Sybase Adaptive Server Enterprise 15

Prêt pour Sybase Adaptive Server Enterprise 15 ?

Novembre 2006

Documentation technique # 29

sqlpac

Introduction

- Cette présentation liste les fonctionnalités importantes de Sybase Adaptive Server Enterprise 15.0.
- L'optimiseur a été complètement "revampé" et refondu afin de gérer le partitionnement logique sémantique introduit dans Sybase ASE 15.0.
- D'autres nouvelles fonctionnalités majeures ont été implémentées : les curseurs déplaçables, les colonnes calculées, les indexes fonctions...

sqlpac

Plan

- Améliorations de l'exécution des requêtes (query processing)
- Le cache de requêtes et la paramétrisation littérale
- Nouveautés T-SQL
- Partitionnement sémantique
- Colonnes calculées et indexes fonctions
- Curseurs déplaçables (scrollable cursors)
- Stockage
- Système

sqlpac

Améliorations de l'exécution des requêtes 1/6

- ASE 15.0 introduit les jointures par hachage (hash joins) :
  - Performances améliorées avec des stratégies "index union" et "index intersection" pour les requêtes avec des précidats AND/OR.
  - Performances améliorées pour les requêtes avec des clauses GROUP BY et ORDER BY grâce au groupement et au tri en mémoire. Suppression de la matérialisation de tables de travail pour réaliser les opérations.
  - Traitements DISTINCT et EXISTS améliorés.
  - Traitements des aggrégats scalaires MIN/MAX améliorés avec des indexes utiles.
  - Sélectivité représentative basée sur les histogrammes de jointures calculées dynamiquement.

sqlpac

Améliorations de l'exécution des requêtes 2/6

- Optimization goals : selon les besoins, un but (goal) peut être appliqué dans les optimisations au niveau serveur, session, ou au niveau de la requête.

  - allrows_oltp : optimise les requêtes pour trouver le meilleur plan d'exécution en environnement OLTP (OnLine Transactional Processing).
  - allrows_dss : génère le plan optimal pour les requêtes complexes en environnement DSS (environnements décisionnels).
  - allrows_mix : génère des plans pour des environnements mixtes (OLTP et DSS).
  - fastfirstrow : génère des plans qui enverra les premières lignes du jeu de résultat le plus tôt possible.

sqlpac

Améliorations de l'exécution des requêtes 3/6

- Nombre de valeurs dans les listes IN illimité

```
select * from tab
where col1 in (1,2,3,…,16000)
```

- La limitation sur le nombre de valeurs dans les listes IN (IN-list) a été supprimée avec ASE 15.0.
- La liste est désormais seulement limitée par la taille du cache de procédures (cache procedure).

sqlpac

Améliorations de l'exécution des requêtes 4/6

- Suppression des limites sur le nombre de colonnes dans les aggrégats groupés.

```
select col1,col2,...col50,max(col51) * from tab
group by col1,col2,...col50
```

- ASE 15.0 exécutera avec succès la requête si la longueur col1...col50 plus le résultat du MAX est inférieur à 16K. La largeur maximale est augmentée à 16K, indépendamment de la taille de page du serveur.
- La limite des 31 colonnes dans une clause GROUP BY a été supprimée.

sqlpac

Améliorations de l'exécution des requêtes 5/6

- Utilisation des indexes lors de différences de types de données (mismatched data types)

```
select * from A,B where A.a=B.b

A.a est de type float
B.b est de type int avec un index
B est une grosse table
```

- ASE 15.0 garantit dans l'exécution de la requête l'utilisation de l'index sur la colonne B.b (ce qui n'était pas le cas dans les versions pré 15.0 à cause de la différence de types, un scan de table était choisi pour la table B)

sqlpac

Améliorations de l'exécution des requêtes 6/6

- Contrôle des optimisations des jointures (permutations)
- Avant Sybase 15.0, le contrôle des optimisations des jointures était réalisé avec le paramètre de session "set table count". Ce paramètre convient bien pour peu de jointures, mais une optimisation de requête pouvait durer 10 min pour 30 secondes d'exécution en augmentant le paramètre de session "set table count".
- Avec Sybase 15.0, l'optimisation des jointures peut être contrôlé et arrêté (timeout) après un certain pourcentage d'optimisation. Le plan le plus optimal sera déterminé au timeout. Ce pourcentage est gouverné par le paramètre serveur "optimization timeout limit".

sqlpac

Cache de requêtes : paramétrisation littérale

- La cache de requêtes (statement cache) est une nouveauté ASE 12.5.2 pour réduire les coûts de compilation des requêtes. Le cache de requêtes stocke les plans des requêtes pour éviter de parser à nouveau une requête identique.
- A partir de la version 15.0.1, la paramétrisation littérale finalise la puissance du cache de requêtes. Chaque valeur littérale est remplacée par une variable. Cette fonctionnalité offre une meilleure efficacité pour le partage des plans des requêtes.

```
select * from mytable where id=1
select * from mytable where id = @@@V0_INT
```

sqlpac

Nouveautés T-SQL 1/2

- A partir de la version 15.0.1, les nouvelles fonctions isdate() et isnumeric() sont introduites.
  - isnumeric(string) : si la chaîne est bien de type integer/float/money, la fonction retourne 1, sinon 0.
  - isdate(string) : si la chaîne est une date valide, la fonction retourne 1, sinon 0. La fonction isdate() prend en compte l'option de session "set dateformat".

sqlpac

Nouveautés T-SQL 2/2

- A partir de la version 15.0, la commande “select into existing table” est possible pour les tables utilisateur (seules les tables proxy étaient autorisées dans les versions 12.5):

```
select a,b,c into existing table t from mytable
```

- Comme il s’agit d’une opération faisant l’objet d’une journalisation minimale “minimally logged” :
  - La table doit avoir des indexes.
  - La table ne doit pas apparaître dans la clause FROM.

sqlpac

Partitionnement sémantique 1/9

- ASE 15 offre maintenant 4 méthodes de partitionnement des tables :
  - RANGE
  - LIST
  - HASH
  - ROUND ROBIN (implémenté avec ASE 11.0)

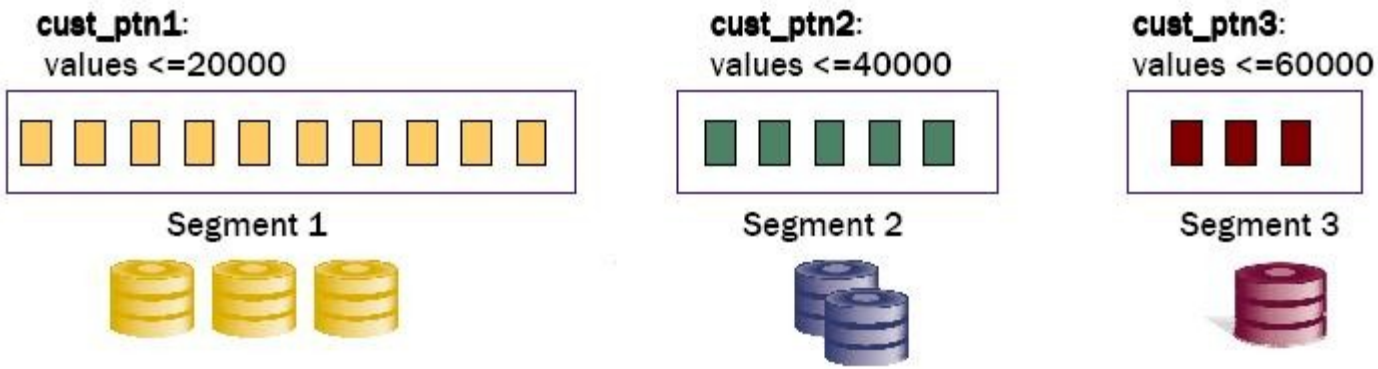
- Tout type d'applications bénéficie du partitionnement en n'agissant que sur des portions des tables :
  - Les applications DSS qui opèrent sur des partitions anciennes.
  - Les applications OLTP qui opèrent sur les partitions récentes (hot).

sqlpac

Partitionnement sémantique 2/9

- Partitionnement par intervalles (RANGE PARTITIONING)

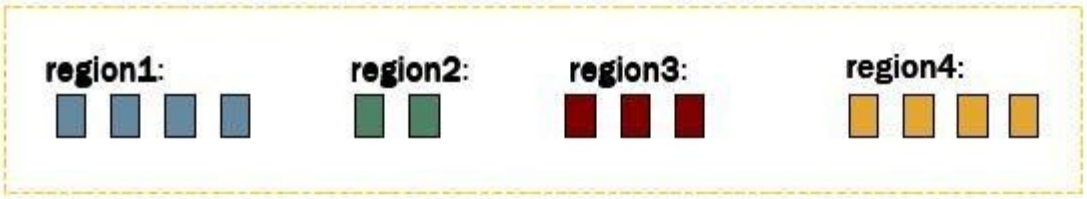
```
create table customer
( c_custkey     integer                 not null,
  c_name                varchar(20)     not null,
  c_address     varchar(40)     not null,
  other columns …
) partition by range (c_custkey)
   (cust_ptn1 values <= (20000) on segment1,
    cust_ptn2 values <= (40000) on segment2,
    cust_ptn3 values <= (60000) on segment3 )
```



Partitionnement sémantique 3/9

- Partitionnement par liste (LIST PARTITIONING)

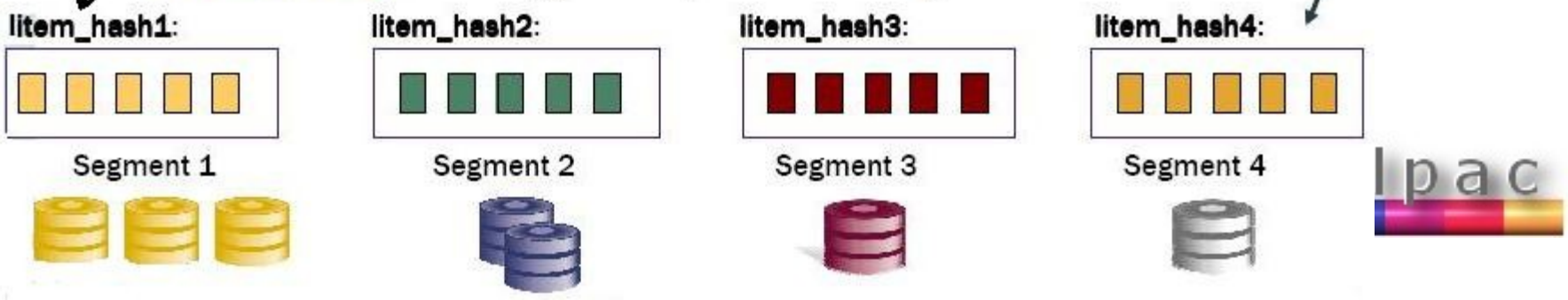
```
create table nation ( n_nationkey integer not null
                    , n_name     char(25)     not null
                    , n_regionkey varchar(30)       not null
                    , n_comment          varchar(152)not null
) on segment 1
   partition by list  (n_regionkey)
   ( region1 values ('Americas'),
      region2 values ('Asia'),
      region3 values ('Europe'),
      region4 values ('Australia', 'Other') )
```



Partitionnement sémantique 4/9

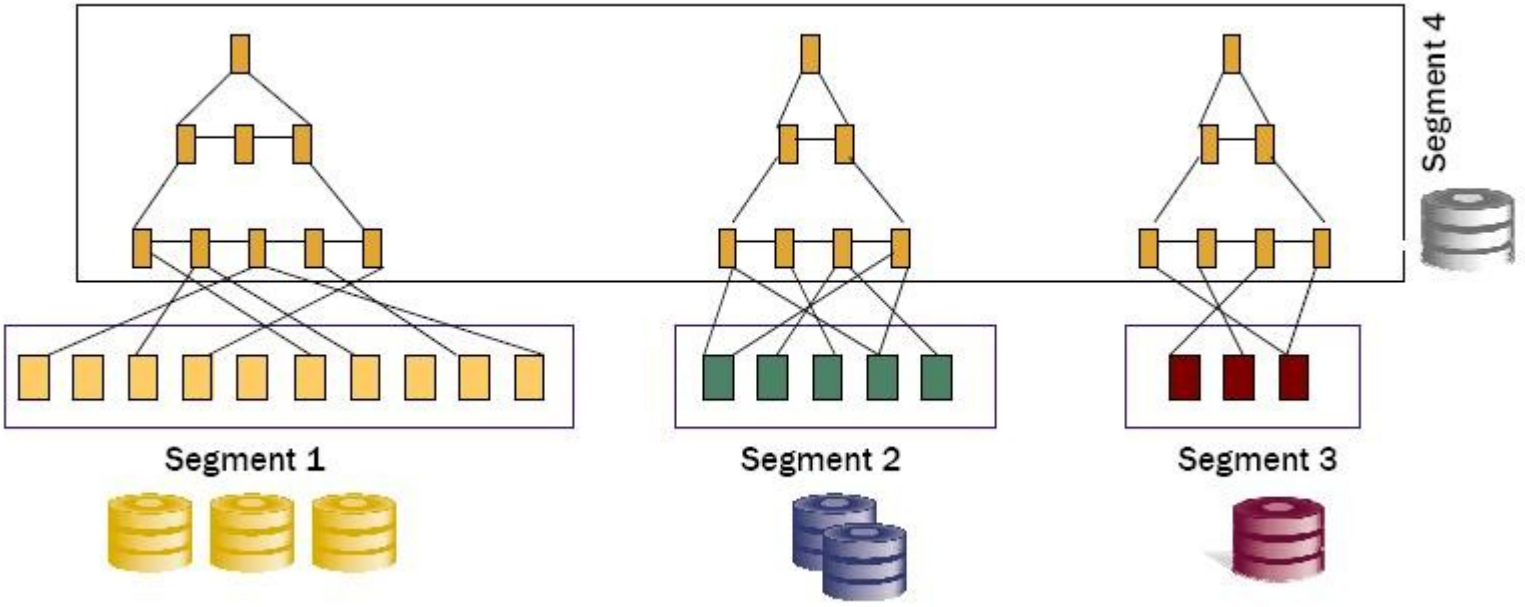
- Partitionnement par hachage (HASH PARTITIONING)

```
create table lineitem ( l_orderkey integernot null
                      , l_partkey   integernot null
                      , l_suppkey   integernot null
                      , l_linenumberintegernot null
                      , l_quantity  double not null
                      , l_extendedprice    double not null
                      , other columns …
) partition by hash  (l_orderkey, l_linenumber)
  (litem_hash1 on segment1,
   litem_hash2 on segment2,
   litem_hash3 on segment3,
   litem_hash4 on segment4 )
```



Partitionnement sémantique 5/9

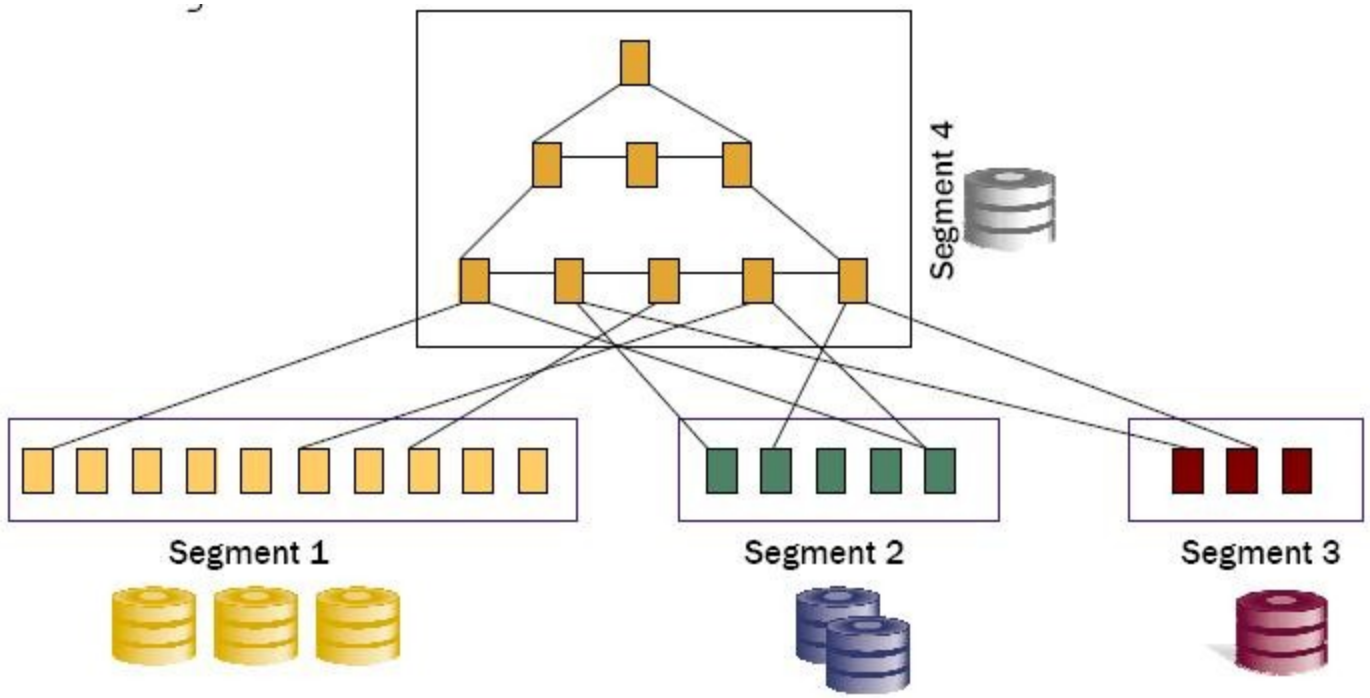
- Les indexes nonclustered sur les tables partitionnées peuvent être définis comme locaux ou globaux.
- Indexes nonclustered locaux



sqlpac

Partitionnement sémantique 6/9

- Index nonclustered global



sqlpac

Partitionnement sémantique 7/9

- Les indexes clustered (uniques ou non) peuvent uniquement être définis comme locaux pour les tables partitionnées en mode RANGE, LIST ou HASH.
- Les indexes clustered pour les tables partitionnées en mode aléatoire (round robin) peuvent être définis comme globaux ou locaux, sauf si la table a plus de 255 partitions, dans ce cas l'index clustered peut uniquement être défini comme local.

Partitionnement sémantique 8/9

- La maintenance est facilitée avec les partitions :
  - Commandes ADD, ALTER, DROP sur une ou plusieurs partitions (drop est disponible depuis la version 15.0.1).
  - UPDATE STATISTICS sur une ou plusieurs partitions.
  - REORG REBUILD sur une ou plusieurs partitions.
  - Opérations DBCC sur une ou plusieurs partitions.
  - Opérations TRUNCATE sur une ou plusieurs partitions.
  - Bcp (bulk copy binary) in/out prend en compte les clauses de partitionnement

```
bcp [[db_name.]owner.]table_name[:slice_num]
[partition pname] {in |out} [filename]…
```

- Les partitions diminuent grandement les fenêtres de maintenance.

sqlpac

Partitionnement sémantique 9/9

- Elimination de partitions dans les plans d'exécution:

```
QUERY PLAN FOR STATEMENT 1 (at line 1).
2 operator(s) under root
The type of query is SELECT.
ROOT:EMIT Operator
|SCALAR AGGREGATE Operator
| Evaluate Ungrouped COUNT AGGREGATE
|
| |SCAN Operator
| | FROM TABLE
| | telco_facts_ptn
| | [ Eliminated Partitions : 2 3 4 5 6 7 8 ]
| | Index : primary_key_idx
| | Forward Scan.
| | Positioning by key.
| | Index contains all needed columns. Base table will not be read.
| | Keys are:
| | month_key ASC
| | Using I/O Size 4 Kbytes for index leaf pages.
| | With LRU Buffer Replacement Strategy for index leaf pages.
```

sqlpac

Colonnes calculées 1/4

- Une colonne calculée (computed column) est définie par une expression impliquant des colonnes de la même ligne, des fonctions, des opérateurs arithmétiques...
- Déterministes ou non déterministes (deterministic vs non deterministic)
  - Les expressions et fonctions déterministes retournent toujours le même résultat si elles sont évaluées avec les mêmes valeurs en entrée.

```
create table t ( a int, b as a*10)
```

  - Les expressions et fonctions non déterministes peuvent retourner des valeurs différentes même si elles sont évaluées avec les mêmes valeurs en entrée.

```
create table t ( a int, b as rand( )*a)
```

sqlpac

Colonnes calculées 2/4

- Colonne matérialisée – non matérialisée (virtuelle)
- Les colonnes calculées peuvent être matérialisées ou virtuelles (non matérialisées)
  - Colonnes calculées matérialisées : l'expression est évaluée et stockée, l'expression est réévaluée seulement si une des colonnes figurant dans l'expression est modifiée.
  - Colonnes calculées virtuelles : l'expression n'est pas stockée et réévaluée à chaque accès.

sqlpac

Colonnes calculées 3/4

```
create table customer (
first_name varchar(20) not null,
last_name varchar(40) not null,
birth_date date not null,
phone_number varchar(10) not null,

-- Exemple de colonne matérialisée déterministe
customer_id as soundex(first_name) + soundex(last_name)
+ convert(varchar(10),phone_number)
materialized not null,

-- Exemple de colonne non déterministe virtuelle
age_yrs as datediff(yy,birth_date,getdate()),

primary key (customer_id)
)
```

sqlpac

Colonnes calculées 4/4

- Colonnes calculées indexées
  - Les colonnes calculées indexées doivent être matérialisées mais ne sont pas forcément déterministes (même si dans l'usage, cela paraît guère utile).

```
alter table orders
add up11 as unit*price*1.1 materialized
go
create index up11_ix on orders (up11)
Go

select unit*price*1.1 from orders
where up11 > 200

=> utilise l'index up11_ix.
```

sqlpac

Indexes fonctions 1/2

- Les indexes fonctions sont des indexes qui contiennent une ou plusieurs expressions dans les clés de l'index.

```
create index total on orders (unit*price*1.1)
```

- Les indexes fonctions sont toujours matérialisés et doivent être non déterministes.
- Les indexes fonctions fournissent les mêmes avantages qu'une colonne calculée, cependant aucun changement du schéma de la table en ajoutant une colonne n'est nécessaire avec ces derniers.
  - Les expressions de l'index clé sont pré-évaluées.
  - Les indexes fonctions n'ont pas besoin d'être une nouvelle fois évalués lorsqu'ils sont accédés.

sqlpac

Indexes fonctions 2/2

- Les clés de l'index fonction sont prises en compte par l'optimiseur pour les jointures.

```
create index total on orders (unit*price*1.1)
go
select unit*price*1.1 from orders
where unit*price*1.1 > 200
```

- unit*price*1.1 est reconnu comme une clé d'index, l'index est utilisé.
- unit*price*1.1 n'est pas réévaluée, la valeur pré-évaluée est utilisée.

sqlpac

Curseurs déplaçables 1/3

- Aucun positionnement n'était possible avec les curseurs pré 15.0 (uniquement unidirectionnel)
- Avec ASE 15.0, les curseurs déplaçables (scrollable cursors) sont introduits :
  - Déplacement directionnel (backwards, forwards) : à partir du début, à partir de la fin.
  - Positionnement explicite : première ligne, dernière ligne, ligne spécifique.
- La sensibilité est également introduite avec les curseurs ASE 15.0 (sensibilité aux changements réalisés dans la table) : modes insensitive, sensitive et semi-sensitive.

sqlpac

Curseurs déplaçables 2/3

- Positionnement explicite avec les curseurs 15.0

```
fetch last [from] <cursor_name>

fetch absolute 500 [from] <cursor_name>

fetch {next | prior | first | last | absolute # | relative #}
[from] <cursor_name>
```

sqlpac

Curseurs déplaçables 3/3

- La définition de la sensibilité aux modifications est autorisée avec les curseurs déplaçables ASE 15.

```
declare C3 insensitive scroll cursor
for select name from emp_tab
```

- La sensibilité d'un curseur détermine si les modifications de données sont visibles ou non pour le curseur :
  - Insensitive : les lignes sont copiées dans une table de travail ainsi les modifications ne sont pas visibles.
  - Semi sensitive : les lignes sont copiées dans une table de travail au fur et à mesure que les lignes sont extraites, les modifications sont visibles pour les lignes qui n'ont pas encore été récupérées.

sqlpac

Stockage

- Avec les versions pré 15.0, les limitations étaient :
  - 256 devices par base de données max.
  - Chaque device était limité à 32 Gb.
- ASE 15.0 étend les limites du stockage système :
  - 2^31 devices par base de données.
  - Chaque device est désormais limité à 4 Tb.

Système 1/2

- Row locked catalogs (RLC) : les tables systèmes dans les bases de données sont désormais en verrouillage ligne (datarows locking scheme)
  - Réduit la contention pour les opérations DDL.
  - Les verrous mortels (Deadlocks) dus à la contention sur le catalogue système sont éliminés.
  - La contention dans tempdb est réduite de manière significative.

sqlpac

Système 2/2

- Nouveaux types de données
  - Int, smallint et tinyint non signés (unsigned).
  - Bigint (8-byte integer, signé ou non signé).

- Identifiants longs
  - Suppression de la limite à 30 caractères sur les noms d'objets (255 ou 253 avec les quotes).
  - Les colonnes hostname, program_name et hostprocesses de la table sysprocesses sont maintenant en varchar(30) (contre 10, 16 & 8 dans les versions 12.5).
  - Les logins, users et noms de curseurs sont toujours limités à 30 caractères.

sqlpac

Conclusion

- Quelques conclusions à propos des migrations vers ASE 15.0

  - L'optimiseur a subi une refonte majeure : une revue complète des plans d'exécution pour traquer toutes les régressions de performance doit être menée.
  - Un revamping a été introduit pour les jointures et les opérations GROUP BY et ORDER BY. Une étude poussée doit être menée pour déterminer la mémoire nécessaire dans le cache de procédure pour les opérations de hachage.
  - La migration sera focalisée sur les opérations GROUP BY car les tris ne sont plus implicites lorsque l'opération de groupement est réalisée en mémoire. Le traceflag 450 a été introduit avec la version 15.0 ESD#2 mais ce n'est pas suffisant.

sqlpac