Université IBM i 2018

16 et 17 mai

IBM Client Center Paris

Session S41 – LFs, vues et index, ne les confondez pas !

Nathanaël Bonnet
Gaia
nathanael.bonnet@gaia.fr

Gaia

- Conseil et formation IBM i depuis 1995
  - Inter et intra entreprise
- Base de connaissance en ligne
  - https://know400.gaia.fr
- Organisateur des matinées 400 iday
  - https://www.gaia.fr/400iday-3

GAIA

https://www.gaia.fr

https://twitter.com/GaiaFrance

Plan de la présentation

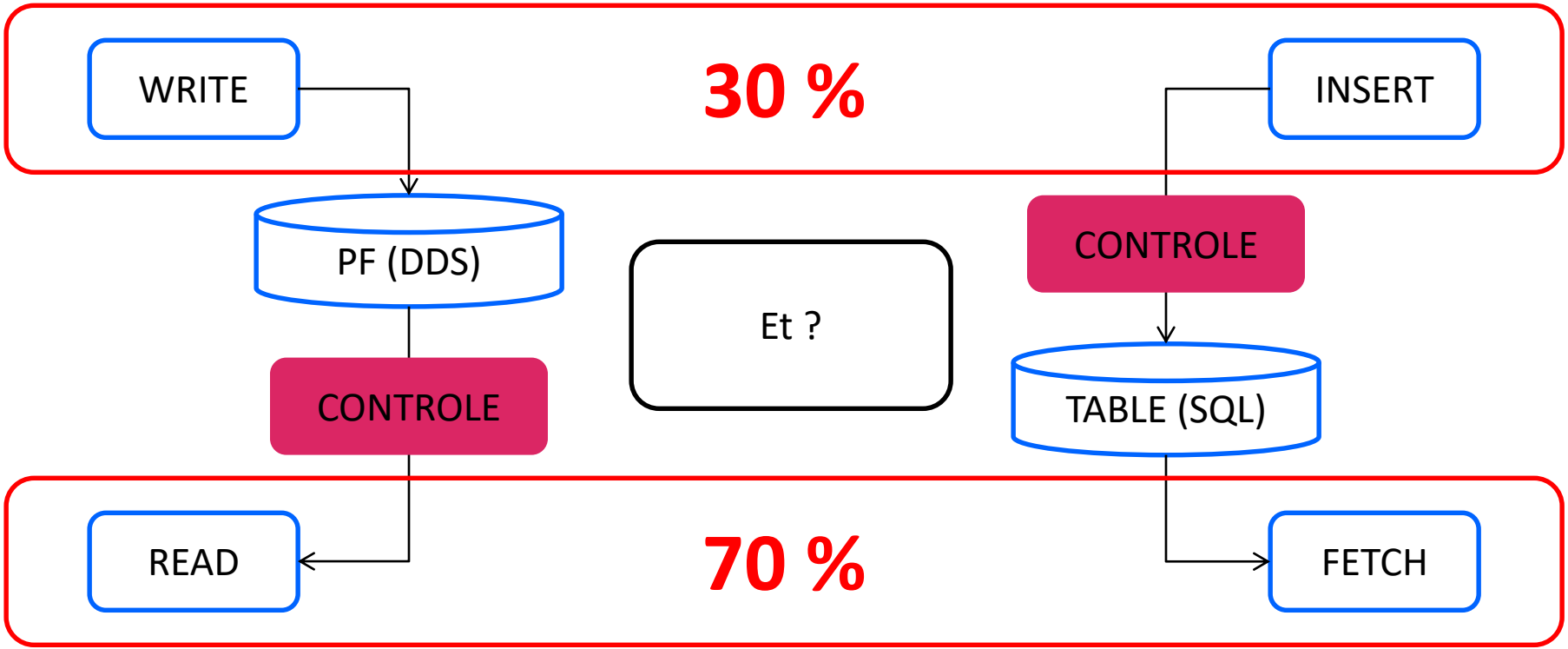
- Introduction
- DDS (RLA) vs DDL (SQL)
- LF vs Index
- LF vs Vue
- Modernisation
- Conclusion

Introduction

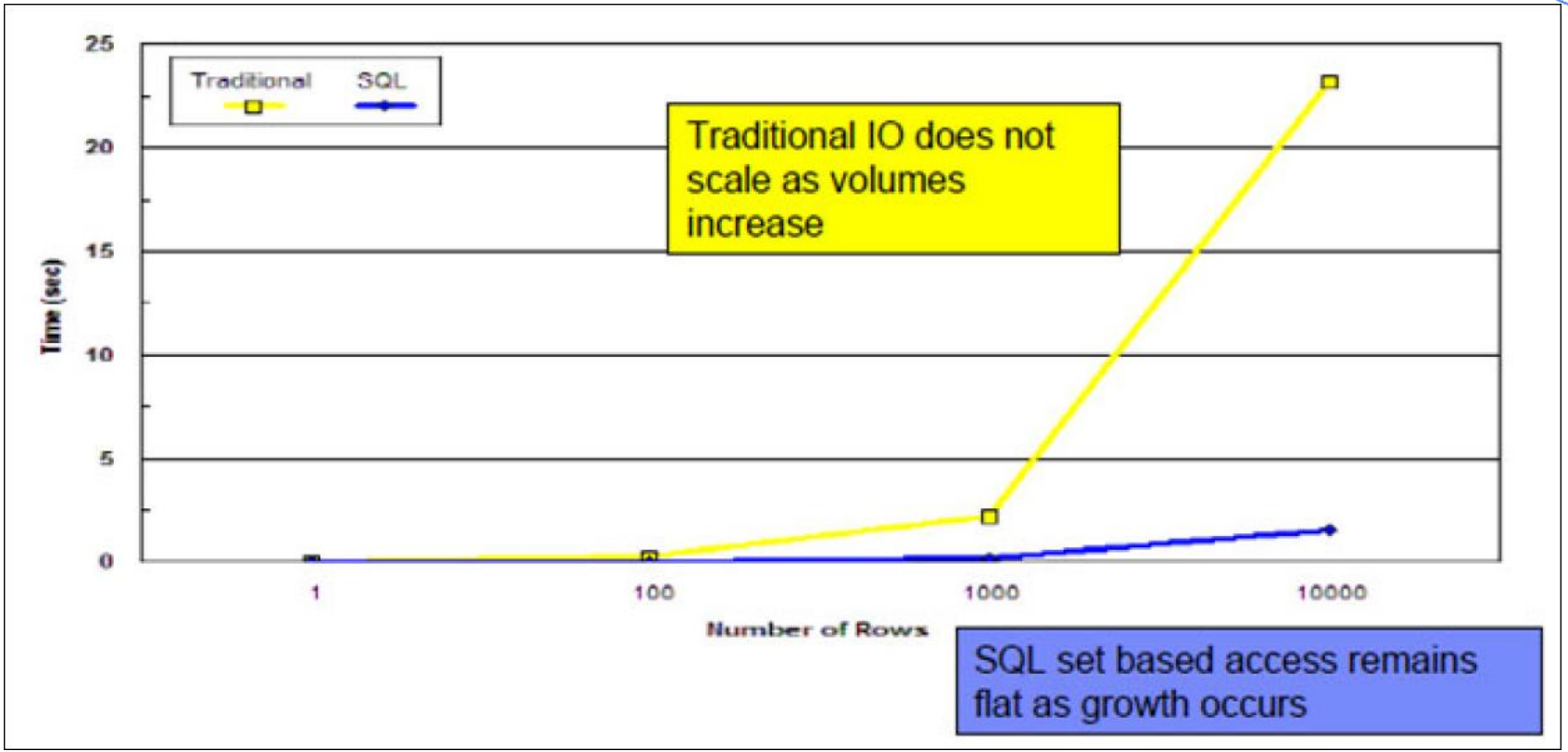
LF et Index

- Modernisation BD
  - DDS vers DDL
  - DDS stabilisé (6.1)
  - Nouvelles fonctionnalités uniquement en SQL
  - Ressources plus faciles à trouver
- Transparent
  - Equivalences DDS <-> DDL gérées par le système
- Qualité des données
  - Contrôle avant / après
  - Temps de réponse

Qualité des données



Performance



Source : red book « Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between »
http://www.redbooks.ibm.com/abstracts/sg248185.html?Open

DDS (RLA) vs DDL (SQL)

Concepts et vocabulaire

IBM i	DB2 for i
Bibliothèque	Schéma, Collection ou BD
Fichier physique	Table
Enregistrement	Ligne
Champ / Zone	Colonne
Logique	Index, vue
Format	
Membre	
Nom = 10 caractères	Nom = 128 caractères
	Catalogue
Journalisation	Journalisation
	Intégrité référentielle

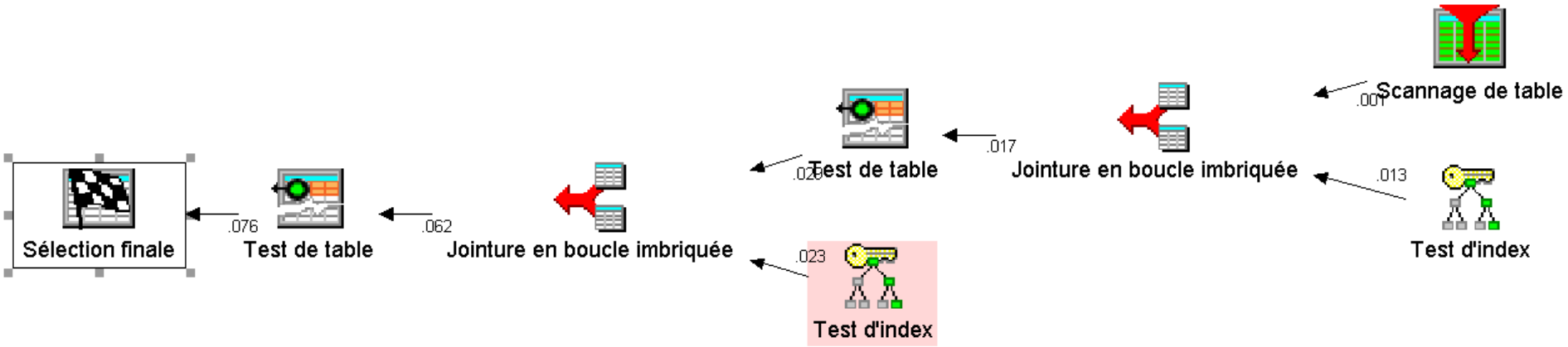
Fonctionnement

- DDS / « carte F »
  - Compilé
  - Traitement ligne à ligne (RLA)
  - Le programmeur prévoit la logique d'accès et la code dans ses programmes
    - Le programmeur fait l'optimisation
    - Avantage : maitrisé par les développeurs
- DDL / SQL
  - Script interprété
  - Traitement ensembliste
  - Le programmeur indique le résultat souhaité
    - Le moteur SQL fait l'optimisation
      - Analyse et réécrit les requêtes. Nécessite des infos contextuelles (statistiques).
    - Avantage : moins de travail pour le développeur et meilleures performances

Plan d'exécution

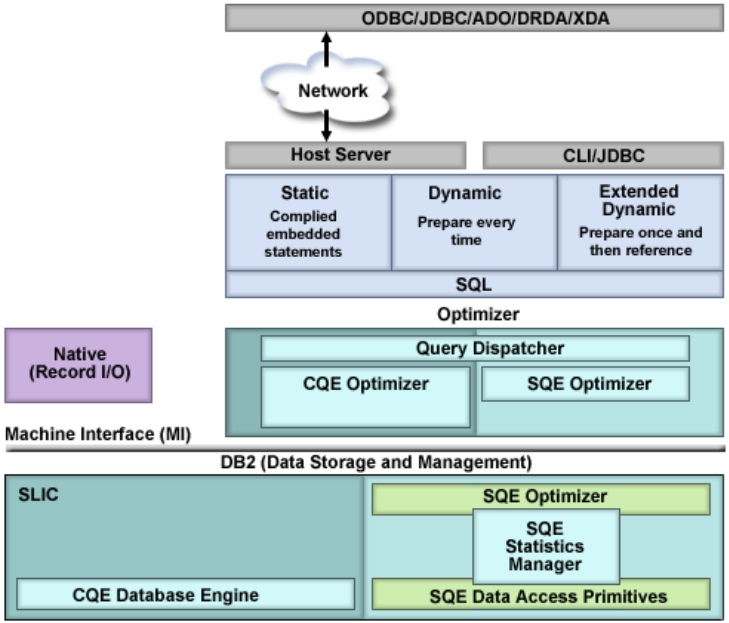
- Requête

```
SELECT e.empno, TRIM(e.firstnme)|| ' ' || TRIM(e.lastname) AS employe , e.job, e.workdept,
    d.deptname, TRIM(mgr.firstnme)|| ' ' || TRIM(mgr.lastname) AS manager, d.admrdept
    FROM employee e
    JOIN department d ON e.workdept = d.deptno LEFT
    JOIN employee mgr ON d.mgrno = mgr.empno
```



Optimiseur(s)

- 2 moteurs d'optimisation des requêtes
  - CQE : Classical Query Engine
    - Moteur historique pour QRY/400
  - SQE : SQL Query Engine
    - Moteur SQL
- Aujourd'hui
  - La quasi-totalité des requêtes passent par SQE
  - Même si les données sont dans des PF/LF
- Références
  - https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20i%20Technology%20Updates/page/Classic%20Query%20Engine%20(CQE)%20and%20SQL%20Query%20Engine%20(SQE)%20Differences



LF / Index ou vue

- En SQL, nous disposons de deux types de « fichiers logiques »
  - Index
    - Non utilisable dans une requête SQL
      - Uniquement pour le fonctionnement de l'optimiseur
    - Utilisable dans les I/O natives (RLA)
    - Dispose d'une clé, unique ou non
  - Vue
    - Utilisable dans une requête SQL
    - Utilisable dans les I/O natives (RLA)
      - En lecture (restrictions pour modification)
    - Aucune clé possible

PF / Table

- Comparaison

	PF	TABLE
Objet	*FILE / PF	
Nom	Système : 10	Système : 10 SQL : 128
Création	CRTPF + source	CREATE TABLE
Formats	Plusieurs	1 Par défaut format = membre = table RCDFMT
Membres	Plusieurs	1
Clés	Aucune Unique Non unique	Aucune Unique

LF vs Index

Présentation

- Les index sont souvent présentés comme étant l'équivalent des LF pour SQL
  - Ils peuvent faire bien plus
  - Et sont indispensables au bon fonctionnement de DB2 en mode SQL
- DB2 supporte maintenant de multiples possibilités
  - Index radix (depuis 1988)
  - Index EVI (depuis 1998)
  - Index dérivé
- Hors périmètre
  - Index Omnifind
    - Stocké dans l'IFS pour recherche « contient » et/ou linguistique

LF / Index

- Comparaison

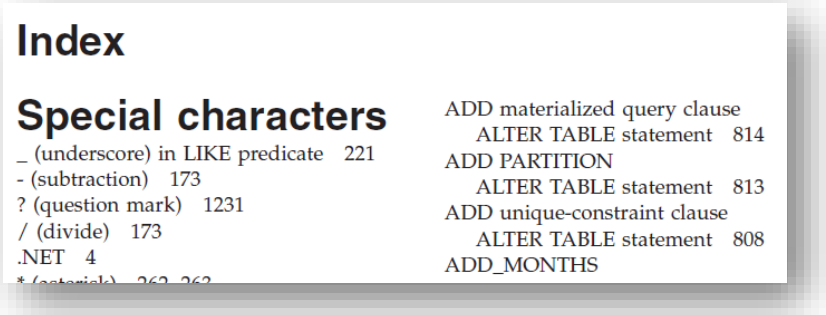
	LF	INDEX
Objet		*FILE / LF
Nom	Système : 10	Système : 10 SQL : 128
Création	CRTLF + source	CREATE INDEX
Formats	Plusieurs	1 Hérité de la table RCDFMT
Membres	Plusieurs	1
Taille de la page	Défaut : 8 à 32 Ko	Défaut : 64 Ko

LF / Index

	LF	INDEX
Clés	Aucune Unique Non unique	 Unique Non unique
Ordre	Liste de zones	Liste de colonnes
Sélection de zones / colonnes	Toutes les zones du format Liste Valeurs calculées	Toutes les colonnes du format Liste Valeurs calculées
Sélection d'enregistrements / lignes	Sélection / omission	WHERE
Jointure	JOIN/JFILE/JFLD	n/a

Principe

- Un index
  - Est une structure entretenue automatiquement par DB2
  - Permet d'accélérer les opérations de recherche, tri, jointure ou d'agrégation réalisées par la base de données
  - Référence certaines informations contenues dans une table suivant un ordre déterminé
  - Peut être comparé à un index de livre qui référence des pages contenant un mot-clé



Statistiques

- Des statistiques
  - Sont associées à chaque index
  - Elles sont prises en charge par le moteur SQE pour déterminer si l'index est pertinent, au-delà des zones déclarées
  - Une partie des statistiques est accessible via **DSPFD**
  - Plus de détails via le catalogue DB2

Création, suppression

- Les opérations possibles
  - Création et suppression
  - `CREATE INDEX nom_index ... ;`
  - `DROP INDEX nom_index ;`
    - L'index ne doit pas être verrouillé
  - Il est impossible de modifier un index
    - Il faudra le supprimer et le recréer
- Ces instructions peuvent être exécutées
  - Via n'importe quelle interface SQL
    - STRSQL, RUNSQLSTM, RUNSQL, SQL embarqué, ODBC/JDBC/.Net provider ...
  - Via les assistants spécifiques de System i Navigator et IBM Navigator for i

Bonnes pratiques

- Pour ajouter un texte descriptif à vos index

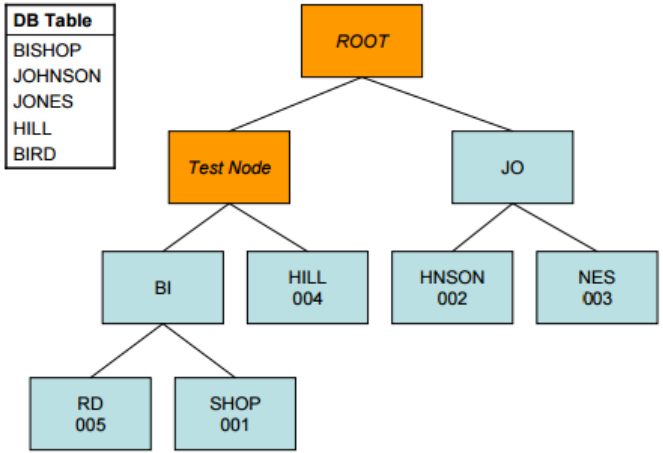
```
LABEL ON INDEX nom_index
  IS 'Description de l''index' ;
```

- Bibliothèque
  - Il est recommandé de placer l’index dans la même bibliothèque que la table sous-jacente

Index Radix

Définition

- Les index de type radix disposent d'une structure en "B-arbre"
  - Structure hiérarchique qui tend à équilibrer les différentes branches



Usage

- Cette structure est particulièrement adaptée
  - Au stockage d'un grand nombre de valeurs
    - Les branches seront multipliées pour maintenir un nombre limité de niveaux
  - Avec des temps d'accès très rapides
    - Globalement le nombre de niveaux de l'arbre
  - C'est la structure par défaut d'index utilisé par DB2
    - Aussi bien par **`CREATE INDEX`** que par **`CRTLF`**
  - C'est également la structure utilisé par DB2 pour stocker les clés
    - Clé primaire, clé unique ou clé étrangère

Syntaxe 1/7

- Générale

```
CREATE [UNIQUE] INDEX nom_index
   ON nom_table (colonne1 [ASC | DESC], ... )
```

  - Les noms d'index et de table peuvent être qualifiés ou non

UNIQUE

  - Chaque valeur de clé insérée dans la table devra être unique
  - Sinon l'enregistrement sera rejeté

- Exemples
  - Index avec clé unique sur la colonne **ACTNO** (ASC par défaut)

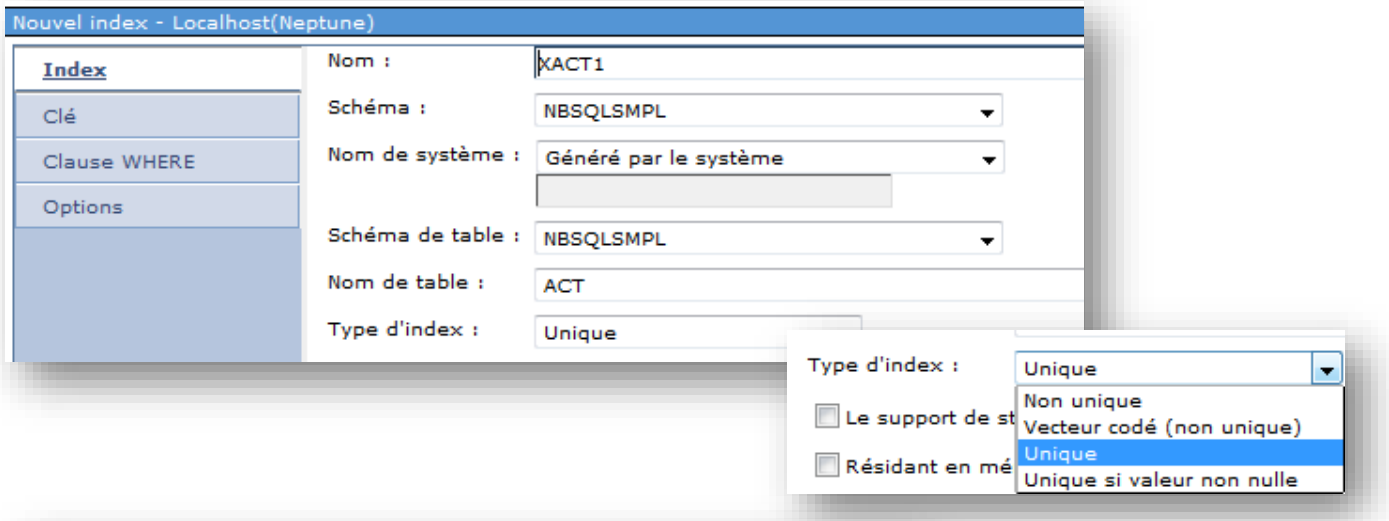
```
CREATE UNIQUE INDEX NBSQLSMPL.XACT1
                    ON NBSQLSMPL.ACT ( ACTNO ) ;
```

  - Index avec clé non unique sur les colonnes **EMPNO** et **PHOTO_FORMAT** avec sens de tri explicite

```
CREATE INDEX XEMP_PHOTO
            ON EMP_PHOTO ( EMPNO DESC, PHOTO_FORMAT ASC ) ;
```

Syntaxe 2/7

- Equivalent via l'assistant



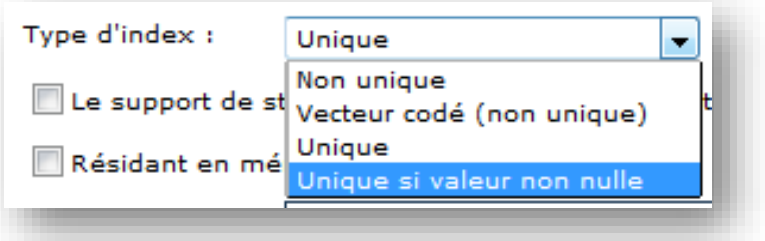
Syntaxe 3/7

- Autres options

UNIQUE WHERE NOT NULL

  - UNIQUE, sauf si la valeur de la clé est NULL

```
CREATE UNIQUE WHERE NOT NULL INDEX XACT1
                                    ON ACT ( ACTNO ) ;
```

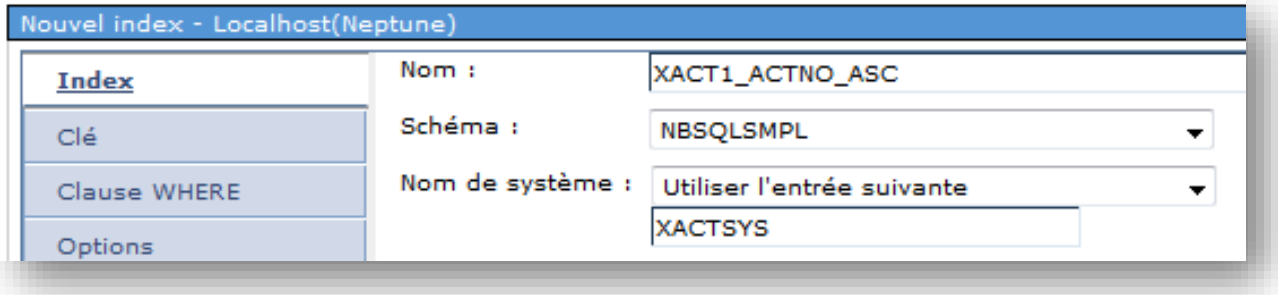


Syntaxe 4/7

FOR SYSTEM NAME *nom_système*

- Permet d'indiquer un nom de fichier système sur 10 caractères
- Pour rappel, le nom SQL peut aller jusqu'à 128 caractères « mixed case »

```
CREATE UNIQUE INDEX XACT1_ACTNO_ASC
    FOR SYSTEM NAME XACTSYS
    ON ACT (ACTNO ASC) ;
```



Syntaxe 5/7

- **RCDFMT**
  - Permet d'indiquer un nom de format pour la compatibilité avec le système de fichier natif, principalement en migration de structures de données DDS vers SQL
  - Par défaut en SQL
    - nom de format = nom du fichier
  - Il est aussi possible d'indiquer

ADD ALL COLUMNS

Ajout de toutes les colonnes dans l'index

ADD KEYS ONLY (défaut)

Seules les colonnes spécifiées dans l'index sont comprises dans le format

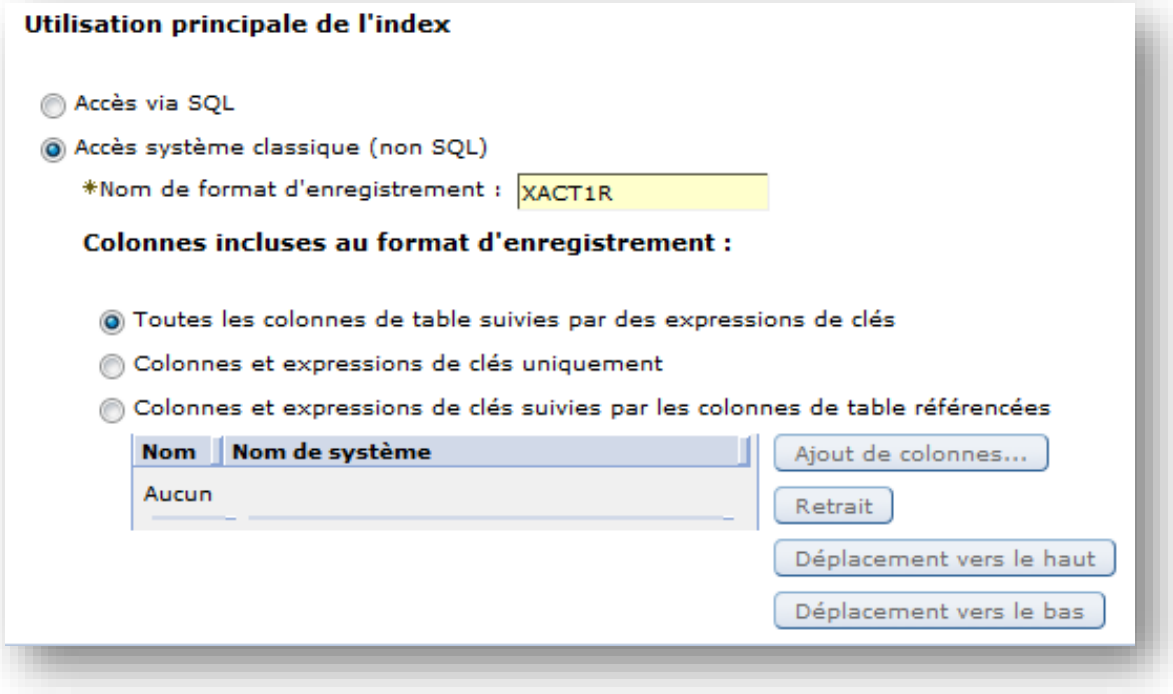
ADD col1, col2 …

les colonnes indiqués sont ajoutées au format

```
CREATE INDEX XACTNB ON ACT (ACTNO ASC)
RCDFMT XACT1R ADD ALL COLUMNS ;
```

Syntaxe 6/7

- **RCDFMT**
  - Equivalent assistant, via l'onglet « options »

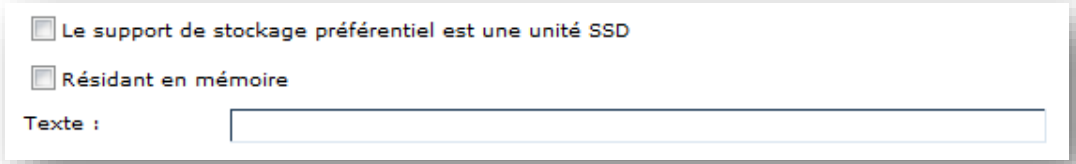


Syntaxe 7/7

- **UNIT ANY** ou **UNIT SSD**
  - Indique si le stockage doit se faire de préférence sur support SSD
- **KEEP IN MEMORY YES** ou **NO**
  - Indique que l'on souhaite maintenir cet index en mémoire principale lorsqu'utilisé pour une requête

```
CREATE INDEX XACTNB ON NBSQLSMPL.ACT (ACTNO ASC)
RCDFMT XACT1R ADD ALL COLUMNS
UNIT ANY
KEEP IN MEMORY NO ;
```

- Dans l'onglet « Index »



EVI – Encoded Vector Index

Encoded Vector Index

- Ce sont des index bitmaps avancés
  - Brevet spécifique IBM
  - La structure n’est pas un arbre-b comme les index radix
- Usage
  - Très performant pour indexer un petit nombre de valeurs pour un grand nombre de lignes (code produit, taux tva, année, …)
  - Des données peu vivantes : les valeurs et domaines de valeurs ne doivent pas changer en permanence
  - Gros volumes

Principe

- Par exemple, un index EVI sur la colonne DEPT de la table des employés EMPLOYE

```
SELECT * FROM EMPLOYE WHERE DEPT = 'INFO'
```

Table EMPLOYE

RRN	DEPT	…
1	COMPTA	
2	INFO	
3	INFO	
4	PROD	
5	COMPTA	
6	PROD	
7	INFO	
8	COMPTA	

Table des symboles

Clé	Code	Début	Fin	Nb
COMPTA	A	1	8	3
INFO	B	2	7	3
INFO	B	2	7	3
PROD	C	4	6	2
COMPTA	A	1	8	3
PROD	C	4	6	2
INFO	B	2	7	3
COMPTA	A	1	8	3

Vector
A
B
B
C
A
C
B
A

Bitmap
0
1
1
0
0
0
1
0

- Index bitmap classique

Valeur	bitmap
COMPTA	10001001
INFO	01100010
PROD	00010100

Principe

- Les index bitmaps
  - Permettent un stockage performant en termes d'espace occupé
  - Permettent des opérations très rapides sur les sélections, même multiples
    - `WHERE DEPT = 'INFO' OR DEPT = 'COMPTA'`
    - Résolu par une opération OR binaire sur les vecteurs bitmaps correspondants
  - L'optimiseur peut utiliser plusieurs index EVI pour satisfaire une unique condition

Syntaxe 1/2

- Générale

```
CREATE ENCODED VECTOR INDEX nom_index
    ON nom_table (colonne1 [ASC | DESC], ... )
```

  - Les noms d'index et de table peuvent être qualifiés ou non

- Exemples
  - EVI sur colonne `DEPT` de la table `EMPLOYE`

```
CREATE ENCODED VECTORE INDEX EMP_DEPT
                              ON EMPLOYE ( DEPT ) ;
```

  - EVI sur colonnes `DEPT` et `JOB` de la table `EMPLOYE`

```
CREATE ENCODED VECTORE INDEX EMP_DEPT
                              ON EMPLOYE ( DEPT, JOB DESC ) ;
```

Syntaxe 2/2

- **`WITH x DISTINCT VALUES`**
  - Indique un nombre estimé de clés distinctes, afin de déterminer la taille initiale des codes pour la table des symboles (1, 2 ou 4 octets)

```
CREATE ENCODED VECTOR INDEX EVI_IDX
  ON EMPLOYEE (WORKDEPT ASC)
  WITH 255 DISTINCT VALUES ;
```

- Les options **`UNIT ANY`** ou **`SSD`** et **`KEEP IN MEMORY NO`** ou **`YES`**
  - Sont identiques aux index radix

Comparaison

- Par rapport aux index radix
  - Avec une clé composée

```
CREATE INDEX nom_idx ON nom_tbl (col1, col2, col3 );
```

  - Radix
    - L'ordre des zones dans la clé est très important
    - Du plus discriminant au moins discriminant
  - EVI
    - Vous pouvez créer des EVI contenant plusieurs colonnes

```
CREATE ENCODED VECTORE INDEX nom_idx ON nom_tbl (col1, col2, col3 );
```

    - Ou plusieurs EVI

```
CREATE ENCODED VECTOR INDEX nom_idx ON nom_tbl (col1);
CREATE ENCODED VECTOR INDEX nom_idx ON nom_tbl (col2);
CREATE ENCODED VECTOR INDEX nom_idx ON nom_tbl (col3);
```

    - Le moteur SQE est capable de mixer ces 3 EVI pour effectuer sa requête

Index dérivés

Principe

- Capacités d'utiliser
  - Des expressions
    - Des colonnes calculées
  - Des sélections
    - Sélectionner les lignes utiles
- Exemple

```
CREATE INDEX EMPLOYEE_SEARCH ON EMPLOYEE (
   UPPER(FIRSTNME) AS FIRSTNME_UP,
   SALARY +  BONUS  +  COMM AS REVENUS )
WHERE JOB <> 'MANAGER' ;
```

Principe

- Nom des colonnes
  - Si vous n'indiquez pas **AS**
    - DB2 calcule lui-même le nom des colonnes : **IXCOL00001**, **IXCOL00002** ...
  - Si vous indiquez **AS**
    - Vous devez fournir un nom de colonne qui n'existe pas dans la table
  - Vous pouvez indiquer
    - **AS** et **FOR SYSTEM NAME**

Index EVI

- Pour les index EVI, il est possible d'ajouter la clause **INCLUDE**
  - Permet d'ajouter à l'index des valeurs d'agrégation par les fonctions
    - **AVG, COUNT, COUNT_BIG, SUM, STDDEV, STDDEV_SAMP, VARIANCE** ou **VARIANCE_SAMP**
    - Le mot-clé **DISTINCT** ne peut pas être utilisé

```
CREATE ENCODED VECTOR INDEX EVI_DEMO
  ON EMPLOYEE (EMPNO ASC, UPPER(FIRSTNME) ASC)
  WHERE JOB <> 'MANAGER'
  INCLUDE(AVG(SALARY), COUNT(*)) ;
```

  - Ces informations sont exploitées par les requêtes utilisant des fonctions OLAP telles que **GROUPING SET, ROLLUP** ou **CUBE**

Limites

- Expressions de clé et sélection
  - Ne peuvent pas être utilisés
    - Les sous-requêtes
    - Les fonctions d’agrégation
    - Les variables globales
    - Les séquences
    - Les fonctions OLAP
    - L'expression ROW_CHANGE
    - Le prédicat REGEXP_LIKE
    - Toute fonction non déterministe

Limites

- Les fonctions suivantes
  - **ATAN2, CARDINALITY, CONTAINS, CURDATE, CURTIME, DATAPARTITIONNAME, DATAPARTITIONNUM, DAYNAME, DBPARTITIONNAME, DECRYPT_BINARY, DECRYPT_BIT, DECRYPT_CHAR, DECRYPT_DB, DIFFERENCE, DLURLCOMPLETE,DLURLPATH, DLURLPATHONLY, DLURLSCHEME, DLURLSERVER, DLVALUE, ENCRYPT_AES, ENCRYPT_RC2, ENCRYPT_TDES, GENERATE_UNIQUE, GETHINT, IDENTITY_VAL_LOCAL, INSERT, LPAD, LOCATE_IN_STRING, MAX_CARDINALITY, MONTHNAME, MONTHS_BETWEEN, NEXT_DAY, OVERLAY, NOW, RAISE_ERROR, RAND, REGEXP_COUNT, REGEXP_INSTR, REGEXP_REPLACE, REGEXP_SUBSTR, REPEAT, REPLACE, ROUND_TIMESTAMP, RPAD, SCORE, SOUNDEX, TABLE_NAME, TABLE_SCHEMA, TIMESTAMP_FORMAT, TIMESTAMPDIFF, TRUNC_TIMESTAMP, VARCHAR_FORMAT, VERIFY_GROUP_FOR_USER, WEEK_ISO, WRAP, XMLPARSE, XMLVALIDATE, XSLTRANSFORM**

LF vs Vue

Présentation

- Une vue est souvent présentée, à tort, comme l'équivalent SQL d'un LF avec sélection/omission et/ou jointure

- Une vue
  - Contient une instruction SELECT
    - Décrit les données à utiliser
  - Ne contient aucune donnée
    - Ne peut pas contenir de clé

LF / Vue

- Comparaison

	LF	Vue
Objet		*FILE / LF
Nom	Système : 10	Système : 10 SQL : 128
Création	CRTLF + source	CREATE VIEW
Formats	Plusieurs	1 RCDFMT
Membres	Plusieurs	1
Taille de la page	Défaut : 8 à 32 Ko	n/a

LF / Vue

	LF	Vue
Clés	Aucune Unique Non unique	Aucune
Ordre	Liste de zones	n/a
Sélection de zones / colonnes	Toutes les zones du format Liste Valeurs calculées	Toutes les colonnes du format Liste Valeurs calculées
Sélection d'enregistrements / lignes	Sélection / omission	WHERE
Jointure	JOIN/JFILE/JFLD	JOIN (INNER, FULL, CROSS, LEFT/RIGHT, EXCEPTION)
Agrégation	n/a	GROUP BY

Usages

- Besoins techniques
  - Conversion de données
    - Dates numériques ou alpha -> date
  - Nom et structure des données à adapter
    - Renommage, concaténation, découpage, jointure, …
  - Sécurité
    - Interdire l’accès aux données sensibles
  - Réduire les dépendances
    - Entre la structure de la BD et les consommateurs

Usages

- Performance
  - Contrairement aux LF et INDEX, la création de vues n'a aucune impact sur les performances
    - Aucune donnée à maintenir
    - Le SELECT contenu dans la vue est évalué lors de son utilisation
- Imbrication de vues
  - Une vue peut être construite sur d'autres vues
  - Permet d'obtenir des résultats nécessitant des étapes

Exemples

- Limiter les informations, calculer et renommer les colonnes

```
/* Infos générales employés */
create view employeeInfo for system name empinfo as (
  select empno as matricule,
         trim( firstnme ) || ' ' || trim(lastname) as nom,
         job as poste,
         workdept as dept,
         year( hiredate - birthdate ) as age_embauche
  from employee ) ;
```

MATRICULE	NOM	POSTE	DEPT	AGE_EMBAUCHE
000010	CHRISTINE HAAS	PRES	A00	31
000020	MICHAEL THOMPSON	MANAGER	B01	25
000030	SALLY KWAN	MANAGER	C01	33
000050	JOHN GEYER	MANAGER	E01	23
000060	IRVING STERN	MANAGER	D11	28
000070	EVA PULASKI	MANAGER	D21	27
000090	EILEEN HENDERSON	MANAGER	E11	29
000100	THEODORE SPENSER	MANAGER	E21	23
000110	VINCENZO LUCCHESSI	SALESREP	A00	28
000120	SEAN O'CONNELL	CLERK	A00	21

Exemples

```
/* Département avec info du manager et nombre d'employés */
create view deptInfo as (
  with cte_nb_emp as (select workdept, count(*) as nb from employee group by workdept)
  select d.deptno ,
         d.deptname as nom_dept,
         e.firstnme as prenom_mgr,
         e.lastname as nom_mgr,
         n.nb as nb_emp
  from           department d
       left join employee e     on d.mgrno = e.empno
       left join cte_nb_emp n   on n.workdept = d.deptno ) ;
```

DEPTNO	NOM_DEPT	PRENOM_MGR	NOM_MGR	NB_EMP
A00	SPIFFY COMPUTER SERVICE DIV.	CHRISTINE	HAAS	5
B01	PLANNING	MICHAEL	THOMPSON	1
C01	INFORMATION CENTER	SALLY	KWAN	4
D01	DEVELOPMENT CENTER	-	-	-
D11	MANUFACTURING SYSTEMS	IRVING	STERN	11
D21	ADMINISTRATION SYSTEMS	EVA	PULASKI	7
E01	SUPPORT SERVICES	JOHN	GEYER	1
E11	OPERATIONS	EILEEN	HENDERSON	7
E21	SOFTWARE SUPPORT	THEODORE	SPENSER	6
F22	BRANCH OFFICE F2	-	-	-
G22	BRANCH OFFICE G2	-	-	-
H22	BRANCH OFFICE H2	-	-	-

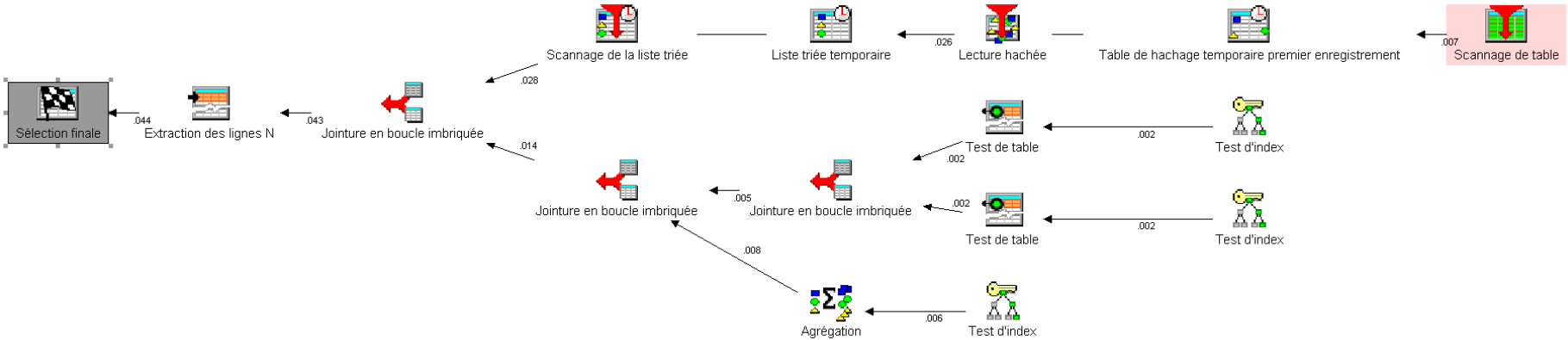
Exemples

```
/* 3 départements avec les employés les plus jeunes à l'embauche */
select d.deptno,
       d.nom_dept,
       d.prenom_mgr,
       d.nom_mgr,
       d.nb_emp,
       a.age_embauche_moy
from deptInfo d
join ( select dept,
              avg( age_embauche ) as age_embauche_moy
       from employeeInfo e
       group by dept ) a on d.deptno = a.dept
order by a.age_embauche_moy asc
limit 3 ;
```

DEPTNO	NOM_DEPT	PRENOM_MGR	NOM_MGR	NB_EMP	AGE_EMBAUCHE_MOY
E01	SUPPORT SERVICES	JOHN	GEYER	1	23
D11	MANUFACTURING SYSTEMS	IRVING	STERN	11	24
B01	PLANNING	MICHAEL	THOMPSON	1	25

Exemple

- Plan d’exécution



Possibilités avancées

- L’utilisation de vue offre de nombreuses possibilités non admises pour les LF
  - Toutes les fonctions scalaires
  - Fonction utilisateur (UDF), fonction utilisateur table (UDTF)
    - JSON_TABLE, XMLTABLE …
  - Groupage
  - Requêtes hiérarchiques, récursives
  - Sous-requêtes
  - UNION / INTERSECT / EXCEPT
  - Registres
  - Variables globales
  - Vues imbriquées

Modernisation

Etape 1 : migration DDL

- Passage de DDS à DDL : 2 composantes
  - Les données
    - Gérer les données incohérentes
  - Impacts sur les programmes
    - Mots-clés d'édition principalement
    - Multi-formats, multi-membres

- Utilisez des outils !

Etape 2 : mise en œuvre des contraintes

- Différents types de contraintes
  - Clé primaire
    - Ce devrait déjà être le cas …
    - Exemple : l'identifiant de la facture est le n° de facture « numfact »
  - D'intégrité de domaine de valeurs
    - Simple
    - Exemple : donnée possible pour la colonne « type » = { 'M.', 'Mme', 'X'}
  - D'intégrité référentielle
    - Gestion des impacts
    - Exemple : une ligne de facture fait référence à une entête de facture
  - D'intégrité fonctionnelle
    - Via des triggers
    - Exemple : pas de facture non payée pour insérer une nouvelle commande
- Permettent d'assurer la qualité des données
  - Des impacts sur la cinématique des programmes

Etape 3 : utiliser les possibilités de SQL

- Plus de types de données disponibles
  - XML, DATALINK, LOB, ...
  - Colonnes auto-générées
- Gestion naturelle des formats d'échange de données
  - XML, JSON
- Des fonctions avancées
  - CTE, requêtes récursives, tables temporelles …
- Des fonctions intégrées
  - Sur les données : calculs sur les dates, …
  - Autres : HTTP, …
- Des services IBM i
  - Simplifient l'administration
- Utilisables directement en RPG/COBOL !

Objectifs

- Déplacer une partie des règles de gestion métier dans la BD
  - Via des vues
    - Utilisables à la place des tables dans les instructions SQL
    - Permettent de masquer la structure et la complexité des données
    - Facilitent une réutilisation
      - Quel que soit le langage
  - Via des triggers
    - Déclenchés quelle que soit l'interface d'accès à la donnée
      - SQL, programme RPG, Java, .Net, QM, QRY400 …
    - Les triggers peuvent eux-mêmes être écrits en SQL ou RPG/COBOL
  - Via des procédures ou fonctions
    - Eux-mêmes également écrits en SQL ou RPG/COBOL
    - Les traitements sont au plus proche de la donnée
    - Utilisables ensuite dans tous les langages supportant le SQL
      - RPG/COBOL
      - Tous les langages capables de se connecter à la BD, c'est-à-dire tous les langages de gestion !

Objectifs

- Faciliter l’accès à la donnée
  - Par un standard technique (SQL)
  - Par une donnée de qualité (contraintes)
  - Par une donnée structurée (intégrité référentielle)
  - Par une donnée disponible (performance)

Nécessités

- Administration de la BD
  - Aspect performance
    - Surveillance des temps de réponse
    - Adaptation des requêtes
    - Indexation
  - Aspect structure de la donnée
    - Cohérence du modèle
  - Assistance aux développeurs
  - Prise en main des outils disponibles

Conclusion

Bonnes pratiques

- Structure
  - Se rapprocher de la 3ème FN
    - Clé primaire sur toutes les tables
    - Pas de redondance dans une table
- Requêtes SQL
  - Uniquement sur des tables ou des vues
  - Pas de LF
  - Pas de SELECT *
  - Soyez le plus explicite possible
- Langage ensembliste
  - Il est inutile de remplacer un CHAIN par un SELECT
  - Très efficace pour manipuler un ensemble de données

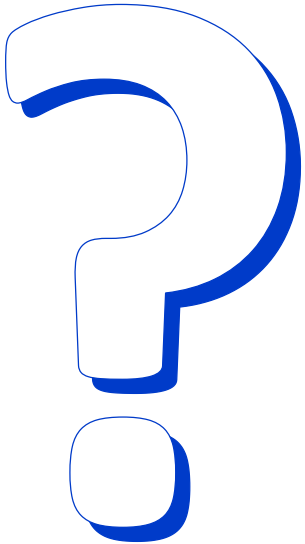
Bonnes pratiques

- Index / vue
  - À quel moment créer un index ?
  - À quel moment créer un index dérivé ?
  - À quel moment créer un index EVI ?
  - À quel moment créer une vue ?
- Le débat
  - Optimisation
    - Requête par requête ?
    - Globale ?
    - Dénormalisation ?
  - Clés techniques vs fonctionnelles ?
  - DBE / DBA ?

Références

- Formes normales
  - https://fr.wikipedia.org/wiki/Forme_normale_(bases_de_donn%C3%A9es_relationnelles)
- SQL indexes and native I/O – no contradiction (Birgitta Hauser)
  - https://www.ibm.com/developerworks/ibmi/library/i-sql-indexs-and-native-io/index.html
- SQL Reference
  - http://www-01.ibm.com/support/knowledgecenter/ssw_ibm_i_73/db2/rbafzpdf.pdf?lang=en
- Red book
  - http://www.redbooks.ibm.com/abstracts/sg248185.html?Open

Q/R



Merci de votre attention