ABIS NIEUWSBRIEF VOOR DB2 PROFESSIONALS

JAARGANG 10 - NUMMER 2, SEPTEMBER 2016

OPEN CURSOR

In dit nummer besteden we speciaal aandacht aan "Big data":
Een modewoord, uiteraard, al sinds enkele jaren, en een term die vele ladingen dekt.
Als data-specialisten kunnen we uiteraard door de façade prikken. DB2 beheert al decennia lang volumineuze data. En laat toe, zeker sinds de introductie van PureXML in versie 9, om complexe datastructuren op te slaan. En SQL laat toe om op een leesbare manier analytische queries los te laten op die data.
Maar door de explosie aan beschikbare data en het inzicht dat die data geëxploiteerd kan worden voor steeds slimmere "Business Intelligence" (BI) moeten we onze blik verruimen. DB2 is één van de vele spelers in de Big Data context van vandaag. Maar een zeer belangrijke speler!

Veel leesgenot,
Het ABIS DB2-team.

IN DIT NUMMER:

Een themanummer Big Data, uiteraard met DB2-focus:

- *Als de geschiedenis...zich herhaalt!* Inderdaad, de nieuwe ideeën omtrent NoSQL, data-analytics, volume, variabiliteit, ... zijn niet zo nieuw. Leren uit het verleden, om beter voorbereid te zijn op de toekomst!
- Het tweede hoofdartikel, *'Nieuwe' SQL: Window Functions*, biedt u een klare kijk op een belangrijk SQL-tool voor (big) data analytics. Dit biedt nieuwe mogelijkheden voor DB2 als centrale data-store, ook in een modern BI landschap.
- En zoals steeds ook weer een "Dossier 11", met twee nieuwe mogelijkheden van DB2 11 for z/OS: de mediaan-functie, en het gebruik van de FLWOR-syntax in PureXML.

2

CLOSE CURSOR

In de volgende nummers van Exploring DB2 nemen we het performance-aspect van de optimizer onder de loep: query rewrites, nieuwe explain-tabellen, virtuele indexen, predicate selectivity, ...

En verder houden we u uiteraard op de hoogte van belangrijke nieuwigheden, zoals de nieuwe versie 12 van DB2 voor z/OS.

Als de geschiedenis... zich herhaalt!

Kris Van Thillo (ABIS)

Plots herinnerde ik me een mooi artikel van een Oracle specialist en evangelist, enige tijd geleden gepubliceerd. Iemand die in de Oracle wereld bekend staat om zijn technische kennis; om de eenvoud en de duidelijkheid waarmee hij vaak complexe materie uitlegt. Zijn presentaties op conferenties trekken steevast volle zalen.

Oracle Magazine, September 2015

Ik citeer een paragraaf.

Yes, in the 1980s I was working on three-tier application server programs in virtual machines using NoSQL databases! It just took three decades for technology to come fully back around again. The names have changed, the languages have changed, and the hardware has really changed, but many of the fundamentals that were true back then are still very true today. (TECHNOLOGY: Ask Tom, Time-out and Thanks, Tom Kyte, Oracle Corporation, 2015).

Over *virtual machines in the 1980s*.

Reeds in de jaren '60 dacht IBM na over wat 'multi-programming' genoemd werd: hoe ontwikkelen op multi-user systemen met dedicated randapparatuur die niet voortdurend exclusief ter beschikking staat van diezelfde ontwikkelaar? Het idee van het 'delen' -*sharing* - van resources (o.a. op basis van tijd slots) is op dat moment ontstaan. Onderliggend aan dit *sharing* idee: stel virtuele machines ter beschikking die eigenlijk niet veel meer zijn dan een kopie van de onderliggende 'reële' computer-architectuur. Een *monitor*, *control program*, *supervisor*, ... zou dan wel instaan voor het beheer van deze omgevingen.

Baanbrekend werk werd verricht door IBM op IBM/360/370/390 familie van computers. VM - nu z/VM - staat vandaag bekend als één van de voorlopers van de hedendaagse virtualisatietechnologie. De IBM VM/370 wordt algemeen beschouwd als de 'bron', de origine van de zogenaamde *hypervisor* technologie. Software technologie die het mogelijk maakt om op een set van fysieke resources, naast mekaar en concurrent, verschillende operating systemen - 'gast' operating systemen - aan te bieden. Op niet-mainframe platformen is bijvoorbeeld VMWare vandaag een vaak geciteerde hypervisor. Er is trouwens op z-Systems een nieuwe speler opgedoken: KVM for IBM z-Systems.

Merk op dat hypervisors kieskeurig zijn wat betreft de 'gast' operating systemen die ze ondersteunen: een z/VM hypervisor ondersteunt bijvoorbeeld geen Windows-based OS als gast.

Over LPARs hebben we het in deze context gemakshalve even niet...

Meer info in hoofdlijnen over z/OS en virtualisatie op z/OS architecturen? Contacteer ABIS!

Over *NoSQL in the 1980s*.

Het heeft ongetwijfeld met leeftijd te maken; mogelijk ook met de ontwikkelplatformen waar we in een IBM-context mee in contact kwamen. De term NoSQL bestond nog niet; maar we hadden in die tijd 'allemaal' ervaring met DB2 (en dus met SQL en relationele systemen), IMS (en dus met hiërarchische systemen), VSAM data set programmeren (o.a. key based systemen). Met COBOL of PL/I als programmeertaal - of Assembler? En zelfs op niet-IBM mainframe platformen waren deze technologieën *dominant*.

Het was een tijd waarin de keuze voor een systeem en architectuur werd bepaald door eigenschappen van de te implementeren applicatie; alsook door de performance eigenschappen inherent aan deze systemen. Werd er niet - wordt er niet - beweerd dat IMS-applicaties altijd efficiënter zullen zijn dan DB2-applicaties? DB2-systemen (en bij uitbreiding relationele systemen) hebben de bovenhand gehaald om redenen die in eerste instantie niets met applicaties *an sich* te maken hadden (flexibiliteit in design van systemen, flexibiliteit bij ontwikkeling, ...). En dankzij de explosie van de mogelijkheden geboden door computer hardware werd het credo bij de keuze van opslagsystemen zelfs vaak 'Relationeel, behalve als'.

En dus gingen we met z'n allen 'alles' in relationele systemen opslaan: gestructureerde en niet-gestructureerde data, documenten, beeld, ... zelfs geluid zou mogelijk moeten zijn, alhoewel ik deze laatste nog nooit zelf ben tegen gekomen.

DB2, IMS, VSAM, Assembler, ... we hebben nog steeds de nodige kennis in huis!

En nu - *full circle*?

De hardwareëvolutie is wat ze is: steeds krachtiger machines, al dan niet gevirtualiseerd, gebruik makende van proprietary dan wel - steeds vaker - commodity hardware; en ook de hoeveelheid data die we wensen te verwerken is explosief toegenomen. Maar ook en vooral de wijze waarmee we met deze data omgaan, is gewijzigd: niet meer initieel om op te slaan, maar vooral om te analyseren, conclusies te formuleren, 'business intelligence'.

De 'massaliteit' van wat we wensen te verwerken tegen deze achtergrond, heeft het belang van het niet-relationele denken een duidelijk nieuw elan gegeven.... *AND many of the fundamentals that were true back then are still very true today.* En daar weten wij nu net alles van!

Big Data & Data Science: How to become a Data Scientist?
Zie *http://www.abis.be/resources/mailings/DataScience3.pdf*

‘Nieuwe’ SQL: Window Functions

Arnout Veugelen (ABIS)

SQL is een levende taal. Regelmatig wordt de standaard aangevuld met nieuwe mogelijkheden. Deze uitbreidingen druppelen echter niet altijd door tot bij de gebruikers, die vaak kunnen bogen op een stevige, maar niet noodzakelijk zeer actuele, kennis van SQL. Onder het motto ‘onbekend is onbemind’, begonnen we daarom in ons vorige nummer aan een mini-reeks over nuttige en relatief ‘nieuwe’ SQL. We hadden het over super-groups, waarmee we verschillende GROUP BY-criteria kunnen combineren in een query.

In deze editie bespreken we window-functies, misschien wel de krachtigste toevoeging aan de standaard query-syntax van de laatste decennia. Ze behoren sinds SQL-2003 tot de ANSI-standaard (uitgebreid in SQL-2008), en zijn ondertussen te gebruiken in de meeste relationele databases. DB2 for z/OS begon met de implementatie in versie 9 en bouwde verder in versie 10. DB2 for LUW begon er al in versie 8 aan en heeft in versie 11.1 een zeer uitgebreid aanbod.

Mede door de naar SQL-normen vrij pittige syntax is de populariteit van window functions beperkt, maar voor iedereen die aan (big) data-analyse wil doen met SQL, loont het absoluut de moeite om ze onder de knie te krijgen!

Voor de voorbeelden in dit artikel maken we gebruik van volgende tabel ‘Orders’:

```
CREATE TABLE Orders ( Ordernr     SMALLINT NOT NULL PRIMARY KEY,
                      Orderdate   DATE,
                      Salesperson VARCHAR(20),
                      Prodid      CHAR(4),
                      Qty         SMALLINT,
                      Amount      DECIMAL(10,2)
                    );
```

Ordernr	Orderdate	Salesperson	Prodid	Qty	Amount
1	2014-05-06	Ann	1234	16	150
2	2014-07-08	Ann	5678	5	25
3	2015-08-16	Bob	1234	4	40
4	2015-10-20	Bob	1234	1	10
5	2015-10-22	Ann	9000	10	900
6	2015-10-24	Ann	9000	5	500
7	2016-01-10	Bob	1234	4	40

What’s in a name?

Sommige producenten gebruiken de term ‘Window Functions’, Oracle spreekt van ‘Analytical Functions’, DB2 heeft het over ‘OLAP’: Online Analytical Processing. Al deze benamingen houden steek. Deze functies analyseren de data binnen een lokaal beschreven venster,

we voeren de analyse lijn per lijn uit, en de lijnen behouden hun individualiteit. OLAP combineert zo analytische functionaliteit (het bepalen van rangordes, gemiddeldes etc.) met een grote flexibiliteit.

De OVER()-syntax

Nog een derde naam: over-functies; window-functies worden namelijk steeds vergezeld door het woord OVER, waarna beschreven wordt in welk datavenster de functie moet uitgevoerd worden. Als u even doorbladert naar het einde van dit artikel, kan u de volledige OVER()-syntax bekijken. Dat lijkt een hele mond vol, en op het eerste zicht niet vanzelfsprekend, maar als we de verschillende onderdelen één voor één bekijken, valt het wel mee. De OVER()-syntax valt uiteen in drie delen, die we verderop met volgende namen zullen bespreken:

```
function([ arguments ])
       OVER ([ partitition clause ] [ order clause [ windowing clause ] ])
```

Partition clause

Het eerste deel van de OVER()-syntax is de partition clause. Na PARTITION BY wordt een soort lokale grouping (een partitie) omschreven waarop de functie berekend wordt. Wanneer we PARTITION BY niet expliciet vermelden, wordt de functie over de hele resultaatstabel uitgevoerd.

In dit voorbeeld vermelden we bij elke verkoop hoeveel een verkoop van hetzelfde product gemiddeld opbrengt:

```
SELECT  Ordernr, Orderdate, Prodid, Amount,
        AVG(Amount) OVER(PARTITION BY Prodid) AS Avg_amount
FROM    Orders
ORDER BY Ordernr
```

Ordernr	Orderdate	Prodid	Amount	Avg_amount
1	2014-05-06	1234	150	60
2	2014-07-08	5678	25	25
3	2015-08-16	1234	40	60
4	2015-10-20	1234	10	60
5	2015-10-22	9000	900	700
6	2015-10-24	9000	500	700
7	2016-01-10	1234	40	60

Dankzij PARTITION BY wordt het dus mogelijk om het resultaat van de klassieke kolomfuncties (ook wel aggregate functions genoemd) zoals COUNT(), AVG(), SUM() ... te gaan combineren met individuele (niet-gegroepeerde) waarden. Op die manier worden deze functies een heel stuk flexibeler.

Order clause

Het tweede deel van de OVER()-clausule is ORDER BY, waar we een lokale rangorde binnen de gekozen partitie bepalen.

Hieronder gebruiken we de window-functie RANK(), een zogenaamde rankingfunctie, om voor elke verkoper de bestellingen volgens opbrengst te rangschikken:

```
SELECT Ordernr, Orderdate, Salesperson, Amount,
       RANK() OVER(PARTITION BY Salesperson ORDER BY Amount DESC) As Ranking
FROM   Orders
Order By Salesperson, Ranking
```

Salesperson	Ordernr	Orderdate	Amount	Ranking
Ann	5	2015-10-22	900	1
Ann	6	2015-10-24	500	2
Ann	1	2014-05-06	150	3
Ann	2	2014-07-08	25	4
Bob	3	2015-08-16	40	1
Bob	7	2016-01-10	40	1
Bob	4	2015-10-20	10	3

RANK() geeft elke rij in de lokale partitie een ranking; bij ex-aequo's krijgen rijen hetzelfde volgnummer, en wordt het volgende volgnummer overgeslagen.

De andere rankingfuncties in DB2 for z/OS zijn:

- DENSE_RANK(): bij ex-aequo krijgen rijen hetzelfde volgnummer, maar er worden geen volgnummers overgeslagen.
- ROW_NUMBER(): alle rijen krijgen sowieso een uniek volgnummer.

Andere producten (inclusief DB2 11.1 for LUW) gaan een stukje verder, met bijvoorbeeld NTILE(), een functie voor het berekenen van percentielen en dergelijke.

De order clause van window-functies bepaalt enkel de rangorde die voor het uitrekenen van die functie gebruikt wordt, en staat dus uiteraard volledig los van de gewone ORDER BY aan het einde van de query, die de volgorde in de resultaatstabel bepaalt.

LAG() en LEAD()

Het volgende voorbeeld illustreert de mogelijkheden van de order clause via de functie LAG(), waarmee we een waarde uit de vorige record in de rangschikking kunnen ophalen, in dit geval de vorige datum waarop hetzelfde product verkocht werd.

```
SELECT  Ordernr, Orderdate, Prodid, Qty, Amount,
       LAG(Orderdate) OVER(PARTITION BY Prodid ORDER BY Orderdate) AS Prev_sale
FROM     Orders
ORDER BY Ordernr
```

Ordernr	Orderdate	Salesperson	Prodid	Qty	Amount	Prev_sale
1	2014-05-06	Ann	1234	16	150	(null)
2	2014-07-08	Ann	5678	5	25	(null)
3	2015-08-16	Bob	1234	4	40	2014-05-06
4	2015-10-20	Bob	1234	1	10	2015-08-16
5	2015-10-22	Ann	9000	10	900	(null)
6	2015-10-24	Ann	9000	5	500	2015-10-22
7	2016-01-10	Bob	1234	4	40	2015-10-20

Naast LAG() bestaat ook LEAD(), om de volgende rij in de rangschikking aan te spreken. LEAD() EN LAG() zijn beschikbaar in DB2 for LUW, maar (nog) niet in DB2 for z/OS.

Windowing clause

ORDER BY kan in veel gevallen nog aangevuld worden met een windowing clause, die de omvang van het venster bepaalt. Dat kan met behulp van het keyword ROWS, waarna we, op basis van de zopas gemaakte partitierangschikking, aanduiden hoeveel rijen voor (PRECEDING) en na (FOLLOWING) de te verwerken rij meegenomen moeten worden in de analyse. UNBOUNDED PRECEDING betekent vanaf het begin van de partitierangschikking, UNBOUNDED FOLLOWING tot het einde van de partitierangschikking. We kunnen zo bijvoorbeeld naast elke verkoop een 'running total' berekenen van alle verkopen van het jaar tot dan toe:

```
SELECT  Ordernr, Orderdate, Prodid, Qty, Amount,
        SUM(Amount) OVER(PARTITION BY EXTRACT(YEAR FROM Orderdate)
                         ORDER BY Orderdate
                         ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
                        ) AS Running_total
FROM Orders
ORDER BY Ordernr
```

Ordernr	Orderdate	Salesperson	Prodid	Qty	Amount	Running_total
1	2014-05-06	Ann	1234	16	150	150
2	2014-07-08	Ann	5678	5	25	175
3	2015-08-16	Bob	1234	4	40	40
4	2015-10-20	Bob	1234	1	10	50
5	2015-10-22	Ann	9000	10	900	950
6	2015-10-24	Ann	9000	5	500	1450
7	2016-01-10	Bob	1234	4	40	40

De windowing clause kan ook opgesteld worden met het keyword RANGE in plaats van ROWS, dat zich dan baseert op het aantal verschillende waarden (zoals een SELECT DISTINCT dat doet).

Let op: de defaults van deze laatste clausule kunnen verraderlijk zijn, het is dus in de meeste gevallen aangewezen om de windowing clause expliciet te specificeren.

Alles samen

Samenvattend hieronder nog eens de volledige OLAP-syntax:

```
window_function( <argument list> )
      OVER
       ( [ PARTITION BY partition-expression1,
                        partition-expression2,
                        ...
         ]
         [ORDER BY sort-expression1 [ASC | DESC] [NULLS FIRST | LAST],
                   sort-expression2,
                   ...
             [ ROWS|RANGE [BETWEEN]
                          [UNBOUNDED PRECEDING | n PRECEDING | CURRENT ROW)]
                          [AND]
                          [UNBOUNDED FOLLOWING | n FOLLOWING | CURRENT ROW)]
             ]
         ]
       )
```

Nog steeds een grote brok, maar ondertussen hopelijk toch al een stukje duidelijker ... Dit artikel is echter te kort om alle nuances van window functions volledig te bespreken en het vergt enige oefening en ervaring om SQL OLAP volledig onder de knie te krijgen. Ervaring die u bijvoorbeeld kan komen opdoen tijdens onze cursus *SQL voor BI rapportering en analyse!*

Window Functions documentatie

- DB2 for z/OS:
  http://www.ibm.com/support/knowledgecenter/SSEPEK_11.0.0/sqlref/src/tpc/db2z_olapspecification.html
- DB2 for LUW:
  http://www.ibm.com/support/knowledgecenter/SSEPGG_11.1.0/com.ibm.db2.luw.sql.ref.doc/doc/r0023461.html
- Oracle:
  https://docs.oracle.com/cd/E11882_01/server.112/e41084/functions004.htm#SQLRF06174
- SQLServer:
  https://msdn.microsoft.com/en-us/library/ms189461.aspx

DOSSIER 11

"Big data" analytics op z/OS met DB2 versie 11

In dit themanummer kort iets over twee nieuwe mogelijkheden in de analytics-hoek:

De MEDIAN-functie

Zoals gekend ondersteunt DB2 sinds jaar en dag de vijf standaard groeperingsfuncties COUNT, SUM, AVG, MIN en MAX. Deze vijf functies zijn sortering-neutraal, en dus efficiënt te implementeren door één enkele traversal doorheen de data.

De mediaan van een groep getallen is, net zoals het gemiddelde (AVG), een goede "centrummaat", een "karakteristieke vertegenwoordiger" van de groep dus.
De definitie van mediaan is zeer eenvoudig: de helft van de getallen uit de groep zijn kleiner dan de mediaan, de andere helft zijn groter.

In tegenstelling tot som en gemiddelde is het echter niet mogelijk om de mediaan te berekenen door één enkele data-traversal: de data moet eerst gesorteerd worden; de mediaan is dan de "middelste" waarde van die gesorteerde lijst.

Performance-gewijze is sorteren voor DB2 een "dure" operatie, zowel qua geheugengebruik als qua rekentijd (minstens O(n log n)). Anderzijds is sorteren goed te *parallelliseren*. Mits voldoende processoren wordt de doorlooptijd dan zelfs sublineair!

Dat verklaart waarom DB2 11 for z/OS (voor het eerst!) de beschikbaarheid van een functie laat afhangen van de aanwezigheid van een "massively parallel external processor", met name *IDAA*. U leest het goed: de MEDIAN-functie werkt dus enkel indien u een IDAA aan DB2 hebt gekoppeld.

XML-expressies met FLWOR-notatie

Data warehouses vormen een relationele database die geoptimaliseerd (en geprepareerd) is voor het stellen van analytisch georiënteerde vragen.

Een totaal andere invalshoek voor het opslaan van complexe gestructureerde data op een overzichtelijke en (hopelijk) efficiënt te ondervragen manier, is XML. Sinds versie 9 ondersteunt DB2 de geïntegreerde opslag van XML naast (of eigenlijk: in) relationele tabellen.

SQL is niet geschikt om (complexe) XML te ondervragen. DB2 voorziet een aantal "brug"-functies zoals XMLEXISTS en XMLQUERY waarbinnen de eigenlijke zoekopdracht moet geformuleerd worden met behulp van XPath-expressies.

XML heeft echter een eigen SELECT-FROM-WHERE-ORDERBY tegenhanger, en die wordt nu dus ook ondersteund door de genoemde brug-functies, sinds DB2 11. De XML-tegenhanger gebruikt echter de keywords for-let-where-orderby-return, afgekort tot FLWOR. (Klinkt als "flower".) Een voorbeeldje, op basis van een eenvoudig XML-bestand (zie Example 2 van http://www.w3schools.com/xml/):

```
for $food in /breakfast_menu/food
let $name := $food/name/text()
let $desc := $food/description/text()
where $food/calories < 500
order by $food/price descending
return <my_menu>{$desc}</my_menu>
```

Peter Vanroose

CURSUSPLANNING, OKT - DEC 2016

Basiskennis SQL & relationele databases	810 EUR	27.10(L), 03.11(W), 28.11(L)
DB2 for z/OS basiscursus	1365 EUR	14.12(W)
DB2 for LUW basiscursus	1365 EUR	14.12(W)
SQL-QMF voor eindgebruikers		op aanvraag
SQL workshop	860 EUR	17.10(L), 08.12(W), 19.12(L)
SQL voor gevorderden	480 EUR	14.11(L), 22.12(W)
SQL voor BI rapportering & analyse	910 EUR	24.10(L), 19.12(W)
Software-ontwikkeling met SQL PL	960 EUR	21.12(L)
DB2 triggers, stored procedures, en User-Defined Functions	480 EUR	13.12(W)
DB2 for z/OS: programmeren voor gevorderden	960 EUR	21.12(L)
DB2 for z/OS: SQL performance	1440 EUR	14.12(L)
XML in DB2		op aanvraag
DB2 for z/OS: database administratie	2020 EUR	05.12(L)
DB2 for z/OS: installation & migration	1115 EUR	29.09(UK), 10.10(UK), 01.12(UK)
DB2 for z/OS: data recovery	1115 EUR	06.12(UK)
DB2 for z/OS: systems performance and tuning	875 GBP	20.10(UK), 15.12(UK)
DB2 LUW DBA - Kernvaardigheden	1920 EUR	29.11(W)
DB2 LUW DBA - Configure, monitor&tune		op aanvraag
Database applicatieprogrammering met JDBC	480 EUR	09.11(L)
DB2 10 for z/OS: changes & new features		op aanvraag
DB2 10 for LUW: new features		op aanvraag
DB2 11 for z/OS: changes & new features	510 EUR	op aanvraag
Data warehouse concepten	480 EUR	31.10(W)
Big Data concepten	480 EUR	21.10(L), 01.11(W)
Big Data in de praktijk met Hadoop	960 EUR	24.10(W), 24.11(L)
Big Data in de praktijk met Spark	960 EUR	14.10(L), 07.11(W), 19.12(L)
MongoDB	1010 EUR	09.11(L), 06.12(W)

Plaats: L = Leuven, W = Woerden, UK = High Wycombe (bij Londen);
alle cursussen ook op aanvraag en/of op uw locatie;
Voor details en andere cursussen, zie `http://www.abis.be/html/nlTraining.html`
Pour détails et autres cours, voir `http://www.abis.be/html/frTraining.html`
For details and other courses, see `http://www.abis.be/html/enTraining.html`