Leren Programmeren in het Voortgezet Onderwijs:

Probleemoplossen met SQL[1]

E.M.A.G. van Dijk en H.P.M.Krammer, Universiteit Twente, Interfacultair Werkverband Lerarenopleiding
J.J.G. van Merriënboer, Universiteit Twente, Vakgroep Instructietechnologie

Summary
The claim that computer programming enhances problem solving skills is one of the main arguments to teach imperative, general-purpose languages (e.g. BASIC, Pascal) in secondary education. However, evidence is beginning to accumulate that imperative languages are inappropriate for educational demands. Reported learning outcomes have been rather disappointing, and there is yet little evidence for transfer of acquired skills to other domains. In this article, it will be argued that a database query language such as SQL is more appropriate to teach introductory programming in secondary education. Working with SQL only requires a very simple mental model and its syntax is easy to learn, so that full attention can be given to problem solving itself; this enhances the possibility of transfer to other domains. Moreover, in the near future only specialists will write programs in imperative languages, whereas SQL is likely to become an important, widely used language.

1. Inleiding

Het kan nauwelijks toeval zijn dat, juist nu besloten is dat informatica deel zal gaan uitmaken van het reguliere programma van het voortgezet onderwijs, er discussie is ontstaan over de vraag of leren programmeren deel moet uitmaken van informatica-onderwijs. Dit artikel wil aan deze discussie een bijdrage leveren, door de argumenten ten gunste van programmeeronderwijs te inventariseren. Daarbij moet niet in de eerste plaats gedacht worden aan leren programmeren in imperatieve talen (zoals BASIC en Pascal), aangezien langzamerhand duidelijk wordt dat er bezwaren verbonden zijn aan het leren programmeren in deze talen. Op zoek naar meer geschikte talen voor

het programmeeronderwijs wordt één voorbeeld gegeven: de database vraagtaal SQL (Structured Query Language). De keuze van dit voorbeeld betekent niet dat andere talen, zoals LOGO of PROLOG, niet geschikt zouden kunnen zijn voor programmeeronderwijs; mogelijk zijn de redenen die genoemd zullen worden om SQL te kiezen evenzeer van toepassing op deze talen.

In een doelstellingenonderzoek voor het vak informatica in de bovenbouw (uitgaande van de toen geldende beleidsvoornemens van de minister: een afzonderlijk examenvak informatica in HAVO en VWO), bleek ruim 80% van de geënquêteerden van mening dat leerlingen in het voortgezet onderwijs (HAVO en VWO) een programmeertaal dienen te leren (Koers, 1985). Gemiddeld zou volgens de respondenten 40% van de voor het vak informatica beschikbare tijd moeten worden uitgetrokken voor programmeren. Overigens gaven de resultaten van factoranalyses voor VWO- en HAVO-doelstellingen aanleiding om de volgende twee aspecten van programmeren te onderscheiden: probleemoplossen en codeervaardigheden. Terwijl er wat betreft het belang van probleemoplossen weinig verschil tussen de respondentgroepen te zien was, vonden hoogleraren informatica en informaticadocenten uit het HBO de codeervaardigheden minder belangrijk dan de andere respondentgroepen (Krammer & Koers, 1986).

Inmiddels heeft de minister van onderwijs en wetenschappen in een beleidsplan aangekondigd dat informatica niet als eindexamenvak in HAVO en VWO zal worden ingevoerd, maar dat dit vak een plaats zal moeten krijgen in de middenbouw (derde of vierde klas), en dat daarnaast elementen van informatica moeten worden geïntegreerd in daarvoor in aanmerking komende examenvakken. De vraag is nu of programmeeronderwijs in enigerlei vorm onderdeel moet worden van het zogenoemde basisprogramma informatica in de middenbouw.

Voorzover er in het voortgezet onderwijs ervaring is opgedaan met inleidend programmeeronderwijs, betreft dat meestal lessen waarin gebruik gemaakt wordt van imperatieve, general-purpose talen. De oplossing voor een probleem bestaat bij een imperatieve taal uit een opsomming van instructies die de computer achtereenvolgens moet uitvoeren om tot het gewenste resultaat te komen. Een programma in een imperatieve taal voldoet derhalve aan een definitie van 'programma' uit het woordenboek: 'Een geschrift waarin werkzaamheden die volgens een bepaald plan zullen worden verricht, staan opgesomd' (Van

Dale, 1984). Tegenwoordig voldoen computerprogramma's niet altijd aan deze definitie. Programma's die geschreven zijn in zogenaamde declaratieve talen (bijvoorbeeld Lisp en PROLOG) beschrijven alleen de kenmerken van het gewenste eindresultaat, niet hoe dit resultaat bereikt moet worden. Ook toepassingsprogrammatuur is in die zin vaak meer declaratief van aard.

In dit artikel zal onder 'programma' worden verstaan: De formele beschrijving van de oplossing van een probleem in een taal die het mo elijk maakt dat de beschrijving door de computer 'executeerbaar' is. In het vervolg van dit artikel is een programmeertaal niet per definitie een general-purpose taal; ook een taal die gebruikt wordt voor speciale toepassingen (bijvoorbeeld een database vraagtaal) wordt programmeertaal genoemd. Met 'programmeren' wordt bedoeld het planmatig ontwerpen van een oplossing voor open problemen en het vervolgens coderen van deze oplossing, zodat een programma ontstaat dat door de computer kan worden uitgevoerd. Met open problemen worden problemen bedoeld waarvoor verschillende correcte oplossingen mogelijk zijn.

De opbouw van dit artikel is als volgt. Allereerst worden redenen gegeven vóór het opnemen van programmeeronderwijs in het curriculum van het voortgezet onderwijs, en wordt uiteengezet waarom imperatieve, general-purpose talen minder geschikt zijn om in dat programmeeronderwijs gebruikt te worden. Daarna worden criteria ontwikkeld waaraan een voor het programmeeronderwijs geschikte taal zou moeten voldoen. Vervolgens wordt een korte schets van SQL gepresenteerd, waarna de geschiktheid van deze relationele database-vraagtaal voor het programmeeronderwijs wordt beargumenteerd. Hierbij worden ook de eerste ervaringen gerapporteerd van informatica-lessen met SQL. Tenslotte worden enkele manieren beschreven waarop de door leerlingen opgedane ervaring in het werken met databases ook op andere plaatsen in het onderwijs gebruikt kan worden.

2. Programmeeronderwijs: Argumenten en Ervaringen

Het vak informatiekunde in de onderbouw van het voortgezet onderwijs heeft tot doel een beknopte, algemene oriëntatie in de informatica te bieden (Ministerie van Onderwijs en Wetenschappen, 1985). Volgens het algemeen beleidskader van de minister (Ministerie van Onderwijs en Wetenschappen, 1986) zal in de middenbouw die kennis moeten worden uitgebreid. Daar zullen de

eerste stappen gezet moeten worden op de weg naar het zelf realiseren van oplossingen voor problemen. Daarmee komt in het middenbouw-programma systematisch probleemoplossen centraal te staan.

Binnen diverse vakken in het voortgezet onderwijs neemt het systematisch leren ontwerpen van oplossingen voor open problemen een belangrijke plaats in. Als voorbeelden van open problemen kunnen genoemd worden: het schrijven van een betogend opstel binnen het vak Nederlands, het ontwerpen van een proefje bij natuur- of scheikunde, of het leveren van een meetkundig bewijs bij wiskunde.

Voor het planmatig leren oplossen van informatica-problemen lijkt programmeeronderwijs bij uitstek geschikt. De leerlingen kunnen zelf, in interactie met de computer, onmiddellijk nagaan of de door hen ingevoerde oplossingen aan hun verwachtingen voldoen. Bij het oplossen van open problemen is dit een uitzonderlijke situatie. Snelle terugkoppeling wordt door leerlingen doorgaans sterk gewaardeerd: zowel de controle die zij op hun eigen werk kunnen uitoefenen, als de feedback die zij van de computer ontvangen, werken een hoge motivatie in de hand (Lepper, 1985).

Een tweede reden om een cursus programmeren te verzorgen binnen het voortgezet onderwijs is dat de computer tegenwoordig een dusdanig belangrijke rol speelt in de maatschappij, dat het voor de hand ligt dat leerlingen in het voortgezet onderwijs kennis maken met informatica-aspecten en inzicht krijgen in informatieverwerking met behulp van de computer. De lessen informatiekunde in de onderbouw zijn daarvoor te inleidend. Een cursus programmeren zou deze kennis en inzichten kunnen verdiepen.

Een derde argument om een programmeercursus aan te bieden in het voortgezet onderwijs is dat, gezien de toenemende inschakeling van de computer in het onderwijs en in de maatschappij, programmeren een vaardigheid is die door de leerlingen gebruikt kan worden in verdere studie of beroep.

Een vierde argument tenslotte betreft de vaak geopperde mogelijkheid van transfer naar andere vakgebieden of domeinen: als programmeren inderdaad een sterk beroep doet op het vermogen om systematisch problemen op te lossen, en als een vergelijkbare systematiek van toepassing is op andere vakgebieden dan de informatica, dan zou verwacht mogen worden dat de

vaardigheden die bij het leren programmeren worden opgedaan, ook elders kunnen worden toegepast (Nickerson, 1983; Papert, 1980).

Uit het voorgaande kan geconcludeerd worden dat er goede redenen zijn om een cursus programmeren deel te laten uitmaken van het vak informatica in de middenbouw van het voorgezet onderwijs. Benadrukt dient te worden dat daarbij niet uitsluitend gedacht moet worden aan programmeren in een imperatieve taal: speciaal wat imperatieve talen betreft zijn er tegenargumenten te noemen. In de toekomst zullen weinig niet-informatici zelf programmeren in een imperatieve general-purpose taal, vanwege de toename van het gebruik van programmatuur voor speciale toepassingen. Verder vergt het leren programmeren in een imperatieve taal een aanzienlijke hoeveelheid training (Shneiderman, 1985; Linn, 1985; Perkins & Martin, 1986; Pea & Kurland, 1984), te veel om haalbaar te zijn in de korte tijd die in het voortgezet onderwijs beschikbaar is (De Corte & Verschaffel, 1986).

Als de elementaire beginselen van het programmeren onvoldoende beheerst worden, worden probleemoplosvaardigheden als plannen en debuggen niet geoefend (Linn & Dalbey, 1985; Spohrer & Soloway, 1986; Putnam, Sleeman, Baxter & Kuspa, 1986), waardoor transfer naar andere vakgebieden of domeinen niet kan optreden (Goodyear, 1987; Salomon & Perkins, 1987; Kurland, Pea, Clement & Mawby, 1986).

3. Criteria voor de keuze van een taal

Op basis van de genoemde argumenten voor programmeeronderwijs, en rekening houdend met de ervaringen die zijn opgedaan met programmeren in een imperatieve taal, zijn criteria te formuleren aan de hand waarvan talen beoordeeld kunnen worden op hun geschiktheid voor programmeeronderwijs in het voortgezet onderwijs. De volgende algemene criteria zijn al genoemd: de te kiezen taal zal voldoende mogelijkheden moeten bieden tot het *planmatig ontwerpen van oplossingen* voor open problemen, in die taal zullen *executeerbare oplossingen* geformuleerd moeten kunnen worden en er zal *terugkoppeling* gegeven moeten worden op de ingevoerde oplossingen. Daarnaast kan een aantal specifieke criteria worden geformuleerd.

De *leertijd voor de benodigde taalelementen* dient *gering* te zijn in verhouding tot de tijd die besteed kan worden aan het planmatig leren ontwerpen van oplossingen voor open problemen.

Dit is te meer belangrijk omdat in de middenbouw van het voortgezet onderwijs weinig tijd beschikbaar zal zijn voor het vak informatica.

Het is moeilijk gebleken leerlingen een goed mentaal model te laten verwerven van de werking van de computer (DuBoulay, 1986). Na een inleidende programmeercursus in een imperatieve taal, schrijven leerlingen nog dikwijls 'menselijke' eigenschappen toe aan de computer, zoals de mogelijkheid om interpreterend te werk te gaan of gevolgtrekkingen te maken (Pea, 1986; Sleeman, Putnam, Baxter & Kuspa, 1986). De voorkeur gaat dan ook uit naar een taal waarvoor een relatief *eenvoudig mentaal model* van de werking van de computer volstaat.

Het planmatig oplossen van een open probleem *vereist analyse* van het probleem om tot een goed werkende oplossing te komen. Het moet vrijwel uitgesloten zijn dat een leerling door 'trial and error', 'hacking' tot een goed werkende oplossing komt. Dit impliceert tevens dat een commandogestuurde taal wordt geprefereerd boven een menugestuurde taal.

De te kiezen taal moet *niet op te specifieke problemen gericht* zijn. Met een taal die ontwikkeld is met het oog op specifieke inhoudelijke toepassingen (zoals bijvoorbeeld boekhoudkundige programma's) zullen voornamelijk probleemoplosvaardigheden geoefend worden die specifiek zijn voor het met de taal samenhangende probleemdomein. De mogelijkheid van transfer naar andere domeinen wordt daardoor beperkt. Met de te kiezen taal moeten algemene probleemoplosvaardigheden geoefend kunnen worden op basis van vraagstellingen die uit informaticaoogpunt relevant zijn.

Het laatste criterium heeft niet zozeer te maken met het oplossen van ontwerpproblemen, maar met het toenemend gebruik van computers, zowel in het onderwijs als daarbuiten. Indien mogelijk zal een taal gekozen moeten worden waarvan, naar verwachting, in de toekomst *veelvuldig gebruik* zal worden gemaakt.

Op grond van deze criteria lijkt de keuze voor de vraagtaal SQL gerechtvaardigd.

4. Korte schets van SQL

In deze paragraaf worden een aantal kenmerken van SQL beschreven. De bedoeling daarvan is niet om een volledig overzicht van deze vraagtaal te presenteren; daarvoor wordt ver-

wezen naar Van der Lans (1986), Eilers, Jansen & Volder (1986) of het manual van SQL/QIT (Queensland Information Technology Pty. Ltd., 1986). Taalkenmerken en voorbeelden beogen een globaal beeld te geven van de wijze waarop met SQL gewerkt wordt.

SQL is geen general-purpose taal maar een vraagtaal die is bedoeld om gegevens in een database te zetten en deze gegevens te raadplegen en te veranderen. SQL is gebaseerd op het zogeheten relationele datamodel, zoals ontwikkeld door Codd (1970). Kenmerkend voor een relationele database is, dat de gegevens worden gepresenteerd in de vorm van tabellen ofwel relaties. Alle tabellen hebben een naam, en elke kolom van elke tabel kan worden aangeduid met een kolomnaam. In figuur 1 staan twee (delen van) tabellen afgebeeld, die beide gegevens bevatten van leerlingen.

Leerling

llnr	naam	klas
1753	Bevers	HAVO 3B
1219	Smits	VWO 4A
2142	Jaspers	HAVO 5C
.	.	.
.	.	.

Rapportcijfers

llnr	vak	rapport	cijfer
1219	wiskunde	3	4
2142	wiskunde	1	7
1219	engels	4	6
.	.	.	.
.	.	.	.

De tabel met naam LEERLING heeft kolommen met de namen LLNR, NAAM en KLAS. Elke rij in deze tabel bevat het leerlingnummer, de naam en de klas van een leerling. Op dezelfde manier bestaat de tabel RAPPORTCIJFERS uit de kolommen LLNR, VAK, RAPPORT en CIJFER; elke rij van deze tabel bevat een rapportcijfer (in de kolom CIJFER), een leerlingnummer, het vak waarvoor het cijfer behaald is en, in kolom RAPPORT, het nummer van het rapport waarop het cijfer staat. Op de school worden jaarlijks vier rapporten uitgereikt, zodat in kolom RAPPORT de waarden 1, 2, 3 en 4 kunnen voorkomen.

Met behulp van de vraagtaal SQL kunnen uit deze tabellen gegevens worden opgevraagd. SQL maakt daarbij gebruik van de sleutelwoorden SELECT, FROM en WHERE die in bijna elk SQL-commando voorkomen. Hierna volgen enkele voorbeelden.

De namen van de leerlingen uit klas HAVO 3B kunnen als volgt worden opgevraagd. Door de laatste regel van dit commando wordt de lijst met namen alfabetisch geordend.

```
SELECT NAAM
FROM LEERLING
WHERE KLAS = 'HAVO 3B'
ORDER BY NAAM
```

In SQL zijn standaard aanwezig de functies COUNT, MIN, MAX, SUM en AVG. Het gemiddelde van alle rapportcijfers van de leerling met nummer 1219 kan bijvoorbeeld worden opgevraagd door middel van het commando:

```
SELECT AVG (CIJFER)
FROM RAPPORTCIJFERS
WHERE LLNR = '1219'
```

Om niet het gemiddelde over alle cijfers van het hele jaar te laten berekenen, maar de gemiddelden voor elk rapport afzonderlijk, moeten de rapportcijfers van een bepaalde leerling worden verdeeld in groepen met een gelijke waarde in de kolom RAPPORT, waarna per groep het gemiddelde cijfer moet worden berekend. In SQL wordt daarvoor GROUP BY gebruikt:

```
SELECT RAPPORT, AVG (CIJFER)
FROM RAPPORTCIJFERS
WHERE LLNR = '1219'
GROUP BY RAPPORT
```

Tabellen moeten soms met elkaar worden gecombineerd. Om bijvoorbeeld de rapportcijfers van leerling Smits voor het vak wiskunde op te vragen, is een commando nodig waarin beide tabellen uit figuur 1 worden bevraagd. In dit commando wordt tevens gebruik gemaakt van de logische operator AND.
De op te vragen gegevens, rapportnummer en cijfer, stonden in dit voorbeeld in dezelfde tabel (RAPPORTCIJFERS). Het raadplegen van de tabel LEERLING was alleen nodig om het leerlingnummer van Smits op te zoeken. Als de op te vragen gegevens

```
SELECT RAPPORT, CIJFER
FROM RAPPORTCIJFERS
WHERE LLNR = (SELECT LLNR
              FROM LEERLING
              WHERE NAAM = 'Smits')
      AND
      VAK = 'wiskunde'
```

niet in dezelfde tabel staan, is het nodig om tabellen samen te voegen ('joinen').

In onderstaand voorbeeld wordt van elke leerling de naam opgevraagd met de vakken uit tabel RAPPORTCIJFERS die betrekking hebben op die leerling.

```
SELECT NAAM, VAK
FROM LEERLING, RAPPORTCIJFERS
WHERE LEERLING.LLNR = RAPPORTCIJFERS.LLNR
```

In dit voorbeeld staat in de voorwaarde dat de waarde in kolom LLNR van tabel LEERLING (LEERLING.LLNR) dezelfde moet zijn als de waarde in kolom LLNR van tabel RAPPORTCIJFERS (RAPPORTCIJFERS.LLNR). Deze voorwaarde zorgt ervoor dat de naam van een leerling alleen gekoppeld wordt aan de vakken die deze leerling zelf volgt, en niet aan alle waarden die in kolom VAK voorkomen.

Met de hier genoemde taalelementen van SQL kan een belangrijk deel van de aan een database te stellen vragen worden geformuleerd.

5. SQL als taal voor programmeeronderwijs

In deze paragraaf zal eerst de keuze voor de relationele database vraagtaal SQL worden toegelicht. Daarna wordt nagegaan of met deze keuze voldoende tegemoet wordt gekomen aan de criteria die in paragraaf 3 zijn geformuleerd.

Zoals eerder vermeld zal in toenemende mate gebruik worden gemaakt van special-purpose languages, waardoor voornamelijk informatici nog zelf zullen programmeren in een general-purpose language. Om die reden is gekozen voor een special-purpose language, en wel een database vraagtaal, met als belangrijk argument dat het werken met grote gegevensbestanden in toenemende mate van belang is.

Uitgaande van het werken met databases, is de keuze voor SQL nog niet vanzelfsprekend. Er zijn immers diverse soorten databases, die onderling sterk verschillen. Met name zijn er uiteenlopende manieren waarop de gegevens worden gepresenteerd aan degene die informatie opvraagt (datamodellen).

Date (1981) bespreekt drie datamodellen: het relationele model, waarin de gegevens worden gepresenteerd in de vorm van tabellen, het hiërarchische model, waaraan een boomstructuur ten grondslag ligt en het netwerkmodel, gebaseerd op algemene netwerkstructuren.

Onderzoek naar verschillen in gebruiksgemak (ease of use) tussen deze datamodellen leverde als belangrijke uitkomst op, dat op het gebied van computers onervaren proefpersonen betere resultaten behaalden met databases gebaseerd op het relationele model dan met de databases gebaseerd op de beide andere modellen (Lochovsky & Tsichritzis, 1977). Dit is wellicht te verklaren uit het feit dat bij het werken met een hiërarchische database of een netwerkdatabase de gebruiker zelf de bestanden op het achtergrondgeheugen moet localiseren en door de database 'navigeren': de gebruiker van de database moet aangeven hoe de gegevens benaderd moeten worden. Bij een relationele database wordt dit automatisch door het systeem gedaan.

Op dit moment worden voornamelijk relationele databases gebruikt (Date, 1986), en hierin zal in de nabije toekomst geen verandering komen. Van de vraagtalen die gebaseerd zijn op het relationele model, zijn dBase en SQL de bekendste. Wat betreft de soorten problemen die kunnen worden opgelost zijn deze talen zeer vergelijkbaar. Wel is er verschil in de hoeveelheid benodigde taalelementen, maar belangrijker is het verschil in de mate waarin kennis nodig is van implementatie-aspecten. SQL doet nergens een beroep op inzicht in de organisatie van het achtergrondgeheugen: De wijze waarop de bestanden op het achtergrondgeheugen worden gemanipuleerd blijft volledig verborgen. SQL kent minder taalelementen dan dBase; bovendien is bij het werken met dBase wel enige kennis nodig van de manipulatie van bestanden. Een introductie in het gebruik van dBase zou dan ook meer tijd vergen dan een introductie in het gebruik van SQL. Overigens biedt de nieuwe versie van dBase (dBase IV) de mogelijkheid om te werken met de vraagtaal SQL.

Daarmee is aangegeven hoe de keuze voor SQL tot stand is gekomen. De vraag is nu, of met deze keuze voldoende tegemoet

gekomen wordt aan de criteria die in paragraaf 3 zijn geformuleerd. Allereerst kan worden opgemerkt dat het aantal syntactische elementen van SQL zeer gering is, waardoor deze in vrij korte tijd te leren zijn (Reisner, 1981). Verder zijn SQL-commando's in het algemeen veel korter dan programma's in een imperatieve taal (getuige ook de voorbeelden in paragraaf 4), terwijl de computer toch duidelijk veel 'werk verricht'. Dit heeft te maken met het feit dat in SQL alleen hoeft te worden geformuleerd *wat* de karakteristieken zijn van de gegevens die opgevraagd worden; *hoe* deze gegevens moeten worden opgezocht hoeft niet te worden gespecificeerd. Beginnende programmeurs in een imperatieve taal vinden het vaak verrassend dat de taken die de computer moet uitvoeren zo gedetailleerd moeten worden geprogrammeerd (DuBoulay, 1986). Met dit toch enigszins ontmoedigende verschijnsel krijgt men bij het werken met SQL niet te maken. Het voorgaande impliceert dat het merendeel van de tijd zal kunnen worden besteed aan het planmatig ontwerpen van oplossingen voor problemen.

Het mentale model van de werking van de computer dat vereist is om met SQL te kunnen werken, is relatief eenvoudig. De gegevens in de database worden steeds gepresenteerd in de vorm van tabellen, een manier om informatie vorm te geven die ook voor niet-professionele gebruikers van databases heel overzichtelijk is. Aangezien de wijze van opslaan en manipuleren van gegevens tijdens het werken met SQL geheel verborgen blijft, is inzicht in de manier waarop de computer deze handelingen verricht niet nodig. Daarom hoeft bij het werken met SQL het mentale model van de werking van de computer zich niet uit te strekken tot de wijz van opslag van de gegevens op het achtergrondgeheugen. Dit betekent een aanzienlijke vereenvoudiging, vergeleken met het mentale model dat nodig is om te kunnen programmeren in een imperatieve taal. De aandacht kan daardoor geconcentreerd worden op de inhoud van de tabellen en de relaties tussen de tabellen.

Ondanks de eenvoud van de taal, is voor het oplossen van realistische problemen met behulp van SQL een planmatige aanpak vereist (Remmen, 1985). Om een idee te geven hoe concrete problemen op zo'n planmatige manier kunnen worden opgelost, is in de volgende paragraaf een uitgewerkt voorbeeld opgenomen. In het daar gevolgde oplossingstraject kan een drietal aspecten onderscheiden worden:

- Ten eerste worden informatievragen doorgaans in een natuurlijke taal (Nederlands) gesteld, waardoor de terminologie niet direct aansluit bij de manier waarop de gegevens in de database beschreven staan (Landauer, Dumais, Gomez & Furnas, 1982). Daardoor is systematische analyse van het probleem en nadere specificatie van de informatievraag noodzakelijk. Leerlingen moeten de informatievraag herformuleren zodanig dat zowel een nauwkeurige afbakening van e vraag, als een aanzet tot oplossing in termen van benodigde gegevens (en de daaraan te stellen voorwaarden) het resultaat is.
- Ten tweede zal in realistische toepassingen in het algemeen gebruik gemaakt worden van een database die bestaat uit meerdere onderling gerelateerde tabellen. Daarmee komt een volgend aspect van de complexiteit van het werken met databases aan de orde: een leerling moet namelijk beslissen welke tabellen gebruikt moeten worden om de gewenste informatie te verkrijgen. Dat vergt kennis van de inhoud van tabellen, maar ook begrip van de onderlinge relaties tussen de tabellen.
- Tenslotte moet het ontwerp van de oplossing van de informatievraag omgezet worden in een formele taal, zodat het antwoord met behulp van de computer verkregen kan worden.

Relatief eenvoudig lijkende problemen kunnen al zo complex zijn dat zonder een planmatige aanpak oplossingen vaak fout zullen zijn (Reitsma, 1985). Het gebruik van algemene probleemoplosstrategieën, zoals het analyseren en specificeren van een probleem en het opsplitsen in deelproblemen, zal dan ook vrijwel vanaf het begin geoefend moeten worden. Transfer naar algemene probleemoplos-vaardigheden kan daarvan het gevolg zijn. Daarnaast is het niet uitgesloten dat transfer zal optreden naar een aantal specifieke onderwerpen binnen de wiskunde. Inzicht in de verzamelingenleer en logica is voor het werken met SQL hoogst wenselijk (Remmen, 1985).
Het omgekeerde kan echter ook verondersteld worden: het kunnen werken met databases zou een positieve invloed kunnen hebben op inzicht in verzamelingenleer en logica.

Verder zal mogelijk transfer optreden naar het vermogen om gegevens te identificeren, organiseren en evalueren. White (1987) vond dat scholieren die gewerkt hadden met een geautomatiseerde database, vergeleken met scholieren in een controlegroep die gewerkt hadden met kaartenbakken, beter in staat waren om

informatie te selecteren ten behoeve van een bepaald probleem: ze konden zowel beter de relevantie van gegevens bepalen, als aangeven welke gegevens voldoende waren om een bepaald probleem op te lossen. Bovendien waren ze beter in staat om gegevens te sorteren op een manier die het oplossen van een gegeven probleem vereenvoudigde.

Tenslotte maken databases het verwerken van grote hoeveelheden gegevens mogelijk. Het kunnen omgaan met grote gegevensbestanden wordt in onze maatschappij steeds belangrijker (Nijssen, 1986), waardoor verwacht mag worden dat SQL in de toekomst veelvuldig gebruikt zal worden, temeer omdat de vraagtaal SQL voor professioneel gebruik de standaard lijkt te worden.

Hiermee zijn alle criteria uit paragraaf drie aan de orde geweest en kan op grond van theoretische overwegingen gesteld worden dat SQL geschikt is als taal voor het programmeeronderwijs. Om na te gaan of deze vraagtaal ook voldoet in de onderwijspraktijk is lesmateriaal ontwikkeld op basis van SQL (Van Dijk, 1988a).

In de periode februari-juni 1988 is met dit lesmateriaal geëxperimenteerd in twee 4VWO-klassen, die er respectievelijk twaalf en tien lesuren mee gewerkt hebben. Bij de rapportage over dit experiment, dat de status van vooronderzoek had, is gebruik gemaakt van zowel de resultaten van een eindtoets, als van scores op een evaluatieformulier (Van Dijk, 1988b).

Uit de resultaten van de eindtoets bleek dat leerlingen het selecteren van kenmerken, de keuze van de juiste tabel en het stellen van voorwaarden (ook samengestelde voorwaarden met AND en OR) goed beheersten. Moeilijke onderwerpen bleken te zijn: groeperen (GROUP BY), het gebruik van de functies SUM en COUNT (deze werden vaak verwisseld) en het schrijven van een goed commando wanneer de formulering van een opgave erg afweek van het uiteindelijke SQL-commando. Opmerkelijk was de afwezigheid van een verband tussen de gemiddelde prestatie op de eindtoets en ervaring in het werken met computers.

Een belangrijke bevinding van de evaluatie was, dat de leerlingen de lessen in het algemeen als niet bijzonder afwisselend beschouwden. Een oorzaak is wellicht dat in de lessen slechts één werkvorm werd gebruikt; verder kregen de leerlingen alleen les in het bevragen van databases, waarbij voornamelijk gebruik gemaakt werd van één database. Deze mogelijke oor-

zaken geven aanwijzingen voor verbetering van de vormgeving van de lessen.

Hoewel geen systematische observatie heeft plaatsgevonden, werd tijdens de lessen duidelijk dat de leerlingen, werkend in groepjes van drie achter de computer, niet vaak om hulp vroegen van de aanwezige docent, terwijl ze uiteindelijk toch meestal tot een goede oplossing van de gepresenteerde problemen kwamen. De hulp die ze vroegen betrof meestal syntactische fouten. Inderdaad zijn de foutmeldingen van het gebruikte SQL-pakket (SQL/QIT) niet altijd duidelijk. Voor zover bekend geldt hetzelfde voor andere SQL-pakketten. Daarnaast is de bij het pakket behorende editor niet ideaal. Foutmeldingen en editor leverden tijdens de experimenten overigens geen noemenswaardige problemen op.

6. Toepassingen van SQL in niet-informatica onderwijs

Werken met relationele databases in informatica-lessen levert ervaring op die desgewenst elders in de school kan worden gebruikt. Daarbij zou bijvoorbeeld gedacht kunnen worden aan vakken als aardrijkskunde, biologie, scheikunde, economie, geschiedenis en maatschappijleer. In genoemde vakken wordt al geëxperimenteerd met het gebruik van databases in de les. Naast vakinhoudelijke zijn ook andere toepassingen van databases op school denkbaar. Bijvoorbeeld zou voor het begeleiden van de keuzen van pakket en vervolgopleiding, een deel van het werk van schooldecanen vereenvoudigd kunnen worden door het gebruik van een database.

In de les kan op verschillende manieren gebruik gemaakt worden van databases:

- Als gebruik gemaakt wordt van een vooraf gecreëerde database kunnen allerlei delen van de database geselecteerd worden, maar ook kan op allerlei manieren worden gesorteerd en gegroepeerd. Met numerieke gegevens uit de database kan worden gerekend.
  Op deze manier kan dan een beter overzicht verkregen worden van de gegevens in de database (Rawitsch, 1987-88). Met name in combinatie met een klassegesprek lijkt dit een nuttige toepassing van het gebruik van databases binnen een aantal vakken.
- De gegevens in een database kunnen ook worden geanalyseerd op aanwezige onderlinge verbanden; mogelijke oorzaken van

gesignaleerde verbanden kunnen geverifieerd of gefalsifieerd worden. De leerlingen kunnen op deze manier zelf hypothesen opstellen en controleren (Blankenbaker, 1987). Ook dit kan dan gevolgd worden door een klassegesprek, waarin de bevindingen van de verschillende (groepen van) leerlingen worden gerapporteerd, vergeleken en besproken.

- Binnen project-onderwijs is de computer goed bruikbaar om verzamelde gegevens op te slaan. Gegeven een bepaalde vraagstelling moeten de leerlingen eerst beslissen welke gegevens moeten worden verzameld, en op welke manier de verzamelde gegevens moeten worden vastgelegd. In de gegevens moet structuur worden aangebracht: ze moeten worden gecategoriseerd en geordend. Als tenslotte de gestructureerde gegevens zijn ingevoerd in de computer, is het resultaat een database waaraan informatie onttrokken kan worden op de manier die hiervoor is gepresenteerd (Schoenmaker, 1985). Door aldus de mogelijkheden van databases te benutten, hoeft het verzamelen van gegevens niet de afsluiting te zijn van het project, maar kunnen de verzamelde gegevens gebruikt worden om informatie te verkrijgen die relevant is voor de projectvraagstelling (Strickland, 1987).

7. Samenvatting

Hiervoor zijn verschillende redenen genoemd, waarom programmeeronderwijs nuttig geacht kan worden: bij het leren oplossen van open problemen werkt snelle terugkoppeling door de computer in hoge mate motiverend; door de toenemende automatisering wordt het vertrouwd zijn met computers steeds belangrijker; programmeren zou een vaardigheid zijn die door leerlingen gebruikt kan worden in verdere studie of beroep; er zou transfer naar andere vakgebieden optreden. Vervolgens werden belangrijke bezwaren geformuleerd tegen het leren programmeren in een imperatieve taal.

Tenslotte werd beargumenteerd waarom deze bezwaren niet gelden voor programmeeronderwijs waarin gewerkt wordt met relationele databases met als vraagtaal SQL. Een belangrijke toepassing van de computer, beperkt genoeg om in korte tijd te leren, althans wat hoeveelheid benodigde taalelementen betreft, en geschikt voor het planmatig leren oplossen van open problemen die uit informatica-oogpunt relevant zijn.

Nader onderzoek zal moeten uitwijzen of programmeeronder-

wijs met SQL ook werkelijk de gunstige resultaten oplevert die op grond van theoretische overwegingen verwacht mogen worden.

Noot

1. De auteurs danken H. van Berne, A.Duijvestijn, S.Dijkstra, H.Koppelman en G.Vissers voor hun waardevolle commentaar.

Literatuur

Blankenbaker, R. (1987) Databases in the English class: A valuable lesson, *The Computing Teacher, 15*(2), 17-18.

Codd, E.F. (1970) A relational model of data of large shared data banks, *Communications of the ACM, 13*, 377-387.

Corte, E. de & L. Verschaffel (1986) Effects of computer experience on children's thinking skills, *Journal of structural Learning, 9*, 161-174.

Date, C.J. (1981) *An introduction to database systems, Vol I*, 3rd edition. Reading, Mass.: Addison-Wesley publishing company.

Date, C.J. (1986) *An introduction to database systems, Vol I*, Fourth edition. Reading, Mass.: Addison-Wesley publishing company.

DuBoulay, B. (1986) Some difficulties of learning to program, *Journal of Educational Computing Research, 2*, 57-73.

Dijk, E.M.A.G. van (1988a) *Databases en de vraagtaal SQL*: Experimentele Leerboeken 1 en 2, Enschede: Universiteit Twente, Faculteit Informatica.

Dijk, E.M.A.G. van (1988b) *Probleemoplossen met SQL*: Verslag van een vooronderzoek, Memorandum INF-88-58, Enschede: Universiteit Twente, Faculteit Informatica.

Eilers. H.B., W.F. Jansen & H.H.J. de Volder (1986) *SQL in de praktijk*, Den Haag: Academic Service.

Goodyear, P. (1987) Sources of difficulty in assessing the cognitive effects of learning to program, *Journal of Computer Assisted Learning, 3*, 214-223.

Koers, H. (1985) *Doelstellingen keuzevak informatica, 2e fase VO*: Eindrapport, Enschede: Universiteit Twente, Faculteit der Toegepaste Onderwijskunde.

Krammer, H.P.M. & H. Koers (1986) Doelstellingen voor het keuzevak informatica: Een factoranalyse, *Informatie, 28*, 457-463.

Kurland, D.M., R.D. Pea, C. Clement & R. Mawby (1986) A study of the development of programming ability and thinking skills in high school students, *Journal of Educational Computing Research, 2*, 429-458.

Landauer, T.K., S.T. Dumais, L.M. Gomez & G.W. Furnas (1982) Human factors in data access, *The Bell System Technical Journal, 61*, 2486-2509.

Lans, R.F. van der (1986) *Het SQL Leerboek: De standaard relationele databasetaal*, Den Haag: Academic Service.

Lepper, M. (1985) Microcomputers in education: Motivational and social issues, *American Psychologist, 40*, 1-18.

Linn, M. C. (1985) The cognitive consequences of programming instruction in classrooms, *Educational Researcher, 14* (5), 14-29.

Linn, M. C. & J. Dalbey (1985) Cognitive consequences of programming instruction: Instruction, Access, and Ability, *Educational Psychologist, 20*, 191-206.

Ministerie van Onderwijs en Wetenschappen (1985) *Beleidskader voor activiteiten 1985/1986. Informatietechnologie in de eerste fase voortgezet onderwijs (Cluster II.2)*, PSOI-reeks nr. 6.

Ministerie van Onderwijs en Wetenschappen (1986) *Aanvullend beleidskader Informatica bovenbouw HAVO/VWO*, PSOI-reeks nr. 12.

Nickerson, R.S. (1983) Computer programming as a vehicle for teaching thinking skills, *Thinking. The Journal of Philosophy for Children, 4*, 42-48.

Nijssen, G.M. (1986) Nijssen gelooft nog alleen in IBM's SQL. *Computable*, 19 december 1986, 23.

Papert, S. (1980) *Mindstorms: Children, computers and powerful ideas*, New York: Basic Books.

Pea, R.D. (1986) Language-independent conceptual 'bugs' in novice programming, *Journal of Educational Computing Research, 2*, 25-36.

Pea, R.D. & M. Kurland (1984) On the cognitive effects of learning computer programming, *New Ideas in Psychology, 2*, 131-168.

Perkins, D. & F. Martin (1986) Fragile knowledge and neglected strategies in novice programmers. In E. Soloway & S. Iyengar (Eds.), *Empirical Studies of Programmers*, Norwood, NJ: Ablex.

Putnam, R.T., D. Sleeman, J.A. Baxter & L.K. Kuspa (1986) A summary of misconceptions of high school BASIC programmers, *Journal of Educational Computing Research, 2*, 459-471.

Queensland Information Technology Pty. Ltd. (1986) *SQL User's Guide.*

Rawitsch, D. (1987-88) The computerized database: Not a simple solution, *The Computing Teacher, 15* (4), 34-37.

Reisner, P. (1981) Human factors studies of database query languages: A survey and assessment, *Computing Surveys, 13*, 13-31.

Reitsma, E.H. (1985) Grenzen en beperkingen van SQL, *Informatie, 27*, 198-201.

Remmen, F. (1985) Hoe vriendelijk zijn vraagtalen in het gebruik? *Informatie, 27*, 666-673.

Salomon, G. & D.N. Perkins (1987) Transfer of cognitive skills from programming: when and how?, *Journal of Educational Computing Research, 3*, 149-169.

Schoenmaker, J. (1985) Gegevensbanken in het onderwijs, In: E.J.W.M. van Hees, G.A. Vonk & J.M. Schoenmaker (Eds.), *Handboek onderwijs en computer*, Alphen aan den Rijn: Samson.

Shneiderman, B. (1985) Overcoming limitations imposed by current programming languages, In: R. Jernigan, B.W. Hamill & D.M. Weintraub (Eds.), *The Role of Language in Problem Solving 1*, North-Holland: Elseviers Science Publishers.

Sleeman, D., R.T. Putnam, J.A. Baxter & L.K. Kuspa (1986) Pascal and high school students: A study of errors, *Journal of Educational Computing Research, 2*, 5-24.

Spohrer, J. & E. Soloway (1986) Novice mistakes: Are the folk wisdoms correct?, *Communications of the ACM, 29*, 624-632.

Strickland, A.W. (1987) AppleWorks afield. *The Computing Teacher, 15* (2), 17-18.

Van Dale (1984) *Groot woordenboek der Nederlandse taal*, Utrecht/Antwerpen: Van Dale Lexicografie bv.White, C.S. (1987) Developing information-processing skills through structured activities with a computerized file-management program, *Journal of Educational Computing Research, 3*, 355-375.