

1. DIMENSIONS

Les dimensions sont des structures du dictionnaire de données qui définissent des hiérarchies basées sur des colonnes existantes dans la ou les tables sur lesquelles elles sont définies.

Les dimensions sont optionnelles mais hautement recommandées du fait que:

- Elles assurent la réécriture des requêtes (query rewrites) sans utiliser des contraintes qui alourdissent l'environnement DW
- Elles constituent des informations sur les hiérarchies
- Elles peuvent être utilisées par les outils OLAP

Les dimensions décrivent les entités d'une manière hiérarchique: produits, départements, etc. Une dimension peut être constituée de une ou plusieurs hiérarchies. Dans notre exemple, la dimension temps consiste en une hiérarchie calendrier !

Une hiérarchie est constituée de plusieurs niveaux. Chaque niveau est l'enfant d'un seul parent présent au niveau immédiatement supérieur dans la hiérarchie. Un niveau parent représente une agrégation du niveau enfant. Dans l'exemple, le calendrier est constitué des niveaux années, semestre, mois, jour.

Traverser la hiérarchie du bas vers le haut signifie une agrégation des informations du niveau immédiatement inférieur. En anglais : rollup data !

1.1. LEVEL KEYS ET ATTRIBUTES

Une 'level key' identifie un niveau d'une hiérarchie. Il peut y avoir d'autres attributs qui identifient un niveau, identifiables par la 'level key'. Un niveau peut être renommé avec un attribut qui devient un alias.

Des relations hiérarchiques peuvent être définies entre deux colonnes d'une table de dimensions ou entre deux colonnes de tables différentes si celles-ci font partie d'un schéma normalisé (de type flocon de neige).

Un nouveau type objet, la dimension est ajouté aux types existants. La relation parent-enfant dans une dimension est plus contraignante qu'une relation d'intégrité référentielle du fait qu'elle exige que la clé enfant ne soit pas nulle.

Le créateur de la dimension doit assurer que les colonnes dans chaque niveau de

la hiérarchie sont non nulles et que l'intégrité de la dimension est préservée. Cette démarche peut être assurée avec des contraintes référentielles dans le cas d'une dimension normalisée.

Les clés et attributs pour un niveau donné doivent être basées sur des colonnes de la même table.

L'exemple suivant montre une seule hiérarchie dans la dimension temps, mais il est possible d'avoir plusieurs hiérarchies pour une même dimension.

Le privilège CREATE DIMENSION est nécessaire pour créer une dimension dans un schéma, dimension basée sur les tables du même schéma. Le privilège CREATE ANY DIMENSION permet de créer des dimensions dans n'importe quel schéma.

1.2. DEFINIR UNE DIMENSION BASEE SUR PLUSIEURS TABLES.

```
CREATE DIMENSION GEOGRAPHY_DIM
LEVEL country IS region.country
LEVEL region IS region.region_id
LEVEL state IS state.state_code
LEVEL city IS (city.state_code, city.city_code)
HIERARCHY geography_rollup (city
CHILD OF state
CHILD OF region
CHILD OF country
JOIN KEY city.state_code REFERENCES state
JOIN KEY state.region_id REFERENCES region)
ATTRIBUTE city DETERMINES (city_name, office_address, population)
ATTRIBUTE state DETERMINES state_name
ATTRIBUTE region DETERMINES region_name ;
```

Dans le cas d'une dimension basée sur colonnes de plusieurs tables, la clause JOIN KEY est utilisée pour mettre en relation la colonne enfant avec le niveau père. Dans le cas suivant, CITY_CODE n'est pas unique et nous l'associons avec state pour le rendre unique.

1.3. REECRITURES DES REQUETES ET DIMENSIONS

La requête suivante utilise un rollup dans la dimension TIME_DIM:

```
SELECT t.year, p.brand , c.city_name, SUM(s.amt)
FROM sales s, city c, timetab t, product p
WHERE s.sdate = t.sdate
AND s.city_name = c.city_name AND s.state_code =
c.state_code
```

```
AND s.prod_code = p.prod_code
GROUP BY t.year, p.brand, c.city_name;
SELECT v.year, s.brand, s.city_name, SUM(s.tot_sales)
FROM sales_sumry s,
(SELECT distinct t.month, t.year FROM timetab t) v
WHERE s.month = v.month
GROUP BY v.year, s.brand, s.city_name;
```

L'exemple montre une réécriture rendue possible par l'existence de la dimension TIME_DIM. La relation entre mois et année est déduite grâce à la définition de la dimension et est utilisée pour faire un rollup du sommaire des ventes pour obtenir les ventes annuelles.

1.4. ACTIVER ET CONTROLER LES REECRITURES

Les réécritures sont effectuées automatiquement et sont transparentes pour l'application. Les paramètres suivants contrôlent la réécriture en mode COST.

- **OPTIMIZER_MODE**: Les réécritures des sommaires sont disponibles uniquement en mode COST. Le mode RULE inhibe leur comportement.
- **MVIEW_REWRITE_ENABLED**: Ce paramètre peut être mis à FALSE pour supprimer la réécriture. Le paramètre est modifiable avec alter session ou alter system.
- **REWRITE_INTEGRITY**: Ce paramètre peut être mis à:
 - **ENFORCE** pour permettre des rewrites uniquement si Oracle peut garantir la consistance. Uniquement des vues matérialisées et contraintes activées validate seront utilisées pour la réécriture.
 - **NOENFORCE** pour permettre des réécritures basées sur des relations déclarées mais pas forcément validées.
 - **USE_STALE** pour permettre des réécritures basées sur des vues matérialisées non mises à jour ou des relations déclarées.

Les query rewrites sont similaires aux plan d'exécutions, l'utilisation des sommaires ne nécessite pas des privilèges spéciaux. Les utilisateurs qui ont accès aux tables peuvent utiliser les sommaires.

Un nouveau hint, REWRITE, peut être employé pour choisir parmi les sommaires, un autre, NOREWRITE, est disponible pour supprimer les réécritures dans la requête elle-même.

1.5. DBMS_OLAP

Le package DBMS_OLAP contient des fonctions et procédures utiles pour la gestion des sommaires. Les fonctions consultatives (advisory) des sommaires présentes dans le package tirent profit de deux sources d'informations:

- Les statistiques de charges, fournies par Oracle Trace (Enterprise Manager) ou d'autres produits. Un nouvel événement est disponible, MATERIALIZED VIEW USAGE
- Dictionnaire de données: Les informations puisées concernent les description des sommaires et dimensions.

D'autres informations ainsi obtenus:

- Utilisation des sommaires: nombre de réécritures qui utilisent un sommaire, le volume occupé par un sommaire, un rapport COST/bénéfice pour chaque sommaire.
- Recommandations pour les sommaires : création, maintien et suppression des sommaires.
- Besoins en volume pour la création des sommaires.

1.5.1.EVALUATION DES SOMMAIRES EXISTANTS

Les procédures EVALUATE_UTILIZATION et EVALUATE_UTILIZATION_W sont utilisées pour évaluer les sommaires existantes.

1. EVALUATE_UTILIZATION utilise des statistiques hypothétiques concernant les tables détail et leur colonnes, estimant leur utilisation
2. EVALUATE_UTILIZATION_W utilise les statistiques collectées par Oracle Trace dans WORK\$_IDEAL_MVIEW et WORK\$_MVIEW_USAGE pour générer les résultats.

La vue MVIEW\$_EVALUATIONS contient pour les deux procédures les éléments suivants:

- Nom du sommaire
- Volume de stockage (en octets)
- Fréquence d'utilisation
- Bénéfice cumulatif
- Rapport bénéfice/COST

1.5.2.RECOMMANDATIONS

DBMS_OLAP.RECOMMEND_MV_W('SALES',102400000,NULL,80);

Les étapes suivantes sont nécessaires pour obtenir des conseils de la part du conseiller des sommaires

- Analyse des tables dans le schéma concernés
- Rassembler des informations sur la charge de travail avec Oracle Trace.
- Peupler MVIEW\$_RECOMMENDATIONS avec RECOMMEND_MV_W qui accepte les paramètres suivants :
 - Liste de tables à évaluer.
 - Volume maximum (en octets) pour stocker les vues matérialisées.
 - Liste de vues matérialisées à conserver.
 - Pourcentage de rétention: si le volume utilisé par une vue matérialisée est inférieur (en pourcentage) à ce seuil, elle sera retenue.

1.5.3.VISUALISER LES RECOMMANDATIONS

La vue MVIEW\$_RECOMMENDATIONS peut être utilisée pour vérifier les recommandations du conseiller des sommaires :

```
SELECT recommended_action, mview_name,
group_by_columns, measures_list
FROM mview$_recommendations;
RECOMM MVIEW_NAME GROUP_BY_COL MEASURES_L
-----
RETAIN SALES_SUMRY
DROP COMM_SUMRY
RETAIN BRAND_SUMRY
...
CREATE EMPNO, PROD_CODE SUM(AMT)
```

1.5.4.ESTIMER LES BESOINS EN VOLUMETRIE

La procédure ESTIMATE_SIZE peut être utilisée pour estimer le volume nécessaire pour un sommaire ou pour une vue matérialisée. La procédure accepte un nom et une requête utilisée dans la création d'une vue matérialisée et estime le nombre de lignes et le volume estimé de la vue matérialisée :

DBMS_OLAP.ESTIMATE_SIZE('TRIAL1','SELECT ...', NROWS, NBYTES);

Ou les paramètres sont:

- La requête à analyser
- Statement id for EXPLAIN

- Nombre estimée de lignes
- Volumétrie estimée (en octets)

1.5.5.BENEFICE DE L'UTILISATION DES SOMMAIRES

- Performance des requêtes
- Utilisation transparente
- Administration diminué (par le sommaire)
- Fonctions de conseil
- Registration of existing summary tables
- Divers mécanismes de rafraîchissement
- Supportés par SQL *LOADER et export/import
- Support pour les index