ABIS NIEUWSBRIEF VOOR DB2 PROFESSIONALS

JAARGANG 10 - NUMMER 3, FEBRUARI 2017

EXPLORING DB2

OPEN CURSOR

Velen van jullie zitten ondertussen wellicht op DB2 z/OS versie 11. Daarom in dit nummer twee uitgebreide bijdragen over toch wel nuttige DB2 11 functionaliteiten. Wellicht ook nuttige informatie voor wie nog op DB2 10 werkt, want extra argumenten voor een spoedige migratie!

Los van de DB2-versie wordt het steeds belangrijker om de DB2-data op z/OS toegankelijk te maken voor steeds meer zgn. "distributed" applicaties. Deze evolutie is natuurlijk al langer aan de gang. Kennis van zowel DB2 for z/OS als DB2 for LUW is daarbij geen overbodige luxe. In die context is het belangrijk om de evolutie van de "brug" tussen beide in het oog te houden, nl. DB2 Connect. Hoog tijd dus voor een overzicht van de huidige stand van zaken.

Veel leesgenot,
Het ABIS DB2-team.

IN DIT NUMMER:

3

CLOSE CURSOR

In de volgende nummers van Exploring DB2 wordt de reeks over de optimizer verdergezet met o.a. virtuele indexen, query rewrites en predicate pushdown.

Ook de nieuwe versie 12 van DB2 for z/OS zal aan bod komen. Over welke onderwerpen wilt u meer lezen? Meld het ons: training@abis.be

Where has the DB2 Connect Gateway gone?

Koen De Backer (ABIS)

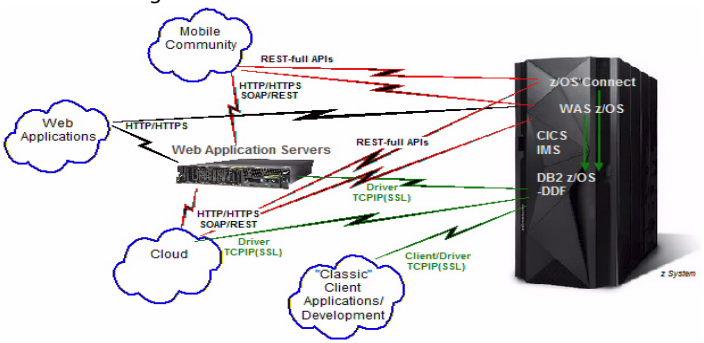
Abstract

The path taken by application requests to reach the DB2 data store on z/OS for many years pivoted around the, "classic" DB2 client and the DB2 Connect gateway. These pieces of software, for DB2, bridged the gap between the "distributed" (DB2 for LUW) environment and the host (DB2 for z/OS, IBM i). The type of application ("legacy" client, web, mobile or cloud) will for sure be a criterion to decide on the connection topology.

What is the current state of the connection landscape for applications that have a DB2 backend as a target? What are the options, if any?

DB2 connection topologies, the big picture.

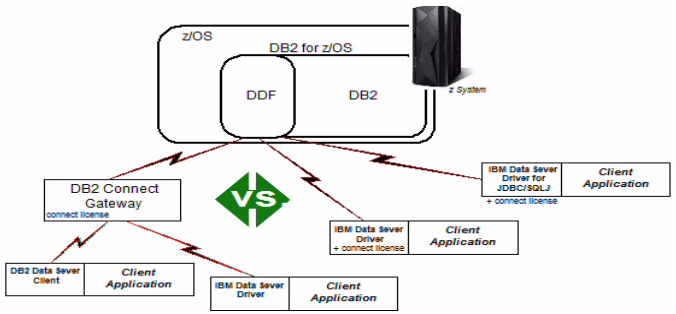
What is missing?



DB2 Connect or IBM Data Server driver?

What we are looking for when a client DB2 application wants to connect to DB2 for z/OS is a piece of software that will translate the application's data interface protocol to DRDA, which is what DB2 for z/OS's DDFs language is. This is where DB2 Connect and the IBM Data Server driver come in

Currently there is a strong preference for the connection path without the DB2 Connect Gateway. This option results in a simpler infrastructure and better performance caused by the elimination of an intermediate stage between the client and DB2. Software upgrade and code maintenance becomes less complex. The Data Server driver, builds on "type 4" driver technology, and delivers "Connect" features as there are connection pooling, transaction pooling, and Sysplex workload balancing.



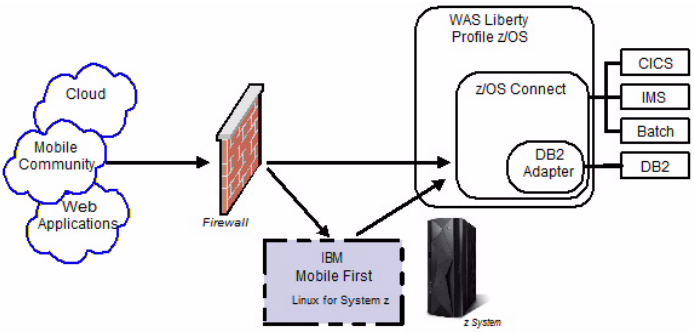
You still need Connect licensing, basically each client connecting directly to the DB2 for z/OS backend needs a "Connect" license. Where we used to set this up via the Connect Gateway, this can now be centrally managed via DB2 Connect Unlimited (Advanced) Edition, the alternative being that you distribute a license to all client systems that need to directly connect.

What concerns client functionality the IBM Data Server Driver replaces the "classic" client instance/directory configuration approach by the db2dsdriver.cfg file. The driver package is quite complete and includes support for JCC, ODBC/CLI, Open Source interfaces, CLPPlus, OLE DB and .NET. (In a Java only environment the IBM Data Server Driver for JDBC and SQLJ can be a valid alternative)

Only if you need administrative tools and features on the client side, or you want to compile programs and make use of the Embedded SQL application interface, the setup and configuration of the DB2 Data Server Client is necessary.

Broaden the connectivity spectrum: DB2 connectivity becomes z/OS connectivity.

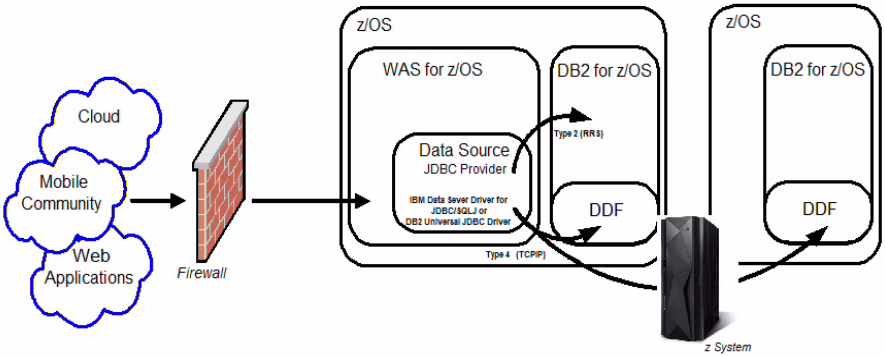
If you want to optimally reuse the assets running in the mainframe environment, if on the client side the REST/JSON technique is used, then on the mainframe side there is z/OS Connect.



z/OS Connect provides a common and consistent REST/JSON interface to the different mainframe backend systems. The z/OS Connect code is written to run in WAS Liberty Profile for z/OS. Access to data in DB2 for z/OS via z/OS Connect is through CICS, IMS and batch programs. Direct access to DB2 from z/OS is available since (delivered in) IBM DB2 Accessories for z/OS v3.3. The advantages of using z/OS Connect that it can function as a common and consistent entry point for mobile access, its Java so it runs on specialty engines and it shields backend systems from RESTful URIs and JSON data formats.

z/OS Connect can also be used as the way into the mainframe environment by the IBM MobileFirst Server. The MobileFirst Server is part of the IBM MobileFirst platform, and designed to seamlessly channel back-end enterprise systems to mobile devices. Currently the IBM MobileFirst platform is Linux for zSystems only.

For a wider Web/Java application support, there is the full functional WebSphere Application Server on z/OS that you can use as a hub into the z/OS environment. Client/Web applications can access the Application Server environment on z/OS directly and WAS Data Source objects through JDBC providers will further enable DB2 on z/OS access. Traditionally there is the choice between type 2 (single LPAR) or type 4 (DRDA distributed) connections. The following scheme tries to give you a idea of the setup:



Conclusion

Accessing DB2 for z/OS data from the “distributed” environment for many applications, existing and new, still is an important issue. This access can be achieved based on a “database connection protocol” or a “distributed/web application protocol” the choice you make depends on the type of the application.

Today for DB2 connections the idea is to let IBM Data Server Driver or Client replace the DB2 Connect Gateway. This solution delivers at least equivalent function, probably improves performance, and in all circumstances reduces complexity both in configuration and layers (direct connection).

The alternative solutions make use of the Java runtime on z/OS and let distributed/web/mobile applications connect to an application server run time, that in its turn will connect via drivers to the DB2 for z/OS back-end. The latest addition in this field is called z/OS connect (with DB2 adapter), a z/OS application gateway specialized in support for REST/JSON.

Optimizer-flexibiliteit

Peter Vanroose (ABIS)

In deze eerste bijdrage over de nieuwe mogelijkheden van de optimizer van DB2 11 voor z/OS hebben we het over twee interessante en minder gekende hulpmiddelen voor de performance-specialist: een nieuw (beter) hinting-mechanisme, en feedback van de optimizer in het geval van ontbrekende of foutieve statistieken.

Predicate selectivity overrides

Het `BIND QUERY` command werd ingevoerd in DB2 10, met als doel het centraliseren van optimizer *access-path hints*, in de catalog dus. Tot versie 9 kon dat alleen via een persoonlijke `PLAN_TABLE`.

`BIND QUERY` leest de tabellen die door `EXPLAIN`-statements of een vorige `BIND EXPLAIN(YES)` werden gevuld, met name de tabellen `PLAN_TABLE` en `DSN_PREDICAT_TABLE`. Hiertoe moet de gebruiker eerst instructies plaatsen in z'n `DSN_USERQUERY_TABLE`, essentieel eigenlijk gewoon "pointers" (via de kolom `QUERYNO`) naar welbepaalde rijen van z'n `PLAN_TABLE`, die hij manueel gewijzigd heeft (zoals b.v. een andere index of een andere join-method).
`BIND QUERY` exporteert dan die hints naar de catalog-tabellen (prefix `SYSIBM`) `SYSQUERY`, `SYSQUERYOPTS`, `SYSQUERYPLAN` en `SYSQUERYPREDICATE`.

De aldus gecreëerde catalog-hints worden steeds automatisch gehonoreerd, bij elke BIND of REBIND, dus zonder de noodzaak van een expliciet mee te geven BIND-optie `OPTHINT`.

Sinds DB2 11 is echter nog een belangrijke stap gezet in het hinting-verhaal. Het is nu namelijk niet langer nodig om eerst een gedetailleerde, manuele performance-analyse te gaan doen om b.v. het gebruik van een andere index of een andere join-method te verantwoorden.

Wanneer heeft de optimizer het bij het verkeerde eind, en willen we dus een hint gebruiken? Meestal omdat de optimizer een *filter factor* verkeerd inschat, dikwijls omwille van de aanwezigheid van host-variabelen in WHERE-predikaten.

Als we nu eens zouden kunnen, voor één specifiek predikaat van een query, enkel die foutief ingeschatte filter factor corrigeren? En inderdaad, dat is nu precies wat *predicate selectivity* betekent, en wat de nieuwe hinting-techniek toelaat: het volstaat, een *override* op te geven voor de foutief ingeschatte filter factor(s).

DB2 voorziet een nieuwe explain-tabel voor dit doel: `DSN_PREDICATE_SELECTIVITY`. Explain schrijft (bij normale uitvoering) één rij per predikaat naar deze tabel. De belangrijkste kolommen zijn:

`SELECTIVITY` (`FLOAT`) de geschatte filter factor voor dit predicaat
`WEIGHT` (`FLOAT`) % van gevallen waar deze selectiviteit geldt
`ASSUMPTION` (`VARCHAR`) met als waarde "`NORMAL`" of "`OVERRIDE`"

Er kunnen meerdere rijen voorkomen per predikaat, maar de som van de gewichten per predikaat is altijd 100%. De waarde “NORMAL” geeft aan dat het EXPLAIN is (de optimizer dus) die deze filter factor en dit gewicht heeft geproduceerd, zonder hinting dus.

Wanneer we voor een bepaald predikaat een realistischer filter factor willen communiceren naar de optimizer, dan volstaat het om manueel de eigen tabel DSN_PREDICATE_SELECTIVITY te wijzigen (insert/update/delete) en de “assumption”-waarde op “OVERRIDE” te zetten.

Om ten slotte deze “hint” te activeren, volstaat het uitvoeren van BIND QUERY, die sinds versie 11 ook DSN_PREDICATE_SELECTIVITY leest. Uiteraard gestuurd door instructies in DSN_USERQUERY_TABLE. Hierdoor worden de voorgestelde filter factors permanent (tot herroeping) in de catalog-tabel SYSQUERYPREDICATE geplaatst waar ze door elke volgende (RE)BIND gevonden zullen worden.

We vermijden zo een manuele optimisatie, en hoeven enkel de intuitief duidelijke filter factor-informatie aan de optimizer te geven.

Optimizer-feedback over ontbrekende statistieken

De nieuwe explain-tabel DSN_STAT_FEEDBACK wordt vanaf DB2 11 door EXPLAIN gevuld en bevat aanbevelingen voor enerzijds het verzamelen van nuttige, ontbrekende statistieken, en anderzijds het melden van conflicterende of verouderde statistieken.

Wanneer men deze aanbevelingen ter harte neemt, dus door het uitvoeren van RUNSTATS met bijkomende opties, zal dat de performance ten goed komen van queries die de genoemde tabellen of indexen gebruiken.

Concreet gaat het over de volgende feedback van de optimizer:

- Voor een bepaalde tablespace of index die de optimizer overwoog te gebruiken, de melding dat zelfs de basis-statistieken ontbreken.

- Voor een bepaalde kolom van een tabel, of voor een bepaalde groep van kolommen, de suggestie om hetzij histogram-statistieken of gewone frequentie-statistieken te verzamelen. Dit kan met één van de volgende RUNSTATS-opties:

```
HISTOGRAM NUMQUANTILES <n>
FREQVAL COUNT <n> MOST | LEAST | BOTH
```

- Conflicterende statistiek-waarden, b.v. over de grootte van een tabel en van z’n tablespace of z’n index, of de partitie-groottes.

- Een kolom met een opvallend laag aantal verschillende waarden, terwijl de tabel-cardinaliteit redelijk groot is. In dat geval is de kans groot dat de waarden niet gelijkmatig voorkomen (een zogenaamde “scheve” kansverdeling), en dan is FREQVAL-informatie aangewezen.

Merk op dat de momenteel geproduceerde hints soms wat te “uitbundig” zijn; een volgende versie van deze functionaliteit zal de scherpste hoekjes nog moeten bijschaven...

Dossier 11: Ditjes-en-datjes

Kris Van Thillo (ABIS)

Alter table add column, alter table drop column.

Het is reeds zeer lang mogelijk kolommen toe te voegen aan bestaande tabellen – de meesten onder ons hebben het ongetwijfeld nooit anders geweten. Naast de keuze van het te gebruiken datatype was het enkel even nodig na te denken over de null-problematiek. (Toevoegen vereist nog steeds dat betreffende kolom nullable moet zijn). Mogelijk was het daarnaast ook wenselijk even stil te staan bij de plek van de toe te voegen kolom in de tabel, en werd er alsnog de voorkeur aan gegeven de tabel te hermaken met een nieuwe, aangepaste kolomdefinitie. Third-party tools type Alter/Change manager van BMC werden hiertoe vaak aangewend. De toevoeging gebeurt in logisch opzicht onmiddellijk in de relevante DB2 cataloog-tabellen; fysiek wordt e.e.a. pas daadwerkelijk uitgevoerd op de tablespace bij een INSERT/UPDATE of bij de eerstvolgende REORG.

Sinds DB2 11 for z/OS is het ook mogelijk kolommen te *verwijderen*. Belangrijk is dat de implementatie verschilt van wat hierboven uiteengezet is over het toevoegen van een kolom. Inderdaad, IBM heeft er de voorkeur aan gegeven de *drop* van de kolom – fysiek en logisch – uit te stellen tot de eerstvolgende REORG; de wijziging wordt enkel geregistreerd als een '*pending change*', en dus pas op een later ogenblik daadwerkelijk uitgevoerd. Cataloog-wijzigingen op het betreffende object worden uitgesteld tot na de REORG. De techniek van de 'pending changes' werd in DB2 10 ingevoerd binnen de bredere problematiek van 'schema-evolutie'.

De '*drop column*'-actie wordt een standaard onderdeel van een transactie en kan dus worden gerollbacked. Ook na de commit – vóór de REORG – kan de wijziging nog worden geannuleerd door alle uitgestelde, *pending* acties op het betreffende object te vergeten (op tablespace-niveau: `alter tablespace drop pending changes`).

Er zijn een belangrijk aantal elementen waarmee we moeten rekening houden bij het weghalen van kolommen: niet alle kolommen kunnen worden weggehaald; online REORG moet worden uitgevoerd op tablespace-niveau en niet op partitie-niveau; en we moeten ons bewust zijn van de problematiek van copy/recover van objecten waaruit kolommen zijn weggehaald.

Autonome stored procedures.

Het is sinds DB2 v11 mogelijk autonome, native SQL stored procedures aan te maken. *Autonoom* wil in deze context zeggen: gestart als een autonome, onafhankelijke transactie die los staat van de transactie (en dus applicatie) die de autonome transactie heeft opgeroepen (b.v. via een applicatie, een andere procedure, of een trigger). Typisch beslist immers de oproepende transactie wat er

moet gebeuren (commit of rollback) met de wijzigingen aangebracht door de opgeroepen transactie. Dit is nu dus niet meer noodzakelijk het geval. Inderdaad, in standaard scenario's zouden transactie-management-instructies in opgeroepen procedures steevast genegeerd worden wanneer de oproeper verderop faalt (rollback).

Mooie case: het (manueel) loggen in tabellen van uitgevoerde acties, inclusief falende acties. B.v. een negatieve SQL code genereren zou aanleiding geven tot een globale rollback in een standaard scenario. Het gebruik van autonome stored procedures voor andere, eerder functionele redenen moet uiteraard met de nodige kritische ingesteldheid worden bekeken!

Merk op dat deze autonome transactie start in een nieuwe thread/nieuwe sessie – met alle normale gevolgen vandien (read consistency, locking, ...). In het verleden werd e.e.a. vaak gesimuleerd door een externe procedure op te roepen die onder bepaalde omstandigheden 'alleen maar' autonoom kon worden aangewend.

Eén belangrijke beperking: het oproepen van autonome transacties vanuit een autonome transactie is niet toegestaan.

Globale sessievariabelen.

Het komt eigenlijk hierop neer: een globale sessie-variabele is een stukje *gedeeld geheugen* waarvan verschillende delen van dezelfde *thread* (of sessie dus) gebruik kunnen maken om informatie te delen.

De variabele wordt globaal aangemaakt – al dan niet expliciet binnen een bepaald schema – en kan dan worden gebruikt door verschillende applicaties binnen een z/OS subsysteem. Gebruik van de variabele, alsook de mogelijkheid deze variabele aan te passen, wordt bepaald op basis van rechten die via grant/revoke statements dienen te worden uitgedeeld. De variabele kan worden gebruikt binnen de hele DB2 instance; de inhoud – toegekende waarde – is echter enkel zichtbaar binnen een specifieke sessie/thread.

Binnen een bepaalde batch job (thread) bijvoorbeeld, zal de inhoud van elke variabelen behouden blijven overheen commits.

DB2 11 definieert alvast een aantal standaard globale systeem-variabelen – bijvoorbeeld `SYSIBM.CLIENT_IPADDR`.

Voorbeeld:

```
CREATE VARIABLE CURS VARCHAR(30) DEFAULT 'xxxx';

GRANT READ, WRITE ON VARIABLE CURS TO PUBLIC;

SET CURS = 'NOSQL';

SELECT CGRP, CSTITLE, CAPRICE
FROM TBACCAD.TUTCOURSES
WHERE CGRP = CURS
ORDER BY CSTITLE;
```

CURSUSPLANNING, MAR - JUN 2017

Basiskennis SQL & relationele databases	850 EUR	20.03(L),03.04(W),02.05(L),06.06(W)
DB2 for z/OS basiscursus	1425 EUR	10.04(W)
DB2 for LUW basiscursus	1425 EUR	10.04(W)
SQL workshop	900 EUR	09.03(W), 22.05(L), 26.06(W)
SQL voor gevorderden	500 EUR	31.03(W), 30.06(L)
SQL voor BI rapportering & analyse	950 EUR	03.04(L), 08.06(W)
SQL PL database programmeren		op aanvraag
DB2 triggers, stored procedures, en User-Defined Functions	500 EUR	29.06(W)
DB2 for z/OS: programmeren voor gevorderden	1000 EUR	22.05(L)
DB2 for z/OS: SQL performance	1500 EUR	06.06(L)
XML in DB2		op aanvraag
DB2 for z/OS: database administratie	2100 EUR	19.06(L)
DB2 for z/OS: installation & migration	1080 EUR	09.03(UK)
DB2 for z/OS: data recovery		op aanvraag
DB2 for z/OS: systems performance and tuning		op aanvraag
DB2 LUW DBA - Kernvaardigheden	2000 EUR	20.06(W)
DB2 LUW DBA - Configure, monitor&tune		op aanvraag
JDBC	500 EUR	02.05(L)
DB2 10 for LUW: new features		op aanvraag
DB2 11 for z/OS: changes & new features	525 EUR	24.05(L)
DB2 12 for z/OS: changes & new features		op aanvraag
Data warehouse concepten		op aanvraag
Dimensionaal modelleren	2120 EUR	04.04 (Hilversum)
Big Data concepten	500 EUR	13.04(W), 18.04(L)
Big Data in de praktijk met Hadoop	1000 EUR	20.03(W), 27.04(L)
Big Data in de praktijk met Spark	1000 EUR	02.05(W)
MongoDB	1050 EUR	23.03(L), 04.04(W)

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