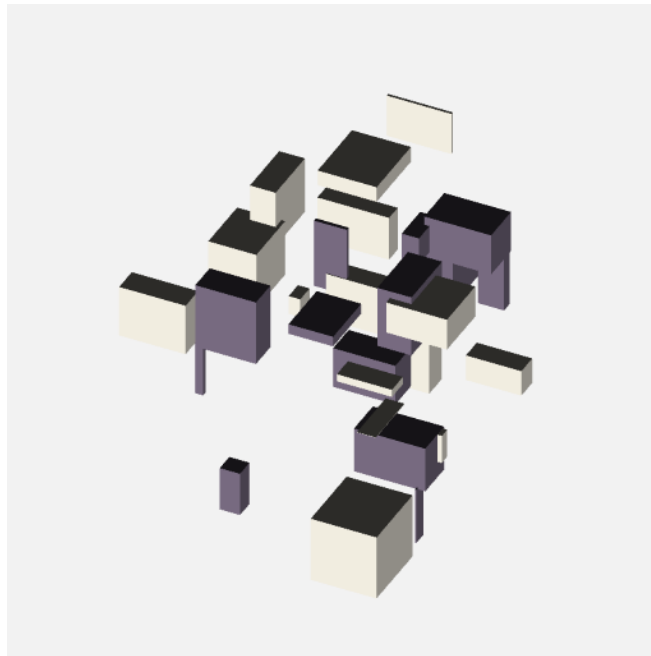


DB2 SQL

Pour développeurs IBMi

Nouveautés de la V7



Sommaire

Préambule.....	6
1 Préambule.....	8
1.1 Clause LIMIT.....	9
2 DB2 for i Services.....	11
2.1 Application Services.....	11
2.1.1 QSYS2.QCMDEXC.....	11
2.1.2 QSYS2.DELIMIT_NAME.....	12
2.1.3 QSYS2.OVERRIDE_TABLE.....	13
2.1.4 QSYS2.LIBRARY_LIST_INFO.....	14
2.2 TCP/IP Services.....	15
2.2.1 SYSIBMADM.ENV_SYS_INFO.....	15
2.2.2 QSYS2.TCPIP_INFO.....	17
2.3 PTF Services.....	18
2.3.1 QSYS2.PTF_INFO.....	18
2.3.2 QSYS2.GROUP_PTF_INFO.....	20
2.3.3 SYSTOOLS.GROUP_PTF_CURRENCY.....	21
2.4 Security Services.....	22
2.4.1 QSYS2.USER_INFO.....	22
2.4.2 QSYS2.FUNCTION_INFO.....	25
2.4.3 QSYS2.FUNCTION_USAGE.....	26
2.4.4 SQL_CHECK_AUTHORITY.....	28
2.4.5 Sécuriser des colonnes.....	29

2.5 Work Management Services.....	30
2.5.1 QSYS2.SYSTEM_VALUE_INFO.....	30
2.5.2 QSYS2.GET_JOB_INFO.....	31
2.5.3 QSYS2.SCHEDULED_JOB_INFO.....	32
2.6 Storage Services.....	33
2.6.1 QSYS2.SYSDISKSTAT.....	33
2.6.2 QSYS2.USER_STORAGE.....	34
2.7 Journal Services.....	35
2.7.1 QSYS2.DISPLAY_JOURNAL.....	35
2.8 Object Services.....	37
2.8.1 QSYS2.OBJECT_STATISTICS.....	37
2.9 Utility Services.....	42
2.9.1 QSYS2.GENERATE_SQL.....	44
2.9.2 SYSTOOLS.CHECK_SYSRoutine.....	46
2.10 Performance Services.....	48
2.11 Health Services.....	50
2.11.1 QSYS2.SYSLIMTBL.....	50
2.11.2 Valeurs limites.....	52
2.12 Support de JSON dans DB2.....	54
3 Outils pour Développeurs SQL.....	55
3.1 Variables globales.....	55
3.2 L'ordre MERGE.....	57
3.3 Utilisation du XML avec XMLTABLE.....	62
3.3.1 SQL vers XML.....	62
3.3.2 XML vers SQL.....	64
3.3.3 XSLT.....	66

3.3.4 Lecture de XML avec Namespace.....	68
3.4 Hiérarchie récursive.....	69
3.5 Evolution du Timestamp.....	72
3.6 Result Set et procédures stockées.....	74
3.6 La syntaxe « OR REPLACE ».....	76
3.7 Les paramètres de procédures.....	78
3.8 L'ordre TRUNCATE TABLE.....	79
3.9 Pagination avec DB2.....	81
4 DB2 et la sécurité des données.....	83
4.1 Sécuriser les données avec les Field Procedure.....	83
4.2 Contrainte Violation.....	86
4.3 Row and Column Access Control (RCAC).....	87
5 Compléments.....	93
5.1 DSPFFD amélioré et autres outils.....	93
5.2 Analyse de dépendances via les tables systèmes.....	98
6 Nouveautés V7R3 et V7R4.....	100
HISTORY_LOG_INFO table function.....	100
Exemples.....	100
ACTIVE_JOB_INFO table function.....	101
Exemples.....	101
JOB_INFO table function.....	102
Exemples.....	102
JOBLOG_INFO table function.....	103
Exemples.....	103
Utilisation des données JSON.....	104
IFS Services.....	106

OBJECT_STATISTICS table function.....	110
Exemple.....	110
SYSPARTITIONSTAT.....	112
RECORD_LOCK_INFO view.....	113
Exemple.....	113
SPOOLED_FILE_DATA table function.....	114
IFS_OBJECT_STATISTICS table function.....	116
Exemple.....	116
HTTPPOSTCLOB et HTTPPOSTBLOB scalar functions.....	117

Dépôt d'archivage de ce dossier :

<https://github.com/gregja/SQLMasters>

Dernière mise à jour : le 4 mai 2022

Préambule

La base de données DB2 for i a pour réputation (justifiée) de nécessiter une surveillance restreinte.

Mais avec la montée en puissance du "Big Data", et la consommation d'espace disque en augmentation constante qui en résulte, les administrateurs système et bases de données ont de plus en plus besoin de disposer d'informations en temps réel sur l'état des systèmes et des bases qu'ils supervisent.

Au travers du catalogue système, DB2 for i recèle de nombreuses pépites qui répondent aux besoins des administrateurs systèmes (accès aux valeurs système, aux PTFs, à la consommation disque, etc.), et en particulier des DevOps.

Ce document présente certaines de ces fonctionnalités, et la manière dont on peut les utiliser au quotidien pour administrer des bases DB2 for i. Il vient compléter deux autres documents présents dans le même dépôt Github, les documents « SQL_DBTwo_Quickstart » et « SQL_DBTwo_Masterclass ».

J'avais rédigé la première version de ce document peu de temps après la sortie de la V7R2, mais depuis lors, de nombreuses nouveautés sont apparues sur les V7R3 et V7R4, aussi j'ai décidé, en avril 2022, de le compléter en ajoutant un chapitre 6, dans lequel j'ai répertorié les fonctionnalités qui m'ont semblé les plus utiles dans mon activité DevOps. Cette mise à jour n'est donc pas exhaustive, même si elle couvre beaucoup de nouveautés. Mais j'ai ajouté ci-dessous quelques liens complémentaires qui vont permettre de compléter vos connaissances sur le sujet.

Parmi les nouveautés très intéressantes que j'ai eu l'occasion d'expérimenter récemment, il y a notamment les fonctions DB2 dédiées à la gestion de l'IFS. Grâce à ces fonctions SQL, on peut lire ou écrire des fichiers dans des répertoires de l'IFS :

<https://www.ibm.com/docs/en/i/7.3?topic=services-ifs>

Autre nouveauté qui m'a rendu un grand service, la vue DB2 SPOOLED_FILE_DATA, qui permet de récupérer le contenu d'un spoule IBM i dans une table DB2, pour pouvoir ensuite en extraire certaines informations (cf. exemple dans le dernier chapitre).

Pour un tour d'horizon des nouveautés de DB2 for i, plus orientées sur la gestion des travaux et des PTF, je vous invite à consulter cette synthèse publiée par la société Gaïa :

<https://www.gaia.fr/wpfb-file/s46-les-nouveautes-v7-de-la-gestion-des-travaux-et-des-ptfs-pdf/>

Une page de présentation très détaillée proposée par la société Volubis :

<https://www.volubis.fr/news/liens/courshtm/V7.2/SQLasaService.html>

Le slide ci-dessous, publié par IBM en mars 2022 propose une synthèse des nouveautés apparues sur DB2 for i en V7R3 et V7R4 :

<https://www.itheis.com/wp-content/uploads/2018/08/Webinar-ITHEIS-IBM-du-17-mars-2022-Partie-IBM-nouveautes-DB2.pdf>

Autre liens utiles :

<https://www.foothing.net/>

1 Préambule

Avec les Technology Refresh, DB2 for i se positionne comme un auxiliaire des administrateurs systèmes et bases de données, en simplifiant un certain nombre de tâches d'administration.

Permet l'accès à des fonctions système via SQL

Solution alternative aux commandes CL et APIs

Nouvelle rubrique dans les Technology Updates :

DB2 for i updates by category
DB2 for i Functional Enhancements
DB2 for i Security Enhancements
DB2 for i Performance Enhancements
DB2 for i Database Management Enhancements
DB2 for i Availability/Recovery Enhancements
OmniFind for IBM i
DB2 for i Services

La documentation officielle pour ces nouveaux services est accessible dans le "IBM Knowledge Center":

http://www-01.ibm.com/support/knowledgecenter/ssw_ibm_i_72/rzajq/rzajqservicessys.htm

Lien vers le Wiki :

<https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20i%20Technology%20Updates/page/DB2%20for%20i%20-%20Services>

1.1 Clause LIMIT

Attention, il y a une nouveauté importante à noter sur la V7R1 qui ne fait pas partie des « DB2 for i Services », c'est la nouvelle clause LIMIT.

Jusqu'à l'arrivée de la V7R1, DB2 for i n'offrait pas d'équivalent des clauses LIMIT et OFFSET. Il a fallu attendre l'arrivée de la V7R1 TR11, et de la V7R2 TR3, pour enfin bénéficier de ce mécanisme. Ce petit tableau extrait d'une documentation IBM vous explique le principe :

Syntax	Alternative Syntax	Action
LIMIT x	FETCH FIRST x ROWS ONLY	Return the first x rows
LIMIT x OFFSET y	OFFSET y ROWS FETCH FIRST x ROWS ONLY	Skip the first y rows and return the next x rows
LIMIT y,x	OFFSET y ROWS FETCH FIRST x ROWS ONLY	Skip the first y rows and return the next x rows

II

devient donc plus facile de porter du code SQL provenant notamment de MySQL ou de PostgreSQL par exemple.

Attention, il y a une petite restriction : la clause OFFSET n'est autorisée que dans le cadre d'une requête de type Full-Select externe appliqué à un DECLARE CURSOR, ou d'un « prepared statement » sur une requête de type SELECT. Vous n'avez pas compris ? Ne vous inquiétez pas, à vrai moi non plus. De toute façon, on s'en moque, utilisez la 3ème solution (LIMIT x, y), et c'est marre.

Il faut quand même noter que, avant l'arrivée de la clause LIMIT (en V7), on pouvait quand même gérer une pagination en SQL sans passer par un curseur scrollable. C'était juste un peu plus compliqué à écrire. Dans notre contexte de pagination sur la liste des pays, on devait écrire la requête suivante :

```
SELECT * FROM (
  SELECT CODFRA, CODISO, LIBELLE,
    ROW_NUMBER() OVER (ORDER BY CODFRA ASC) AS RN
  FROM LSTPAYS
  WHERE CODFRA LIKE ?
) AS FOO WHERE RN BETWEEN ? AND ?
```

Attention : si la technique SQL ci-dessus fonctionne bien sur des tables SQL de taille raisonnable (de l'ordre de quelques dizaines milliers de lignes), j'ai par le passé – vers 2010 ou 2011 - rencontré des problèmes de latence en appliquant cette technique sur des tables de tailles supérieures. A contrario, le curseur scrollable répondait sans latence, et semblait plus robuste. Je n'ai pas eu l'occasion de refaire de tests récemment pour vérifier si ce problème est encore d'actualité.

Pour de plus amples précisions sur l'utilisation du curseur scrollable, prière de vous reporter au document « Livre_blanc_PHP_IBMi_v3 » qui se trouve dans le dépôt ci-dessous. J'y présente un exemple d'implémentation en PHP :

<https://github.com/gregja/phpLibrary4i>

2 DB2 for i Services

2.1 Application Services

2.1.1 QSYS2.QCMDEXC

La procédure stockée QSYS2.QCMDEXC() peut être utilisée pour exécuter différentes commandes systèmes IBMi.

Ce n'est pas vraiment une nouveauté, mais ce qui est nouveau, c'est que - depuis la TR7 - on n'est plus obligé de préciser la longueur de la commande système à exécuter, car la procédure est en mesure de le déterminer d'elle-même.

Deux exemples d'utilisation :

- Ajout d'une bibliothèque dans la "library list" :

```
CALL QSYS2.QCMDEXC('ADDLIB PRODLIB2');
```

- La même chose mais, via une "expression" concaténée à la volée :

```
DECLARE V_LIBRARY_NAME VARCHAR(10);  
...  
SET V_LIBRARY_NAME = 'PRODLIB2';  
...  
CALL QSYS2.QCMDEXC('ADDLIB ' CONCAT V_LIBRARY_NAME);
```

2.1.2 QSYS2.DELIMIT_NAME

Annoncé sur la TR8, mais en réalité déjà disponible sur la TR7, la fonction DELIMIT_NAME renvoie une valeur avec des délimiteurs (guillemets et/ou apostrophes) répondant à différentes problématiques rencontrées par les développeurs SQL.

Le schéma de la fonction est QSYS2 (il est implicite et on n'a pas besoin de le préciser à chaque utilisation).

Le paramètre d'entrée est une chaîne de 128 caractères maximum (en cas de dépassement, la valeur renvoyée est tronquée à cette longueur). La valeur renvoyée est un VARCHAR contenant une chaîne correctement délimitée.

Exemple :

```
SELECT
DELIMIT_NAME('ABC'), -- ABC
      DELIMIT_NAME('ABC') , -- "ABC"
      DELIMIT_NAME('TEST"NAME'), -- "TEST""NAME"
      DELIMIT_NAME('TEST' 'NAME2'), -- "TEST'NAME2"
      DELIMIT_NAME('NEW') -- "NEW"
FROM SYSIBM.SYSDUMMY1;
```

2.1.3 QSYS2.OVERRIDE_TABLE

Il est parfois nécessaire, pour des raisons de performance, sur des applications critiques, d'agir sur le taux de transfert des données d'une table, en jouant sur la taille du buffer utilisé par DB2 pour les transferts de données. On peut réaliser ce type de manipulation en demandant au système d'utiliser un buffer intermédiaire de 256 K, comme dans l'exemple suivant :

```
CALL QCMDEXC ( 'OVRDBF FILE(PRODUCT) TOFILE(GJABASE/PRODUCT)
SEQONLY(*YES *BUF256KB)' ) ;
```

Mais avec l'arrivée de la TR7, on n'est plus obligé de recourir à la commande système OVRDBF, on peut recourir à la procédure stockée DB2 QSYS2/OVERRIDE_TABLE, comme dans les exemples suivants :

```
-- Override sur la table Product de la bibliothèque, en utilisant un
buffer bloqué à 256K
```

```
CALL QSYS2.OVERRIDE_TABLE('GJABASE', 'PRODUCT', '*BUF256KB');
```

```
-- Suppression de l'override
```

```
CALL QSYS2.OVERRIDE_TABLE('GJABASE', 'PRODUCT', 0);
```

A noter : pour l'override, un nombre d'octets spécifique peut être fourni, ou on peut recourir aux valeurs spéciales prédéfinies suivantes : *BUF32KB, *BUF64KB, *BUF128KB, *BUF256KB.

```
FROM QSYS2.GROUP_PTF_INFO
WHERE PTF_GROUP_NAME IN ('SF99610', 'SF99710')
AND PTF_GROUP_STATUS = 'INSTALLED' ;
```

2.1.4 QSYS2.LIBRARY_LIST_INFO

Disponibilité de la fonctionnalité : IBM i 7.2 TR3/IBM i 7.1 TR9

La vue QSYS2.LIBRARY_LIST_INFO permet de récupérer la liste des bibliothèques courante du travail en cours d'exécution :

```
SELECT * FROM QSYS2.LIBRARY_LIST_INFO;
```

2.2 TCP/IP Services

2.2.1 SYSIBMADM.ENV_SYS_INFO

La vue DB2 ENV_SYS_INFO permet de récupérer via une simple requête SQL différentes informations qui peuvent intéresser tout le monde.

```
SELECT * FROM SYSIBMADM.ENV_SYS_INFO ;
```

OS_NAME	OS_VERSION	OS_RELEASE	HOST_NAME	TOTAL_CPUS	CONFIGURED_CPUS	TOTAL_MEMORY
IBM i	7	1	XXX SIX-AXA fr	1	1	4096

On peut par exemple se servir des valeurs de OS_VERSION et OS_RELEASE pour savoir si le code s'exécute sur un serveur en V7R1, ce qui autorise à utiliser certaines instructions SQL comme par exemple MERGE (qui est franchement géniale pour les opérations de mise à jour).

```
MERGE INTO My_LIBRARY.testmerge A
USING (SELECT * FROM SYSIBM.SYSDUMMY1) B
ON A.macle = 'CLE1'
WHEN MATCHED THEN
UPDATE SET
  a.codea = 'A1' ,
  a.coden = a.coden + 1
WHEN NOT MATCHED THEN
INSERT ( a.macle , a.codea , a.coden )
VALUES( 'CLE1' , 'A1' , 1 )
;
```

Si on détecte que l'on est dans une version antérieure à la V7R1, alors il faut utiliser une solution de rechange, plus laborieuse à écrire certes, mais qui permettra à votre application de fonctionner sur différentes versions d'OS de manière optimale.

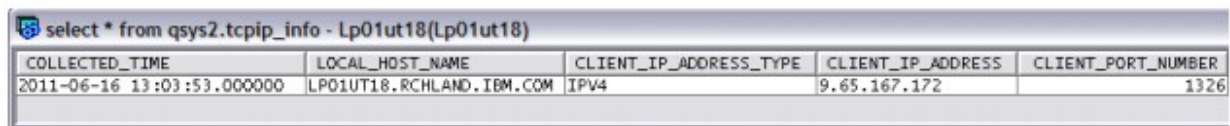
A noter que dans le MERGE que j'ai utilisé, la requête déclarée dans le paramètre USING est une requête sur la table pivot SYSDUMMY1. Ce n'est pas la manière la plus courante d'utiliser MERGE (elle est d'ailleurs rarement présentée dans les documentations), mais elle est très pratique quand les données à mettre à jour

proviennent de variables programmes (variables de programme RPG, ou variables de procédure stockée DB2).

2.2.2 QSYS2.TCPIP_INFO

La vue TCPIP_INFO fournit différentes informations intéressantes sur la connexion au serveur courant.

```
select * from QSYS2.TCPIP_INFO;
```



The screenshot shows a SAS query window titled "select * from qsys2.tcpip_info - Lp01ut18(Lp01ut18)". The results are displayed in a table with the following columns and data:

COLLECTED_TIME	LOCAL_HOST_NAME	CLIENT_IP_ADDRESS_TYPE	CLIENT_IP_ADDRESS	CLIENT_PORT_NUMBER
2011-06-16 13:03:53.000000	LP01UT18.RCHLAND.IBM.COM	IPV4	9.65.167.172	1326



The screenshot shows a SAS query window with the following columns and data:

SERVER_IP_ADDRESS_TYPE	SERVER_IP_ADDRESS	SERVER_PORT_NUMBER	HOST_VERSION
IPV4	9.5.36.87	8471	V7R1M0

2.3 PTF Services

2.3.1 QSYS2.PTF_INFO

La vue QSYS2.PTF_INFO fournit de précieuses informations aux administrateurs systèmes :

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	PTF_PRODUCT_ID	LICPGM	VARCHAR	7
2	PTF_PRODUCT_OPTION	PRODOPT	VARCHAR	6
3	PTF_PRODUCT_RELEASE_LEVEL	PRODRLS	VARCHAR	6
4	PTF_PRODUCT_DESCRIPTION	PRODDISC	VARCHAR	132
5	PTF_IDENTIFIER	PTFID	VARCHAR	7
6	PTF_RELEASE_LEVEL	PTFRLS	VARCHAR	6
7	PTF_PRODUCT_LOAD	PRODLOAD	VARCHAR	4
8	PTF_LOADED_STATUS	LOADSTAT	VARCHAR	19
9	PTF_SAVE_FILE	SAVF	VARCHAR	3
10	PTF_COVER_LETTER	COVER	VARCHAR	3
11	PTF_ON_ORDER	ONORD	VARCHAR	3
12	PTF_IPL_ACTION	IPLACT	VARCHAR	19
13	PTF_ACTION_PENDING	ACTPEND	VARCHAR	3
14	PTF_ACTION_REQUIRED	ACTREQ	VARCHAR	12
15	PTF_IPL_REQUIRED	IPLREQ	VARCHAR	9
16	PTF_IS_RELEASED	RELEASED	VARCHAR	3
17	PTF_MINIMUM_LEVEL	MINLVL	VARCHAR	2
18	PTF_MAXIMUM_LEVEL	MAXLVL	VARCHAR	2
19	PTF_STATUS_TIMESTAMP	STATTIME	TIMESTAMP	10
20	PTF_SUPERCEDED_BY_PTF	SUPERCEDE	VARCHAR	7
21	PTF_CREATION_TIMESTAMP	CRTTIME	TIMESTAMP	10
22	PTF_TECHNOLOGY_REFRESH_PTF	TRPTF	VARCHAR	3

SELECT * FROM QSYS2.PTF_INFO ;

PTF_PRODUC...	PTF_PRO...	PTF_PRO...	PTF_PRODUCT_DESCRIPTION	PTF_IDENTIFIER	PTF_RELEASE_LEVEL	PTF_PRODUCT_LOAD
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF06003	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47854	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47855	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47856	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47857	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47858	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47869	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47870	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47871	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47872	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47873	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47874	V7R1M0	5050

Exemples :

- trouver toutes les PTF qui seront impactées par le prochain IPL :

```
SELECT PTF_IDENTIFIER, PTF_IPL_ACTION, A.*
FROM QSYS2.PTF_INFO A
WHERE PTF_IPL_ACTION <> 'NONE'
```

- trouver les PTF chargées mais non encore appliquées :

```
SELECT PTF_IDENTIFIER, PTF_IPL_REQUIRED, A.*
FROM QSYS2.PTF_INFO A
WHERE PTF_LOADED_STATUS = 'LOADED'
ORDER BY PTF_PRODUCT_ID
```

2.3.2 QSYS2.GROUP_PTF_INFO

La vue GROUP_PTF_INFO contient des informations sur les PTF de groupe pour le serveur.

L'API des PTF de Groupes (QpzListPtfGroups) API est utilisée pour récupérer ces informations.

L'information retournée est similaire à celle disponible sur la commande WRKPTFGRP.

Le tableau suivant décrit les colonnes renvoyées par la vue. Le schéma est QSYS2.

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	COLLECTED_TIME	COLLE00001	TIMESTAMP	10
2	PTF_GROUP_NAME	PTF_G00001	VARCHAR	60
3	PTF_GROUP_DESCRIPTION	PTF_G00002	VARCHAR	100
4	PTF_GROUP_LEVEL	PTF_G00003	INTEGER	9
5	PTF_GROUP_TARGET_RELEASE	PTF_G00004	VARCHAR	6
6	PTF_GROUP_STATUS	PTF_G00005	VARCHAR	20

SELECT * FROM QSYS2.GROUP_PTF_INFO ;

COLLECTED_TIME	PTF_GROUP_N...	PTF_GROUP_DESCRIPTION	PTF_GROUP_LE...	PTF_GRO...	PTF_GROUP_ST
2014-05-09 12:28:23.570065	SF99145	...	6	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99145	PERFORMANCE TOOLS	5	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99363	...	14	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99363	WEBSphere APP SERVER V7.0	13	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99366	...	9	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99366	PRINT PTFS	8	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99367	...	8	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99367	TCP/IP PTF	8	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99368	...	27	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99368	IBM HTTP SERVER FOR I	26	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99572	...	16	V7R1M0	INSTALLED

Exemple : déterminer le niveau de la dernière cumulative de PTF installée sur le système

```
SELECT MAX(PTF_GROUP_LEVEL) AS CUM_LEVEL
FROM QSYS2.GROUP_PTF_INFO
WHERE PTF_GROUP_NAME IN ('SF99610', 'SF99710')
AND PTF_GROUP_STATUS = 'INSTALLED' ;
```

2.3.3 SYSTOOLS.GROUP_PTF_CURRENCY

Disponibilité de la fonctionnalité : IBM i 7.2 TR3/IBM i 7.1 TR9

La vue SYSTOOLS.GROUP_PTF_CURRENCY permet de déterminer si les groupes de PTF installés sont à jour. Ce service utilise les fonctions de SYSTOOLS.HTTPGETBLOB pour se connecter sur un site d'IBM et ainsi récupérer la liste des derniers groupes de PTF disponibles. Grâce à cette information, et aux infos renvoyées par La vue GROUP_PTF_INFO (cf. chapitre précédent), la vue est en mesure de renvoyer les écarts.

La vue est facile à utiliser:

```
SELECT * FROM SYSTOOLS.GROUP_PTF_CURRENCY
```

PTF Group Currency	Group Id	Group Title	Level Inst.	Level Avail.	IBM Last Updated	Status
INSTALLED LEVEL IS CURRENT	SF99702	720 DB2 for IBM i	3	3	11/11/2014	INSTALLED
UPDATE AVAILABLE	SF99713	720 IBM HTTP Server for i	4	5	12/23/2014	INSTALLED
UPDATE AVAILABLE	SF99716	720 Java	3	4	12/10/2014	INSTALLED

2.4 Security Services

2.4.1 QSYS2.USER_INFO

La vue USER_INFO contient des informations à propos des profils utilisateurs :
Le tableau suivant fournit le détail des 66 colonnes renvoyées par la vue. Le schéma est QSYS2.

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	AUTHORIZATION_NAME	USER_NAME	VARCHAR	10
2	PREVIOUS_SIGNON	PRVSIGNON	TIMESTAMP	10
3	SIGN_ON_ATTEMPTS_NOT_VALID	SIGNONINV	INTEGER	9
4	STATUS	STATUS	VARCHAR	10
5	PASSWORD_CHANGE_DATE	PWDCHGDAT	TIMESTAMP	10
6	NO_PASSWORD_INDICATOR	NOPWD	VARCHAR	3
7	PASSWORD_EXPIRATION_INTERVAL	PWDEXPITV	SMALLINT	4
8	DATE_PASSWORD_EXPIRES	PWDEXPDAT	TIMESTAMP	10
9	DAYS_UNTIL_PASSWORD_EXPIRES	PWDDAYSEXP	INTEGER	9
10	SET_PASSWORD_TO_EXPIRE	PWDEXP	VARCHAR	3
11	USER_CLASS_NAME	USRCLS	VARCHAR	10
12	SPECIAL_AUTHORITIES	SPCAUT	VARCHAR	88
13	GROUP_PROFILE_NAME	GRPPRF	VARCHAR	10
14	OWNER	OWNER	VARCHAR	10
15	GROUP_AUTHORITY	GRPAUT	VARCHAR	10
16	ASSISTANCE_LEVEL	ASTLVL	VARCHAR	10
17	CURRENT_LIBRARY_NAME	CURLIB	VARCHAR	10
18	INITIAL_MENU_NAME	INLMNU	VARCHAR	10
19	INITIAL_MENU_LIBRARY_NAME	INLMNULIB	VARCHAR	10
20	INITIAL_PROGRAM_NAME	INITPGM	VARCHAR	10
21	INITIAL_PROGRAM_LIBRARY_NAME	INITPGMLIB	VARCHAR	10
22	LIMIT_CAPABILITIES	LMTCPB	VARCHAR	10
23	TEXT_DESCRIPTION	TEXT	VARCHAR	50
24	DISPLAY_SIGNON_INFORMATION	DSPSGNINF	VARCHAR	10
25	LIMIT_DEVICE_SESSIONS	LMTDEVSSN	VARCHAR	10
26	KEYBOARD_BUFFERING	KBDBUF	VARCHAR	10
27	MAXIMUM_ALLOWED_STORAGE	MAXSTGLRG	BIGINT	18
28	STORAGE_USED	STGUSED	BIGINT	18
29	HIGHEST_SCHEDULING_PRIORITY	PTYLMT	CHAR	1
30	JOB_DESCRIPTION_NAME	JOB	VARCHAR	10
31	JOB_DESCRIPTION_LIBRARY_NAME	JOBDLIB	VARCHAR	10

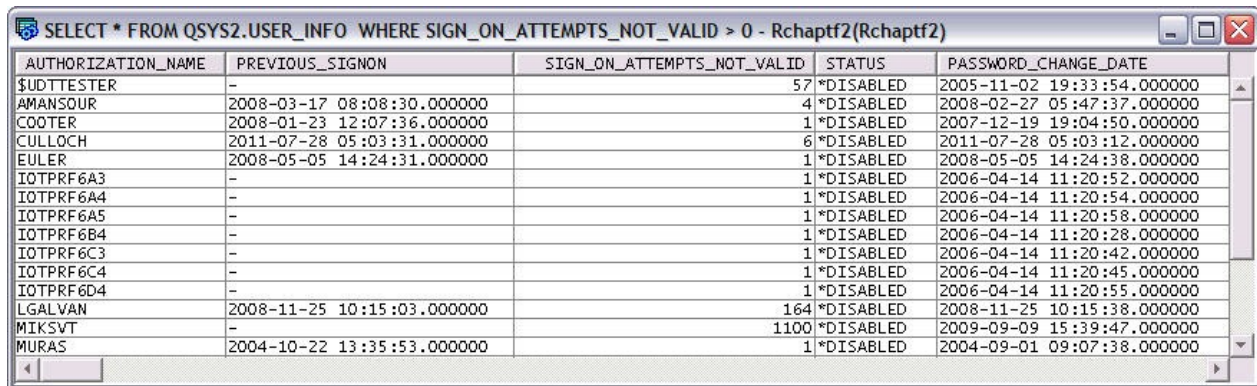
32	ACCOUNTING_CODE	ACGCDE	VARCHAR	15
33	MESSAGE_QUEUE_NAME	MSGQ	VARCHAR	10
34	MESSAGE_QUEUE_LIBRARY_NAME	MSGQLIB	VARCHAR	10
35	MESSAGE_QUEUE_DELIVERY_METHOD	DLVRY	VARCHAR	10
36	MESSAGE_QUEUE_SEVERITY	SEV	SMALLINT	4
37	OUTPUT_QUEUE_NAME	OUTQ	VARCHAR	10
38	OUTPUT_QUEUE_LIBRARY_NAME	OUTQLIB	VARCHAR	10
39	PRINT_DEVICE	PRTDEV	VARCHAR	10
40	SPECIAL_ENVIRONMENT	SPCENV	VARCHAR	10
41	ATTENTION_KEY_HANDLING_PROGRAM_NAME	ATNPGM	VARCHAR	10
42	ATTENTION_KEY_HANDLING_PROGRAM_LIBRARY_NAME	ATNPGMLIB	VARCHAR	10
43	LANGUAGE_ID	LANGID	VARCHAR	10
44	COUNTRY_OR_REGION_ID	CNTRYID	VARCHAR	10
45	CHARACTER_CODE_SET_ID	CCSID	VARCHAR	6
46	USER_OPTIONS	USROPT	VARCHAR	77
47	SORT_SEQUENCE_TABLE_NAME	SRTSEQ	VARCHAR	10
48	SORT_SEQUENCE_TABLE_LIBRARY_NAME	SRTSEQLIB	VARCHAR	10
49	OBJECT_AUDITING_VALUE	OBJAUD	VARCHAR	10
50	USER_ACTION_AUDIT_LEVEL	AUDLVL	VARCHAR	341
51	GROUP_AUTHORITY_TYPE	GRPAUTTYP	VARCHAR	10
52	USER_ID_NUMBER	UID	BIGINT	18
53	GROUP_ID_NUMBER	GID	BIGINT	18
54	LOCALE_JOB_ATTRIBUTES	SETJOBATR	VARCHAR	88
55	GROUP_MEMBER_INDICATOR	GRPMBR	VARCHAR	3
56	DIGITAL_CERTIFICATE_INDICATOR	DCIND	VARCHAR	3
57	CHARACTER_IDENTIFIER_CONTROL	CHRIDCTL	VARCHAR	10
58	LOCAL_PASSWORD_MANAGEMENT	LCLPWDGMT	VARCHAR	3
59	BLOCK_PASSWORD_CHANGE	PWDCHGBLK	VARCHAR	10
60	USER_ENTITLEMENT_REQUIRED	ENTITLERQD	VARCHAR	3
61	USER_EXPIRATION_INTERVAL	USREXPITV	SMALLINT	4
62	USER_EXPIRATION_DATE	USREXPDATE	TIMESTAMP	10
63	USER_EXPIRATION_ACTION	ACTION	VARCHAR	8
64	HOME_DIRECTORY	HOMEDIR	VARGRAPHIC	1024
65	LOCALE_PATH_NAME	LOCALE	VARGRAPHIC	1024
66	USER_DEFAULT_PASSWORD	DFTPWD	VARCHAR	3

Attention :

- seuls les objets de type *USRPRF sur lesquels l'utilisateur dispose de l'autorité *READ sont renvoyés par la vue.
- les valeurs renvoyées correspondent aux informations fournies par l'API QSYRUSRI.

Exemple : Déterminer quels utilisateurs ont rencontré des problèmes de "SIGN ON".

```
SELECT * FROM QSYS2.USER_INFO  
WHERE SIGN_ON_ATTEMPTS_NOT_VALID > 0 ;
```



AUTHORIZATION_NAME	PREVIOUS_SIGNON	SIGN_ON_ATTEMPTS_NOT_VALID	STATUS	PASSWORD_CHANGE_DATE
\$UDTTESTER	-	57	*DISABLED	2005-11-02 19:33:54.000000
AMANSOUR	2008-03-17 08:08:30.000000	4	*DISABLED	2008-02-27 05:47:37.000000
COOTER	2008-01-23 12:07:36.000000	1	*DISABLED	2007-12-19 19:04:50.000000
CULLOCH	2011-07-28 05:03:31.000000	6	*DISABLED	2011-07-28 05:03:12.000000
EULER	2008-05-05 14:24:31.000000	1	*DISABLED	2008-05-05 14:24:38.000000
IOTPRF6A3	-	1	*DISABLED	2006-04-14 11:20:52.000000
IOTPRF6A4	-	1	*DISABLED	2006-04-14 11:20:54.000000
IOTPRF6A5	-	1	*DISABLED	2006-04-14 11:20:58.000000
IOTPRF6B4	-	1	*DISABLED	2006-04-14 11:20:28.000000
IOTPRF6C3	-	1	*DISABLED	2006-04-14 11:20:42.000000
IOTPRF6C4	-	1	*DISABLED	2006-04-14 11:20:45.000000
IOTPRF6D4	-	1	*DISABLED	2006-04-14 11:20:55.000000
LGALVAN	2008-11-25 10:15:03.000000	164	*DISABLED	2008-11-25 10:15:38.000000
MIKSVT	-	1100	*DISABLED	2009-09-09 15:39:47.000000
MURAS	2004-10-22 13:35:53.000000	1	*DISABLED	2004-09-01 09:07:38.000000

2.4.2 QSYS2.FUNCTION_INFO

La vue FUNCTION_INFO fournit un équivalent SQL à l'API QSYRTVFI (QsyRetrieveFunctionInformation).

Le tableau suivant fournit le détail des colonnes renvoyées par la vue. Le schéma est QSYS2.

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	FUNCTION_ID	FCNID	VARCHAR	30
2	FUNCTION_CATEGORY	FCNCAT	VARCHAR	10
3	FUNCTION_TYPE	FCNTYP	VARCHAR	13
4	FUNCTION_NAME_MESSAGE_TEXT	FCNMSGTXT	VARGRAPHIC	330
5	FUNCTION_NAME	FCNNAM	VARGRAPHIC	330
6	FUNCTION_DESCRIPTION_MESSAGE_TEXT	FCNDESCTXT	VARGRAPHIC	330
7	FUNCTION_DESCRIPTION	FCNDESC	VARGRAPHIC	330
8	FUNCTION_PRODUCT_ID	FCNPRDID	VARCHAR	30
9	FUNCTION_GROUP_ID	FCNGRPID	VARCHAR	30
10	DEFAULT_USAGE	DFTUSG	VARCHAR	7
11	ALLOBJ_INDICATOR	ALLOBJ	VARCHAR	8
12	USAGE_INFORMATION_INDICATOR	USGINFO	VARCHAR	3

Exemple:

SELECT * FROM QSYS2.FUNCTION_INFO ORDER BY FUNCTION_ID ;

FUNCTION_ID	FUNCTION_CATEGORY	FUNCTION_TYPE	FUNCTION_NAME
QIBM_ACCESS_ALLOBJ_JOBLOG	3 - HOST	ADMINISTRABLE	Access job log
QIBM_ALLOBJ	3 - HOST	GROUP	All object
QIBM_ALLOBJ_TRACE_ANY_USER	3 - HOST	ADMINISTRABLE	Trace any user
QIBM_BASE_OPERATING_SYSTEM	3 - HOST	PRODUCT	i5/OS
QIBM_DB	3 - HOST	GROUP	DATABASE
QIBM_DB_DDMDRDA	3 - HOST	ADMINISTRABLE	DDM & DRDA APP
QIBM_DB_SQLADM	3 - HOST	ADMINISTRABLE	DATABASE ADMIN
QIBM_DB_SYSMON	3 - HOST	ADMINISTRABLE	DATABASE INFOR
QIBM_DB_ZDA	3 - HOST	ADMINISTRABLE	TOOLBOX APPLIC
QIBM_DIRSRV_ADMIN	3 - HOST	ADMINISTRABLE	IBM Tivoli Dir
QIBM_QCST_SERVICE_CLUSTADMIN	3 - HOST	ADMINISTRABLE	Cluster Admini
QIBM_QCST_SERVICE_CLUSTMGMT	3 - HOST	GROUP	Cluster Manage
QIBM_QCST_SERVICE_CLUSTOPER	3 - HOST	ADMINISTRABLE	Cluster Operat
QIBM_QINAV_NAVIGATOR_WEB	3 - HOST	PRODUCT	iSeries Naviga
QIBM_QINAV_WEB_CONFIGURE	3 - HOST	ADMINISTRABLE	Configure iSer
QIBM_QINAV_WEB_FUNCTIONS	3 - HOST	ADMINISTRABLE	Manage Server

2.4.3 QSYS2.FUNCTION_USAGE

La vue FUNCTION_USAGE fournit un équivalent SQL de l'API QSYRTFUI (QsyRetrieveFunctionUsageInfo).

Seuls les utilisateurs ayant l'autorité *SECADM peuvent examiner les informations renvoyées par cette vue.

Les utilisateurs ne disposant pas de cette autorité recevront un SQLCODE -443.

Le tableau suivant fournit le détail des colonnes renvoyées par la vue. Le schéma est QSYS2.

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	FUNCTION_ID	FCNID	VARCHAR	30
2	USER_NAME	USER_NAME	VARCHAR	10
3	USAGE	USAGE	VARCHAR	7
4	USER_TYPE	USER_TYPE	VARCHAR	5

Exemple :

Déterminer quelles fonctions ont fait l'objet de modifications de droits (GRANT ou REVOKE) :

```
SELECT * FROM QSYS2.FUNCTION_USAGE ORDER BY FUNCTION_ID, USER_NAME ;
```

FUNCTION_ID	USER_NAME	USAGE	USER_TYPE
QIBM_QSY_SYSTEM_CERT_STORE	QDIRSRV	ALLOWED	USER
QIBM_QSY_SYSTEM_CERT_STORE	QTCP	ALLOWED	USER
QIBM_QSY_SYSTEM_CERT_STORE	QYPSJSVR	ALLOWED	USER
QIBM_Q1A_ARC	QSECOFR	ALLOWED	USER
QIBM_Q1A_ARC_CTLG_BRM.ARCGRP	QSECOFR	ALLOWED	USER
QIBM_Q1A_ARC_PCY	QSECOFR	ALLOWED	USER
QIBM_Q1A_BKU	QSECOFR	ALLOWED	USER
QIBM_Q1A_BKU_CTLG_BRM.BKUGRP	QSECOFR	ALLOWED	USER

2.4.4 SQL_CHECK_AUTHORITY

La fonction scalaire SQL_CHECK_AUTHORITY permet de contrôler si l'utilisateur courant est habilité à effectuer des requêtes sur un objet donné.

Les paramètres d'appel sont :

- nom de la bibliothèque (schéma)
- nom de l'objet DB2

La valeur renvoyée est de type SMALLINT, sa signification est la suivante :

- 0 : l'utilisateur n'est pas autorisé à "requêter" cet objet, ou l'objet n'est pas de type *FILE, ou l'objet n'existe pas
- 1 : l'utilisateur est autorisé à effectuer des requêtes sur cet objet

Exemple :

```
SELECT SQL_CHECK_AUTHORITY ('QSYS2' , 'FUNCTION_USAGE') FROM  
SYSIBM.SYSDUMMY1 ; -- 0
```

```
SELECT SQL_CHECK_AUTHORITY ('GJABASE' , 'CONTRAT_TB') FROM  
SYSIBM.SYSDUMMY1 ; -- 1
```

2.4.5 Sécuriser des colonnes

Disponibilité de cette fonctionnalité : TR2 (en V7R2) ou TR10 (en V7R1)

Exemple : sécuriser le contenu de la colonne « credit card » (CCNBR) dans la table ORDERS de la bibliothèque LIB1.

```
CALL SYSPROC.SET_COLUMN_ATTRIBUTE('LIB1', 'ORDERS', 'CCNBR',  
'SECURE YES');
```

Une fois cette opération réalisée, la colonne

The SET_COLUMN_ATTRIBUTE procedure sets the SECURE attribute for a column so variable values used for the column can't be seen in the database monitor or plan cache.

Les valeurs possibles pour le 4^{ème} paramètre sont :

- **SECURE NO**
Cette colonne ne contient pas de données nécessitant d'être masquée dans le moniteur de base de données, ou dans le cache de plan d'accès
- **SECURE YES**
Cette colonne contient des données nécessitant d'être masquées dans le moniteur de base de données, et dans le cache de plan d'accès. Les colonnes concernées apparaîtront avec la valeur *SECURE, sauf si le profil connecté a le niveau de sécurité QSECOFR.

Le paramètre de sécurité apparaît dans la colonne SECURE de la vue système QSYS2/SYSCOLUMNS2.

2.5 Work Management Services

2.5.1 QSYS2.SYSTEM_VALUE_INFO

La vue SYSTEM_VALUE_INFO renvoie différentes valeurs systèmes. C'est l'équivalent SQL de l'API "Retrieve System Values" (QWCRSVAL).

Les autorités spéciales *ALLOBJ ou *AUDIT sont nécessaires pour pouvoir récupérer le contenu des valeurs systèmes suivantes : QAUDCTL, QAUDENDACN, QAUDFRCLVL, QAUDLVL, QAUDLVL2, et QCRTOBJAUD.

Les colonnes sélectionnées pour lesquels l'utilisateur n'a pas les autorisations adéquates contiendront en sortie '*NOTAVL' ou -1.

Exemple : Examiner les valeurs systèmes de type "maximum".

```
SELECT * FROM SYSTEM_VALUE_INFO  
WHERE SYSTEM_VALUE_NAME LIKE '%MAX%' ;
```

SYSTEM_VALUE_NAME	CURRENT_NUMERIC_VALUE	CURRENT_CHARACTER_VALUE
QMAXACTLVL	32767	-
QMAXSIGN	-	000003
QPWDMAXLEN	10	-
QMAXSGNACN	-	3
QMAXJOB	163520	-
QMAXSPLF	9999	-

2.5.2 QSYS2.GET_JOB_INFO

La fonction GET_JOB_INFO est une "table function", c'est à dire une fonction renvoyant une structure équivalente à une table. Cette structure contient ici une seule ligne renvoyant des informations relatives au travail dont l'identifiant a été transmis à la fonction.

Le schéma de la fonction est QSYS2.

Le paramètre d'entrée a une structure bien connue des utilisateurs IBMi, comme le montre l'exemple suivant :

Envoi des informations relatives au travail suivant : 816516/GJA/QPADEV0006.

```
SELECT * FROM TABLE(QSYS2.GET_JOB_INFO('816516/GJA/QPADEV0006')) A;
```

V_JOB_STATUS	V_ACTIVE_JOB_STATUS	V_RUN_PRIORITY	V_SBS_NAME	V_CPU_USED	V_TEMP_STORAGE_USED_...	V_AUX_IO_REQUESTED	V_PAGE_FAULTS	V.
*ACTIVE	DSPW	20	QINTER	13	3	95	74	

Pour pouvoir utiliser cette fonction, l'appelant doit disposer au minimum de l'autorité spéciale *JOBCTL, ou il doit être autorisé à utiliser les fonctions systèmes QIBM_DB_SQLADM, ou QIBM_DB_SYSMON.

2.5.3 QSYS2.SCHEDULED_JOB_INFO

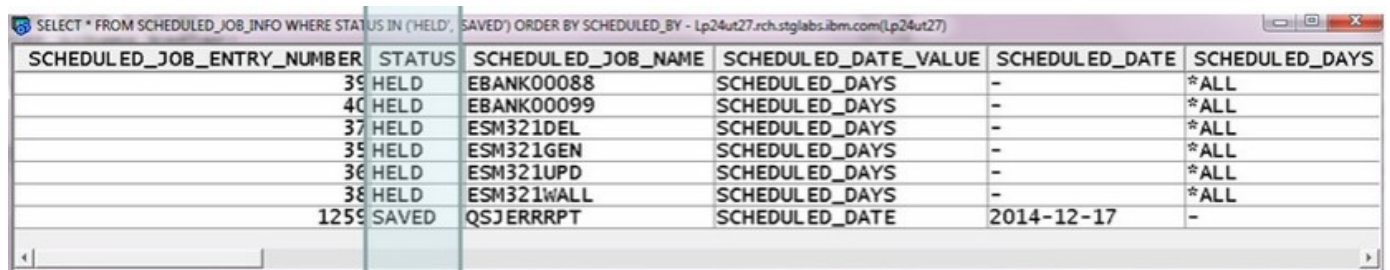
La vue SCHEDULED_JOB_INFO permet de consulter en temps réel le contenu du planning de travaux de l'IBM i (auquel on accède habituellement au moyen de la commande WRKJOBSCDE).

Le schéma de la fonction est QSYS2.

Disponibilité de cette fonctionnalité : TR2(en V7R2) ou TR10(en V7R1)

Exemple :

```
SELECT * FROM QSYS2.SCHEDULED_JOB_INFO A
WHERE A.STATUS IN ('HELD', 'SAVED')
ORDER BY SCHEDULED_BY;
```



SCHEDULED_JOB_ENTRY_NUMBER	STATUS	SCHEDULED_JOB_NAME	SCHEDULED_DATE_VALUE	SCHEDULED_DATE	SCHEDULED_DAYS
35	HELD	EBANK00088	SCHEDULED_DAYS	-	*ALL
40	HELD	EBANK00099	SCHEDULED_DAYS	-	*ALL
37	HELD	ESM321DEL	SCHEDULED_DAYS	-	*ALL
35	HELD	ESM321GEN	SCHEDULED_DAYS	-	*ALL
36	HELD	ESM321UPD	SCHEDULED_DAYS	-	*ALL
38	HELD	ESM321WALL	SCHEDULED_DAYS	-	*ALL
1259	SAVED	QSJERRRPT	SCHEDULED_DATE	2014-12-17	-

Structure détaillée du contenu renvoyé par la vue sur :

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20i%20Technology%20Updates/page/QSYS2.SCHEDULED_JOB_INFO%20-%20view

2.6 Storage Services

2.6.1 QSYS2.SYSDISKSTAT

La vue SYSDISKSTAT contient les informations relatives aux disques.

Le tableau suivant fournit le détail des colonnes renvoyées par la vue. Le schéma est QSYS2.

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	ASP_NUMBER	ASP_NUMBER	SMALLINT	4
2	DISK_TYPE	DISK_TYPE	VARCHAR	4
3	DISK_MODEL	DISK_MODEL	VARCHAR	4
4	UNIT_NUMBER	UNITNBR	SMALLINT	4
5	UNIT_TYPE	UNIT_TYPE	SMALLINT	4
6	UNIT_STORAGE_CAPACITY	UNITSCAP	BIGINT	18
7	UNIT_SPACE_AVAILABLE	UNITSPACE	BIGINT	18
8	PERCENT_USED	PERCENTUSE	DECIMAL	7
9	UNIT_MEDIA_CAPACITY	UNITMCAP	BIGINT	18
10	LOGICAL_MIRRORED_PAIR_STATUS	MIRRORPS	CHAR	1
11	MIRRORED_UNIT_STATUS	MIRRORUS	CHAR	1

Exemple :

```
SELECT * FROM QSYS2.SYSDISKSTAT
```

ASP_N...	DISK_...	DISK_...	UNIT_N...	UNIT_...	UNIT_STORAGE_CAPACI...	UNIT_SPACE_AVAILAB...	PERCENT_US...
14327	0070		1	0	70564970496	23976001536	66.022
14327	0078		2	0	35282485248	11987042304	66.025
14327	0078		3	0	35282485248	11981381632	66.041
14327	0078		4	0	35282485248	11986284544	66.027
14327	0070		5	0	70564970496	23976603648	66.021

Autre exemple : Renvoie des informations pour toutes les unités SSD.

```
SELECT * FROM QSYS2.SYSDISKSTAT WHERE UNIT_TYPE = 1
```

2.6.2 QSYS2.USER_STORAGE

La vue USER_STORAGE renvoie le pourcentage d'utilisation des ressources disques par profil utilisateur. C'est l'équivalent de l'API QSYRUSRI (Retrieve User Information).

Le tableau suivant fournit le détail des 66 colonnes renvoyées par la vue. Le schéma est QSYS2.

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	AUTHORIZATION_NAME	USER_NAME	VARCHAR	10
2	ASPGRP	ASPGRP	VARCHAR	10
3	MAXIMUM_STORAGE_ALLOWED	MAXSTG	BIGINT	18
4	STORAGE_USED	STGUSED	BIGINT	18

Vous devez disposer de l'autorité *READ sur les profils utilisateurs où la vue ne vous renverra aucune information.

Les données sont fournies par SYSBAS, IASP et profil utilisateur.

Exemple :

```
SELECT * FROM QSYS2.USER_STORAGE
WHERE USER_NAME = 'GJA';
```

AUTHORIZATION_NAME	ASPGRP	MAXIMUM_STORAGE_ALLOWED	STORAGE_USED
GJA	*SYSBAS	-	1747372

2.7 Journal Services

2.7.1 QSYS2.DISPLAY_JOURNAL

L'affichage des entrées d'un journal via une interface graphique nécessitait jusqu'ici l'utilisation d'API. C'était contraignant, et généralement peu performant.

L'exploitation des entrées de journaux est intéressante pour les administrateurs, car elle leur permet de traquer différents types de problèmes (comme des manipulations de données non conformes aux spécifications des applications utilisées).

La fonction QSYS2/Display_Journal est une nouvelle "table function" permettant à l'utilisateur de visualiser les entrées dans un journal, en exécutant une simple requête SQL.

Exemple 1: afficher toutes les entrées du récepteur courant pour le journal MJATST/QSQJRN.

```
select * from table (
Display_Journal(
'MJATST', 'QSQJRN', -- Journal library and name
'', '' -- Receiver library and name
CAST(null as TIMESTAMP), -- Starting timestamp
CAST(null as DECIMAL(21,0)), -- Starting sequence number
'', -- Journal codes
'', -- Journal entries
'', '', '', '', -- Object library, Object name, Object type, Object member
'', -- User
'', -- Job
'' -- Program
) ) as x;
```

Exemple 2 : trouver tous les changements effectués par SUPERUSER à l'intérieur de la table PRODDATA/SALES

```
select journal_code, journal_entry_type, object, object_type, X.* from table
(
  QSYS2.Display_Journal(
    'PRODDATA', 'QSQJRN', -- Journal library and name
    '', '', -- Receiver library and name
    CAST(null as TIMESTAMP), -- Starting timestamp
    CAST(null as DECIMAL(21,0)), -- Starting sequence number
    '', -- Journal codes
    '', -- Journal entries
    'PRODDATA', 'SALES', '*FILE', 'SALES', -- Object library, Object name, Object
    type, Object member
    '', -- User
    '', -- Job
    '' -- Program
  ) ) as x
WHERE journal_entry_type in ('DL', 'PT', 'PX', 'UP') AND "CURRENT_USER" =
'SUPERUSER'
order by entry_timestamp desc
```

Pour de plus amples sur les journaux et leurs récepteurs, reportez-vous à la documentation de l'API QjoRetrieveJournalEntries API dans l'infocenter d'IBM.

2.8 Object Services

2.8.1 QSYS2.OBJECT_STATISTICS

La fonction table OBJECT_STATISTICS renvoie un certain nombre d'informations sur les objets d'une liste.

Exemple : renvoi de tous les objets de type *FILE de la bibliothèque GJABASE

```
SELECT OBJNAME, OBJTYPE, OBJOWNER, OBJDEFINER,  
OBJCREATED, OBJSIZE, OBJTEXT, OBJLONGNAME,  
LAST_USED_TIMESTAMP, DAYS_USED_COUNT,  
LAST_RESET_TIMESTAMP, IASP_NUMBER, OBJATTRIBUTE  
  
FROM TABLE (QSYS2.OBJECT_STATISTICS('GJABASE ', '*FILE') ) AS X ;
```

ou, strictement équivalent d'un point de vue fonctionnel :

```
SELECT OBJNAME, OBJTYPE, OBJOWNER, OBJDEFINER,  
OBJCREATED, OBJSIZE, OBJTEXT, OBJLONGNAME,  
LAST_USED_TIMESTAMP, DAYS_USED_COUNT,  
LAST_RESET_TIMESTAMP, IASP_NUMBER, OBJATTRIBUTE  
  
FROM TABLE (QSYS2.OBJECT_STATISTICS('GJABASE ', '*ALL') ) AS X  
  
WHERE X.OBJTYPE = '*FILE' ;
```

OBJNAME	OBJT...	OBJOW...	OBJDE...	OBJCREATED	OBJSIZE	OBJTEXT
CONTRAT_TB	*FILE	GJA	GJA	2012-05-24 14:...	249856	Table des con
CONTR00001	*FILE	GJA	GJA	2012-05-24 14:...	184320	-
CONTR00002	*FILE	GJA	GJA	2012-05-24 14:...	184320	-
CONTR00003	*FILE	GJA	GJA	2012-05-24 14:...	184320	-
EXCEPTION2	*FILE	GJA	GJA	2011-05-18 14:...	102400	-
IMPENG	*FILE	GJA	GJA	2012-12-13 10:...	40960	-
IMPFRA	*FILE	GJA	GJA	2012-12-13 11:...	40960	-

Autres exemples :

- trouver les journaux contenus dans la bibliothèque MJATST :

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS( 'MJATST ', 'JRN' ) ) AS X
```

ou

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS( 'MJATST ', '*JRN' ) ) AS X
```

- trouver les journaux et récepteurs de journaux dans la bibliothèque MJATST.

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS( 'MJATST ', 'JRN JRNRCV' ) )  
AS X
```

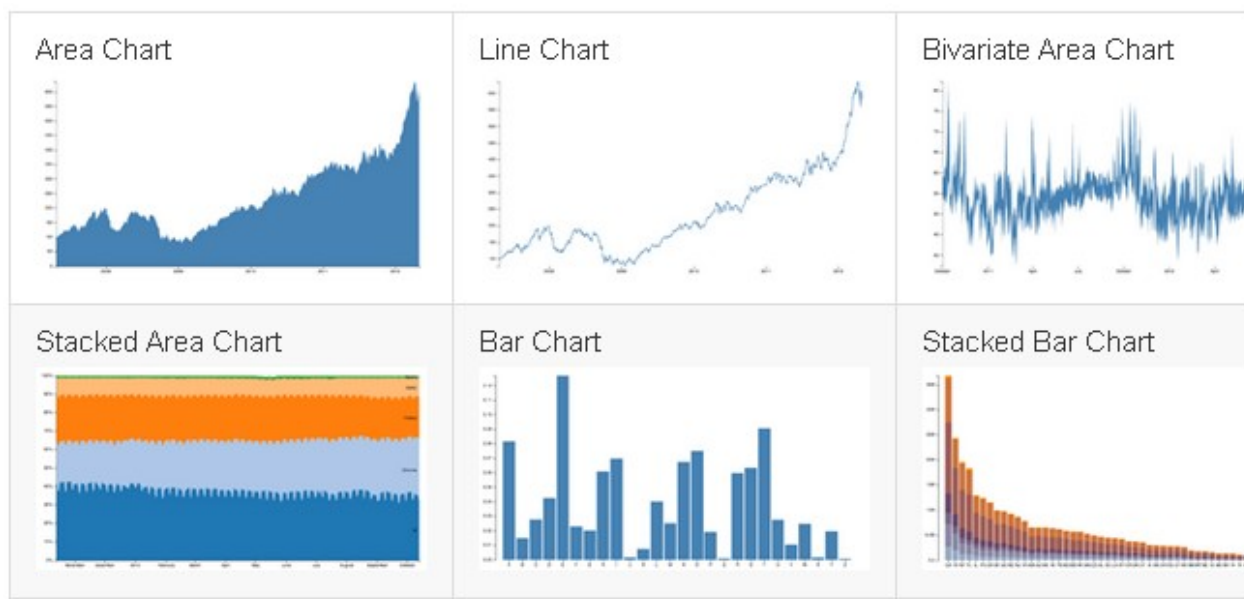
ou

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS( 'MJATST ', '*JRN  
*JRNRCV' ) ) AS X
```

Grâce à cette fonction, on peut envisager de prendre des clichés périodiques de l'état des bases de données, et éventuellement surveiller leur évolution, sous forme de tableaux HTML, ou de tableaux de bord plus sophistiqués (graphiques), avec des solutions payantes comme DB2 WebQuery d'IBM, ou des solutions open-source comme par exemple le projet D3 (<http://d3js.org>).

Exemple de graphiques pouvant être mis en oeuvre facilement avec D3 :

<https://github.com/mbostock/d3/wiki/Gallery#basic-charts>



On notera que la fonction OBJECT_STATISTICS n'est pas la seule manière de surveiller le contenu des tables. On peut aussi s'appuyer sur la table système QSYS2.SYSTABLESTAT qui permet d'obtenir en temps réel le nombre de lignes de chaque table d'une bibliothèque, ainsi que le nombre de lignes supprimées (pour les REORG notamment), et pas mal d'autres informations (cf. tableau ci-dessous) :

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	TABLE_SCHEMA	TABSCHEMA	VARCHAR	128
2	TABLE_NAME	TABNAME	VARCHAR	128
3	PARTITION_TYPE	PARTTYPE	CHAR	1
4	NUMBER_PARTITIONS	NBRPARTS	INTEGER	9
5	NUMBER_DISTRIBUTED_PARTITIONS	DSTPARTS	INTEGER	9
6	NUMBER_ROWS	CARD	BIGINT	18
7	NUMBER_ROW_PAGES	NPAGES	BIGINT	18
8	NUMBER_PAGES	FPAGES	BIGINT	18
9	OVERFLOW	OVERFLOW	BIGINT	18
10	CLUSTERED	CLUSTERED	CHAR	1
11	ACTIVE_BLOCKS	ACTBLOCKS	BIGINT	18
12	AVGCOMPRESSEDROWSIZE	ACROWSIZE	BIGINT	18
13	AVGROWCOMPRESSIONRATIO	ACROWRATIO	FLOAT	29
14	AVGROWSIZE	AVGROWSIZE	BIGINT	18
15	PCTROWSCOMPRESSED	PCTCROWS	FLOAT	29

16	PCTPAGESSAVED	PCTPGSAVED	SMALLINT	4
17	NUMBER_DELETED_ROWS	DELETED	BIGINT	18
18	DATA_SIZE	SIZE	BIGINT	18
19	VARIABLE_LENGTH_SIZE	VLSIZE	BIGINT	18
20	FIXED_LENGTH_EXTENTS	FLEXTENTS	BIGINT	18
21	VARIABLE_LENGTH_EXTENTS	VLEXTENTS	BIGINT	18
22	COLUMN_STATS_SIZE	CSTATSSIZE	BIGINT	18
23	MAINTAINED_TEMPORARY_INDEX_SIZE	MTISIZE	BIGINT	18
24	NUMBER_DISTINCT_INDEXES	DISTINCTIX	INTEGER	9
25	OPEN_OPERATIONS	OPENS	BIGINT	18
26	CLOSE_OPERATIONS	CLOSES	BIGINT	18
27	INSERT_OPERATIONS	INSERTS	BIGINT	18
28	UPDATE_OPERATIONS	UPDATES	BIGINT	18
29	DELETE_OPERATIONS	DELETES	BIGINT	18
30	CLEAR_OPERATIONS	DSCLEAR	BIGINT	18
31	COPY_OPERATIONS	DSCOPIES	BIGINT	18
32	REORGANIZE_OPERATIONS	DSREORGS	BIGINT	18
33	INDEX_BUILDS	DSINXBLDS	BIGINT	18
34	LOGICAL_READS	LGLREADS	BIGINT	18
35	PHYSICAL_READS	PHYREADS	BIGINT	18
36	SEQUENTIAL_READS	SEQREADS	BIGINT	18
37	RANDOM_READS	RANREADS	BIGINT	18
38	LAST_CHANGE_TIMESTAMP	LASTCHG	TIMESTAMP	10
39	LAST_SAVE_TIMESTAMP	LASTSAVE	TIMESTAMP	10
40	LAST_RESTORE_TIMESTAMP	LASTRST	TIMESTAMP	10
41	LAST_USED_TIMESTAMP	LASTUSED	TIMESTAMP	10
42	DAYS_USED_COUNT	DAYSUSED	INTEGER	9
43	LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP	10
44	NUMBER_PARTITIONING_KEYS	NBRPKEYS	INTEGER	9
45	PARTITIONING_KEYS	PARTKEYS	VARCHAR	2880
46	SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR	10
47	SYSTEM_TABLE_NAME	SYS_TNAME	CHAR	10

Exemple de requête permettant d'identifier les écarts - en termes de nombre de lignes - entre 2 bibliothèques (MA_BIB_1 et MA_BIB_2), pour toutes les tables dont le nom est préfixé par "DIM" :

```
WITH TMPSTAT AS (  
    SELECT A.TABLE_SCHEMA, A.TABLE_NAME, A.NUMBER_ROWS AS  
    NUMBER_ROWS_APPBIB2,  
        (SELECT B.NUMBER_ROWS FROM QSYS2.SYSTABLESTAT B  
            WHERE A.TABLE_NAME = B.TABLE_NAME  
            AND B.TABLE_SCHEMA = 'MA_BIB_1'  
        ) AS NUMBER_ROWS_APPBIB1  
    FROM QSYS2.SYSTABLESTAT A  
    WHERE A.TABLE_SCHEMA = 'MA_BIB_2'  
    AND SUBSTR(A.TABLE_NAME, 1, 3) = 'DIM'  
)  
SELECT * FROM TMPSTAT  
WHERE NUMBER_ROWS_APPBIB2 > 0  
    AND NUMBER_ROWS_APPBIB2 <> NUMBER_ROWS_APPBIB1  
ORDER BY TABLE_NAME  
;
```

2.9 Utility Services

Les procédures suivantes fournissent des interfaces permettant d'interagir avec différents éléments du système.

Nous allons les passer en revue brièvement, je vous invite à vous reporter à la documentation officielle pour de plus amples informations (et nous verrons plus en détail les 2 procédures que j'ai indiquées en rouge) :

http://www-01.ibm.com/support/knowledgecenter/api/content/ssw_ibm_i_72/rzajq/rzajqservicesutility.htm

CANCEL_SQL

La procédure CANCEL_SQL demande l'annulation d'une instruction SQL pour le travail spécifié en paramètre.

CHECK_SYSCST

La procédure CHECK_SYSCST compare les entrées dans le tableau QSYS2.SYSCONSTRAINTS entre deux systèmes .

CHECK_SYSROUTINE

La procédure CHECK_SYSROUTINE compare des entrées dans la table QSYS2.SYSROUTINES entre deux systèmes.

DUMP_SQL_CURSORS

La procédure DUMP_SQL_CURSORS répertorie les curseurs ouverts pour un travail donné.

FIND_AND_CANCEL_QSQRVR_SQL

La procédure FIND_AND_CANCEL_QSQRVR_SQL identifie un ensemble de travaux ayant une activité SQL, et les annule en toute sécurité .

FIND_QSQRVR_JOBS

La procédure FIND_QSQRVR_JOBS renvoie des informations sur un travail de QSQRVR .

GENERATE_SQL

La procédure GENERATE_SQL génère le code SQL permettant de recréer un

objet de base de données. Les résultats sont retournés dans le membre de fichier source de base de données spécifiée, ou sous la forme d'un "result set".

RESTART_IDENTITY

La procédure RESTART_IDENTITY examine une table source, détermine la colonne contenant l'identifiant et détermine également sa prochaine valeur. Cette valeur et le nom de la colonne sont transmis à une table cible qui les utilisera pour la prochaine insertion de ligne.

Dans la suite de ce chapitre, nous allons regarder un peu plus en détail les procédures CHECK_SYSROUTINE et GENERATE_SQL.

2.9.1 QSYS2.GENERATE_SQL

Annoncé sur la TR8, mais en réalité déjà disponible sur la TR7, la procédure stockée GENERATE_SQL génère le code SQL DDL (Data Definition Language) permettant de recréer un objet DB2. Le résultat peut être renvoyé dans un membre de fichier source, ou sous forme de "result set".

A noter : si le source en sortie est dirigé vers la table temporaire QTEMP/Q_GENSQL, avec pour nom de membre Q_GENSQL, alors le résultat est renvoyé simultanément sous forme de "result set".

Exemples :

- Générer le code DDL pour toutes les tables du schéma SAMPLE_CORPDB, et renvoyer le résultat sous forme de "result set" :

```
CALL QSYS2.GENERATE_SQL('%', 'SAMPLE_CORPDB', 'TABLE',  
    REPLACE_OPTION => '0');
```

- Générer le code DDL d'une vue ou d'une procédure :

```
CALL QSYS2.GENERATE_SQL('CHECK_SYSRoutine', 'SYSTOOLS', 'PROCEDURE');  
CALL QSYS2.GENERATE_SQL('COMMANDES2', 'FPHSAW', 'VIEW');
```

- Générer le code DDL pour tous les indexs du schéma SAMPLE_CORPDB, dont le nom commence par un 'X', et placer le résultat dans le fichier source nommé DDLSOURCE/GENFILE, membre INDEXSRC.

```
CALL QSYS2.GENERATE_SQL('X%', 'SAMPLE_CORPDB', 'INDEX',  
    'GENFILE', 'DDLSOURCE', 'INDEXSRC',  
    REPLACE_OPTION => '0');
```

Pour de plus amples renseignements :

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20i%20Technology%20Updates/page/QSYS2.GENERATE_SQL%28%29%20procedure

2.9.2 SYSTOOLS.CHECK_SYSROUTINE

La procédure CHECK_SYSROUTINE compare les entrées dans la table système QSYS2.SYSROUTINES du système courant, avec les entrées de la même table sur un autre serveur, pour une base dont le nom est indiqué en paramètre d'entrée de la procédure (ce nom devant être identique sur les 2 serveurs à comparer).

Le schéma est SYSTOOLS.

Les paramètres d'entrée sont les suivants :

- remote-rdb-name : chaîne de caractères contenant le nom de la base de données "remote"
- schema-name : chaîne de caractères contenant le nom d'un schéma du système hôte à comparer avec le même schéma de la base de données "remote"
- result set : un INTEGER indiquant si un result set doit être renvoyé (valeur 0) ou pas (valeur 1).

La procédure renvoie un "result set" au client SQL "appelant". Si aucun "result set" n'est demandé, alors les différences entre les 2 systèmes sont "loguées" dans la table temporaire SESSION.SYSRTNDIFF.

Exemple :

Comparer le système courant avec un système "remote" (L001UT18) pour identifier les routines qui ne sont pas identiques dans la bibliothèque CORPDB_EX.

```
CALL SYSTOOLS.CHECK_SYSROUTINE( 'LP01UT18', 'CORPDB_EX' ) ;
```

Il faut souligner que le code source de la procédure CHECK_SYSROUTINE peut être régénéré (soit via le logiciel System i Navigator, soit via la nouvelle procédure GENERATE_SQL). Il peut dès lors être adapté à vos besoins. Par exemple, si le nom de la base de données "remote" et le nom de la base de données locale diffèrent, il est souhaitable de pouvoir ajouter un paramètre supplémentaire à la procédure, pour tenir compte de cette différence de codification intervenant entre 2 systèmes. C'est ce que j'ai fait en créant une procédure CHECK_SYSROUTINE2,

dérivée de la précédente, mais autorisant la transmission de 2 paramètres supplémentaires de manière à pouvoir un environnement de référence (serveur + bibliothèque), à comparer avec un environnement "cible" (serveur + bibliothèque), ce qui, en termes d'utilisation, donne ceci :

```
CALL SYSTOOLS.CHECK_SYSRoutine2( 'TEST' , 'FEHSAP' , 'PREPROD' , 'FPHSAP' ,  
0);
```

2.10 Performance Services

Plusieurs procédures stockées sont mises à la disposition des administrateurs bases de données, pour faciliter le travail de surveillance et d'optimisation des indexs.

Nous allons les passer en revue brièvement, je vous invite à vous reporter à la documentation officielle pour de plus amples informations :

http://www-01.ibm.com/support/knowledgecenter/api/content/ssw_ibm_i_72/rzajq/rzajqservicesperf.htm

ACT_ON_INDEX_ADVICE

La procédure ACT_ON_INDEX_ADVICE crée de nouveaux index pour une table en se basant sur les indexs conseillés par l'optimiseur SQL pour cette table.

Exemple d'utilisation :

```
CALL SYSTOOLS.ACT_ON_INDEX_ADVICE( 'PRODLIB', NULL, NULL, 1000, NULL );
```

HARVEST_INDEX_ADVICE

La procédure HARVEST_INDEX_ADVICE génère une ou plusieurs instructions de type CREATE INDEX dans les membres d'un fichier source, pour une table spécifiée, sur la base des indexs conseillés par SQL pour cette table.

REMOVE_INDEXES

La procédure REMOVE_INDEXES supprime les indexs correspondant aux critères spécifiés.

Exemple d'utilisation :

```
CALL SYSTOOLS.REMOVE_INDEXES( 'MYLIB', 1, '1 MONTH' ) ;
```

RESET_TABLE_INDEX_STATISTICS

La procédure RESET_TABLE_INDEX_STATISTICS efface les statistiques d'utilisation des index définis sur une ou plusieurs tables.

Exemple d'utilisation :

```
CALL qsys2.Reset_Table_Index_Statistics ( 'MJATST', 'AMON%' ) ;
```


On notera également la disponibilité, depuis la V6R1, de la procédure QSYS2.OVERRIDE_QAQQINI, qui peut être utilisée pour générer une table temporaire équivalente au fichier QAQQINI.

Exemple d'utilisation :

```
CALL QSYS2.OVERRIDE_QAQQINI(1, '', '') ;
```

2.11 Health Services

2.11.1 QSYS2.SYSLIMTBL

Un nouveau type d'indicateur de santé, le "suivi automatique des limites du système", est un dispositif mis à la disposition des administrateurs système, pour les aider à prévenir certaines situations de blocage.

L'outil met l'accent sur un sous-ensemble de limites du système (définies par IBM dans la table QSYS2.SQL_SIZING). Chaque fois que les limites définies dans cette table sont atteintes ou dépassées, des informations de suivi sont inscrites dans une table système DB2 appelée QSYS2/SYSLIMTBL. Une vue nommée QSYS2/SYSLIMITS est construite sur la table SYSLIMTBL, et permet d'obtenir rapidement de nombreux renseignements contextuels sur les lignes de la table.

Les différentes limites définies par IBM sont les suivantes (intitulés non traduits) :

ASP limits

- Maximum number of spool files

Database limits

- Maximum number of all rows in a partition
- Maximum number of valid rows in a partition
- Maximum number of deleted rows in a partition
- Maximum number of overflow rows in a partition
- Maximum number of variable-length segments
- Maximum number of indexes over a partition

File system limits

- Maximum number of object description entries in a library

Job limits

- Maximum number of rows locked in a unit of work
- Maximum number of row change operations in a unit of work

Journal limits

- Maximum size of a journal receiver
- Maximum number of objects that can be associated with a *MAX10M journal
- Maximum number of objects that can be associated with a *MAX250K journal
- Maximum sequence number of a *MAXOPT3 journal
- Maximum sequence number of a *MAXOPT1 or *MAXOPT2 journal

Object limits

Maximum number of members in a source physical file

System limits

Maximum number of jobs

Exemple 1. Examiner les travaux actifs au fil du temps

```
SELECT SBS_NAME, SIZING_NAME, CURRENT_VALUE, MAXIMUM_VALUE , A.*
FROM QSYS2.SYSLIMITS A
WHERE LIMIT_ID = 19000
ORDER BY CURRENT_VALUE DESC
```

SBS_NAME	SIZING_NAME	CURRENT_VAL...	MAXIMUM_VALUE	LAST_CHANGE_TIMESTAMP	LIMIT_CATEGORY	LIMIT_TYPE	SIZING
-	MAXIMUM NUMBER OF JOBS	1801	485000	2013-05-12 10:10:07.051744	WORK MANAGEMENT	SYSTEM	MAXIMU
-	MAXIMUM NUMBER OF JOBS	1401	485000	2013-05-12 10:09:42.928877	WORK MANAGEMENT	SYSTEM	MAXIMU
-	MAXIMUM NUMBER OF JOBS	1265	485000	2013-05-12 10:10:34.337091	WORK MANAGEMENT	SYSTEM	MAXIMU
-	MAXIMUM NUMBER OF JOBS	1001	485000	2013-05-11 10:27:37.403905	WORK MANAGEMENT	SYSTEM	MAXIMU

Exemple 2. Examiner les valeurs maximales définies par IBM par défaut :

```
SELECT SIZING_ID, SUPPORTED_VALUE, SIZING_NAME, COMMENTS
FROM QSYS2.SQL_SIZING
ORDER BY SIZING_ID DESC
```

SIZING_ID	SUPPORTED_VALUE	SIZING_NAME	COMMENTS
25005	10	MAXIMUM SYSTEM USER LENGTH	Maximum length of a system authorization ID
25004	10	MAXIMUM SESSION USER LENGTH	Maximum length of a session authorization ID
25003	-	MAXIMUM CURRENT ROLE LENGTH	-
25002	3483	MAXIMUM CURRENT PATH LENGTH	Maximum length of an SQL path
25001	-	MAXIMUM CURRENT TRANSFORM GROUP LENGTH	-
25000	-	MAXIMUM CURRENT DEFAULT TRANSFORM GROUP LENGTH	-
20004	32718	MAXIMUM DATALINK LENGTH	Maximum length of a datalink
20002	2097151	MAXIMUM STATEMENT OCTETS SCHEMA	Maximum length of an SQL data definition language (D
20001	2097151	MAXIMUM STATEMENT OCTETS DATA	Maximum length of an SQL data manipulation language
20000	2097151	MAXIMUM STATEMENT OCTETS	Maximum length of an SQL statement
19003	10000000	MAXIMUM NUMBER OF SPOOLED FILES IN EACH INDEPENDENT ASP	Maximum number of spooled files in each independent
19002	2610000	MAXIMUM NUMBER OF SPOOLED FILES IN THE SYSTEM AND BAS...	Maximum number of spooled files in the system and ba
19001	600000	MAXIMUM NUMBER OF SPOOLED FILES PER JOB	Maximum number of spooled files per job

2.11.2 Valeurs limites

Pour éviter une consommation trop excessive d'espace de stockage au niveau de la table de SYS2/SYSLIMITBL, DB2 for i va automatiquement supprimer des données selon différents critères.

DB2 for i fournit différentes variables globales, stockées dans le schéma SYSIBMADM :

Les valeurs fixées par IBM par défaut sont les suivantes :

Nom de variable globale	Valeur
QIBM_SYSTEM_LIMITS_PRUNE_BY_ASP	100
QIBM_SYSTEM_LIMITS_PRUNE_BY_JOB	50
QIBM_SYSTEM_LIMITS_PRUNE_BY_OBJECT	20
QIBM_SYSTEM_LIMITS_PRUNE_BY_SYSTEM	100
QIBM_SYSTEM_LIMITS_SAVE_HIGH_POINTS_BY_ASP	25
QIBM_SYSTEM_LIMITS_SAVE_HIGH_POINTS_BY_JOB	5
QIBM_SYSTEM_LIMITS_SAVE_HIGH_POINTS_BY_OBJECT	5
QIBM_SYSTEM_LIMITS_SAVE_HIGH_POINTS_BY_SYSTEM	25

Pour chaque type de limite, il y a deux variables globales. La variable de taille est utilisée pour choisir le nombre des entrées les plus récemment enregistrées devrait être conservé. La variable point haut permet de choisir le nombre d'entrées de la plus haute valeur de la consommation devrait être conservé.

Vous pouvez modifier ces limites en redéfinissant les valeurs de ces variables globales, via le code SQL suivant :

```
CREATE OR REPLACE VARIABLE SYSIBMADM.QIBM_SYSTEM_LIMITS_PRUNE_BY_SYSTEM  
    INTEGER  
    DEFAULT 50 ;
```

Dans l'exemple ci-dessus, on conserve les 50 lignes les plus récentes pour tous les types de limites du système.

Les modifications de valeurs sont prises en compte après le prochain IPL.

2.12 Support de JSON dans DB2

Disponibilité de cette fonctionnalité : TR2 (en V7R2) ou TR10 (en V7R1)



DB2 for i & JSON

▪ JSON DB2 Store - Technology Preview

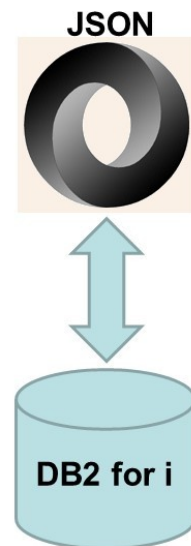
- Java applications will be able to use the DB2 JSON API to **store and retrieve JSON** as BLOB data from DB2 for i tables

What can be done...

- **Create** JSON collections (single BLOB column table)
- **Insert** JSON documents into a JSON collection
- **Retrieve** JSON documents
- **Interfaces**
 - ❑ DB2 JSON command line
 - ❑ DB2 JSON Java API
- **Convert** JSON documents from BLOB to character data with the SYSTOOLS.BSON2JSON() UDF

What are the details...

IBM i developerWorks article coming soon



<https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20i%20Technology%20Updates/page/JSON%20DB2%20Store>

3 Outils pour Développeurs SQL

3.1 Variables globales

Apparues avec la V7, les variables globales permettent de stocker dans l'emplacement de son choix des données qui peuvent être exploitées par les requêtes d'une base de données particulière.

Cette approche peut être particulièrement intéressante pour pouvoir distinguer plusieurs environnements d'exécution (test, recette, préproduction, production) se trouvant sur un même serveur et une même partition.

Exemple de variables globales définissant un environnement d'exécution :

```
CREATE VARIABLE MABIB.APP_TYP_ENV CHAR(3) DEFAULT 'R';
LABEL ON VARIABLE MABIB.APP_TYP_ENV IS 'environnement de recette';

CREATE VARIABLE MABIB.APP_COD_SOC CHAR(3) DEFAULT '010';
LABEL ON VARIABLE MABIB.APP_COD_SOC IS 'code société 010';
```

La lecture des variables globales se fait très simplement :

```
SELECT MABIB.APP_TYP_ENV FROM SYSIBM.SYSDUMMY1; -- Renvoie "R"
SELECT MABIB.APP_COD_SOC FROM SYSIBM.SYSDUMMY1; -- Renvoie "010"
```

Si on travaille avec des listes de bibliothèques (donc en syntaxe « système » au lieu de « SQL »), on doit faire abstraction de la bibliothèque de stockage des variables globales :

```
SELECT APP_TYP_ENV FROM SYSIBM.SYSDUMMY1; -- renvoie "R"
SELECT APP_COD_SOC FROM SYSIBM.SYSDUMMY1; -- renvoie "010"
```

Autre manière d'arriver au même résultat :

```
SELECT * FROM (VALUES(MABIB.APP_TYP_ENV)) VARIABLES(TYP_ENV) ;
```

La table QSYS2.SYSVARIABLES contient la liste des variables globales qui ont été créées sur le système. Il est dès lors facile de retrouver toutes les bases de données dans lesquelles une variable est déclarée, au moyen d'une requête du type :

```
SELECT * FROM QSYS2.SYSVARS WHERE VARIABLE_NAME = 'APP_COD_SOC' ;
```

A noter : la valeur associée à chaque variable est stockée dans la table SYSVARIABLES sous forme d'un pointeur. On ne peut donc pas visualiser cette information directement à partir de cette table.

La mise à jour d'une variable globale se fait de la façon suivante :

```
SET MABIB.APP_TYP_ENV = 'P' ;
```

On peut aussi alimenter le contenu d'une variable globale via une sous-requête scalaire :

```
SET MABIB.APP_TYP_ENV = (SELECT TYP_ENV FROM TABENV FETCH FIRST 1 ROW ONLY) ;
```


3.2 L'ordre MERGE

L'ordre MERGE est une avancée majeure du SQL DB2, apparue avec la V7R1 :

```
-- drop table My_LIBRARY.testmerge ;
create table My_LIBRARY.testmerge (
  macle char(10) default null,
  codea char(10) default null,
  coden integer default null
);

-- drop table My_LIBRARY.testmerge2 ;
create table My_LIBRARY.testmerge2 (
  macle char(10) default null,
  codea char(10) default null,
  coden integer default null
);
```

Plutôt que d'alimenter la table testmerge via un INSERT, autant le faire directement avec un premier MERGE. Dans ce cas de figure, les données ne viennent pas d'une table source mais de valeurs fixées "en dur", du coup la clause USING ne sert à rien, mais comme elle n'est pas optionnelle, on l'alimente avec la table pivot SYSDUMMY1 :

```
MERGE INTO My_LIBRARY.testmerge A
  USING (SELECT * FROM SYSIBM.SYSDUMMY1) B
  ON A.macle = 'CLE1'
  WHEN NOT MATCHED THEN
  INSERT ( a.macle , a.codea , a.coden )
  VALUES ( 'CLE1' , 'A1' , 1 )
;
```

Insérons une seconde ligne dans la table testmerge avec la même technique. On sait que l'on ne passera pas dans le bloc WHEN MATCHED, mais on l'a mis ici à titre de premier exemple de cette syntaxe :

```
MERGE INTO My_LIBRARY.testmerge A
  USING (SELECT * FROM SYSIBM.SYSDUMMY1) B
  ON A.macle = 'CLE2'
  WHEN MATCHED THEN
  UPDATE SET
    a.codea = 'A2' ,
    a.coden = a.coden + 1
  WHEN NOT MATCHED THEN
```

```
INSERT ( a.macle , a.codea , a.coden )  
VALUES ( 'CLE2' , 'A2' , 2 )  
;
```

Vérifions le contenu de la table testmerge :

```
select * from My_LIBRARY.testmerge ;
```

Vous pouvez vous amuser à réexécuter les 2 requêtes ci-dessus, et notamment la seconde en modifiant les valeurs fixées dans l'UPDATE, pour voir si vos modifications sont bien prises en compte.

Alimentons maintenant la table testmerge2 à partir du contenu de la table testmerge :

```
MERGE INTO My_LIBRARY.testmerge2 a  
  USING (SELECT macle , codea , coden FROM My_LIBRARY.testmerge ) b  
  ON a.macle = b.macle  
  WHEN MATCHED THEN  
    UPDATE SET  
      a.codea = b.codea ,  
      a.coden = a.coden + b.coden  
  WHEN NOT MATCHED THEN  
    INSERT ( a.macle , a.codea , a.coden )  
    VALUES ( b.macle , b.codea , b.coden )  
;
```

Vérifions le contenu de testmerge2 :

```
select * from My_LIBRARY.testmerge2 ;
```

Exemple avec des conditions dans le WHEN MATCHED, pour effectuer soit une suppression, comme dans l'exemple ci-dessous, où l'on supprime la ligne dans la table « testmerge2 » si elle existe déjà et a pour valeur « A1 » dans la colonne « codea ».

Pour compléter l'exemple, j'ai fait en sorte que des valeurs « en dur » soient affectées à la ligne mise à jour dans le cas où une ligne a la colonne « codea » fixée à « A2 ».

```
MERGE INTO My_LIBRARY.testmerge2 a
  USING (SELECT macle , codea , coden FROM My_LIBRARY.testmerge ) b
  ON a.macle = b.macle
  WHEN MATCHED and a.codea = 'A1' THEN
    DELETE
  WHEN MATCHED and a.codea = 'A2' THEN
    UPDATE SET
      a.codea = 'X2' ,
      a.coden = 9999
  WHEN MATCHED and a.codea <> 'A1' and a.codea <> 'A2' THEN
    UPDATE SET
      a.codea = b.codea ,
      a.coden = a.coden + b.coden
  WHEN NOT MATCHED THEN
    INSERT ( a.macle , a.codea , a.coden )
    VALUES ( b.macle , b.codea , b.coden )
;
```

Autre cas de figure qui peut être utilisé dans une procédure stockée (PL/SQL), un script PHP ou un programme RPG : on a des données de variables du programme (ou de la procédure stockée) et on veut insérer ces données dans une table sans passer par une table « source », on peut dans ce cas utiliser la table pivot SYSDDUMMY1 comme table source, et écrire ceci (les variables programmes sont reconnaissables au fait qu'elles sont préfixées par « : ») :

```
MERGE INTO My_LIBRARY.testmerge A
  USING (SELECT * FROM SYSDDUMMY1) B
  ON A.macle = :VARPGM1
  WHEN MATCHED THEN
    UPDATE SET
      a.codea = :VARPGM1 ,
      a.coden = :VARPGM2 + 1
  WHEN NOT MATCHED THEN
    INSERT ( a.macle , a.codea , a.coden )
    VALUES (:VARPGM1, :VARPGM2 , 1 )
;
```

Si l'on souhaite développer en SQL dynamique (technique utilisable en RPG, PL/SQL et PHP), on peut écrire ceci :

```
sql = "MERGE INTO qtemp.testmerge A
      USING (SELECT * FROM SYSIBM.SYSDUMMY1) B
      ON A.macle1 = ?
      WHEN MATCHED THEN
      UPDATE SET
        a.codea = ? ,
        a.coden = ?
      WHEN NOT MATCHED THEN
      INSERT ( a.macle , a.codea , a.coden )
      VALUES ( ? , ? , ? )" ;
EXEC SQL PREPARE s1 FROM :sql;
EXEC SQL EXECUTE s1
      USING :VARPGM1 , :VARPGM1 , :VARPGM2 :VARPGM1 , :VARPGM2 , 1
      ;
```

Mais la technique ci-dessus présente l'inconvénient d'obliger le développeur à disséminer les points d'interrogation dans les différents blocs WHEN MATCHED, WHEN NOT MATCHED, en les démultipliant.

Une approche plus pragmatique consiste à déclarer les variables programmes au sein d'une table virtuelle, déclarée au sein de la clause USING, comme dans l'exemple ci-dessous, pris dans la documentation officielle de DB2 pour Z/OS :

```
sql = "MERGE INTO employee AS t
      USING TABLE(VALUES(
        CAST (? AS CHAR(6)),
        CAST (? AS VARCHAR(12)),
        CAST (? AS CHAR(1)),
        CAST (? AS VARCHAR(15)),
        CAST (? AS SMALLINT),
        CAST (? AS INTEGER))
      ) s (empno, firstnme, midinit, lastname, edlevel, salary)
      ON t.empno = s.empno
      WHEN MATCHED THEN
        UPDATE SET
          salary = s.salary
      WHEN NOT MATCHED THEN
        INSERT
          (empno, firstnme, midinit, lastname, edlevel, salary)
          VALUES (s.empno, s.firstnme, s.midinit, s.lastname, s.edlevel,
s.salary)";
EXEC SQL PREPARE s1 FROM :sql;
EXEC SQL EXECUTE s1 USING '000420', 'SERGE', 'K', 'FIELDING', 18,
39580
;
```

Cette approche évite de disséminer, et surtout de démultiplier, les points d'interrogation au sein de la requête. On obtient ainsi une requête plus lisible et plus maintenable, que dans l'exemple précédent.

Le seul inconvénient que l'on pourrait trouver à l'utilisation du MERGE, c'est l'impossibilité de récupérer un « result set » des données impactées par une instruction de mise à jour. En effet, avec une instruction de type DELETE, INSERT ou UPDATE, il est possible de récupérer le "result set" relatif aux données mises à jour, en utilisant la syntaxe SQL DB2 suivante :

```
SELECT * FROM FINAL TABLE (INSERT ...)
```

Cette technique utilisant la clause « FINAL TABLE » peut être très utile pour récupérer la liste des identifiants créés par un INSERT SQL, ou tout simplement le dernier identifiant généré par une série d'INSERTs, on peut dans ce cas écrire une requête du genre :

```
SELECT MAX(ID) FROM FINAL TABLE (INSERT ...)
```

Le MERGE SQL ne peut pas être combiné avec la clause « FINAL TABLE », donc on ne peut pas récupérer le result set résultant d'un MERGE. C'est néanmoins un inconvénient mineur au vu des possibilités qu'apporte le MERGE.

3.3 Utilisation du XML avec XMLTABLE

3.3.1 SQL vers XML

Exemples de génération de données au format XML, à partir de données SQL.

-- Création d'une table exemple :

```
CREATE TABLE MY_LIBRARY.HQEMPLOYEE (  
  EMPID INTEGER NOT NULL PRIMARY KEY,  
  FIRSTNAME VARCHAR (10),  
  LASTNAME VARCHAR (10),  
  SALARY DECIMAL (8, 2),  
  MGRID INTEGER);
```

INSERT INTO MY_LIBRARY.HQEMPLOYEE VALUES

```
(1, 'John', 'Brett', 66000, 6),  
(2, 'Peter', 'Robert', 35000, 5),  
(3, 'Kim', 'Reynolds', 40000, 5),  
(4, 'Lindsey', 'Bowen', 80000, NULL),  
(5, 'Paul', 'Taylor', 80000, 4),  
(6, 'Tim', 'Johnson', 41000, 5),  
(7, 'Lauren', 'Brook', 36000, 5),  
(8, 'Smith', 'Wright', 34000, 4),  
(9, 'Mohan', 'Kumar', 50000, 5);
```

-- 1er exemple

```
SELECT XMLAGG(XMLROW(  
  empid,  
  FIRSTNAME concat ' ' concat LASTNAME as name_emp  
  OPTION ROW "employee" as ATTRIBUTES ) ) AS XML_DATA  
FROM MY_LIBRARY.HQEMPLOYEE e;
```

=> résultat obtenu :

```
<employee EMPID="1" NAME_EMP="John Brett"/><employee EMPID="2" NAME_EMP="Peter  
Robert"/><employee EMPID="3" NAME_EMP="Kim Reynolds"/><employee EMPID="4"  
NAME_EMP="Lindsey Bowen"/><employee EMPID="5" NAME_EMP="Paul Taylor"/><employee  
EMPID="6" NAME_EMP="Tim Johnson"/><employee EMPID="7" NAME_EMP="Lauren  
Brook"/><employee EMPID="8" NAME_EMP="Smith Wright"/><employee EMPID="9"  
NAME_EMP="Mohan Kumar"/>
```

```
-- 2ème exemple
SELECT
XMLROW (
empid,
FIRSTNAME concat ' ' concat LASTNAME as name_emp
OPTION ROW "employee" as ATTRIBUTES )
as XML_DATA
FROM MY_LIBRARY.HQEMPLOYEE e;
```

=> résultat obtenu :

```
<employee EMPID="1" NAME_EMP="John Brett"/>
<employee EMPID="2" NAME_EMP="Peter Robert"/>
<employee EMPID="3" NAME_EMP="Kim Reynolds"/>
<employee EMPID="4" NAME_EMP="Lindsey Bowen"/>
<employee EMPID="5" NAME_EMP="Paul Taylor"/>
<employee EMPID="6" NAME_EMP="Tim Johnson"/>
<employee EMPID="7" NAME_EMP="Lauren Brook"/>
<employee EMPID="8" NAME_EMP="Smith Wright"/>
<employee EMPID="9" NAME_EMP="Mohan Kumar"/>
```

```
-- 3ème exemple : renvoie le même résultat que la requête précédente
SELECT XMLSERIALIZE(
XMLROW (
empid,
FIRSTNAME concat ' ' concat LASTNAME as name_emp
OPTION ROW "employee" as ATTRIBUTES )
AS varchar(32000)
VERSION '1.0' -- paramètre optionnel
) as XML_DATA
FROM MY_LIBRARY.HQEMPLOYEE e;
```

3.3.2 XML vers SQL

On peut aussi utiliser des données au format XML stockées dans l'IFS et les mettre au format SQL :

```
SELECT X.*
FROM
XMLTABLE ('$d/dept/employee' passing XMLPARSE(DOCUMENT
GET_XML_FILE('/home/DUFEIL/test_dept.xml')) as "d"
COLUMNS
  empID      INTEGER      PATH '@id',
  firstname  VARCHAR(20)   PATH 'name/first',
  lastname   VARCHAR(25)   PATH 'name/last') AS X
;
```

Fichier test_dept.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<dept bldg="101">
  <employee id="901">
    <name>
      <first>John</first>
      <last>Doe</last>
    </name>
    <office>344</office>
    <salary currency="USD">55000</salary>
  </employee>
  <employee id="902">
    <name>
      <first>Peter</first>
      <last>Pan</last>
    </name>
    <office>216</office>
    <phone>905-416-5004</phone>
  </employee>
</dept>
<dept bldg="114">
  <employee id="903">
    <name>
      <first>Mary</first>
      <last>Jones</last>
    </name>
    <office>415</office>
    <phone>905-403-6112</phone>
    <phone>647-504-4546</phone>
    <salary currency="USD">64000</salary>
  </employee>
</dept>
```


-- exemple 2 :

```
SELECT * FROM XMLTABLE ('$result/document/data/element'  
PASSING XMLPARSE(DOCUMENT  
SYSTOOLS.HTTPGETBLOB('http://data.nantes.fr/api/publication/LOC_AIRES_COV_NM/  
LOC_AIRES_COV_NM_STBL/content/?format=xml', '' )  
) as "result"  
COLUMNS  
  nom CHAR(50) PATH 'geo/name',  
  cdpst CHAR(5) PATH 'CODE_POSTAL',  
  places INT PATH 'CAPACITE_VOITURE'  
) as RESULT  
;
```

NOM	CDPST	PLACES
Aire de covoiturage Piano'cktail	44340	350
Aire de covoiturage Le Paradis	44220	16
Aire de covoiturage Le Vélodrome	44220	160
Aire de covoiturage Les Ormeaux	44830	156
Aire de covoiturage Georges Brassens	44620	85
Aire de covoiturage La Herdrie	44115	120
Aire de covoiturage Pas Enchantés	44230	180
Aire de covoiturage Les 3 Brasseurs	44230	140
Aire de covoiturage La Charmelière	44470	100
Aire de covoiturage La Croix de Mauves	44470	40
Aire de covoiturage Sautron	44880	30

3.3.3 XSLT

Transformation de flux XML avec XSLT

Nécessite pour fonctionner d'avoir au préalable installé le XML Toolkit d'IBM

Exemple pris sur le site suivant (et légèrement adapté) :

<http://pic.dhe.ibm.com/infocenter/iseres/v7r1m0/index.jsp?topic=%2Fsqlp%2Frbafyxml3610.htm>

Voir aussi : <http://www.volubis.fr/news/liens/courshtm/XML/SQLXML.htm#level22>

```
DECLARE GLOBAL TEMPORARY TABLE XML_TAB (  
  DOCID INTEGER,  
  XML_DOC XML CCSID 1208,  
  XSL_DOC CLOB(1M) CCSID 1208  
) WITH REPLACE ;
```

```
INSERT INTO QTEMP.XML_TAB VALUES  
  (1,  
    '<students xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
      <student studentID="1" firstName="Steffen" lastName="Siegmund"  
        age="23" university="Rostock"/>  
    </students>',  
  
    '<?xml version="1.0" ?>  
<xsl:stylesheet version="1.0"  
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
<xsl:param name="headline"/>  
<xsl:param name="showUniversity"/>  
<xsl:template match="students">  
  <html>  
  <head/>  
  <body>  
  <h1><xsl:value-of select="$headline"/></h1>  
  <table border="1">  
  <th>  
  <tr>  
  <td width="80">StudentID</td>  
  <td width="200">First Name</td>  
  <td width="200">Last Name</td>  
  <td width="50">Age</td>  
  <xsl:choose>  
    <xsl:when test="$showUniversity = ''true''">  
      <td width="200">University</td>  
    </xsl:when>  
  </xsl:choose>  
  </tr>  
  </th>  
  <xsl:apply-templates/>  
  </table>  
  </body>  
  </html>  
</xsl:template>
```

```

        <xsl:template match="student">
            <tr>
                <td><xsl:value-of select="@studentID"/></td>
                <td><xsl:value-of select="@firstName"/></td>
                <td><xsl:value-of select="@lastName"/></td>
                <td><xsl:value-of select="@age"/></td>
                <xsl:choose>
                    <xsl:when test="$showUniversity = ''true'' ">
                        <td><xsl:value-of select="@university"/></td>
                    </xsl:when>
                </xsl:choose>
            </tr>
        </xsl:template>
    </xsl:stylesheet>'
);

```

```

SELECT XSLTRANSFORM (XML_DOC USING XSL_DOC AS CLOB(1M)) FROM QTEMP.XML_TAB;

```

Attention : la fonction XSLTRANSFORM nécessite pour fonctionner que soit installé le « IBM XML Toolkit for i » (5733XT2), ainsi que le « International Components for Unicode » (5770-SS1). Dans le cas contraire, une erreur SQL est renvoyée :

Etat SQL : 560CR

Code fournisseur : -7056

Message : [SQL7056] Prise en charge de la base de données XML non disponible pour la raison 1.
Cause : Un programme sous licence obligatoire n'est pas installé. Code raison : 1. 1 - IBM XML Toolkit for i (5733XT2) ou International Components for Unicode (5770-SS1) n'est pas installé. 2 - Java Developer Kit 5.0 (5770JV1), J2SE 5.0 32 bits (5770JV1), J2SE 5.0 64 bits (5770JV1) ou Portable App Solutions Environment (5770-SS1) n'est pas installé. Que faire . . . : Assurez-vous que les programmes obligatoires sous licence sont correctement installés. Renouvelez ensuite la demande.

3.3.4 Lecture de XML avec Namespace

Exemple pris sur le site suivant :

<http://www.ibm.com/developerworks/data/library/techarticle/dm-0708nicola/index.html>

```
SELECT X.*
FROM
  XMLTABLE (XMLNAMESPACES(DEFAULT 'http://www.w3.org/2001/XMLSchema-
instance'),
    '$d/*:Document/*:BkToCstmrDbtCdtNtfctn/*:Ntfctn/*:Acct'
    passing XMLPARSE(DOCUMENT GET_XML_FILE('/home/DUFEIL/testxml.xml') ) as
"d"
  COLUMNS
    iban CHAR(37) PATH '/*:Id/*:IBAN'
) AS X
for fetch only
;
```

Exemple de flux XML avec namespace stocké dans l'IFS (on souhaite ici récupérer le code IBAN) :

```
<?xml version="1.0" encoding="UTF-8"?>
- <Document xmlns="urn:iso:std:iso:20022:tech:xsd:camt.054.001.02"
  xsi:schemaLocation="urn:iso:std:iso:20022:tech:xsd:camt.054.001.02 camt.054.001.02.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <BkToCstmrDbtCdtNtfctn>
    + <GrpHdr>
    - <Ntfctn>
      <Id>30066109260001013340115060503283648</Id>
      + <CreDtTm>
      - <Acct>
        - <Id>
          <IBAN>FR7630066109260001013340190</IBAN>
        </Id>
        <Ccy>EUR</Ccy>
```

NB : Cette technique fonctionne avec le contrôle de validation *UR et *CHG.

3.4 Hiérarchie récursive

Exemple :

```
CREATE TABLE MY_EMP(  
  EMPID  INTEGER NOT NULL PRIMARY KEY,  
  NAME   VARCHAR(10),  
  SALARY DECIMAL(9, 2),  
  MGRID  INTEGER);  
  
INSERT INTO MY_EMP VALUES ( 1, 'Jones',    30000, 10);  
INSERT INTO MY_EMP VALUES ( 2, 'Hall',     35000, 10);  
INSERT INTO MY_EMP VALUES ( 3, 'Kim',      40000, 10);  
INSERT INTO MY_EMP VALUES ( 4, 'Lindsay',  38000, 10);  
INSERT INTO MY_EMP VALUES ( 5, 'McKeough', 42000, 11);  
INSERT INTO MY_EMP VALUES ( 6, 'Barnes',   41000, 11);  
INSERT INTO MY_EMP VALUES ( 7, 'O''Neil',  36000, 12);  
INSERT INTO MY_EMP VALUES ( 8, 'Smith',    34000, 12);  
INSERT INTO MY_EMP VALUES ( 9, 'Shoeman',  33000, 12);  
INSERT INTO MY_EMP VALUES (10, 'Monroe',   50000, 15);  
INSERT INTO MY_EMP VALUES (11, 'Zander',   52000, 16);  
INSERT INTO MY_EMP VALUES (12, 'Henry',    51000, 16);  
INSERT INTO MY_EMP VALUES (13, 'Aaron',    54000, 15);  
INSERT INTO MY_EMP VALUES (14, 'Scott',    53000, 16);  
INSERT INTO MY_EMP VALUES (15, 'Mills',    70000, 17);  
INSERT INTO MY_EMP VALUES (16, 'Goyal',    80000, 17);  
INSERT INTO MY_EMP VALUES (17, 'Urbassek', 95000, NULL);
```

-- 1er exemple : La requête suivante retourne tous les employés travaillant pour Goyal, ainsi que des informations supplémentaires, notamment la hiérarchie de chaque employé

```
SELECT NAME,
       LEVEL,
       SALARY,
       CONNECT_BY_ROOT NAME AS ROOT,
       SUBSTR(SYS_CONNECT_BY_PATH(NAME, ':'), 1, 25) AS CHAIN
FROM MY_EMP
START WITH NAME = 'Goyal'
CONNECT BY PRIOR EMPID = MGRID
ORDER SIBLINGS BY SALARY;
;
```

-- résultat obtenu

NAME	LEVEL	SALARY	ROOT	CHAIN
Goyal	1	80000.00	Goyal	:Goyal
Henry	2	51000.00	Goyal	:Goyal:Henry
Shoeman	3	33000.00	Goyal	:Goyal:Henry:Shoeman
Smith	3	34000.00	Goyal	:Goyal:Henry:Smith
O Neil	3	36000.00	Goyal	:Goyal:Henry:O Neil
Zander	2	52000.00	Goyal	:Goyal:Zander
Barnes	3	41000.00	Goyal	:Goyal:Zander:Barnes
McKeough	3	42000.00	Goyal	:Goyal:Zander:McKeough
Scott	2	53000.00	Goyal	:Goyal:Scott

-- 2ème exemple : Retour de la structure organisationnelle de la table DEPARTMENT, à partir du code département transmis dans la clause WHERE.

```
SELECT LEVEL, CAST(SPACE((LEVEL - 1) * 4) concat '/' || NAME
      AS VARCHAR(40)) AS NAME
FROM notos.MY_EMP
START WITH NAME = 'Urbassek'
CONNECT BY NOCYCLE PRIOR EMPID = MGRID
;
```

-- résultat obtenu

LEVEL	NAME
1	/Urbassek
2	/Goyal
3	/Scott
3	/Henry
4	/Shoeman
4	/Smith
4	/O'Neil
3	/Zander
4	/Barnes
4	/McKeough
2	/Mills
3	/Aaron
3	/Monroe
4	/Lindsay
4	/Kim
4	/Hall
4	/Jones

3.5 Evolution du Timestamp

Pour rappel, voici un exemple de timestamp renvoyé par DB2 jusqu'à la version 7.1 : 2015-06-05-14.28.09.406890

A partir de la V7R2, le type de données TIMESTAMP bénéficie d'une amélioration, puisque le développeur peut spécifier quelle partie fractionnaire d'une seconde (0-12) doit être conservée. Une précision fractionnaire de zéro signifie que l'horodatage (ou « timestamp ») stocke une date et une heure à la seconde près. Une précision fractionnaire de trois signifie que l'horodatage peut stocker une valeur à la milliseconde près. Une précision fractionnaire de 12, signifie que l'horodatage stocke l'information avec une précision à la picoseconde près.

La précision fractionnaire est spécifiée sur le type de données TIMESTAMP entre parenthèses, comme indiqué ci-dessous :

```
CREATE TABLE DEV.TS_TEST (  
T1 TIMESTAMP(0) NOT NULL DEFAULT CURRENT_TIMESTAMP,  
T2 TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,  
T3 TIMESTAMP(6) NOT NULL DEFAULT CURRENT_TIMESTAMP,  
T4 TIMESTAMP(9) NOT NULL DEFAULT CURRENT_TIMESTAMP,  
T5 TIMESTAMP(12) NOT NULL DEFAULT CURRENT_TIMESTAMP  
);  
INSERT INTO DEV.TS_TEST  
VALUES (DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT);
```

Les valeurs insérées sont montrés ici avec la longueur de chaque colonne TIMESTAMP :

T1	2014-05-16-20.46.23
T2	2014-05-16-20.46.23.898
T3	2014-05-16-20.46.23.898633
T4	2014-05-16-20.46.23.898633773
T5	2014-05-16-20.46.23.898633773437

Le manuel de référence SQL actuelle ne donne pas beaucoup d'informations sur ce point, mais un synonyme de CURRENT_TIMESTAMP est LOCALTIMESTAMP qui fonctionne de manière similaire :

```
VALUES (CURRENT_TIMESTAMP(2), LOCALTIMESTAMP(8))
```

Il convient d'être prudent dans l'utilisation du timestamp tel qu'il a évolué, notamment lors de son utilisation avec des outils d'interrogation tiers (connecteur JDBC, OLE DB, etc...) car certains drivers peuvent ne pas le supporter (surtout s'ils sont trop anciens).

Le manuel de référence SQL indique que la taille de TIMESTAMP varie de 7 à 13 octets, mais dans la pratique, un examen par DSPFFD semble indiquer que la taille réelle oscille entre 19 et 32 octets selon la précision retenue.

3.6 Result Set et procédures stockées

La fonctionnalité présentée dans ce chapitre a été annoncée officiellement en V7R1 mais en réalité elle était déjà disponible – officieusement - en V6R1.

La production de "result set" à l'intérieur de programmes RPG, encapsulés dans des procédures stockées de type externe, est identique au principe utilisé dans les procédures stockées « full SQL ».

Voici à titre d'exemple un extrait de code RPG provenant d'une étude de cas :

```

//*****
// Si demandé par le programme appelant,
// génération d'un result set à partir de la
// table temporaire
//*****
If ( Resultset = 'YES' ) ;
    sql3 = 'SELECT distinct Job_name, Job_user_name,
Job_number ' +
        'FROM QTEMP/OBJL0100 FOR FETCH ONLY ' ;
    EXEC SQL
        PREPARE REQ1 FROM :sql3 ;
    EXEC SQL
        DECLARE C1 CURSOR FOR REQ1 ;
    EXEC SQL
        OPEN C1 ;
    EXEC SQL
        SET RESULT SETS CURSOR C1 ;
Endif ;
```

Dans l'exemple ci-dessus, on a pris soin d'ajouter au programme RPG un paramètre optionnel, que nous avons appelé « Resultset », permettant de définir si l'on souhaite que le programme RPG renvoie un « result set » ou pas. On voit ici que la production du « result set » consiste à ouvrir un curseur sur une requête SQL lisant le contenu d'une table temporaire générée par le programme RPG lui-même.

Dans la copie d'écran ci-dessous, tirée d'une étude de cas, on voit que la procédure stockée externe MODHERASQ2 encapsule un programme RPG MODHERA2, et que le fait d'appeler cette procédure par un simple CALL permet de récupérer en sortie le « result set » produit par le programme RPG :

```

DROP PROCEDURE JARRIGE/MODHERASQ2;
CREATE PROCEDURE JARRIGE/MODHERASQ2 (
    INOUT NADHR CHAR(6) CCSID 297,
    INOUT ERREUR CHAR(1) CCSID 297,
    INOUT HORO CHAR(26) CCSID 297)
    DYNAMIC RESULT SETS 1
    LANGUAGE RPGLE
    SPECIFIC JARRIGE/MODHERASQ2
    NOT DETERMINISTIC
    MODIFIES SQL DATA
    CALLED ON NULL INPUT
    EXTERNAL NAME 'JARRIGE/MODHERA2'
    PARAMETER STYLE SQL ;

COMMENT ON SPECIFIC PROCEDURE JARRIGE/MODHERASQ2
    IS 'Procédure externe pour pgm MODHERA' ;

call JARRIGE/MODHERASQ2 ('000002', '', '');

```

HORO	RES
2015-04-13 11:32:56.638000	<GPM><ADHERENTS><ADHERENT><IDACTEUR>9</
<div> <div></div> <div></div> </div>	
<div> <div>Messages</div> <div>call JARRIGE/MODHERASQ2 ('000002', '', '')</div> </div>	

3.6 La syntaxe « OR REPLACE »

Avec la V7R1, il devient possible de créer (ou recréer) des objets DB2 en utilisant la syntaxe "CREATE OR REPLACE", telle qu'elle est présentée dans l'exemple ci-dessous.



```
CREATE OR REPLACE PROCEDURE FORMATION/MA_PROC (  
    IN VCODSOC DECIMAL(3, 0) ,  
    IN VPERIOD CHAR(1) )  
...
```

Vous pouvez utiliser la syntaxe OR REPLACE sur :

- les vues SQL,
- les triggers,
- les fonctions,
- les procédures,
- les variables globales,
- les séquences,
- les alias.

En plus de faciliter la recréation d'objets existants, cette syntaxe offre l'avantage très appréciable de recréer l'objet considéré en conservant les droits tels qu'ils étaient définis sur cet objet avant sa recréation.


A partir de la TR2 (en V7R2) et de la TR10 (en V7R1), le support de CREATE OR REPLACE est étendu à la gestion de tables :



Create OR REPLACE Table

Data Definition Language (DDL) SQL statements that support the optional 'OR REPLACE' clause:

- ☐ CREATE OR REPLACE ALIAS
- ☐ CREATE OR REPLACE FUNCTION
- ☐ CREATE OR REPLACE MASK
- ☐ CREATE OR REPLACE PERMISSION
- ☐ CREATE OR REPLACE PROCEDURE
- ☐ CREATE OR REPLACE SEQUENCE
- ☐ **CREATE OR REPLACE TABLE**
- ☐ CREATE OR REPLACE TRIGGER
- ☐ CREATE OR REPLACE VARIABLE
- ☐ CREATE OR REPLACE VIEW



Replacing a table:

- ✓ Data-Centric
- ✓ Dependent Views & MQTs preserved
- ✓ Auditing preserved
- ✓ Authorizations preserved
- ✓ Comments and Labels preserved
- ✓ Triggers preserved
- ✓ Rows optionally deleted

Knowledge Center

http://www-01.ibm.com/support/knowledgecenter/ssw_ibm_i_72/db2/rbafzhctabl.htm?lang=en

Article for previous OR REPLACE statements

<http://iprodeveloper.com/database/use-sql-create-or-replace-improve-db2-i-object-management>

Lors du remplacement d'une table, DB2 fournit plusieurs modes de remplacement, via l'option ON REPLACE clause. On a 3 options à notre disposition :

1. ON REPLACE PRESERVE ALL ROWS (default)

C'est l'option la plus sûre, car elle garantit de récupérer toutes les lignes de la table après remplacement. Lors de cette opération, certaines colonnes pourront être ajoutées, supprimées ou modifiées.

2. ON REPLACE PRESERVE ROWS

Si la table n'est pas partitionnée, alors les options PRESERVE ALL ROWS et PRESERVE ROWS fournissent le même résultat.

Dans le cas de tables partitionnées, les données des lignes partitionnées selon les mêmes critères sont conservées, les autres lignes sont perdues, sans déclencher de trigger. Lors de cette opération, certaines colonnes pourront être ajoutées, supprimées ou modifiées.

3. ON REPLACE DELETE ROWS

Toutes les lignes sont supprimées et aucun trigger n'est exécuté.

Restrictions :

- Les colonnes ne peuvent être supprimées si des vues, indexs, trigger, requêtes de MQT, des permissions, masques ou contraintes sont définies sur ces colonnes.
- Les colonnes ayant des contraintes de type « Unique » ne peuvent être supprimées si des contraintes de type « clé étrangère » sont liées à ces colonnes.

3.7 Les paramètres de procédures

A partir de la V7R1, il devient possible de définir des valeurs par défaut pour certains paramètres d'une procédure stockée, comme dans l'exemple suivant :

```
CREATE OR REPLACE PROCEDURE FORMATION/MA_PROC (  
    INOUT NOEG CHAR(3) DEFAULT '0',  
    INOUT NOADH CHAR(6) DEFAULT '0',  
    INOUT NOENT CHAR(4) DEFAULT '9999',  
    INOUT NOSENT CHAR(2) DEFAULT '99',  
    INOUT DATEFF CHAR(10) DEFAULT '0001-01-01')  
...
```

La V7R1 offre également la possibilité de nommer les paramètres dans le CALL de la procédure. Ainsi, et en profitant des valeurs par défaut définies sur les paramètres d'entrée, il est possible de remplacer l'appel de procédure suivant, dans lequel tous les paramètres sont déclarés :

```
CALL FORMATION/MA_PROC ('0', '0', '9999', '99', '0001-01-01');
```

... par l'appel simplifié suivant :

```
CALL FORMATION/MA_PROC (NOEG => '1', NOENT => '4564', DATEFF => '2015-04-15');
```

Explication : dans l'exemple du CALL ci-dessus, on ne déclare que les paramètres devant recevoir des valeurs différentes de leurs valeurs par défaut. De plus, à partir du moment où chaque paramètre utilisé est nommé explicitement, l'ordre dans lequel les paramètres sont passés dans le CALL devient indifférent.

3.8 L'ordre TRUNCATE TABLE

Apparu en V7R2, l'ordre TRUNCATE TABLE est utilisé pour effectuer une suppression de l'intégralité des lignes d'une table. Il s'apparente donc à un DELETE effectué sans clause WHERE (ou à la commande système CLRPFM), mais il offre davantage d'options que le DELETE. De plus, si la commande CLRPFM ne fonctionne que sur le nom court des tables DB2, TRUNCATE TABLE travaille en revanche avec le nom long des objets considérés.

TRUNCATE TABLE MA_TABLE ;

Les premiers tests effectués avec cette instruction donnent à penser que les performances sont sensiblement équivalentes à celle d'un DELETE sans clause WHERE. L'intérêt de cette nouvelle instruction réside donc principalement dans sa compatibilité avec les autres déclinaisons de DB2 (pour LUW et Z/OS), ainsi que dans le jeu d'options qui lui sont rattachées et que nous allons détailler ci-dessous.

DROP STORAGE (default) ou REUSE STORAGE

DROP STORAGE va avoir pour effet de récupérer l'intégralité de la place allouée aux enregistrements supprimés. A l'inverse, REUSE STORAGE va supprimer les lignes mais sans récupérer l'espace libéré.

IGNORE DELETE TRIGGERS (default) ou RESTRICT WHEN DELETE TRIGGERS

La première option correspond au comportement par défaut, et elle consiste à supprimer les lignes d'une table sans faire appel aux déclencheurs de suppression définis sur cette table. Si l'option RESTRICT est sélectionnée, et si des déclencheurs de suppression sont définis sur une table, alors la suppression est interrompue et une erreur est renvoyée par le moteur SQL. Pour effectuer un vidage avec déclenchement Si vous voulez que votre déclencheur de suppression fonctionne normalement, alors utilisez le DELETE traditionnel.

CONTINUE IDENTITY (default) ou RESTART IDENTITY

Dans le cas d'une table utilisant une colonne identifiant en incrémentation automatique, cette option permet d'indiquer si on souhaite que l'identifiant soit réinitialisé à sa valeur d'origine à l'issue du TRUNCATE, ou conserve la dernière valeur connue pour les prochaines insertions.

IMMEDIATE

Cette option permettra d'empêcher toute utilisation de ROLLBACK à l'issue du TRUNCATE. Sans cette option, un ROLLBACK serait possible.

L'instruction TRUNCATE TABLE suivante efface la table MA_TABLE, en récupérant la place correspondants aux lignes supprimées, et en réinitialisant la colonne identifiant, et en inhibant toute possibilité de ROLLBACK sur les données supprimées :

```
TRUNCATE TABLE MA_TABLE  
    DROP STORAGE RESTART IDENTITY IMMEDIATE ;
```


3.9 Pagination avec DB2

ATTENTION : contrairement à ce que de nombreux développeurs SQL pensent, le code suivant ne permet pas de gérer une pagination sous DB2 :

```
SELECT * FROM matable FETCH FIRST 10 ROWS ONLY
```

*Le code SQL ci-dessus renvoie les 10 premières lignes du jeu de données identifié par le SGBD.
Cette technique est totalement inadaptée à la gestion de listes avec pagination.*

Avec MySQL, on peut gérer facilement une pagination au moyen du code SQL suivant :

```
SELECT * FROM matable LIMIT 10, 20
```

(renvoie les lignes 10 à 20 du jeu de données identifié par le SGBD)

Avec DB2, c'est un peu plus compliqué :

Voici un exemple de requête simple ne gérant pas pour l'instant de pagination :

```
SELECT A.* FROM MATABLE A WHERE A.COL1 = ? AND A.COL2 = ?
```

Voici les modifications à opérer pour que la requête précédente soit en mesure de gérer la pagination, qui est effectuée ici par plages de 10 lignes :


```
SELECT foo.* FROM (  
    SELECT row_number() over (ORDER BY TABLE_NAME) as rn,  
    A.*  
    FROM MATABLE A  
    WHERE A.COL1 = ? AND A.COL2 = ?  
) AS foo  
WHERE foo.rn BETWEEN ? AND ?
```


La technique « full SQL » présentée ci-dessus donne de bons résultats sur des tables de taille raisonnable (difficile de donner un chiffre précis car cela dépend beaucoup de la puissance du (des) processeurs(s) de votre serveur IBM i). Mais elle présente quelques défauts :

Elle est « intrusive » dans le sens où elle nécessite de modifier la requête SQL pour y insérer un certain nombre d'éléments (modification du début du SELECT, inclusion du tri dans la clause OVER...).

Avec cette technique, DB2 a tendance à s'effondrer sur les tables de grande taille, donc si vous rencontrez des difficultés avec cette technique, on ne pouvait que recommander l'utilisation d'un curseur scrollable (technique utilisable aussi bien avec PDO qu'avec ibm_db2).

A partir de DB2 V7R2 TR3, et DB2 V7R1 TR11, il devient possible de gérer la pagination plus simplement via l'arrivée des clauses LIMIT et OFFSET, comme dans les exemples suivants :



Welcome to the Waitless World 

PAGINATION


MOBILE
SQL
STATELESS
DB2 for i
LIMIT


LIMIT and OFFSET

- LIMIT and OFFSET support is popular, but non-standard. The DB2 Family recently decided to add the support.
- This style of data access is most useful for those cases where you only need a subset (page) of rows.
- The **offset-clause** is only allowed as part of the **outer fullselect** of a DECLARE CURSOR statement or a prepared *select-statement*.
- Initially, there is no support in STRSQL. That restriction may be removed later.

Syntax	Alternative Syntax	Action
LIMIT x	FETCH FIRST x ROWS ONLY	Return the first x rows
LIMIT x OFFSET y	OFFSET y ROWS FETCH FIRST x ROWS ONLY	Skip the first y rows and return the next x rows
LIMIT y,x	OFFSET y ROWS FETCH FIRST x ROWS ONLY	Skip the first y rows and return the next x rows

© 2015 IBM Corporation



Welcome to the Waitless World 

LIMIT and OFFSET

```

CREATE OR REPLACE PROCEDURE TOYSTORE.FIND_EMPLOYEES
(IN P_PAGESIZE BIGINT, IN P_OFFSET BIGINT)
  DYNAMIC RESULT SETS 1
  LANGUAGE SQL
BEGIN
  DECLARE V_PREP_STMT1 VARCHAR(4096) ;
  DECLARE CEMP_RESULT_SET1 CURSOR
    WITH RETURN FOR PREP_STMT1;
  SET V_PREP_STMT1 =
    'SELECT EMPNO, HIREDATE, LASTNAME FROM
     toystore.employee
     ORDER BY hiredate DESC
     LIMIT ? OFFSET ?';
  PREPARE PREP_STMT1 FROM V_PREP_STMT1 ;
  OPEN CEMP_RESULT_SET1 USING P_PAGESIZE, P_OFFSET;
END;

CALL TOYSTORE.FIND_EMPLOYEES(10, 0);
CALL TOYSTORE.FIND_EMPLOYEES(10, 10);
        
```

© 2015 IBM Corporation

Page 1

Page 2

EMPNO	HIREDATE	LASTNAME
0000220	09/30/80	PULASKI
0000070	09/30/80	PULASKI
000100	06/19/80	SPENSER
000290	05/30/80	PARKER
200240	12/05/79	MONTEVERDE
000240	12/05/79	MARINO
000210	04/11/79	JONES
200170	09/15/78	YAMAMOTO
000170	09/15/78	YOSHIMURA
000160	10/11/77	PIANKA
200140	12/15/76	NATZ
000140	12/15/76	NICHOLLS
200330	02/23/76	WONG
000330	02/23/76	LEE
000260	09/11/75	JOHNSON
000030	04/05/75	KWAN
000190	07/26/74	WALKER
000020	10/10/73	THOMPSON
000060	09/14/73	STEIN
000180	07/07/73	SCOUTTEN
000300	06/19/72	SMITH
200120	05/05/72	ORLANDO
000150	02/12/72	ADAMSON
000130	07/28/71	QUINTANA
000090	08/15/70	HENDERSON
000250	10/30/69	SMITH
200220	08/29/68	JOHN
000220	08/29/68	LUTZ
200280	03/24/67	SCHWARTZ
000280	03/24/67	SCHNEIDER
000230	11/21/66	JEFFERSON
000200	03/03/66	BROWN
000320	07/07/65	MEHTA

4 DB2 et la sécurité des données

4.1 Sécuriser les données avec les Field Procedure

Apparu en V7R1, les Field Procedure (en abrégé : « fieldproc ») permettent d'assurer le cryptage des données d'une colonne, soit totalement, soit partiellement.

Le cryptage est géré par un programme d'exit (écrit en RPG), appelé à chaque action sur la colonne (insert/update/read). Le programme d'exit se comporte comme une sorte de trigger qui serait affecté à une colonne.

On peut ajouter un fieldproc via un ALTER TABLE, ou un CREATE TABLE.

Un seul *field procedure* par colonne.

Structure du programme appelé :

- Le programme appelé est un *PGM ILE
 - Pas d'OPM, pas de *SRVPGM, pas de Java
 - Pas de SQL autorisé, pas de ACTGRP(*NEW)
- Reçoit 9 paramètres
- Assez complexe à écrire
- Pour un exemple de programme RPG de type FIELD_PROC :

<http://www.mcpressonline.com/database/db2/enable-transparent-encryption-with-db2-field-procedures.html>

Exemple : cryptage des 4 premiers caractères du n° carte

```
CREATE TABLE dgayte.fieldproc(  
  z1 INT,  
  z2 CHAR(16));
```

```
ALTER TABLE dgayte.fieldproc  
ALTER COLUMN Z2 SET FIELDPROC dgayte.field_proc;
```

```
INSERT INTO dgayte.fieldproc VALUES(1, '123456789012345');  
INSERT INTO dgayte.fieldproc VALUES(1, '3210654987123122');
```

```
SELECT * FROM dgayte.fieldproc;
```

Selon l'utilisateur

Z1	Z2
1	*****56789012345
2	*****654987123122

Z1	Z2
1	123456789012345
2	3210654987123122

Z1	Z2
1	123456789012345
2	3210654987123122

Dans System i Navigator :

Nom de colonne : Z2

Nom de système : Z2

Type de données : CHARACTER

Longueur : 16

Codage : Utilisation de CCSID CCSID : 297

Texte :

Ligne d'en-tête 1 : Z2

Ligne d'en-tête 2 :

Ligne d'en-tête 3 :

☒ Valeur indéfinie admise

☐ Implicitement masquée

Valeur par défaut : Valeur indéfinie

☒ Implémente une procédure de zone

Spécification de procédure de zone

Schéma : DGAYTE

Programme : FIELD_PROC

Liste des paramètres :

OK Annulation Aide ?

Via un DSPFFD :

```

Informations de niveau zone
Zone      Type      Long   Long   Position  Usage  En-tête
          données   zone   tampon tampon      zone  colonne
Z1        BINAIR    9  0    4        1    E-S    Z1
  Accepte la valeur indéfinie
Z2        ALPHA    16     16     5    E-S    Z2
  Accepte la valeur indéfinie
ID codé de jeu de caractères . . . . . : 297
Nom procédure zone . . . . . : FIELD_PROC
Biblio. procédure zone . . . . . : DGAYTE

```

4.2 Contrainte Violation

DB2 en V7R2 apporte une nouvelle clause VIOLATION sur les contraintes de type CHECK :

- ON INSERT VIOLATION SET column-name = DEFAULT

- L'erreur n'est pas signalée, la valeur par défaut est insérée

- ON UPDATE VIOLATION PRESERVE column-name

- L'erreur n'est pas signalée, la valeur précédente est conservée

Exemple :

```
drop table gjarrige.tstcheck;
```

```
create table gjarrige.tstcheck (  
  cle int as identity,  
  libelle char(50),  
  verifOK char(1) default 'o' check(verifok in ('o', 'n'))  
    on insert violation set verifok = DEFAULT  
    on update violation preserve verifOK  
);
```

```
insert into gjarrige.tstcheck (libelle, verifOK)  
values  
( 'ligne 1', 'o' ) ,  
( 'ligne 2', 'x' ) ,  
( 'ligne 3', 'n' ) ;
```

```
update gjarrige.tstcheck set verifOK = 'X' where libelle = 'ligne 3' ;
```

```
select * from gjarrige.tstcheck;
```

CLE	LIBELLE	VERIFOK
1	ligne 1	o
3	ligne 2	o
4	ligne 3	n

4.3 Row and Column Access Control (RCAC)

Disponible sur DB2 for i à partir de la V7R2, RCAC s'installe avec l'option 47 du logiciel sous licence 5770SS1 (non facturable).

RCAC permet de limiter l'accès à certaines données de type ligne et/ou colonne, aux seules personnes, ou groupes de personnes, qui sont habilitées à connaître le contenu de ces données.

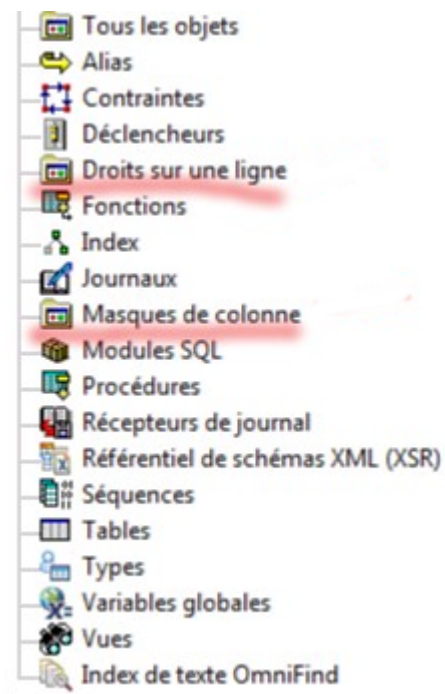
Dans DB2 for i, RCAC est implémenté en utilisant 2 approches différentes et complémentaires :

- Des permissions sur lignes
- Des masques sur colonnes

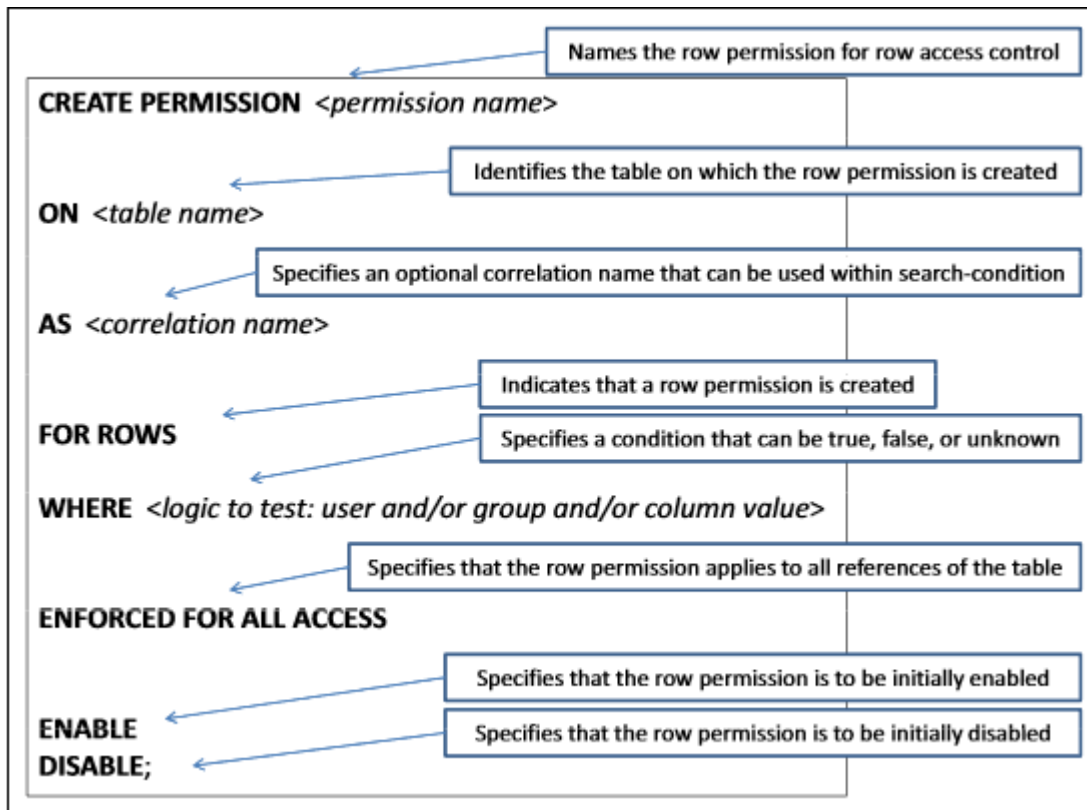
Même les utilisateurs qui ont des droits *ALLOBJ ne peuvent passer outre les autorisations qui ont été définies au travers de RCAC.

RCAC est basé sur des règles spécifiques qui sont transparentes pour les applications utilisant les bases de données, qu'il s'agisse d'applications « métier » ou de logiciels de type « client SQL ». Ces règles s'appliquent donc sans qu'il soit nécessaire d'apporter des applications utilisant les base de données concernées.

D'un point de System i Navigator, l'arrivée de RCAC se traduit par l'apparition de 2 nouvelles options :

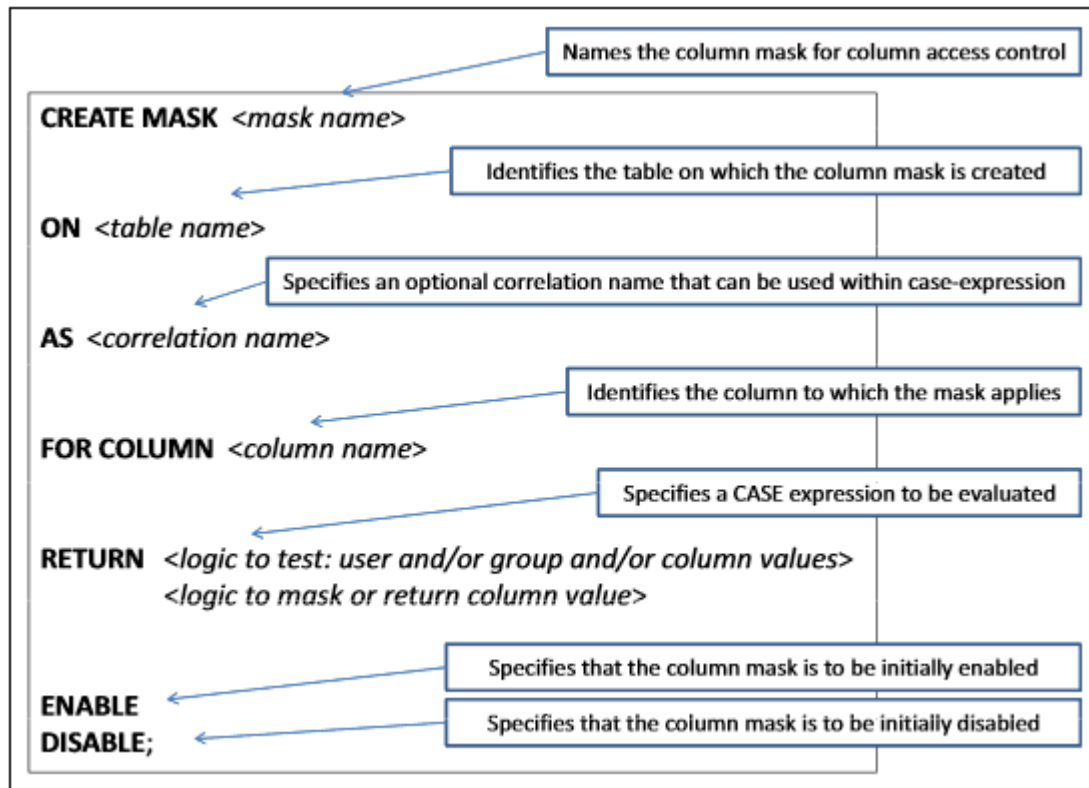


Une permission sur ligne est un objet DB2 créé selon le principe suivant :



(image extraite du redbook REDP-5110-00)

Un masque est un objet DB2 créé selon le principe suivant :



(image extraite du redbook REDP-5110-00)

Exemple de masque :

```
CREATE [OR REPLACE] MASK cc_mask ON client
FOR COLUMN credit_card RETURN
CASE
  WHEN
    SESSION_USER = 'QSECOFR'
    THEN credit_card
  WHEN
    VERIFY_GROUP_FOR_USER(SESSION_USER, 'ADMIN_CPT') = 1
    THEN credit_card
  ELSE
    'XXXXXX' CONCAT SUBSTR(credit_card, 8, 3)
END
ENABLE;

ALTER TABLE client ACTIVATE COLUMN ACCESS CONTROL;
```

Pour la désactivation temporaire des masques s'appliquant à une table :

```
ALTER TABLE client DEACTIVATE COLUMN ACCESS CONTROL;
```

Exemple de permissions sur lignes :

```
CREATE PERMISSION access_to_row ON bibl.client
FOR ROWS WHERE
  SESSION_USER = 'QSECOFR'
OR
  (VERIFY_GROUP_FOR_USER (SESSION_USER, 'ADMIN_CPT') = 1 )
OR
  (SESSION_USER = 'COMMERCIAL' AND appel_code IS NOT NULL)
ENFORCED FOR ALL ACCESS ENABLE ;
```

```
ALTER TABLE client ACTIVATE ROW ACCESS CONTROL;
```

Pour la désactivation temporaire des permissions s'appliquant à une table :

```
ALTER TABLE client DEACTIVATE ROW ACCESS CONTROL;
```

Tant que l'on n'a pas activé les droits par ALTER TABLE, masques et permissions sont inopérants.

A partir du moment où le « Row Access Control » est activé sur une table (via l'ALTER TABLE), seules les lignes répondant aux permissions définies peuvent être retournées aux utilisateurs concernés. En résumé, tout ce qui n'est pas autorisé est interdit.

Une PERMISSION, peut empêcher une insertion ou une mise à jour, qui ne respecte pas la (ou les) règle(s) définie(s).

Pour modifier une règle : ALTER MASK | PERMISSION

Pour supprimer une règle : DROP MASK | PERMISSION

En V7R2, l'ordre ALTER TRIGGER s'enrichit de quelques paramètres liés à RCAC :

- ENABLE : le trigger est actif (dft)
- DISABLE : le trigger n'est plus actif
- SECURED : le trigger est sécurisé (compatible) avec les droits RCAC
- NOT SECURED : le trigger n'est pas compatible avec les droits RCAC (valeur par défaut). Il est impossible de modifier cet attribut sur un trigger existant quand des droits RCAC sont actifs. De même, il est impossible de créer un trigger NOT SECURED quand des droits RCAC sont actifs.

Pour voir la liste des droits RCAC existants, regarder le contenu des tables systèmes SYSCONTROLS et SYSCONTROLSDEP de QSYS2.

Les droits RCAC sont stockés dans la table elle même, ils sont donc sauvegardés par SAVLIB et SAVOBJ, déplacés par MOVOBJ, dupliqués (par défaut) par CRTDUPOBJ.

Une table (ou fichier physique) avec des droits RCAC ne peut pas être sauvegardée dans une version d'OS précédente. Une table (ou fichier physique) avec des droits RCAC, restaurée sur un système ne possédant pas l'option 47 ne peut plus être ouverte.

5 Compléments

5.1 DSPFFD amélioré et autres outils

Je rappelle que vous disposez, dans la bibliothèque QSYS2, d'un jeu de tables systèmes dont les noms se passent de commentaire :

- SYSTABLES
- SYSCOLUMNS
- SYSVIEWS
- SYSVIEWDEP
- SYSROUTINES
- SYSROUTINEDEP
- SYSTABLESTAT
- etc..

Vous pouvez par exemple vous appuyer sur la table SYSTABLES pour vérifier qu'une même table se trouvant dans plusieurs bibliothèques a bien la même structure dans chacune de ces bibliothèques.

Exemple de vue obtenue à partir d'un outil "maison" développé en PHP : affichage de toutes les vues DB2 dont le nom est SYSTABLES, quelles que soient les bibliothèques où elles se trouvent :

Schéma	Table (sqlname)	Table (sysname)	Type	Nb.Cols.	Buffer	Description
GJABASE	SYSTABLES	SYSTABLES	V	29	3003	Vue du rép des données SQL
GJARRIGE	SYSTABLES	SYSTABLES	V	29	3003	Vue du rép des données SQL
GJARRIGE2	SYSTABLES	SYSTABLES	V	29	3003	Vue du rép des données SQL
GJABASE2	SYSTABLES	SYSTABLES	V	29	3003	Vue du rép des données SQL
OPENCART2	SYSTABLES	SYSTABLES	V	29	3003	Vue du rép des données SQL
GJARRIGE3	SYSTABLES	SYSTABLES	V	30	5091	Vue du rép des données SQL
SAMPLES	SYSTABLES	SYSTABLES	V	30	5091	Vue du rép des données SQL

Le tableau ci-dessus est produit au moyen d'une requête sur la vue SYSTABLES justement.

On voit que dans certains cas SYSTABLES contient 29 colonnes, et dans d'autres cas 30 colonnes (ce qui impacte aussi le buffer). Comparer le nombre de colonnes et le buffer constitue un moyen pratique et rapide de détecter des écarts entre bases de données.

ATTENTION : la comparaison de vues et de MQT sur la notion de buffer peut prêter à confusion. Certaines vues ou MQT, strictement identiques d'un point de vue du code source, peuvent présenter des buffers différents, dès lors qu'elles utilisent des fonctions d'agrégation (SUM par exemple) et/ou des formules de calcul.

Question : combien d'entre vous ont, dans le passé, développé un "DSPFFD" amélioré, pour consulter la structure des tables de vos applications ?

Pour développer ce type d'outil, dans les années 90, nous avons pour la plupart utilisé des tables temporaires générées par les commandes DSPFD et surtout DSPFFD.

Voici un "DSPFFD" amélioré produit au moyen d'un peu de code PHP exploitant le résultat d'une requête sur la table système QSYS2.SYSCOLUMNS.

Description de la vue : GJARRIGE3/SYSTABLES

Datastructure

Requêtage

Conversions

Objets utilisés

Objets utilisateurs

Source SQL

Verrouillages

Liste des colonnes renvoyées par la vue : GJARRIGE3/SYSTABLES

Nombre de colonnes : 30

N°	Nom de colonne (long)	Nom court	Libellé	Type	Longueur	Précision	CCSID	Null	Identité
1	TABLE_NAME	NAME	Long file name	VARCHAR	128		297	N	NO
2	TABLE_OWNER	CREATOR	TABLE_OWNER	VARCHAR	128		297	N	NO
3	TABLE_TYPE	TYPE	TABLE_TYPE	CHAR	1		297	N	NO
4	COLUMN_COUNT	COLCOUNT	COLUMN_COUNT	INTEGER	9	0		N	NO
5	ROW_LENGTH	RECLength	ROW_LENGTH	INTEGER	9	0		N	NO
6	TABLE_TEXT	LABEL	File text	VARGRAPHIC	50		1200	N	NO
7	LONG_COMMENT	REMARKS	Long file description	VARGRAPHIC	2000		1200	Y	NO
8	TABLE_SCHEMA	DRNAME	Library name	VARCHAR	128		297	N	NO

La structure affichée ici est justement celle de la table système SYSTABLES dont je vous mets pour info la structure ci-dessous (récupérée via le même outil, mais sans la colonne « libellé », faute de place) :

N°	Nom de colonne (long)	Nom court	Type	Longueur	Précision
1	TABLE_NAME	NAME	VARCHAR	128	
2	TABLE_OWNER	CREATOR	VARCHAR	128	
3	TABLE_TYPE	TYPE	CHAR	1	
4	COLUMN_COUNT	COLCOUNT	INTEGER	9	0
5	ROW_LENGTH	RECLENGTH	INTEGER	9	0
6	TABLE_TEXT	LABEL	VARGRAPHIC	50	
7	LONG_COMMENT	REMARKS	VARGRAPHIC	2000	
8	TABLE_SCHEMA	DBNAME	VARCHAR	128	
9	LAST_ALTERED_TIMESTAMP	ALTEREDTS	TIMESTAMP	10	
10	SYSTEM_TABLE_NAME	SYS_TNAME	CHAR	10	
11	SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR	10	
12	FILE_TYPE	FILETYPE	CHAR	1	
13	BASE_TABLE_CATALOG	LOCATION	VARCHAR	18	
14	BASE_TABLE_SCHEMA	TBDBNAME	VARCHAR	128	
15	BASE_TABLE_NAME	TBNAME	VARCHAR	128	
16	BASE_TABLE_MEMBER	TBMEMBER	VARCHAR	10	
17	SYSTEM_TABLE	SYSTABLE	CHAR	1	
18	SELECT_OMIT	SELECTOMIT	CHAR	1	
19	IS_INSERTABLE_INTO	INSERTABLE	VARCHAR	3	
20	IASP_NUMBER	IASPNUMBER	SMALLINT	4	0
21	ENABLED	ENABLED	VARCHAR	3	
22	MAINTENANCE	MAINTAIN	VARCHAR	6	
23	REFRESH	REFRESH	VARCHAR	9	
24	REFRESH_TIME	REFRESHDTS	TIMESTAMP	10	
25	MQT_DEFINITION	MQTDEF	DBCLOB	2097152	
26	ISOLATION	ISOLATION	CHAR	2	
27	PARTITION_TABLE	PART_TABLE	VARCHAR	11	
28	TABLE_DEFINER	DEFINER	VARCHAR	128	
29	MQT_RESTORE_DEFERRED	MQTRSTDFR	CHAR	1	
30	ROUNDING_MODE	DECFLTRND	CHAR	1	

A partir du moment où vous décidez d'exploiter les trésors contenus dans les tables systèmes DB2, et à condition de maîtriser un langage de développement web (par exemple PHP), il n'y a pas de limites aux types d'outils que vous pouvez développer pour administrer plus facilement vos bases de données DB2.

Voici pour information, la structure de la vue QSYS2.SYSCOLUMNS. A noter que cette structure diffère elle aussi sensiblement selon les versions d'OS (39 colonnes en V7R1, un peu moins dans les versions antérieures) :

N°	Nom de colonne (long)	Nom court	Type	Longueur	Précision
1	COLUMN_NAME	NAME	VARCHAR	128	
2	TABLE_NAME	TBNAME	VARCHAR	128	
3	TABLE_OWNER	TBCREATOR	VARCHAR	128	
4	ORDINAL_POSITION	COLNO	INTEGER	9	0
5	DATA_TYPE	COLTYPE	VARCHAR	8	
6	LENGTH	LENGTH	INTEGER	9	0
7	NUMERIC_SCALE	SCALE	INTEGER	9	0
8	IS_NULLABLE	NULLS	CHAR	1	
9	IS_UPDATABLE	UPDATES	CHAR	1	
10	LONG_COMMENT	REMARKS	VARGRAPHIC	2000	
11	HAS_DEFAULT	DEFAULT	CHAR	1	
12	COLUMN_HEADING	LABEL	VARGRAPHIC	60	
13	STORAGE	STORAGE	INTEGER	9	0
14	NUMERIC_PRECISION	PRECISION	INTEGER	9	0
15	CCSID	CCSID	INTEGER	9	0
16	TABLE_SCHEMA	DBNAME	VARCHAR	128	
17	COLUMN_DEFAULT	DFTVALUE	VARGRAPHIC	2000	
18	CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER	9	0
19	CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER	9	0
20	NUMERIC_PRECISION_RADIX	RADIX	INTEGER	9	0
21	DATETIME_PRECISION	DATPRC	INTEGER	9	0
22	COLUMN_TEXT	LABELTEXT	VARGRAPHIC	50	
23	SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR	10	
24	SYSTEM_TABLE_NAME	SYS_TNAME	CHAR	10	
25	SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR	10	
26	USER_DEFINED_TYPE_SCHEMA	TYPESCHEMA	VARCHAR	128	
27	USER_DEFINED_TYPE_NAME	TYPENAME	VARCHAR	128	
28	IS_IDENTITY	IDENTITY	VARCHAR	3	
29	IDENTITY_GENERATION	GENERATED	VARCHAR	10	
30	IDENTITY_START	START	DECIMAL	31	0
31	IDENTITY_INCREMENT	INCREMENT	DECIMAL	31	0
32	IDENTITY_MINIMUM	MINVALUE	DECIMAL	31	0
33	IDENTITY_MAXIMUM	MAXVALUE	DECIMAL	31	0
34	IDENTITY_CYCLE	CYCLE	VARCHAR	3	
35	IDENTITY_CACHE	CACHE	INTEGER	9	0
36	IDENTITY_ORDER	ORDER	VARCHAR	3	
37	COLUMN_EXPRESSION	EXPRESSION	DBCLOB	2097152	

38	HIDDEN	HIDDEN	VARCHAR	1	
39	HAS_FLDPROC	FLDPROC	VARCHAR	1	

5.2 Analyse de dépendances via les tables systèmes

Une vue DB2 peut faire appel à une ou plusieurs tables et/ou vues. Les vues dépendantes, peuvent elle-même faire appel à d'autres vues, ce qui peut aboutir à des niveaux de dépendance élevés.

En cas de nécessité de modifier une vue, il est utile de connaître les dépendances par rapport à l'objet à modifier.

Pour ce faire, on peut par exemple s'appuyer sur la table SYSVIEWDEP et sur le principe de la récursivité tel qu'il est implémenté dans DB2, pour analyser les dépendances entre objets.

WITH

```
-- création d'une CTE définissant les paramètres de la requête et l'objet de
départ de l'analyse
```

```
TMP_PARAM (LIB_REF, OBJ_REF) AS (
    SELECT 'mabib' as LIB_REF,
           'mabib.mavueDB2' as OBJ_REF
    FROM SYSIBM.SYSDUMMY1
),
```

```
-- création d'une CTE consolidant les vues et les objets dépendants dans une
seule liste
```

```
TMP_LISTOBJ (PARENT_LIB, PARENT_OBJ, CHILD_LIB, CHILD_OBJ) AS (
    SELECT A.TABLE_SCHEMA AS PARENT_LIB, A.TABLE_NAME AS PARENT_OBJ,
           B.OBJECT_SCHEMA AS CHILD_LIB, B.OBJECT_NAME AS CHILD_OBJ
    FROM QSYS2.SYSVIEWS A
    LEFT OUTER JOIN QSYS2.SYSVIEWDEP B
        ON A.TABLE_SCHEMA = B.VIEW_SCHEMA AND A.TABLE_NAME = B.VIEW_NAME
    WHERE A.TABLE_SCHEMA = (SELECT LIB_REF FROM TMP_PARAM)
),
```

```
-- CTE simplifiant l'écriture du nom des parents et enfants
```

```
TMP_BASE (PARENT, CHILD) AS (
    SELECT trim(PARENT_LIB) CONCAT '.' CONCAT trim(PARENT_OBJ) AS PARENT,
           trim(CHILD_LIB) CONCAT '.' CONCAT trim(CHILD_OBJ) AS CHILD
    FROM TMP_LISTOBJ
),
```

```
-- Dernière CTE définissant l'arbre hiérarchique (technique récursive)
```

```
TREE ( PARENT, CHILD, LVL) AS (
    SELECT PARENT, CHILD, 1
```

```

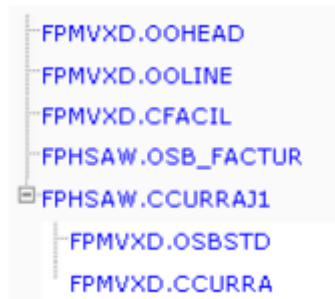
        FROM TMP_BASE
        WHERE PARENT = (SELECT OBJ_REF FROM TMP_PARAM)
    UNION ALL
    SELECT D.PARENT, D.CHILD, T.LVL + 1
        FROM TMP_BASE D, TREE T
        WHERE D.PARENT = T.CHILD
            AND D.PARENT != D.CHILD AND T.LVL < 20
    )
    SELECT PARENT, CHILD, LVL FROM TREE;

```

La requête ci-dessus permet d'obtenir le tableau suivant (en considérant que le point de départ est la vue OO_COMMANDES2 de la bibliothèque FPHSAW) :

PARENT	CHILD	LVL
FPHSAW.OO_COMMANDES2	FPMVXD.OOHEAD	1
FPHSAW.OO_COMMANDES2	FPMVXD.OOLINE	1
FPHSAW.OO_COMMANDES2	FPMVXD.CFACIL	1
FPHSAW.OO_COMMANDES2	FPHSAW.OSB_FACTUR	1
FPHSAW.OO_COMMANDES2	FPHSAW.CCURRAJ1	1
FPHSAW.OSB_FACTUR	FPMVXD.OSBSTD	2
FPHSAW.CCURRAJ1	FPMVXD.CCURRA	2

On peut utiliser ce résultat pour produire une liste HTML (balises et), et utiliser un module Javascript (comme par exemple le plugin jQuery Treeview) pour produire un affichage de type arborescent tel que celui ci-dessous :



On notera qu'il est possible de décliner cette technique sur une table de références produite par la commande système DSPPGMREF (mais attention, DSPPGMREF produit une table ne contenant que les noms courts des objets DB2).

On peut aussi compléter la technique ci-dessus en ajoutant à la requête la table système QSYS.SYSROUTINEDEP, de manière à disposer de références croisées plus exhaustives, incluant les procédures stockées.

6 Nouveautés V7R3 et V7R4

HISTORY_LOG_INFO table function

[HISTORY_LOG_INFO table function - IBM Documentation](#)

La fonction de table HISTORY_LOG_INFO renvoie une ligne pour chaque message dans le journal d'historique en fonction de la plage d'horodatage spécifiée. Il renvoie des informations similaires à celles retournées par la commande CL Display Log (DSPLOG) et l'API Open List of History Log Messages (QMHOLHST).

Exemples

Retourne une liste des messages du journal historique pour hier et aujourd'hui..

```
SELECT * FROM TABLE(QSYS2.HISTORY_LOG_INFO()) X
```

Retourne une liste de tous les messages de l'historique des 24 dernières heures.

```
SELECT * FROM TABLE(QSYS2.HISTORY_LOG_INFO(CURRENT_TIMESTAMP - 1 DAY)) X
```

Retourne les informations du journal d'historique depuis la dernière IPL, en supposant que le dernier horodatage IPL est dans une variable globale nommée LAST_IPL_TIME..

```
SELECT * FROM TABLE(QSYS2.HISTORY_LOG_INFO(LAST_IPL_TIME, CURRENT_TIMESTAMP)) A
```

Retourne les informations syslog formatées avec un en-tête RFC3164 pour tous les messages du journal d'historique du début d'aujourd'hui à l'avenir. Lorsque tous les messages du journal d'historique ont été retournés à l'appelant, la requête est mise en pause pendant 5 minutes (300 secondes) avant de vérifier à nouveau les messages.

```
SELECT syslog_facility, syslog_severity, syslog_event
FROM TABLE (QSYS2.HISTORY_LOG_INFO(START_TIME => CURRENT DATE,
                                     GENERATE_SYSLOG => 'RFC3164',
                                     EOF_DELAY => 300
) ) AS X;
```

ACTIVE_JOB_INFO table function

[ACTIVE_JOB_INFO table function - IBM Documentation](#)

La fonction de table ACTIVE_JOB_INFO renvoie une ligne pour chaque tâche active.

L'information retournée est similaire au détail vu de la commande Work with Active Jobs (WRKACTJOB) et de l'API List Job (QUSLJOB). La fonction table ACTIVE_JOB_INFO a deux utilisations :

Pour voir les détails de tous ou d'un sous-ensemble de tâches actives. Un sous-ensemble de tâches actives peut être demandé en utilisant les paramètres de filtre optionnels.

Mesurer les statistiques relatives aux emplois actifs. Vous pouvez utiliser un paramètre optionnel pour réinitialiser les statistiques, similaire à la commande WRKACTJOB F10 Restart Statistics. Les mesures seront calculées en fonction de ce nouveau point de départ.

Exemples

- **Exemple 1:** En ne regardant que les emplois QZDASOINIT, trouvez les 10 principaux consommateurs de Elapsed I/O.

```
SELECT JOB_NAME, AUTHORIZATION_NAME, ELAPSED_TOTAL_DISK_IO_COUNT,  
       ELAPSED_CPU_PERCENTAGE  
FROM TABLE(QSYS2.ACTIVE_JOB_INFO(  
    JOB_NAME_FILTER => 'QZDASOINIT',  
    SUBSYSTEM_LIST_FILTER => 'QUSRWRK')) X  
ORDER BY ELAPSED_TOTAL_DISK_IO_COUNT DESC  
FETCH FIRST 10 ROWS ONLY;
```

Remarque : Les données des colonnes ELAPSED_xxx sont mises à jour à chaque nouvelle exécution de la requête. Les données écoulées ne seront pas retournées la première fois qu'une requête est lancée pour ACTIVE_JOB_INFO pour une connexion. Voir le paramètre reset-statistics pour plus de détails.

- **Exemple 2:** Trouver les tâches actives en utilisant le stockage le plus temporaire. Incluez l'instruction SQL la plus récente exécutée pour chaque tâche cible.

```
SELECT JOB_NAME, AUTHORIZATION_NAME, TEMPORARY_STORAGE, SQL_STATEMENT_TEXT  
FROM TABLE (QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL')) X  
WHERE JOB_TYPE <> 'SYS'  
ORDER BY TEMPORARY_STORAGE DESC;
```

- **Exemple 3:** Décompose le champ JOB_NAME en colonnes distinctes pour chaque partie du nom de poste qualifié.

```
SELECT SUBSTR(JOB_NAME,1,6) AS JOB_NUMBER,  
       SUBSTR(JOB_NAME,8,POSSTR(SUBSTR(JOB_NAME,8),'/')-1) AS JOB_USER,  
       SUBSTR(SUBSTR(JOB_NAME,8),POSSTR(SUBSTR(JOB_NAME,8),'/')+1)
```

```
AS JOB_NAME  
FROM TABLE (QSYS2.ACTIVE_JOB_INFO()) AS X;
```

JOB_INFO table function

[JOB_INFO table function - IBM Documentation](#)

La fonction table JOB_INFO retourne une ligne pour chaque job répondant aux critères de sélection. Il renvoie des informations similaires à celles qui sont retournées par les commandes CL Work with User Jobs (WRKUSRJOB), Work with Subsystem Jobs (WRKSBSJOB) et Work with Submitted Jobs (WRKSBMJOB) et l'API List Job (QUSLJOB).

Exemples

- Trouve tous les travaux interactifs

```
SELECT * FROM TABLE(QSYS2.JOB_INFO(JOB_TYPE_FILTER => '*INTERACT')) X;
```

- Trouve les jobs soumis par SCOTTF qui n'ont pas encore démarré :

```
SELECT * FROM TABLE(QSYS2.JOB_INFO(JOB_USER_FILTER => 'SCOTTF',  
                                     JOB_STATUS_FILTER => '*JOBQ')) X;
```

JOBLOG_INFO table function

[JOBLOG_INFO table function - IBM Documentation](#)

La fonction de table JOBLOG_INFO renvoie une ligne pour chaque message dans un journal de tâches.

Exemples

- Retourner les informations du journal de tâches pour la tâche 347117/Quser/Qzdasoini.

```
SELECT * FROM TABLE(QSYS2.JOBLOG_INFO('347117/Quser/Qzdasoini')) A
```

- Extraire la dernière commande saisie par l'utilisateur.

```
SELECT MESSAGE_TEXT  
FROM TABLE(QSYS2.JOBLOG_INFO('817029/QUSER/QPADEV0004')) A  
WHERE A.MESSAGE_TYPE = 'REQUEST'  
ORDER BY ORDINAL_POSITION DESC  
FETCH FIRST 1 ROW ONLY
```

Utilisation des données JSON

[Working with JSON data - IBM Documentation](#)

En V7R2, Db2 for i peut consommer et générer des données JSON formatées.

[JSON concepts](#)

JSON (JavaScript Object Notation) est un format populaire pour l'échange d'informations. Il a une structure simple et est facile à lire par les humains et les machines. En raison de sa simplicité, il est utilisé comme une alternative à XML et ne nécessite pas de schémas prédéterminés. Bien qu'initialement créé pour une utilisation avec JavaScript, il est indépendant du langage et portable. Db2 for i est conforme au support SQL standard pour JSON.

[Using JSON TABLE](#)

La fonction JSON_TABLE table convertit un document JSON en une table relationnelle.

[Generating JSON data](#)

En utilisant les fonctions SQL, vous pouvez générer des données JSON formatées à partir de tables relationnelles.

La fonction `JSON_OBJECT` génère un objet JSON en utilisant les paires clé:valeur spécifiées. Si aucune paire clé:valeur n'est fournie, un objet vide est retourné.

<https://www.ibm.com/docs/en/i/7.2?topic=functions-json-object>

Exemples :

Générer un objet JSON pour un nom.

```
VALUES (JSON_OBJECT(KEY 'first' VALUE 'John', KEY 'last' VALUE 'Doe'));
```

```
VALUES (JSON_OBJECT('first' : 'John', 'last' : 'Doe'));
```

Le résultat produit par les deux lignes d'instructions ci-dessus, est la chaîne JSON suivante :

```
{"first":"John","last":"Doe"}
```

Génère un objet JSON contenant le nom de famille, la date d'embauche et le salaire de l'employé ayant un numéro d'employé « 000020 » :

```
SELECT JSON_OBJECT(
            'Last name' : LASTNAME,
            'Hire date' : HIREDATE,
            'Salary'    : SALARY)
FROM EMPLOYEE
WHERE EMPNO = '000020';
```

Le résultat de cette instruction est la chaîne JSON suivante :

```
{"Last name":"THOMPSON","Hire date":"1973-10-10","Salary":41250.00}
```

Pour autre un exemple d'implémentation, se reporter au chapitre relatif à la fonction `HTTPPOSTCLOB`.

IFS Services

[IFS Services - IBM Documentation](#)

Ces services fournissent des informations sur le système de fichiers intégré.

IFS_JOB_INFO table function

La fonction table IFS_JOB_INFO retourne une table contenant des informations sur les références système de fichiers intégrées pour une tâche.

IFS_OBJECT_LOCK_INFO table function

La fonction table IFS_OBJECT_LOCK_INFO retourne une table de résultats qui contient une ligne pour chaque tâche dont on sait qu'elle contient une référence ou un verrou sur l'objet.

IFS_OBJECT_PRIVILEGES table function

La fonction table IFS_OBJECT_PRIVILEGES renvoie une ligne pour chaque utilisateur autorisé à l'objet identifié par le nom de chemin, ainsi que les autorités objet et données associées.

IFS_OBJECT_REFERENCES_INFO table function

La fonction table IFS_OBJECT_REFERENCES_INFO retourne une table de résultats à une seule ligne contenant des informations sur les références système de fichiers intégrées sur un objet.

IFS_OBJECT_STATISTICS table function

La fonction table IFS_OBJECT_STATISTICS retourne une table d'objets contenus dans le nom du chemin de départ ou accessibles à partir du nom du chemin de départ.

IFS_READ, IFS_READ_BINARY, and IFS_READ_UTF8 table functions

Les fonctions de la table IFS_READ, IFS_READ_BINARY et IFS_READ_UTF8 lisent un fichier de flux de système de fichiers intégré identifié par path-name. Les données du fichier sont retournées sous forme de données de caractères, binaires ou UTF-8. Il peut être retourné sous forme d'une chaîne de données, ou il peut être divisé en plusieurs lignes en utilisant une longueur ou une fin de ligne de caractères spécifiés.

IFS_WRITE, IFS_WRITE_BINARY, and IFS_WRITE_UTF8 procedures

Les procédures IFS_WRITE, IFS_WRITE_BINARY et IFS_WRITE_UTF8 écrivent les données dans un fichier de flux système de fichiers intégré. Les données peuvent être écrites en caractères, en binaires ou en UTF-8. Les données peuvent être remplacées ou ajoutées à un fichier existant, ou un nouveau fichier peut être créé.

SERVER_SHARE_INFO view

La vue SERVER_SHARE_INFO renvoie des informations sur les partages IBM® i NetServer.

Exemple :

J'ai eu l'opportunité d'utiliser la fonction IFS_WRITE_UTF8 début 2022, dans le cadre d'un projet DevOps. Il s'agissait en l'occurrence de copier des fichiers sources dans l'IFS. Pour ce faire, j'ai écrit un script Node.js dont le rôle était de lire (via SQL) le membre d'un fichier source, et de recopier son contenu, ligne à ligne, dans un fichier de l'IFS.

```
/**
 * Crée l'enveloppe d'un fichier dans l'IFS (ou la remplace si déjà présente)
 * @param {*} path
 * @param {*} newfile
 * @returns
 */
function IFScreateOrReplaceFile(path, newfile) {
  return `CALL QSYS2.IFS_WRITE_UTF8('${path}/${newfile}', '',
    OVERWRITE => 'REPLACE')`;
}

/**
 * Ajoute des lignes dans un fichier de l'IFS
 * @param {*} path
 * @param {*} newfile
 * @returns
 */
function IFSappendFile(path, newfile) {
  return `CALL QSYS2.IFS_WRITE_UTF8('${path}/${newfile}', ?,
    OVERWRITE => 'APPEND')`;
}
```

On notera qu'il n'existe pas de fonction permettant de supprimer un fichier de l'IFS, mais on peut facilement pallier le manque via l'appel de commandes QSH, exécutées sur SQL DB2 via la procédure QCMDEXC. Voici un exemple avec deux fonctions Javascript, la première est un wrapper préparant l'appel de la procédure QCMDEXC, la seconde génère la commande de suppression des fichiers de l'IFS :

```
/**
 * Generate the Wrapper to execute Sys commands via SQL DB2
 * @param {*} cmd
 * @returns String
 */
function genCmdSys (cmd) {
  return `CALL QCMDEXC ('${cmd}')` ;
}

/**
 * Drop all files contained by the directory specified
 * @param {*} dir
 * @returns String
 */
function dropFileFromIFS (dir) {
  const cmd = `QSH CMD('rm -f ${dir}');` ;
  return genCmdSys(cmd);
}
```

Il n'existe pas non plus de fonctions DB2 pour créer un répertoire dans l'IFS, ou le vider. Les fonctions Javascript suivantes permettent de pallier ce manque :

```
/**
 * Crée un répertoire dans l'IFS
 * (ne produit pas d'erreur si le répertoire existe déjà)
 * @param {*} dir
 * @returns
 */
function createDirIntoIFS(dir) {
  const cmd = `QSH CMD('mkdir ${dir}');` ;
  return genCmdSys(cmd);
}
```

```
/**
 * Vide le contenu d'un répertoire de l'IFS
 * (ne produit pas d'erreur si le répertoire n'existe pas)
 * @param {*} dir
 * @returns
 */
function emptyDirIntoIFS(dir) {
  const cmd = `QSH CMD('rm -f ${dir}/*')`;
  return genCmdSys(cmd);
}
```

A lire, sur le même sujet :

<https://blog.faq400.com/en/db2-for-i/exploring-the-ifs-with-db2-services/>

OBJECT_STATISTICS table function

[OBJECT_STATISTICS table function - IBM Documentation](#)

La fonction table OBJECT_STATISTICS renvoie des informations sur les objets d'une bibliothèque.

Autorisations:

- Pour un objet qui n'est pas un profil utilisateur :
 - Si l'appelant a *EXÉCUTER l'autorisation de la bibliothèque,
 - Si l'appelant a *OBJOPR et *READ autorité à un objet, tous les détails sont retournés.
 - Sinon, des informations partielles sont retournées avec un avertissement SQL de '01548'.

Sinon, les informations de l'objet ne sont pas retournées.

- Pour un objet de profil utilisateur :
 - L'appelant doit avoir au moins l'un des éléments suivants :
 - Une certaine autorité au profil de l'utilisateur, ou
 - Autorisation de l'identificateur d'utilisation de la fonction QIBM_DB_SECADM.

Sinon, les informations de l'objet de profil utilisateur ne sont pas retournées.

Exemple

- Trouver tous les journaux dans la bibliothèque MJATST :

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS('MJATST ', 'JRN') ) AS X  
OU
```

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS('MJATST ', '*JRN') ) AS X
```

- Trouver tous les journaux et récepteurs de journaux dans la bibliothèque MJATST:

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS('MJATST ', 'JRN JRNRCV') ) AS X  
OU
```

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS('MJATST ', '*JRN *JRNRCV') ) AS X
```

- Trouver tous les programmes et programmes de service présents dans la bibliothèque MYLIB. Utiliser *ALLSIMPLE pour récupérer la liste plus rapidement (moins détaillée) :

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS('MYLIB','PGM SRVPGM','*ALLSIMPLE'))  
X
```

- Trouvez les commandes CL dont les paramètres par défaut ont été modifiés.

```
SELECT * FROM TABLE(QSYS2.OBJECT_STATISTICS('QSYS', '*CMD'))  
WHERE APAR_ID = 'CHGDFT';
```

Autres exemples d'utilisation :

[Using SQL for object's statistics @ RPGPGM.COM](#)

Fonction Javascript utile :

```
function getAllObjects() {  
    return `SELECT objname, objtype, objattribute, source_library, source_file,  
        source_member, source_timestamp, created_system  
        FROM TABLE(qsys2.object_statistics(?,'*ALL'))`;   
}
```

SYSPARTITIONSTAT

[SYSPARTITIONSTAT - IBM Documentation](#)

La vue SYSPARTITIONSTAT contient une ligne pour chaque partition de table ou membre de table. Si la table est une table distribuée, les partitions qui résident sur d'autres nœuds de base de données ne sont pas contenues dans cette vue catalogue. Ils sont contenus dans les vues catalogue des autres nœuds de base de données.

Fonctions utiles :

```
/**
 * Retrieve SQL query to get All members from one physical file
 * @returns String
 */
function getAllMembersFromFile() {
    return `SELECT trim(SYSTEM_TABLE_MEMBER) as SYSTEM_TABLE_MEMBER,
trim(SOURCE_TYPE) as SOURCE_TYPE
    FROM QSYS2.SYSPARTITIONSTAT
    WHERE SYSTEM_TABLE_SCHEMA = ? AND SYSTEM_TABLE_NAME = ?`;
}

/**
 * Retrieve SQL query to get All members from one library
 * @returns String
 */
function getAllMembersFromLib() {
    return `SELECT trim(SYSTEM_TABLE_MEMBER) as SYSTEM_TABLE_MEMBER,
trim(SOURCE_TYPE) as SOURCE_TYPE
    FROM QSYS2.SYSPARTITIONSTAT
    WHERE SYSTEM_TABLE_SCHEMA = ?`;
}
```


RECORD_LOCK_INFO view

[RECORD_LOCK_INFO view - IBM Documentation](#)

La vue RECORD_LOCK_INFO renvoie une ligne pour chaque verrouillage d'enregistrement de la partition.

Les valeurs retournées pour les colonnes de la vue sont étroitement liées aux valeurs retournées par l'API [Retrieve Record Locks API](#). Reportez-vous aux API pour plus d'informations.

Autorisation: L'appelant doit avoir :

- l'autorité *EXECUTE pour la bibliothèque contenant le fichier de base de données, et
- les autorisations *OBJOPR et *READ pour le fichier de base de données

Exemple

Trouver la liste des travaux verrouillant des tables en mise à jour

```
SELECT JOB_NAME  
FROM QSYS2.RECORD_LOCK_INFO  
WHERE TABLE_SCHEMA = 'DBFIC'  
AND LOCK_STATE = 'UPDATE'
```

Voir aussi :

[Finding record locks using SQL @ RPGPGM.COM](#)

SPOOLED_FILE_DATA table function

La fonction table SPOOLED_FILE_DATA retourne le contenu d'un fichier spooled.

Si le fichier spooled contient des données à double octet, le CCSID de la tâche doit être un CCSID mixte.

Autorisation : Cette fonction de tableau utilise la commande CPYSPLF CL. Toute exigence d'autorité pour la commande CL s'applique à l'utilisation de cette fonction.

Exemple

Retourner le fichier QSYSPRT le plus récent pour une tâche spécifique :

```
SELECT * FROM TABLE(SYSTOOLS.SPOOLED_FILE_DATA(
    JOB_NAME =>'193846/SLROMANO/QPADEV0009',
    SPOOLED_FILE_NAME =>'QSYSPRT'))
ORDER BY ORDINAL_POSITION;
```

Autre exemple :

```
SELECT ORDINAL_POSITION, SPOOLED_DATA
FROM TABLE (SYSTOOLS.SPOOLED_FILE_DATA
    ('193846/SLROMANO/QPADEV0009', 'QSYSPRT')
)
```

Exemple extrait d'une procédure stockée DB2 que j'ai développée pour automatiser la compilation de procédures stockées (dans le cadre d'un projet DevOps) :

```
-- étape 1 : création d'une table temporaire pour stocker le spoule produit
--           par la compilation
DECLARE GLOBAL TEMPORARY TABLE TMPSPool (
    ORDINAL_POSITION INTEGER DEFAULT NULL,
    SPOOLED_DATA VARCHAR(200) ALLOCATE(0) CCSID 297 DEFAULT NULL
) WITH REPLACE ;

INSERT INTO SESSION.TMPSPool (ORDINAL_POSITION, SPOOLED_DATA)
SELECT ORDINAL_POSITION, SPOOLED_DATA
FROM TABLE (SYSTOOLS.SPOOLED_FILE_DATA (V_JOBID , V_SHORT))
;
```

```
-- étape 2 : extraction et stockage du code gravité de la
-- compilation
SELECT code_fin INTO V_GRAVITE FROM (
  SELECT substr(spooled_data, 20, 2) as code_fin
  FROM (
    SELECT ordinal_position, spooled_data
    FROM (SELECT * FROM session.tmpspool)
    WHERE ordinal_position = (SELECT max(ordinal_position) - 1
                              FROM session.tmpspool)
  )
);
```

```
-- étape 3 : extraction de la récap de compil au format JSON
DECLARE GLOBAL TEMPORARY TABLE tmpclob (
  clobcol clob default null,
  pivotcol CHAR(1) default 'Y'
) WITH REPLACE ;
INSERT INTO session.tmpclob (clobcol)
WITH cte1 (spooled_data) as (
  SELECT trim(spooled_data) as spooled_data FROM SESSION.TMPSPPOOL
  WHERE ORDINAL_POSITION BETWEEN
    (select ordinal_position FROM SESSION.TMPSPPOOL
     where trim(spooled_data) =
       'ID-MSG GRAV ENREG TEXTE')
  AND
    (select max(ordinal_position) - 1 from qtemp.tmpspool)
)
, cte2 (jsondata) as (
  select json_object(key 'msg' value spooled_data) as jsondata
  from cte1
)
select json_arrayagg(jsondata) as jsondata from cte2 ;
```

Voir aussi :

[Using SQL to retrieve data from spooled files. @ RPGPGM.COM](#)

[Reading spool files with SQL – BlogFaq400](#)

IFS_OBJECT_STATISTICS table function

[IFS_OBJECT_STATISTICS table function - IBM Documentation](#)

La fonction table IFS_OBJECT_STATISTICS retourne la liste des objets contenus dans le nom du chemin de départ, ou accessibles à partir du nom du chemin de départ.

Cette information est similaire à ce qui est retourné par la commande de récupération des informations du répertoire (RTVDIRINF), ou l'API Qp0lGetAttr()--Get Attributes.

Aucune ligne n'est retournée pour les objets système de fichiers distants (remote). Cela signifie que pour le système de fichiers QNTC, seule une ligne pour /QNTC est retournée. Pour les systèmes de fichiers Network File System (NFS) et QFileSvr.400, aucune ligne n'est retournée.

Autorisation : L'utilisateur a besoin d'une autorisation *ALLOBJ ou des autorisations suivantes :

- Pour chaque répertoire inclus dans le nom du chemin utilisé pour lancer la recherche, *X
- Pour chaque répertoire traité récursivement par le service, *RX et *OBJMGT
- Pour chaque objet retourné par le service, *OBJMGT

Exemple

- Liste des informations de base pour tous les objets dans le répertoire /usr.

```
SELECT PATH_NAME, OBJECT_TYPE, DATA_SIZE, OBJECT_OWNER
FROM TABLE (QSYS2.IFS_OBJECT_STATISTICS(
    START_PATH_NAME => '/usr',
    SUBTREE_DIRECTORIES => 'NO'));
```

- Liste les informations de base pour tous les objets dans /usr, en traitant également tous les sous-répertoires.

```
SELECT PATH_NAME, OBJECT_TYPE, DATA_SIZE, OBJECT_OWNER
FROM TABLE (QSYS2.IFS_OBJECT_STATISTICS(
    START_PATH_NAME => '/usr',
    SUBTREE_DIRECTORIES => 'YES'
));
```

HTTPPOSTCLOB et HTTPPOSTBLOB scalar functions

<https://www.ibm.com/docs/en/i/7.3?topic=overview-httppostblob-httppostclob-scalar-functions>

Apparues en V7R3, les fonctions REST HTTPPOSTBLOB et HTTPPOSTCLOB mettent à jour une ressource binaire ou texte, sous l'URL spécifiée, via une requête HTTP de type POST.

Le schéma est SYSTOOLS.

Paramètres :

- url : spécifie l'URL à laquelle adresser la requête. Cet argument est défini comme une valeur VARCHAR(2048) CCSID 1208.
- httpHeader: spécifie un document XML d'en-tête optionnel. Pour utiliser l'en-tête HTTP par défaut, spécifiez NULL ou la chaîne vide. Ce paramètre est un CLOB(10K) CCSID 1208 ou une valeur XML.
- requestmsg : spécifie les données à mettre à jour à l'URL spécifiée. Pour la fonction HTTPPOSTBLOB, cet argument est défini comme BLOB(2G). Pour la fonction HTTPPOSTCLOB, cet argument est défini comme CLOB(2G) CCSID 1208.

Pour la fonction HTTPPOSTBLOB, la réponse est retournée sous forme de BLOB(2G). Pour la fonction HTTPPOSTCLOB, la réponse est retournée sous forme de CLOB(2G) CCSID 1208.

Exemple : Construction d'une requête HTTPPOSTCLOB, pour consommer un webservice sous protocole HTTP, avec passage de paramètres au format JSON

- étape 1 : préparation des paramètres au format JSON

```
with jsonDatas as (  
  values json_object(  
    'Param1' value 'truc1',  
    'Param2' value 'truc2',  
  )  
)  
select * from jsonDatas;
```

Résultat obtenu : {"Param1": "truc1", "Param2": "truc2"}

- étape 2 : préparation de l'entête HTTP

```
with
headerDatas(tname, tvalue) as (Values
    ('Content-Type', 'application/json; charset=utf-8')
),
httpHeader as (
    SELECT
        XMLGROUP(RTRIM(T.tname) AS "name", RTRIM(T.tvalue) AS "value"
        OPTION ROW "header" ROOT "httpHeader" AS ATTRIBUTES)
    From headerDatas T
)
select * from httpHeader;
```

Résultat :

```
<httpHeader><header name="Content-Type" value="application/json;
charset=utf-8"/></httpHeader>
```

- étape 3 : finalisation de la requête

```
WITH
url as (
    SELECT 'http://monserveur.com:8080/websevice/ParamJSON?flow=SrvTest\_IBMi\_InOut&flowType=EAll&actionJSON=launch' as data
    FROM SYSIBM.SYSDUMMY1
),
headerDatas(tname, tvalue) as (Values
    ('Content-Type', 'application/json; charset=utf-8')
),
httpHeader as (
    SELECT
        XMLGROUP(RTRIM(T.tname) AS "name", RTRIM(T.tvalue) AS "value"
        OPTION ROW "header" ROOT "httpHeader" AS ATTRIBUTES)
    FROM headerDatas T
),
jsonData as (
    values json_object(
        'Param1' value 'truc1',
        'Param2' value 'truc2'
    )
),
finalize as (
    SELECT (SELECT * FROM url) as url,
           (SELECT * FROM httpHeader) as header,
           (SELECT * FROM jsonData) as jsontda
    FROM SYSIBM.SYSDUMMY1
)
SELECT SYSTOOLS.HTTPPOSTCLOB(URL, HEADER, JSONTDA) as response FROM finalize
;
```