ABIS NIEUWSBRIEF VOOR DB2 PROFESSIONALS

JAARGANG 2 - NUMMER 8, MEI 2004

OPEN CURSOR

DB2 voor z/OS staat niet op een eiland!
Steeds vaker wordt het gebruikt in totaal nieuwe en tot voor kort onverwachte omgevingen - Java, XML, .Net. Ook de interesse voor integratie van DB2 voor z/OS data met andere databronnen neemt steeds toe.
In vorige nummers zijn een aantal van deze topics reeds uitgebreid aan bod gekomen. In dit nummer staan we nog eens even stil bij een aantal specifieke performance karakteristieken van UDB federated servers. En in een volgend nummer zullen we de belangrijkste verschillen tussen alternatieve RDBMS-en op een rijtje te zetten.
Want DB2 voor z/OS staat niet op een eiland.

Veel leesgenot!

Het ABIS DB2-team.

IN DIT NUMMER:

- Over performance in UDB federated servers, in *UDB federated databases - performance.*
- Fundamentele wijzigingen op het gebied indexen in V8 - *Dossier 8: indexen op variabele lengte kolommen.*
- En ook aandacht voor cursors: *Scrollable cursors in DB2 voor z/OS.*
- *Cursusplanning mei 2004 - juni 2004.*

CLOSE CURSOR

In een volgend nummer - alweer het laatste van deze jaargang - hebben we het onder andere over de concrete performance voordelen van het werken met nieuwe SQL-constructies in DB2 versie 7.

Tot dan!

UDB federated databases - performance

Eric Everaert (ABIS)

Eigenlijk is de bedoeling van een UDB federated database zeer eenvoudig uiteen te zetten. Een federated database biedt gebruikers de mogelijkheid aan de hand van eenvoudige SQL-statements, volledig transparant, data te consulteren opgeslagen in een veelheid aan bronnen. Zowel relationele als niet-relationele bronnen kunnen worden geconsulteerd. Dit alles klinkt eenvoudig, maar is het allerminst. De taak van de UDB federated database is immers niet beperkt tot het voorzien van communicatie tussen deze data bronnen. De taak is veel complexer dan dat. De vraag die moet worden beantwoord is namelijk de volgende: 'Hoe bepaal je in een disparate omgeving (DB2, MSSQL, Oracle, Informix, ...) een optimaal toegangspad naar de gewenste data?' In dit artikel willen we dan ook stilstaan bij dit aspect - performance.

Een korte herhaling van cruciale UDB federated server concepten - voor details, zie 'Exploring DB2', Jaargang 1, Nummer 4, December 2002.

WRAPPER, een logische naam gegeven aan een externe data bron - één WRAPPER per databron-type is vereist. 'Ora8' kan de naam van een WRAPPER zijn gebruikt voor alle Oracle 8i externe databronnen;

SERVER, een concrete implementatie van een WRAPPER - toegang tot 3 databronnen van Oracle 8i vergt de definitie van 3 SERVER-objecten (van hetzelfde type WRAPPER);

NICKNAME, een lokale, logische naam opgeslagen in de cataloog van de UDB federated server, die verwijst naar een specifiek benoemd object (e.g. tabel, view) uit de door een SERVER benoemde externe databron - het te bewerken object.

Hoe wordt dit optimaal toegangspad bepaald? De UDB federated optimizer houdt voornamelijk rekening met informatie opgeslagen in de UDB federated databasecataloog, en met informatie opgeslagen in de WRAPPER. Op basis van deze informatie wordt het SQL-statement opgesplitst in fragmenten, die door de originele SERVER-databron, dan wel door de UDB federated server zullen worden uitgevoerd. Het optimaal toegangspad beschrijft deze fragmenten, alsook waar ze zullen worden uitgevoerd.

- Fragmenten die aanleiding geven tot typisch resource intensieve operaties - denk hierbij aan sorteeropdrachten, het berekenen van aggregaten, etc. - worden vaak door de UDB federated database uitgevoerd. De taak van de SERVER-databron is dan meestal beperkt tot leverancier van data (dus rijen);
- De optimizer kan ook bepalen dat een aantal fragmenten effectiever en efficiënter door de SERVER-databron kunnen worden uitgevoerd - ze zullen dus door deze laatste worden uitgevoerd. Het

resultaat van deze bewerking wordt daarna teruggezonden naar de UDB federated databaseserver voor verdere verwerking.

Concreet zijn 2 optimisatiestappen belangrijk. De PUSHDOWN analyse gaat na of, en welke, query-fragmenten door de originele SERVER-databronnen dan wel door de UDB federated server kunnen worden uitgevoerd. De GLOBAL OPTIMISATION analyse gaat tenslotte fragmenten toewijzen aan SERVERs, op basis van een analyse van wat voor de globale query het meest efficiënt is; analyse en coördinatie op het niveau van de originele query is hier dus relevant.

Met welke gegevens houdt de optimizer concreet rekening? Het aantal te verwerken rijen, de performantie karakteristieken van de SERVER-databron, het aantal verwachte rijen opgeleverd per query-fragment, en de beschikbare (netwerk) bandbreedte tussen de verschillende betrokken databases. In wat volgt bekijken we de relevante optimisatiestappen in detail, en kijken we na of en hoe het mogelijk is de optimizer op bepaalde momenten te beïnvloeden.

PUSHDOWN-analyse

PUSHDOWN analyse, enkel beschikbaar voor relationele databases, vervult een tweetal taken. In eerste instantie wordt nagegaan welke onderdelen van een query - query fragmenten - door de originele SERVER-databron kunnen worden uitgevoerd. Daarnaast wordt de originele query herschreven, zodat het op een meer efficiënte wijze door de UDB federated server dan wel door de SERVER-databron kan worden uitgevoerd. Om te bepalen of de originele SERVER-databron het fragment kan uitvoeren, wordt met volgende factoren rekening gehouden: de eigenschappen van deze SERVER, de karakteristieken van de NICKNAME waarop de query is gebaseerd, en natuurlijk ook met de originele query. Het spreekt voor zich dat als een bepaalde SQL-functie niet door de SERVER-databron kan worden uitgevoerd, de UDB federated server hiervoor zal moeten instaan. Afhankelijk van de functie kan dit aanleiding geven tot het transfereren van grote hoeveelheden data (misschien de volledige tabel?) van de SERVER-databron naar de UDB federated server. De impact op het netwerk en op de UDB federated server is dan ook niet verwaarloosbaar.

PUSHDOWN: SERVER-karakteristieken

De twee belangrijkste verschillen tussen (relationele) dataservers zijn typisch: het ondersteunde SQL-dialect enerzijds, en de sorteervolgorde (COLLATING SEQUENCE) anderzijds.

Wat betreft het SQL-dialect, is het volgende zeer belangrijk. Elk SQL-fragment moet worden uitgevoerd door de meest relevante SERVER-databron. Concreet betekent dit dat deze fragmenten SQL-dialecten moeten kunnen bevatten die specifiek zijn voor de SERVER-databron. Hiertoe is het daadwerkelijk gebruik van deze dialecten bij het samenstellen van de initiële query toegelaten - de taak van de UDB federated database is dan deze constructies 'ongemoeid' te laten. Het gebruik van deze SQL-dialecten kan echter ook een doelbewuste keuze zijn van de UDB federated database; een SQL-instructie wordt

dan door deze laatste transparant vertaald naar het SQL-dialect dat door de SERVER-databron gekend is. Dit gebeurt na een gedegen analyse, die van de UDB federated server extra resources vergt. Mislukt deze vertaalslag, dan geeft ook dit meestal aanleiding tot extra verwerking op het niveau van UDB federated server ('compensatie'-verwerkingen). Ook de specifieke behandeling door de SERVER-databron van null-waarden kan voor bijkomende problemen zorgen.

De reeds boven aangehaalde sorteerproblematiek heeft meestal een grote impact op performance. Indien de sorteervolgorde op de UDB federated server en op de SERVER-databron dezelfde is, evalueert de optimizer op basis van de efficiëntie van beide betrokken systemen, waar de sorteeropdracht moet worden uitgevoerd. Merk op dat de oorzaak van deze sorteeropdracht verschillend van aard kan zijn: eliminatie van dubbels, 'order by' of 'group by' instructies, etc. Echter, ook 'range based' operaties, of eenvoudige vergelijkingen geven vaak aanleiding tot sorteerwerk. Indien uitgevoerd door de UDB federated server kan dit opnieuw de transfer van extra data met zich meebrengen, met opnieuw nadelige gevolgen voor netwerk en server!

De configuratieparameter COLLATING_SEQUENCE (Y/N) geeft aan of de sorteervolgorde van de UDB federated server en de SERVER-databron identiek is. Merk op dat de sorteervolgorde geen belang heeft als alle data in hoofd- dan wel kleine letters is opgenomen in de database, en als deze data enkel de getallen 0 t/m 9 bevat. Is dit niet het geval, hou dan met volgende elementen rekening:

- de impact van 'national language support' op de sorteervolgorde;
- de problematiek van hoofd- en kleine letters; bepaalde databronnen (e.g. SQL Server) zijn misschien configureerbaar wat dit betreft;
- de mogelijkheid geboden aan systeembeheerders van bepaalde SERVER-databronnen om zelf een sorteervolgorde te definiëren (e.g. Oracle);
- de impact van NULL-en.

Merk op dat het loutere aspect van 'case sensitivity' een grotere impact heeft dan enkel sorteren: afhankelijk van de case-sensitiviteit van de SERVER-databron wordt de where-conditie x1 = 'a' immers verschillend geëvalueerd. DB2 en Oracle hechten aan dit verschil aanzienlijk belang; bij SQL Server is dit configureerbaar.

Voorbeeld: Impact COLLATING_SEQUENCE

```
Veronderstel volgend scenario:
   UDB federated server (steeds case sensitive)
   SERVER data bron: MSSQL (default niet case sensitive)

SQL statement uitgevoerd op de UDB federated server:
   select * from persons where plname = 'MAK';

alter server mssql options (collating_sequence 'I'): pushdown
alter server mssql options (collating_sequence 'Y'): geen pushdown
```

PUSHDOWN: NICKNAME-karakteristieken

Op het niveau van de gebruikte NICKNAME worden de datatypes van de kolommen in de SERVER-brontabel door de UDB federated server vertaald in een DB2 UDB datatype. De UDB federated server kiest hierbij voor de meeste realistische en efficiënte vertaalslag. Het is echter steeds nodig na te gaan of bij join-operaties het datatype en de lengte van de join-kolommen op het niveau van de UDB federated server identiek is. Is dit niet het geval, dan geeft dit aanleiding tot een join-operatie op het niveau van de UDB federated server, terwijl deze eigenlijk door de SERVER-databron moet worden uitgevoerd. Met opnieuw als negatieve gevolgen: belasting van het netwerk, en extra processing op het niveau van de UDB federated server. U kan het resultaat van deze vertaalslag wijzigen middels het ALTER NICKNAME statement.

Op het niveau van een NICKNAME beïnvloeden een tweetal opties het gedrag van de UDB federated server.

- VARCHAR_NO_TRAILING_BLANKS (Y/N) geeft aan of de SERVER-databron op dezelfde manier omspringt met extra blanco's in variabele lengte kolommen als de UDB federated server. Op basis van deze optie kiest deze laatste een correcte semantiek voor het evalueren van data opgeslagen in kolommen met verschillende alphanumerieke datatypes, en lengtes.
- NUMERIC_STRING (Y/N) geeft aan dat een alphanumerieke kolom alleen maar numerieke waarden bevat, zonder blanco's. Hierdoor kan, onafhankelijk van de sorteerproblematiek als boven beschreven, de sorteer operatie alsnog door de SERVER-databron worden uitgevoerd. De karakteristiek COLLATING_SEQUENCE wordt in dit geval genegeerd. Met een positief effect op de efficiëntie.

En tenslotte is er natuurlijk ook nog de initiële query zelf, die een grote impact heeft op het resultaat van de PUSHDOWN-analyse. Dit is voornamelijk het geval, als deze query data wenst te combineren (e.g. UNION, JOIN) afkomstig van verschillende SERVER-databronnen.

GLOBAL OPTIMIZATION

De bedoeling van deze stap is duidelijk: het resultaat van de PUSHDOWN-analyse evalueren op het globale niveau van de initiële query. Met andere woorden, alle mogelijke acties worden geëvalueerd op een geïntegreerde manier; op basis van de evolutie van de geschatte kosten kunnen bepaalde individuele acties worden geweigerd en vervangen door andere.

De volgende elementen bepalen in belangrijke mate het resultaat van deze analyse: de karakteristeken van de SERVER data bron (snelheid van de machine, bandwidth, etc.), alsook een aantal NICKNAME-specificaties (die ook buiten een UDB federated server problematiek een impact hebben op elke acces-plan door een optimizer aangemaakt: indexes, statistieken).

GLOBAL OPTIMIZATION: SERVER-karakteristieken

De volgende karakteristieken en opties spelen een belangrijke rol.

- CPU_RATIO (default 1) geeft de relatieve snelheid aan van de SERVER-databron ten opzichte van de UDB federated server. Een waarde '1' geeft aan dat de SERVER en de UDB federated server over nagenoeg identieke CPU-karaktersiteken beschikken. Een waarde '0,5' geeft aan dat de SERVER twee maal sneller is dan de UDB federated server. Deze waarde gebruikt UDB federated server om te bepalen op welk platform de meest CPU-intensieve operaties moeten worden uitgevoerd.
- IO_RATIO (default 1) geeft de relatieve I/O-karakteristieken weer van de SERVER-databron ten opzichte van de UDB federated server. Bedoeling is aan te geven welke van beide servers relatief minst I/O bound is, en dus relatief meer I/O operaties aankan. Een waarde '1' geeft aan dat beide servers nagenoeg identieke I/O-karakteristieken hebben. Een waarde '0,5' geeft aan dat de SERVER twee maal sneller is op het gebeid van I/O-operaties dan de UDB federated server.
- COMM_RATE (default 2), in Mb/sec geeft de datatransfersnelheid van het netwerk aan. Dit geeft de optimizer een indicatie van de hoeveelheid data die zonder problemen door het netwerk kan worden verwerkt. Hoe hoger deze waarde, hoe groter de kans dat de optimizer meer werk laat verrichten door de UDB federated server.
- COLLATING_SEQUENCE (default Y), reeds in detail boven beschreven. De mogelijke waarden zijn 'Y', 'N', en 'I'. Deze laatste geeft weer dat er niet alleen een sorteerverschil is tussen SERVER-databron en de UDB federated server, maar dat de SERVER daarenboven ook case-insensitive is.
- PLAN_HINTS (default N) geeft aan of de SERVER optimizer eventueel hints genereert, toegevoegd aan de initiële query-syntax. Denk hierbij aan bijvoorbeeld het afdwingen van indexgebruik, of het opleggen van een specifieke join-volgorde. Deze hint wordt door de UDB federated server doorgegeven aan de SERVER-databron, die hierop zal reageren. Het netto-effect van deze hints is niet steeds duidelijk - met het gebruik hiervan moet omzichtig worden omgesprongen, en het effect moet steeds geëvalueerd worden.

GLOBAL OPTIMIZATION: NICKNAME-karakteristieken

Zoals steeds wordt bij het bepalen van een optimaal toegangspad door de optimizer rekening gehouden met de aanwezigheid van indexen op de data, alsook met de statistische beschrijving van de data. In de context van UDB federated servers zorgt dit voor een aantal bijkomende observaties.

- Indexes. Op het moment van creatie van een NICKNAME, wordt de SERVER-cataloog ondervraagd naar de aanwezigheid van indexes. Deze gegevens worden in de UDB federated server-cataloog opgenomen - aldus kan de optimizer hier later rekening mee houden.

Observaties: a) de gegevens in de DB2 federated server zijn uiteraard statisch van aard: een wijziging van de indexkarakteristieken op de SERVER is op het niveau van de UDB federated server niet gekend; dit probleem kan worden opgelost door de NICKNAME te verwijderen, en opnieuw aan te maken. b) Indien een NICKNAME verwijst naar een SERVER-view, bestaan natuurlijk geen indexen. Indexen op de onderliggende SERVER tabel worden door de UDB federated server NIET opgemerkt. Om performance-problemen te vermijden is het daarom mogelijk op de NICKNAME-index 'beschrijvingen' te definiëren, zeg maar indicaties voor de UDB federated server dat op de SERVER wel degelijk indexen bestaan.

- Statistieken. Op het moment van creatie van een NICKNAME, wordt de statistische informatie die opgeslagen is in de SERVER-cataloog, toegevoegd aan de UDB federated server-cataloog (o.a. de tabellen sysstat.tables en sysstat.indexes). Bij optimisatie achteraf worden deze door de optimizer gebruikt. Zoals steeds is het belangrijk deze statistieken op gezette tijdstippen te herberekenen. Twee technieken zijn mogelijk. Ofwel herberekent men de statistieken op het niveau van de SERVER (gebruik hiervoor de SERVER-specifieke commando's zoals RUNSTATS, dbms_stats.gather, etc.), waarna de NICKNAME op het niveau van de UDB federated server wordt verwijderd en heraangemaakt. Op dit moment worden de statistieken in de UDB federated server-cataloog 'ververst'. Ofwel worden de statistieken op de UDB federated server manueel aangepast door de systeembeheerder: manuele update van alle 'systat'-objecten is toegelaten. Deze laatste techniek is natuurlijk ook toepasbaar indien de statistieken foutief zouden zijn of onvolledig. Merk op dat dit alternatief het enige mogelijke alternatief is als gebruik wordt gemaakt van 'indexbeschrijvingen', als in bovenstaand punt uiteengezet.

De performance van de queries uitgevoerd op een DB2 federated database kan dus op verschillende manieren worden beïnvloed. Dit kan geschieden op het niveau van de individuele vraagstelling - de query - maar ook op het niveau van de opzet en definitie van de DB2 federated server - de SERVER- en NICKNAME-objecten. Het spreekt voor zich dat het nu de taak is van de DBA om, aan de hand van bijvoorbeeld visual explain tools, na te gaan wat het resultaat is van deze specificaties op de efficiëntie van de uitgevoerde queries.

DOSSIER 8

Indexen op variabele lengte kolommen

Indexpagina's bevatten sinds mensenheugenis de waarden uit de kolom waar de index op gecreëerd werd en pointers. Tenzij voor kolommen met de datatypes VARCHAR en VARGRAPHIC: hier bewaart DB2 de data in de tabel in het variabele lengte formaat maar zal paddingkarakters toevoegen tot de maximale lengte bereikt is voor de index. Dit leidt tot het verplicht lezen van data-pages waar in principe index-only-processing mogelijk zou geweest zijn. Reeds in vorige versies van DB2 probeerde men aan het 'index-only' probleem van VARCHARs en VARGRAPHICs een mouw te passen. In versie 8 van DB2 voor z/OS pakt men het ten gronde aan door de index keys van VARCHAR en VARGRAPHIC kolommen niet te padden. Men spreekt in dit kader van varying length index keys. Op de indexpagina's vindt men dus de datawaarde zelf en de lengte indicator.

Wanneer men NOT PADDED specifieert op het CREATE INDEX statement zullen keys van de variabele lengte kolommen niet aangevuld worden met paddingkarakters - met de optie PADDED wel. De default is afhankelijk van de PAD INDEXES BY DEFAULT optie op systeemniveau, die na een nieuwe installatie op NO staat. Na een migratie van V7 naar V8 staat deze op YES.

Er zijn zowel voordelen als nadelen verbonden aan varying length index keys. Index-only access wordt mogelijk gemaakt; en dit zal de performance verbeteren zonder de nadelen die veroorzaakt werden bij vorige versies van DB2. Voorts is er ook een besparing van opslagruimte. Een nadeel is dat voor vergelijken van waarden, strings dezelfde lengte moeten hebben; dus dat bij vergelijken van indexkeys, variabele lengte kolom per variabele lengte kolom die in de index opgenomen is, moet gepad worden.

Met het ALTER INDEX statement kan men PADDED/NON PADDED gedrag omwisselen. Wanneer we het paddinggedrag wijzigen komt de index in een REBUILD-pending (RBDP) toestand. Na het rebuilden aan de hand van een aantal alternatieve utilities (REBUILD INDEX, REORG TABLESPACE, of LOAD REPLACE utility) kan de index terug gebruikt worden. Dit ALTER INDEX statement heeft natuurlijk ook gevolgen voor bestaande plannen en packages (indien relevant) die als onbruikbaar worden aangeduid.

Katrien Platteborze (ABIS)

Scrollable cursors in DB2 for z/OS

Diane Hendrix (ABIS)

Inleiding

Het gebruik van een cursor is voor de meeste DB2-programmeurs dagelijkse kost. Wanneer het resultaat van een zoekactie meer dan 1 rij teruggeeft, kunnen we inderdaad niet anders dan met een cursor de hele resultatenverzameling ophalen en die rij per rij inlezen en verwerken. Vóór DB2 V7 for z/OS and OS/390 kon men een cursor slechts in één richting doorlopen: vooruit.

Natuurlijk hebben vindingrijke programmeurs wel technieken ontwikkeld om deze beperking te omzeilen. Bijvoorbeeld, voor het achterwaarts scrollen in een resultatenverzameling kan men de inhoud volledig lokaal gaan bewaren. Wanneer dan een scroll-back gevraagd wordt, kan men de gevraagde rijen onmiddellijk binnenhalen. Het probleem met deze oplossing is natuurlijk wel, dat wijzigingen die ondertussen in de database aangebracht worden, niet zichtbaar zijn voor de applicatie. Een andere oplossing, die de laatste beperking niet vertoont, bestaat erin de cursor van op een bepaalde positie altijd terug op te bouwen. Het nadeel hierbij is dan weer, dat wanneer je met een groot aantal rijen gaat werken, de antwoordtijd wel eens bedroevend laag zou kunnen zijn.

Vanaf V7 werd alles mogelijk. Een cursor in achterwaartse richting doorlopen, dezelfde rij meerdere malen ophalen op verschillende momenten in de applicatie, je helemaal in het begin van de resultatenverzameling herpositioneren. Het zijn maar enkele van de nieuwe mogelijkheden die een SCROLLABLE CURSOR kan bieden.

Dankzij het gebruik van SCROLLABLE CURSORs kunnen we nog de volgende positieve effecten noteren:

- minder programmeerwerk en dus (sterk) vereenvoudigde applicatielogica;
- minder noodzaak om meervoudige indexen op de tabellen te creëren (denk maar even aan het efficiënt opbouwen van een cursor in voorwaartse en achterwaartse volgorde);
- betere performance doordat minder interne sorteringen nodig zijn.

Niets dan goeds dus! Wat houdt er ons dan nog tegen om dit type van cursors volop te gebruiken? Eén, weliswaar grote, beperking. SCROLLABLE CURSORs kunnen gebruikt worden in een conversationele on-line CICS-DB2 omgeving, in TSO batch en in IMS/TM BMPs. In de traditionele pseudo-conversationele on-line CICS-DB2 omgeving (en ook in een IMS/TM-DB2 MPP omgeving) wordt er echter voor ge-

zorgd dat locks op resources worden vrijgegeven wanneer een scherm getoond wordt. Ook je lock op je SCROLLABLE CURSOR dus. En dat was het dan!

Dit belet ons echter niet, om toch eens alle mogelijkheden van SCROLLABLE CURSORs op een rijtje te zetten.

Types van SCROLLABLE CURSORs

Belangrijk om weten is, dat bij elk type van V7 SCROLLABLE CURSOR, op het moment van de OPEN CURSOR, de rijen van de resultatentabel gekopieerd worden in een zogenaamde system-declared temporary table (system-DTT).

- Bij de INSENSITIVE SCROLL cursor zijn zowel het aantal rijen van de resultatentabel, als de inhoud ervan, een vast gegeven. De cursor is als het ware ongevoelig voor wijzigingen (UPDATE, DELETE, INSERT) die in de basistabel optreden. Dit type van cursor is altijd read-only.

- Bij de SENSITIVE STATIC SCROLL cursor ligt het aantal rijen van de resultatentabel wel vast, maar kan de inhoud ervan wijzigen. Indien in de basistabel rijen verdwijnen door een DELETE of wanneer bepaalde velden veranderen door een UPDATE, zijn deze wijzigingen zichtbaar voor de cursor. Dit type van cursor laat UPDATE/DELETE WHERE CURRENT OF toe (de zogenaamde positioned UPDATE/DELETE), en is dus UPDATABLE. Rijen, die na het openen van de cursor worden toegevoegd zijn nooit zichtbaar.

Zoals we later nog zullen zien wordt de SENSITIVITY van een cursor echter niet enkel beïnvloed door wat je in de DECLARE CURSOR codeert, maar ook door de FETCH.

- De mogelijkheid om de resultatentabel niet enkel variabel te maken qua inhoud, maar ook qua grootte, is pas voorzien vanaf DB2 V8. De SENSITIVE DYNAMIC SCROLL cursor maakt geen gebruik van een DTT, maar werkt rechtstreeks met de rijen van de basistabel. Alle wijzigingen, ook INSERTs, zijn dan ook zichtbaar.

Gebruik van een SCROLLABLE CURSOR

Voorbeeld: zie *System-DTT* op p. 17.

Zoals we hiervoor al vermeld hebben, wordt bij het openen van de SCROLLABLE CURSOR de resultatentabel altijd gematerialiseerd in een DTT. DB2 bewaart deze DTTs in zijn TEMP database, in een TEMP tablespace. De TEMP database moet éénmaal aangemaakt worden, en kan automatisch door elke DB2-gebruiker gebruikt worden. DB2 beslist zelf welke tablespace gebruikt wordt. Let er wel op dat er voldoende tablespaces gedefinieerd zijn (voor elke pagesize) en dat ze voldoende groot zijn.

Net zoals bij een gewone cursor moeten er voor een SCROLLABLE CURSOR een DECLARE, OPEN, FETCH en CLOSE statement gecodeerd worden. We overlopen nu even wat deze statements inhouden.

Declare

In het DECLARE CURSOR statement duidt SCROLL aan, dat de cursor scrollable is, terwijl INSENSITIVE of SENSITIVE STATIC het type van SCROLLABLE cursor bepaalt. Een cursor van het INSENSITIVE type is per definitie read-only. De FOR UPDATE OF clausule mag niet gebruikt worden. Deze optie is wel mogelijk in een SENSITIVE cursor. Wel even opletten: sommige cursors worden door hun syntax read-only. Bijvoorbeeld expliciet, door een FOR FETCH/READ ONLY te coderen, of impliciet, door in het SELECT statement een kolomfunctie te gebruiken of een join te coderen. In tegenstelling tot een NON-SCROLLABLE CURSOR maakt een ORDER BY clausule een SCROLLABLE CURSOR niet read-only. Zie ook voorbeeld 1 - de 'fouten' waarvan sprake in dit voorbeeld worden tijdens het BIND-proces gegenereerd.

Voorbeeld 1: Alternatieve cursordeclaraties

```
DECLARE NSC CURSOR FOR
        SELECT NR, FIRSTNAME, LASTNAME FROM PERSONS ORDER BY NR

DECLARE C1 INSENSITIVE SCROLL CURSOR FOR
        SELECT NR, FIRSTNAME, LASTNAME FROM PERSONS

DECLARE C2 SENSITIVE STATIC SCROLL CURSOR FOR
        SELECT NR, FIRSTNAME, LASTNAME FROM PERSONS

DECLARE C3 SENSITIVE STATIC SCROLL CURSOR FOR
        SELECT NR, FIRSTNAME, LASTNAME FROM PERSONS
        ORDER BY NR
        FOR UPDATE OF FIRSTNAME

DECLARE C4 SENSITIVE STATIC SCROLL CURSOR FOR
        SELECT NR, FIRSTNAME, LASTNAME FROM PERSONS
        ORDER BY NR
        FOR FETCH ONLY

DECLARE C5 INSENSITIVE STATIC SCROLL CURSOR FOR
        SELECT COUNT(*) FROM PERSONS GROUP BY PLNAME

DECLARE C6 SENSITIVE STATIC SCROLL CURSOR FOR
        SELECT COUNT(*) FROM PERSONS GROUP BY PLNAME
Foutief, want kolomfuncties zijn niet toegelaten in een SENSITIVE
SCROLLABLE CURSOR (SQLCODE = -243).

DECLARE C7 INSENSITIVE STATIC SCROLL CURSOR FOR
        SELECT NR, FIRSTNAME, LASTNAME FROM PERSONS
        ORDER BY NR
        FOR UPDATE OF FIRSTNAME
Foutief, want een INSENSITIVE SCROLLABLE CURSOR is per definitie read-
only (SQLCODE = -228).
```

Open

Op het moment van de OPEN CURSOR wordt de DTT aangemaakt en opgevuld met de rijen die voldoen aan het SELECT-statement in de cursordeclaratie. Ook de record-identifier (RID) van elke rij wordt opgehaald en samen met de overige kolommen bewaard in de DTT. De positie van de cursor is net voor de eerste rij in de resultatentabel.

Na het uitvoeren van het OPEN CURSOR statement geven een aantal zones van de SQLCA-structuur informatie weer over de specifieke cursorkarakteristieken:

- SQLWARN1 duidt aan of de cursor SCROLLABLE(S) of NON-SCROLLABLE(N) is;
- SQLWARN4 duidt aan of de cursor INSENSITIVE(I) of SENSITIVE(S) is;
- SQLWARN5 duidt aan of de cursor kan gebruikt worden voor read-only(1), voor read/delete(2) of voor read/update/delete(4).

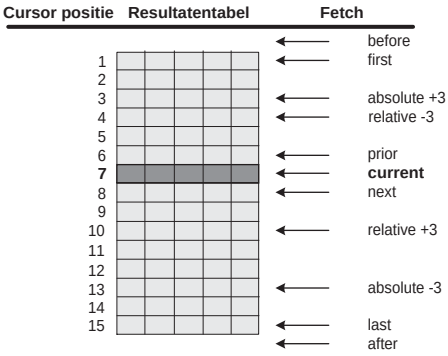
Voor de bovenvermelde voorbeelden - voorbeeld 1 - worden volgende waarden weerhouden:

CURSOR	SQLWARN1	SQLWARN4	SQLWARN5
NSC			
C1	S	I	1
C2	S	S	2
C3	S	S	4
C4	S	S	1
C5	S	I	1

Fetch

Bij een FETCH wordt de cursor gepositioneerd op een bepaalde rij in de resultatentabel en worden de bijbehorende kolomwaarden in de host-variabelen geplaatst. In een SCROLLABLE CURSOR kan je via de optie die je meegeeft bij het FETCH-statement, bepalen welke rij je zal ophalen. Figuur 1 geeft een overzicht van een aantal relevante opties. Merk echter op dat bij het gebruik van de opties AFTER en BEFORE, geen data wordt opgehaald!

Figuur 1: Diverse fetch opties (1)



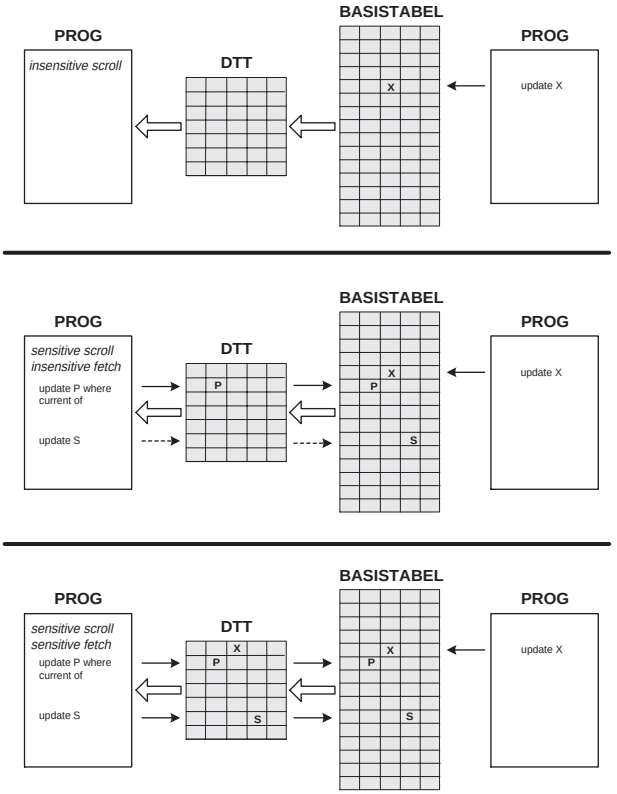
Een andere optie van het FETCH statement laat toe om de SENSITIVITY te beïnvloeden.

- Bij een INSENSITIVE CURSOR is enkel een INSENSITIVE FETCH toegelaten. UPDATEs of DELETEs die door de eigen applicatie of door een andere applicatie aangebracht worden in de basistabel, zijn niet zichtbaar. Dit komt omdat enkel de rijen uit de DTT worden geraadpleegd.

- Bij een SENSITIVE CURSOR kan men zowel een SENSITIVE FETCH (default) als een INSENSITIVE FETCH gebruiken. Een INSENSITIVE FETCH laat enkel toe om positioned UPDATEs en DELETEs te zien. Rijen gewijzigd door een 'searched' UPDATE/DELETE of wijzigingen aangebracht door een andere applicatie (en bevestigd met een COMMIT), zijn niet zichtbaar. Daarvoor moet men een SENSITIVE FETCH gebruiken.

Figuur 2 toont de verschillende situaties in geval van UPDATE-acties.

Figuur 2: Diverse fetch opties (2)



De volgende rijen zijn nooit zichtbaar in geen enkel type V7 SCROLLABLE CURSOR:

- rijen die na het openen van de cursor worden toegevoegd,
- rijen die door een UPDATE nu (=na het openen van de cursor) wel voldoen aan de WHERE conditie in de cursordeclaratie.

In een V8 SCROLLABLE DYNAMIC CURSOR zal men deze rijen wel kunnen zien.

zie *Voorbeeldapplicaties* op p. 17.

Close

Op het moment van de CLOSE CURSOR wordt de DTT verwijderd uit de TEMP database. Wanneer de cursor in een stored procedure wordt gebruikt, wordt de DTT pas verwijderd wanneer het oproepende programma eindigt.

Delete en Update holes

Zoals al werd vermeld, worden bij het openen van een SCROLLABLE CURSOR, niet enkel de gevraagde kolommen van elke rij in de DTT gekopieerd, maar bovendien ook de RID. Bij elke FETCH voor een SENSITIVE CURSOR, gebruikt DB2 deze RIDs om de corresponderende rij in de basistabel terug te vinden. Wanneer de rij tussen het openen van de cursor en de FETCH gewijzigd werd, kan dit resulteren in een

- delete hole: de rij werd verwijderd uit de basistabel. Er is geen corresponderende rij meer voor dit RID,
- update hole: de rij werd aangepast, zodat de rij niet meer voldoet aan de WHERE-conditie van de cursordeclaratie.

In beide gevallen zal DB2 een SQLCODE +222 teruggeven en worden de host-variabelen niet gebruikt. Er ontstaat geen update hole, wanneer het een UPDATE betreft van een kolom die niet vermeld is in de WHERE-conditie van de cursordeclaratie. In dit geval zal een SENSITIVE FETCH de rij met wijziging tonen. Een INSENSITIVE FETCH toont de rij zoals ze was voor de UPDATE.

Gebruik van functies

DB2 beschikt over twee soorten functies:

- kolomfuncties, die alle waarden van een kolom herleiden tot 1 waarde (bv. AVG, COUNT, MAX, ...). Deze kunnen niet gebruikt worden in een SENSITIVE CURSOR, omdat ze de cursor per definitie read-only maken. Wanneer ze gedefinieerd worden in een INSENSITIVE CURSOR wordt hun waarde vastgelegd op het moment van de OPEN CURSOR.
- scalaire functies, die werken op een kolom (bv. DATE, MONTH, VALUE, ...). Bij een INSENSITIVE CURSOR wordt de functie geëvalueerd op het moment van de OPEN CURSOR. Bij een SENSITIVE CURSOR gebeurt de evaluatie van de functie op het moment van de FETCH.

Dezelfde regels gelden voor het gebruik van bewerkingen in het SELECT- of WHERE-statement.

Locking

DB2 gebruikt voor de locking van een SCROLLABLE CURSOR bijna hetzelfde mechanisme als voor een gewone cursor :

- RR: lock op elke page/row die gelezen wordt (fetch)
- RS: lock op elke page/rij die voldoet aan stage 1 predicate
- CS: geen locks op de basistabel na het openen van de cursor; lock op elke gelezen page/rij (CURRENT DATA(YES)) of lock op elke gelezen page/rij enkel met de FOR UPDATE OF clausule (CURRENT DATA(NO))
- UR: geen locking

Voor een SCROLLABLE CURSOR maakt DB2 bovendien ook gebruik van 'optimistic locking concur by value'. We bekijken even hoe dit in zijn werk gaat wanneer isolation level CS en locksize row wordt gebruikt. Na het openen van de cursor blijft geen enkele rij van de basistabel gelocked. DB2 gaat ervan uit, dat er geen positioned UPDATE/DELETE zal gebeuren, m.a.w. DB2 is optimistisch. Bij elke FETCH wordt een lock op de rij aangevraagd en onmiddellijk terug vrijgegeven. Indien er dan toch een UPDATE/DELETE WHERE CURRENT OF gebeurt, zal er eerst een lock worden aangevraagd, en vervolgens nagegaan worden of de inhoud van de basistabel verschilt met wat er in de DTT gevonden wordt. Enkel de kolommen die in de SELECT van de cursordeclaratie staan, worden geëvalueerd. Worden er verschillen opgemerkt, dan volgt er geen UPDATE (SQLCODE -224). Dezelfde rij kan dan eventueel opnieuw opgehaald worden (inclusief de nieuwe wijzigingen) met een FETCH CURRENT, zodat de huidige kolomwaarden zichtbaar zijn.

Besluit

SCROLLABLE CURSORs vertonen zeker en vast een aantal interessante eigenschappen. Vanuit performance standpunt is het echter zeker niet bewezen dat het gebruik van een SCROLLABLE CURSOR een goede zaak is. Voorlopige testen wijzen immers uit dat de kost meestal hoger ligt, zeker in het geval van een SENSITIVE SCROLLABLE CURSOR.

Referenties

DB2 for z/OS Application Programming Topics (SG24-6300-00)

DB2 UDB Server for OS/390 and z/OS Version 7 Presentation Guide (SG24-6121-00)

DB2 for z/OS and OS/390 Version 7 Performance Topics (SG24-6129-00)

DB2 UDB for z/OS Version 8 Technical Preview (SG24-6871-00)

CURSUSPLANNING MEI - JUN 2004

DB2 concepten	375 EUR	07/06 (W)
DB2 for OS/390, een totaaloverzicht	1625 EUR	24-28/05 (W), 07-11/06 (L)
DB2 UDB, een totaaloverzicht	1625 EUR	24-25/05&01-03/06 (W)
RDBMS concepten	325 EUR	24/05 (W), 07/06 (L)
Basiskennis SQL	325 EUR	25/05(W), 08/06 (L)
DB2 for OS/390 basiscursus	975 EUR	09-11/06 (L)
DB2 UDB basiscursus	975 EUR	01-03/06 (W)
SQL workshop	700 EUR	21-22/06 (L)
DB2 for OS/390: SQL performance	1200 EUR	23-25/06 (W)
DB2 UDB applicatieperformance	400 EUR	08/06 (W)
Database applicatieprogrammering met JDBC	800 EUR	01-02/06 (W)
DB2 for OS/390 database administratie	1600 EUR	24-27/05(L)
DB2 UDB systeembeheer en performance	400 EUR	22/06 (W)
DB2 UDB en zijn extenders: XML en text search	200 EUR	04/06 (W)
DB2 UDB integratie met MQSeries	200 EUR	04/06 (W)
Tunen van Java applicaties en DB2	800 EUR	24-25/05(L), 21-22/06(W)
DB2 for OS/390 systems performance and tuning	1500 EUR	21-23/06(W)
DB2 week - een overzicht van uitdagende nieuwe features in DB2!		
° The procedural DBA	400 EUR	16/06 (L)
° JAVA and .NET in a DB2 environment	400 EUR	21/06 (L)
° Federated databases	400 EUR	23/06 (L)
° Extended SQL in DB2	400 EUR	28/06 (L)
° XML in DB2	400 EUR	29/06 (L)

Plaats: L = Leuven; W = Woerden; details en extra cursussen: www.abis.be

Postbus 220
Diestsevest 32
BE-3000 Leuven
Tel. 016/245610
Fax 016/245691
training@abis.be

abis
TRAINING & CONSULTING

Postbus 122
Pelmolenlaan 1-K
NL-3440 AC Woerden
Tel. 0348-435570
Fax 0348-432493
training@abis.be

Bijlagen

System-DTT

```
CREATE DATABASE DBTEMP
       STOGROUP TBGTSO01
       BUFFERPOOL BP1
       INDEXBP BP1
       AS TEMP

CREATE TABLESPACE TSTEMP01 IN DBTEMP
        USING STOGROUP TBGTSO01
              SEGSIZE 4
              PRIQTY 100
              SECQTY 100
              ERASE NO
        BUFFERPOOL BP1

CREATE TABLESPACE TSTEMP02 IN DBTEMP
        USING STOGROUP TBGTSO01
              SEGSIZE 4
              PRIQTY 100
              SECQTY 100
              ERASE NO
        BUFFERPOOL BP8K1
```

Voorbeeldapplicaties

Omgeving: COBOL, DB2 V7 for OS/390

Applicatie 1 - scrollable cursor

```
       IDENTIFICATION DIVISION.
      *************************
       PROGRAM-ID.    SCROLL01.
      *
      ****************************************************************
      *** PROGRAM 01: USE OF A INSENSITIVE SCROLLABLE CURSOR
      ****************************************************************
      *
       ENVIRONMENT DIVISION.
      ***********************
       INPUT-OUTPUT SECTION.
      *---------------------*
       FILE-CONTROL.
      *-------------*
           SELECT FILE-OUT  ASSIGN TO FILEOUT.

       DATA DIVISION.
      *==============
       FILE SECTION.
      *-------------
      *
       FD FILE-OUT.
       01 RECORD-OUT               PIC X(80).

       WORKING-STORAGE SECTION.
      *------------------------
      *
```

```
* HOST VARIABLES.
*----------------
 01  DCLTBPERSONS.
     10 PNO                  PIC S9(4) USAGE COMP.
     10 PLNAME               PIC X(40).
 01  NOT-FOUND               PIC S9(9) COMP VALUE +100.
 01  COUNTER-COMP            PIC 99 COMP VALUE 0.

* PRINTLINES
*-----------
 01 MESSAGE-LINE             PIC X(80).
 01 PRINTLINE1.
    03                       PIC X(10) VALUE 'COUNTER'.
    03                       PIC X(10) VALUE 'PERSON NO'.
    03                       PIC X(40) VALUE 'PERSON NAME'.
    03                       PIC X(20) VALUE ALL SPACES.
 01 PRINTLINE2               PIC X(80) VALUE ALL '-'.
 01 PRINTLINE3.
    03 COUNTER               PIC 99.
    03                       PIC X(8) VALUE SPACES.
    03 PNO                   PIC 9(4).
    03                       PIC X(6) VALUE SPACES.
    03 PLNAME                PIC X(40).
    03                       PIC X(20) VALUE ALL SPACES.
 01 ERROR-LINE.
    03                       PIC X(20) VALUE 'SQLCODE = '.
    03 SQL-CODE              PIC S9(4) SIGN LEADING SEPARATE.
    03                       PIC X(5) VALUE SPACES.
    03 SQL-MESSAGE           PIC X(50).
*
* SQL DECLARES.
*--------------
     EXEC SQL
       INCLUDE SQLCA
     END-EXEC.
*
     EXEC SQL DECLARE TBPERSONS TABLE
      (PNO                            SMALLINT NOT NULL,
       PLNAME                         CHAR(40) NOT NULL)
       END-EXEC.
*
     EXEC SQL
        DECLARE CI INSENSITIVE SCROLL CURSOR FOR
             SELECT PNO, PLNAME
             FROM TBPERSONS
             ORDER BY PNO
     END-EXEC.
*
 PROCEDURE DIVISION.
********************
 MAIN.
     PERFORM OPEN-FILE
     PERFORM OPEN-CURSOR
     PERFORM FETCH-AND-DISPLAY-ALL-PERSONS
     PERFORM REPOSITION-ON-POSITION-10
     PERFORM SEVERAL-FETCHES
     PERFORM CLOSE-CURSOR
     PERFORM CLOSE-FILE
     STOP RUN
     .

 OPEN-FILE.
     OPEN OUTPUT FILE-OUT
```

```
     .

 OPEN-CURSOR.
*------------
     EXEC SQL
        OPEN CI
     END-EXEC.
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN OPEN CURSOR CI ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR-AND-STOP
     END-IF
     MOVE 'CHARACTERISTICS OF THE CURSOR ' TO MESSAGE-LINE
     WRITE RECORD-OUT FROM MESSAGE-LINE
     WRITE RECORD-OUT FROM PRINTLINE2
     MOVE 'SQLWARN1 = '  TO MESSAGE-LINE
     WRITE RECORD-OUT FROM MESSAGE-LINE
     MOVE SQLWARN1   TO MESSAGE-LINE
     WRITE RECORD-OUT FROM MESSAGE-LINE
     MOVE 'SQLWARN4 = '  TO MESSAGE-LINE
     WRITE RECORD-OUT FROM MESSAGE-LINE
     MOVE SQLWARN4   TO MESSAGE-LINE
     WRITE RECORD-OUT FROM MESSAGE-LINE
     MOVE 'SQLWARN5 = ' TO MESSAGE-LINE
     WRITE RECORD-OUT FROM MESSAGE-LINE
     MOVE SQLWARN5   TO MESSAGE-LINE
     WRITE RECORD-OUT FROM MESSAGE-LINE
     .

 FETCH-AND-DISPLAY-ALL-PERSONS.
*------------------------------
     MOVE 'TOTAL LIST OF PERSONS' TO MESSAGE-LINE
     WRITE RECORD-OUT FROM MESSAGE-LINE
     WRITE RECORD-OUT FROM PRINTLINE2
     WRITE RECORD-OUT FROM PRINTLINE1
     WRITE RECORD-OUT FROM PRINTLINE2
     PERFORM FETCH-FORWARD
     PERFORM UNTIL SQLCODE NOT = 0
        ADD 1 TO COUNTER-COMP
        PERFORM DISPLAY-PERSON
        PERFORM FETCH-FORWARD
     END-PERFORM
     EVALUATE SQLCODE
        WHEN 0
           CONTINUE
        WHEN NOT-FOUND
           MOVE 'END-OF-LIST' TO SQL-MESSAGE
           PERFORM DISPLAY-ERROR
        WHEN OTHER
           MOVE 'ERROR IN FETCH-CI ' TO SQL-MESSAGE
           PERFORM DISPLAY-ERROR-AND-STOP
     END-EVALUATE
     .

 REPOSITION-ON-POSITION-10.
*--------------------------------------
     MOVE 0 TO COUNTER-COMP COUNTER
     EXEC SQL
        FETCH BEFORE FROM CI
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'REPOSITION ERROR IN FETCH BEFORE ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR-AND-STOP
     END-IF
```

```
           EXEC SQL
               FETCH ABSOLUTE 10 FROM CI
               INTO :DCLTBPERSONS
           END-EXEC
           IF SQLCODE NOT = 0
            MOVE 'REPOSITION ERROR IN FETCH ABSOLUTE 10' TO SQL-MESSAGE
              PERFORM DISPLAY-ERROR-AND-STOP
           END-IF

           MOVE 'REPOSITION FETCH ABSOLUTE 10' TO MESSAGE-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           WRITE RECORD-OUT FROM MESSAGE-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           PERFORM DISPLAY-PERSON
           .

       SEVERAL-FETCHES.
           EXEC SQL
               FETCH FIRST FROM CI
               INTO :DCLTBPERSONS
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE 'ERROR IN FETCH FIRST CI ' TO SQL-MESSAGE
              PERFORM DISPLAY-ERROR-AND-STOP
           END-IF
           MOVE 'FETCH FIRST' TO MESSAGE-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           WRITE RECORD-OUT FROM MESSAGE-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           PERFORM DISPLAY-PERSON

           EXEC SQL
               FETCH LAST FROM CI
               INTO :DCLTBPERSONS
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE 'ERROR IN FETCH LAST CI ' TO SQL-MESSAGE
              PERFORM DISPLAY-ERROR-AND-STOP
           END-IF
           WRITE RECORD-OUT FROM PRINTLINE2
           MOVE 'FETCH LAST' TO MESSAGE-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           WRITE RECORD-OUT FROM MESSAGE-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           PERFORM DISPLAY-PERSON

           EXEC SQL
               FETCH PRIOR FROM CI
               INTO :DCLTBPERSONS
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE 'ERROR IN FETCH PRIOR ' TO SQL-MESSAGE
              PERFORM DISPLAY-ERROR-AND-STOP
           END-IF
           MOVE 'FETCH PRIOR' TO MESSAGE-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           WRITE RECORD-OUT FROM MESSAGE-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           PERFORM DISPLAY-PERSON

           EXEC SQL
               FETCH ABSOLUTE 4 FROM CI
               INTO :DCLTBPERSONS
```

```
END-EXEC
IF SQLCODE NOT = 0
   MOVE 'ERROR IN FETCH ABSOLUTE 4 CI ' TO SQL-MESSAGE
   PERFORM DISPLAY-ERROR-AND-STOP
END-IF
MOVE 'FETCH ABSOLUTE 4' TO MESSAGE-LINE
WRITE RECORD-OUT FROM PRINTLINE2
WRITE RECORD-OUT FROM MESSAGE-LINE
WRITE RECORD-OUT FROM PRINTLINE2
PERFORM DISPLAY-PERSON
EXEC SQL
    FETCH ABSOLUTE -5 FROM CI
    INTO :DCLTBPERSONS
END-EXEC
IF SQLCODE NOT = 0
   MOVE 'ERROR IN FETCH ABSOLUTE -5 ' TO SQL-MESSAGE
   PERFORM DISPLAY-ERROR-AND-STOP
END-IF
MOVE 'FETCH ABSOLUTE -5' TO MESSAGE-LINE
WRITE RECORD-OUT FROM PRINTLINE2
WRITE RECORD-OUT FROM MESSAGE-LINE
WRITE RECORD-OUT FROM PRINTLINE2
PERFORM DISPLAY-PERSON

EXEC SQL
    FETCH RELATIVE 2 FROM CI
    INTO :DCLTBPERSONS
END-EXEC
IF SQLCODE NOT = 0
   MOVE 'ERROR IN FETCH RELATIVE 2 CI ' TO SQL-MESSAGE
   PERFORM DISPLAY-ERROR-AND-STOP
END-IF
MOVE 'FETCH RELATIVE 2' TO MESSAGE-LINE
WRITE RECORD-OUT FROM PRINTLINE2
WRITE RECORD-OUT FROM MESSAGE-LINE
WRITE RECORD-OUT FROM PRINTLINE2
PERFORM DISPLAY-PERSON

EXEC SQL
    FETCH RELATIVE -5  FROM CI
    INTO :DCLTBPERSONS
END-EXEC
IF SQLCODE NOT = 0
   MOVE 'ERROR IN FETCH RELATIVE -5 CI ' TO SQL-MESSAGE
   PERFORM DISPLAY-ERROR-AND-STOP
END-IF
MOVE 'FETCH RELATIVE -5' TO MESSAGE-LINE
WRITE RECORD-OUT FROM PRINTLINE2
WRITE RECORD-OUT FROM MESSAGE-LINE
WRITE RECORD-OUT FROM PRINTLINE2
PERFORM DISPLAY-PERSON

EXEC SQL
    FETCH AFTER FROM CI
END-EXEC
IF SQLCODE NOT = 0
   MOVE 'ERROR IN FETCH AFTER ' TO SQL-MESSAGE
   PERFORM DISPLAY-ERROR-AND-STOP
END-IF
MOVE 'FETCH AFTER' TO MESSAGE-LINE
WRITE RECORD-OUT FROM PRINTLINE2
WRITE RECORD-OUT FROM MESSAGE-LINE
WRITE RECORD-OUT FROM PRINTLINE2
```

```
     PERFORM DISPLAY-PERSON

     EXEC SQL
         FETCH NEXT FROM CI
         INTO :DCLTBPERSONS
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN FETCH NEXT / AFTER ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR-AND-STOP
     END-IF
     MOVE 'FETCH NEXT/AFTER' TO MESSAGE-LINE
     WRITE RECORD-OUT FROM PRINTLINE2
     WRITE RECORD-OUT FROM MESSAGE-LINE
     WRITE RECORD-OUT FROM PRINTLINE2
     PERFORM DISPLAY-PERSON
     .

 CLOSE-CURSOR.
*------------
     EXEC SQL
        CLOSE CI
     END-EXEC.

     IF SQLCODE NOT = 0
        MOVE 'ERROR IN CLOSE CURSOR CI ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR-AND-STOP
     END-IF
     .

 CLOSE-FILE.
*------------
     CLOSE FILE-OUT
     .

 DISPLAY-PERSON.
*--------------
     MOVE CORRESPONDING DCLTBPERSONS TO PRINTLINE3
     MOVE COUNTER-COMP TO COUNTER
     WRITE RECORD-OUT FROM PRINTLINE3
     .

 FETCH-FORWARD.
*--------------
     EXEC SQL
        FETCH CI INTO :DCLTBPERSONS
     END-EXEC
     .

 DISPLAY-ERROR.
*--------------
     MOVE SQLCODE TO SQL-CODE
     WRITE RECORD-OUT FROM PRINTLINE2
     WRITE RECORD-OUT FROM ERROR-LINE
     WRITE RECORD-OUT FROM PRINTLINE2
     .
 DISPLAY-ERROR-AND-STOP.
*--------------
     MOVE SQLCODE TO SQL-CODE
     WRITE RECORD-OUT FROM PRINTLINE2
     WRITE RECORD-OUT FROM ERROR-LINE
     WRITE RECORD-OUT FROM PRINTLINE2
     GOBACK
     .
```

```
OUTPUT OF THIS PROGRAM
----------------------


CHARACTERISTICS OF THE CURSOR
------------------------------------------------------------------------
SQLWARN1 =
S
SQLWARN4 =
I
SQLWARN5 =
1
TOTAL LIST OF PERSONS
------------------------------------------------------------------------
COUNTER   PERSON NO PERSON NAME
------------------------------------------------------------------------
01        0001      SMITHS
02        0002      TAVERNIER
03        0003      DE KEYSER
04        0004      MAK
05        0005      NIEHOF
06        0006      VAN HEIJKOOP
07        0007      DE GROOT
08        0008      PEREZ
09        0009      LIVIER
10        0010      LUTZ
11        0011      LOOSE
12        0012      BENOIT
13        0013      BENOIT
14        0014      DETROIT
15        0015      SPENSER
16        0016      HANEGREEFS
17        0017      SCHUMACHER
18        0018      GELADE
19        0019      COPPIETERS
20        0020      DE WINDT
21        0021      DE SCHRIJVER
22        0022      HENDERSON
23        0023      DELANGHE
24        0024      VAN DE BROECK
25        0025      MEURIS
26        0026      HEBBELYNCK
27        0027      DE GRAAF
28        0028      TYTGAT
29        0029      DEVISSER
30        0030      DE WILDE
31        0031      DEHEM
32        0032      BUENK
33        0033      PIELAGE
34        0034      DE BRUYN
35        0035      DE SMET
36        0036      ADAMSON
37        0037      GOYENS
38        0038      GERRIES
39        0039      DE CORTE
40        0040      PARKER
------------------------------------------------------------------------
SQLCODE =           +0100     END-OF-LIST
------------------------------------------------------------------------
------------------------------------------------------------------------
```

```
REPOSITION FETCH ABSOLUTE 10
--------------------------------------------------------------------------------
00        0010      LUTZ
--------------------------------------------------------------------------------
FETCH FIRST
--------------------------------------------------------------------------------
00        0001      SMITHS
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
FETCH LAST
--------------------------------------------------------------------------------
00        0040      PARKER
--------------------------------------------------------------------------------
FETCH PRIOR
--------------------------------------------------------------------------------
00        0039      DE CORTE
--------------------------------------------------------------------------------
FETCH ABSOLUTE 4
--------------------------------------------------------------------------------
00        0004      MAK
--------------------------------------------------------------------------------
FETCH ABSOLUTE -5
--------------------------------------------------------------------------------
00        0036      ADAMSON
--------------------------------------------------------------------------------
FETCH RELATIVE 2
--------------------------------------------------------------------------------
00        0038      GERRIES
--------------------------------------------------------------------------------
FETCH RELATIVE -5
--------------------------------------------------------------------------------
00        0033      PIELAGE
--------------------------------------------------------------------------------
FETCH AFTER
--------------------------------------------------------------------------------
00        0033      PIELAGE
--------------------------------------------------------------------------------
SQLCODE =           +0100     ERROR IN FETCH NEXT / AFTER
--------------------------------------------------------------------------------
```

Applicatie 2 - sensitive en insensitive fetch - verschillende cursors

```
       IDENTIFICATION DIVISION.
      ************************
       PROGRAM-ID.    SCROLL17.
      *
      ****************************************************************
      *** PROGRAM 17: CURSOR TESTING
      ****************************************************************
      *
       ENVIRONMENT DIVISION.
      *********************
       INPUT-OUTPUT SECTION.
      *---------------------*
       FILE-CONTROL.
      *------------*
           SELECT FILE-OUT  ASSIGN TO FILEOUT.

       DATA DIVISION.
      *==============
       FILE SECTION.
      *-------------
      *
       FD FILE-OUT.
       01 RECORD-OUT              PIC X(80).
       WORKING-STORAGE SECTION.
      *------------------------
      *
      * HOST VARIABLES.
      *----------------
       01  DCLTBPERSONS.
           10 PNO                  PIC S9(4) USAGE COMP.
           10 PLNAME               PIC X(40).
       01  NOT-FOUND               PIC S9(9) COMP VALUE +100.
       01  COUNTER-COMP            PIC 99 COMP VALUE 0.
       01  CHOICE                  PIC 99.


      * PRINTLINES
      *-----------
       01 MESSAGE-LINE             PIC X(80).
       01 PRINTLINE1.
          03                      PIC X(10) VALUE 'COUNTER'.
          03                      PIC X(10) VALUE 'PERSON NO'.
          03                      PIC X(40) VALUE 'PERSON NAME'.
          03                      PIC X(20) VALUE ALL SPACES.
       01 PRINTLINE2               PIC X(80) VALUE ALL '-'.
       01 PRINTLINE3.
          03 COUNTER               PIC 99.
          03                      PIC X(8) VALUE SPACES.
          03 PNO                   PIC 9(4).
          03                      PIC X(6) VALUE SPACES.
          03 PLNAME                PIC X(40).
          03                      PIC X(20) VALUE ALL SPACES.
       01 ERROR-LINE.
          03                      PIC X(20) VALUE 'SQLCODE = '.
          03 SQL-CODE              PIC S9(4) SIGN LEADING SEPARATE.
          03                      PIC X(5) VALUE SPACES.
          03 SQL-MESSAGE           PIC X(50).
       01 SQLWARNING1.
          03                      PIC X(30) VALUE 'SQLWARN1 = '.
          03 SQLW1                 PIC X(20).
       01 SQLWARNING4.
          03                      PIC X(30) VALUE 'SQLWARN4 = '.
```

```
    03 SQLW4                  PIC X(20).
 01 SQLWARNING5.
    03                        PIC X(30) VALUE 'SQLWARN5 = '.
    03 SQLW5                  PIC X(20).
*
* SQL DECLARES.
*--------------
     EXEC SQL
       INCLUDE SQLCA
     END-EXEC.
*
     EXEC SQL DECLARE TBPERSONS TABLE
      (PNO                            SMALLINT NOT NULL,
       PLNAME                         CHAR(40) NOT NULL
       )
       END-EXEC.
     EXEC SQL
        DECLARE C1  CURSOR FOR
             SELECT PNO, PLNAME
             FROM TBPERSONS
             ORDER BY PNO
     END-EXEC.
     EXEC SQL
        DECLARE C2  INSENSITIVE SCROLL CURSOR FOR
             SELECT PNO, PLNAME
             FROM TBPERSONS
             ORDER BY PNO
     END-EXEC.
     EXEC SQL
        DECLARE C3  SENSITIVE STATIC SCROLL CURSOR FOR
             SELECT PNO, PLNAME
             FROM TBPERSONS
             ORDER BY PNO
     END-EXEC.
     EXEC SQL
        DECLARE C4 SENSITIVE STATIC SCROLL CURSOR FOR
             SELECT PNO, PLNAME
             FROM TBPERSONS
             ORDER BY PNO
             FOR UPDATE OF PLNAME
     END-EXEC.
* THIS CURSOR PRODUCES A BIND ERROR (-228)
*    EXEC SQL
*       DECLARE C5  INSENSITIVE SCROLL CURSOR FOR
*            SELECT PNO, PLNAME
*            FROM TBPERSONS
*            FOR UPDATE OF PLNAME
*   END-EXEC.
     EXEC SQL
        DECLARE C6  SENSITIVE STATIC SCROLL CURSOR FOR
             SELECT PNO, PLNAME
             FROM TBPERSONS
             ORDER BY PNO
             FOR FETCH ONLY
     END-EXEC.
* THIS CURSOR PRODUCES A BIND ERROR ( -243)
*    EXEC SQL
*       DECLARE C7  SENSITIVE STATIC SCROLL CURSOR FOR
*            SELECT COUNT(*)
*            FROM TBPERSONS
*            GROUP BY PLNAME
*    END-EXEC.
     EXEC SQL
```

```
        DECLARE C8  INSENSITIVE SCROLL CURSOR FOR
             SELECT COUNT(*)
             FROM TBPERSONS
             GROUP BY PLNAME
     END-EXEC.
*
 PROCEDURE DIVISION.
*******************
 MAIN.
     PERFORM OPEN-FILE
     PERFORM OPEN-CURSORS
     PERFORM CLOSE-CURSORS
     STOP RUN
     .

 OPEN-FILE.
*----------
     OPEN OUTPUT FILE-OUT
     .

 OPEN-CURSORS.
*------------
     EXEC SQL
        OPEN C1
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN OPEN CURSOR C1 ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR
     END-IF
     MOVE 'CHARACTERISTICS OF THE C1 ' TO MESSAGE-LINE
     PERFORM DISPLAY-CURSOR-STATUS

     EXEC SQL
        OPEN C2
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN OPEN CURSOR C2 ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR
     END-IF
     MOVE 'CHARACTERISTICS OF THE C2 ' TO MESSAGE-LINE
     PERFORM DISPLAY-CURSOR-STATUS

     EXEC SQL
        OPEN C3
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN OPEN CURSOR C3 ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR
     END-IF
     MOVE 'CHARACTERISTICS OF THE C3 ' TO MESSAGE-LINE
     PERFORM DISPLAY-CURSOR-STATUS

     EXEC SQL
        OPEN C4
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN OPEN CURSOR C4 ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR
     END-IF
     MOVE 'CHARACTERISTICS OF THE C4 ' TO MESSAGE-LINE
     PERFORM DISPLAY-CURSOR-STATUS

*    EXEC SQL
```

```
*       OPEN C5
*    END-EXEC
*    IF SQLCODE NOT = 0
*       MOVE 'ERROR IN OPEN CURSOR C5 ' TO SQL-MESSAGE
*       PERFORM DISPLAY-ERROR
*    END-IF
*    MOVE 'CHARACTERISTICS OF THE C5 ' TO MESSAGE-LINE
*    PERFORM DISPLAY-CURSOR-STATUS

     EXEC SQL
        OPEN C6
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN OPEN CURSOR C6 ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR
     END-IF
     MOVE 'CHARACTERISTICS OF THE C6 ' TO MESSAGE-LINE
     PERFORM DISPLAY-CURSOR-STATUS

*    EXEC SQL
*       OPEN C7
*
*    END-EXEC
*    IF SQLCODE NOT = 0
*       MOVE 'ERROR IN OPEN CURSOR C7 ' TO SQL-MESSAGE
*       PERFORM DISPLAY-ERROR
*    END-IF
*    MOVE 'CHARACTERISTICS OF THE C7 ' TO MESSAGE-LINE
*    PERFORM DISPLAY-CURSOR-STATUS

     EXEC SQL
        OPEN C8
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN OPEN CURSOR C8 ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR
     END-IF
     MOVE 'CHARACTERISTICS OF THE C8 ' TO MESSAGE-LINE
     PERFORM DISPLAY-CURSOR-STATUS

     .

 CLOSE-CURSORS.
*------------
     EXEC SQL
        CLOSE C1
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN CLOSE CURSOR C1 ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR
     END-IF
     EXEC SQL
        CLOSE C2
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN CLOSE CURSOR C2 ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR
     END-IF
     EXEC SQL
        CLOSE C3
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN CLOSE CURSOR C3 ' TO SQL-MESSAGE
```

```
           PERFORM DISPLAY-ERROR
        END-IF
        EXEC SQL
           CLOSE C4
        END-EXEC
        IF SQLCODE NOT = 0
           MOVE 'ERROR IN CLOSE CURSOR C4 ' TO SQL-MESSAGE
           PERFORM DISPLAY-ERROR
        END-IF
*       EXEC SQL
*          CLOSE C5
*       END-EXEC
*       IF SQLCODE NOT = 0
*          MOVE 'ERROR IN CLOSE CURSOR C5 ' TO SQL-MESSAGE
*          PERFORM DISPLAY-ERROR
*       END-IF
*
        EXEC SQL
           CLOSE C6
        END-EXEC
        IF SQLCODE NOT = 0
           MOVE 'ERROR IN CLOSE CURSOR C6 ' TO SQL-MESSAGE
           PERFORM DISPLAY-ERROR
        END-IF
*       EXEC SQL
*          CLOSE C7
*       END-EXEC
*       IF SQLCODE NOT = 0
*          MOVE 'ERROR IN CLOSE CURSOR C7 ' TO SQL-MESSAGE
*          PERFORM DISPLAY-ERROR
*       END-IF
        EXEC SQL
           CLOSE C8
        END-EXEC
        IF SQLCODE NOT = 0
           MOVE 'ERROR IN CLOSE CURSOR C8 ' TO SQL-MESSAGE
           PERFORM DISPLAY-ERROR
        END-IF
        .

    CLOSE-FILE.
   *------------
        CLOSE FILE-OUT
        .

    DISPLAY-CURSOR-STATUS.
   *----------------------
        WRITE RECORD-OUT FROM MESSAGE-LINE
        WRITE RECORD-OUT FROM PRINTLINE2
        MOVE SQLWARN1   TO SQLW1
        MOVE SQLWARN4   TO SQLW4
        MOVE SQLWARN5   TO SQLW5
        WRITE RECORD-OUT FROM SQLWARNING1
        WRITE RECORD-OUT FROM SQLWARNING4
        WRITE RECORD-OUT FROM SQLWARNING5
        WRITE RECORD-OUT FROM PRINTLINE2
        .

    DISPLAY-ERROR.
   *--------------
        MOVE SQLCODE TO SQL-CODE
        WRITE RECORD-OUT FROM PRINTLINE2
        WRITE RECORD-OUT FROM ERROR-LINE
```

```
           WRITE RECORD-OUT FROM PRINTLINE2
           .
       DISPLAY-ERROR-AND-STOP.
      *--------------
           MOVE SQLCODE TO SQL-CODE
           WRITE RECORD-OUT FROM PRINTLINE2
           WRITE RECORD-OUT FROM ERROR-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           GOBACK
           .
```

```
OUTPUT OF THIS PROGRAM
----------------------

CHARACTERISTICS OF THE C1
------------------------------------------------------------------------
SQLWARN1 =
SQLWARN4 =
SQLWARN5 =
------------------------------------------------------------------------
CHARACTERISTICS OF THE C2
------------------------------------------------------------------------
SQLWARN1 =                    S
SQLWARN4 =                    I
SQLWARN5 =                    1
------------------------------------------------------------------------
CHARACTERISTICS OF THE C3
------------------------------------------------------------------------
SQLWARN1 =                    S
SQLWARN4 =                    S
SQLWARN5 =                    2
------------------------------------------------------------------------
CHARACTERISTICS OF THE C4
------------------------------------------------------------------------
SQLWARN1 =                    S
SQLWARN4 =                    S
SQLWARN5 =                    4
------------------------------------------------------------------------
CHARACTERISTICS OF THE C6
------------------------------------------------------------------------
SQLWARN1 =                    S
SQLWARN4 =                    S
SQLWARN5 =                    1
------------------------------------------------------------------------
CHARACTERISTICS OF THE C8
------------------------------------------------------------------------
SQLWARN1 =                    S
SQLWARN4 =                    I
SQLWARN5 =                    1
------------------------------------------------------------------------
```

Applicatie 3 - verschillende cursordefinities

```
       IDENTIFICATION DIVISION.
      ************************
       PROGRAM-ID.    SCROLL20.
      *
      ***************************************************************
      *** PROGRAM 20:
      ***           : C0 SENSITIVE SCROLLABLE  CURSOR
      ***              WITHOUT WHERE CONDITION
      ***             CI SENSITIVE SCROLLABLE  CURSOR
      ***               WITH WHERE CONDITION  ON PLNAME
      ***             CII SENSITIVE SCROLLABLE  CURSOR
      ***               WITH WHERE CONDITION  ON PNO
      ***             CIII SENSITIVE SCROLLABLE  CURSOR
      ***               WITH WHERE CONDITION  ON PLNAME
      ***               FOR UPDATE OF PLNAME
      ***             CIV SENSITIVE SCROLLABLE  CURSOR
      ***               WITH WHERE CONDITION  ON PNO
      ***               FOR UPDATE OF PNO
      ***            POSITION UPDATE IN TABLE DURING CURSOR PROCESSING
      ***               (UPDATE OF PLNAME WHERE PNO = 3)
      ***               (UPDATE OF PNO 3 TO PNO = 99)
      ***            SEARCHED UPDATE IN TABLE DURING CURSOR PROCESSING
      ***               (UPDATE OF PLNAME WHERE PNO = 3)
      ***               (UPDATE OF PNO 3 TO PNO = 88)
      ***             SENSITIVE FETCH FOR ALL CURSORS
      ***               BEFORE AND AFTER UPDATES
      ***             IF YOU UPDATE PLNAME
      ***             POS UPDATE VISIBLE IN C0 (SQLCODE = 0)
      ***             POS UPDATE VISIBLE IN CII(SQLCODE = 0)
      ***             POS UPDATE VISIBLE IN CIV (SQLCODE = 0)
      ***             SEA UPDATE VISIBLE IN C0 (SQLCODE = 0)
      ***             SEA UPDATE VISIBLE IN CII(SQLCODE = 0)
      ***             SEA UPDATE VISIBLE IN CIV(SQLCODE = 0)
      ***             UPDATE HOLE DETECTED IN CI (SQLCODE = 222)
      ***             UPDATE HOLE DETECTED IN CIII (SQLCODE = 222)
      ***             IF YOU UPDATE PNO
      ***             POS UPDATE VISIBLE IN C0 (SQLCODE = 0)
      ***             POS UPDATE VISIBLE IN CI (SQLCODE = 0)
      ***             POS UPDATE VISIBLE IN CIII(SQLCODE = 0)
      ***             SEA UPDATE VISIBLE IN C0 (SQLCODE = 0)
      ***             SEA UPDATE VISIBLE IN CI (SQLCODE = 0)
      ***             SEA UPDATE VISIBLE IN CIII SQLCODE = 0)
      ***             UPDATE HOLE DETECTED IN CII(SQLCODE = 222)
      ***             UPDATE HOLE DETECTED IN CIV (SQLCODE = 222)
      ***************************************************************
      *
       ENVIRONMENT DIVISION.
      *********************
       INPUT-OUTPUT SECTION.
      *--------------------*
       FILE-CONTROL.
      *------------*
           SELECT FILE-OUT  ASSIGN TO FILEOUT.

       DATA DIVISION.
      *==============
       FILE SECTION.
      *-------------
      *
       FD FILE-OUT.
       01 RECORD-OUT              PIC X(80).
       WORKING-STORAGE SECTION.
```

```
*------------------------
*
* HOST VARIABLES.
*----------------
 01  DCLTBPERSONS.
     10 PNO                  PIC S9(4) USAGE COMP.
     10 PLNAME               PIC X(40).
 01  NOT-FOUND               PIC S9(9) COMP VALUE +100.
 01  COUNTER-COMP            PIC 99 COMP VALUE 0.
 01  CHOICE                  PIC 99.


* PRINTLINES
*-----------
 01 MESSAGE-LINE            PIC X(80).
 01 PRINTLINE1.
    03                      PIC X(10) VALUE 'COUNTER'.
    03                      PIC X(10) VALUE 'PERSON NO'.
    03                      PIC X(40) VALUE 'PERSON NAME'.
    03                      PIC X(20) VALUE ALL SPACES.
 01 PRINTLINE2              PIC X(80) VALUE ALL '-'.
 01 PRINTLINE3.
    03 PNO                  PIC 9(4).
    03                      PIC X(6) VALUE SPACES.
    03 PLNAME               PIC X(40).
    03                      PIC X(20) VALUE ALL SPACES.
 01 ERROR-LINE.
    03                      PIC X(20) VALUE 'SQLCODE = '.
    03 SQL-CODE             PIC S9(4) SIGN LEADING SEPARATE.
    03                      PIC X(5) VALUE SPACES.
    03 SQL-MESSAGE          PIC X(50).
 01 SQLWARNING1.
    03                      PIC X(30) VALUE 'SQLWARN1 = '.
    03 SQLW1                PIC X(20).
 01 SQLWARNING4.
    03                      PIC X(30) VALUE 'SQLWARN4 = '.
    03 SQLW4                PIC X(20).
 01 SQLWARNING5.
    03                      PIC X(30) VALUE 'SQLWARN5 = '.
    03 SQLW5                PIC X(20).
*
* SQL DECLARES.
*--------------
     EXEC SQL
       INCLUDE SQLCA
     END-EXEC.
*
     EXEC SQL DECLARE TBPERSONS TABLE
      (PNO                          SMALLINT NOT NULL,
       PLNAME                       CHAR(40) NOT NULL
       )
       END-EXEC.
*
     EXEC SQL
        DECLARE C0 SENSITIVE STATIC SCROLL CURSOR FOR
             SELECT PNO, PLNAME
             FROM TBPERSONS
             ORDER BY PNO
     END-EXEC.

     EXEC SQL
        DECLARE CI SENSITIVE STATIC SCROLL CURSOR FOR
             SELECT PNO, PLNAME
```

```
              FROM TBPERSONS
              WHERE PLNAME LIKE '%DE%'
              ORDER BY PNO
     END-EXEC.
*

     EXEC SQL
        DECLARE CII SENSITIVE STATIC SCROLL CURSOR FOR
              SELECT PNO, PLNAME
              FROM TBPERSONS
              WHERE PNO <= 39
              ORDER BY PNO
     END-EXEC.

     EXEC SQL
        DECLARE CIII SENSITIVE STATIC SCROLL CURSOR FOR
              SELECT PNO, PLNAME
              FROM TBPERSONS
              WHERE PLNAME LIKE '%DE%'
              ORDER BY PNO
              FOR UPDATE OF PLNAME
     END-EXEC.
*

     EXEC SQL
        DECLARE CIV SENSITIVE STATIC SCROLL CURSOR FOR
              SELECT PNO, PLNAME
              FROM TBPERSONS
              WHERE PNO <= 39
              ORDER BY PNO
              FOR UPDATE OF PNO
     END-EXEC.

 PROCEDURE DIVISION.
********************
 MAIN.
     PERFORM OPEN-FILE
*    PERFORM ACCEPT-CHOICE
     PERFORM OPEN-CURSORS-S
     PERFORM FETCH-CURSORS
*    EVALUATE CHOICE
*         WHEN 01
*            MOVE 'SEARCHED UPDATE ' TO MESSAGE-LINE
*            WRITE RECORD-OUT FROM MESSAGE-LINE
*            WRITE RECORD-OUT FROM PRINTLINE2
*            PERFORM UPDATE-PLNAME
*         WHEN 02
*            MOVE 'POSITIONED UPDATE ' TO MESSAGE-LINE
*            WRITE RECORD-OUT FROM MESSAGE-LINE
*            WRITE RECORD-OUT FROM PRINTLINE2
*            PERFORM UPDATE-PNO
*         WHEN OTHER  MOVE 'WRONG CHOICE' TO MESSAGE-LINE
*                     WRITE RECORD-OUT FROM MESSAGE-LINE
*                     GOBACK
*    END-EVALUATE
     PERFORM FETCH-CURSORS
     PERFORM CLOSE-CURSORS-S
     STOP RUN
     .

*ACCEPT-CHOICE.
*--------------
*    ACCEPT CHOICE
```

```
*    .

 FETCH-CURSORS.
*--------------
     PERFORM FETCH-C0
     PERFORM FETCH-CI
     PERFORM FETCH-CII
     PERFORM FETCH-CIII
     PERFORM FETCH-CIV
     .

 OPEN-FILE.
*----------
     OPEN OUTPUT FILE-OUT
     .


 OPEN-CURSORS-S.
*------------
     EXEC SQL
        OPEN C0
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN OPEN CURSOR C0 ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR
     END-IF
     MOVE 'CHARACTERISTICS OF THE C0 ' TO MESSAGE-LINE
     PERFORM DISPLAY-CURSOR-STATUS

     EXEC SQL
        OPEN CI
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN OPEN CURSOR CI ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR-AND-STOP
     END-IF
     MOVE 'CHARACTERISTICS OF THE CI ' TO MESSAGE-LINE
     PERFORM DISPLAY-CURSOR-STATUS

     EXEC SQL
        OPEN CII
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN OPEN CURSOR CII' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR-AND-STOP
     END-IF
     MOVE 'CHARACTERISTICS OF THE CII' TO MESSAGE-LINE
     PERFORM DISPLAY-CURSOR-STATUS

     EXEC SQL
        OPEN CIII
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN OPEN CURSOR CIII' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR-AND-STOP
     END-IF
     MOVE 'CHARACTERISTICS OF THE CIII ' TO MESSAGE-LINE
     PERFORM DISPLAY-CURSOR-STATUS

     EXEC SQL
        OPEN CIV
     END-EXEC
     IF SQLCODE NOT = 0
```

```
           MOVE 'ERROR IN OPEN CURSOR CIV' TO SQL-MESSAGE
           PERFORM DISPLAY-ERROR-AND-STOP
        END-IF
        MOVE 'CHARACTERISTICS OF THE CIV' TO MESSAGE-LINE
        PERFORM DISPLAY-CURSOR-STATUS

        .

 UPDATE-PLNAME.
*---------------
        PERFORM SEARCHED-UPDATE-PLNAME
        PERFORM POSITIONED-UPDATE-PLNAME
        .

 UPDATE-PNO.
*---------------
        PERFORM SEARCHED-UPDATE-PNO
        PERFORM POSITIONED-UPDATE-PNO
        .

 SEARCHED-UPDATE-PLNAME.
*----------------------
        EXEC SQL
             UPDATE TBPERSONS
             SET PLNAME = 'SEARCHED UPDATE'
             WHERE PNO = 27
        END-EXEC
        IF SQLCODE NOT = 0
           MOVE 'ERROR IN SEARCHED-UPDATE-PLNAME ' TO SQL-MESSAGE
           PERFORM DISPLAY-ERROR-AND-STOP
        END-IF
        MOVE 'SEARCHED-UPDATE-PLNAME WAS DONE' TO MESSAGE-LINE
        WRITE RECORD-OUT FROM PRINTLINE2
        WRITE RECORD-OUT FROM MESSAGE-LINE
        WRITE RECORD-OUT FROM PRINTLINE2
        .

 POSITIONED-UPDATE-PLNAME .
*-------------------------
        EXEC SQL
             FETCH FIRST FROM CIII
             INTO :DCLTBPERSONS
        END-EXEC
        IF SQLCODE NOT = 0
           MOVE 'ERROR IN FETCH FIRST CIII FOR UPDATE PLNAME' TO
              SQL-MESSAGE
           PERFORM DISPLAY-ERROR-AND-STOP
        END-IF
        EXEC SQL
            UPDATE TBPERSONS
            SET PLNAME = 'POSITIONED UPDATE'
            WHERE CURRENT OF CIII
        END-EXEC
        IF SQLCODE NOT = 0
           MOVE 'ERROR IN POSITIONED UPDATE PLNAME' TO
              SQL-MESSAGE
           PERFORM DISPLAY-ERROR-AND-STOP
        END-IF
        MOVE 'POSITIONED-UPDATE-PLNAME WAS DONE' TO MESSAGE-LINE
        WRITE RECORD-OUT FROM PRINTLINE2
        WRITE RECORD-OUT FROM MESSAGE-LINE
        WRITE RECORD-OUT FROM PRINTLINE2
        .
```

```
 SEARCHED-UPDATE-PNO.
*-----------------------
     EXEC SQL
          UPDATE TBPERSONS
          SET PNO = 88
          WHERE PNO = 7
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN SEARCHED-UPDATE-PNO ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR-AND-STOP
     END-IF
     MOVE 'SEARCHED-UPDATE-PNO WAS DONE' TO MESSAGE-LINE
     WRITE RECORD-OUT FROM PRINTLINE2
     WRITE RECORD-OUT FROM MESSAGE-LINE
     WRITE RECORD-OUT FROM PRINTLINE2
     .

 POSITIONED-UPDATE-PNO .
*-------------------------
     EXEC SQL
          FETCH LAST FROM CIV
          INTO :DCLTBPERSONS
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN FETCH LAST CIV FOR UPDATE PNO' TO
           SQL-MESSAGE
        PERFORM DISPLAY-ERROR-AND-STOP
     END-IF
     EXEC SQL
         UPDATE TBPERSONS
         SET PNO = 99
         WHERE CURRENT OF CIV
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN POSITIONED UPDATE PNO' TO
           SQL-MESSAGE
        PERFORM DISPLAY-ERROR-AND-STOP
     END-IF
     MOVE 'POSITIONED-UPDATE-PNO  WAS DONE' TO MESSAGE-LINE
     WRITE RECORD-OUT FROM PRINTLINE2
     WRITE RECORD-OUT FROM MESSAGE-LINE
     WRITE RECORD-OUT FROM PRINTLINE2
     .

 FETCH-C0.
*------------
     EXEC SQL
          FETCH BEFORE FROM C0
     END-EXEC
     IF SQLCODE NOT = 0
        MOVE 'ERROR IN FETCH BEFORE CO   ' TO SQL-MESSAGE
        PERFORM DISPLAY-ERROR-AND-STOP
     END-IF
     EXEC SQL
          FETCH FROM C0
          INTO :DCLTBPERSONS
     END-EXEC
     MOVE 'RESULTS OF C0' TO MESSAGE-LINE
     WRITE RECORD-OUT FROM MESSAGE-LINE
     WRITE RECORD-OUT FROM PRINTLINE2
     PERFORM UNTIL SQLCODE NOT = 0 AND NOT = 222
       IF SQLCODE = 222
```

```
          MOVE '----- UPDATE HOLE IN C0 ----' TO PLNAME IN
                       DCLTBPERSONS
          MOVE 0 TO PNO IN DCLTBPERSONS
        END-IF
        PERFORM DISPLAY-PERSON
        EXEC SQL
                FETCH FROM C0
                INTO :DCLTBPERSONS
        END-EXEC
      END-PERFORM
      EVALUATE SQLCODE
            WHEN 100
              MOVE 'END OF LIST C0' TO SQL-MESSAGE
              PERFORM DISPLAY-ERROR
            WHEN OTHER
            MOVE 'ERROR OCCURRED DURING FETCH C0' TO SQL-MESSAGE E
              PERFORM DISPLAY-ERROR-AND-STOP
      END-EVALUATE
      .
 FETCH-CI.
*------------
      EXEC SQL
           FETCH BEFORE FROM CI
      END-EXEC
      IF SQLCODE NOT = 0
         MOVE 'ERROR IN FETCH BEFORE CI   ' TO SQL-MESSAGE
         PERFORM DISPLAY-ERROR-AND-STOP
      END-IF
      EXEC SQL
           FETCH FROM CI
           INTO :DCLTBPERSONS
      END-EXEC
      MOVE 'RESULTS OF CI' TO MESSAGE-LINE
      WRITE RECORD-OUT FROM MESSAGE-LINE
      WRITE RECORD-OUT FROM PRINTLINE2
      PERFORM UNTIL SQLCODE NOT = 0 AND NOT = 222
        IF SQLCODE = 222
           MOVE '----- UPDATE HOLE IN CI ----' TO PLNAME IN
                       DCLTBPERSONS
           MOVE 0 TO PNO IN DCLTBPERSONS
        END-IF
        PERFORM DISPLAY-PERSON
        EXEC SQL
                FETCH FROM CI
                INTO :DCLTBPERSONS
        END-EXEC
      END-PERFORM
      EVALUATE SQLCODE
            WHEN 100
              MOVE 'END OF LIST CI' TO SQL-MESSAGE
              PERFORM DISPLAY-ERROR
            WHEN OTHER
            MOVE 'ERROR OCCURRED DURING FETCH CI' TO SQL-MESSAGE E
              PERFORM DISPLAY-ERROR-AND-STOP
      END-EVALUATE
      .
 FETCH-CII.
*------------
      EXEC SQL
           FETCH BEFORE FROM CII
      END-EXEC
      IF SQLCODE NOT = 0
         MOVE 'ERROR IN FETCH BEFORE CII ' TO SQL-MESSAGE
```

```
           PERFORM DISPLAY-ERROR-AND-STOP
         END-IF
         EXEC SQL
              FETCH FROM CII
              INTO :DCLTBPERSONS
         END-EXEC
         MOVE 'RESULTS OF CII' TO MESSAGE-LINE
         WRITE RECORD-OUT FROM MESSAGE-LINE
         WRITE RECORD-OUT FROM PRINTLINE2
         PERFORM UNTIL SQLCODE NOT = 0 AND NOT = 222
           IF SQLCODE = 222
              MOVE '----- UPDATE HOLE IN CII----' TO PLNAME IN
                           DCLTBPERSONS
              MOVE 0 TO PNO IN DCLTBPERSONS
           END-IF
           PERFORM DISPLAY-PERSON
           EXEC SQL
                   FETCH FROM CII
                   INTO :DCLTBPERSONS
           END-EXEC
         END-PERFORM
         EVALUATE SQLCODE
               WHEN 100
                 MOVE 'END OF LIST CII' TO SQL-MESSAGE
                 PERFORM DISPLAY-ERROR
               WHEN OTHER
               MOVE 'ERROR OCCURRED DURING FETCH CI' TO SQL-MESSAGE E
                 PERFORM DISPLAY-ERROR-AND-STOP
         END-EVALUATE
         .
      FETCH-CIII.
     *------------
         EXEC SQL
              FETCH BEFORE FROM CIII
         END-EXEC
         IF SQLCODE NOT = 0
            MOVE 'ERROR IN FETCH BEFORE CIII ' TO SQL-MESSAGE
         END-IF
         EXEC SQL
              FETCH FROM CIII
              INTO :DCLTBPERSONS
         END-EXEC
         MOVE 'RESULTS OF CIII' TO MESSAGE-LINE
         WRITE RECORD-OUT FROM MESSAGE-LINE
         WRITE RECORD-OUT FROM PRINTLINE2
         PERFORM UNTIL SQLCODE NOT = 0 AND NOT = 222
           IF SQLCODE = 222
              MOVE '----- UPDATE HOLE IN CIII---' TO PLNAME IN
                           DCLTBPERSONS
              MOVE 0 TO PNO IN DCLTBPERSONS
           END-IF
           PERFORM DISPLAY-PERSON
           EXEC SQL
                   FETCH FROM CIII
                   INTO :DCLTBPERSONS
           END-EXEC
         END-PERFORM
         EVALUATE SQLCODE
               WHEN 100
                 MOVE 'END OF LIST CIII' TO SQL-MESSAGE
                 PERFORM DISPLAY-ERROR
               WHEN OTHER
              MOVE 'ERROR OCCURRED DURING FETCH CIII' TO SQL-MESSAGEE
```

```
            PERFORM DISPLAY-ERROR-AND-STOP
      END-EVALUATE
      .
 FETCH-CIV.
*------------
      EXEC SQL
           FETCH BEFORE FROM CIV
      END-EXEC
      IF SQLCODE NOT = 0
         MOVE 'ERROR IN FETCH BEFORE CIV  ' TO SQL-MESSAGE
         PERFORM DISPLAY-ERROR-AND-STOP
      END-IF
      EXEC SQL
           FETCH FROM CIV
           INTO :DCLTBPERSONS
      END-EXEC
      MOVE 'RESULTS OF CIV' TO MESSAGE-LINE
      WRITE RECORD-OUT FROM MESSAGE-LINE
      WRITE RECORD-OUT FROM PRINTLINE2
      PERFORM UNTIL SQLCODE NOT = 0 AND NOT = 222
        IF SQLCODE = 222
           MOVE '----- UPDATE HOLE IN CIV----' TO PLNAME IN
                       DCLTBPERSONS
           MOVE 0 TO PNO IN DCLTBPERSONS
        END-IF
        PERFORM DISPLAY-PERSON
        EXEC SQL
                FETCH FROM CIV
                INTO :DCLTBPERSONS
        END-EXEC
      END-PERFORM
      EVALUATE SQLCODE
            WHEN 100
              MOVE 'END OF LIST CIV' TO SQL-MESSAGE
              PERFORM DISPLAY-ERROR
            WHEN OTHER
            MOVE 'ERROR OCCURRED DURING FETCH CIV' TO SQL-MESSAGEE
              PERFORM DISPLAY-ERROR-AND-STOP
      END-EVALUATE
      .

 CLOSE-CURSORS-S.
*----------------
      EXEC SQL
         CLOSE C0
      END-EXEC
      IF SQLCODE NOT = 0
         MOVE 'ERROR IN CLOSE CURSOR C0' TO SQL-MESSAGE
         PERFORM DISPLAY-ERROR-AND-STOP
      END-IF
      EXEC SQL
         CLOSE CI
      END-EXEC
      IF SQLCODE NOT = 0
         MOVE 'ERROR IN CLOSE CURSOR CI' TO SQL-MESSAGE
         PERFORM DISPLAY-ERROR-AND-STOP
      END-IF
      EXEC SQL
         CLOSE CII
      END-EXEC
      IF SQLCODE NOT = 0
         MOVE 'ERROR IN CLOSE CURSOR CII' TO SQL-MESSAGE
         PERFORM DISPLAY-ERROR-AND-STOP
```

```
           END-IF
           EXEC SQL
              CLOSE CIII
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE 'ERROR IN CLOSE CURSOR CIII' TO SQL-MESSAGE
              PERFORM DISPLAY-ERROR-AND-STOP
           END-IF
           EXEC SQL
              CLOSE CIV
           END-EXEC
           IF SQLCODE NOT = 0
              MOVE 'ERROR IN CLOSE CURSOR CIV' TO SQL-MESSAGE
              PERFORM DISPLAY-ERROR-AND-STOP
           END-IF
           .
       CLOSE-FILE.
      *------------
           CLOSE FILE-OUT
           .

       DISPLAY-CURSOR-STATUS.
      *----------------------
           WRITE RECORD-OUT FROM MESSAGE-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           MOVE SQLWARN1   TO SQLW1
           MOVE SQLWARN4   TO SQLW4
           MOVE SQLWARN5   TO SQLW5
           WRITE RECORD-OUT FROM SQLWARNING1
           WRITE RECORD-OUT FROM SQLWARNING4
           WRITE RECORD-OUT FROM SQLWARNING5
           WRITE RECORD-OUT FROM PRINTLINE2
           .
       DISPLAY-PERSON.
      *--------------
           MOVE CORRESPONDING DCLTBPERSONS TO PRINTLINE3
           WRITE RECORD-OUT FROM PRINTLINE3
           .


       DISPLAY-ERROR.
      *--------------
           MOVE SQLCODE TO SQL-CODE
           WRITE RECORD-OUT FROM PRINTLINE2
           WRITE RECORD-OUT FROM ERROR-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           .
       DISPLAY-ERROR-AND-STOP.
      *--------------
           MOVE SQLCODE TO SQL-CODE
           WRITE RECORD-OUT FROM PRINTLINE2
           WRITE RECORD-OUT FROM ERROR-LINE
           WRITE RECORD-OUT FROM PRINTLINE2
           GOBACK
           .
```

OUTPUT OF THE PROGRAM

CHARACTERISTICS OF THE C0

```
SQLWARN1 =                    S
SQLWARN4 =                    S
SQLWARN5 =                    2
--------------------------------------------------------------------------------
CHARACTERISTICS OF THE CI
--------------------------------------------------------------------------------
SQLWARN1 =                    S
SQLWARN4 =                    S
SQLWARN5 =                    2
--------------------------------------------------------------------------------
CHARACTERISTICS OF THE CII
--------------------------------------------------------------------------------
SQLWARN1 =                    S
SQLWARN4 =                    S
SQLWARN5 =                    2
--------------------------------------------------------------------------------
CHARACTERISTICS OF THE CIII
--------------------------------------------------------------------------------
SQLWARN1 =                    S
SQLWARN4 =                    S
SQLWARN5 =                    4
--------------------------------------------------------------------------------
CHARACTERISTICS OF THE CIV
--------------------------------------------------------------------------------
SQLWARN1 =                    S
SQLWARN4 =                    S
SQLWARN5 =                    4
--------------------------------------------------------------------------------
RESULTS OF C0
--------------------------------------------------------------------------------
0001      SMITHS
0002      TAVERNIER
0003      DE KEYSER
0004      MAK
0005      NIEHOF
0006      VAN HEIJKOOP
0007      DE GROOT
0008      PEREZ
0009      LIVIER
0010      LUTZ
0011      LOOSE
0012      BENOIT
0013      BENOIT
0014      DETROIT
0015      SPENSER
0016      HANEGREEFS
0017      SCHUMACHER
0018      GELADE
0019      COPPIETERS
0020      DE WINDT
0021      DE SCHRIJVER
0022      HENDERSON
0023      DELANGHE
0024      VAN DE BROECK
0025      MEURIS
0026      HEBBELYNCK
0027      DE GRAAF
0028      TYTGAT
0029      DEVISSER
0030      DE WILDE
0031      DEHEM
0032      BUENK
0033      PIELAGE
```

```
0034      DE BRUYN
0035      DE SMET
0036      ADAMSON
0037      GOYENS
0038      GERRIES
0039      DE CORTE
0040      PARKER
------------------------------------------------------------------------
SQLCODE =           +0100     END OF LIST C0
------------------------------------------------------------------------
RESULTS OF CI
------------------------------------------------------------------------
0003      DE KEYSER
0007      DE GROOT
0014      DETROIT
0018      GELADE
0020      DE WINDT
0021      DE SCHRIJVER
0022      HENDERSON
0023      DELANGHE
0024      VAN DE BROECK
0027      DE GRAAF
0029      DEVISSER
0030      DE WILDE
0031      DEHEM
0034      DE BRUYN
0035      DE SMET
0039      DE CORTE
------------------------------------------------------------------------
SQLCODE =           +0100     END OF LIST CI
------------------------------------------------------------------------
RESULTS OF CII
------------------------------------------------------------------------
0001      SMITHS
0002      TAVERNIER
0003      DE KEYSER
0004      MAK
0005      NIEHOF
0006      VAN HEIJKOOP
0007      DE GROOT
0008      PEREZ
0009      LIVIER
0010      LUTZ
0011      LOOSE
0012      BENOIT
0013      BENOIT
0014      DETROIT
0015      SPENSER
0016      HANEGREEFS
0017      SCHUMACHER
0018      GELADE
0019      COPPIETERS
0020      DE WINDT
0021      DE SCHRIJVER
0022      HENDERSON
0023      DELANGHE
0024      VAN DE BROECK
0025      MEURIS
0026      HEBBELYNCK
0027      DE GRAAF
0028      TYTGAT
0029      DEVISSER
0030      DE WILDE
```

```
0031      DEHEM
0032      BUENK
0033      PIELAGE
0034      DE BRUYN
0035      DE SMET
0036      ADAMSON
0037      GOYENS
0038      GERRIES
0039      DE CORTE
------------------------------------------------------------------------
SQLCODE =           +0100     END OF LIST CII
------------------------------------------------------------------------
RESULTS OF CIII
------------------------------------------------------------------------
0003      DE KEYSER
0007      DE GROOT
0014      DETROIT
0018      GELADE
0020      DE WINDT
0021      DE SCHRIJVER
0022      HENDERSON
0023      DELANGHE
0024      VAN DE BROECK
0027      DE GRAAF
0029      DEVISSER
0030      DE WILDE
0031      DEHEM
0034      DE BRUYN
0035      DE SMET
0039      DE CORTE
------------------------------------------------------------------------
SQLCODE =           +0100     END OF LIST CIII
------------------------------------------------------------------------
RESULTS OF CIV
------------------------------------------------------------------------
0001      SMITHS
0002      TAVERNIER
0003      DE KEYSER
0004      MAK
0005      NIEHOF
0006      VAN HEIJKOOP
0007      DE GROOT
0008      PEREZ
0009      LIVIER
0010      LUTZ
0011      LOOSE
0012      BENOIT
0013      BENOIT
0014      DETROIT
0015      SPENSER
0016      HANEGREEFS
0017      SCHUMACHER
0018      GELADE
0019      COPPIETERS
0020      DE WINDT
0021      DE SCHRIJVER
0022      HENDERSON
0023      DELANGHE
0024      VAN DE BROECK
0025      MEURIS
0026      HEBBELYNCK
0027      DE GRAAF
0028      TYTGAT
```

```
0029      DEVISSER
0030      DE WILDE
0031      DEHEM
0032      BUENK
0033      PIELAGE
0034      DE BRUYN
0035      DE SMET
0036      ADAMSON
0037      GOYENS
0038      GERRIES
0039      DE CORTE
------------------------------------------------------------------------
SQLCODE =           +0100     END OF LIST CIV
------------------------------------------------------------------------
------------------------------------------------------------------------
SEARCHED-UPDATE-PLNAME WAS DONE
------------------------------------------------------------------------
------------------------------------------------------------------------
POSITIONED-UPDATE-PLNAME WAS DONE
------------------------------------------------------------------------
------------------------------------------------------------------------
SEARCHED-UPDATE-PNO WAS DONE
------------------------------------------------------------------------
------------------------------------------------------------------------
POSITIONED-UPDATE-PNO  WAS DONE
------------------------------------------------------------------------
RESULTS OF C0
------------------------------------------------------------------------
0001      SMITHS
0002      TAVERNIER
0003      POSITIONED UPDATE
0004      MAK
0005      NIEHOF
0006      VAN HEIJKOOP
0088      DE GROOT
0008      PEREZ
0009      LIVIER
0010      LUTZ
0011      LOOSE
0012      BENOIT
0013      BENOIT
0014      DETROIT
0015      SPENSER
0016      HANEGREEFS
0017      SCHUMACHER
0018      GELADE
0019      COPPIETERS
0020      DE WINDT
0021      DE SCHRIJVER
0022      HENDERSON
0023      DELANGHE
0024      VAN DE BROECK
0025      MEURIS
0026      HEBBELYNCK
0027      SEARCHED UPDATE
0028      TYTGAT
0029      DEVISSER
0030      DE WILDE
0031      DEHEM
0032      BUENK
0033      PIELAGE
0034      DE BRUYN
0035      DE SMET
```

```
0036      ADAMSON
0037      GOYENS
0038      GERRIES
0099      DE CORTE
0040      PARKER
------------------------------------------------------------------------
SQLCODE =           +0100     END OF LIST C0
------------------------------------------------------------------------
RESULTS OF CI
------------------------------------------------------------------------
0000      ----- UPDATE HOLE IN CI ----
0088      DE GROOT
0014      DETROIT
0018      GELADE
0020      DE WINDT
0021      DE SCHRIJVER
0022      HENDERSON
0023      DELANGHE
0024      VAN DE BROECK
0000      ----- UPDATE HOLE IN CI ----
0029      DEVISSER
0030      DE WILDE
0031      DEHEM
0034      DE BRUYN
0035      DE SMET
0099      DE CORTE
------------------------------------------------------------------------
SQLCODE =           +0100     END OF LIST CI
------------------------------------------------------------------------
RESULTS OF CII
------------------------------------------------------------------------
0001      SMITHS
0002      TAVERNIER
0003      POSITIONED UPDATE
0004      MAK
0005      NIEHOF
0006      VAN HEIJKOOP
0000      ----- UPDATE HOLE IN CII----
0008      PEREZ
0009      LIVIER
0010      LUTZ
0011      LOOSE
0012      BENOIT
0013      BENOIT
0014      DETROIT
0015      SPENSER
0016      HANEGREEFS
0017      SCHUMACHER
0018      GELADE
0019      COPPIETERS
0020      DE WINDT
0021      DE SCHRIJVER
0022      HENDERSON
0023      DELANGHE
0024      VAN DE BROECK
0025      MEURIS
0026      HEBBELYNCK
0027      SEARCHED UPDATE
0028      TYTGAT
0029      DEVISSER
0030      DE WILDE
0031      DEHEM
0032      BUENK
```

```
0033      PIELAGE
0034      DE BRUYN
0035      DE SMET
0036      ADAMSON
0037      GOYENS
0038      GERRIES
0000      ----- UPDATE HOLE IN CII----
------------------------------------------------------------------------
SQLCODE =           +0100     END OF LIST CII
------------------------------------------------------------------------
RESULTS OF CIII
------------------------------------------------------------------------
0000      ----- UPDATE HOLE IN CIII---
0088      DE GROOT
0014      DETROIT
0018      GELADE
0020      DE WINDT
0021      DE SCHRIJVER
0022      HENDERSON
0023      DELANGHE
0024      VAN DE BROECK
0000      ----- UPDATE HOLE IN CIII---
0029      DEVISSER
0030      DE WILDE
0031      DEHEM
0034      DE BRUYN
0035      DE SMET
0099      DE CORTE
------------------------------------------------------------------------
SQLCODE =           +0100     END OF LIST CIII
------------------------------------------------------------------------
RESULTS OF CIV
------------------------------------------------------------------------
0001      SMITHS
0002      TAVERNIER
0003      POSITIONED UPDATE
0004      MAK
0005      NIEHOF
0006      VAN HEIJKOOP
0000      ----- UPDATE HOLE IN CIV----
0008      PEREZ
0009      LIVIER
0010      LUTZ
0011      LOOSE
0012      BENOIT
0013      BENOIT
0014      DETROIT
0015      SPENSER
0016      HANEGREEFS
0017      SCHUMACHER
0018      GELADE
0019      COPPIETERS
0020      DE WINDT
0021      DE SCHRIJVER
0022      HENDERSON
0023      DELANGHE
0024      VAN DE BROECK
0025      MEURIS
0026      HEBBELYNCK
0027      SEARCHED UPDATE
0028      TYTGAT
0029      DEVISSER
0030      DE WILDE
```

```
0031      DEHEM
0032      BUENK
0033      PIELAGE
0034      DE BRUYN
0035      DE SMET
0036      ADAMSON
0037      GOYENS
0038      GERRIES
0000      ----- UPDATE HOLE IN CIV----
------------------------------------------------------------------------
SQLCODE =           +0100     END OF LIST CIV
------------------------------------------------------------------------
```