar beneden'. Conceptueel gezien kan zo'n join dus ook een herhalende groep introduceren. In het relationele model zoals we dat vooralsnog kennen, kunnen we zo'n herhalende groep natuurlijk alleen 'relationeel platgeslagen' ofwel genormaliseerd voor de dag toveren. De nadruk op *linker* heeft te maken met het strak volgen van het navigatiepad, conceptueel en *dus* ook in de programmeerstijl. Ondermeer wordt geëist dat alle volgorden van tabellen (als joinoperands en in condities) overeenkomen met het navigatiepad. Hierdoor wordt ook het ad hoc gebruik van de right-outer-joinoperator (in onze visie een 'lelijke' en overbodige operator) voorkomen.

Probleemaanpak en programmeerstijl

We beperken ons tot opvragingen. Veelal kan parallel aan de reis door de database, langs het navigatiepad, stap voor stap de select-query worden opgesteld. Hierbij wordt een bekend top-down principe gevolgd: beginnend in natuurlijke taal (het probleem), gaan we in zoveel stappen als nodig is over op SQL-met-subproblemen-in-natuurlijke taal. Tot uiteindelijk álles SQL is. Alles? Formeel wel, maar de deelproblemen-in-natuurlijke-taal blíjven zichtbaar in de vorm van commentaar. Gecombineerd met strenge regels voor naamgeving en opmaak is het resultaat: leesbare en controleerbare SQL-code. Aanvullend kunnen natuurlijk om technische redenen (performance) modificaties worden gemaakt. Zo is het soms wenselijk een ander startpunt van de navigatie te kiezen dan overeenkomt met het conceptuele startpunt. Je moet daarmee echter niet willen beginnen. Zie figuur 4 voor een fragment van het 'oplossingsverhaal' bij een probleem van een Ruimtereisbureau: "welke deelnemers (deelnames!) hebben geboekt voor een interplanetaire reis zonder eerst een Maanreisje te hebben gemaakt?". Het 'verhaal' gaat over veel meer dan navigatie en stapsgewijs

oplossen: het begint bij goed (of eigenlijk: verkeerd) begrip van de vraagstelling en laat tevens zien wat het gevolg is van overhaast uitwerken van een eerste idee.

...

Stap 2: hoe zou je het probleem handmatig oplossen?
Je zou de voorbeeldpopulatie als volgt kunnen onderzoeken: per deelname kijk je of het een deelname aan een niet-Maanreisje is. Als dat het geval is, kijk je of er een eerdere deelname is van dezelfde klant aan een Maan-reisje.

Stap 3: kan het ook anders, handiger misschien?
Zoals helaas in de praktijk vaak gebeurt, slaan we deze belangrijke stap even over. We zullen zien wat de gevolgen zijn.

Stap 4: stapsgewijze transformatie naar SQL

Eerste deelstap:

```
SELECT K.naam, D.reisnr, R.vertrekdatum
FROM   Deelname D
       JOIN Reis R  ON ...
       JOIN Klant K ON ...
WHERE  dit is een deelname aan een niet-Maanreisje
       AND
       er is géén Maanreisje van dezelfde klant, met een eerdere vertrekdatum
```

Tweede deelstap:
...

Figuur 4 Stapsgewijze probleemaanpak (fragment)

Leren denken op metaniveau

Waarom is rommelen in de systeemcatalogus zo leuk? Misschien omdat je iets doet dat eigenlijk niet mag? Ongetwijfeld, maar een deel van het plezier is dat je er 'eindelijk' bepaalde dingen door gaat snappen, dat er van alles op zijn plaats valt. Je gaat 'eindelijk' snappen wat er met al je CREATE TABLE-statements is gebeurd. En ook: je ziet welke verrassende eenvoud schuil gaat onder alle complexiteit. Je ziet dat DDL eigenlijk gewoon rijen invoegt, verwijdert of wijzigt. En wat is er dan leuker dan een tabel te creëren via INSERT-statements, en die vervolgens met DROP TABLE weer te verwijderen?
Een Firebird-database is van één eigenaar, en de data dictionary zit glashelder in elkaar. Didactisch is dat mooi meegenomen. Het bleek weinig moeite te kosten de lezer mee te nemen in een verhaal waarin metagegevens hele gewone gegevens zijn geworden. Maar die lezer overziet het relationele landschap daarna toch met andere ogen en met meer inzicht.

3. De electronisch toegevoegde waarde: Boekverkenner, IQU en rdbms

De Boekverkenner kan niet zonder de SQL-programmeeromgeving, de Interactive Query Utility. En deze kan op zijn beurt niet zonder het rdbms, Firebird.

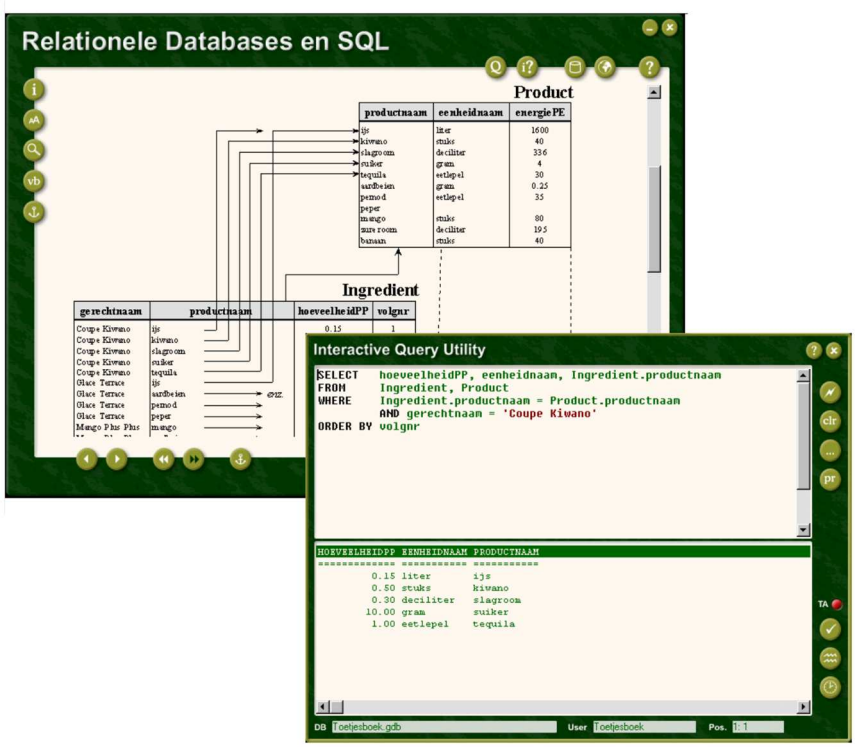
Interactive Query Utility

De Interactive Query Utility, kortweg IQU, is een hoogwaardige SQL-programmeeromgeving, die in de plaats is gekomen van het SQL-tool van Firebird. Zie figuur 5, waarin IQU vanuit het hoofdvenster van de Boekverkenner is geopend. Ook hierbij was de ontwerpleidraad: "wat zouden we willen kunnen?", met inhoudelijke en didactische functionaliteiten voorop. Bijvoorbeeld: de mogelijkheid verschillende transactiemodellen in te stellen, een 'lampje' dat aangeeft of er een transactie gaande is, en een knopje voor tijdmeting en queryplan (voor het hoofdstuk over optimalisatie). Voor illustratie van multi-user problematiek (concurrente transacties) kan met één muisklik een tweede IQU-venster worden geopend, en eventueel nog één en nog één ... Vervolgens zijn er de 'gewone' prettige dingen zoals navigeren door de opdrachtenhistorie, en sneltoetsen om even snel een tabelstructuur of

tabelinhoud te inspecteren bij een geselecteerde tabelnaam. Het is een tool geworden voor de verwende programmeur/student/docent.

Boekverkenner

Met IQU kun je alles doen, als het om programmeren gaat. De eigenlijke Boekverkenner maakt dat echter een stuk handiger. In de elektronische tekstversie kan bijna elk SQL-statement worden aangeklikt, waarna de voorbeelddatabase wordt herkend (als deze nog niet was geïnstalleerd gebeurt dat alsnog), de connectie wordt gemaakt (indien nodig), en het statement in het IQU-venster verschijnt. Daar kan de gebruiker ermee doen wat hij wil: direct uitvoeren, of eerst aanpassen. Wie er een puinhoop van maakt, hoeft niet te vrezen: met een paar muisklikken is de oude situatie hersteld. Daarom kunnen ook veilig de metadata worden geruïneerd, wat van harte wordt aanbevolen in het hoofdstuk over de systeemcatalogus (zie de paragraaf 'Leren denken op metaniveau'). Via het databaseknopje ('koektrommeltje') kan los van de tekst worden ingelogd op een voorbeelddatabase. Ook kunnen deze hiermee in een paar muisklikken worden geïnstalleerd, of verwijderd en desgewenst geherinstalleerd.



Figuur 5 Boekverkenner met Interactive Query Utility

Een andere belangrijke voorziening zit verstopt onder het Vb-knopje. De student heeft hier alle informatie over elke voorbeelddatabase direct bij de hand: een strokendiagram, een cardinaliteitendiagram, een diagram met de voorbeeldpopulatie, alle scripts en bij het Toetjesboek ook de grafische eindgebruikersapplicatie met SQL-'Q'inspector'.
Dit was de kern, de rest is franje. Wel mooie franje. Wie graag wil zoeken in de tekst met zoekexpressies vol AND's en OR's of wie een bookmark wil plaatsen, kan zijn gang gaan. Handig zijn vooral de historienavigatie en de sneltoetsen (zoals voor snel wisselen tussen opgave en uitwerking).

Firebird

De keuze voor het Firebird-rdbms lag van begin af aan vast. We wisten dat dit de keuze was voor een robuuste database, die wel tegen een stootje ('ruïneren en weer opbouwen') kon. Mattic Software (zie de auteursinformatie) had veel ervaring met Firebird in grotere projecten, onder andere in een gedistribueerde omgeving met zeven servers en een honderdtal clients, die al meerdere jaren zonder

één enkele databasestoring draaien. Ook belangrijk was dat Firebird-SQL dicht bij de SQL2-standaard ligt. We wilden zo min mogelijk dialectafhankelijke SQL-code. Andere overwegingen waren: eenvoudige installatie (een niet te onderschatten detail) en het feit dat de Firebird-API een goede koppeling mogelijk maakt met de Boekverkenner-programmatuur.
Misschien aardig om nog te vermelden: wie goed oplet zal merken dat Firebird-SQL via IQU nog wat is opgepoetst. Daardoor wordt bijvoorbeeld de SQL2-manier van commentaar geven ondersteund. Ook vallen hieronder het instellen van verschillende transactiemodellen en het automatisch installeren van een batterij user defined functions.

4. De Boekverkenner-technologie

De 'boekdatabase'

De kern van de technologie is de 'boekdatabase', genaamd RelSQL: het hele boek in al zijn onderdelen uiteengerafeld en opgeslagen in een relationele database. Ook de voorbeelden met hun diagrammen en scripts maken daar deel van uit. Vanuit de boekdatabase worden de Boekverkenner-schermen gegenereerd, met hun dynamische (aanklikbare) SQL-statements. De boekdatabase is in een aantal stappen tot stand gekomen, zie figuur 6.

Figuur 6 Van boekdocument naar boekdatabase

Toelichting: startpunt is een boektekst-document in Word, met een zeer strakke toepassing van stijlen (stap 1). Via verborgen tekst is bij elk SQL-statement informatie opgenomen over de voorbeelddatabase. Het Word-document wordt geëxporteerd naar HTML, met XML-codes (stap 2). De nogal vervuilde code moet danig worden opgeschoond (stap 3), waarna het document in zijn verscheidene onderdelen wordt uiteengerafeld en opgeborgen in de boekdatabase (stap 4). Tenslotte moeten er handmatig nog heel wat puntjes op de i worden gezet, waaronder verbetering van de opmaak (stap 5). Stap 4 is natuurlijk het meest complex, en omvat het uit de tekst halen en in de database opbergen van de SQL-statements, de figuren, koppen van verschillende niveaus, de tekstfragmenten, trefwoorden, enzovoort. En dat alles gekoppeld aan informatie over de hiërarchische tekststructuur.

Boekdatabase tevens voorbeelddatabase

Aardige details zijn dat de boekdatabase RelSQL ook het interfaceontwerp (gemaakt in Fireworks) bevat, en dat hij via de Boekverkenner zelfs opvraagbaar is en te gebruiken als voorbeelddatabase. Wie dat wil kan de structuur zó achterhalen. De inhoud zal misschien wat teleurstellen: RelSQL is een multimedia database, met zeer veel blobs. In verband met copyright moesten deze wel encrypted en gecomprimeerd worden opgeslagen.

5. Conclusie

We hebben een SQL-cursus ontworpen waarbij degelijke inhoud en doordachte didactiek voorop stonden en die uitgaat van de eigen, onvervangbare kwaliteit van mooi drukwerk. Door een conceptuele aanpak, ondersteund door veel figuren met strakke tekenconventies, wordt de basis gelegd voor inzicht en overzicht, voor een goede programmeerstijl en een gestructureerde aanpak van problemen. Databasenavigatie is hierbij een verbindend thema.
Vanuit onderzoek naar échte meerwaarde van een electronisch medium is de 'Boekverkenner' ontworpen, die de tekst van het boek koppelt aan de SQL-programmeeromgeving. Belangrijkste functionaliteiten: query's die vanuit de tekst kunnen worden uitgevoerd, automatisch installeren van en inloggen op de voorbeelddatabases, directe beschikbaarheid van alle voorbeeldinformatie, zoals diagrammen en scripts. Voor communicatie met het rdbms (Firebird) is een programmeursvriendelijke programmeeromgeving ontworpen: IQU (Interactive Query Utility). Via IQU is extra functionaliteit

toegevoegd, zoals het instellen van meerdere transactiemodellen en een didactische vorm van multi-user gebruik.

Het 'RelSQL'-boek

Leo Wiegerink, Jeanot Bijpost en Marco de Groot (2000), *Relationele databases en SQL*, 3e geheel herziene druk - Boom, Amsterdam, isbn 978 90 395 2714 6.
Websites: relsql.nl en boomhogeronderwijs.nl/product/100-4176_Relationele-databases-en-SQL.

De auteurs

drs. L.J.G.M. Wiegerink, voorheen werkzaam bij de Hogeschool van Amsterdam (HIO), in het bedrijfsleven en bij de Open Universiteit Nederland (e: leowiegerink@gmail.com).
ing. J.W.Bijpost en ing. M.H. de Groot, werkzaam bij Mattic Software (w: mattic.com).

Verder werkten mee: dr. N. van Vugt, B. Pauw en drs. H. Pootjes, allen werkzaam of werkzaam geweest bij de Open Universiteit Nederland.