Université IBM i

uii2024
#ibmi
#uii2024

19 et 20 novembre 2024

IBM Innovation Studio Paris

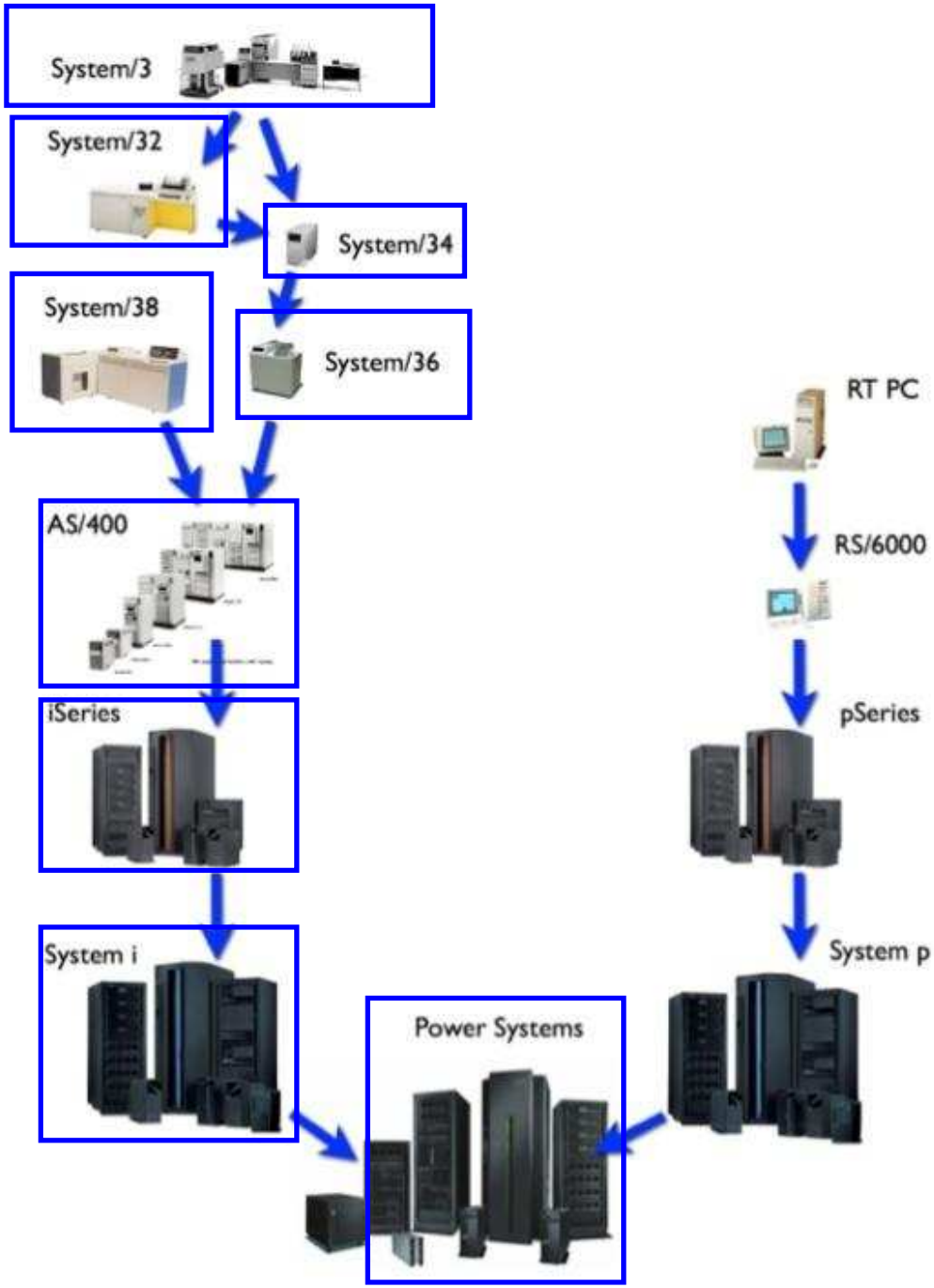
S36 – Tout savoir sur les suggestions d'index de Db2 for i

20 novembre 11:30 - 12:30

Christian GRIERE
CG Services ex IBM France
cgriere@gmail.com

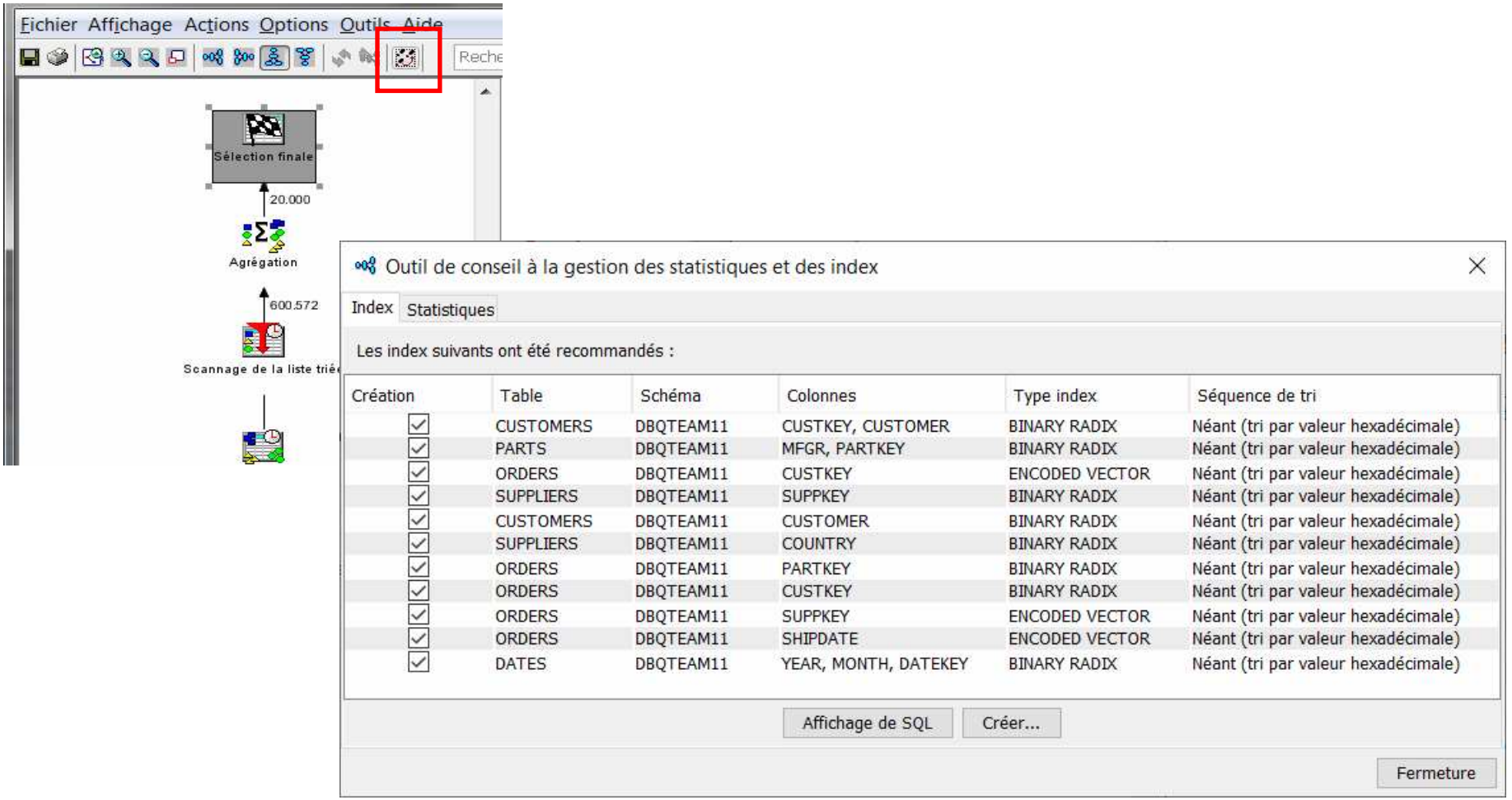
IBM

common FRANCE


System/3
System/32
System/34
System/38
System/36
AS/400
iSeries
System i
RT PC
RS/6000
pSeries
System p
Power Systems

De quoi vais-je vous parler ?

Des suggestions d'index de Db2 for i



Plan de la session

- Analogie avec la vie courante et Pub
- Qui émet les suggestions d'index ?
- Où trouve-t-on les suggestions d'index ?
- Quels types d'index sont concernés ?
- Y a-t-il des limitations sur les suggestions d'index ?
- Y a-t-il des relations entre les suggestions d'index ?
- Qu'est-ce qu'une suggestion condensée ?
- Peut-on exploiter les suggestions condensées (ou non) de façon globale ?
- Questions/Réponses

Analogie avec la vie courante - Pub

Un client demande à un architecte de lui construire une maison avec 1 chambre en RdC, 3 chambres au 1er étage, 1 cuisine et 1 salle de bain.

Un client demande le résultat d'une requête SQL à un optimiseur sur une table avec 3 index.

L'architecte présente un plan au client avec la salle de bain au 1er étage et suggère au client d'avoir une salle de bain supplémentaire pour la chambre du RdC ce qui serait plus pratique.

L'optimiseur présente un plan au client et suggère au client d'avoir un index supplémentaire pour la table ce qui serait plus pratique.

Analogie avec la vie courante et Pub

Le client ignore la suggestion de l'architecte.

Le client ignore la suggestion de l'optimiseur.

Le maitre d'oeuvre exécute le plan de l'architecte.

Le moteur SQL exécute le plan de l'optimiseur.

La maison est construite. Le client regrette de ne pas avoir écouté son architecte.

Le moteur SQL estime que sa suggestion est justifiée et crée l'index. Il construit le résultat très rapidement. Le client ne regrette pas d'avoir choisi Db2 for i.

Qui émet les suggestions ?

- L'optimiseur : au moment de la création du plan. Elles font partie du plan.
- Le moteur SQL : lorsqu'il utilise un plan avec des suggestions.

Où trouve-t-on les suggestions d'index ?

- Origine Optimiseur :
  - Visual Explain (interface graphique)
  - Moniteur de performance SQL
  - Cache de plan
  - Image instantanée de cache de plan

- Origine Moteur SQL : Lorsque le moteur SQL utilise un plan avec des suggestions il les écrit dans la table QSYS2.SYSIXADV

  - Elles sont émises par les exécutions en full et pseudo open
  - Pour des questions de performance ces écritures sont bufferisées.
  - Elles peuvent mettre jusqu'à 2 minutes pour être visibles.

Quels types d'index sont concernés ?

- Index texte (utilisé par OmniFind Text Search Server)

  Pas de suggestion

- Index binaire

  Suggestions pour une optimisation de base

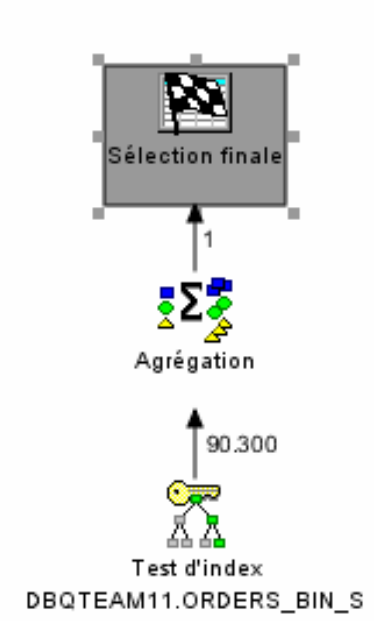
- Index à vecteurs encodés (EVI)

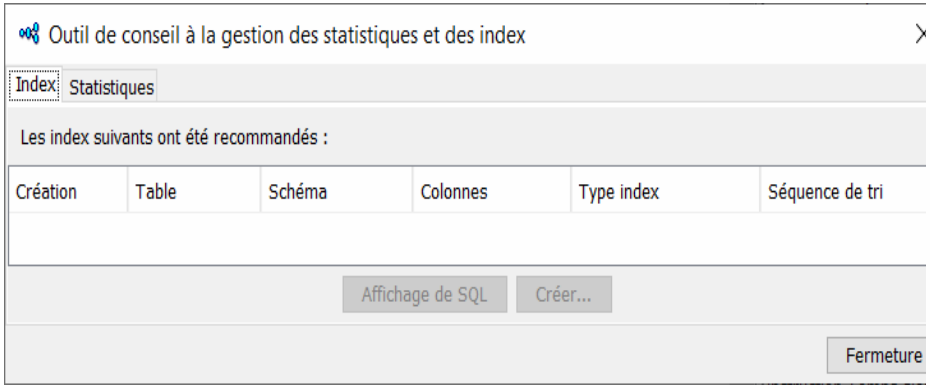
  Suggestions pour une optimisation de base

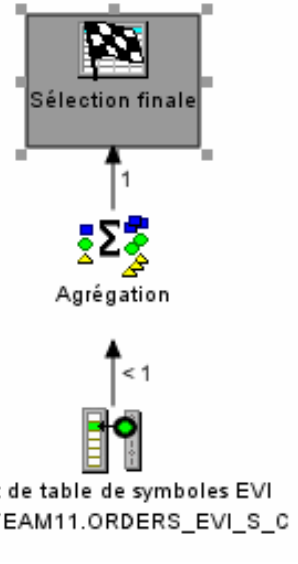
Suggestions EVI

SELECT count(*) FROM orders WHERE shipmode='RAIL' ;

Index binaire suggéré : shipmode





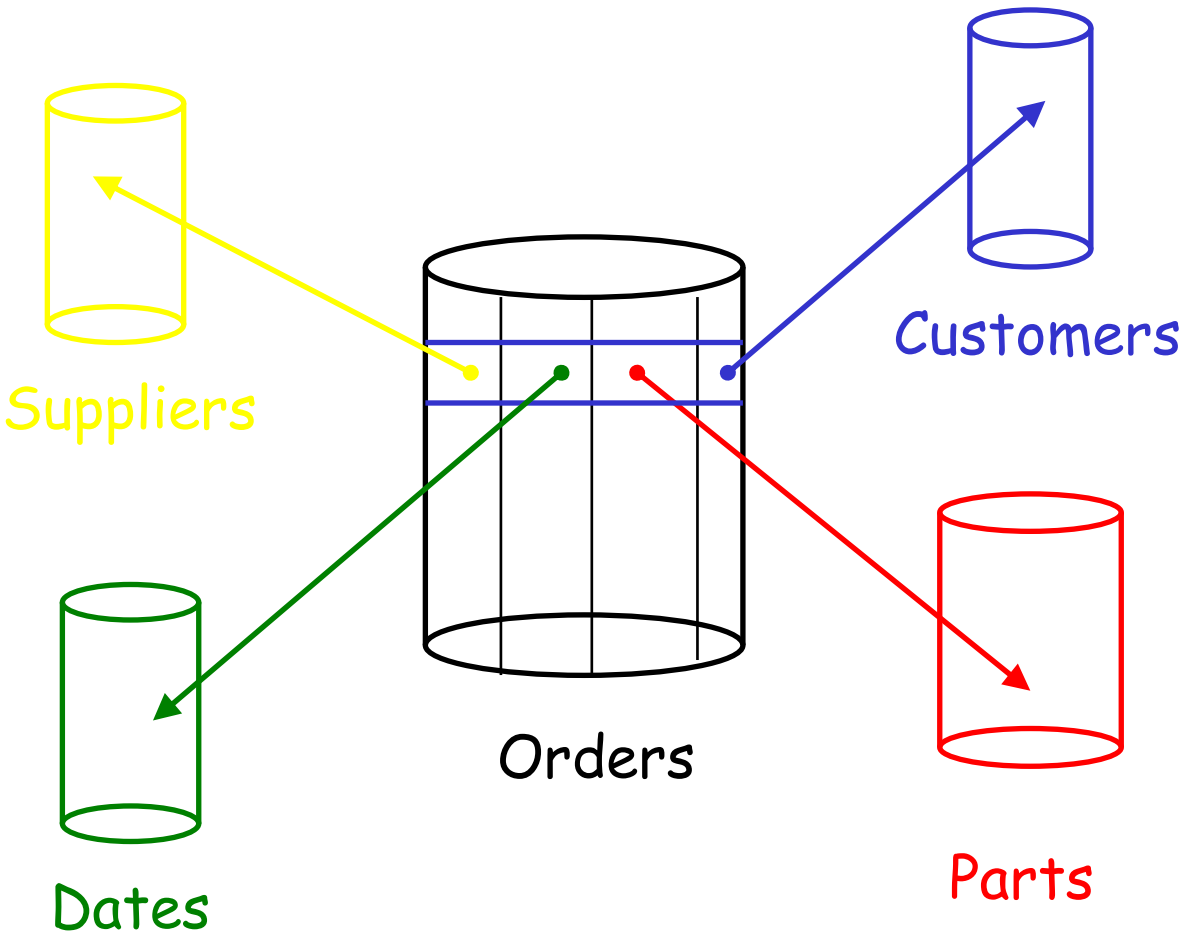


Durée estimée : 1 µs versus 48 ms

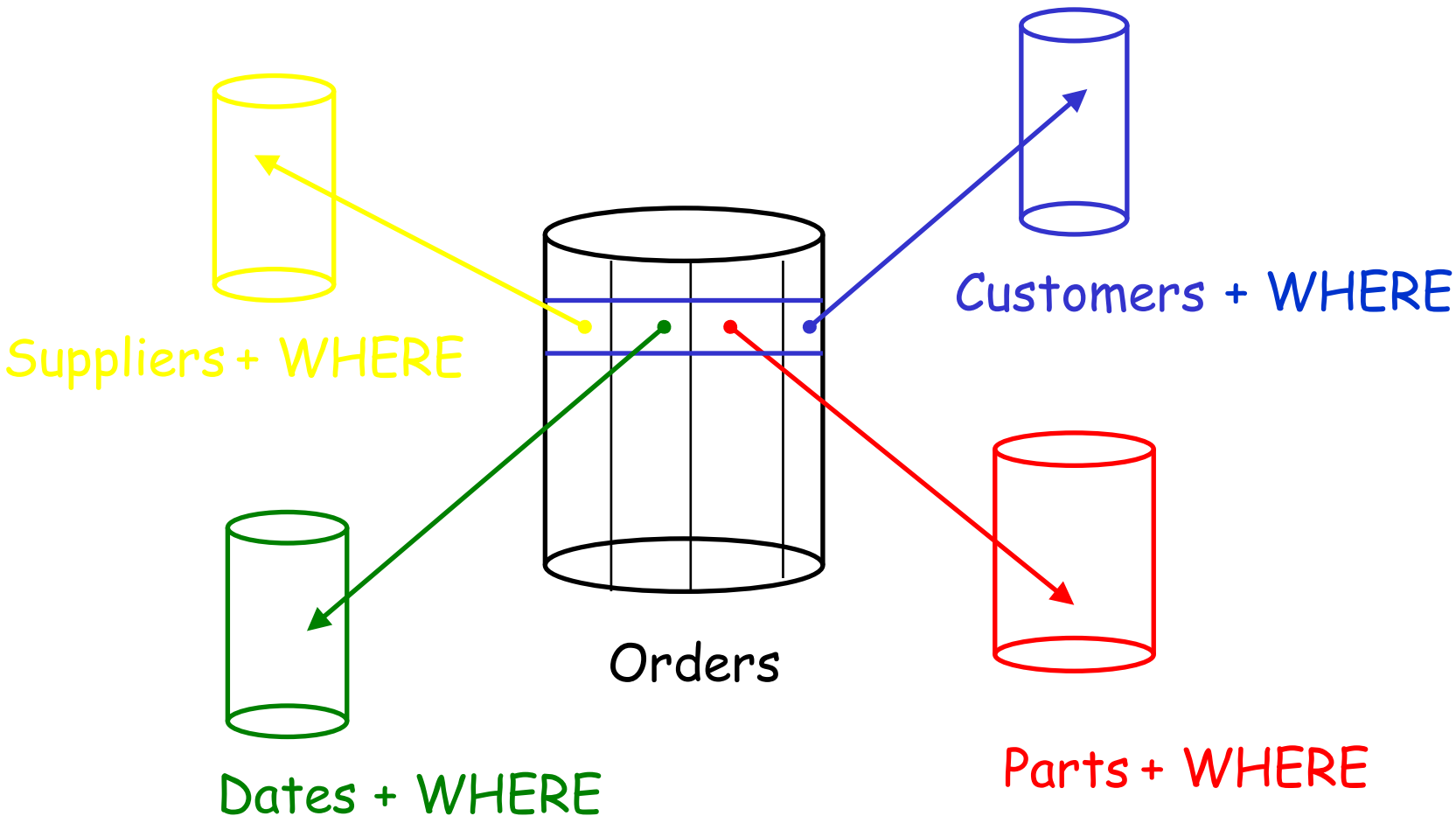
Suggestions EVI pour Lookahead Predicate Generation (LPG)

Une stratégie de réécriture (par l'optimiseur) de la requête pour déplacer des sélections faites sur des tables vers une autre table.

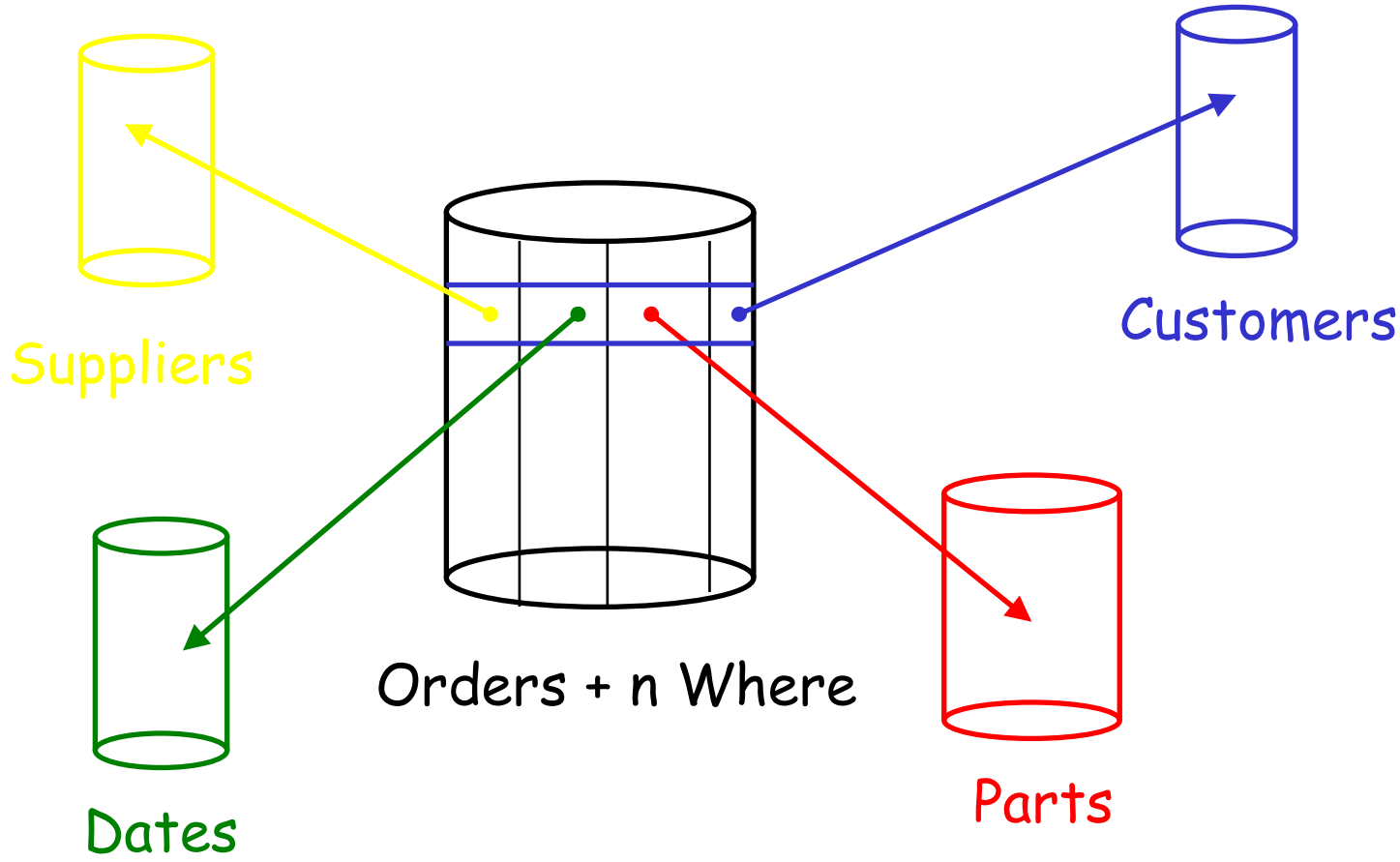
LPG - Jonctions en étoile



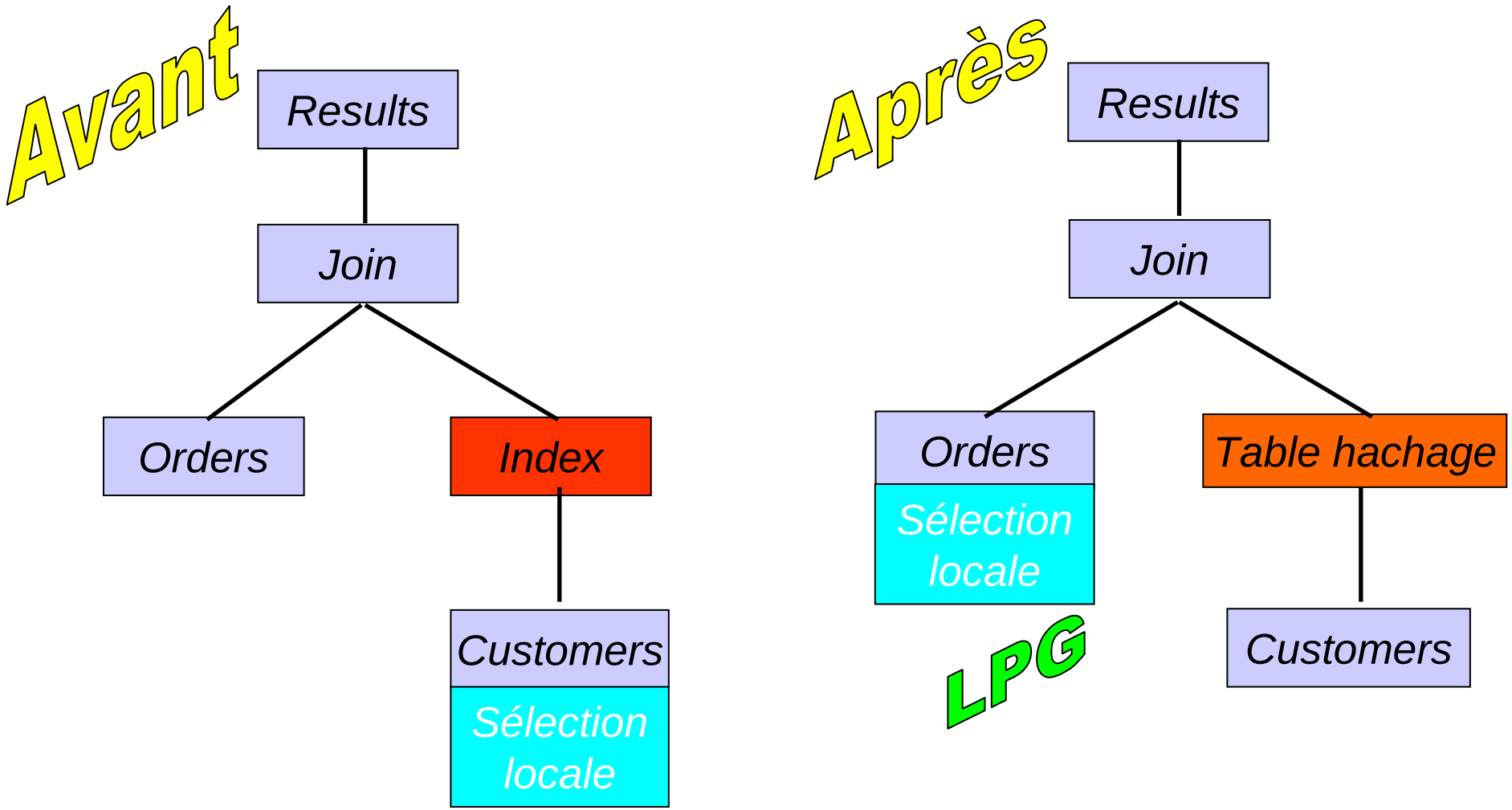
LPG - Jonctions en étoile + sélections



LPG - Jonctions en étoile + sélections LPG



LPG – Avant / Après



LPG – Exemple de requête en étoile avec sélections

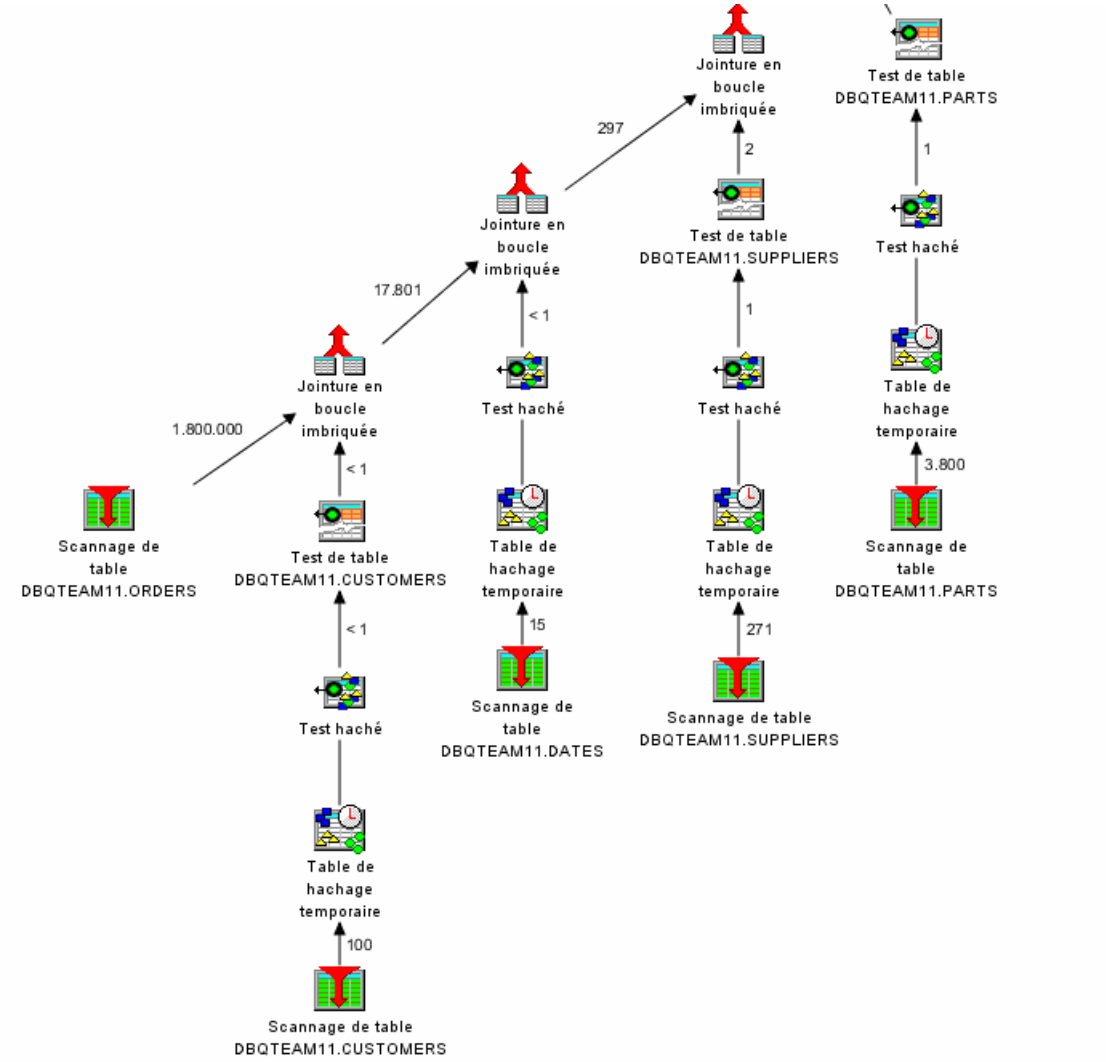
```
SELECT c.CUSTOMER, c.CUSTKEY, p.PART, o.PARTKEY, s.SUPPLIER,
       o.ORDERDATE, o.COMMITDATE, o.SHIPDATE, o.RECEIPTDATE,
       o.SHIPMODE
FROM   orders o, parts p, customers c, suppliers s, dates t

WHERE  t.YEAR = 2004  AND t.MONTH = 6
AND    p.MFGR = 'Manufacturer#1'
AND    c.CUSTOMER BETWEEN 'Customer#000000001'
AND    Customer#000000100'
AND    (s.COUNTRY = 'CANADA' OR s.COUNTRY = 'FRANCE'
       OR s.COUNTRY = 'UNITED KINGDOM')

AND    o.SHIPDATE = t.DATEKEY
AND    o.PARTKEY = p.PARTKEY
AND    o.CUSTKEY = c.CUSTKEY
AND    o.SUPPKEY = s.SUPPKEY

OPTIMIZE FOR ALL ROWS ;
```

LPG – Plan sans index

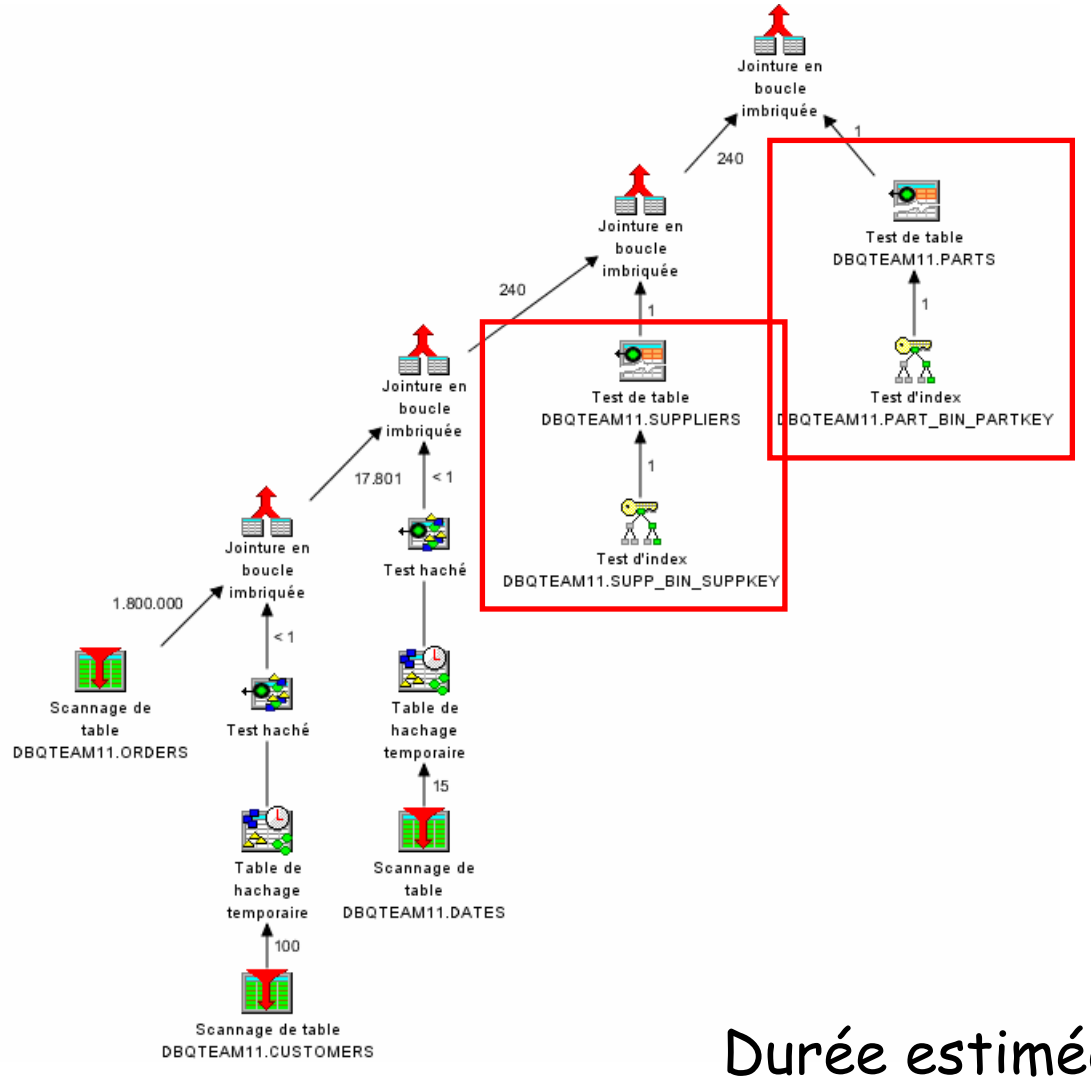


Durée estimée : 518 ms

LPG - Indexation binaire de base

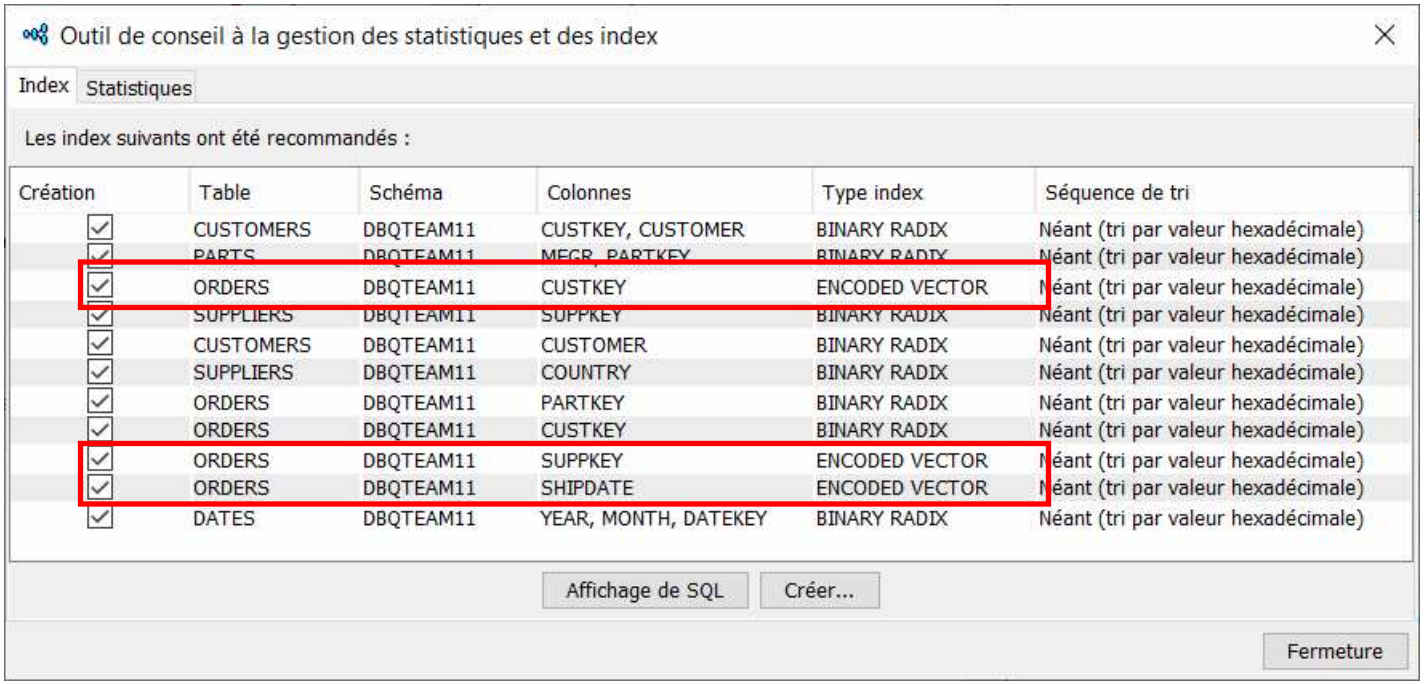
```
create  index CUST_BIN_CUSTKEY on customers (custkey) ;
create  index PART_BIN_PARTKEY on parts (partkey) ;
create  index SUPP_BIN_SUPPKEY on suppliers (suppkey) ;
create  index DATE_BIN_DATEKEY on dates (datekey) ;
```

LPG – Plan avec indexation binaire de base



Durée estimée : 400 ms

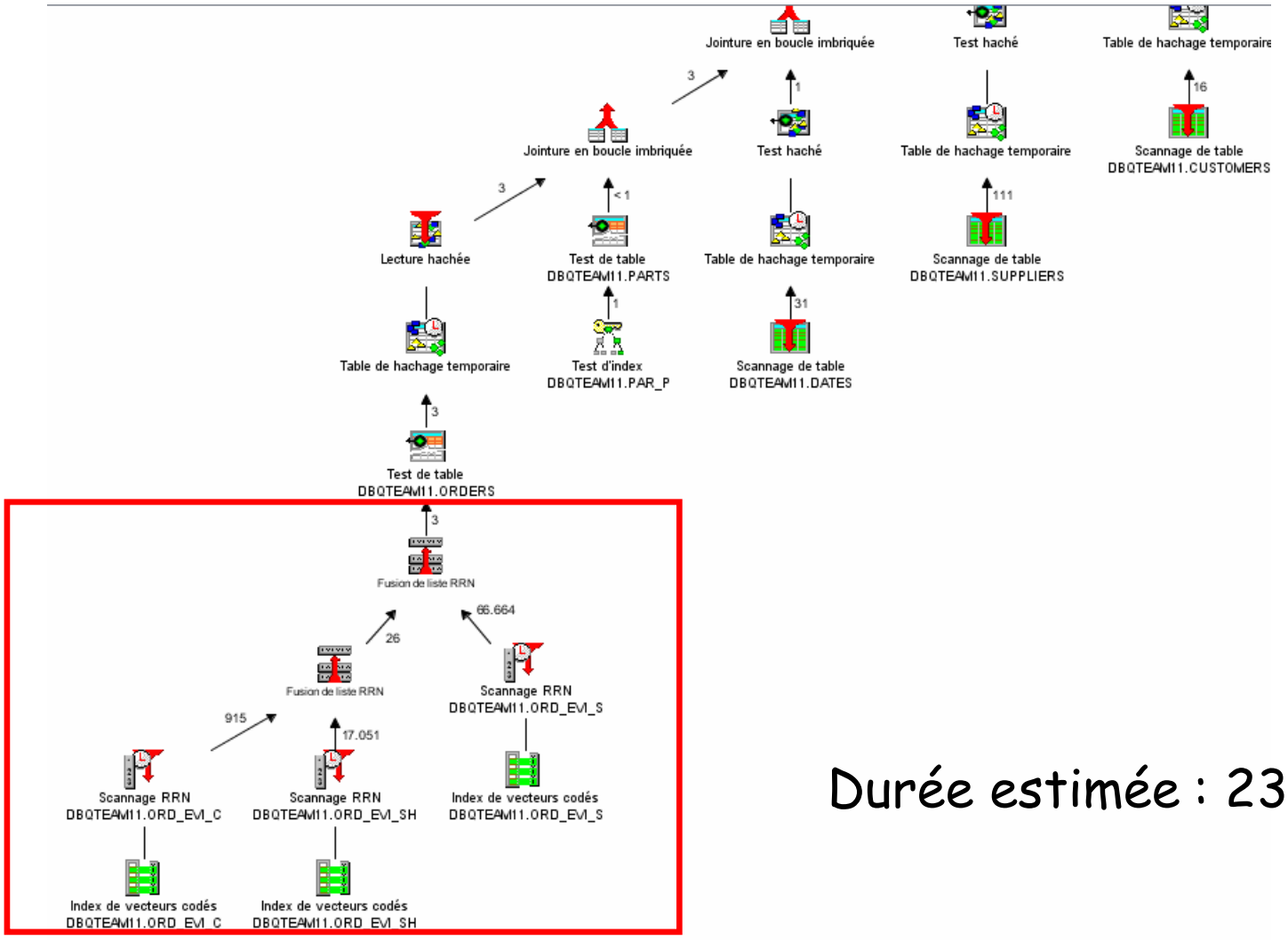
LPG – Suggestions d'index



LPG – Indexation EVI pour LPG

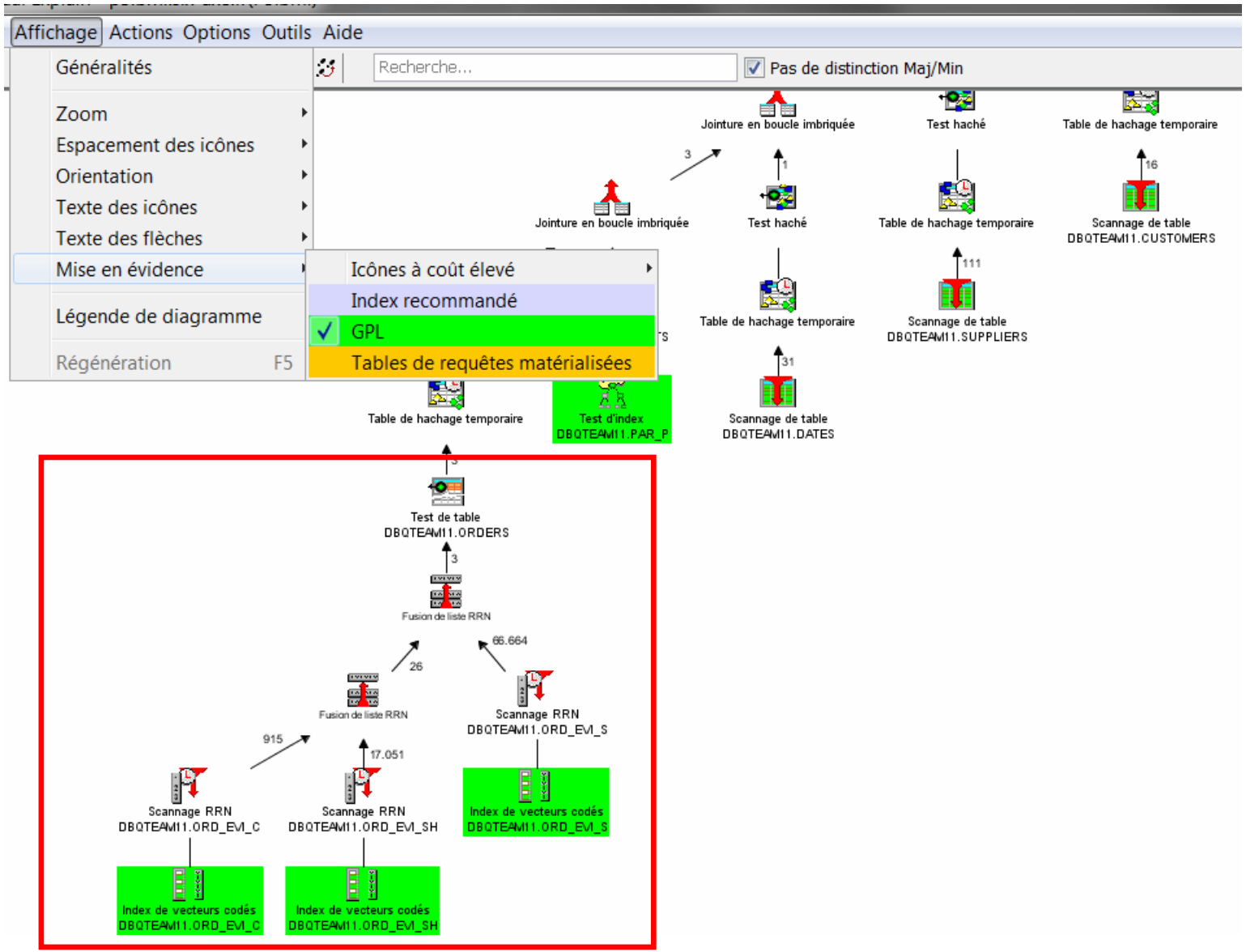
```
create encoded vector index ord_evi_c  on orders (custkey) ;
create encoded vector index ord_evi_s  on orders (suppkey) ;
create encoded vector index ord_evi_sh on orders (shipdate) ;
```

LPG – Plan avec LPG

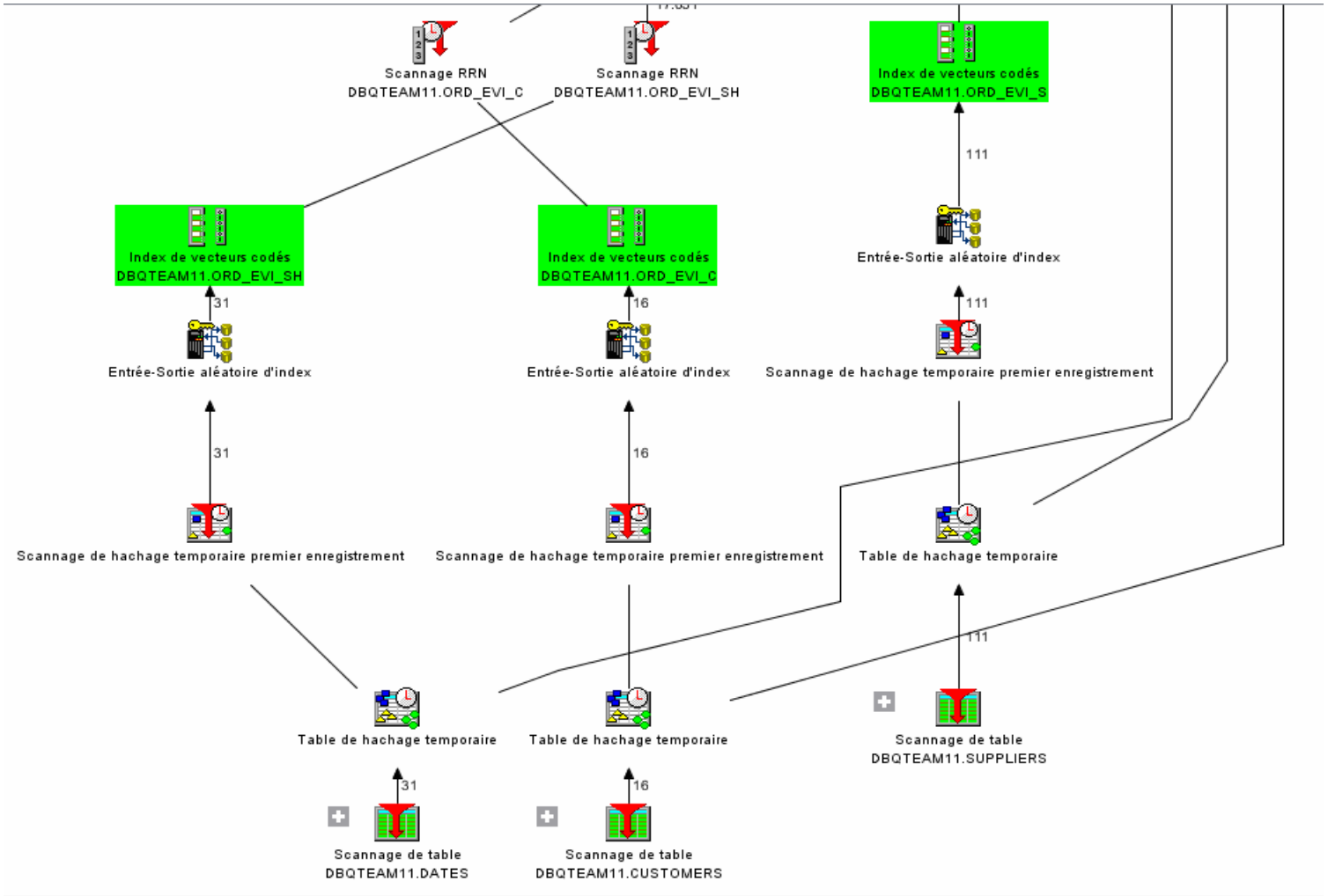


Durée estimée : 23 ms

LPG – Visualisation LPG



LPG - Visualisation LPG - Détail



Limitations sur les suggestions d'index binaire

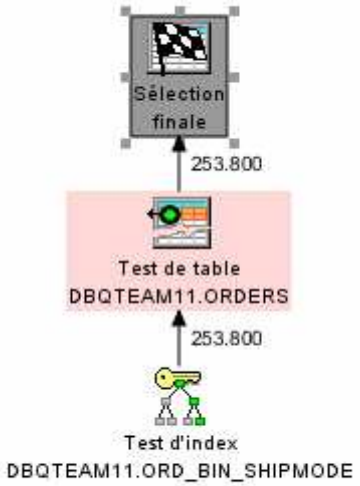
Elles sont émises pour une optimisation de base :

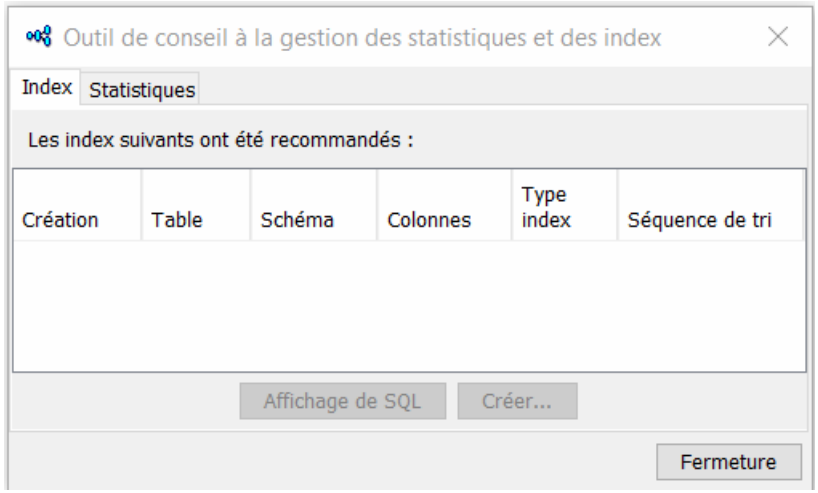
- Pas de suggestions d'index binaire Only Access
- Pas de suggestions d'index binaire FBI

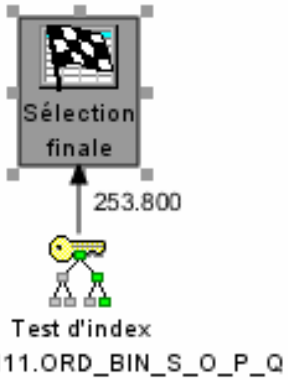
Limitation : Index binaire « Only Access »

SELECT orderkey, partkey, quantity FROM orders
WHERE shipmode ='AIR' ;

Index suggéré : shipmode



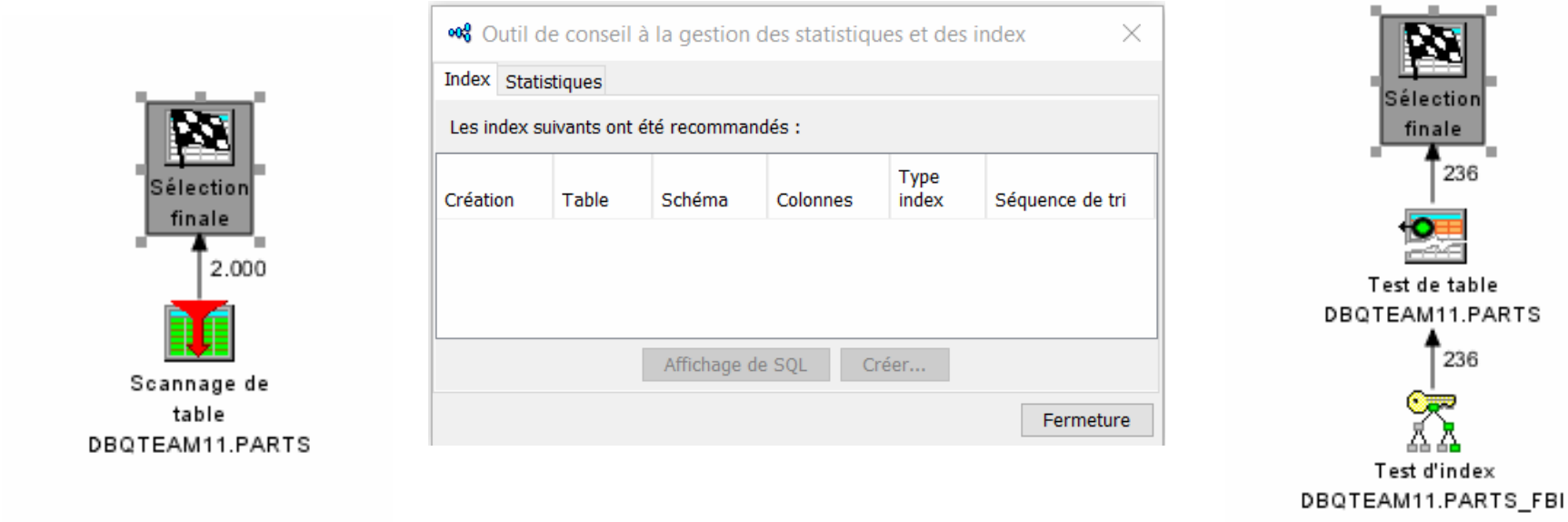




CREATE INDEX ord_bin_s_p_q ON
orders(shipmode, orderkey, partkey, quantity)
Durée estimée 0,014 ms versus 1,645 ms

Limitation : Index FBI

SELECT * FROM parts WHERE upper(part) LIKE 'STEEL%'

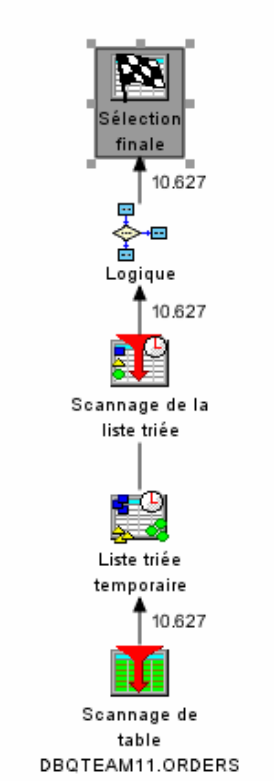


CREATE INDEX parts_fbi ON parts (upper(part))

Durée estimée : 0,142 ms versus 4,307 ms

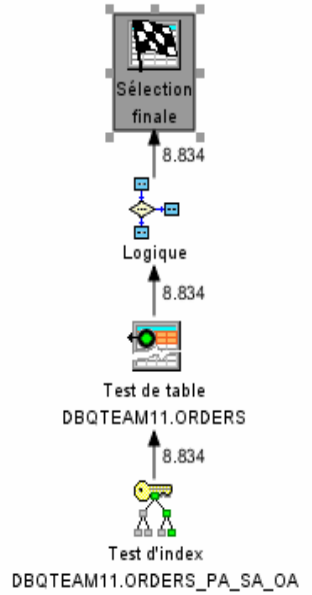
Limitation : Séquence

SELECT * FROM orders WHERE partkey BETWEEN ? AND ?
ORDER BY partkey ASC, suppkey ASC, orderkey ASC ;



Les index suivants ont été recommandés :

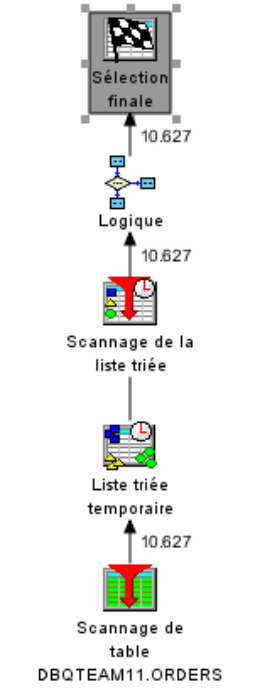
Création	Table	Schéma	Colonnes	Type index
☑	ORDERS	DBQTEAM11	PARTKEY, SUPPKEY, ORDERKEY	BINARY RADIX



Durée estimée 5,2 ms versus 1242 ms

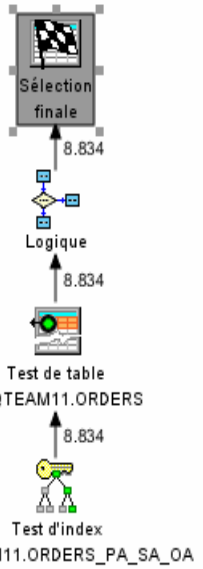
Limitation : Séquence

SELECT * FROM orders WHERE partkey between ? AND ?
ORDER BY partkey DESC, suppkey DESC, orderkey DESC ;



Les index suivants ont été recommandés :

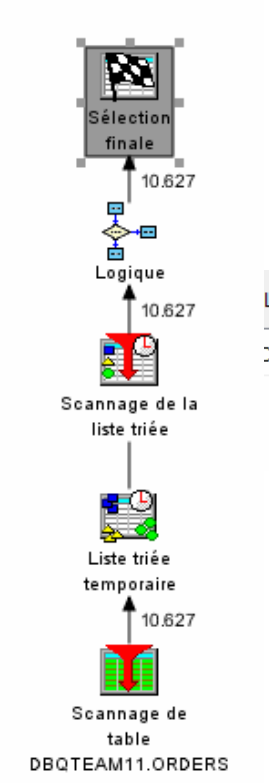
Création	Table	Schéma	Colonnes	Type index
☑	ORDERS	DBQTEAM11	PARTKEY, SUPPKEY, ORDERKEY	BINARY RADIX



Durée estimée : 5,2 ms versus 1242 ms

Limitation : Séquence

SELECT * FROM orders WHERE partkey BETWEEN ? AND ?
ORDER BY partkey ASC, suppkey DESC, orderkey ASC ;



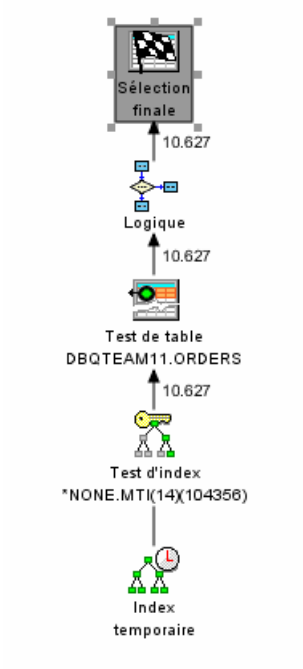
Les index suivants ont été recommandés :

Création	Table	Schéma	Colonnes	Type index
☑	ORDERS	DBQTEAM11	PARTKEY, SUPPKEY, ORDERKEY	BINARY RADIX

Durée estimée : 84 ms
versus 1242 ms

Limitation : Séquence

SELECT * FROM orders WHERE partkey BETWEEN ? AND ?
ORDER BY partkey ASC, suppkey DESC, orderkey ASC ;



Infos index	
Nombre d'entrées d'index	1.800.000
Cardinalité des clés	1.800.000
Taille d'index, en octets	5,749E7
Valeur d'amortissement	1
Liste des colonnes clés	ORDERS_1,PARTKEY. ORDERS_1,SUPPKEY DESC. ORDERS_1,ORDERKEY
Taille de clé (octets)	20
Partage possible	Oui
Index réutilisé	Oui

Durée estimée : 3,5 ms (hors temps de création du MTI)
versus 84 ms

Y a-t-il des relations entre les suggestions d'index ?

Certaines suggestions sont exclusives.

```
SELECT * FROM orders o INNER JOIN parts p
ON o.partkey=p.partkey
```

Comme c'est un INNER JOIN il y aura deux suggestions d'index binaires : l'une lorsque la jonction se fait de *orders* vers *parts*, l'autre lorsque la jonction se fait de *parts* vers *orders*.

Dans tous les cas l'optimiseur choisira un ordre et n'utilisera qu'un index pour faire la jonction.

Y a-t-il des relations entre les suggestions d'index ?

Certaines suggestions sont dépendantes.

SELECT * FROM orders WHERE suppkey=? OR orderkey=? ;

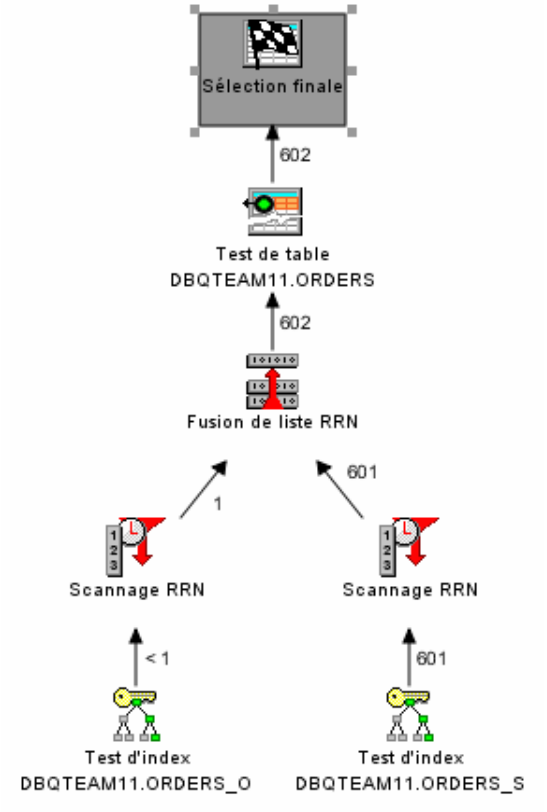
Les index suivants ont été recommandés :

Création	Table	Schéma	Colonnes	Type index	Séquence de tri
☑	ORDERS	DBQTEAM11	SUPPKEY	BINARY RADIX	Néant (tri par valeur hexadécimale)
☑	ORDERS	DBQTEAM11	ORDERKEY	BINARY RADIX	Néant (tri par valeur hexadécimale)

Avec uniquement l'un des 2 index le plan sera :

Y a-t-il des relations entre les suggestions d'index ?

Pour que le plan soit efficace il faudra que les deux suggestions soient satisfaites :



Y a-t-il des relations entre les suggestions d'index ?

Si l'instruction a été exécutée 4 fois le moteur générera ces lignes dans la table SYSIXADV :

Index recommandés pour DBQTEAM11.ORDERS

Schéma système	Table pour laquelle un index a été recommandé	Clés recommandées	Nombre de recommandations d'utilisation par les requêtes	Nombre de recommandations dépendant d'une autre recommandation
DBQTEAM11	ORDERS	SUPPKEY	4	4
DBQTEAM11	ORDERS	ORDERKEY	4	4

Indicateurs BD d'une partition de production

Indicateur	Nombre
1 - Fichiers physiques/tables	25 580
2 - Structure d'accès hors index	10 825
3 - Index binaires	10 160
4 - Encoded Vector Index	0
5 - Index non utilisés	105
6 - Index temporaires	490
7 - Suggestions élémentaires	20 212

```
SELECT COUNT(*) FROM qsys2.sysixadv
WHERE table_schema NOT LIKE 'Q%' ;
```

Qu'est-ce qu'une suggestion condensée ?

C'est une suggestion qui peut inclure les colonnes d'une autre suggestion.

Supposons que vous ayez les requêtes suivantes :

```
SELECT * FROM orders WHERE suppkey = ? AND partkey = ? AND
orderkey = ? ;
SELECT * FROM orders WHERE suppkey = ? AND partkey = ? ;
```

Si vous n'avez aucun index sur la table *orders* vous pourrez avoir (dans la table SYSIXADV) deux suggestions d'index :

- *suppkey, partkey, orderkey*
- *suppkey, partkey*

Elles peuvent être condensées en une seule : suppkey, *partkey, orderkey*.

Qu'est-ce qu'une suggestion condensée ?

Elle peut aussi regrouper plusieurs suggestions dont les colonnes majeures (celles à gauche) peuvent être interverties sans impacter la suggestion initiale.

Supposons que vous ayez les requêtes suivantes :

SELECT * FROM orders WHERE orderkey = ? AND partkey = ? and suppkey = ? ;

SELECT * FROM orders ORDER BY partkey, suppkey ;

Index recommandés pour DBQTEAM11.ORDERS

Table pour laquelle un index a été recommandé	Schéma système	Clés recommandées	Clés principales, ordre indépendant	Nombre de recommandations d'utilisation par les requêtes 1
ORDERS	DBQTEAM11	SUPPKEY, PARTKEY, ORDERKEY	SUPPKEY, PARTKEY, ORDERKEY	1
ORDERS	DBQTEAM11	PARTKEY, SUPPKEY		1

Suggestions élémentaires/condensées

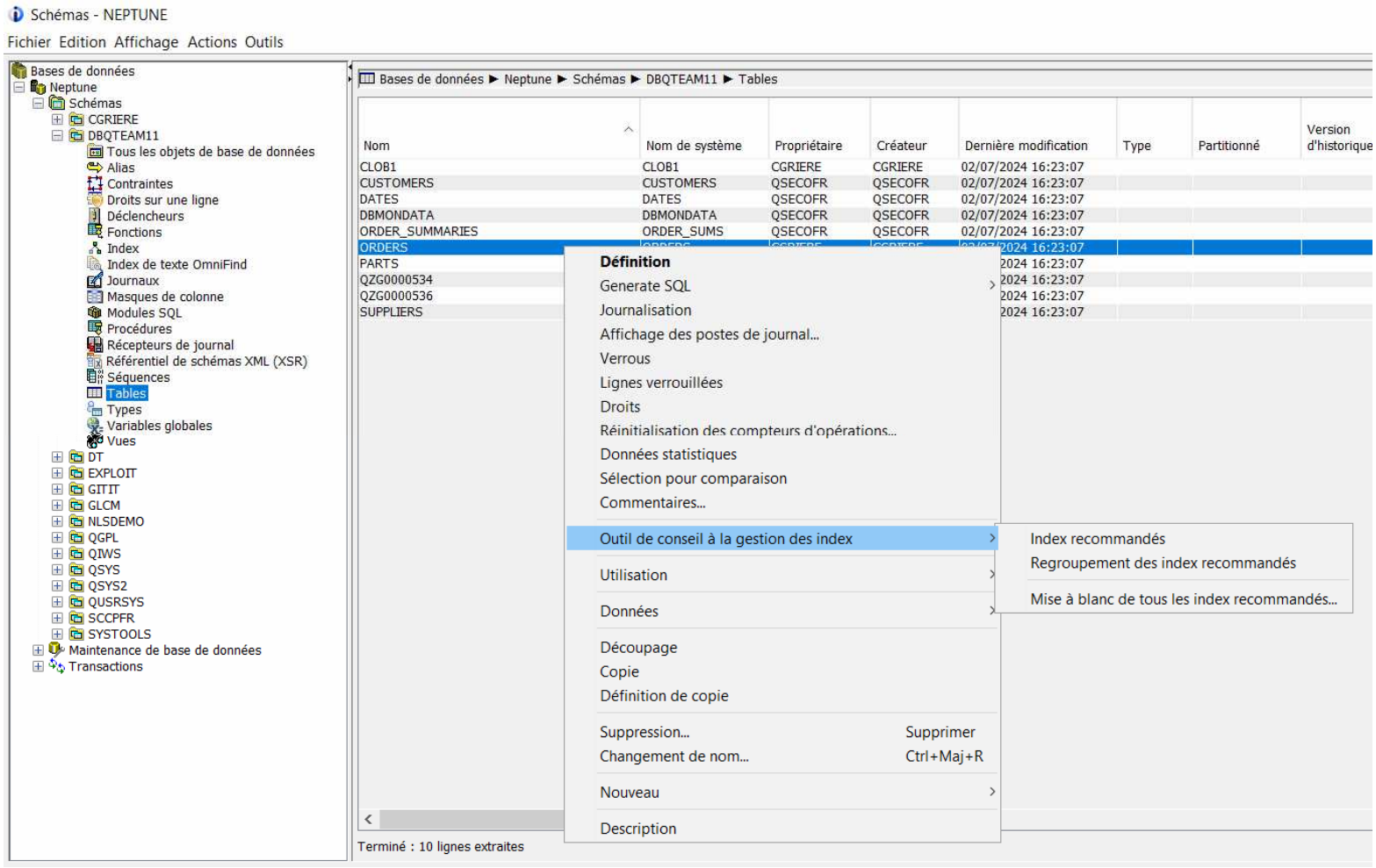
Index recommandés pour DBQTEAM11.ORDERS

Table pour laquelle un index a été recommandé	Schéma système	Clés recommandées	Clés principales, ordre indépendant	Nombre de recommandations d'utilisation par les requêtes ⌄ 1
ORDERS	DBQTEAM11	SUPPKEY, PARTKEY, ORDERKEY	SUPPKEY, PARTKEY, ORDERKEY	1
ORDERS	DBQTEAM11	PARTKEY, SUPPKEY		1

Index recommandés (condensés) pour DBQTEAM11.ORDERS

Table pour laquelle un index a été recommandé	Schéma	Clés recommandées	Nombre de recommandations d'utilisation par les requêtes ⌄ 1
ORDERS	DBQTEAM11	PARTKEY, SUPPKEY, ORDERKEY	2

Accès ACS aux suggestions élémentaires/condensées



Accès SQL aux suggestions condensées

On accède aux suggestions condensées par une vue QSYS2.CONDENSEDINDEXADVICE ou par ACS

- Elle s'appuie sur la table SYSIXADV.

- Elle donne la liste des suggestions condensées pour une table, un schéma ou une partition.

- Scott Forstie a habillé cette vue
  - pour générer le script des index à créer via la procédure : SYSTOOLS.HARVEST_INDEX_ADVICE
  - pour créer les index directement via la procédure : SYSTOOLS.ACT_ON_INDEX_ADVICE.

Optimisation SQL

Sous Db2 for i la meilleure démarche pour optimiser son environnement SQL est celle qui consiste à :

- optimiser les requêtes les plus consommatrices.
- optimiser les requêtes les plus utilisées.
- remplacer les MTI par des index permanents.
- éliminer les index inutilisés et redondants.
- s'assurer de la bonne réutilisation des curseurs.
- avoir les bons objectifs d'optimisation.
- avoir un bon environnement d'exécution (mémoire, CPU, IO).
- …

Peut-on utiliser les suggestions de façon globale ?

Ci-après les indicateurs BD d'une partition de production :

Indicateur	Nombre
1 - Fichiers physiques/tables	25 580
2 - Structure d'accès hors index	10 825
3 - Index binaires	10 160
4 - Encoded Vector Index	0
5 - Index non utilisés	105
6 - Index temporaires	490
7 - Suggestions élémentaires	20 212
8 - Suggestions condensées	12 826

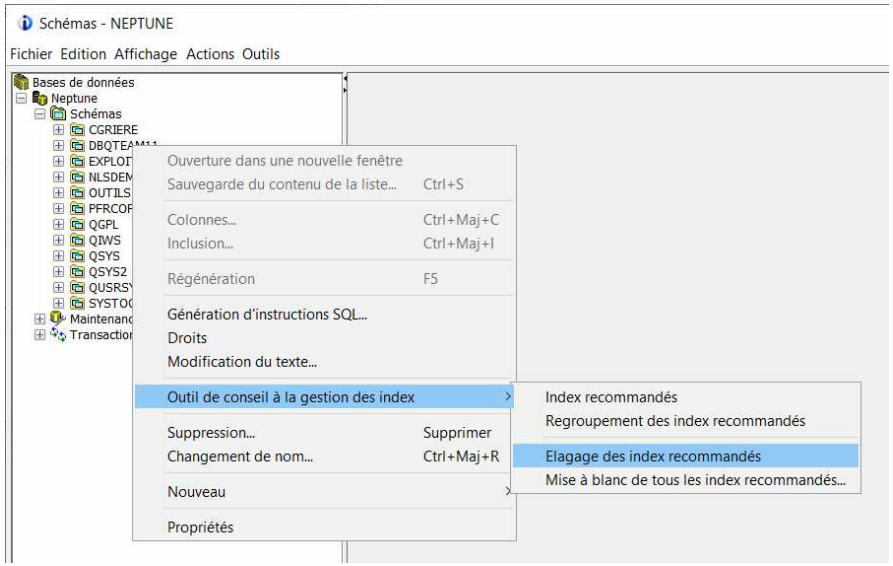
```
SELECT COUNT(*) FROM qsys2.condenseindexadvice
WHERE table_schema NOT LIKE 'Q%' ;
```

Comment diminuer leur nombre ?

- La requête n'est plus utilisée.

  WHERE last_advised > '2023-12-31 00:00:00.000000'

- La table n'existe plus.



Comment diminuer leur nombre ?

- En priorité prendre celles qui ont un MTI utilisé depuis le dernier IPL.

Index recommandés (condensés) pour DBQTEAM11.ORDERS

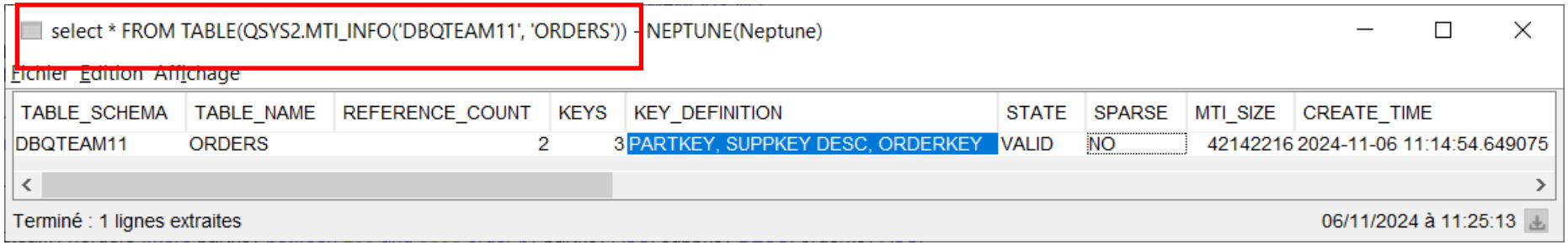
Table pour laquelle un index a été recommandé	Schéma	Clés recommandées	Dernière utilisation de l'index MTI	Nombre d'utilisations de l'index MTI	Nombre de créations de l'index MTI	Utilisation de l'index MTI pour les statistiques	Dernière utilisation de l'index MTI pour les statistiques
ORDERS	DBQTEAM11	PARTKEY, SUPPKEY, ORDERKEY	06/11/2024 11:14:54	4	2	0	01/01/0001 00:00:00

MTI réellement présent ? Vérifier la liste obtenue avec :

```
SELECT * FROM TABLE (QSYS2.MTI_INFO('*ALL','*ALL'))
```

Comment diminuer leur nombre ?

- Ajouter ceux que l'on trouve dans MTI_INFO et qui n'ont pas de MTI dans la vue CONDENSEINDEXADVICE.



Comment diminuer leur nombre ?

- Pour celles qui n'ont pas de MTI prendre un % de celles qui ont le nombre de suggestion le + élevé.

Schéma	Table nc	Table nl	Nb max lig. table	Nb suggestions depuis RAZ	Temps total estimé suggestions (s)	Index suggéré	Réduit ?	Type index	Table conv.
EXEMPLE	ZTTMVZH	ZTTMVZH	49 086 894	873 347	12 305 655	MZZZZOMHTW, MZN4WXWSZL, MZK7ZONTRT, MZDTKMHWZZ,	N	RADIX	*HEX
EXEMPLE	GZTZHTR	GZTZHTR	36 116 787	38 232	38 282	ZHZ3SOZZZR, ZHK2TVWNTN, ZHK2SWZTKO, ZHK2NUMWXW, Z	N	RADIX	*HEX
EXEMPLE	WFUTFZH	WFUTFZH	2 999 483	21 033	4 225	WKZ1GWSZZR, WKK7ZONTRT, WKZ3SOZZZR	N	RADIX	*HEX
EXEMPLE	WFUTFZH	WFUTFZH	2 999 483	15 430	15 430	WKK5ZONSOU, WKZ3SOZZZR	N	RADIX	*HEX
EXEMPLE	SKBSKNH	SKBSKNH	169 296	10 820	506 318	SKZ1GWSZZR, SKK7KTWMZZ, SKN4DOZUMW	N	RADIX	*HEX
EXEMPLE	WFUTFZH	WFUTFZH	2 999 483	6 979	6 471	WKN4RWFDOZ, WKZ1GWSZZR, WKK7ORDKTW	N	RADIX	*HEX
EXEMPLE	ZTRWNVH	ZTRWNVH	6 839	5 567	9	TJZ3SOZZZR, TJK7ZONTRT, TJZ1WTTWNV, TJZ3HWRKOD	N	RADIX	*HEX
EXEMPLE	WFUTFZH	WFUTFZH	2 999 483	4 526	0	WKN4RWFDOZ, WKK7ORDKTW, WKK5ZONSOU	N	RADIX	*HEX
EXEMPLE	ZHTDOZH	ZHTDOZH	834 303	3 153	301	DOZ1GWSDOZ, DOK7DOZUMW, DON4DOZUMW	N	RADIX	*HEX
EXEMPLE	ZHTSOLH	ZHTSOLH	1 322 047	3 130	511 796	SLZ3KMHSOL, SLZ3SOZZZR, SLDTSTKSOL, SLK5SOZKW2, SL	N	RADIX	*HEX
EXEMPLE	ZHTMVLH	ZHTMVLH	4 686 190	2 502	2 502	MLK7KTWMZZ, MLN4DOZUMW	N	RADIX	*HEX
EXEMPLE	ZTTLOGH	ZTTLOGH	5 787	2 242	2 242	LOK7KTWMZZ, LON4RWFDOZ	N	RADIX	*HEX
EXEMPLE	ZHTMVTH	ZHTMVTH	3 874 919	2 006	2 006	MTK7KTWMZZ, MTN4DOZUMW	N	RADIX	*HEX
EXEMPLE	ZHTDOZH	ZHTDOZH	834 303	1 897	272	DON4DOZUMW, DOK7KTWMZZ, DOZ3SOZZZR	N	RADIX	*HEX
EXEMPLE	ZTTWDLH	ZTTWDLH	3 570 346	1 485	50	DLZ3SOZZZR, DLK7DOZUMW, DLN4RWFDOZ	N	RADIX	*HEX
EXEMPLE	ZTTNTLR	ZTTNTLR	31 131	906	163	NLZ3SOZZZR, NLK6NTTFKN, NLZ1NTTFKN, NLK3ORDTNL, NLTS	N	RADIX	*HEX
EXEMPLE	ZHTMVLH	ZHTMVLH	4 686 190	866	7 849 802	MLK2NUMWXL, MLZ2JOURNT, MLN4WXWSZL, MLK3LKGWZR,	N	RADIX	*HEX
EXEMPLE	ZHTDOZH	ZHTDOZH	834 303	789	1 051	DOZ1GWSDOZ, DON4DOZUMW, DOK7DOZUMW	N	RADIX	*HEX
EXEMPLE	ZTTKNTH	ZTTKNTH	1 463 113	476	266	KNZ3SOZZZR, KNK2SWZTKO, KNN2MOKFZR, KNN4TNNZZR, KN	N	RADIX	*HEX
EXEMPLE	ZTRMVZH	ZTRMVZH	2 094 803	464	81 979	TTZ3SOZZZR, TTDTWXKGKB, TTZ3HWRZOM, TTK5KNTWRM, TT	N	RADIX	*HEX
EXEMPLE	SKBSKNR	SKBSKNR	506 215	463	4	SKZ3SOZZZR, SKZ1GWSZZR	N	RADIX	*HEX
EXEMPLE	ZHTKZTH	ZHTKZTH	1 478 075	436	222	ZTZ3SOZZZR, ZTVTWFLTYH	N	RADIX	*HEX
EXEMPLE	ZTTKNTH	ZTTKNTH	1 463 113	433	433	KNN4DOZUMW, KNK7KTWMZZ	N	RADIX	*HEX
EXEMPLE	ZTRZHZH	ZTRZHZH	636 387	417	417	THK7ORDKTW, THN4RWFDOZ	N	RADIX	*HEX
EXEMPLE	ZTRMVZH	ZTRMVZH	2 094 803	378	252	TTZZZOMHTW, TTDTKMHWZZ	N	RADIX	*HEX
EXEMPLE	ZHTRWGR	ZHTRWGR	434 937	351	3	RWZ3SOZZZR, RWK7RWGLWM, RWTSZRWTTK, RWN4DOZUM	N	RADIX	*HEX
EXEMPLE	ZTRMVHH	ZTRMVHH	639 593	349	8	TYZ3SOZZZR, TYZ2JOURNT, TYK2NUMWXL, TYN4WXWSZL, TY	N	RADIX	*HEX
EXEMPLE	ZTTDZOR	ZTTDZOR	1 094	331	331	ZOZ3SOZZZR	N	RADIX	*HEX

Comment diminuer leur nombre ?

Mais il faut savoir que vous risquez de créer des index inutiles dû :

- aux suggestions exclusives
- aux suggestions dépendantes dont les compteurs peuvent évoluer différemment

Quand peut-on considérer qu'un index est inutilisé ?

- Ses 2 compteurs d'utilisation SQL sont à 0.

- Il n'est pas utilisé à la place d'un logique dans un programme avec IO natifs.

*WHERE SELECT days_used_count FROM TABLE(QSYS2.OBJECT_STATISTICS (index_schema,'*FILE', system_index_name)) = 0*

- Il ne faut pas que l'index soit déclaré UNIQUE

- Il faut détecter une utilisation de la table via les autres structures d'accès

Comment diminuer leur nombre ?

Autre critère possible : pour les suggestions qui n'ont pas de MTI prendre le temps moyen d'exécution de la requête qui a généré la suggestion et qui est présent dans la vue.

Schéma	Table nc	Table nl	Nb max lig. table	Nb suggestions depuis RAZ	Temps total estimé suggestions (s)	Index suggéré	Réduit ?	Type index	Table conv.
EXEMPLE	ZTTMVZH	ZTTMVZH	49 086 894	873 347	12 305 655	MZZZZOMHTW, MZN4WXWSZL, MZK7ZONTRT, MZDTKMHWZZ,	N	RADIX	*HEX
EXEMPLE	ZHTMVLH	ZHTMVLH	4 686 190	866	7 849 802	MLK2NUMWXL, MLZ2JOURNT, MLN4WXWSZL, MLK3LKGWZR,	N	RADIX	*HEX
EXEMPLE	ZHTSOLH	ZHTSOLH	1 322 047	3 130	511 796	SLZ3KMHSOL, SLZ3SOZZZR, SLDTSTKSOL, SLK5SOZKW2, SL	N	RADIX	*HEX
EXEMPLE	SKBSKNH	SKBSKNH	169 296	10 820	506 318	SKZ1GWSZZR, SKK7KTWMZZ, SKN4DOZUMW	N	RADIX	*HEX
EXEMPLE	ZTRMVZH	ZTRMVZH	2 094 803	464	81 979	TTZ3SOZZZR, TTDTWXKGKB, TTZ3HWRZOM, TTK5KNTWRM, TT	N	RADIX	*HEX
EXEMPLE	GZTZHTR	GZTZHTR	36 116 787	38 232	38 282	ZHZ3SOZZZR, ZHK2TVWNTN, ZHK2SWZTKO, ZHK2NUMWXW, Z	N	RADIX	*HEX
EXEMPLE	WFUTFZH	WFUTFZH	2 999 483	15 430	15 430	WKK5ZONSOU, WKZ3SOZZZR	N	RADIX	*HEX
EXEMPLE	WFUTFZH	WFUTFZH	2 999 483	6 979	6 471	WKN4RWFDOZ, WKZ1GWSZZR, WKK7ORDKTW	N	RADIX	*HEX
EXEMPLE	WFUTFZH	WFUTFZH	2 999 483	21 033	4 225	WKZ1GWSZZR, WKK7ZONTRT, WKZ3SOZZZR	N	RADIX	*HEX
EXEMPLE	ZHTMVLH	ZHTMVLH	4 686 190	2 502	2 502	MLK7KTWMZZ, MLN4DOZUMW	N	RADIX	*HEX
EXEMPLE	ZTTLOGH	ZTTLOGH	5 787	2 242	2 242	LOK7KTWMZZ, LON4RWFDOZ	N	RADIX	*HEX
EXEMPLE	ZHTMVTH	ZHTMVTH	3 874 919	2 006	2 006	MTK7KTWMZZ, MTN4DOZUMW	N	RADIX	*HEX
EXEMPLE	ZHTDOZH	ZHTDOZH	834 303	789	1 051	DOZ1GWSDOZ, DON4DOZUMW, DOK7DOZUMW	N	RADIX	*HEX
EXEMPLE	ZTTKNTH	ZTTKNTH	1 463 113	433	433	KNN4DOZUMW, KNK7KTWMZZ	N	RADIX	*HEX
EXEMPLE	ZTRZHZH	ZTRZHZH	636 387	417	417	THK7ORDKTW, THN4RWFDOZ	N	RADIX	*HEX
EXEMPLE	ZTTDZOR	ZTTDZOR	1 094	331	331	ZOZ3SOZZZR	N	RADIX	*HEX
EXEMPLE	ZHTDOZH	ZHTDOZH	834 303	3 153	301	DOZ1GWSDOZ, DOK7DOZUMW, DON4DOZUMW	N	RADIX	*HEX
EXEMPLE	ZHTDOZH	ZHTDOZH	834 303	1 897	272	DON4DOZUMW, DOK7KTWMZZ, DOZ3SOZZZR	N	RADIX	*HEX
EXEMPLE	ZTTKNTH	ZTTKNTH	1 463 113	476	266	KNZ3SOZZZR, KNK2SWZTKO, KNN2MOKFZR, KNN4TNNZZR, KN	N	RADIX	*HEX
EXEMPLE	ZTRMVZH	ZTRMVZH	2 094 803	378	252	TTZZZOMHTW, TTDTKMHWZZ	N	RADIX	*HEX
EXEMPLE	ZHTKZTH	ZHTKZTH	1 478 075	436	222	ZTZ3SOZZZR, ZTVTWFLTYH	N	RADIX	*HEX
EXEMPLE	ZTTNTLR	ZTTNTLR	31 131	906	163	NLZ3SOZZZR, NLK6NTTFKN, NLZ1NTTFKN, NLK3ORDTNL, NLTS	N	RADIX	*HEX
EXEMPLE	ZTTWDLH	ZTTWDLH	3 570 346	1 485	50	DLZ3SOZZZR, DLK7DOZUMW, DLN4RWFDOZ	N	RADIX	*HEX
EXEMPLE	ZTRWNVH	ZTRWNVH	6 839	5 567	9	TJZ3SOZZZR, TJK7ZONTRT, TJZ1WTTWNV, TJZ3HWRKOD	N	RADIX	*HEX
EXEMPLE	ZTRMVHH	ZTRMVHH	639 593	349	8	TYZ3SOZZZR, TYZ2JOURNT, TYK2NUMWXL, TYN4WXWSZL, TY	N	RADIX	*HEX
EXEMPLE	SKBSKNR	SKBSKNR	506 215	463	4	SKZ3SOZZZR, SKZ1GWSZZR	N	RADIX	*HEX
EXEMPLE	ZHTRWGR	ZHTRWGR	434 937	351	3	RWZ3SOZZZR, RWK7RWGLWM, RWTSZRWTTK, RWN4DOZUM	N	RADIX	*HEX
EXEMPLE	WFUTFZH	WFUTFZH	2 999 483	4 526	0	WKN4RWFDOZ, WKK7ORDKTW, WKK5ZONSOU	N	RADIX	*HEX

Conclusion

- L'optimiseur délivre des suggestions équilibrées qui aident à l'optimisation unitaire des requêtes.
- La meilleure optimisation SQL est celle qui adresse l'environnement et les requêtes SQL les + consommatrices/utilisées.
- L'exploitation globale des suggestions peut être considérée sous réserve des limitations que nous avons vues.

Questions/Réponses

Merci à GAIA

pour l'accès à leur système qui m'a permis de réaliser cette présentation

Merci pour
votre présence
et
votre participation