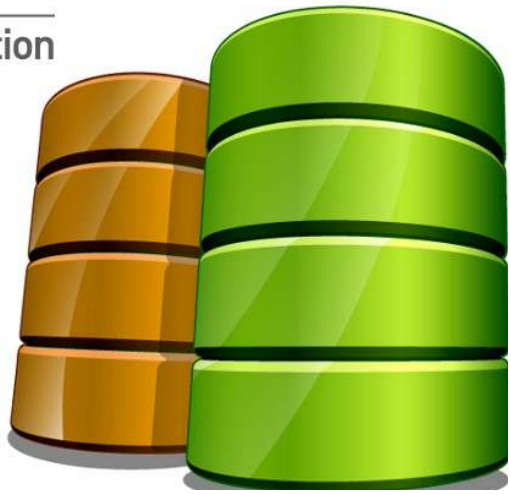
Data warehouse

Data warehouse et outils décisionnels

AI07

stph.scenari-community.org



Stéphane Crozat

15 septembre 2016

Table des matières

Objectifs

- Connaître les principaux domaines d'application des data warehouses
- Connaître le paradigme du décisionnel (et son articulation avec le paradigme transactionnel)
- Connaître les principes, les étapes et les méthodes de la modélisation dimensionnelle
- Savoir faire une étude de besoins
- Savoir faire une étude des données existantes
- Savoir faire un modèle dimensionnel en étoile
- Savoir implémenter un data warehouse avec un SGBD relationnel
- Savoir implémenter un processus ETL vers un data warehouse
- Savoir interroger en SQL un data warehouse en vue d'applications décisionnelles
- Savoir faire un modèle dimensionnel en étoile et en flocon
- Savoir gérer les dimensions dégénérées et attributs de documentation, de segmentation et d'agrégation
- Savoir implémenter un data mart pour l'analyse de tickets de caisse

I

Introduction au domaine du décisionnel et aux data warahouses

A. Le décisionnel

Objectifs

Connaître le paradigme du décisionnel (et son articulation avec le paradigme transactionnel)
Connaître les principaux domaines d'application des data warehouses

1. Décisionnel

Définition

Le système d'information décisionnel est un ensemble de données organisées de façon spécifiques, facilement accessibles et appropriées à la prise de décision [...].

La finalité d'un système décisionnel est le pilotage d'entreprise.

Les systèmes de gestion sont dédiés ***aux métiers*** *de l'entreprise [...].*

Les systèmes décisionnels sont dédiés ***au management*** *de l'entreprise [...].*

(Goglin, 2001, pp21-22)

Synonymes : informatique décisionnelle, *business intelligence*, BI

2. Enjeux du décisionnel

La prise de décisions stratégiques dans une organisation nécessite le recours et le croisement de multiples informations qui concernent tous les départements : production, RH, DAF, achats, ventes, marketing, service après-vente, maintenance, R&D...

Or ces données sont généralement :

- éparpillées au sein des départements et non connectées entre elles
- hétérogènes dans leurs formats techniques et leurs organisations structurelles, voire leurs sémantiques

- implémentées pour l'action (par construction) et non pour l'analyse
- volatiles, au sens où leur mise à jour peut conduire à oublier des informations obsolètes

Exemple

Un catalogue de produits sera conçu pour permettre de trouver facilement un produit en fonction de caractéristiques précises, de faire des mises à jour rapides et fiables, de gérer des stocks...

Mais un système décisionnel souhaitera :

- connaître l'organisation des produits selon certaines caractéristiques et regroupements qui ne sont pas forcément premiers dans la gestion quotidienne ;
- croiser le catalogue avec les ventes...

Fondamental

L'enjeu des systèmes décisionnels est de donner accès aux données existantes dans l'organisation, sous une forme intégrée, afin de faciliter leur interrogation croisée et massive.

Complément : Voir aussi

Différence entre un DW et un système transactionnel

3. Exploitation des données

Les données agrégées dans un système décisionnel servent à trois grandes catégories d'usage :

- La production de rapport récurrents (*reporting*)
- L'exploration manuelle
- L'analyse de données (descriptive ou prédictive)

Définition : Reporting

Le principe du *reporting* est d'agréger et de synthétiser des données nombreuses et complexes sous forme d'indicateurs, de tableaux, de graphiques permettant d'en avoir une appréhension globale et simplifiée.

Le *reporting* s'appuie principalement sur les agrégats (GROUP BY en SQL par exemple) afin de faire apparaître des comptages, sommes ou moyennes en fonction de critères d'analyses.

Le *reporting* est généralement récurrent, le même rapport sera produit à intervalles réguliers pour contrôler les variations des indicateurs.

Définition : Exploration manuelle

Une autre exploitation de données en contexte décisionnel consiste à pouvoir explorer les données de façon peu dirigée (heuristique) afin de trouver des réponses à des questions que l'on ne s'est pas posées (sérendipité). L'idée générale est plutôt que les réponses aux premières questions que l'on se posent conduiront à se poser de nouvelles questions.

L'exploration de données s'appuie sur des outils permettant de manipulation (IHM) et de visualiser (infovis) les données selon des requêtes dynamiquement produites par des utilisateurs experts du domaine.

Définition : Analyse de données

L'analyse de données est une branche de la statistique qui permet de mettre en évidence des tendances des données ou corrélations entre les données non évidentes a priori.

- Dans le cas de l'analyse descriptive, ll s'agit de rechercher une information statistique "cachée" que l'on ne connaît pas a priori.
- L'approche prédictive consiste à réaliser un modèle statistique des corrélations entre les données à partir d'échantillons d'apprentissage, puis à appliquer le modèle à des données nouvelles pour prédire leur comportement, avec des raisonnements du type "si ... alors" ; ou pour classifier des données (tel objet caractérisé par telles données appartient-il à telle classe ?). Les résultats sont généralement qualifiés par une probabilité d'occurrence.

4. Éthique et limites des systèmes décisionnels

Rationalisation excessive et processus complexes

Les systèmes décisionnels produisent des indicateurs ou s'appuient sur des modèles dont l'objectif est de simplifier la réalité pour aider à la prise de décision.

Mais la décision doit bien réintégrer des évaluations humaines qui la replacent dans sa réalité, qui est restée complexe.

- Le modèle ou l'indicateur n'est pas la réalité, s'en est une représentation.
- La décision ne s'applique pas à une représentation, mais à la réalité.

Sélectivité des données et organisations humaines

Les systèmes décisionnels s'appuient sur les données que l'on est en mesure de produire, mais ces données ne peuvent pas intégrer toutes les dimensions d'une organisation et de son environnement, en particulier les dimensions humaines.

Or ces dimensions cachées au système décisionnel déterminent de nombreux fonctionnements de l'organisation, et doivent continuer d'être prises en compte.

L´interprétation est humaine

Un système informatique produit des indicateurs qui nécessitent des interprétations humaines, expertes dans le cas du décisionnel. Un système informatique ne produit pas des directives qu'une organisation humaine doit suivre !

L´erreur est informatique

Les résultats produits par les systèmes décisionnels sont le résultat de conceptions informatiques et mathématiques complexes, qui peuvent receler des erreurs ou des raccourcis, par ailleurs les résultats sont souvent statistiques, donc non déterministes.

La possibilité d'une erreur ou d'une approximation inadaptée devra toujours être prise en compte dans les décisions.

CNIL

Le fait de constituer des fichiers informatisés relatifs des personnes doit généralement faire l'objet d'une déclaration à la CNIL et nécessite le respect de certaines règles comme le droit de rectification et de radiation.

5. Architecture d'un système décisionnel

Tout système décisionnel est architecturé globalement de la même façon :

- En amont un accès au système transactionnel en lecture seule
- Un DW✯ fusionnant les données requises
- Un ETL✯ permettant d'alimenter le DW à partir des données existantes
- Des applications d'exploitation de *reporting*, exploration et/ou prédiction
- D'éventuels DM✯ permettant de simplifier le DW en vue de certaines applications

Architecture générale d´un systèmes décisionnel

Fondamental : Principe de fonctionnement

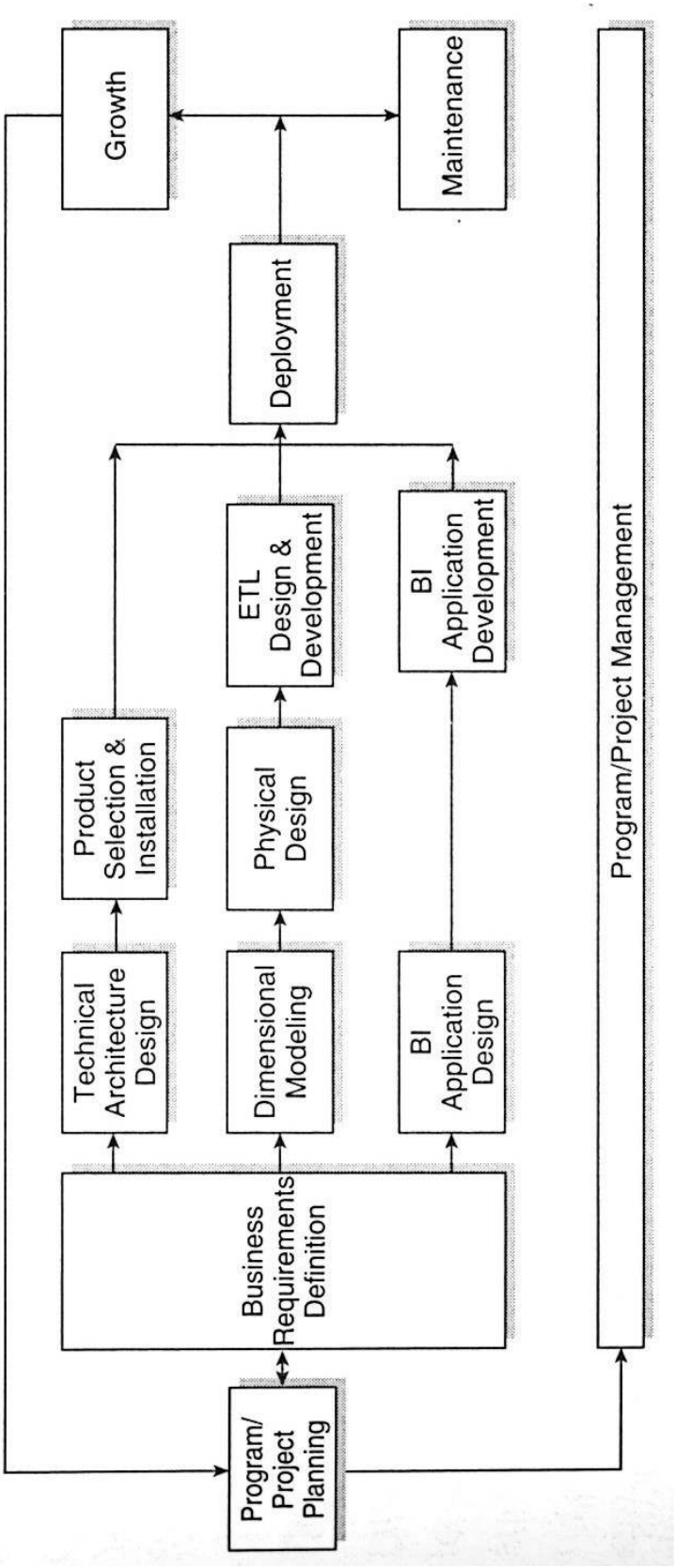
Le but du système est globalement d'être capable de présenter des tableaux de données (fichiers plats) en intrants des applications d'exploitation.

6. Conception d'un système décisionnel

Un projet de système décisionnel se structure selon quatre grands axes :

1. Étude des besoins et de l'existant
   - Étude des besoins utilisateurs
   - Étude des données existantes
2. Modélisation et conception
   - Modélisation dimensionnelle
   - Architecture technique
   - Spécification des outils d'exploitation
3. Implémentation du data warehouse
   - Implémentation du DW✯ et des DM✯
   - Mise en place de l'ETL✯
4. Implémentation des outils d'exploitation
   - Implémentation des outils de *reporting*
   - Implémentation des outils d'exploration
   - Implémentation des outils de prédiction

Complément



Lifecycle approach to DW/BI (Kimball, 2008, p3)

Complément

(Kimball, 2008) [(Kimball et al., 2008)]

7. Quelques exemples d'application

- Analyse du comportement de consommateurs ou de citoyens, en fonction de leurs caractéristiques (sexe, age...), de critères socio-économiques (profession...), géographiques...
- Analyse de ventes en fonction de l'implantation géographique de magasins (densité, caractéristiques des régions...), de l'organisation de magasins (rayonnage, marketing, RH...)
- Analyse des structures de paniers (quel produit est vendu en même temps que quel autre à quelles conditions ?)
- Prédiction de ventes en fonctions de données conjoncturelles, gestion des stocks, des approvisionnements
- Contrôle qualité et analyse de défaut des chaînes de production en fonction des centres de production, des organisations, des fournisseurs...
- ...

B. Le data warehouse

Objectifs

Comprendre ce qu'est et à quoi sert un data warehouse.
Comprendre les différences entre un data warehouse et une base de données transactionnelle.

1. Data warehousing

Définition : Définition historique de Inmon

A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management´s decisions. The data warehouse contains granular corporate data.

(Inmon, 2002, p31)

Définition

Un data warehouse (DW★) est une base de données construite par copie et réorganisation de multiples sources (dont principalement le système transactionnel de l'entreprise), afin de servir de source de données à des applications décisionnelles :

- il agrège de nombreuses données de l'entreprise (**intégration**) ;
- il mémorise les données dans le temps (**historisation**) ;
- il les organise pour faciliter les requêtes de prise de décision (**optimisation**).

(Goglin, 2001, p27) [(Goglin, 2001)]

Synonymes : entrepôt de données, base de données décisionnelle

Fondamental

L'objectif du data warehouse est de permettre des requêtes sur de grands ensembles des données, la plupart du temps sous forme d'agrégats (`GROUP BY`) afin d'en obtenir une vision synthétique (propre à la prise de décision).

Remarque

Le data warehouse dédié au décisionnel est séparé du système transactionnel dédié à la gestion quotidienne.

Complément : Voir aussi

Data warehouse et data mart

2. Différence entre un DW et un système transactionnel

BD transactionnelle

Une base données classique est destinée à assumer des **transactions** en temps réel :

- Ajout, mise à jour suppression de données
- Questions sur des données identifiées ou questions statistiques

Datawarehouse

Un DW★ est uniquement destiné à l'exécution de **questions statistiques** sur des données statiques (ou faiblement dynamiques).

PRIMITIVE DATA/OPERATIONAL DATA	DERIVED DATA/DSS DATA
• application oriented	• subject oriented
• detailed	• summarized, otherwise refined
• accurate, as of the moment of access	• represents values over time, snapshots
• serves the clerical community	• serves the managerial community
• can be updated	• is not updated
• run repetitively	• run heuristically
• requirements for processing understood a priori	• requirements for processing not understood *a priori*
• compatible with the SDLC	• completely different life cycle
• performance sensitive	• performance relaxed
• accessed a unit at a time	• accessed a set at a time
• transaction driven	• analysis driven
• control of update a major concern in terms of ownership	• control of update no issue
• high availability	• relaxed availability
• managed in its entirety	• managed by subsets
• nonredundancy	• redundancy is a fact of life
• static structure; variable contents	• flexible structure
• small amount of data used in a process	• large amount of data used in a process
• supports day-to-day operations	• supports managerial needs
• high probability of access	• low, modest probability of access

Un changement d'approche, extrait de (Inmon, 2002, p15)

3. Implémentation du DW avec un SGBDR

Fondamental

Les deux problématiques fondamentales des DW★ sont l'**optimisation** et la **simplification** : comment rester **performant** et **lisible** avec de très gros volumes de données et des requêtes portant sur de nombreuses tables (impliquant beaucoup de jointures) ?

On utilise massivement :

- **Les vues concrètes : Un data warehouse procède par copie depuis le ou les systèmes transactionnels**
- **La dénormalisation : Un data warehouse est hautement redondant**

Fondamental

Le caractère **statique** du data warehouse efface les inconvénients de ces techniques lorsqu'elles sont mobilisées dans des systèmes transactionnels.

Rappel

Dénormalisation

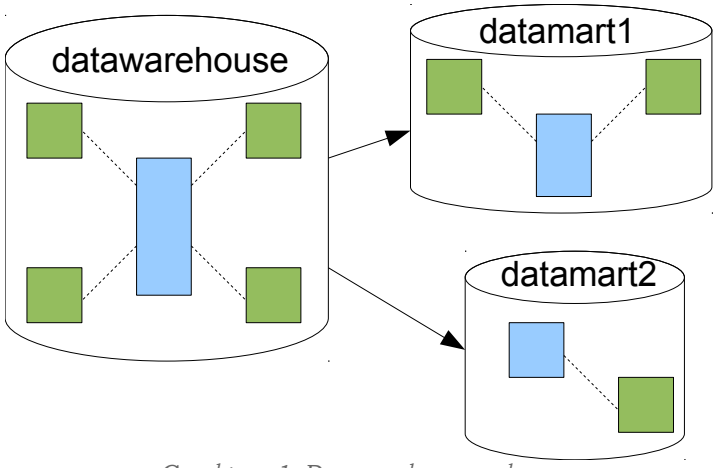
Vues concrètes

4. Data warehouse et data mart

Un data warehouse et un *data mart* se distinguent par le spectre qu'il recouvre :

- Le *data warehouse* recouvre l'ensemble des données et problématiques d'analyse visées par l'entreprise.
- Le *data mart* recouvre une partie des données et problématiques liées à un métier ou un sujet d'analyse en particulier

Un *data mart* est fréquemment un sous-ensemble du *data warehouse* de l'entreprise, obtenu par extraction et agrégation des données de celui-ci.



Graphique 1 Datawarehouse et datamarts

Pourquoi des data marts ?

Les *data marts* sont destinés à pré-agréger des données disponibles de façon plus détaillée dans les *data warehouse*, afin à traiter plus facilement certaines questions spécifiques, critiques, etc.

Exemple : Ticket de caisse

Si un *data warehouse* enregistre un ensemble de ventes d'articles avec un grain très fin, un *data mart* peut faciliter une analyse dite de **ticket de caisse** (co-occurrence de ventes de produits par exemple) en adoptant un grain plus grossier (le ticket plutôt que l'article).

Complément : Ticket de caisse

La lecture de Entrepôts de données : guide pratique de modélisation dimensionnelle [(Kimbal, Ross, 2003)] est recommandée pour la modélisation dimensionnelle des tickets de caisse (en particulier pages 31-60 sur la grande distribution).

C. Le modèle en étoile

Objectifs

Connaître les principes de la modélisation dimensionnelle

1. Modélisation logique de données en étoile

Définition : Le modèle en étoile

Le **modèle en étoile** est une représentation fortement **dénormalisée** qui assure un haut niveau de performance des requêtes même sur de gros volumes de données.

Exemple

Exemple de modèle dimensionnel en étoile

Complément : Modèle en flocon

Le modèle en flocon est aussi un modèle dénormalisé, mais un peu moins que le modèle en étoile : il conserve un certain niveau de décomposition pour chaque dimension prise isolément.

Complément : Voir aussi

Modélisation en flocon

2. Objectifs du modèle dimensionnel

La modélisation par schéma en étoile, par opposition aux schémas normalisés en 3NF, permet de répondre à deux besoins caractéristiques des systèmes décisionnels : la **performance** et la **simplicité** des requêtes.

Performance

En effet en tant que structures **redondantes** les schémas en étoiles permettent d'agréger la table des faits avec n'importe qu'elle dimension en **une seule opération de jointure** (deux ou trois pour les schémas en flocons).

Ce gain de performance est souvent critique puisque les volumes de données sont généralement d'un ordre de grandeur très supérieur à celui des systèmes transactionnels.

Cette redondance ne pose pas les mêmes problèmes que dans les systèmes transactionnels, en effet :

- les données étant statiques (importées), il n'y a pas de risque de divergence d'information lors de mises à jour
- l'usage du *datawarehouse* étant essentiellement statistique (regroupement), la conséquence d'une éventuelle erreur n'est pas du même ordre que dans un système transactionnel.

Simplicité

La présentation en étoile des données, avec les faits au centre et les dimensions autour, est particulièrement adaptée à **l'écriture rapide de requêtes simples** pour agréger des données de la table des faits selon des regroupements sur les tables de dimensions.

L'enjeu est de pouvoir répondre simplement et rapidement à une question simple, tandis qu'un modèle transactionnel, qui répond à d'autres contraintes, nécessitera souvent un code SQL complexe et des opérations multiples pour répondre à la même question.

Cela permet notamment aux utilisateurs finaux de construire facilement de nouvelles requêtes au fil de leur exploration des données.

Fondamental : Caractéristiques d'un bon modèle décisionnel

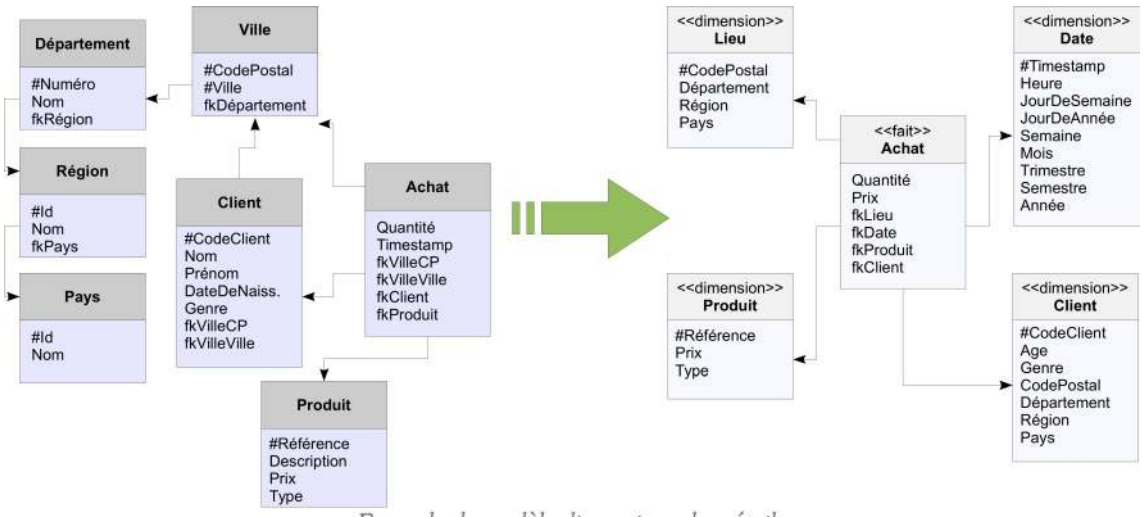
- **Être performant pour le calcul d'agrégats sur de gros volumes de données (exploration de données, *reporting*)**
- **Être appréhendable par un utilisateur final, en particulier pour formuler facilement des requêtes (exploration de données)**
- **Être suffisamment performant au chargement pour répondre aux sollicitations de mise à jour (ETL★)**
- **Être évolutif en fonction des évolutions amont (sources transactionnels) et aval (besoins d'exploitation) (ETL, métadonnées)**

3. Extraction Transformation Loading

Définition : ETL

L'ETL (*Extraction Transformation Loading*) est le processus de copie des données depuis les tables des systèmes transactionnels vers les tables du modèle en étoile du data warehouse.

Exemple



Exemple de modèle dimensionnel en étoile

Remarque

Les tables du modèle dimensionnel peuvent être vues comme des vues concrètes sur le systèmes transactionnel, à la nuance que des transformations (correction d'erreur, extrapolation...) peuvent avoir été apportées dans le processus ETL.

D. Les outils du décisionnel

Objectifs

Connaître les grandes classes d'outils du domaine du décisionnel

Connaître quelques outils du marché

1. ETL, reporting, exploration, analyse

Fondamental : Principaux types d´outils d´une architecture décisionnel

- **ETL★**
- ***Reporting***
- **Exploration**
- **Analyse**

(Smile, 2012) [(Smile, 2012)]

Exemple : ETL

Ils permettent de concevoir et d'organiser les processus de migration du système transactionnel vers le système décisionnel.

Exemple : Outils de reporting

Ils permettent :

- la création graphique de rapport
- l'accès aux sources de données via des API dédiées
- ...

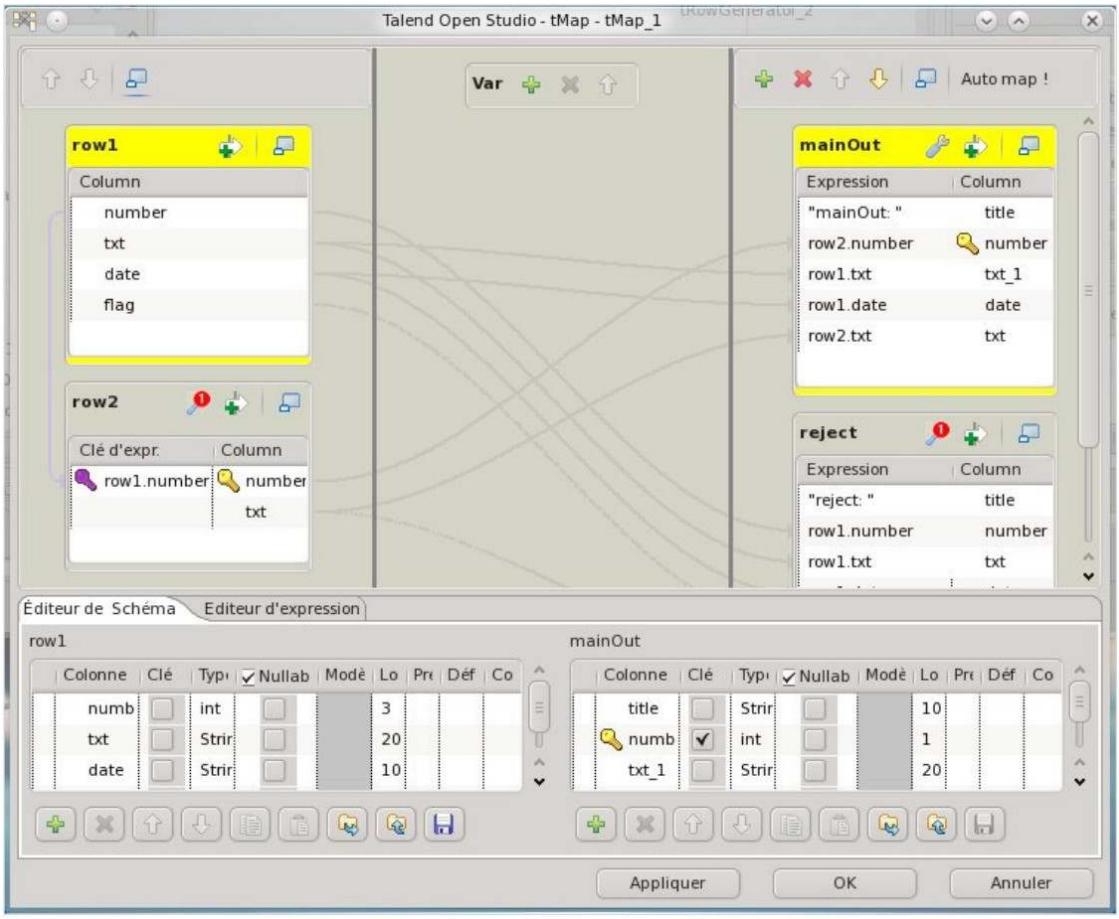
Exemple : Outils d´exploration

Ils permettent de manipuler interactivement des cubes multidimensionnels (choix des dimensions à croiser et des types d'agrégations à effectuer)

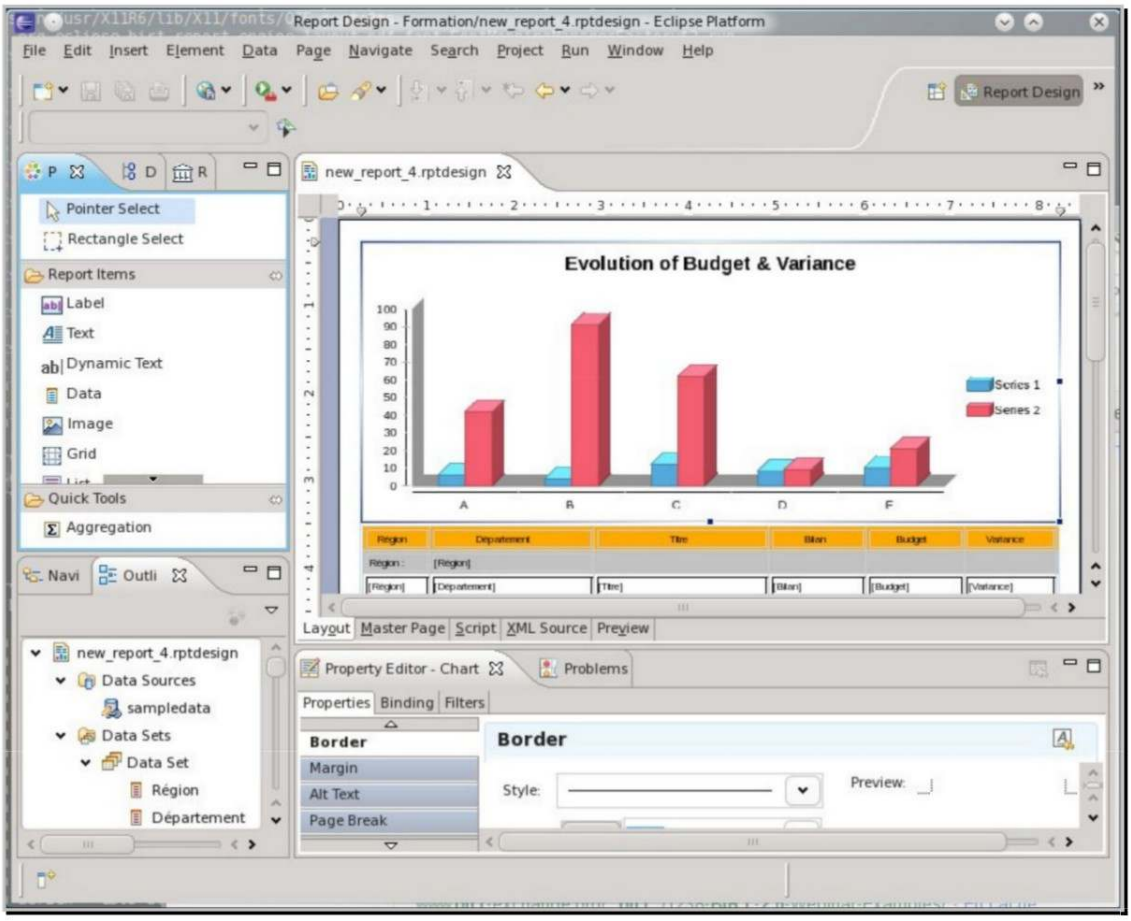
Exemple : Outils d´analyse

Ils permettent l'analyse statistique de données.

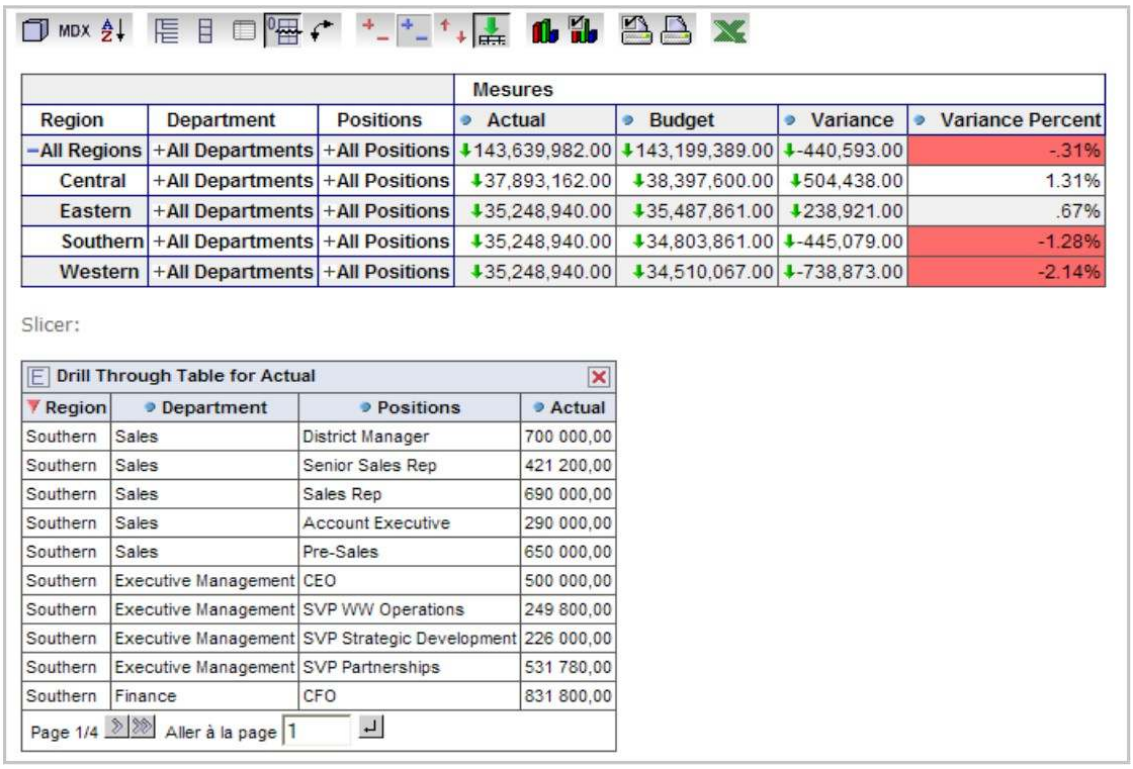
Exemples d´outils Open Source



Outil d´ETL Talend



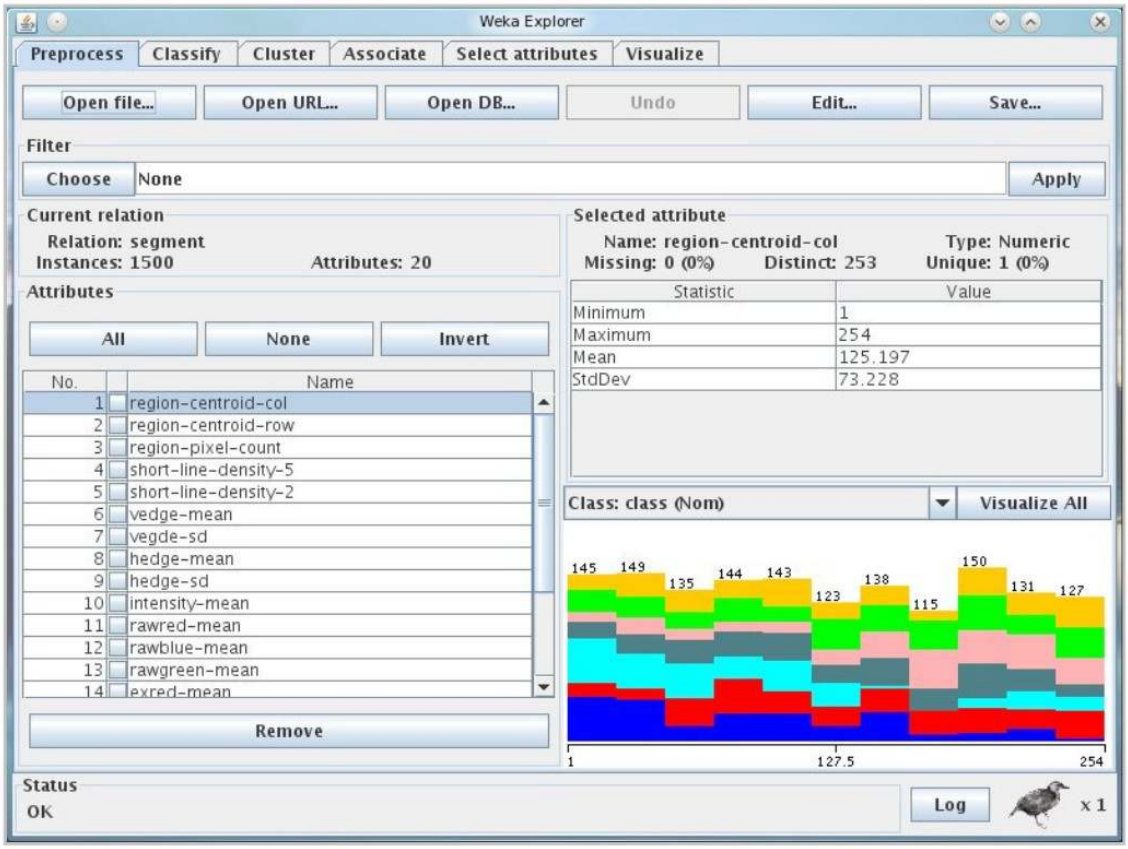
Outil de reporting Birt



			Mesures			
Region	Department	Positions	Actual	Budget	Variance	Variance Percent
−All Regions	+All Departments	+All Positions	143,639,982.00	143,199,389.00	-440,593.00	-.31%
Central	+All Departments	+All Positions	37,893,162.00	38,397,600.00	504,438.00	1.31%
Eastern	+All Departments	+All Positions	35,248,940.00	35,487,861.00	238,921.00	.67%
Southern	+All Departments	+All Positions	35,248,940.00	34,803,861.00	-445,079.00	-1.28%
Western	+All Departments	+All Positions	35,248,940.00	34,510,067.00	-738,873.00	-2.14%

Region	Department	Positions	Actual
Southern	Sales	District Manager	700 000,00
Southern	Sales	Senior Sales Rep	421 200,00
Southern	Sales	Sales Rep	690 000,00
Southern	Sales	Account Executive	290 000,00
Southern	Sales	Pre-Sales	650 000,00
Southern	Executive Management	CEO	500 000,00
Southern	Executive Management	SVP WW Operations	249 800,00
Southern	Executive Management	SVP Strategic Development	226 000,00
Southern	Executive Management	SVP Partnerships	531 780,00
Southern	Finance	CFO	831 800,00

Outil d´exploration de données JPivot



Outil d´analyse statistique Weka

2. SGBD orientés décisionnel

Il est possible d'utiliser une base relationnelle classique pour implémenter un entrepôt de données modélisé en étoile (c'est même aujourd'hui encore la forme la plus largement mobilisée).

Il existe également des technologies dédiées (qui peuvent s'appuyer sur des bases relationnelles ou sur des structures de données dédiées).

Le mouvement NoSQL réintègre progressivement des problématiques décisionnelles, reconfigurant petit à petit les approches technologiques liées à ce domaine.

Exemple : Teradata

Teradata est une technologie dédiée aux BD massivement parallèles, c'est à dire capable de faire exécuter une requête par plusieurs machines en parallèle, afin d'en accélérer le traitement. C'est à la fois un SGBD, un OS dédié (Unix) et des machines dédiées.

Complément

Voir Entrepôts de données : guide pratique de modélisation dimensionnelle [(Kimbal, Ross, 2003)], p.14.

Bases de la modélisation dimensionnelle

II

A. Principes de la modélisation dimensionnelle

Objectifs

Connaître les principes, les étapes et les méthodes de la modélisation dimensionnelle

1. Approche générale de modélisation

Rappel

La modélisation en étoile

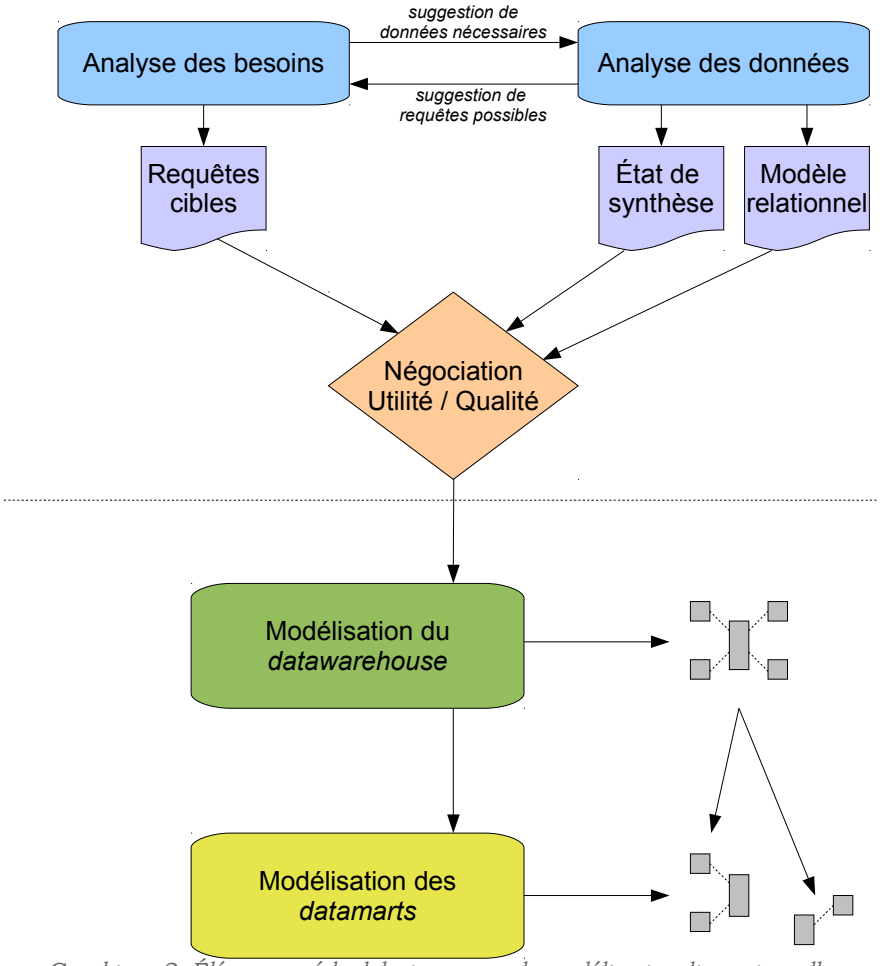
Fondamental

Un modèle dimensionnel est le résultat :

- **d'une analyse des besoins :** ce que je souhaite étudier.
- **d'une analyse des données disponibles :** ce que je peux étudier.

Méthode : Méthode générale de modélisation

1. Analyse des données
   a. Étude des sources de données (quantification, analyses générales)
   b. Qualification des données (qualité et intérêt)
   c. Intégration logique des données (simulation d'un schéma relationnel virtuel)
   d. Normalisation du schéma virtuel en 3NF pour en avoir une vue cohérente
2. Analyse des besoins clients
   a. Exprimer les besoins sous la forme de requêtes décisionnelles
   b. Réaliser les vues hiérarchiques pour chaque requête
3. Sélectionner les requêtes qui seront effectivement réalisables en fonction des données disponibles
4. Conception du data warehouse et des data marts
   a. Séparer les requêtes en fonction de la granularité de la table des faits (grain fin des ventes, grain plus grossier du ticket de caisse, etc.)
   b. Créer un data warehouse intégrant toutes les requêtes de grain fin
   c. Extraire un data mart par niveau de grain supérieur et/ou pour des thématiques particulières nécessitant par exemple une pré-agrégation



Graphique 2 Éléments méthodologiques pour la modélisation dimensionnelle

Remarque

Il est généralement intéressant de paralléliser les tâches d'analyse des besoins et d'analyse des données.

En particulier il est inutile d'aller trop loin dans l'expression de besoins que l'on sait **a priori** impossibles à satisfaire pour cause d'absence de donnée ou d'absence de donnée exploitable.

Rappel : Informations

Il est conseillé de conserver certains champs d'information dans le modèle dimensionnel, même s'ils ne seront pas exploités pour les calculs ou les agrégats.

Cela permettra par exemple d'identifier des enregistrements, comme les désignations de produits.

On pourra noter en italique ces champs dans le modèle dimensionnel.

2. Table des faits

Définition

A row in a fact table corresponds to a measurement. A measurement is a row in a fact table. All the measurements in a fact table must be the same grain. (Kimball, Ross, 2008, p.17)

Fact tables express the many-to-many relationships between dimensions in dimensional models.

(Kimball, Ross, 2008, p.19)

Remarque

La table des faits est (dans la plupart des cas) la table la plus volumineuse (avec le plus grand nombre de lignes) du modèle.

Exemple

Daily Sales Fact Table
Date Key (FK)
Product Key (FK)
Store Key (FK)
Quantity Sold
Dollar Sales Amount

Exemple de table des faits (Kimball, Ross, 2008, p.17)

Fondamental : Faits additifs et numériques

The most useful facts are numeric and additive. (Kimball, Ross, 2008, p.17)

Méthode : Granularité minimale

Preferably you should develop dimensional models for the most atomic information captured by a business process. Atomic data is the most detailed information collected; such data cannot be subdivided further. (Kimball, Ross, 2008, p.34)

Méthode : Granularité des data marts

Pour un data mart on peut pré-agréger sur un grain plus gros que le data warehouse : des colonnes d'agrégation (somme, moyenne, compte...) peuvent alors apparaître pour rendre compte statistiquement d'informations perdues à l'agrégation.

3. Table des dimensions

Définition

Dimension tables are the entry points into the fact table. [...] The dimension implement the user interface to the data warehouse. (Kimball, Ross, 2008, p.20)

Exemple

Product Dimension Table
Product Key (PK)
Product Description
SKU Number (Natural Key)
Brand Description
Category Description
Department Description
Package Type Description
Package Size
Fat Content Description
Diet Type Description
Weight
Weight Units of Measure
Storage Type
Shelf Life Type
Shelf Width
Shelf Height
Shelf Depth
... and many more

Exemple de table de dimension (Kimball, Ross, 2008, p.20)

Conseil : Intelligibilité

The best attributes are textual and discrete. Attributes should consist of real words rather than cryptic abbreviations. (Kimball, Ross, 2008, p.20)

B. Projet Fantastique : Problème posé

Vous travaillez en tant qu'ingénieur spécialisé dans les systèmes décisionnels au siège de l'entreprise française "Fantastique".

L'entreprise "Fantastique" vend principalement des ouvrages de divertissement de type science fiction, thriller, policier... Elle dispose pour cela de plusieurs magasins de vente dans les centres des grandes villes en France.

La direction de l'entreprise souhaite faire une étude large sur les ventes de l'année passée afin de prendre des orientations stratégiques nouvelles : ouverture de nouveaux magasins, fermeture ou transfert de magasins mal implantés, extension territoriale à de nouveaux départements français, réorganisation des directions, réorientation du marketing, élargissement ou réduction du catalogue, etc.

Fondamental

La question posée est donc : **quels sont les facteurs sur lesquels l'on pourrait jouer pour augmenter les ventes ?**

Elle vous charge dans ce cadre de mettre en place une solution logicielle permettant d'intégrer les données pertinentes et de pouvoir les interroger efficacement sous des angles divers.

Notons que bien entendu, la direction éclairée de l'entreprise ne compte pas se fier à ces seuls facteurs de ventes pour prendre ses décisions, mais bien privilégier les facteurs sociaux et territoriaux, en dialoguant avec ses salariés et ses clients, pour maintenir sa mission culturelle et son rôle d'entreprise citoyenne. Votre posture d'ingénieur est bien entendu de se préoccuper de ces dimensions fondamentales, même si elles seront largement ignorées dans le cadre de cet exercice à vocation essentiellement technique. Elle pourront néanmoins être brièvement abordées en marge de vos rapports d'analyse.

C. Projet Fantastic : Données disponibles

Catalogue des livres

Une base Oracle contient le catalogue complet de l'entreprise que chaque magasin a à sa disposition.

- Cette base, composée d'une seule table publique `catalogue`, est disponible sur le serveur Oracle `sme-oracle.sme.utc`, sous le schéma `nf26`.

Fichier des ventes

Un fichier contient une consolidation de l'ensemble des ventes de l'année passée réalisées dans chaque magasin.

- Ces données sont disponibles sous la forme d'un fichier CSV dans un répertoire du serveur `sme-oracle.sme.utc` : `/home/nf26/data`
- La structure du fichier est : Numéro de ticket, date de ticket, produit, magasin

Fichier des magasins

Un fichier ODS géré par la direction marketing contient pour chaque magasin l'organisation des rayonnages : `marketing.ods`

- Le responsable des ventes de chaque département décide de l'organisation des rayonnages des magasins de son département.
- Il existe 3 types de rayonnage : par Auteur (A), par Année (Y), par Éditeur (E)
- Le fichier est déposé dans un répertoire du serveur `sme-oracle.sme.utc` : `/home/nf26/fantastic`

Données géographique sur les départements

Un stagiaire a trouvé sur Internet un fichier permettant de connaître la population de chaque département, présageant que cette information sera utile.

- Le stagiaire parvient à trouver une information un peu datée qui pourra suffire sous la forme d'un fichier CSV : `departementsInsee2003.txt`.

departementsInsee2003.txt.zip

Document 1 departementsInsee2003.txt (population par département)

Méthode

Inspecter les données pour chaque source :

1. Se connecter à la base Oracle avec SQL Developer et inspecter le schéma de la table Oracle.
2. Ouvrir un terminal et se connecter en `ssh` au serveur.
   Utiliser la commande Unix `more` pour regarder les premières lignes du fichier `data.csv`.
3. Récupérer le fichier CSV `departementsInsee2003.txt` et l'ouvrir avec un éditeur de texte.
4. Récupérer le fichier ODS et l'ouvrir avec un traitement de texte.

D. Étude des besoins utilisateurs

Objectifs

Savoir formaliser une étude de besoins sous forme de requêtes multidimensionnelles

1. Requête décisionnelle

Définition : Requête décisionnelle

Une requête décisionnelle exprime toujours la mesure d'une quantification de **faits** par rapport à des **dimensions**, sous une forme du type : "Quelle a été la quantité de ... en fonction de ...".

Synonyme : Vue, requête multidimensionnelle

Fondamental

- **Les faits sont des grandeurs que l'on cherche à mesurer (prix, quantité...)**
- **Les dimensions sont des axes d'analyse (date, lieu, produit, personne...)**

Syntaxe

```
quantité de faits
/ dimension1
/ dimension2
...
```

Exemple

"Quelle a été la quantité de produits vendus en fonction des départements et des mois de l'année."

```
quantité de produits vendus
/ département
/ mois
```

2. Rapport

Définition : Rapport

La réponse à une requête décisionnelle est un rapport, généralement sous une forme tabulaire ou graphique.

Synonyme : État

Exemple

01													02												03											
J	F	M	A	M	J	J	A	S	O	N	D		J	F	M	A	M	J	J	A	S	O	N	D		J	F	M	A	M	J	J	A	S	O	N	D
04													05												06											
J	F	M	A	M	J	J	A	S	O	N	D		J	F	M	A	M	J	J	A	S	O	N	D		J	F	M	A	M	J	J	A	S	O	N	D
07													08												09											
J	F	M	A	M	J	J	A	S	O	N	D		J	F	M	A	M	J	J	A	S	O	N	D		J	F	M	A	M	J	J	A	S	O	N	D

Exemple de rapport

Méthode

Les besoins des utilisateurs s'expriment en général plus facilement sous la forme d'exemples de rapports recherchés.

3. Hiérarchie

Définition : Hiérarchie

Une hiérarchie est un ensemble de paramètres d'étude de granularité croissante appartenant à une même dimension au sein d'un modèle décisionnel.

Exemple

```
quantité de produits vendus
/ lieu (département, région)
/ date (jour, mois, trimestre, semestre)
```

Remarque

Une même dimension peut contenir plusieurs hiérarchies.

E. Projet Fantastique : Étude des besoins

[30 min]

À partir de l'étude des besoins sommaire effectuée par un de vos collègues, et en fonction des données disponibles, exprimer les **requêtes cibles** de votre système.

Le contexte de l'exercice ne permet pas de dialoguer réellement avec des utilisateurs, **en situation réelle il faudra développer cette phase de recueil des besoins des utilisateurs**. Vous pourrez amendez l'indicateur d'utilité des données en fonction de cette étude.

Question 1

La direction marketing est en charge de l'implantation des magasins dans les départements et de l´organisation des rayonnages (type de rangement et présence de rayons spécifiques pour les best-sellers). Elle cherche à savoir si l´organisation du rayonnage des magasins a une influence sur les volumes ventes, et si cela varie en fonction des jours de la semaine ou de certaines périodes de l´année. Elle voudrait également savoir si certains magasins ou départements sont plus dynamiques que d´autres.

Question 2

La direction éditoriale se demande si certains livres se vendent mieux à certaines dates et/ou dans certains magasins ou départements. Elle aimerait également savoir si certains auteurs ou éditeurs se vendent mieux, et s´il existe un lien entre l´ancienneté des livres et les ventes. Elle se demande aussi si certaines périodes sont plus propices que d´autres à l´écoulement des livres les plus anciens.

F. Étude des données

Objectifs

Savoir faire une étude des données existantes

1. Étude séparée des sources données

Méthode

Pour chaque source de données identifiée on proposera une synthèse avec les informations suivantes :

- Nom, origine précise, nombre de lignes de la source
- Nom, type et description des colonnes
- Qualité
  - 0 : données inexploitables
  - 1 : données peu exploitables (traitements incertains)
  - 2 : données exploitables après traitements
  - 3 : données exploitables sans traitement
- Utilité
  - 0 : données sans intérêt

- 1 : données utiles pour la documentation uniquement
- 2 : données utiles a priori
- 3 : données utiles avec certitude

- Commentaires.

Exemple

Nom	Description	Type	Qualité	Utilité	Commentaire
isbn	Identifiant international d'un livre publié	char(3) et char(13)	3	3	Certaines ISBN ne sont pas corrects (3 caractères au lieu de 13) ; clé de référencement dans le fichier de ventes
titre	Titre du livre	varchar(255)	2	1	
auteur	Auteur(s) du livre	varchar(255)	1	2	Peut contenir plusieurs auteurs ; certains auteurs sont inscrits différemment d'un livre à l'autre
...					

Tableau 1 Table Oracle oracle.utc.fr/schema.table [1000 lignes]

2. Étude intégrée des sources de données

Méthode

Afin d'avoir une vision globale de l'ensemble des données, il est conseillé de rétro-concevoir :

- une représentation relationnelle des données telles qu'elles existent
- une représentation relationnelle des données idéalisées (telles qu'elles existeraient si elles étaient normalisées dans une même BD)
- une représentation conceptuelle en UML

G. Projet Fantastique : Étude des données

[1h]

Afin de réaliser votre travail, l'entreprise vous met à disposition les données suivantes.

Dans le contexte de cet exercice, les données vous sont livrées *a priori*, notez que dans un contexte réel, vous aurez la plupart du temps à rechercher vous même les données qui peuvent exister.

Données disponibles

Question 1

Établissez le modèle **relationnel** sous-jacent aux données présentes.

Indice :

Pour initier une connexion ssh : `ssh user@serveur`

Question 2

Étudiez les données dont vous disposez et proposez une synthèse des données disponibles pour chaque source.

Indice :

Pour compter les lignes d´un fichier texte sous Linux, on peut utiliser la commande `wc -l`

Question 3

Afin de clarifier les données et leur organisation, rétro-concevez un modèle relationnel **normalisé** en troisième forme normale unifiant toutes les données grâce à l'identification de clés primaires et l'expression de clé étrangères.

Question 4

Rétro-concevez le modèle **conceptuel** en UML correspondant au modèle relationnel normalisé (un modèle UML peut être plus facile à utiliser ensuite).

H. Modélisation du datawarehouse

Objectifs

Savoir faire un modèle dimensionnel en étoile pour un cas simple

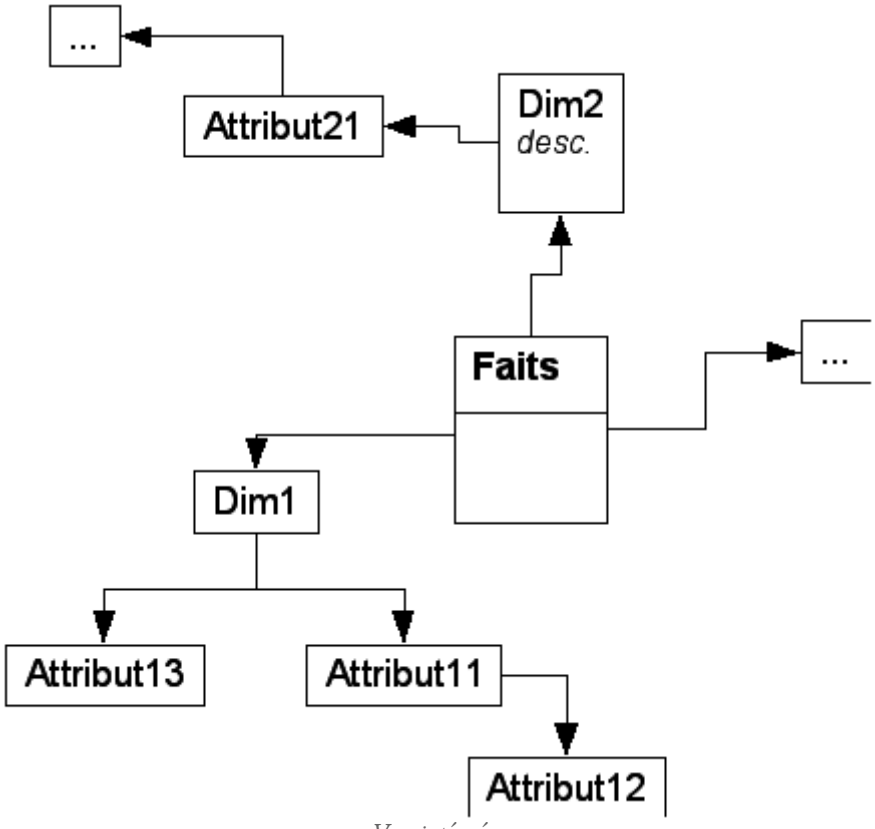
1. Intégration des besoins

Méthode

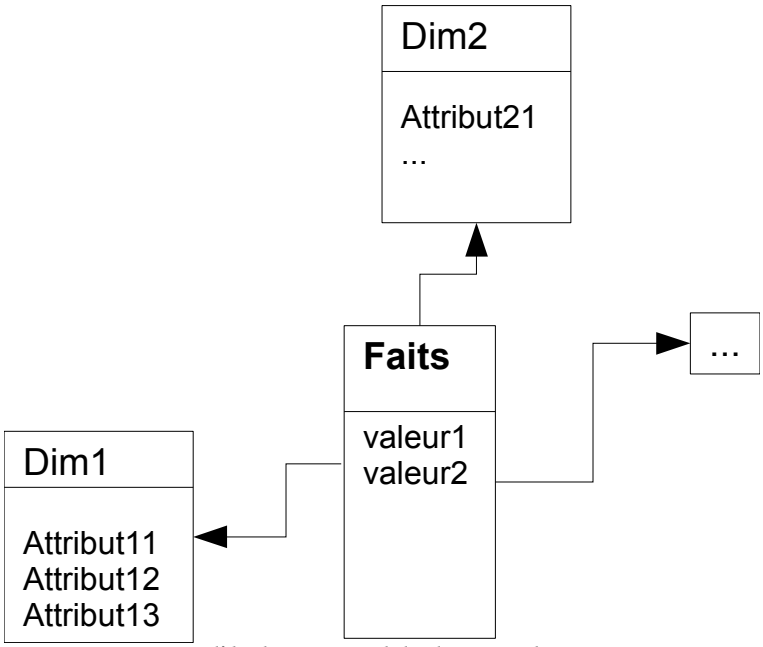
Les différentes vues formalisées lors de l'étude des besoins utilisateurs doivent être intégrées :

- une pour le data warehouse,
- et pour chaque data mart.

Syntaxe



Vue intégrée



Modèle dimensionnel du data warehouse

2. Arbitrages pour le choix des données

Conseil : Pilotage par les besoins

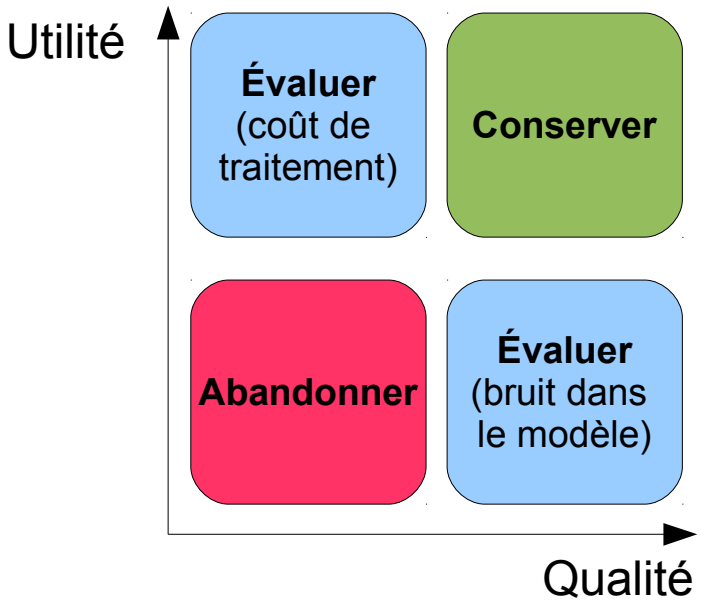
Un projet de *datawarehouse* est destiné à répondre à des besoins d'analyse, en cela il doit prioritairement prendre en considération les besoins réels d'analyse des utilisateurs. En particulier il est déconseillé de travailler uniquement à partir des données sources existantes, puisque les utilisateurs cherchent justement à sortir de la réalité transactionnelle (Entrepôts de données : guide pratique de modélisation dimensionnelle [(Kimbal, Ross, 2003)], p.34).

Bien entendu, la possibilité de répondre aux besoins dépend des données disponibles et de leur qualité.

Méthode : Diagramme Utilité / Qualité

Chaque données peut être qualifiée selon une utilité (donnée intéressante ou non) et une qualité (données facile à exploiter ou coûteuse à nettoyer).

Ces deux paramètres peuvent ensuite servir à choisir les données que l'on conserve (et que l'on va devoir éventuellement nettoyer) de celle que l'on abandonne.



Graphique 3 Diagramme Utilité / Qualité

3. Métadonnées

Méthode

Documentez la table des faits ainsi que chaque dimension de votre data warehouse, avec pour chaque attribut :

- Le nom
- Le type précis
- Une description

On identifiera les clés candidates et la clé primaire de chaque dimension, les informations uniquement descriptives.

On pourra préciser lorsque ce n'est pas évident le mode de calcul des attributs en fonction des sources.

I. Projet Fantastique : Modélisation

À partir de l'étude des données et des besoins utilisateurs, proposez un modèle **dimensionnel** pour un **data warehouse** permettant de traiter la question générale qui a été posée.

Question 1

Proposez une modélisation dimensionnelle **en étoile** pour chaque contexte d'usage (directions marketing et éditoriale) :

1. identifiez la table des faits
2. identifiez les dimensions en intégrant les vues exprimées (utilisez le rapport qualité/utilité pour décider des données que vous souhaitez conserver)

Question 2

Intégrer vos deux sous-modèles pour proposer le modèle de votre **data warehouse**.

Vous pourrez augmenter vos dimensions de données disponibles bien que non identifiées a priori lors de l'analyse des besoins.

Question 3

Établissez les métadonnées du modèle dimensionnel : décrivez chaque données (type précis, description...) ; identifiez la clé primaire de chaque dimension, ainsi que les informations descriptives.

Introduction à l'ETL et application avec Oracle

III

A. Principes généraux d'un processus ETL

1. Principe de l'ETL

Définition : Processus "Extraction, Transformation, Load"

L'ETL✯ est le processus qui permet de charger un data warehouse à partir de données externes généralement issues de bases transactionnelles. Son rôle est de récupérer ces données et de les traiter pour qu'elles correspondent aux besoins du modèle dimensionnel.

En général les données sources doivent être "nettoyées" et aménagées pour être exploitables par les outils décisionnels.

Fondamental

You get the data out of its original source location (E), you do something to it (T), and then you load it (L) into a final set of tables for the users to query.

(Kimball et al., 2008, p369)

Fondamental

Selon Kimball (2004, p.xxi) [(Kimball, Caserta, 2004)] 70% de l'effort consacré à un projet de BI est dépensé dans l'ETL.

2. ETL ex nihilo ou outil d'ETL

Un ETL peut être :

- développé *ex nihilo* pour un projet directement dans un langage bas niveau (Java, SQL, PL/SQL...) ;
- ou s'appuyer sur un outil d'ETL (Talend Open Studio, Pentaho Data Integration, Informatica PowerCenter, ...).

Fondamental

ETL Tool versus Hand Coding (Buy a Tool Suite or Roll Your Own?)

The answer is, "It depends."

(Kimball, Caserta, 2004, pp10-13)

Méthode : ETL basés sur un outil

Les avantages offerts par l'outil ETL sont :

- De structurer et de rassembler l'ensemble des morceaux de code nécessaire aux transferts et aux transformations des données
- D'offrir une représentation graphique des flux et opérations
- De faciliter la maintenance et l'évolution de l'ETL
- D'intégrer la gestion des métadonnées
- D'intégrer la gestion des erreurs
- De disposer d'API dédiées (connexion, import/export...) d'accès aux données (CSV✯, BD✯, XML✯...)
- ...

Méthode : ETL ex nihilo

Les avantages offerts par une approche manuelle sont :

- L'homogénéité technologique et la disponibilité interne des compétences : les équipes utilisent les langages qu'elles maîtrisent sans apprentissage et médiation d'un outil tiers.
- La flexibilité : tout est possible.
- Le traitement des fichiers plats (hors BD✯) peut être plus simples et plus performant avec des langages proches des systèmes.
- ...

3. ETL en mode batch ou en mode flux

Fondamental : ETL en mode batch

> *The standard architecture for an ETL system is based on periodic batch extracts from the source data, which then flows through the system, resulting in a batch update of the final end user tables.*
>
> (Kimball, Caserta, 2004, p13)

Un ETL alimente en général un data warehouse par des processus *batch* périodiques.

Remarque : ETL en mode flux

Il existe néanmoins des applications nécessitant des data warehouses alimentés en temps réel en mode flux (qui ne sont pas abordés dans le cadre de ce cours).

4. ETL incrémental

Définition : ETL non incrémental

Un ETL non incrémental est :

- soit un ETL qui ne sert qu'une seule fois (*one shot*) ;
- soit un ETL qui refait 100% du processus de migration à chaque fois que l'on souhaite une mise à jour (le data warehouse est vidé puis rempli à nouveau avec les données actuelles)

  On notera qu'un tel ETL **ne gère pas d'historisation**.

Définition : ETL incrémental

Un ETL instrumente normalement un processus incrémental.

1. Les données sont modifiées dans les systèmes transactionnels :
   - mise à jour des dimensions ;
   - ou ajouts de nouveaux faits.
2. L'ETL répercute les mises à jour dans le data warehouse.

Attention : Accumulation des faits

Classiquement les faits s'accumulent dans le data warehouse, il n'y a jamais ni suppression ni mise à jour (croissance monotone).

Attention : Historisation des dimensions

Lorsqu'une dimension est mise à jour, l'ETL doit garder la mémoire des anciennes valeurs afin que les anciens faits restent bien reliés aux anciennes valeurs.

Ils existent plusieurs stratégies pour gérer l'historique des valeurs des dimensions dans le DW :

- **associer des dates aux dimensions et aux faits afin de savoir quelle valeur de dimension est valide pour quel fait ;**
- **créer de nouvelles entrées dans les dimensions (ne pas faire de mise à jour au sens d'UPDATE) ;**
- **...**

Méthode : Stratégies de mise à jour

- Rafraîchissement périodique
- Rafraîchissement manuel
- Rafraîchissement événementiel
- ...

B. Proposition d'architecture simplifiée pour un ETL ex nihilo, batch, non incrémental

Nous proposons un exemple d'architecture **simplifiée** pour la mise en place d'un ETL **ex nihilo**, en mode **batch**, **sans gestion du caractère incrémental**.

Cette architecture est assez générale et pourra être une base pour de nombreux cas, mais elle devra :

- être adaptée néanmoins en fonctions des spécificités propres à chaque contexte.
- être complétée (gestion incrémentale, gestion des rejets, audits...)

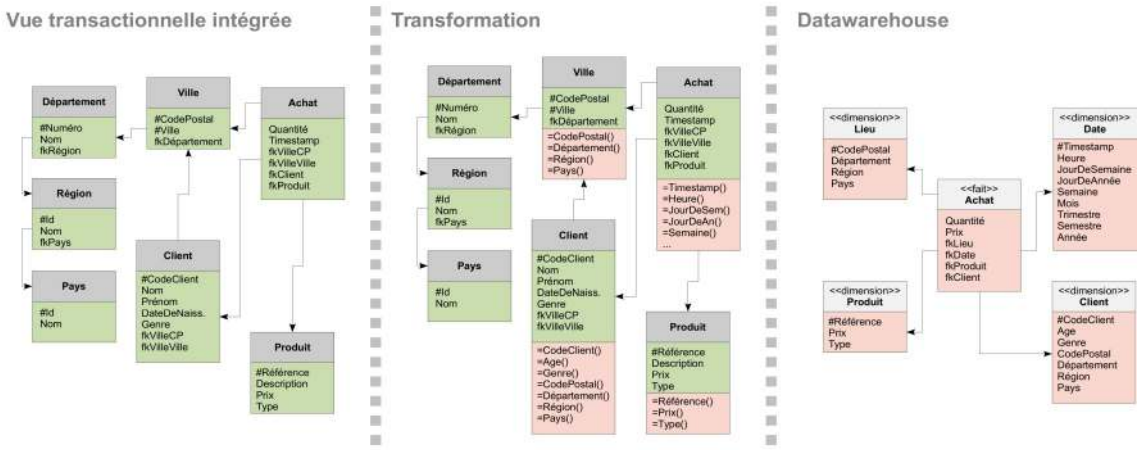
1. Architecture d'ETL à trois zones

Méthode

Nous proposons une architecture d'ETL organisée avec trois zones composées chacune d'une base de données distincte :

- **Zone d'extraction**
  Une base de données destinée à unifier les sources de données et offrir un point d'accès unique.
- **Zone de transformation**
  Une base de données destinée à traiter les sources et offrir une interface d'accès aux données transformées (API★).
- **Zone d'exploitation**
  Une base de données destinée à implémenter le data warehouse et les data marts.

Fondamental



Architecture générale d'un ETL

2. Conseils méthodologiques

Méthode : ETL multi-schéma

Dans la mesure du possible, utiliser un schéma de base de données pour chaque zone de l'ETL (BDE, BDT, DW).

Méthode : ETL mono-schéma (contraintes de nommage)

Si tout doit être réalisé au sein d'un seul schéma, utiliser un système de pré-fixage des noms : `bde_table`, `bdt_table`, `dw_table`.

Rappel

Accès inter-schémas sous Oracle

3. Résumé ETL en image

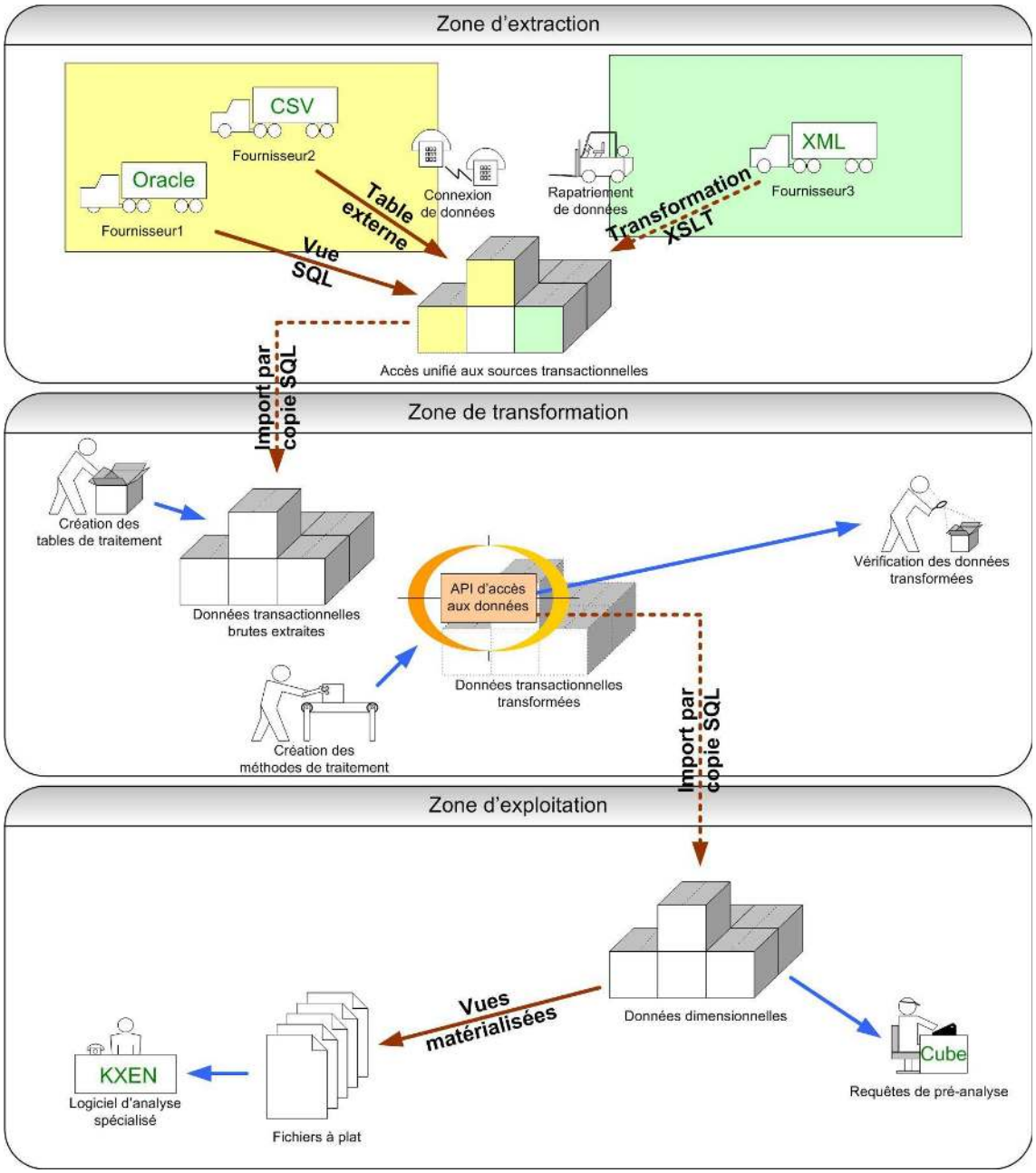


Image 1 Processus ETL

4. Carte des données

Définition : Carte des données.

La carte des données (*logical data map*) est un inventaire et une mise en correspondance des données présentes dans chaque zone.

Exemple

DW				T				E			
Fact/Dim	Table	Column	Data type	Function	Table	Column	Data type	Table	Column	Data type	Source
Dim	dDate	date	YYYY-MM-DD	date()	tDate	date	char(10)	eData	date	varchar(10)	serveur.utc.fr /share/data/ data.csv
Dim	dDate	jds	1..7	jds()	tDate	date					
Dim	dDate	month	1..7	month()	tDate	date					
...											
Dim	dBook	isbn	char(13)	isbn()	tDate	isbn	varchar(255)	eCat	isbn	varchar(255)	serveur-oracle.utc schema.catalogue
Dim	dBook	author	varchar(25)	author()	tDate	author	text	eCat	auteurs	text	
...											

Exemple de cartographie des données

Complément

(Kimball, Caserta, 2004, p56-59) [(Kimball, Caserta, 2004)]

C. Implémentation simplifiée d'une zone d'extraction avec Oracle

Dans cette partie nous précisons comment implémenter pratiquement la zone E d'un processus ETL **simple**, **ex nihilo**, **batch**, **non incrémental** avec une base Oracle RO✯ (version 9i ou postérieure).

1. Zone E : Extraction

Définition : Base de données d´extraction

La BDE✯ est une BD✯ relationnelle destinée à implémenter la zone d'extraction d'un ETL, pour offrir un unique point d'accès à l'ensemble des sources de données.

La BD est composée de :

- tables permettant de **rapatrier** les données à importer depuis des sources externes ;
- et de vues pour se **connecter** à des sources dynamiques situées dans la même BD.

Méthode : Les fichiers CSV

Les données situées dans des fichiers CSV doivent :

1. être rapatriées sur un ou des serveurs accessibles depuis la BDE ;
2. importées dans la BDE : une table pour chaque fichier.

Il faudra automatiser le processus de copie des fichiers si les données sont susceptibles d'être mises à jour.

Remarque : Tables externes

Certains SGBD, comme Oracle, propose une alternative à l'import, grâce à un concept de table externe qui permet de lier dynamiquement une définition de table à un fichier CSV.

Méthode : Les autres fichiers : tableurs, XML...

Pour les fichiers autres que CSV, deux solutions sont à étudier :

- soit votre BDE offre une API d'accès direct à ces formats ;
- soit le fichier est transformé en CSV.

Dans le second cas, il faudra automatiser la transformation si le fichier est susceptible de mises à jour.

Méthode : Les données stockées en BD

Pour les données stockées en BD, trois solutions sont à étudier :

- si la BDE et la BD source sont sur le même SGBD, on créé simplement une vue ;
- sinon, lorsque c'est possible on établit un lien dynamique entre la BDE et les tables sources (propriétaire, ODBC ou JDBC) ;

- sinon, on fait un export de la BD source dans un fichier CSV (en gérant l'automatisation de l'export lorsque la base est vivante).

Méthode : Gestion des contraintes

Les contraintes doivent être relâchées au maximum dans la BDE pour assurer que les données sources seront toutes correctement accessibles.

On veillera à avoir correctement documenté les contraintes connues, notamment pour les données provenant de SGBD dans lesquels ces contraintes sont formalisées dans le schéma.

2. Sources de données

Méthode : Données sur le même serveur Oracle

Réaliser des vues pour accéder dynamiquement aux données sources.

Méthode : Données dynamiques en fichier CSV

Créer une **table externe** Oracle pour accéder dynamiquement à ce fichier.

Si le fichier n'est pas accessible directement depuis le serveur Oracle, procéder à une copie (automatisée par script).

Méthode : Données statiques

Pour les données ne demandant aucune mise à jour, ou des mises à jour très ponctuelles :

- faire un export depuis la source en CSV ;
- créer une table Oracle dans la BDE✯ ;
- importer manuellement le CSV avec SQL Developer.

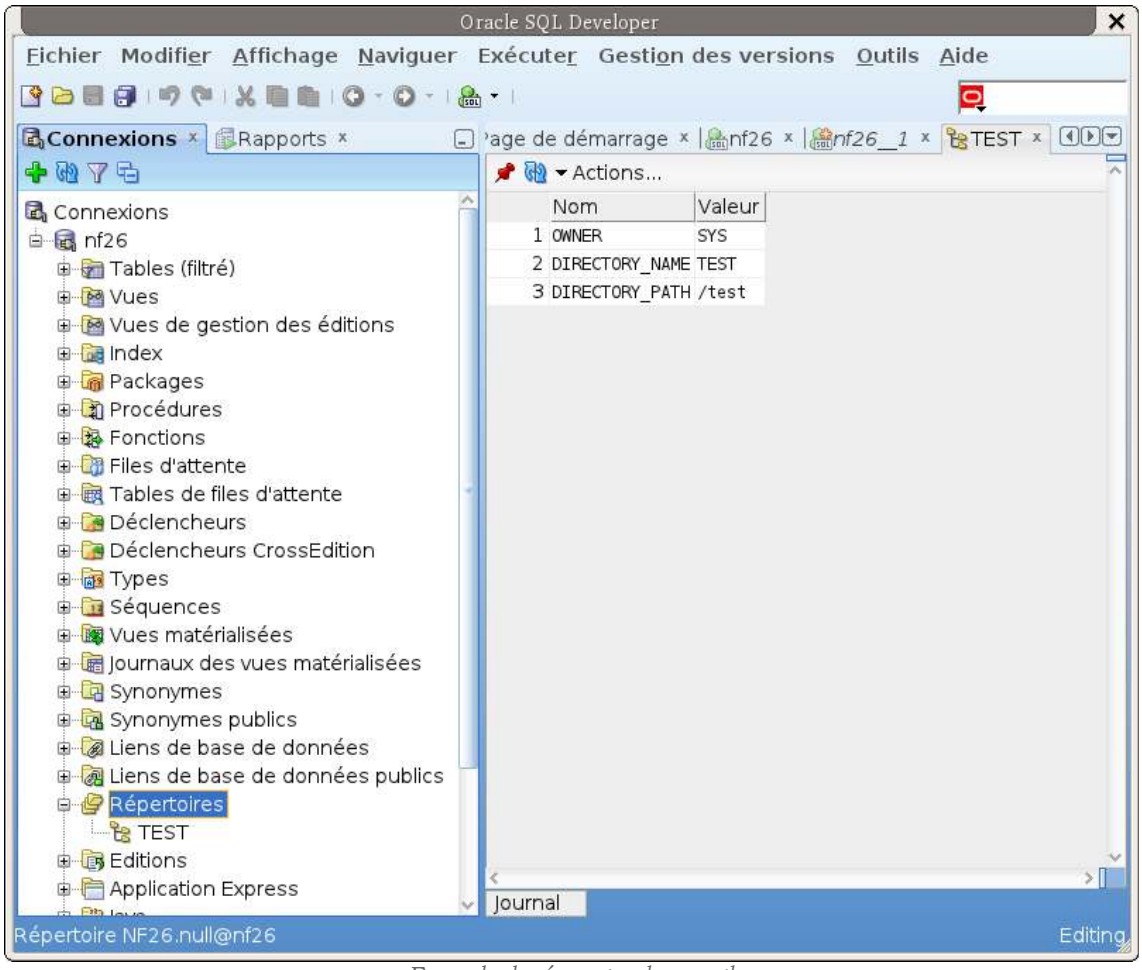
3. Tables externes sous Oracle

Définition : Table externe

Une table externe sous Oracle est une méthode d'accès sans copie à des fichiers CSV dynamiques exactement comme s'il s'agissait d'une table de la BD.

Syntaxe : Préambule : Déclaration des répertoires de travail

```
1 CREATE OR REPLACE DIRECTORY <nom du répertoire source> AS '<chemin du répertoire
  de la source>';
2 CREATE OR REPLACE DIRECTORY <nom du répertoire log> AS '<chemin du répertoire des
  fichiers de log>';
```



Exemple de répertoire de travail

Attention : Accès aux répertoire de travail

Le répertoire de la source et le fichier source doivent être accessibles en lecture pour le processus Oracle.

Le répertoire des fichiers de log doit être accessible en lecture et écriture pour le processus Oracle.

Attention

« All directories are created in a single namespace and are not owned by an individual schema »

http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_5007.htm[1]

Syntaxe : Création d´une table externe

```
1  CREATE TABLE <nom de la table> (
2  <déclaration des attributs avec domaine mais sans contrainte>
3  )
4  ORGANIZATION EXTERNAL
5  (TYPE ORACLE_LOADER
6  DEFAULT DIRECTORY <répertoire déclaré préalablement>
7  ACCESS PARAMETERS
8  (
9  RECORDS DELIMITED BY '<caractère de fin de ligne>'
10 SKIP <nombre de lignes à ignorer>
11 CHARACTERSET <encodage des caractères>
12 BADFILE <répertoire>:'<fichier>'
13 LOGFILE <répertoire>:'<fichier>'
14 FIELDS TERMINATED BY '<séparateur de champ>'
15 OPTIONALLY ENCLOSED BY '<séparateur de chaîne>'
```

1 - http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_5007.htm

```
16 )
17 LOCATION ('<fichier source>'))
18 REJECT LIMIT UNLIMITED;
```

Attention : Ne pas utiliser de commentaire au sein la déclaration des paramètres d´accès

« One important point to remember is that comments must be placed before any access parameters. If you include comments in the access parameter sections, Oracle will throw an error when you query the external table but not when you are creating it. »

https://oracleappsnotes.wordpress.com/2012/02/10/oracle-external-tables-a-few-examples/[2]

Exemple

```
1 CREATE OR REPLACE DIRECTORY monRepertoireSrc AS '/user1c/nf26/nf26/projet/csv/';
2 CREATE OR REPLACE DIRECTORY monRepertoireLog AS
    '/volsme/user1x/uvs/nf26/nf26p099/test/';
3 /
```

```
1  CREATE TABLE tImport (
2  a VARCHAR(50),
3  b NUMBER(10)
4  )
5  ORGANIZATION EXTERNAL
6  (TYPE ORACLE_LOADER
7  DEFAULT DIRECTORY monRepertoireSrc
8  ACCESS PARAMETERS
9  (
10 RECORDS DELIMITED BY newline
11 SKIP 1
12 CHARACTERSET UTF8
13 BADFILE monRepertoireLog:'import.bad'
14 LOGFILE monRepertoireLog:'import.log'
15 FIELDS TERMINATED BY ';'
16 OPTIONALLY ENCLOSED BY '"'
17 )
18 LOCATION ('sources.csv'))
19 REJECT LIMIT UNLIMITED;
```

Description des paramètres

- `DEFAULT DIRECTORY` : Le répertoire où se trouvent le fichier source
- `RECORDS DELIMITED BY` : Séparateur d'enregistrements (`newline` est le caractère de fin de ligne standard du système)
- `SKIP 1` : permet d'ignorer la première ligne d'un fichier CSV lorsque celui-ci contient les entêtes de colonnes
- `CHARACTERSET` : permet de spécifier l'encodage des caractères (UTF8...)
- `BADFILE` : Fichier contenant les enregistrements rejetés à l'import (récréé à chaque exécution)
- `LOGFILE` : Fichier contenant les traces d'exécution (traces ajoutées à chaque exécution)
- `FIELDS TERMINATED BY` : Séparateur de champs (en général `;` dans un CSV)
- `OPTIONALLY ENCLOSED BY` : Séparateur de chaînes (en général " dans un CSV)
- `LOCATION` : Nom du fichier dans le répertoire sélectionné
- `REJECT LIMIT` : Nombre d'enregistrements pouvant être rejetés avant interruption de la requête (un entier ou UNLIMITED)

Syntaxe : Vérification de la déclaration de la table externe

```
1 DESCRIBE Timport;
2 SELECT TABLE_NAME, TYPE_NAME, DEFAULT_DIRECTORY_NAME FROM USER_EXTERNAL_TABLES;
```

2 - https://oracleappsnotes.wordpress.com/2012/02/10/oracle-external-tables-a-few-examples/

Attention : Accès

L'accès à la source externe CSV par Oracle ne se fait en fait qu'à la première interrogation (**SELECT**), donc il est nécessaire d'exécuter un appel à la table pour valider cet accès (seule une vérification syntaxique est faite au moment du **CREATE TABLE**, le fichier CSV peut même ne pas exister).

En pratique si les données comportent des erreurs, les problèmes se déclareront à ce moment là. On consultera le fichiers de log et des enregistrements rejetés pour le savoir.

```
1  SELECT * FROM Timport;
```

Attention : Gros fichiers

Si les volumes de donnée sont importants (s'il y a beaucoup de lignes), privilégiez l'usage de la clause ROWNUM pour éviter de faire transiter des méga-octets de données entre le serveur et le client. Les délais de réaction en seront améliorés.

Vous pouvez aussi utiliser des SUM, MIN, MAX, etc. pour être sûr que toutes les lignes et colonnes sont correctement lues.

Conseil : Tout recompter

Une fois la procédure terminée effectuer un `count(*)` pour déterminer si l'import a bien traité toutes les lignes du fichier source.

Remarque : NUMBER(X)

Pour spécifier des entiers dans les tables externes, utiliser `NUMBER(X)`, avec le `(X)` obligatoire.

Remarque : Caractère de fin de ligne

`RECORDS DELIMITED BY newline` signifie que le caractère standard du système est utilisé comme caractère de fin de ligne.

Or :

- sous Unix le caractère de fin de ligne est "\n"
- sous Windows il est "\r\n"

Donc, si un fichier CSV est encodé sous un système et lu sous un autre, il y aura un problème. Par exemple si le fichier est encodé sous Windows il aura "\r\n" à la fin de chaque ligne, et s'il est lu sous Linux, Oracle cherchera uniquement un "\n", d'où un "\r" résiduel sera considéré comme faisant partie du dernier champ.

La solution la plus robuste est de spécifier **en dur** le caractère de fin de ligne :

- Si le fichier source a été encodé sous Unix : RECORDS DELIMITED BY '\n'
- Si le fichier source a été encodé sous Windows : RECORDS DELIMITED BY '\r\n'

Conseil

Penser à gérer les fichiers de rejet et de log. Le plus simple est de les supprimer après une exécution incorrecte.

- Les fichiers de rejet ne sont pas créés si l'exécution est correcte (et donc un éventuel fichier existant n'est pas modifiée par une exécution correcte)
- Les fichiers de log grossissent à chaque exécution

4. Exemple de chargement de données depuis un CSV par une table externe

Il est possible d'utiliser les tables externes pour charger des données issues d'un fichier CSV dans une table existante.

Exemple

Soit la table `tTypeDefaut` à charger avec un fichier `tTypdeDefaut.txt`.

```
1  tTypeDefaut (#pkTypeDefaut:number(4), libelle:varchar(50),
   fkFamilleDefaut:char(1)=>tFamilleDefaut)
```

On peut utiliser une table externe :

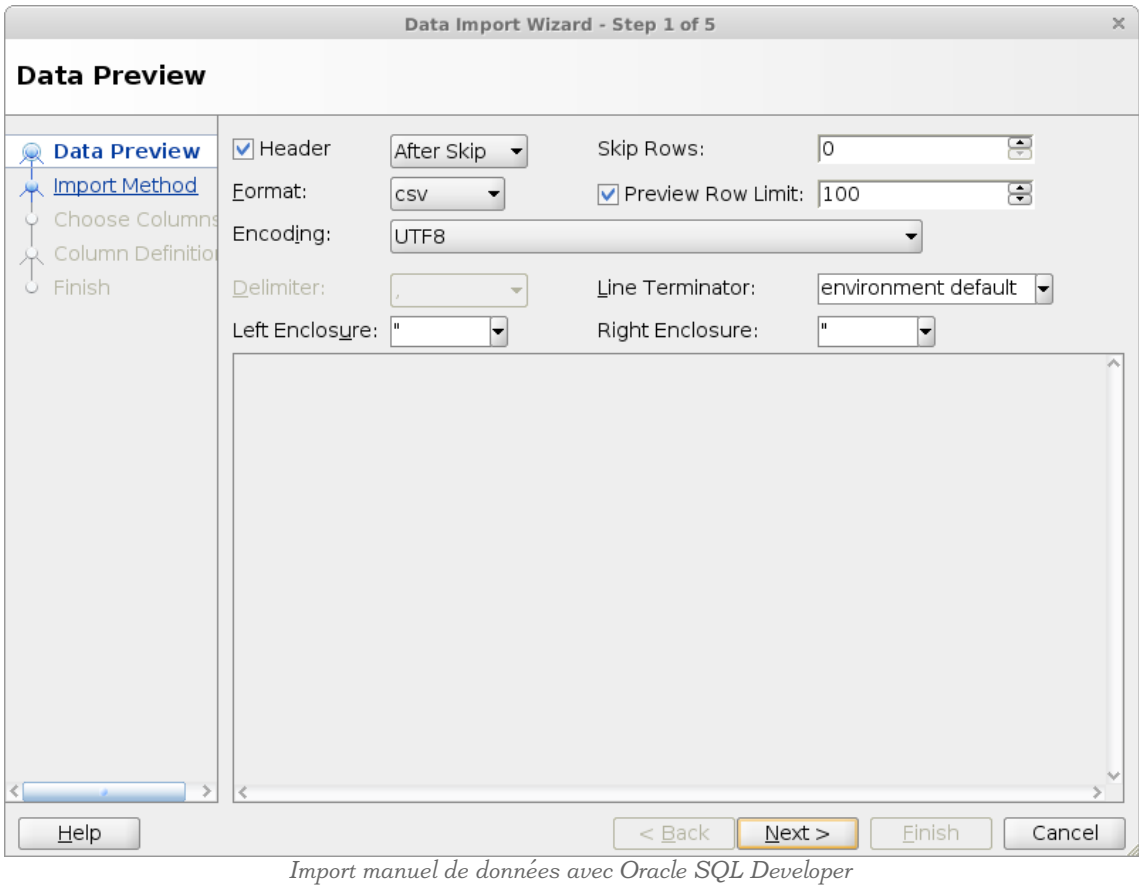
```
1  CREATE TABLE tTypeDefautLoad (
```

```
2  pkTypeDefaut number(4),
3  libelle varchar(50),
4  fkFamilleDefaut char(1)
5  )
6  ORGANIZATION EXTERNAL
7  (TYPE ORACLE_LOADER
8  DEFAULT DIRECTORY srcDir
9  ACCESS PARAMETERS
10 (
11 RECORDS DELIMITED BY NEWLINE
12 BADFILE logDir:'import.bad'
13 LOGFILE logDir:'import.log'
14 FIELDS TERMINATED BY ';'
15 OPTIONALLY ENCLOSED BY '"')
16 LOCATION ('tTypeDefaut.txt'))
17 REJECT LIMIT UNLIMITED;
```

```
1  INSERT INTO tTypeDefaut
2  SELECT * FROM tTypeDefautLoad;
```

5. Insertion CSV manuelle avec SQL Developer

Effectuer un clic droit sur le dossier `tables` à gauche dans SQL Developer et choisir `Import data` pour accéder à un assistant de chargement manuel.



Import manuel de données avec Oracle SQL Developer

D. Projet Fantastic : Mise en place de la zone d'extraction

L'objectif est d'abord de créer la BDE. Les noms des tables et vues seront préfixés :

- `f_` dans le schéma `bde` (f pour projet Fantastique) ;
- ou `f_bde_` si vous ne disposez que d'un seul schéma pour toutes vos BD.

Question 1

Créez une vue `f_bde_catalogue` pour la table de la base Oracle "catalogue".

Indices :

Enregistrez l'instruction de création de votre vue dans un fichier `f_bde_catalogue.sql`.

- *Créez un fichier `f_bde.sql` qui va appeler toutes vos instructions de création d'objets dans la BDE.*

  Sa première ligne est donc : `@f_bde_catalogue.sql`
- *On pourra éventuellement créer un fichier `f.sql` qui appellera les scripts `f_bde.sql`, `f_bdt.sql` et `f_dw.sql`*

Gestion de fichiers SQL

Utiliser l'instruction `CREATE OR REPLACE VIEW` pour la création de la vue, pour permettre la recréation de la vue par le script `f_bde.sql`.

Question 2

Créez une table externe pour chacun des fichiers `marketing.ods` et `departementsInsee2003.txt`.

Indices :

- *Faites un export CSV du fichier `marketing.ods` vers le fichier `marketing.csv`*
- *Copiez le, ainsi que `departementsInsee2003.txt`, dans un dossier `data` de votre compte sur le serveur `sme-oracle.sme.utc`*
- *Ouvrez l'accès à ce dossier en lecture (`chmod 755`)*
- *Créez un dossier `tmp` ouvert en lecture et écriture (`chmod 777`)*
- *Créez les objets `DIRECTORY` permettant de pointer sur les répertoires `data` et `tmp`.*
- *Créez une table externe pour `departementsInsee2003.txt`, en envoyant les fichiers de rejet `departementsInsee2003.txt.bad` et de log `departementsInsee2003.txt.log` dans votre dossier `tmp`.*
- *Testez votre table externe : `SELECT * FROM ...`*
- *Vérifiez que tout s'est bien passé en examinant les fichiers de rejet et de log*
- *De la même façon, créez une table externe pour `marketing.csv` et vérifier l'accès aux données*

Pensez que les objets `DIRECTORY` sont partagés au niveau de toute l'instance et ne sont pas spécifiques à un schéma.

Donc si deux `users` créent un même `DIRECTORY` nommé `tmp`, il y aura un conflit (la seconde création écrasera la première).

Pour accéder à un répertoire `d1` situé dans un répertoire `d0`, `d0` doit être accessible en exécution (`chmod 711`).

Pour lire les fichiers `.log` et `.bad` :

```
1  more ~/tmp/import.log
2
3  more ~/tmp/import.bad
```

Vous pouvez vider les fichiers `.log` régulièrement pour en faciliter la lecture.

```
1  echo > ~/tmp/import.log
```

Vous pouvez supprimer les fichiers `.bad` après avoir traité les causes d'un rejet (si une exécution ne génère pas de rejet elle ne crée pas de fichier de rejet, et ne modifie donc pas un éventuel fichier existant).

```
1 rm ~/tmp/import.bad
```

Question 3

Créez une table externe pour le fichier `data.csv`

Indices :

- *Créez un nouvel objet `DIRECTORY` permettant de pointer sur le répertoire `/home/nf26/data`*

Attention le fichier `data.csv` est très volumineux, aussi ne faites pas de `SELECT *` dessus, sous peine d'attendre longtemps la fin de l'exécution (le serveur devant renvoyer plusieurs Mo de données).

Faites des `SELECT` partiels avec la clause `ROWNUM` pour limiter les données à rapatrier et des select avec opérations de regroupement (min ou max typiquement).

```
1 SELECT ... FROM ... WHERE rownum<=N;
```

Notez qu'en cas d'erreur à l'import portant sur toutes les lignes, le fichier de log risque de devenir très volumineux, pouvant conduire à la saturation de votre compte. Videz le fichier de log après avoir généré une telle erreur.

Pour vérifier la taille du log :

`ls -l ~/tmp/import.log`

Pour tester toutes les lignes : `SELECT count(*), min(...), max(...), max(...) ...`

E. Implémentation simplifiée d'une zone de transformation avec Oracle

Dans cette partie nous précisons comment implémenter pratiquement la zone T d'un processus ETL **simple**, **ex nihilo**, **batch**, **non incrémental** avec une base Oracle RO⋆ (version 9i ou postérieure).

1. Zone T : Transformation

Définition : Base de données de transformation

La BDT⋆ est une BD⋆ relationnelle ou relationnel-objet destinée à implémenter la zone de transformation d'un ETL.

- Elle reçoit une copie des données issue de la zone E.
- Elle permet les transformations des données.
- Elle offre un accès aux données transformées via une API permettant une copie vers la zone L.

Définition : API de la BDT

L'API de la BDT est un ensemble de fonctions - ou méthodes si l'on dispose d'un SGBDRO⋆ qui les autorise - qui permet d'accéder aux données de façon stable (principe d'encapsulation).

L'API de la BDT permet de rendre le chargement du data warehouse moins dépendant aux variations dans les sources de données.

Le principe proposé est le suivant :

- Créer une fonction pour chaque attribut existant dans la data warehouse.
- Appeler ces fonctions lors du chargement du data warehouse, au lieu d'appeler directement les attributs des tables de la BDT.
- Chaque fonction est en charge d'appeler les attributs de la BDT et de faire les traitements nécessaires pour fournir la valeur souhaitée.

Exemple : Exemple d'appel à l'API de la BDT en RO

```
1  SELECT t.fpk(), t.fdate(), t.fjds(), t.fmois(), t.ftrimestre()
2  FROM t_date t
```

Fondamental : Structure de la zone T

La zone T est donc composée :

- **d'une BDT dont le schéma correspond à un schéma transactionnel représentant l'intégration des différentes sources (il est similaire à celui de la zone E).**
- **d'une API permettant d'accéder aux données (et d'exécuter des transformation)**

Attention

On notera que la zone de transformation copie depuis la zone d'extraction les données en l'état, sans transformation. Donc elle contient des données encore "sales".

En revanche, les données qui sortent de cette zone pour aller vers la zone d'analyse - les données disponibles via l'API - sont traitées et donc "propres".

Méthode : Transformation simple

Les transformations simples, typiquement qui ne nécessitent qu'un seul enregistrement en entrée, seront effectuées directement et dynamiquement par les fonctions de l'API (ou des fonctions appelées par celles-ci).

Méthode : Transformations complexes

Les transformations plus complexes devront être réalisées par des procédures exécutées en batch après le chargement de la BDT.

Elle produiront des données complémentaires stockées dans la BDT.

Méthode : Vues

Les vues doivent être utilisées pour unifier (UNION) ou joindre (JOIN) les différentes tables physiques qui représentent une même donnée logique.

Les vue matérialisées peuvent être utilisées et rafraîchies à chaque mise à jour de la zone T.

Méthode : Gestion des contraintes

Les contraintes de la zone T doivent être compatibles avec les données sources afin de :

- laisser passer 100% des données depuis la zone E (sans contrainte) vers la zone T (avec contrainte) ;
- s'assurer le maximum d'information sur la nature de données

Si les contraintes sont trop relâchées, il faudra faire des vérifications inutiles pour contrôler des données, qui en fait avaient déjà les propriétés souhaitées.

Si les contraintes sont trop fortes, toutes les données ne passeront pas.

Complément

Vue matérialisée sous Oracle 9i

2. Implémentation de la zone T en RO

Méthode

- On crée une table pour chaque vue, table externe et table classique de la zone d'extraction.

  Ces tables sont créées selon la syntaxe SQL3 du modèle relationnel-objet (afin de pouvoir accepter des méthodes).

  On a donc un attribut disponible dans la zone T pour chaque attribut de la zone E.
- On déclare une méthode pour chaque attribut que l'on souhaite exporter dans le modèle dimensionnel.

  Cette méthode permettra de réaliser dynamiquement les transformations et vérifications adéquates.

  On a donc une méthode disponible dans la zone T pour chaque attribut voulu dans le DW.

Rappel : Transformation simples et complexes

Pour les méthodes simples ne portant que sur un enregistrement à la fois, la méthode est attachée à la table correspondante.

Pour les transformations complexes nécessitant un script préalable, la méthode est associée à la table où est stockée le résultat de ce script.

Des vues peuvent être créées pour unifier l'accès aux tables.

Rappel : Contraintes

Pour chaque hypothèse de "propreté" des données sources on pose une contrainte associée.

- Par exemple si une donnée doit être une clé primaire et que l'on pense que les sources sont correctes de ce point de vue, on ajoute la clause PRIMARY KEY.
- Par contre il ne faut pas ajouter les contraintes lorsque l'on sait que les données sources sont "sales", sans quoi ces données seront refusées au chargement et ne pourront jamais être nettoyées.
- Dans le doute, il est parfois utile de donner des tailles de champs plus grandes que celle attendues (par exemple une chaîne de 50 caractères au lieu de 20) ou bien des types plus permissifs (une chaîne au lieu d'une date) afin de ne pas bloquer ou tronquer d'enregistrement.

Le relâchement des contraintes demandera un travail plus important d'implémentation des méthodes.

Méthode : Implémentation des méthodes

- Lorsque les données sources sont "propres" et qu'elles sont copiées telle qu'elle dans la cible, la méthode associée se contente d'un `return` de l'attribut correspondant.
- Lorsqu'un traitement est nécessaire, il est implémenté au sein de la méthode.

Fondamental

Pour être "propre" une donnée doit déjà respecter les contraintes souhaitées dans la zone T (condition nécessaire, non suffisante).

Dans tous les autres cas, la méthode doit effectuer des vérifications et traitements.

Attention: Collecte des statistiques

Il faut re-calculer les statistiques nécessaires à l'optimisation du moteur de requêtes (ANALYSE) comme après tout ajout significatif de données.

Pensez également à rafraîchir les vues matérialisées, puis à exécuter les collectes de statistiques sur ces vues.

Complément

Plans d´exécution sous Oracle 9i

3. Désactivation et réactivation de contraintes

Rappel

Le chargement dans la base dimensionnelle, si les méthodes de transformation ont été correctement écrites, ne comporte que des données valides. De plus ce chargement va impliquer un nombre très important de données. Il est donc souhaitable de désactiver les contraintes sur le modèle dimensionnel pendant le temps de chargement, afin d'accélérer cette procédure.

Notons que si les contraintes ne sont pas désactivées :

- à chaque ajout d'une ligne le moteur de la base va devoir vérifier que cette ligne respecte les contraintes ;
- de plus si les données ne sont pas chargées exactement dans le bon ordre, des contraintes de type intégrité référentielle peuvent être temporairement non validées.

Précisons enfin qu'une fois le chargement terminé les contraintes seront réactivées afin de vérifier que les méthodes de transformation ont fait correctement leur travail et que les données respectent effectivement les contraintes du modèle dimensionnel. Si les nouvelles données ne respectent pas les contraintes, ces dernières ne pourront être réactivées tant que les erreurs n'auront pas été corrigées.

Méthode : Préalable à la réactivation

Oracle fournit un script (`utlexcpt.sql`) pour la création d'une table qui va servir à récupérer les éventuelles erreurs détectées suite à la réactivation des contraintes.

```
1  --  utlexcpt.sql
2  create table exceptions(row_id rowid,
3                  owner varchar2(30),
4                  table_name varchar2(30),
5            constraint varchar2(30));
```

Syntaxe : Désactivation de contraintes

```
1  ALTER TABLE nom_table DISABLE CONSTRAINT nom_contrainte;
```

Remarque : Déclaration des contraintes à la création

On notera qu'il est important pour pouvoir aisément désactiver les contraintes de les avoir explicitement nommées lors de la création des tables.

Syntaxe : Réactivation des contraintes

```
1  ALTER TABLE nom_table ENABLE CONSTRAINT nom_contrainte
2
3   EXCEPTIONS INTO exceptions ;
```

Méthode : Erreurs de réactivation des contraintes

Si les contraintes ne peuvent être réactivées du fait que certaines données ne sont plus conformes, les enregistrements en cause seront référencés (par leur *rowid*) dans la table `exceptions` créée par le script `utlexcpt.sql`. Pour retrouver ces enregistrements, exécuter une requête de sélection dans cette table.

```
1  SELECT * FROM nom_table WHERE rowid IN (SELECT row_id FROM exceptions);
```

Une fois les erreurs corrigées, l'opération de réactivation des contraintes peut être renouvelée.

Complément : Désactivation des contraintes et suppression des index

Un contexte qui nécessite la désactivation des contraintes pour améliorer des performances de chargement nécessitera également la suppression des index, également gourmands en ressources lors de la création ou mise à jour des données. Ces index seront recréés une fois le chargement terminé.

4. Processus de chargement BDE->BDT

Attention: Performance du chargement

Afin d'améliorer les performances au chargement dans une zone, on désactive les contraintes et on supprime les index préalablement au chargement, et on les réactive et recrée postérieurement.

Méthode : Chargement BDE->BDT

1. Désactivation des contraintes de la BDT
2. Suppression des index de la BDT
3. Copie des données de la BDE vers la BDT
4. Recréation des index
5. Réactivation des contraintes
6. Vérification que la réactivation des contraintes n'a pas rejeté de données (100% des données sont passées de la BDE à la BDT)
7. Exécution des procédures de pré-traitement
8. Actualisation des vues matérialisées
9. Signalisation de la disponibilité de la zone T

Attention: Passage obligé

Le passage d'une zone à l'autre doit toujours laisser passer toutes les données.

Lors du passage de la zone d'extraction à la zone de transformation, les contraintes qui bloquent doivent être levées pour laisser passer les données et lors du passage de la zone de transformation à la zone d'exploitation, les méthodes doivent gérer tous les cas de figure problématiques.

F. Projet Fantastic : Mise en place de la zone de traitement

L'objectif est à présent de créer la BDT en RO. Les noms des tables et vues seront préfixés :

- `f_` dans le schéma `bdt`
- ou `f_bdt_` si vous ne disposez que d'un seul schéma pour toutes vos BD.

Question 1

Créez une table RO `f_bdt_catalogue` avec les attributs de la vue catalogue `f_bde_catalogue` et une méthode pour chaque attribut de la dimension `produit`..

Indices :

Les méthodes ne sont pas implémentées pour le moment.

Pensez à déclarer les contraintes et index explicitement pour pouvoir les désactiver plus tard, avant les chargements massifs.

Question 2

Créez une table RO `f_bdt_magasin` destinée à recevoir la jointure des tables associées à `marketing.ods` et `departementsInsee2003.txt`.

Indices :

Les départements ne sont pas identifiés exactement de la même façon dans les deux tables `f_bde_dpt` et `f_bde_marketing`, il n'est donc pas possible des les joindre directement.

Une solution consiste à créer deux vues `f_bdt_dpt` et `f_bdt_marketing` qui vont permettre d'ajuster les valeurs avant la jointure. Par exemple :

- *La vue `f_bdt_dpt` renvoie directement les valeurs de `f_bde_dpt` (on peut éventuellement s'en passer)*
- *La vue `f_bdt_marketing` renvoie des valeurs corrigées qui peuvent être jointes à `f_bdt_dpt`*

D'autres solutions auraient été :

- *De gérer la modification directement dans la requête INSERT vers `f_bdt_magasin`*
- *De faire l'insertion dans `f_bdt_magasin` via un script PL/SQL (qui traite les cas problématiques)*
- *De copier les données de `f_bde_marketing` dans une table `f_bdt_marketing`, d'exécuter un script corrigeant les données, puis de faire le INSERT*
- *...*

Pour traiter le problème des numéros de département qui sont de type 1, 2 ... au lieu de 01, 02... on peut utiliser un CASE dans un SELECT :

```
1  SELECT
2  CASE WHEN TO_NUMBER(dpt)<10 THEN '0'||TO_NUMBER(dpt) ELSE dpt END AS
   dpt,
3  ...
4  FROM f_bde_marketing;
```

Question 3

Créez une table RO `f_bdt_date` avec un seul attribut qui recevra les valeurs d'un `select distinct dat ...` depuis la table externe permettant d'accéder à `data.csv`.

Question 4

Créez une table RO `f_bdt_vente` pour alimenter la table des faits.

Question 5

Écrivez la procédure d'import BDE->BDT en suivant bien les étapes du processus de chargement.

Indices :

Désactivez vos contraintes et index avant le chargement.

Vérifiez votre import (pas de rejet, vérification du nombre de lignes...)

Penser à valider votre transaction (`COMMIT`).

Question 6

Implémentez une première version des méthodes qui ne fera aucun traitement pour le moment : la méthode retourne soit un attribut sans transformation, soit une constante si ce n'est pas possible.

G. Implémentation simplifiée d'un data warehouse avec Oracle

Dans cette partie nous précisons comment implémenter pratiquement la zone L d'un processus ETL **simple**, **ex nihilo**, **batch**, **non incrémental** avec une base Oracle RO✯ (version 9i ou postérieure).

1. Zone L : Loading

Définition : Base de données de chargement ou data warehouse

La zone de chargement est en fait la BD en étoile ou en flocon qui implémente le data warehouse et les data marts.

Elle reçoit une copie des données sous leur forme transformée (depuis la zone T) disponible pour l'exploitation.

Rappel

Cette BD dimensionnelle peut être :

- directement exploitée pour effectuer des requêtes ;
- ou utilisée pour effectuer des exports vers des logiciels d'analyse spécialisés.

2. Implémentation du data warehouse en R

L'implémentation d'un modèle dimensionnel en base relationnelle ne diffère pas dans sa première phase de celle d'un modèle transactionnel.

On privilégiera simplement un nommage explicite des contraintes et index pour en faciliter la manipulation ultérieure (désactivation, optimisation...).

Conseil : Déclaration explicite des contraintes et index

On utilisera typiquement la syntaxe ci-après pour déclarer une relation dotée d'une clé primaire dont on connaît le nom et pour laquelle on connaît l'index sur lequel elle se base. Notons qu'une déclaration classique de clé primaire aurait conduit à créer un index automatiquement, donc plus difficile à gérer (avec un nom généré par Oracle).

```
1  CREATE TABLE table1 (pknum number, ...);
2  CREATE UNIQUE INDEX idx_table1_pknum   ON table1 (pknum);
3  ALTER TABLE table1 ADD CONSTRAINT cstr_table1_pknum PRIMARY KEY (pknum) ;
```

Exemple : Implémentation d´un modèle dimensionnel

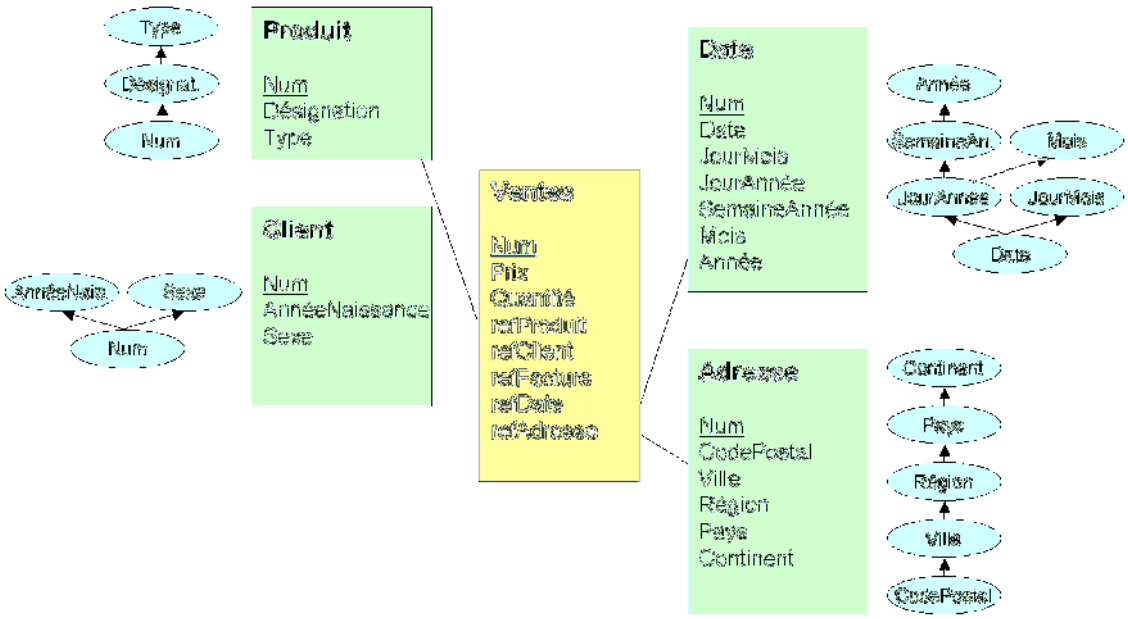


Image 2 Modèle dimensionnel

```
1  CREATE TABLE t_produit (
2    pk_num number,
3    a_designation varchar(50),
4    a_type char(3)
5  );
6
7  CREATE UNIQUE INDEX idx_produit_num
8    ON t_produit (pk_num);
9  ALTER TABLE t_produit
10   ADD CONSTRAINT cstr_produit_num PRIMARY KEY (pk_num)
11   ADD CONSTRAINT cstr_produit_type CHECK (a_type in ('CD', 'DVD'));
12
13 ...
```

3. Processus de chargement BDT->DW

Méthode : Chargement BDT->DW

1. Désactivation des contraintes du DW★ et des DM★
2. Suppression des index du DW et des DM
3. Chargement chaque dimension du data warehouse via l'API de la zone T
4. Chargement la table des faits du data warehouse via l'API de la zone T
5. Recréation des index du DW
6. Réactivation des contraintes du DW
7. Vérification que la réactivation des contraintes n'a pas rejeté de données (100% des données sont passées de la BDT au DW)
8. Extraction des data marts
9. Recréation des index des DM
10. Réactivation des contraintes des DM
11. Extraction des fichiers plats destinés aux applications d'exploitation
12. Signalisation de la disponibilité du DW.

Rappel : Performance du chargement

Afin d'améliorer les performances au chargement dans une zone, on désactive les contraintes et on supprime les index préalablement au chargement, et on les réactive et recrée postérieurement.

Rappel : Passage obligé

Le passage d'une zone à l'autre doit toujours laisser passer **toutes** les données.

H. Projet Fantastic : Mise en place de la zone d'exploitation

L'objectif est maintenant de créer le DW en R. Les noms des tables et vues seront préfixés :

- `f_` dans le schéma `dw`
- ou `f_dw_` si vous ne disposez que d'un seul schéma pour toutes vos BD.

Question 1

Réalisez l'implémentation SQL LDD de votre modèle dimensionnel de DW.

Question 2

Écrivez et testez la procédure d'import BDT->DW.

Question 3

Documentez votre processus ETL complet en effectuant une carte des données.

I. Projet Fantastic : Implémentation des transformations

Afin de finaliser l'ETL, il est à présent nécessaire de terminer l'implémentation des méthodes pour renvoyer les valeurs recherchées.

Question 1

Implémentez les méthodes pour la table des faits.

Indice :

Pour le traitement de la conversion de la date de type `varchar` en type `date` on utilisera la commande `TO_DATE`.

Question 2

Implémentez les méthodes effectuant un simple `return` pour la dimension "magasin" (aucun traitement).

Question 3

Implémentez les méthodes de la dimension "date".

Indice :

- *Utilisez la fonction `TO_DATE` pour obtenir une valeur de type date pour la méthode date( )*
- *Puis utilisez la fonction `TO_CHAR` sur cette méthode pour obtenir les autres attributs*

Fonctions SQL à connaître

Question 4

Implémentez des méthodes effectuant des tests de vérification de format et lorsque c'est nécessaire un reformatage pour la dimension "produit".

Question 5

Exécutez la procédure d'import BDT->DW.

Faites toutes les vérifications nécessaires.

Exploitation mono-dimensionnelle d'un data warehouse en SQL

IV

A. Rappels SQL pour l'étude des données

1. Fichier CSV

Définition : Fichier CSV

CSV ✯ est un format informatique permettant de stocker des données tabulaires dans un fichier texte.

Chaque ligne du fichier correspond à une ligne du tableau. Les valeurs de chaque colonne du tableau sont séparées par un caractère de séparation, en général une **virgule** ou un **point-virgule**. Chaque ligne est terminée par un **caractère de fin de ligne** (*line break*).

Toutes les lignes contiennent **obligatoirement** le même nombre de valeurs (donc le même nombre de caractères de séparation). Les valeurs vides doivent être exprimées par deux caractères de séparation contigus.

La taille du tableau est le nombre de lignes multiplié par le nombre de valeurs dans une ligne.

La première ligne du fichier peut être utilisée pour exprimer le nom des colonnes.

Syntaxe

```
[NomColonne1;NomColonne2;...;NomColonneN]
ValeurColonne1;ValeurColonne2;...;ValeurColonneN
ValeurColonne1;ValeurColonne2;...;ValeurColonneN
...
```

Exemple : Fichier CSV sans entête

```
Pierre;Dupont;20;UTC;NF17
Pierre;Dupont;20;UTC;NF26
Paul;Durand;21;UTC;NF17
Jacques;Dumoulin;21;UTC;NF29
```

Exemple : Fichier CSV avec entête

```
Prenom;Nom;Age;Ecole;UV
Pierre;Dupont;20;UTC;NF17
Pierre;Dupont;20;UTC;NF26
Paul;Durand;21;UTC;NF17
Jacques;Dumoulin;21;UTC;NF29
```

Exemple : Valeur nulle

```
1 Jacques;Dumoulin;;UTC;NF29
```

L'âge est inconnu (NULL).

Attention : Variations...

La syntaxe des fichiers CSV n'est pas complètement standardisée, aussi des variations peuvent exister :

- **Les chaînes de caractères peuvent être protégées par des guillemets (les guillemets s'expriment alors avec un double guillemet).**
- **Le caractère de séparation des nombres décimaux peut être le point ou la virgule (si c'est la virgule, le caractère de séparation doit être différent)**
- **...**

Un des problème les plus importants reste l'encodage des caractères qui n'est pas spécifié dans le fichier et peut donc être source de problèmes, lors de changement d'OS★ typiquement.

Méthode : Usage en base de données

Les fichiers CSV sont très utilisés en BD★ pour échanger les données d'une table (export/import).

Les SGBD★ contiennent généralement des utilitaires permettant d'exporter une table ou un résultat de requête sous la forme d'un fichier CSV, en spécifiant un certain nombre de paramètres (caractère de séparation de valeur, caractère de fin de ligne, présence ou non d'une ligne de définition des noms des colonnes, etc.). De même ils proposent des utilitaires permettant d'importer un fichier CSV dans une table (en spécifiant les mêmes paramètres), voire de créer directement une table à partir du fichier CSV (quand les noms des colonnes sont présents).

Complément : Fichiers à largeur de colonne fixe

Les fichiers à largeur de colonne fixe n'utilisent pas de séparateur de colonne, mais imposent **le même nombre de caractères** pour chaque cellule. L'avantage est de ne pas avoir à spécifier le caractère de séparation, l'inconvénient est la taille de fichier supérieure si les valeurs ne font pas toutes la même largeur.

Complément : XML

Les fichiers XML★ tendent de plus en plus à remplacer les fichiers CSV car ils permettent d'être beaucoup plus expressifs sur le schéma d'origine. Ils sont également plus standards (encodage spécifié, principe de séparation des données par les *tags*, etc.). Leur seul inconvénient est d'être plus verbeux et donc plus volumineux.

Complément : Tables externes

Certains SGBD, comme Oracle, permettent de créer des tables dites **externes**, qui autorisent de créer un schéma de table **directement sur un fichier CSV**, permettant ainsi un accès SQL standard à un fichier CSV, sans nécessité de l'importer d'abord dans une table.

2. Agrégats

Définition : Agrégat

Un agrégat est un partitionnement horizontal d'une table en sous-tables, en fonction des valeurs d'un ou plusieurs attributs de partitionnement, suivi de l'application d'une fonction de calcul à chaque attribut des sous-tables obtenues.

Syntaxe

```
1 SELECT <liste d'attributs de partionnement à projeter et de fonctions de calcul>
2 FROM <liste de relations>
3 WHERE <condition à appliquer avant calcul de l'agrégat>
4 GROUP BY <liste ordonnée d'attributs de partitionnement>
5 HAVING <condition sur les fonctions de calcul>
```

Exemple

```
1  SELECT Societe.Nom, AVG(Personne.Age)
2  FROM Personne, Societe
3  WHERE Personne.NomSoc = Societe.Nom
4  GROUP BY Societe.Nom
5  HAVING Count(Personne.NumSS) > 10
```

Cette requête calcul l'âge moyen du personnel pour chaque société comportant plus de 10 salariés.

Remarque : Restriction

Une restriction peut être appliquée avant calcul de l'agrégat, au niveau de la clause WHERE, portant ainsi sur la relation de départ, mais aussi après calcul de l'agrégat sur les résultats de ce dernier, au niveau de la clause HAVING.

Attention : Projection

Si dans la clause SELECT, un attribut est projeté directement, sans qu'une fonction lui soit appliquée, alors il faut impérativement que cet attribut apparaisse dans la clause GROUP BY (car ce ne peut être qu'un attribut de partitionnement).

Remarque : Fonctions de calcul sans partitionnement

Si une ou plusieurs fonctions de calcul sont appliquées sans partitionnement, le résultat de la requête est un tuple unique.

Remarque : Intérêt de la clause GROUP BY

Pour que l'utilisation de la clause GROUP BY ait un sens, il faut qu'au moins une fonction de calcul soit utilisée, soit dans la clause SELECT, soit dans la clause HAVING.

Remarque : Contrôle imposé par quelques SGBDR

Notons que dans le cas de certains SGBDR★ (par exemple Oracle), l'ensemble des attributs de l'agrégation (clause GROUP BY) doivent être préalablement projetés (donc déclarés dans la clause SELECT).

B. Exploration avec l'agrégation

1. Exploration mono-dimension et mono-niveau avec GROUP BY

Syntaxe

```
1  SELECT d.att, COUNT(*) AS q
2  FROM facts f, dim1 d
3  WHERE f.fk=d.pk
4  GROUP BY d.att
5  ORDER BY q desc;
```

Exemple

```
1  SELECT d.jds, COUNT(*) AS q
2  FROM ventes v, date d
3  WHERE v.dat=d.dat
4  GROUP BY d.jds
5  ORDER BY q desc;
```

2. Isolation de facteur

En créant une vue effectuant une restriction sur dimension on peut mettre à disposition un sous ensemble de la dimension pré-sélectionné selon un premier critère :

- qui ne nous intéresse pas dans certaines analyses ;
- qui risque de compliquer ces mêmes analyses en masquant un facteur.

Exemple

Si un mode de mode d'organisation des produits est dominant, à la fois au sens où il est le plus courant et au sens où il favorise les ventes, on pourra s'intéresser, lorsqu'on étudie d'autres dimensions, uniquement aux produits organisé selon ce mode.

3. Sous-requêtes dans la clause FROM

Conseil

L'enchaînement de sous-requêtes dans la clause FROM sera souvent utile pour calculer des moyennes après regroupement notamment.

Exemple

```
1  SELECT avg(q) FROM
2  (
3  SELECT d.jds AS jds, COUNT(*) AS q
4  FROM ventes v, date d
5  WHERE v.dat=d.dat
6  GROUP BY d.jds
7  )
```

Rappel

Raffinement de questions dans la clause FROM

4. Ajustement des proportions

Exemple

Extrait d´un modèle dimensionnel

```
1  SELECT m.ray AS r1, COUNT(*) AS q1
2  FROM ventes v, mag m
3  WHERE v.mag=m.mag
4  GROUP BY m.ray
```

R1	Q1
Y	9270
A	84142

R1	Q1
E	7812

- Cette requête nous dit a priori que les ventes correspondant au type de rayonnage A sont plus nombreuses que les autres (environ 10 fois).
- Mais si les magasins de type A sont 10 fois plus nombreux que chacun des autres, en fait les ventes par magasin sont du même ordre.
- Il faut donc rapporter le nombre de ventes à la proportion de magasins dans chaque type de rayonnage R1.

Exemple

```
1  SELECT r1 AS Ray, q1 AS Ventes, q2 AS Mag, ROUND(q1/q2) AS VentesParMag FROM
2  (SELECT m.ray AS r1, COUNT(*) AS q1
3  FROM ventes v, mag m
4  WHERE v.mag=m.mag
5  GROUP BY m.ray),
6  (SELECT m.ray AS r2, COUNT(*) AS q2
7  FROM mag m
8  GROUP BY m.ray)
9  WHERE r1=r2
10 ORDER BY VentesParMag DESC;
```

`VentesParMag` nous donne ici le nombre moyen de ventes pour un magasin en fonction de son type.

C. Faciliter l'exploitation avec les vues

1. Usage des vues

Conseil

Les vues sont à utiliser largement en particulier pour :

- Simplifier les requêtes complexes
  - Éviter les sous-requêtes et a fortiori les sous-sous-requêtes
  - Préparer des données (filtrage)
  - ...
- Gérer la récurrence des questions
  - Dès qu'une requête est amenée à être exécutée plusieurs fois
  - Pour les exports CSV récurrents typiquement
  - ...
- Offrir des sources d'accès aux données adaptées aux outils tiers

Complément : Vues matérialisée

Pour des raisons de performance les vues peuvent être matérialisées :

- *Vues concrètes*
- *Vue matérialisée sous Oracle*

2. Isolation de facteur

En créant une vue effectuant une restriction sur dimension on peut mettre à disposition un sous ensemble de la dimension pré-sélectionné selon un premier critère :

- qui ne nous intéresse pas dans certaines analyses ;
- qui risque de compliquer ces mêmes analyses en masquant un facteur.

Exemple

Si un mode de mode d'organisation des produits est dominant, à la fois au sens où il est le plus courant et au sens où il favorise les ventes, on pourra s'intéresser, lorsqu'on étudie d'autres dimensions, uniquement aux produits organisé selon ce mode.

3. Agrégation de faits

En créant une vue projetant un attribut complémentaire dans une dimension, on peut ajouter un nouvel attribut d'analyse, dont la valeur est calculé en fonction des données elles-mêmes.

Exemple

Attribut booléen *best-seller* pour un livre.

Complément

Attributs d´agrégation de faits

D. Projet Fantastic : Exploration avec l'agrégation

Question 1

Effectuez une requête de type GROUP BY pour étudier les ventes en fonction du jour de la semaine.

Question 2

Écrivez une requête avec sous-requête dans la clause FROM pour trouver la moyenne des ventes des jours hors samedi.

Question 3

Transformez la première requête permettant de calculer les ventes en fonction du jour de la semaine en une vue.

Ajouter deux autres vues :

1. la première calculera la moyenne des ventes hors samedi ;
2. la seconde le ratio de ventes le samedi par rapport à la moyenne.

E. Projet Fantastic : Analyse en proportion

Question

Effectuez une requête pour étudier les ventes en fonction du rayonnage.

Effectuez une seconde requête pour étudier la proportion des rayonnages.

Ajuster le résultat en fonction de la proportion de magasins de chaque type de rayonnage.

Indice :

Ray	Ventes	NbMags	VentesParMag
A	415320	?	?
Y	74142	?	?
E	41677	?	?

F. Projet Fantastic : Isolation de facteur

Question

Déduisez des résultats de l'analyse par le rayonnage qu'une isolation de facteur peut être envisagée. Expliquez pourquoi et dans quels cas.

Implémentez la vue correspondante.

G. Projet Fantastic : Agrégation de faits

Question 1

Calculer le nombre moyen de ventes pour un livre.

Question 2

Étudier l'évolution du nombre de ventes (par exemple avec un graphique) et observer une singularité dans l'évolution du nombre de ventes par livre.

Question 3

Créez une vue qui liste les livres selon qu'ils sont des best-sellers ou non (on ajoute un attribut booléen "bs"). Expliciter votre définition d'un best-seller.

H. Projet Fantastic : Exploration de données libre

Question

Explorez votre modèle dimensionnel en mobilisant des agrégations.

L'objectif est de formuler un maximum de conclusions concernant les données disponibles. Proposez un rapport d'exploitation avec résultats d'exploration et états.

Indice :

Rapport d´exploration de données

Modélisation avancée

V

A. Faits

1. Table de faits avec faits et table de faits sans fait

Définition : Table de faits avec faits

En général la table des faits comporte un ou plusieurs attributs représentant des faits, que l'on va sommer sur les dimensions lors de l'analyse.

Exemple : Table des fait avec faits

Par exemple une table des faits représentant des achats de produits pourra contenir la quantité de produits achetés, le chiffre d'affaire de la vente...

`ventes (fk_date, fk_produit, quantite, ca)`

```
1  SELECT sum(v.quantite)
2  FROM ventes v JOIN date d
3  ON fk_date=pk_date
4  GROUP BY d.week
```

Définition : Tables de faits sans fait (factless fact table)

Dans certains cas, on mesure directement dans la table des faits des événements unitaires. Un fait est donc juste un enregistrement dans la table des faits.

Dans ce cas le table des faits ne contient que des clés étrangères, et aucun fait en tant que tel (c'est l'enregistrement qui est le fait).

Attention: Analyse en count

Pour analyser une table de faits sans fait, on ne peut pas utiliser **`sum`** (il n'y a rien à sommer), on utilise **`count`** (on analyse le nombre de faits enregistrés).

Exemple : Tables de faits sans fait

On enregistre ici directement des ventes de produits qui sont toujours vendus à l'unité, en vue d'une analyse en quantité (où le prix de vente n'intervient pas).

`ventes (fk_date, fk_produit)`

```
1  SELECT count(*)
2  FROM ventes v JOIN date d
3  ON fk_date=pk_date
4  GROUP BY d.week
```

Méthode : Ajouter une colonne avec la constante 1

Afin de rendre ce cas plus lisible, il est parfois conseillé d'ajouter une colonne qui contient la valeur 1 pour toutes les lignes. On matérialise ainsi le fait par une valeur, même si elle est toujours la même, et il est de nouveau possible de travailler en somme.

Complément

http://www.kimballgroup.com/1996/09/factless-fact-tables/[3]

2. Clés artificielles

Méthode : Clé artificielles

Every join between dimension and fact tables in the data warehouse should be based on meaningless integer surrogate keys. You should avoid using the natural operational production codes. (Kimball, Ross, 2008, p59)

- Les dimensions doivent être les points entrées dans les faits pour les utilisateurs, donc les clés naturelles n'apporte rien (aucune requête n'est faites directement dans la table des faits, sans jointure)
- L'usage de clés naturelle est plus simple au début, mais plus coûteux sur le long terme : les clés artificielles assurent **l'indépendance aux évolutions futures du système opérationnel** (on rappelle que le data warehouse vise le long terme, au delà de la durée de vie d'une version d'un système opérationnel typiquement)
- Les clés artificielles sont plus performantes (entiers compressés)
- les clés artificielles permettent de gérer les valeurs nulles (date...)
- ...

Méthode : OID

Sous un système relationnel-objet les OID peuvent être utilisés.

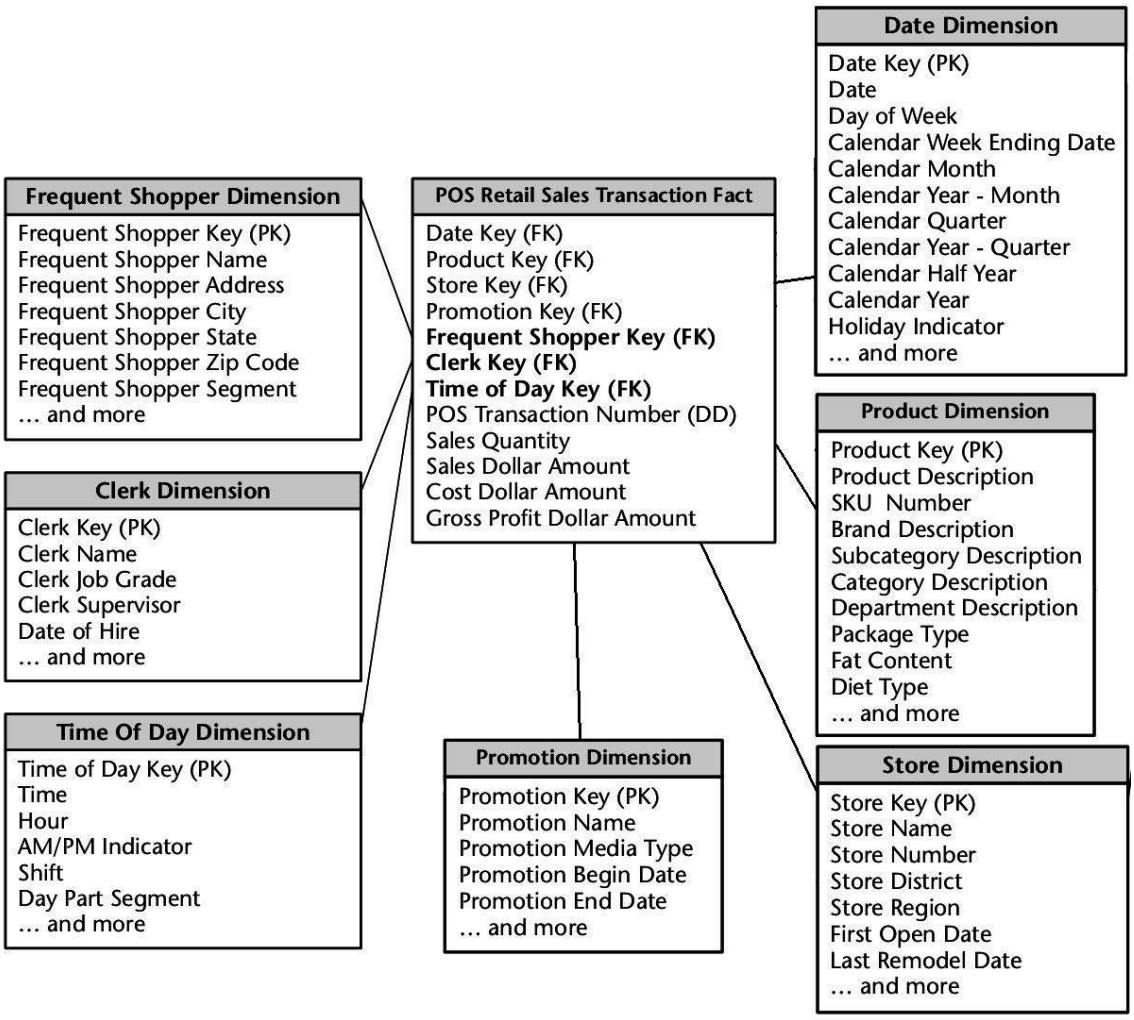
Méthode

La mise en place de clés artificielles complique l'ETL et implique la maintenance d'une table de correspondance par exemple.

3 - http://www.kimballgroup.com/1996/09/factless-fact-tables/

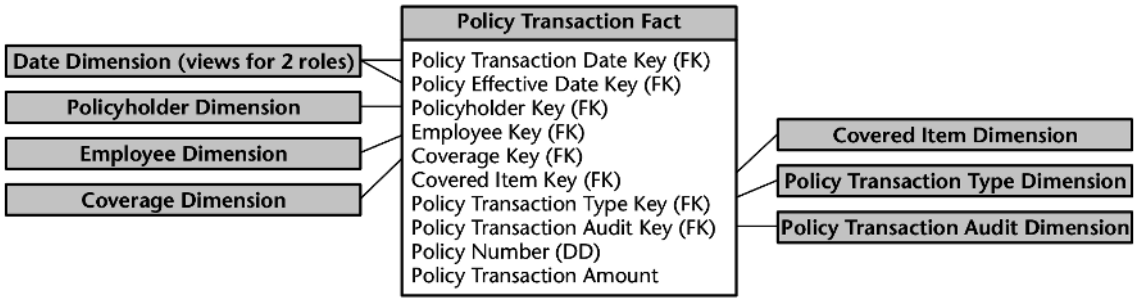
3. Exemples de modèles dimensionnels

Exemple



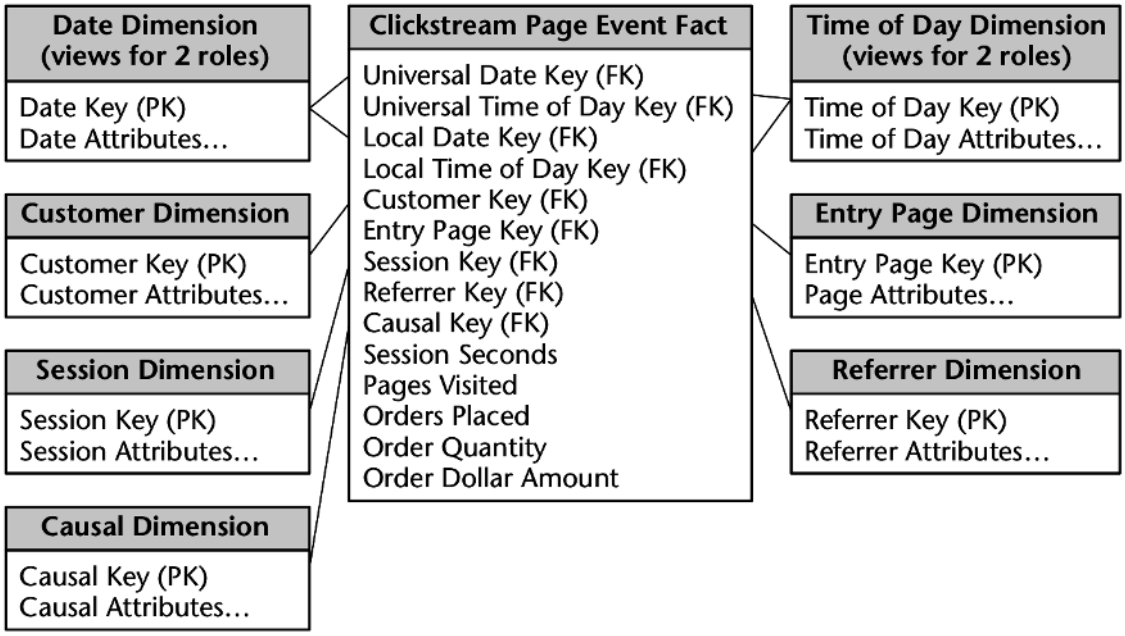
Exemple de modèle dimensionnel d´analyse de ventes (Kimball, Ross, 2008, p.51-53)

Exemple



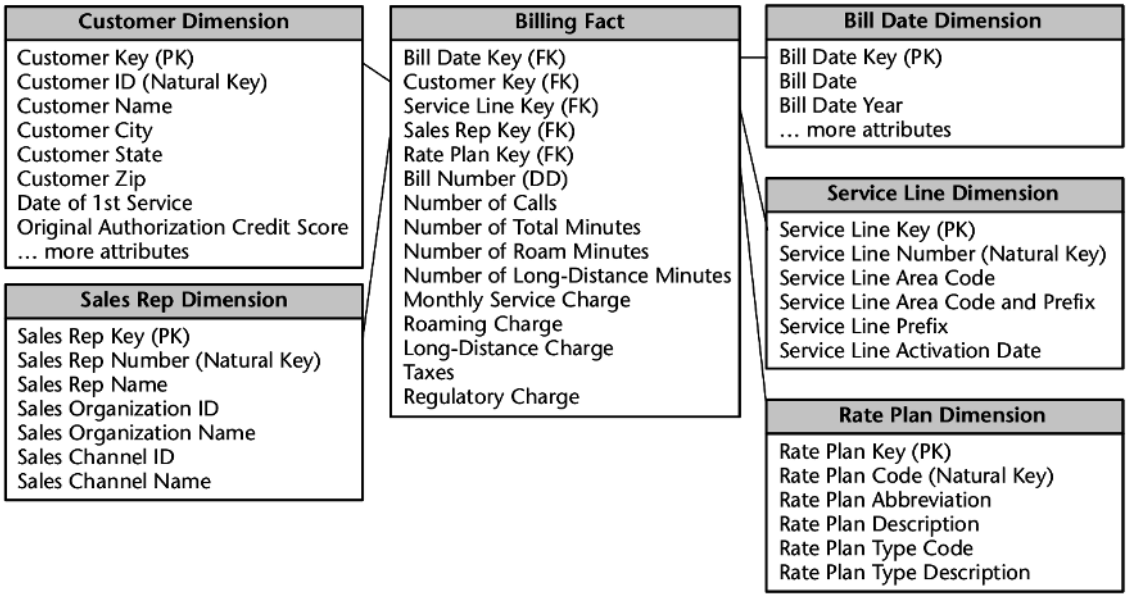
Exemple de modèle dimensionnel dans le domaine de l´assurance (Kimball, Ross, 2008, p.314)

Exemple



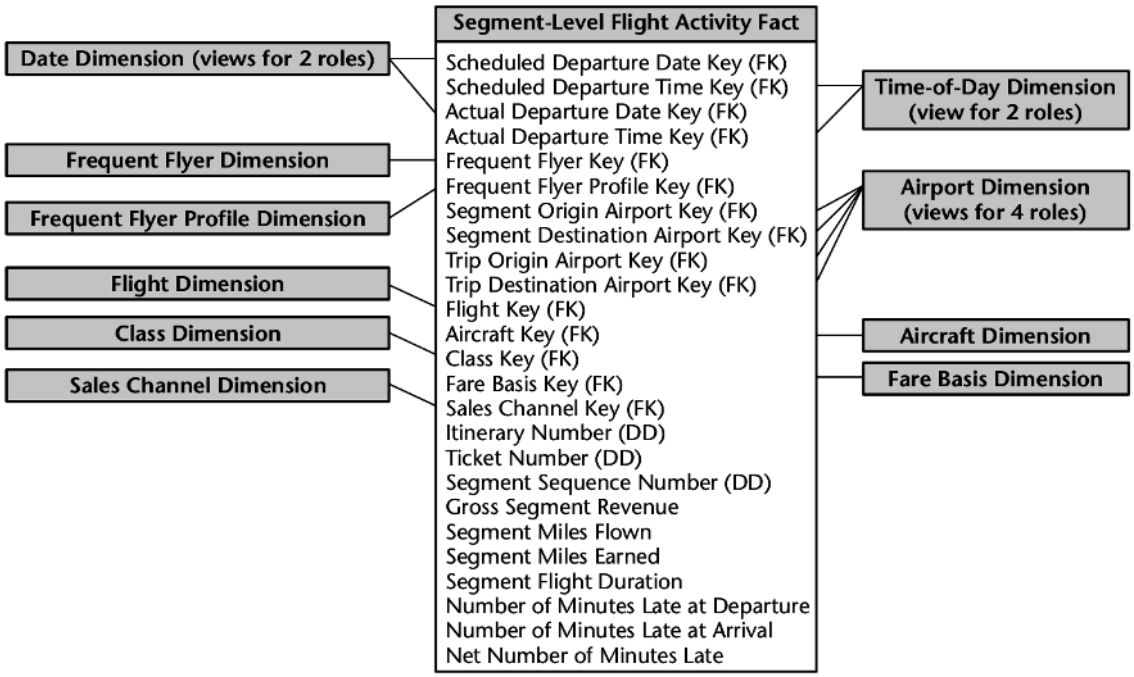
Exemple de modèle dimensionnel dans le domaine du commerce électronique (Kimball, Ross, 2008, p.293)

Exemple



Exemple de modèle dimensionnel dans le domaine des télécommunications (Kimball, Ross, 2008, p.225)

Exemple



Exemple de modèle dimensionnel dans le domaine des transports (Kimball, Ross, 2008, p.233)

4. Gestion des valeurs nulles

Méthode

You must avoid null keys in the fact table. A proper design includes a row in the corresponding dimension table to identify that the dimension is not applicable to the measurement. (Kimball, Ross, 2008, p.49)

Exemple

Lorsqu'un client qui ne possède pas de carte de fidélité achète un produit, il n'est pas possible de lier le fait à un client.

On évite de mettre une valeur nulle en ajoutant une valeur "Client sans carte de fidélité" à la dimension.

5. Gestion des erreurs

Méthode

Il est souvent utile d'ajouter des valeurs dans une dimension afin de gérer les cas d'erreur dans les données.

Exemple

Ces valeurs rendent compte d'une typologie comme par exemple :

- Format invalide
- Valeur tronquée
- Référence inconnue
- ...

Exemple : Gestion des erreurs

FKMag	FKPro	FKDat
12	18	32
12	41	-1
-2	55	21
...	...	...

Tableau 2 Table des faits du DW

PK	Mag	...
-2	"Inconnu"	*null*
-1	"Null"	*null*
...	...	...
12	"M145"	...
13	"M22"	...
...	...	...

Tableau 3 Table de la dimension Mag du DW

PK	Dat	DatStr	...
-1	*null*	"Tronquée"	*null*
...	...	...	...
21	21/1/2013	"21/01/2013"	...
...	...	...	...
32	1/2/2013	"01/02/2013"	...
...	...	...	...

Tableau 4 Table de la dimension Dat du DW

Remarque

Si l'on dispose de suffisamment d'information sur la cause de l'erreur, cette cause peut être utilisée.

Complément : Voir aussi

Gestion des valeurs nulles

6. Faits semi-additifs

Définition : Semiaddictive facts

Un fait est semi-additif s'il est additif sur une partie seulement des dimensions du modèle.

> *All measures that record a static level (inventory levels, financial account balances, ans measures of intensity such as room temperature) are inherently nonadditive across date dimension and possibly other dimensions. In these cases, the measure may be aggregated usefully across time, for example, by averaging over number of time periods.* (Kimball, Ross, 2008, p72)

Méthode

Pour analyser les faits semi-additifs sur les dimensions sur lesquelles ils ne sont pas additifs, il faut faire des **moyennes**.

Exemple

Modèle :

- Dimension compte
- Dimension date
- Fait (fkCompte, fkDate, solde)

Faits :

- (1,1,50)
- (1,2,100)

La somme (150) ne veut rien dire, la moyenne (75) donne bien le solde moyen du compte.

B. Dimensions

1. Conception des dimensions

Conseil

Most business processes can be represented with less than 15 dimensions (Kimball, Ross, 2008, p.58)

Méthode

Il ne faut pas dénormaliser la table des faits :

- les dimensions sont indépendantes entre elles ;
- il ne faut pas représenter des hiérarchies différentes dans des dimensions différentes.

En particulier parce :

- pour des raisons d'intelligibilité ;
- et de performance (la table des faits est la plus volumineuse, elle doit être optimisée)

(Kimball, Ross, 2008, p57) [(Kimball, Ross, 2008)]

2. Dimension dégénérée

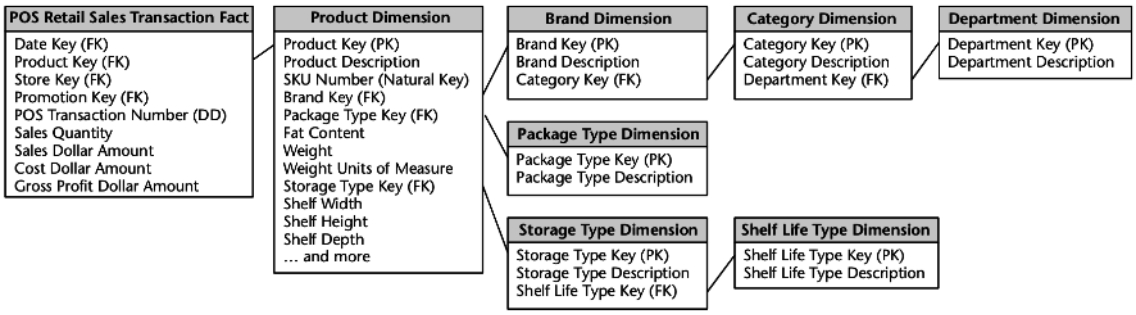
Exemple

Operational control numbers such as order numbers, invoice numbers, and bill-of-lading numbers usually give rise to empty dimensions and are represented as degenerated dimensions (that is, dimension keys without corresponding dimension tables) [...] (Kimball, Ross, 2008, p.50)

3. Modélisation en flocon

Définition

Un modèle en flocon est un modèle pour lequel chaque dimension est représentée avec plusieurs tables. Il est donc plus normalisé (moins redondant) qu'un modèle en étoile.



Exemple de dimension représentée en flocon (Kimball, Ross, 2008, p.55)

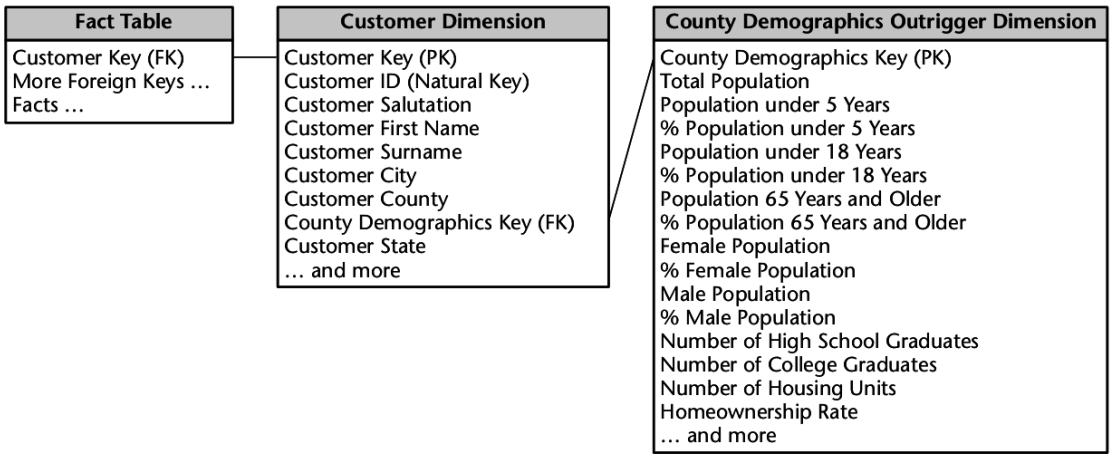
Attention

Les représentation en flocon sont déconseillées en général (Kimball, Ross, 2008, p.56) [(Kimball, Ross, 2008)]:

- **Le modèle en flocon est plus complexe et son appréhension par l'utilisateur difficile**
- **Les performance en requête sont diminuées par les jointures**
- **Le gain en espace disque est faible (les dimensions sont peu volumineuses relativement aux faits)**

Méthode

La normalisation partielle est préconisée lorsqu'il y a des répétitions très nombreuses sur un grand nombre de colonnes (Kimball, Ross, 2008, p.153) [(Kimball, Ross, 2008)].



Exemple de cas pertinent de représentation en flocon d´une dimension (Kimball, Ross, 2008, p.153)

4. Slow Changing Dimension (SCD)

La gestion des changements dans les dimensions est un enjeu de l'historisation dans le data warehouse.

Ces changements sont généralement lent, on parle de SCD.

Méthode

Il y a 5 grands types de solution (Kimball, Ross, 2008, p.95) [(Kimball, Ross, 2008)] :

- Type 1 : Remplacer la valeur (pas de gestion d'historique)
- Type 2 : Ajouter une nouvelle dimension (multiplication du nombre de lignes)
- Type 3 : Ajouter un attribut (gestion d'un seul niveau d'historique)

- Type 3b : Ajouter plusieurs attributs (changements prévisibles)
- Type 6 (1+2+3) : Combiner les type 1, 2 et 3

Exemple

Product Key	Product Description	Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Education	ABC922-Z

Product Key	Product Description	Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Strategy	ABC922-Z

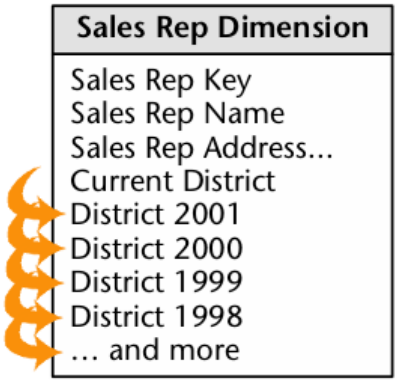
SCD type 1 (Kimball, Ross, 2008, p.96)

Product Key	Product Description	Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Education	ABC922-Z
25984	IntelliKidz 1.0	Strategy	ABC922-Z

SCD type 2 (Kimball, Ross, 2008, p.97)

Product Key	Product Description	Department	Prior Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Strategy	Education	ABC922-Z

SCD type 3 (Kimball, Ross, 2008, p.101)



SCD type 3+ (Kimball, Ross, 2008, p.103)

Product Key	Product Description	Current Department	Historical Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Education	Education	ABC922-Z

Product Key	Product Description	Current Department	Historical Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Strategy	Education	ABC922-Z
25984	IntelliKidz 1.0	Strategy	Strategy	ABC922-Z

Product Key	Product Description	Current Department	Historical Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Critical Thinking	Education	ABC922-Z
25984	IntelliKidz 1.0	Critical Thinking	Strategy	ABC922-Z
31726	IntelliKidz 1.0	Critical Thinking	Critical Thinking	ABC922-Z

SCD type 6 (Kimball, Ross, 2008, p.104)

C. Attributs des dimensions

1. Attributs d'analyse

Définition : Attributs d´analyse

La majorité des attributs d'une dimension qui servent à l'analyse (ils sont mobilisés dans les GROUP BY).

Synonyme : Attribut de regroupement

Syntaxe

Par défaut un attribut mentionné dans le modèle dimensionnel est un attribut d'analyse. Ces attributs sont notés tels quels, sans annotation ni style particulier.

2. Attributs de description

Définition : Attributs de description

Certains attributs ne sont pas utiles à l'analyse, mais peuvent être conservés dans le modèle, afin d'améliorer la qualité des états, souvent parce qu'ils sont plus explicites pour identifier un enregistrement d'une dimension.

Synonyme : Attribut de documentation

Exemple : Numéro et nom de département

Si l'on dispose d'un numéro de département pour l'analyse, le nom peut néanmoins être conservé à des fins d'amélioration des rapports.

Syntaxe

Les attributs de description sont notés en italique dans le modèle dimensionnel et/ou annotés de la mention (d).

3. Attributs de segmentation

Définition

Afin qu'un attribut soit utilisable en analyse, il doit disposer de valeurs discrètes (et non continues) ; et l'échelle de ces valeurs doivent avoir un niveau de précision en adéquation avec les besoins d'analyse.

Lorsque ce n'est pas le cas, l'attribut en question est remplacé par un attribut de segmentation qui projette les valeurs de l'attribut initial dans des segments discrets et utilisables.

Exemple

Population d'un département.

Exemple

- Age
- Revenus

(Kimball, Ross, 2008, p.151) [(Kimball, Ross, 2008)]

Méthode

- Utiliser une segmentation reconnue (habitude socio-économique, pratique de l'organisation...)
- Utiliser une segmentation statistique (parts égales...)

Syntaxe

L'attribut est annoté d'un (s) dans le modèle dimensionnel.

4. Attributs d'agrégation de faits

Définition

Certaines analyses requièrent de regrouper les faits en fonctions de valeurs elles-mêmes issues des faits.

Dans ce cas des attributs d'agrégation des faits sont pré-calculés au sein des dimensions concernées.

Exemple

Les produits les plus vendus (livres best-sellers...)

Exemple

Les clients "high spender" qui plus dépensent le plus.

(Kimball, Ross, 2008, p.152) [(Kimball, Ross, 2008)]

Syntaxe

L'attribut est annoté d'un (a) dans le modèle dimensionnel.

5. La dimension date

Fondamental

Quasiment tous les data warehouses ont une dimension date.

Exemple

Date Dimension
Date Key (PK)
Date
Full Date Description
Day of Week
Day Number in Epoch
Week Number in Epoch
Month Number in Epoch
Day Number in Calendar Month
Day Number in Calendar Year
Day Number in Fiscal Month
Day Number in Fiscal Year
Last Day in Week Indicator
Last Day in Month Indicator
Calendar Week Ending Date
Calendar Week Number in Year
Calendar Month Name
Calendar Month Number in Year
Calendar Year-Month (YYYY-MM)
Calendar Quarter
Calendar Year-Quarter
Calendar Half Year
Calendar Year
Fiscal Week
Fiscal Week Number in Year
Fiscal Month
Fiscal Month Number in Year
Fiscal Year-Month
Fiscal Quarter
Fiscal Year-Quarter
Fiscal Half Year
Fiscal Year
Holiday Indicator
Weekday Indicator
Selling Season
Major Event
SQL Date Stamp
… and more

Exemple de dimension Date (Kimball, Ross, 2008, p.39)

D. Modélisation avancée du data warehouse

[3h]

Afin d'améliorer les analyses du contexte "Fantastic", on va enrichir le modèle dimensionnel et l'implémenter.

Données supplémentaire

Les données supplémentaires suivantes sont apportés au projet :

- fichier `Prices2014.csv` déposé dans un répertoire du serveur `sme-oracle.sme.utc:/home/nf26/fantastic2`
- fichier `Sales2014` déposé dans un répertoire du serveur `sme-oracle.sme.utc:/home/nf26/fantastic2`

Question 1

Améliorer le modèle dimensionnel afin d'ajouter :

- le numéro de ticket(rappeler pourquoi c'est une dimension dégénérée) ;
- le fait quantité (que l'on fixe toujours à 1, ou bien que l'on calcule on regroupant les lignes strictement identiques)
- le fait chiffre d'affaire de la vente que l'on récupère du fichier des prix (on fera l'hypothèse que le prix de vente est toujours le prix enregistré dans ce fichier, on pensera à multiplier le prix par la quantité)
- les attributs apportés par les nouvelles données
- des attributs de documentation (nom du département, genre...)
- des attributs de segmentation (population, age de publication...)
- un attribut d'agrégation pour savoir si un livre est un best-seller ou non

Question 2

Implémenter le modèle dimensionnel et modifier l'ETL en conséquence.

Question 3

Expérimenter le nouveau modèle avec des questions en rapport avec les modifications apportées.

Exploitation multi-hiérarchique et multi-dimensionnelle d'un data warehouse

VI

A. Extensions SQL pour l'exploration de données

Certains SGBDR dont Oracle propose les extensions ROLLUP et CUBE à l'instruction SQL GROUP BY pour faciliter l'exploration de données.

1. Exploration multi-niveaux avec GROUP BY ROLLUP

Syntaxe : GROUP BY ROLLUP

```
1  SELECT ...
2  GROUP BY ROLLUP (a, b, c, ...)
```

GROUP BY ROLLUP (a, b, c, ...) permet de créer tous les sous totaux selon les **attributs ordonnés de groupement** (ici a, b et c) en allant du plus général (a) au plus détaillé (c).

Méthode

Cette clause est typiquement utilisée pour parcourir une hiérarchie.

Exemple

```
1  SELECT d.tri, d.mon, count(*)
2  FROM ventes v, date d
3  WHERE v.dat=d.dat
4  GROUP BY ROLLUP (d.tri, d.mon);
```

1	1	7159
1	2	7630
1	3	9300
1		24089
2	4	7317
2	5	8017
2	6	8977
2		24311
3	7	7397
3	8	8328
3	9	9042
3		24767
4	10	7448
4	11	7764
4	12	12845
4		28057
		101224

Résultat d'exécution GROUP BY ROLLUP (d.tri, d.mon)

Attention: Ordre

L'ordre (a, b, c) est important et doit être du plus gros grain au plus fin.

2. Exploration multi-dimensions avec GROUP BY CUBE

Syntaxe : GROUP BY CUBE

```
1  SELECT ...
2  GROUP BY CUBE (a, b, c, ...)
```

GROUP BY CUBE permet de créer tous les sous totaux possibles pour toutes les combinaisons des attributs de groupement.

Méthode

Cette clause est typiquement utilisée pour faire des **analyses croisées**.

Exemple

```
1  SELECT p.bs, m.bs, count(*)
2  FROM ventes v, f_dw_produit p, f_dw_mag m
3  WHERE v.pro=p.isbn AND v.mag=m.mag
4  GROUP BY CUBE (p.bs, m.bs);
```

Livre BS	Mag BS	Nb Ventes		
		476394		
	0	137368		
	1	339026	71.17%	*Mag BS*
0		324027	68.02%	*Ventes des Livres Non BS en général*
0	0	108736	79.16%	*Ventes des Livres Non BS dans les Mag Non BS*
0	1	215291	63.50%	*Ventes des Livres Non BS dans les Mag BS*
1		152367	31.98%	*Ventes des Livres BS en général*
1	0	28632	20.84%	*Ventes des Livres BS dans les Mag non BS*
1	1	123735	36.50%	*Ventes des Livres BS dans les Mag BS*

Exemple de résultat CUBE analysé dans un tableur

Conseil

Il est souvent nécessaire de récupérer le résultat dans un tableur pour le manipuler et l'interpréter.

Attention

Si l'on projette trop de dimensions dans le CUBE la combinatoire devient grande et les résultats difficiles à interpréter.

B. Rappels Oracle pour l'exploration des données

1. Sous-requêtes dans la clause FROM

Il est possible de raffiner progressivement une requête en enchaînant les questions dans la clause FROM.

Syntaxe

```
1  SELECT ... FROM
2  (SELECT ...
3  FROM ...
4  WHERE ...)
5  WHERE ...
```

Remarque : Sous-sous requête

Il est possible d'imbriquer successivement plusieurs sous-requêtes.

```
1  SELECT ... FROM
2  (SELECT ... FROM
3    (SELECT...
4    FROM ...
5    WHERE)
6  WHERE ...)
7  WHERE ...
```

Syntaxe : Jointure de sous-requêtes

Il est possible de faire le produit ou la jointure de sous-requêtes.

```
1  SELECT ... FROM
2  (SELECT ... FROM
3  WHERE...)
4  (SELECT ... FROM
5  WHERE...)
6  WHERE ...
```

Méthode

Cette extension est particulièrement utile pour les calculs d'agrégat après filtrage ou pour enchaîner les calculs d'agrégat (par exemple pour faire la moyenne de comptage après regroupement).

Exemple : Enchaînement de calculs d´aggrégat

```
1  select avg(c) from (select count(b) as c from t group by a)
```

Exemple : Calcul de la moyenne des ventes par jour de la semaine

```
1  SELECT avg(q) FROM
2  (SELECT jds AS jds, COUNT(*) AS q
3  FROM ventes v
4  GROUP BY jds)
```

2. Fenêtrage des données

Syntaxe : Rownum

```
1  SELECT ... FROM ... WHERE rownum<=N;
2  -- avec N le nombre de lignes désirées.
```

Rownum

La restriction `ROWNUM <= N` dans la clause `WHERE` permet filtrer les N premières lignes de la table.

Remarque

`rownum` est une pseudo colonne qu'il est bien entendu possible de projeter : `SELECT rownum FROM ...`

Syntaxe : Utilisation avancée

```
1  SELECT a1, ..., aN FROM
2  (SELECT a1, ..., aN, rownum AS rnum FROM t)
3  WHERE rnum BETWEEN n1 AND n2
```

Cette syntaxe permet de sélectionner une fenêtre sur les données et pas seulement les N premières lignes.

Méthode : Exploration de données massives

Lorsque l'on est en présence de gros volumes de données, et que l'on veut se faire une idée du contenu de ces données, il n'est **pas souhaitable** de faire un simple SELECT *. En il serait trop long de rapatrier les dizaines de milliers de lignes et de plus cela serait inutile puisque seules quelques unes seraient effectivement lues.

L'usage de `rownum` permet de s'intéresser à des fenêtres de données représentatives, pour se faire une idée générale.

3. SQL*Plus

*Définition : SQL*Plus*

- SQL*Plus est un client Oracle basique en mode texte, qui n'est plus vraiment utilisé (on utilise Oracle SQL Developer à la place).
- SQL*Plus désigne aussi un langage interne à Oracle destiner à gérer la présentation des résultats de requêtes en mode texte (états textuels).

*Complément : SQL*Plus dans SQL Developer*

Oracle SQL Developer utilise également SQL*Plus mais ne supporte pas toutes les fonctions.

http://www.oracle.com/technetwork/developer-tools/sql-developer/sql-worksheet-commands-097146.html[4]

Méthode : Usages

- Le paramétrage de la présentation des résultats de requête est utile au développeur pour avoir des retours lisibles dans son terminal d'exécution.
- Il peut aussi servir à des parties applicatives comme le formatage pour un export CSV.
- ...

Attention

SQL*PLus ne travaille ni sur le contenu ni sur la structure, uniquement sur la présentation.

4 - http://www.oracle.com/technetwork/developer-tools/sql-developer/sql-worksheet-commands-097146.html

a) Variables d'environnement

Syntaxe

SQL*Plus permet de fixer la valeur de variables d'environnement avec la commande :

```
1 SET param valeur
```

Ces paramètres peuvent être lus avec la commande :

```
1 SHOW param
```

Exemple

```
1 SET heading off
```

Permet de désactiver l'affichage des entêtes de colonne dans le résultat affiché.

Complément

http://docs.oracle.com/cd/B19306_01/server.102/b14357/ch12040.htm[5]

b) Fichiers d'entrée et de sortie

Syntaxe : Exécuter un fichier

Pour exécuter un fichier contenant des commandes SQL ou SQL*Plus :

```
1 @path/filename
```

Syntaxe : Sortie dans un fichier

Pour enregistrer les résultats d'une exécution de requêtes dans un fichier :

```
1 SPOOL path/filename
2 -- requêtes dont on veut récupérer les résultats dans le fichier
3 SPOOL OFF
```

Complément

*Création d´un fichier CSV avec SQL*Plus*

c) Formattage d'une colonne de requête

Syntaxe

```
1 COLUMN nom_colonne FORMAT format
```

- Largeur de la colonne : An
- Chiffre (avec ou sans zéro à gauche) : 9 / 0
- Symboles monétaires : $ / L
- Séparateurs de virgule et de milliers : . / ,
- ...

Exemple

```
1 COLUMN ename FORMAT A15
2 COLUMN sal FORMAT $99,990.00
3 COLUMN mgr FORMAT 999999999
```

5 - http://docs.oracle.com/cd/B19306_01/server.102/b14357/ch12040.htm

C. Projet Fantastic : Exploitation multi-dimensionnelle de données

Question 1

Effectuez une requête multi-niveaux permettant de calculer les ventes sur la hiérarchie date-mois-trimestre.

Question 2

Étudiez l'éventuelle influence des magasins (certains sont-ils plus performants ?) et de l'implantation dans les départements (certains sont-ils plus propices ?).

Indice :

Afin de ne pas être dépendant de l´organisation des magasins, effectuez la requête uniquement pour les magasins de type ´A´ avec rayon BS (ce sont à la fois les magasins les plus nombreux et les plus performants).

Isolation de facteur

Question 3

Effectuer une requête multi-dimensionnelle permettant de regarder si les magasins avec un rayon best-sellers vendent plus de livres best-sellers ou non.

On peut utiliser un tableur pour finaliser les analyses de ratio sur les valeurs renvoyées par la requête CUBE obtenue.

Indice :

Agrégation de faits

Datamarts orientés analyse de panier

VII

A. Analyse de panier

1. Définition de l'analyse de panier

Définition

L'analyse de panier (*market basket analysis*) consiste à étudier des ventes en fonction de la structure du panier de l'acheteur.

Un cas typique est l'étude des produits qui sont vendus ensemble.

Synonyme : Analyse de ticket de caisse.

Méthode

L'utilisation du data warehouse en l'état est souvent mal adapté à ces analyse, il faut donc construire des data marts spécifiquement orientés vers ces questions.

2. Analyse de structure de panier

Exemple

Soit le DW :

Graphique 4 Vue intégrée

Soit la question suivante :

« Quelle est l'influence de l'organisation des magasins (rayonnage et rayon best-seller) sur la structure des tickets de caisse, en fonction des dates ? Par exemple les clients achètent-ils plusieurs livres du même auteur, du même éditeur, du même genre ? »

```
1  IndiceProduit, IndiceAuteur, IndiceEditeur, IndiceGenre
2   / magasin (rayonnage, bestseller)
3   / date
```

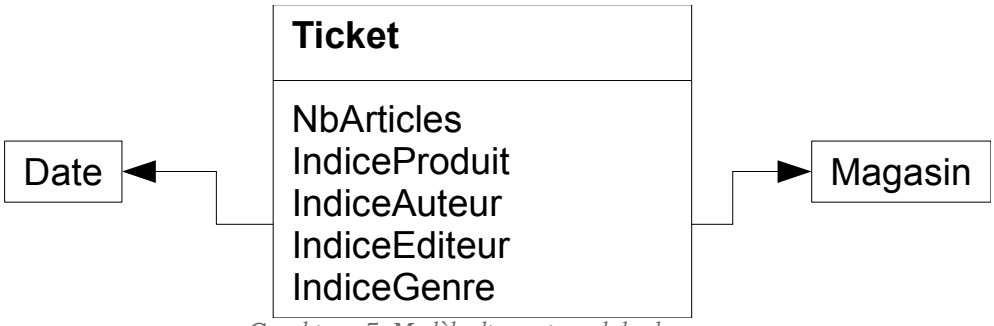
L'indice est un indicateur à calculer rendant compte du nombre de livres achetés ensemble du même auteur, éditeur ou genre :

- `indiceProduit=0.5` signifie que 50% des livres achetés au sein du ticket sont diférents.
- `indiceAuteur=0.5` signifie que 50% des livres différents achetés au sein du ticket sont du même auteur.
- ...

Méthode

Cette requête interroge la structure du ticket de caisse et non le volume des ventes, elle sera traitée par un data mart particulier. En effet elle serait trop complexe à résoudre avec le data warehouse commun, elle nécessite une pré-agrégation.

Exemple



Graphique 5 Modèle dimensionnel du data mart

NbArticles	Entier	Nombre de produits dans le ticket
IndiceProduit	0..1	Nombre de produits différents / Nombre de produits
IndiceAuteur	0..1	Nombre d'auteurs différents / Nombre de produits différents
IndiceEditeur	0..1	Nombre d'éditeurs différents / Nombre de produits différents
IndiceGenre	0..1	Nombre de genre différents / Nombre de produits différents

Tableau 5 Description de la table des faits du data mart

Attention

Les faits ne sont pas additifs et seront analysés en moyenne.

Méthode : Pré-agrégation des faits

Date	NumTicket	RefProduit	...
1	1	1	
1	1	1	
1	1	2	
1	2	1	
1	3	5	
1	3	10	
...	...	...	

Tableau 6 Table de faits du data warehouse (une ligne par article vendu)

Date	NumTicket	...	NbArticles	IndiceProduit	IndiceAuteur	...
1	1		3	0.67	0.5	
1	2		1	1	1	
1	3		2	1	1	
...	...					

Tableau 7 Table de faits pré-agrégée du data mart (une ligne par ticket)

Attention

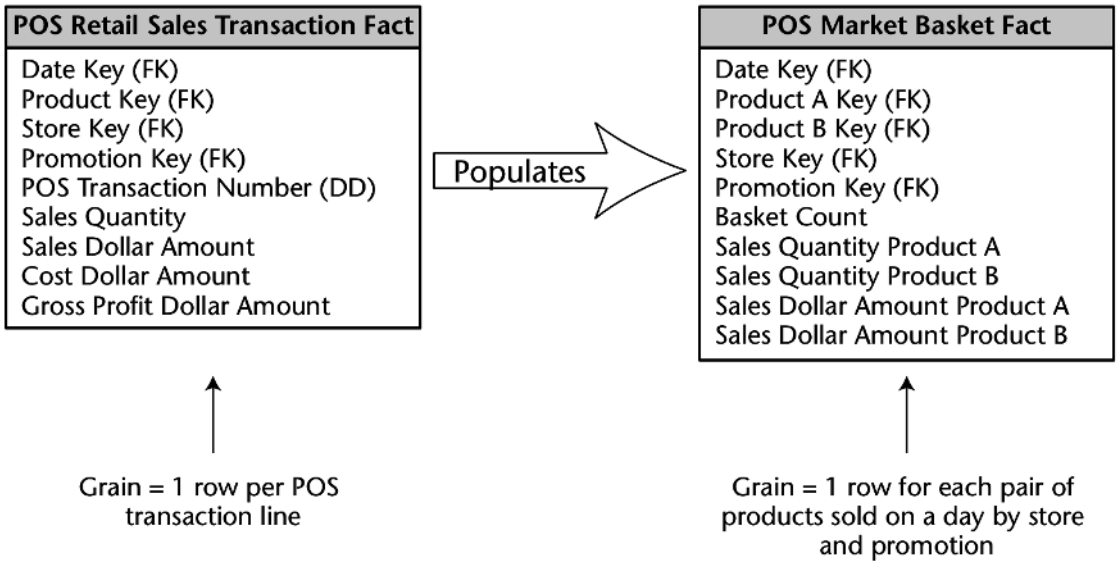
On extrait les informations supplémentaires (indices) de la dimension produit, qui disparaît.

3. Analyse de ventes conjointes

Méthode

Pour savoir quel produit est vendu avec quel produit, on construit une nouvelle table des faits avec un fait pour chaque couple de produits existants (ou le sous ensemble des produits effectivement vendus ensemble) (Kimball, Ross, 2008, p.62) [(Kimball, Ross, 2008)].

Exemple



Exemple de table de faits pour l'analyse de panier (Kimball, Ross, 2008, p.63)

Attention

Il y a potentiellement (N x N-1) combinaisons de produits, ce qui peut conduire à un très grand nombre de faits.

En général l'analyse est effectuée :

- **sur une sélection préalable d'un sous-ensemble des produits : les plus vendus, des combinaisons que l'on sait vouloir étudier, des sous-catégories...**
- **via une approche progressive par produit : on analyse les produits vendus avec X, puis Y en fonction des résultats...**

B. Data mart pour l'analyse de ticket de caisse

[2h]

Afin d'analyser la structure des tickets de caisse, l'on va créer un data mart dédié.

Question 1

Réaliser un data mart pour l'analyse de structure de tickets de caisse. On se contentera d'une analyse en quantité (sans intégrer le chiffre d'affaire donc).

Question 2

Exploiter le data mart pour analyser globalement les comportements d'achats multiples en fonction des rayonnages, des dates...

Question 3

Enrichissez le data mart avec les indicateurs adéquats pour analyser si le rayonnage influe sur la structure du ticket de caisse.

Indice :

Typiquement on va chercher à étudier si :

- *les rayonnages de type A ou E induisent des tickets de caisse avec plusieurs livres du même auteur ou du même éditeur ;*
- *les rayons BS induisent des tickets avec plusieurs livres de type BS*

Compléments

VIII

A. Éléments avancés pour l'ETL

1. Gestion des erreurs

En cas d'erreur, il y a trois approches possibles :

1. Arrêt du chargement, traitement de l'erreur (amélioration des traitements) et reprise du chargement
   Dans un processus incrémental, c'est en général une mauvaise solution
2. Rejeter la donnée dans une table d'erreur
3. Laisser passer la donnée en prenant une décision par défaut et *logger* pour vérification

Complément

(Kimball et al., 2008, p.382-384) [(Kimball et al., 2008)]

Remarque

Les approches 2 et 3 ne pose pas de problème si :

- les problèmes sont très minoritaires
- répartis sur la population des faits
- traités au fur et à mesure

Méthode : Rejeter

- Créer une copie de la structure du DW pour accueillir les données rejetées
- Ajouter un espace de stockage des commentaires (raison du rejet...)

Méthode : Laisser passer

- Adopter une approche permettant de laisser systématiquement passer les données (par exemple en joutant des valeurs d'erreur dans les dimensions)
- Logger dans une table ad hoc les cas traités par défaut

2. Clés artificielles

Méthode

Pour introduire des clés artificielles pour identifier les dimensions :

- Dans la zone de transformation :
  - ajouter à chaque dimension un attribut `pk`
  - ajouter une méthode `getPk()` qui renvoie `pk`
  - (supprimer la méthode qui renvoyait initialement la clé naturelle si elle n'est plus nécessaire par ailleurs)
- À chaque ajout d'un nouvel enregistrement dans une dimension
  - générer une clé artificielle dans `pk` (à l'aide d'une séquence par exemple)

- Lors de l'ajout des dimensions dans le data warehouse
  - utiliser `getPk()` pour identifier les enregistrements
- Lors de l'ajout des faits, il faut substituer les clés
  - joindre les faits avec les dimensions
  - appeler les méthodes `getPk()` de chaque dimension

Exemple

```
1  INSERT INTO dw_facts
2  SELECT d1.pk(), d2.pk()
3  FROM t_fact f, t_dim1 d1, t_dim2 d2
4  WHERE f.a=d1.a AND f.b=d1.b
```

Attention

Il faut mémoriser la correspondance entre la clé identifiant la dimension dans le système transactionnel et la clé artificielle dans le data warehouse.

Pour cela :

- **soit les tables sont persistantes dans la zone T**
- **soit les clés du système transactionnel doivent être conservées dans le DW**

Attention : Optimisation

- **Les jointures à présent nécessaires diminuent les performances de chargement**
- **Une indexation adéquate est requise**

Rappel

Les faits n'ont pas besoin d'être identifiés.

Complément : OID

Sous Oracle en RO, il est possible d'utiliser les OID à la place de clés artificielles.

Le problème sera que les OID ne sont connus qu'après insertion dans la DB, ils ne peuvent être créés dans la BDT puis transféré dans le DW.

- Il faut déclarer les tables du DW en mode RO, en conservant les clés d'origine
- Puis substituer les clés étrangères par des REF dans la table des faits, en faisant **la jointure entre la table des faits de la BDT avec les dimensions du DW**

3. Éléments pour l'ETL incrémental

Dans un DW incrémental le système est mis à jour régulièrement.

Une mise à jour se manifeste par :

- l'ajout de nouveaux faits
- l'éventuel ajout ou mise à jour des dimensions

Ajout de faits

Deux cas de figures :

- La source des faits est remplacée à chaque incrément (les nouveaux faits remplacent les anciens)
  - il faut ajouter les nouveaux faits au fur et à mesure en relançant l'ETL, typiquement :
    i. vider les faits de la BDT (anciens faits)
    ii. repeupler les faits de la BDT (BDE->BDT)
    iii. exécuter le transfert BDT->DW
- Les nouveaux faits sont ajoutés à la même source à chaque incrément :
  - il faut vider et repeupler le DW à chaque incrément
  - ou il faut une méthode qui permette de discriminer les faits déjà intégrés des nouveaux (et l'on se rapporte au cas précédent)

Méthode

Vider la BDT sera de préférence la dernière étape de l'ETL (assimilant le transfert BDT->DW à un déplacement)

1. BDE->BDT
2. BDT->DW
3. Vider BDT

Attention

Si la BDT permet de calculer des attributs d'agrégation de faits, il est nécessaire qu'elle conserve l'ensemble des données pour effectuer ses calculs.

Dans ce cas, plutôt que de vider la BDT on utilisera un attribut de discrimination (*flag*) qui mémorisera les données déjà transférées des nouvelles données.

Ajout de dimensions

L'ajout de dimensions résulte de :

- l'ajout de nouveaux faits,
- associé éventuellement à l'ajout des dimensions dans les sources transactionnelles.

De la même façon :

- soit l'on supprime et récrée la dimension
- soit l'on gère l'ajout des nouveaux enregistrements après les avoir discriminés

Mise à jour de dimensions

La mise à jour de dimension résulte de :

- l'ajout de nouveaux faits,
- associé à la mise à jour des sources transactionnelles des dimensions.

Il faut adopter une stratégie de gestion des SCD.

Méthode : Audit continu des données

Utiliser des triggers pour auditer les données et ainsi affiner les méthodes, trouver des erreurs...

4. Intégration des dimensions multi-sources

Méthode

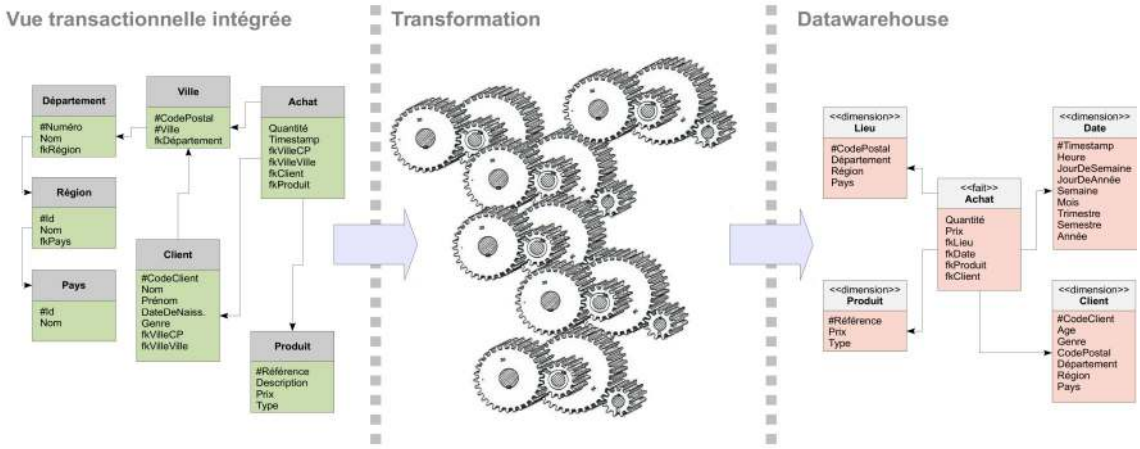
Lorsque les dimensions ont plusieurs sources, il faut :

- Contrôler les doublons (dé-duplication)
- Conformer les différentes sources

5. Performance et maintenance

Méthode

Il est possible de substituer cette architecture à 3 niveaux par une architecture à 2 niveaux seulement.



Architecture générale d'un ETL

B. Extensions Oracle pour l'exploration de données

À partir de sa version 9i, Oracle propose un certain nombre d'extensions au LMD SQL pour faciliter l'exploration de données. Ces extensions sont orientées vers la fouille de données sur un modèle dimensionnel plutôt que vers les requêtes classiques sur un modèle transactionnel ((Abbey01) [Abbey01]).

1. Classements

Syntaxe : Projection de classement

```
1  SELECT RANK() OVER (PARTITION BY a ORDER BY b DESC), ...
2   FROM ...
3  GROUP BY a ...
```

RANK() OVER (PARTITION BY a ORDER BY b DESC) permet de projeter le classement de l'enregistrement par rapport au critère "b" (qui peut être un agrégat), partitionné par le critère "a".

PARTITION BY est optionnel.

On notera que l'attribut de partitionnement du classement "a" est forcément un attribut de regroupement, il doit donc être déclaré dans le GROUP BY.

Exemple

```
1  SELECT * FROM
2  (
3  SELECT p.titre AS titre, count(*) AS ventes, RANK() OVER (ORDER BY count(*) DESC)
    AS rank
4  FROM ventes v, produit p
5  WHERE p.isbn=v.pro
6  GROUP BY p.titre
7  )
8  WHERE rank <= 100;
```

Remarque : Classement sur plusieurs expressions

Il est possible de réaliser des classements sur plusieurs expressions, en spécifiant plusieurs critères dans la clause ORDER BY (traités alors de droite à gauche).

Syntaxe : Extrapolation de classement

```
1  SELECT RANK(n) WITHIN GROUP (ORDER BY b) FROM ...
```

RANK(n) WITHIN GROUP (ORDER BY b) permet de projeter le classement d'un enregistrement hypothétique dont la valeur de "b" serait "n".

Complément : Autres fonctions de classement

On notera qu'il existe d'autres fonctions de classement proposées par Oracle 9i :

- DENSE_RANK
- PERCENT_RANK
- CUME_DIST

2. Totaux cumulés

Syntaxe : Total cumulé

```
1  SELECT SUM(SUM(a)) OVER (ORDER BY b ROWS UNBOUNDED PRECEDING)
2  FROM ...
3  GROUP BY b ...
```

`SUM(SUM(a)) OVER (ORDER BY b ROWS UNBOUNDED PRECEDING)` permet de calculer la somme cumulée des "a" selon l'évolution "b".

On notera que "b" doit être un attribut de regroupement.

Remarque

`SUM(COUNT(a))` est également valide.

Exemple

```
1  SELECT d.sem, SUM(COUNT(*)) OVER (ORDER BY d.sem ROWS UNBOUNDED PRECEDING)
2  FROM ventes v, date d
3  WHERE v.dat=d.dat
4  GROUP BY d.sem;
```

Complément

Notons qu'il est possible de "fenêtrer" les données, c'est à dire de définir une plage de valeurs bornée, en spécifiant une fenêtre de valeurs à l'aide de la clause "RANGE min max" à la place de "ROWS UNBOUNDED PRECEDING".

Le fenêtrage de données est particulièrement utile dans le cas de données temporelles (cf. Oracle 9i : Notions fondamentales [Abbey01]).

3. Création d'un fichier CSV avec SQL*Plus

Il est parfois utile de transférer le résultat d'une requête vers un fichier CSV✯ pour l'exploiter dans un contexte extérieur à Oracle. SQL*Plus est une solution simple pour y parvenir (Oracle 9i [Abbey01], p.213).

Méthode

Pour transférer un résultat de requête dans un fichier CSV, il faut procéder en deux étapes.

1. Écrire un script qui :
   - formate la sortie de façon adéquate (SET) ;
   - oriente le résultat vers un fichier (SPOOL).
   - encadre les données non numériques par des guillemets et sépare les colonnes par des points-virgules (||) ;
2. Appeler ce script pour obtenir le fichier CSV.

La sortie standard du client pourra être désactivée si le flux de données est important.

Exemple

```
1  -- script /tmp/exportCsv.sql
2  SET HEADING OFF
3  SET FEEDBACK OFF
4  SET ECHO OFF
5  SET PAGESIZE 0
6  SPOOL /tmp/out.dat
7  SELECT '"' ||  pkFamilleDefaut || '";"' || libelle || '"' FROM tFamilleDefaut ;
8  SPOOL OFF
```

```
1  -- exécution du script /tmp/exportCsv.sql
2  @/tmp/exportCsv.sql
```

Attention

Il est indispensable de procéder en deux temps : l'exécution directe de `/tmp/exportCsv.sql` ne donnera pas un formatage correct.

Rappel

Fichier CSV

4. Exemple général d'analyse de données sous Oracle

a) Modèle dimensionnel

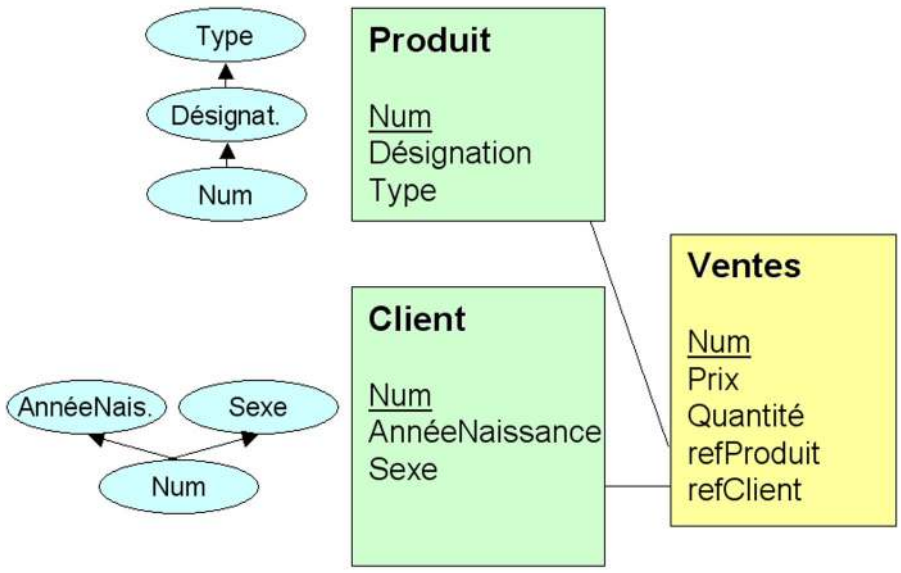


Image 3 Modèle dimensionnel

```
1  CREATE TABLE t_produit (
2   pk_num number,
3   a_designation varchar(50),
4   a_type char(3)
5  );
6  CREATE UNIQUE INDEX idx_produit_num
7   ON t_produit (pk_num);
8  ALTER TABLE t_produit
9   ADD CONSTRAINT cstr_produit_num PRIMARY KEY (pk_num)
10  ADD CONSTRAINT cstr_produit_type CHECK (a_type in ('CD', 'DVD'));
11 CREATE TABLE t_client (
12  pk_num number,
13  a_anneenaiss number,
```

```
 a_sexe char(1)
);
CREATE UNIQUE INDEX idx_client_num
 ON t_client (pk_num);
ALTER TABLE t_client
 ADD CONSTRAINT cstr_client_num PRIMARY KEY (pk_num)
 ADD CONSTRAINT cstr_client_sexe CHECK (a_sexe in ('M', 'F'));
CREATE TABLE t_ventes (
 pk_num number,
 a_prix number,
 a_qte number,
 fk_produit number,
 fk_client number
);
CREATE UNIQUE INDEX idx_ventes_num
 ON t_ventes (pk_num);
ALTER TABLE t_ventes
 ADD CONSTRAINT cstr_ventes_num PRIMARY KEY (pk_num)
 ADD CONSTRAINT cstr_ventes_produit FOREIGN KEY (fk_produit) REFERENCES
t_produit(pk_num)
 ADD CONSTRAINT cstr_ventes_client FOREIGN KEY (fk_client) REFERENCES
t_client(pk_num);
```

```
INSERT INTO t_produit VALUES (1, 'Pink Martini', 'CD');
INSERT INTO t_produit VALUES (2, 'Souad Massi', 'CD');
INSERT INTO t_produit VALUES (3, 'Souad Massi', 'DVD');
INSERT INTO t_produit VALUES (4, 'Raul Paz', 'CD');
INSERT INTO t_produit VALUES (5, 'Star wars', 'DVD');
INSERT INTO t_produit VALUES (6, 'Star wars BO', 'CD');

INSERT INTO t_client VALUES (1, 1980, 'M');
INSERT INTO t_client VALUES (2, 1980, 'M');
INSERT INTO t_client VALUES (3, 1970, 'F');
INSERT INTO t_client VALUES (4, 1970, 'M');
INSERT INTO t_client VALUES (5, 1985, 'F');

INSERT INTO t_ventes VALUES (1, 15, 1, 1, 1);
INSERT INTO t_ventes VALUES (2, 20, 3, 1, 2);
INSERT INTO t_ventes VALUES (3, 30, 1, 1, 3);
INSERT INTO t_ventes VALUES (4, 10, 2, 2, 1);
INSERT INTO t_ventes VALUES (5, 2, 5, 2, 1);
INSERT INTO t_ventes VALUES (6, 10, 1, 3, 1);
INSERT INTO t_ventes VALUES (7, 20, 1, 4, 2);
INSERT INTO t_ventes VALUES (8, 30, 1, 4, 3);
INSERT INTO t_ventes VALUES (9, 10, 1, 5, 3);
INSERT INTO t_ventes VALUES (10, 40, 4, 6, 4);
INSERT INTO t_ventes VALUES (11, 30, 1, 6, 5);
INSERT INTO t_ventes VALUES (12, 10, 100, 6, 5);
```

PK_NUM	A_DESIGNATION	A_TYPE
1	Pink Martini	CD
2	Souad Massi	CD
3	Souad Massi	DVD
4	Raul Paz	CD
5	Star wars	DVD
6	Star wars BO	CD

Tableau 8 T_PRODUIT

PK_NUM	A_ANNEENAISS	A_SEXE
1	1980	M
2	1980	M
3	1970	F
4	1970	M
5	1985	F

Tableau 9 T_CLIENT

PK_NUM	A_PRIX	A_QTE	FK_PRODUIT	FK_CLIENT
1	15	1	1	1
2	20	3	1	2
3	30	1	1	3
4	10	2	2	1
5	2	5	2	1
6	10	1	3	1
7	20	1	4	2
8	30	1	4	3
9	10	1	5	3
10	40	4	6	4
11	30	1	6	5
12	10	100	6	5

Tableau 10 T_VENTES

PK_NUM	A_PRIX	A_QTE	FK_PRODUIT	FK_CLIENT	A_DESIGNATION	A_TYPE	A_ANNEE	A_SEXE	*CA*
1	15	1	1	1	Pink Martini	CD	1980	M	*15*
2	20	3	1	2	Pink Martini	CD	1980	M	*60*
3	30	1	1	3	Pink Martini	CD	1970	F	*30*
4	10	2	2	1	Souad Massi	CD	1980	M	*20*
5	2	5	2	1	Souad Massi	CD	1980	M	*10*
6	10	1	3	1	Souad Massi	DVD	1980	M	*10*
7	20	1	4	2	Raul Paz	CD	1980	M	*20*
8	30	1	4	3	Raul Paz	CD	1970	F	*30*
9	10	1	5	3	Star wars	DVD	1970	F	*10*
10	40	4	6	4	Star wars BO	CD	1970	M	*160*
11	30	1	6	5	Star wars BO	CD	1985	F	*30*
12	10	100	6	5	Star wars BO	CD	1985	F	*1000*

Tableau 11 Jointure des 3 tables et CA

b) ROLLUP

Exemple

```
1  SELECT SUM(v.a_prix * v.a_qte) as CA, p.pk_num as P, p.a_type as T
2  FROM t_ventes v, t_produit p
3  WHERE v.fk_produit=p.pk_num
4  GROUP BY ROLLUP (p.a_type, p.pk_num);
```

```
1          __CA          P __T
2         _105          1 _CD
3          __30          2 _CD
4          __50          4 _CD
5        1190          6 _CD
6        1375 _          _CD
7         __10          3 DVD
8          __10          5 DVD
9          __20      _       DVD
10       1395 _  ___
```

La requête permet de calculer les chiffres d'affaire (quantité multipliée par le prix de vente) selon la dimension "produit", c'est à dire pour chaque produit, mais aussi pour chaque type de produit (granularité plus grossière dans la hiérarchie de la dimension produit).

Comme pour un GROUP BY classique, la première ligne nous donne le chiffre d'affaire du produit 1, la seconde celui du produit 2, etc. Mais la cinquième ligne (une fois tous les produits de type CD traités) propose une consolidation et donne le chiffre d'affaire pour tous les produits de type CD. De même la huitième ligne donne le chiffre d'affaire pour tous les produits de type DVD et enfin la dernière ligne donne le chiffre d'affaire global.

c) CUBE

Exemple

```
1  SELECT SUM(v.a_prix * v.a_qte) as CA, p.a_type as T, c.a_sexe as S
2  FROM t_ventes v, t_produit p, t_client c
3  WHERE v.fk_produit=p.pk_num AND v.fk_client=c.pk_num
4  GROUP BY CUBE (p.a_type, c.a_sexe);
```

```
1  __CA __T   S
2        1395 ___ _
3        1100    ___  F
4         _295   ___  M
5        1375 _CD _
6        1090 _CD  F
7         _285 _CD  M
8          __20 DVD _
9          __10 DVD F
10         __10 DVD M
```

La requête permet de calculer le "cube" des chiffres d'affaire selon les types de produit et le sexe des clients.

La première ligne renvoie le chiffre d'affaire global, la seconde celui pour les sexe=F (donc les femmes) seulement, la troisième pour les sexe=M seulement, la quatrième pour les type=CD, la cinquième pour les type=CD et sexe=F (CD achetés par des femmes), etc.

d) RANK

Exemple

```
1  SELECT * FROM (
2  SELECT
3   RANK() OVER (
4    PARTITION  BY p.a_type
5    ORDER by sum(v.a_qte) DESC
6   ) as R,
7   SUM(v.a_qte) as V,
8   p.a_type as T,
9   p.a_designation as P
10 FROM t_ventes v, t_produit p
11 WHERE v.fk_produit=p.pk_num
12 GROUP BY p.a_type, p.a_designation
13 )
14 WHERE R<=3;
```

```
1        R           __V __T   P
2            1         105 _CD  Star wars BO
3            2           __7 _CD  Souad Massi
4            3           __5 _CD  Pink Martini
5            1           __1 DVD Star wars
6            1           __1 DVD Souad Massi
```

Cette requête renvoie le "top 3" des produits vendus (en quantité) pour les CD et pour les DVD. On notera que pour les DVD, étant donné qu'il n'y a eu que deux ventes et chacune d'une unité, ces deux ventes apparaissent premières ex-aequo et il n'y a pas de troisième vente.

e) RANK(N)

Exemple

```
1  SELECT
2   p.a_type as P,
3   RANK(3) WITHIN GROUP
4    (ORDER BY v.a_qte DESC) as HypotheticalRank
5  FROM t_ventes v, t_produit p
6  WHERE v.fk_produit=p.pk_num
7  GROUP BY p.a_type;
```

```
1  __P   HYPOTHETICALRANK
2  _CD                 4
3  DVD                 1
```

Cette requête permet de se demander combien serait classée une vente de trois unités dans le domaine des CD ainsi que dans celui des DVD. On voit qu'une telle vente serait la quatrième meilleure vente pour des CD et la première pour des DVD.

f) SUM(SUM(...))

Exemple

```
1  SELECT
2   sum(v.a_prix * v.a_qte) as CA,
3   c.a_anneenaiss as D,
4   sum(sum(v.a_prix * v.a_qte))
5    OVER
6    (ORDER BY c.a_anneenaiss ROWS UNBOUNDED PRECEDING)
7    as CA_cumul
8  FROM t_ventes v, t_client c
9  WHERE v.fk_client=c.pk_num
10 GROUP BY c.a_anneenaiss;
```

```
1          __CA           ___D   CA_CUMUL
2                _230       1970        _230
3                _135       1980        _365
4               1030       1985       1395
```

Cette requête permet de faire le calcul cumulé des chiffres d'affaire sur les années de naissance des clients. Le résultat montre ainsi que les clients nés en 1970 ont permis un chiffre d'affaire de 230, ceux de 1980 et moins de 365 et ceux de 1985 et moins de 1395 (le total).

C. Utilisation d'un tableur pour l'exploitation de données

Cette introduction à l'utilisation de tableur sera illustrée avec OpenOffice.org Calc.

1. Reporting

Méthode : Ouvrir un fichier CSV

- Ouvrir le fichier CSV avec OOo Calc
- Suivre l'assistant pour paramétrer l'import

Méthode : Créer un graphique

Sélectionner les données à utiliser pour le graphique, puis exécuter : `insert > chart`

2. Tableaux croisés

Définition : tableau croisé

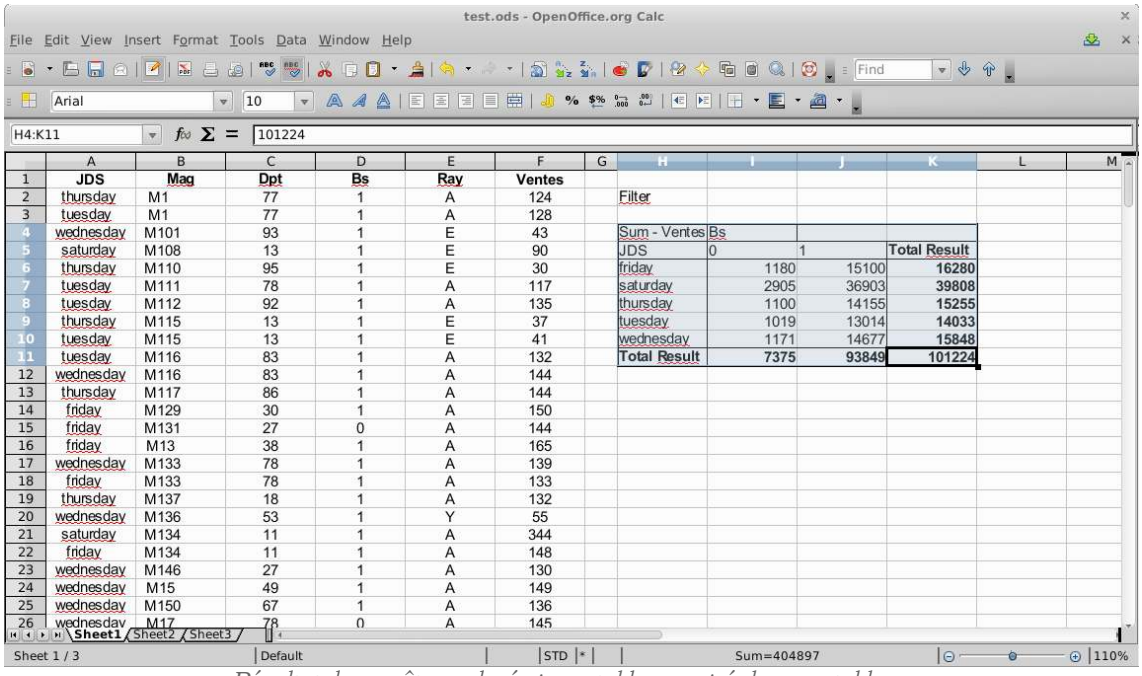
Un tableau croisé dans un tableur est une analyse multi-dimensionnelle de données qui peut être paramétrée dynamiquement.

Méthode

1. Importer les données sources
2. Sélectionner les données sources
3. Exécuter `data > datapilot > start`
4. Glisser-déposer les paramètres d'étude et les valeurs à quantifier
5. Filtrer éventuellement les valeurs souhaitées pour les paramètres d'analyse
6. Compléter éventuellement manuellement le résultat (calcul de pourcentage, graphique...)

Exemple : Un tableau croisé pour analyser des ventes en fonction des jours de la semaine, des magasins, des départements, des rayonnages

```
1  SELECT d.jds, m.mag, m.dpt, m.bs, m.ray, count(*)
2  FROM ventes v, date d, mag m
3  WHERE v.dat=d.dat AND v.mag=m.mag
4  GROUP BY d.jds, m.mag, m.dpt, m.bs, m.ray
```



	JDS	Mag	Dpt	Bs	Ray	Ventes
2	thursday	M1	77	1	A	124
3	tuesday	M1	77	1	A	128
4	wednesday	M101	93	1	E	43
5	saturday	M108	13	1	E	90
6	thursday	M110	95	1	E	30
7	tuesday	M111	78	1	A	117
8	tuesday	M112	92	1	A	135
9	thursday	M115	13	1	E	37
10	tuesday	M115	13	1	E	41
11	tuesday	M116	83	1	A	132
12	wednesday	M116	83	1	A	144
13	thursday	M117	86	1	A	144
14	friday	M129	30	1	A	150
15	friday	M131	27	0	A	144
16	friday	M13	38	1	A	165
17	wednesday	M133	78	1	A	139
18	friday	M133	78	1	A	133
19	thursday	M137	18	1	A	132
20	wednesday	M136	53	1	Y	55
21	saturday	M134	11	1	A	344
22	friday	M134	11	1	A	148
23	wednesday	M146	27	1	A	130
24	wednesday	M15	49	1	A	149
25	wednesday	M150	67	1	A	136
26	wednesday	M17	78	0	A	145

Sum - Ventes	Bs		
JDS	0	1	Total Result
friday	1180	15100	16280
saturday	2905	36903	39808
thursday	1100	14155	15255
tuesday	1019	13014	14033
wednesday	1171	14677	15848
Total Result	7375	93849	101224

Résultat de requête analysé via un tableau croisé dans un tableau

Ici les paramètres d'étude choisis sont JDS (le jour de la semaine) et BS (le fait que le magasin possède ou non un rayon best-sellers).

Rappels

IX

A. Prise en main de Oracle SQL Developer

1. Installation de SQL Developer

Site Web

http://www.oracle.com/technology/products/database/sql_developer[6]
(Téléchargement pour Windows, Mac et Linux)

Complément

Documentation en ligne disponible à la même adresse.

2. Connexion avec SQL Developer

Créer une connexion à la base de données Oracle de l'UTC

1. Clic droit sur `Connexions`, puis `Nouvelle connexion`
2. Entrez vos paramètres :
   - Nom libre
   - Username / Password
   - Hôte : **sme-oracle.sme.utc**
   - SID : **nf26**

6 - http://www.oracle.com/technology/products/database/sql_developer

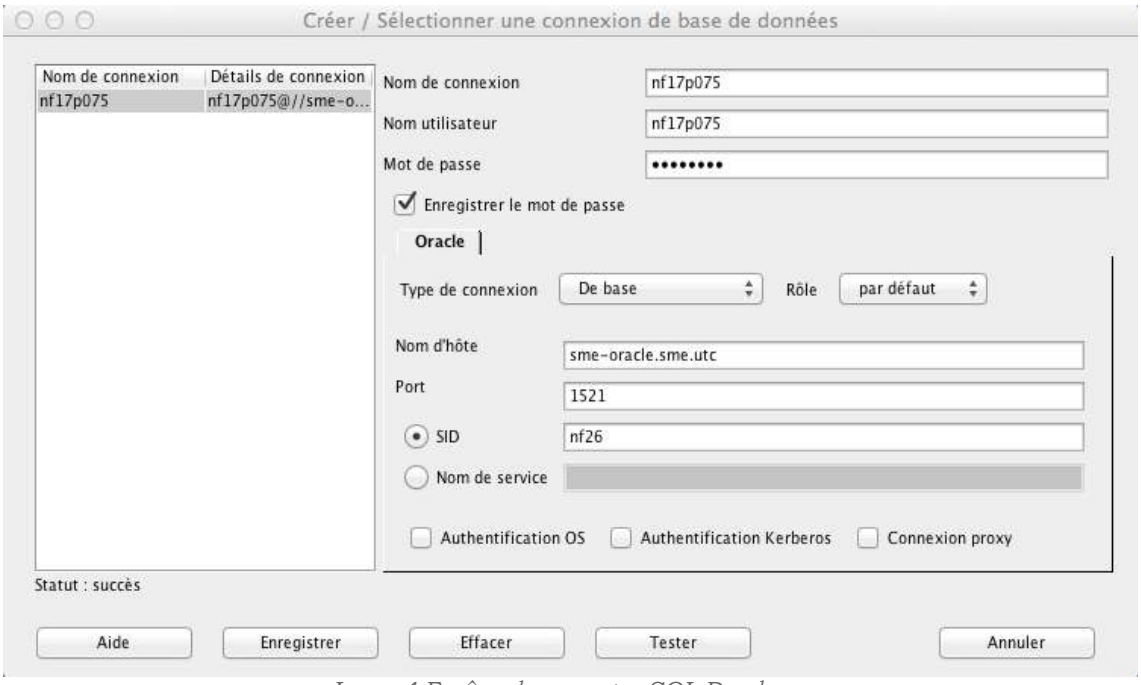


Image 4 Fenêtre de connexion SQL Developer

3. Naviguer dans le catalogue de SQL Developer

L'espace de gauche permet de naviguer dans le catalogue de la base de données et donc de visualiser les tables, vues, index, etc.

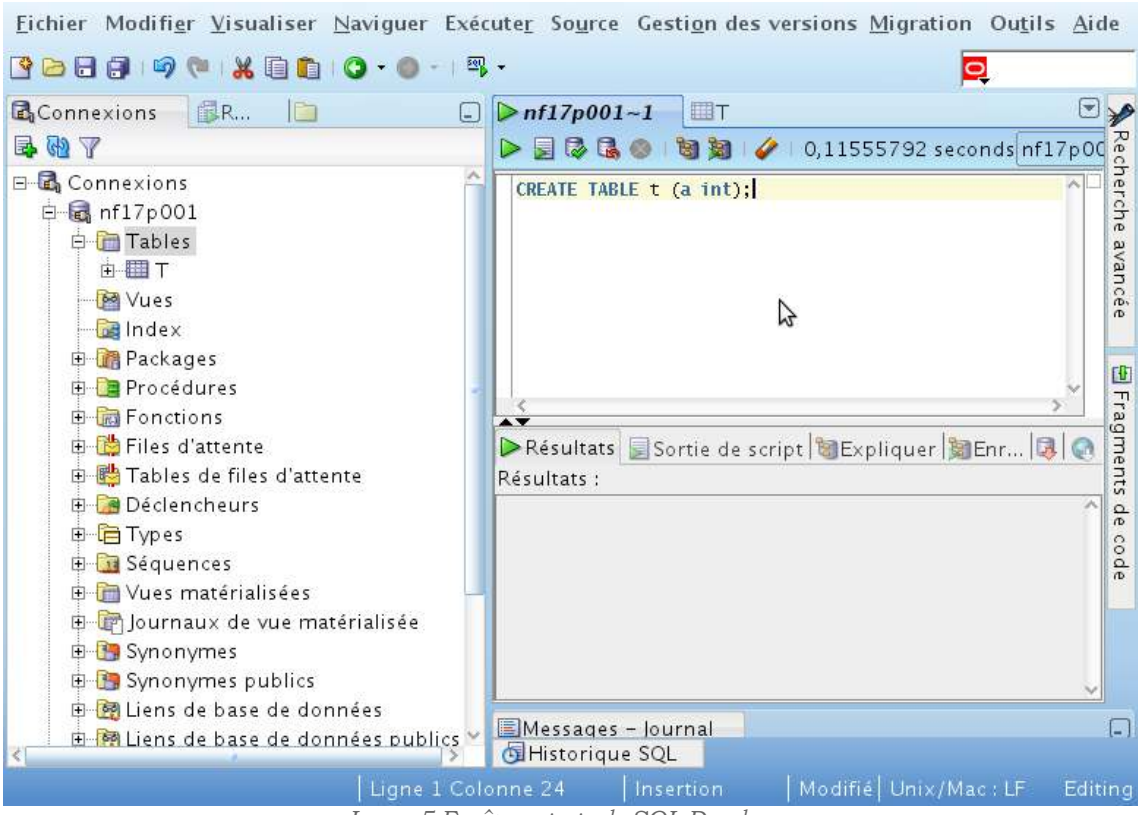


Image 5 Fenêtre principale SQL Developer

Attention

Pour rafraîchir la vue de gauche (catalogue) après une requête LDD, il faut faire un clic droit sur l'élément du catalogue (par exemple `Tables` après une création de table) puis sélectionner `Régénérer`.

4. Exécuter des requêtes SQL avec SQL Developer

L'espace de droite permet d'écrire des requêtes SQL (en haut) et de visualiser le résultat de l'exécution (en bas). Appuyer sur le bouton `Exécuter un script` ou faire `F5` pour exécuter les requêtes saisies.

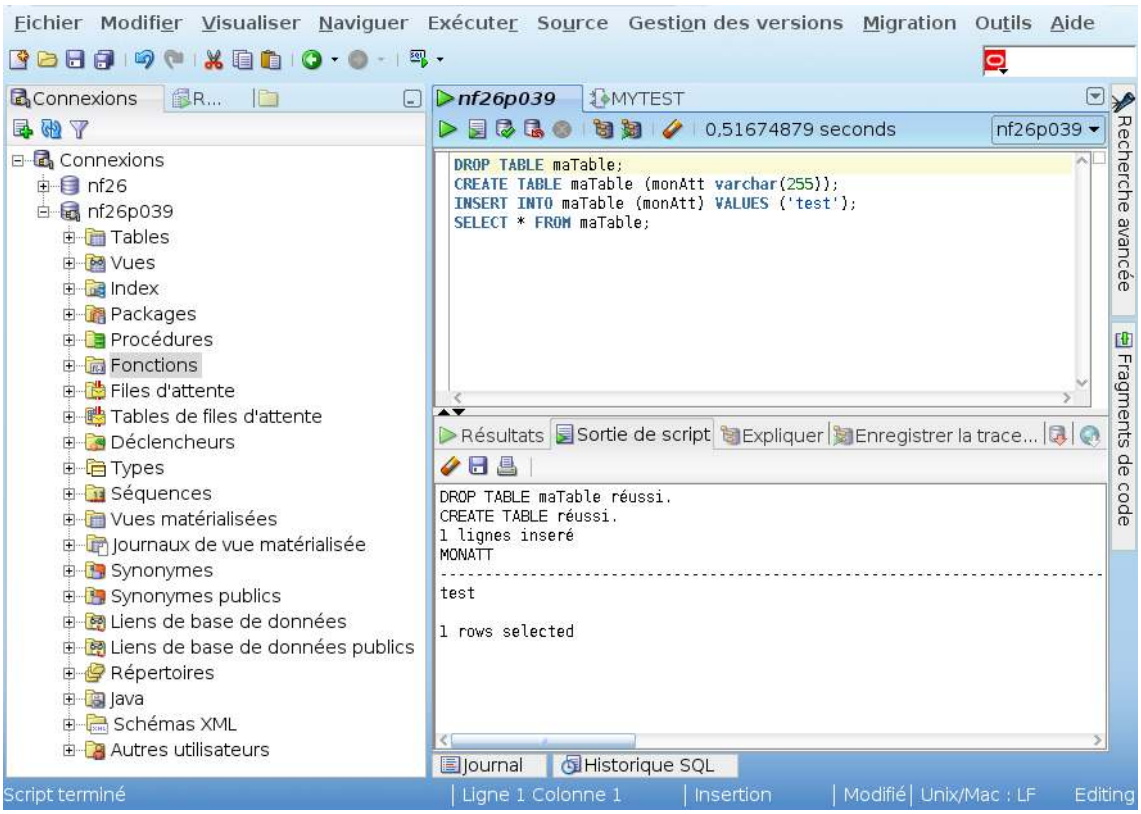


Image 6 SQL (script)

Remarque

Pour effacer les résultats d'exécution précédents, cliquer sur `Effacer`.

Exécuter une seule requête au sein d'un script

Pour n'exécuter qu'une seule requête parmi celle saisies dans la zone du haut, la **sélectionner**, puis cliquer sur `Exécuter un script` ou faire `F5`.

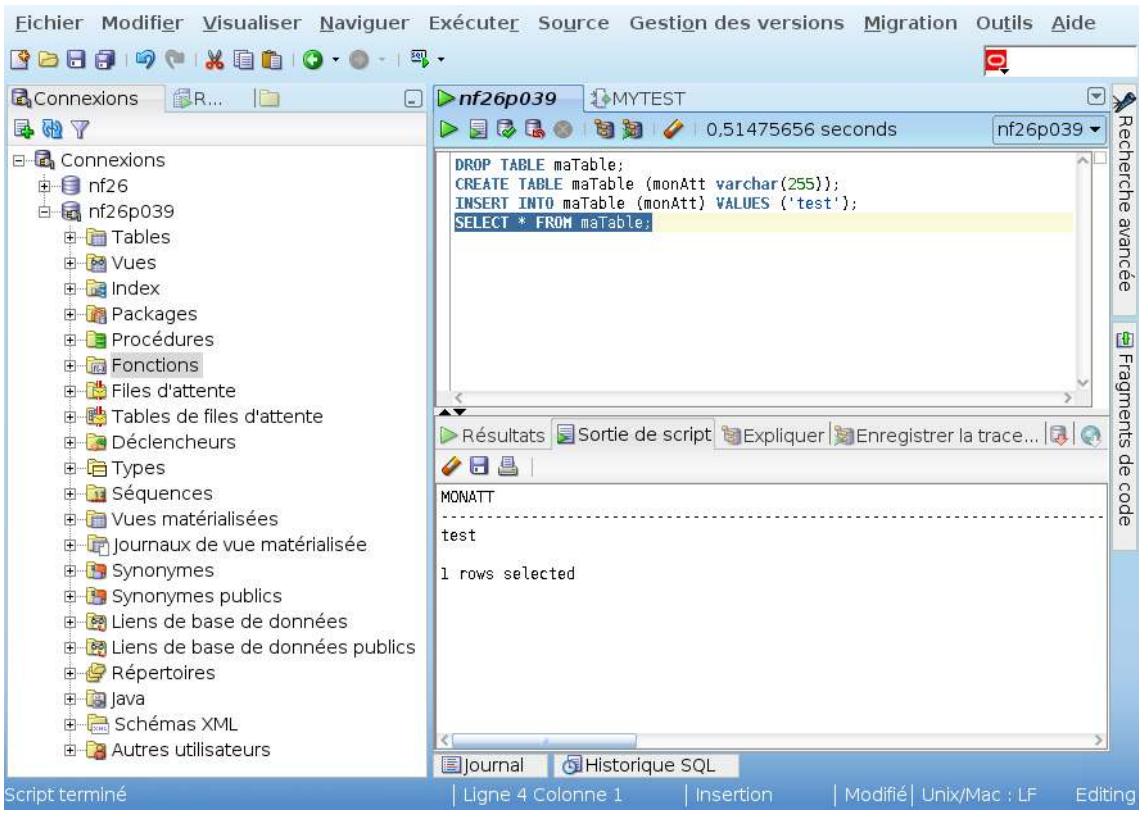


Image 7 SQL (instruction par instruction)

Exécuter une seule requête SELECT

Pour les requêtes SELECT, SQL Developer propose un affichage sous forme de tableau, plus lisible que l'affichage texte. Pour exécuter une requête SELECT et obtenir un tel affichage, cliquer sur `Exécuter l'instruction` ou faire `F9`.

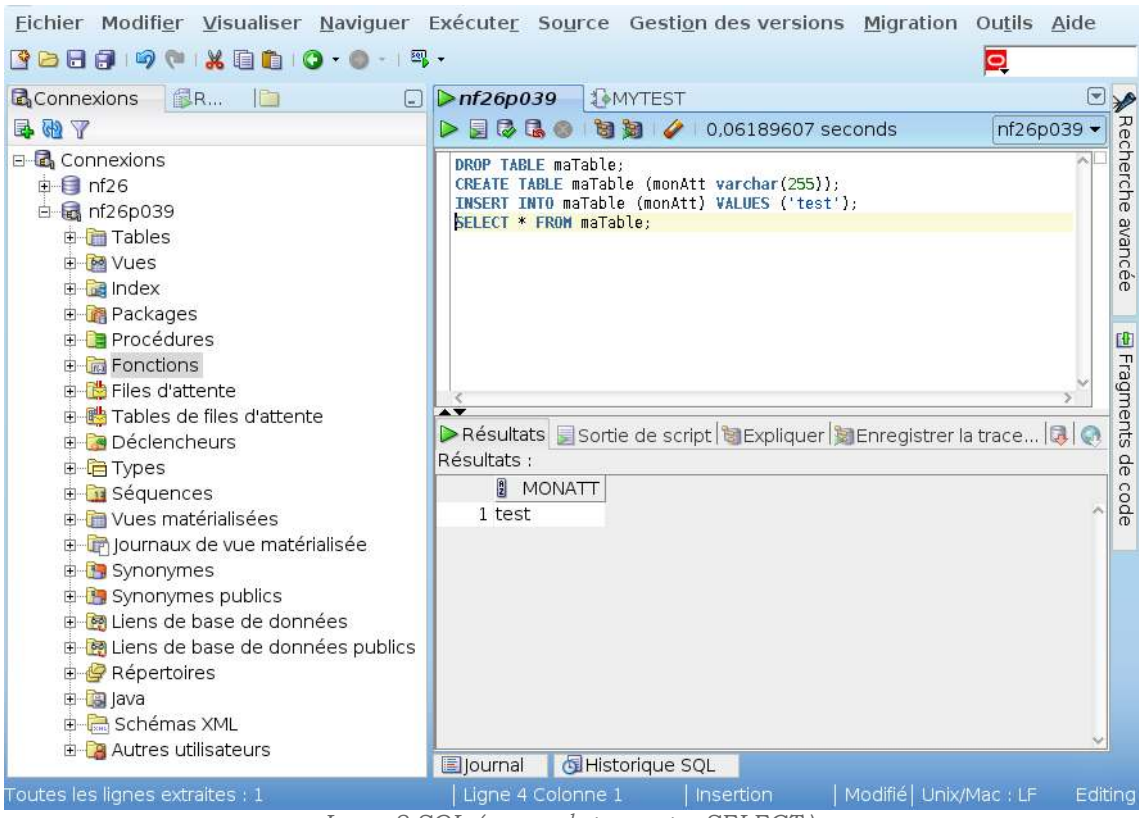


Image 8 SQL (une seule instruction SELECT)

Attention

Le bouton **`Exécuter l'instruction`** n'affiche pas les erreurs d'exécution ou les confirmations de création ou insertion (requêtes CREATE, INSERT, UPDATE), il est donc à réserver aux requêtes SELECT valides (si le résultat n'est pas correct, utiliser **`Exécuter un script`**).

Conseil

Dans le doute utilisez toujours F5 et jamais F9.

5. Écrire du PL/SQL avec SQL Developer

Pour créer des blocs PL/SQL, procéder comme pour le SQL, avec le bouton `Exécuter un script`.

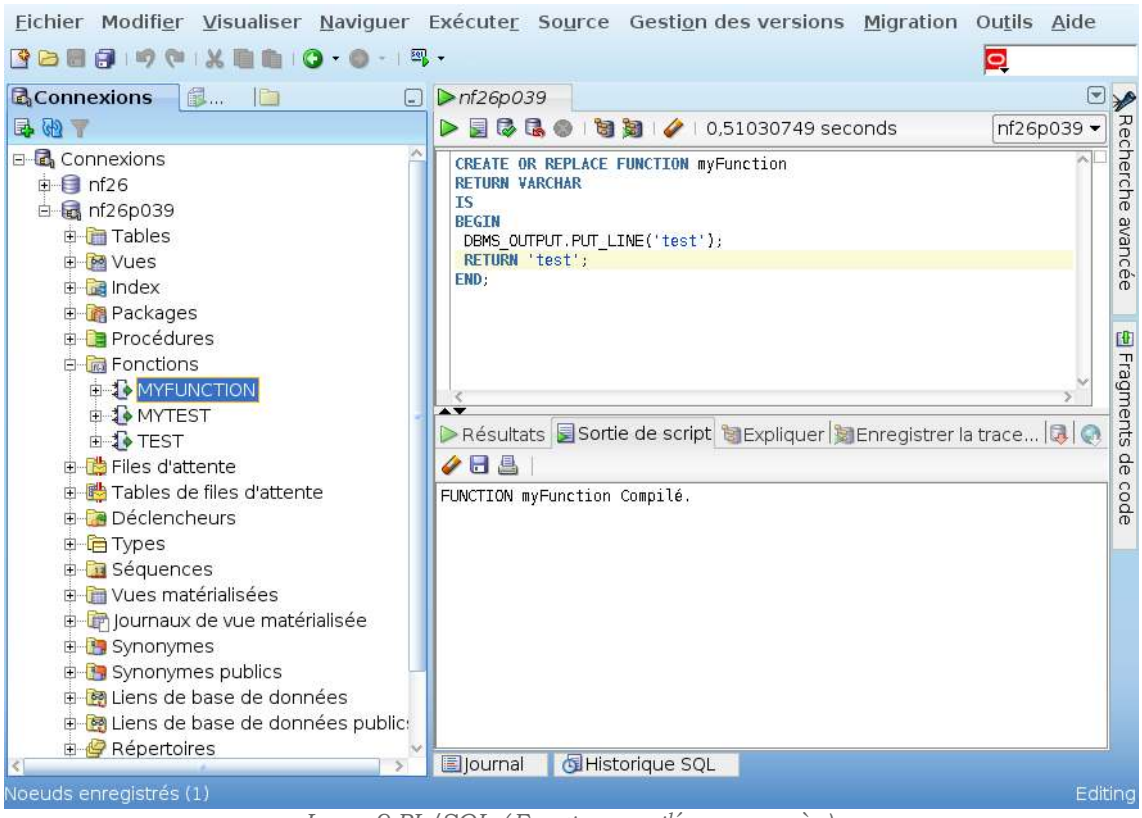


Image 9 PL/SQL (Fonction compilée avec succès)

Attention: Show errors

Terminez vos blocs PL/SQL par l'instruction **`show errors`**, sinon le compilateur ne retourne pas de message.

Complément : Traiter les erreurs de compilation

Pour mieux voir les erreurs de compilation, il faut faire un clic droit sur le nom de la fonction ou de la procédure (zone de gauche), puis choisir `Compiler pour le débogage`.

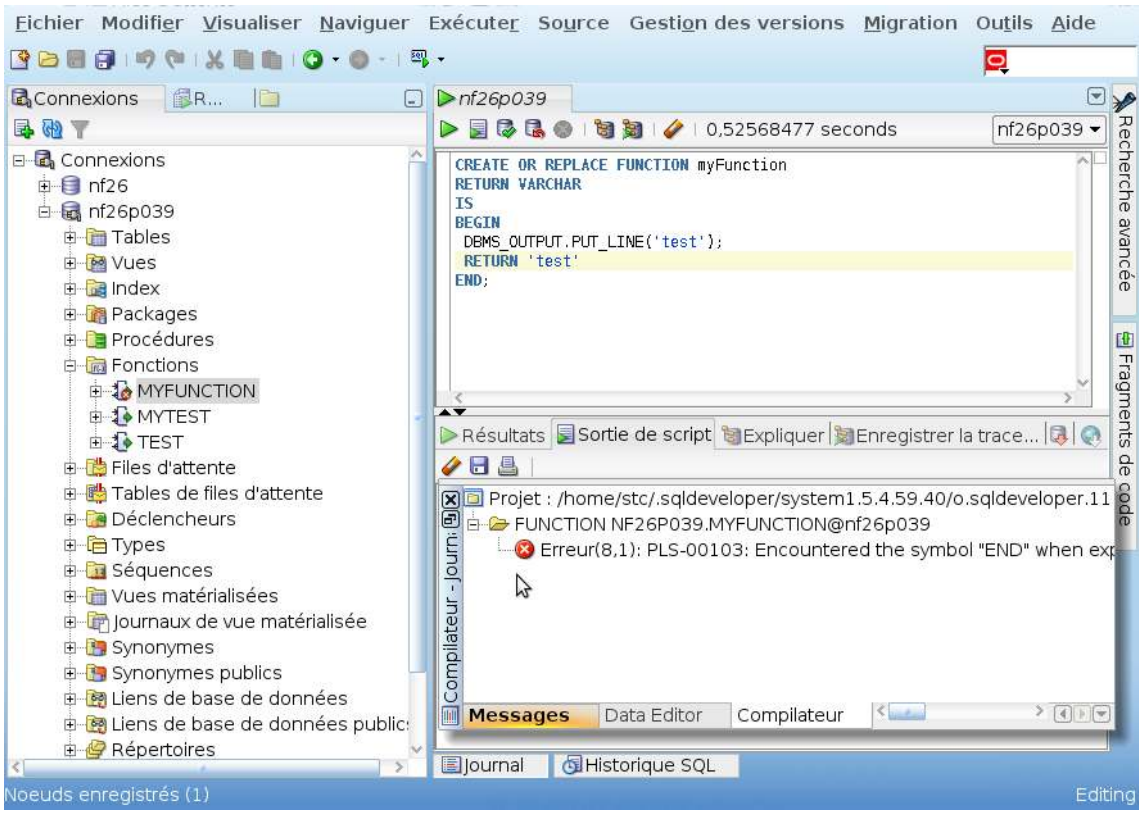


Image 10 PL/SQL (Fonction en erreur)

6. Exécution de fichiers SQL

Méthode

Pour enregistrer un script SQL ou PL/SQL écrit dans *Oracle SQL Developer* sous la forme d'un fichier utiliser la fonction `file > save as`.

Méthode

Pour exécuter un fichier SQL ou PL/SQL utiliser la commande `@fichier.sql`

Exemple : script.sql

```
1 @file1.sql
2 @file2.sql
3 @file3.sql
```

B. Rappels Oracle pour l'ETL

1. Exécution de fichiers SQL

Méthode

Pour enregistrer un script SQL ou PL/SQL écrit dans *Oracle SQL Developer* sous la forme d'un fichier utiliser la fonction `file > save as`.

Méthode

Pour exécuter un fichier SQL ou PL/SQL utiliser la commande `@fichier.sql`

Exemple : script.sql

```
1 @file1.sql
2 @file2.sql
3 @file3.sql
```

2. Insertion de dates avec TO_DATE

Syntaxe : Gestion des dates avec TO_DATE

La gestion des dates peut poser des problèmes selon les formats paramétrés sur le serveur Oracle (en général par défaut le format est DD-MON-YY). La solution la plus rigoureuse consiste à utiliser la fonction de conversion TO_DATE.

Exemple

`TO_DATE('20021130','YYYYMMDD')` équivaut à 30-NOV-2002.

Exemple : Insertion de date dans Oracle

```
1 INSERT INTO Project (Num, Name, Begin, End)
2 VALUES (1, 'Walking on the moon', TO_DATE('20150401','YYYYMMDD'),
  TO_DATE('20160401','YYYYMMDD');
```

Exemple : BD "Gestion des intervenants" : Insert avec date

```
1  INSERT INTO tIntervenant (pknom, prenom, poste)
2  VALUES   ('CROZAT', 'Stéphane', '4287');
3
4  INSERT INTO tCours (pkannee, pknum, titre, type, debut, fkintervenant)
5  VALUES   (2001, 1, 'Introduction','C', TO_DATE('01-01-2001','DD-MM-YYYY'),
   'CROZAT');
6
7  INSERT INTO tCours (pkannee, pknum, titre, type, debut, fkintervenant)
8  VALUES   (2001, 2, 'Modélisation','TD', TO_DATE('08-01-2001','DD-MM-YYYY'),
   'CROZAT');
9
10 SELECT * FROM tCours;
```

```
1 PKANNEE P TITRE           TYPE FKINTERVENANT   DEBUT
2 ------- - --------------- ---- --------------- ---------
3    2001 1 Introduction    C    CROZAT          01-JAN-01
4    2001 2 Modélisation    TD   CROZAT          08-JAN-01
```

Complément

- *TO_DATE (char)*[7] (oracle.com)
- *TO_DATE (char)*[8] (techonthenet.com)

3. Traitement de dates avec TO_CHAR

La fonction TO_CHAR permet de convertir une date en chaîne de caractère, pour l'afficher selon le format souhaité.

Attention: TO_CHAR(date) et "fm" (format mask)

Les paramètres de type `fm` pour *format mask* (`fmday`, `fmDay`, `fmDAY`, `fmMonth`...) permettent de supprimer les zéros et espaces.

7 - http://docs.oracle.com/cd/B19306_01/server.102/b14200/functions183.htm
8 - http://www.techonthenet.com/oracle/functions/to_date.php

Ils sont à privilégier en général :

- **`TO_CHAR(date,'day')`** retourne **`'saturday___'`** (avec des espaces à la fin)
- **`TO_CHAR(date,'fmday')`** retourne **`'saturday'`**

Exemple : BD "Gestion des intervenants" : Question avec date

```
1  SELECT pknum AS cours, TO_CHAR(debut, 'fmday') AS day, TO_CHAR(debut, 'fmww') AS
   week FROM tcours;
```

```
1  COURS DAY       WEEK
2  ----- --------- ----
3      1 monday    1
4      2 monday    2
```

Complément

- *TO_CHAR(date)*[9] ; *TO_CHAR(date)*[10]
- *TO_CHAR(number)*[11]
- *Formatage*[12]

4. Affichage à l'écran

Syntaxe

```
1  SET SERVEROUTPUT ON
```

```
1  BEGIN
2   DBMS_OUTPUT.PUT_LINE ('Hello World');
3  END;
```

5. Transactions en SQL

Introduction

Le langage SQL✯ fournit trois instructions pour gérer les transactions.

Syntaxe : Début d'une transaction

```
1  BEGIN TRANSACTION (ou BEGIN) ;
```

Cette syntaxe est optionnelle (voire inconnue de certains SGBD✯), une transaction étant débutée de façon **implicite** dès qu'instruction est initiée sur la BD✯.

Syntaxe : Fin correcte d'une transaction

```
1  COMMIT TRANSACTION (ou COMMIT) ;
```

Cette instruction SQL signale la fin d'une transaction couronnée de succès. Elle indique donc au gestionnaire de transaction que l'unité logique de travail s'est terminée dans un état cohérent est que les données peuvent effectivement être modifiées de façon durable.

9 - http://docs.oracle.com/cd/B19306_01/server.102/b14200/functions180.htm
10 - http://www.techonthenet.com/oracle/functions/to_char.php
11 - http://docs.oracle.com/cd/B19306_01/server.102/b14200/functions181.htm
12 - http://docs.oracle.com/cd/B19306_01/server.102/b14200/sql_elements004.htm

Syntaxe : Fin incorrecte d'une transaction

```
1 ROLLBACK TRANSACTION (ou ROLLBACK) ;
```

Cette instruction SQL signale la fin d'une transaction pour laquelle quelque chose s'est mal passé. Elle indique donc au gestionnaire de transaction que l'unité logique de travail s'est terminée dans un état potentiellement incohérent et donc que les données ne doivent pas être modifiées en annulant les modifications réalisées au cours de la transaction.

Remarque : Programme

Un programme est généralement une séquence de plusieurs transactions.

C. Rappels triggers pour l'ETL

1. Principes des triggers

Définition : Trigger

Un *trigger* (ou déclencheur) est un bloc PL/SQL associé à une table permettant de déclencher une action avant ou après un INSERT, UPDATE ou DELETE sur cette table.

Les *triggers* sont stockés dans la base.

A quoi servent les triggers ?

- Ils permettent de renforcer l'intégrité des données (mais on préférera des contraintes "check", "unique" ou "foreign key" quand c'est possible).
- Ils permettent d'auditer des actions sur une table.
- Ils permettent de calculer des valeurs dérivées pour d'autres colonnes de la table.

  Ils constituent ainsi une des solutions pour l'implémentation des attributs dérivés.

Types de triggers

Il existe deux types de triggers :

- **Trigger sur ligne**

  le *trigger* est exécuté pour chaque ligne concernée par l'instruction insert, update ou delete (option "for each row").
- **Trigger sur instruction**

  le *trigger* est exécuté une seule fois pour l'instruction insert, update ou delete, même si elle traite plusieurs lignes d'un coup.

Syntaxe : Trigger

```
1 CREATE [OR REPLACE] TRIGGER nom_trigger {BEFORE|AFTER}
2 [INSERT OR][UPDATE [OF nom_colonne] OR][DELETE]
3 ON nom_Table
4 [FOR EACH ROW [WHEN (condition)] ]
5 DECLARE
6  [variable declarations]
7 BEGIN
8  instructions
9 END;
```

Remarque : Avant ou après ?

En général les triggers sont de type "before", en particulier pour les triggers sur ligne, c'est à dire qu'ils s'exécutent avant que l'action considérée soit exécutée, ce qui permet d'infléchir le résultat de cette action. Alors qu'un trigger "after" ne pourra plus modifier le tuple considéré et agira seulement sur d'autres tuples.

Attention : Triggers multiples

Une même table peut avoir plusieurs triggers, mais cela est à éviter en général, pour des raisons de facilité de maintenance et de performance.

Attention : Exception

Si l'exécution du trigger échoue, l'action (insert, update ou delete dans la table) est annulée (et retourne une exception Oracle).

2. Prédicats d'événement au sein des triggers

- INSERTING
- DELETING
- UPDATING
- UPDATING(nom_colonne)

Prédicats pour savoir dans quel contexte d'appel du trigger on est, ce qui permet dans un même trigger de s'adapter aux différents cas de déclenchement.

Rappel : BD "Gestion des intervenants" : Schéma relationnel

```
tIntervenant (#pknom:varchar, prenom:varchar, poste:integer)
tCours (#pkannee:2000..2100, #pknum:integer, titre:varchar, type:C|TD|TP,
fkintervenant=>tIntervenant, debut:date)
```

Exemple : BD "Gestion des intervenants" : Trigger d'archivage de données

```
CREATE TABLE tIntervenantSav (
pknom varchar2(20) PRIMARY KEY,
prenom varchar2(20) NOT NULL
);

CREATE OR REPLACE TRIGGER trIntervenant
BEFORE DELETE OR INSERT ON tIntervenant
FOR EACH ROW
BEGIN
  IF DELETING THEN
    INSERT INTO tIntervenantSav VALUES (:old.pknom, :old.prenom);
  ELSIF INSERTING THEN
    DELETE FROM tIntervenantSav WHERE pknom = :new.pknom;
  END IF;
END;
/

DELETE FROM tCours;
DELETE FROM tIntervenant;
SELECT * FROM tIntervenantSav;
```

```
PKNOM                PRENOM
-------------------- --------------------
CROZAT               Stéphane
JOUGLET              Antoine
VINCENT              Antoine
```

```
INSERT INTO tIntervenant (pknom, prenom, poste)
VALUES  ('CROZAT', 'Stéphane', '4287');

SELECT * FROM tIntervenantSav;
```

```
PKNOM                PRENOM
-------------------- --------------------
```

```
3   JOUGLET             Antoine
4   VINCENT             Antoine
```

3. Manipulation des anciennes et nouvelles valeurs dans les triggers (:old et :new)

Pour les triggers de type "for each row", les colonnes de la ligne courante doivent être référencées spécifiquement selon que l'on veut l' ancienne ou la nouvelle valeur :

- `:old.nom_colonne`
- `:new.nom_colonne`

Fondamental

Il ne faut pas lire des données d'une table en cours de modification autrement que par les accès ":old" et ":new".

Attention : Anciennes valeurs en lecture seule

Il n'est jamais possible de modifier une colonne ":old".

Attention : Valeurs en lecture seule après

Pour les trigger "after", il n'est plus possible de modifier les colonnes ":new".

Remarque : Valeurs nulles

Pour les triggers "on insert" les colonnes ":old" ont la valeur NULL.

Pour les triggers "on delete" les colonnes ":new" ont la valeur NULL.

Attention

Il ne faut pas modifier de données dans les colonnes des "primary key", "foreign key", ou "unique key" d'une table.

Rappel : BD "Gestion des intervenants" : Schéma relationnel

```
1   tIntervenant (#pknom:varchar, prenom:varchar, poste:integer)
2   tCours (#pkannee:2000..2100, #pknum:integer, titre:varchar, type:C|TD|TP,
    fkintervenant=>tIntervenant, debut:date)
```

Exemple : BD "Gestion des intervenants" : Trigger de gestion de cohérence

```
1   CREATE OR REPLACE TRIGGER trCours
2   BEFORE INSERT OR UPDATE OF debut ON tCours
3   FOR EACH ROW
4   DECLARE
5     vAnneeDebut INTEGER;
6   BEGIN
7     vAnneeDebut := TO_NUMBER(TO_CHAR(:new.debut,'YYYY'));
8     IF NOT(vAnneeDebut = :new.pkannee) THEN
9       :new.debut:=null;
10      DBMS_OUTPUT.PUT_LINE('Inconsistency between debut and pkannee, debut set to
    null');
11    END IF;
12  END;
13  /
14
15  SET SERVEROUTPUT ON;
16
17  INSERT INTO tCours (pkannee, pknum, titre, type, fkIntervenant, debut)
18  VALUES ('2001', 3, 'SQL', 'C', 'CROZAT', TO_DATE('15-01-2001','DD-MM-YYYY'));
19
20  UPDATE tCours
21  SET debut=TO_DATE('15-01-2002','DD-MM-YYYY')
22  WHERE pknum=3;
23
24  SELECT pkannee, pknum, debut FROM tCours;
```

```
TRIGGER trCours compiled
1 rows inserted.
1 rows updated.
Inconsistency between debut and pkannee, debut set to null

PKAN PKNUM DEBUT
---- ----- ---------
2001     1 01-JAN-01
2001     2 08-JAN-01
2001     3
```

4. Quelques règles à respecter pour les triggers

Attention

Il ne faut pas modifier de données dans les colonnes des primary key, foreign key, ou unique key d'une table.

Attention

Il ne faut pas lire des données d'une table en cours de modification autrement que par les accès :old et :new.

D. Rappels Oracle RO

1. Création de type en SQL3 sous Oracle (extension au LDD)

Syntaxe : Déclaration de type

```
CREATE TYPE nom_type AS OBJECT (
  nom_attribut1 type_attribut1,
  ...
);
/
```

Exemple : Création de tables d´objets (enregistrements avec OID)

```
CREATE TABLE t OF nom_type (
  ...
)
```

Exemple : Usage des types dans les tables (modèle imbriqué)

```
CREATE TABLE t (
  ...
  nom_attribut nom_type,
  ...
)
```

Complément

Héritage et réutilisation de types

Méthodes de table d´objets

2. Création de table objet (modèles et LDD)

Définition

Une table peut être définie en référençant un type de données plutôt que par des instructions LDD★ classiques. On parle alors de table objet.

Synonymes : table-objet, table d'objets

Syntaxe : Modèle logique

```
nom_type : <...>
nom_table de nom_type (#attributs_clés) autres contraintes
```

Syntaxe : LDD SQL3

```
CREATE TABLE nom_table OF nom_type (
 PRIMARY KEY(attribut1),
 attribut2 NOT NULL,
 UNIQUE (attribut3)
 FOREIGN KEY (attribut4) REFERENCES ...
);
```

Il est possible, sur une table ainsi définie, de spécifier les mêmes contraintes que pour une table créée avec une clause CREATE TABLE (contraintes de table). Ces contraintes doivent être spécifiées au moment de la création de la table, et non au moment de la création du type (bien que la définition de type permet de spécifier certaines contraintes, comme NOT NULL).

Fondamental : OID

Les enregistrements d'une table-objet peuvent être identifiés par un OID★

Complément : Méthodes

Des méthodes peuvent être associées à une table-objet.

Complément : Héritage

Cette modalité de définition de schéma permet de profiter de l'héritage de type pour permettre l'héritage de schéma de table.

Exemple

```
CREATE OR REPLACE TYPE typIntervenant AS OBJECT(
pknom varchar2(20),
prenom varchar2(20)
);
/

CREATE TABLE tIntervenant OF typIntervenant (
PRIMARY KEY(pknom),
prenom NOT NULL
);
```

3. Méthodes de table d'objets

Définition : Méthodes de table

Si le type sur lequel s'appuie la création de la table définit des méthodes, alors les méthodes seront associées à la table (méthodes de table).

Il sera possible d'accéder à ces méthodes de la même façon que l'on accède aux attributs (projection, sélection...).

Syntaxe : Accès aux méthodes d´une table d´objets

```
SELECT t.m1(), t.m2() ...
```

```
2  FROM table t
3  ...
```

Attention

L'utilisation d'un alias est obligatoire pour accéder aux méthodes.

Syntaxe : Déclaration de type avec méthodes

```
1  CREATE TYPE nom_type AS OBJECT (
2   nom_attribut1 type_attribut1
3   ...
4   MEMBER FUNCTION nom_fonction1 (parametre1 IN|OUT type_parametre1, ...) RETURN
    type_fonction1
5   ...
6  );
7  /
8  CREATE TYPE BODY nom_type
9   IS
10  MEMBER FUNCTION nom_fonction1 (...) RETURN type_fonction1
11   IS
12   BEGIN
13   ...
14   END ;
15  MEMBER FUNCTION nom_fonction2 ...
16   ...
17  END ;
18 END ;
19 /
```

Exemple

```
1  CREATE TYPE typCours AS OBJECT (
2    pknum NUMBER(2),
3    debut DATE,
4    MEMBER FUNCTION fin RETURN DATE
5  );
6  /
7  CREATE TYPE BODY typCours IS
8  MEMBER FUNCTION fin RETURN DATE
9    IS
10   BEGIN
11     RETURN SELF.debut + 5;
12   END;
13 END;
14 /
15
16 CREATE TABLE tCours OF typCours (
17 pknum PRIMARY KEY
18 );
19
20 SELECT c.pknum, c.fin()
21 FROM tCours c;
```

Remarque : Type retourné par une méthode

« The datatype cannot specify a length, precision, or scale. »

http://docs.oracle.com/cd/B13789_01/server.101/b10759/statements_5009.htm[13]

13 - http://docs.oracle.com/cd/B13789_01/server.101/b10759/statements_5009.htm

4. Méthodes et SELF

SELF

Lorsque l'on écrit une méthode on a généralement besoin d'utiliser les attributs propres (voire d'ailleurs les autres méthodes), de l'objet particulier que l'on est en train de manipuler.

On utilise pour cela la syntaxe SELF qui permet de faire référence à l'objet en cours.

Syntaxe : SELF

```
1  self.nom_attribut
2  self.nom_méthode(...)
```

Exemple : Total d´une facture

```
1  MEMBER FUNCTION total RETURN number
2  IS
3   t number;
4  BEGIN
5   SELECT sum(f.qte) INTO t
6   FROM facture f
7   WHERE f.num=self.num;
8
9   RETURN t;
10 END total;
```

Remarque : SELF implicite

Dans certains cas simples, lorsqu'il n'y a aucune confusion possible, SELF peut être ignoré et le nom de l'attribut ou de la méthode directement utilisé.

Il est toutefois plus systématique et plus clair de mettre explicitement le self.

Exemple : Exemple de SELF implicite

```
1   MEMBER FUNCTION adresse RETURN varchar2
2   IS
3   BEGIN
4    RETURN num || rue || ville;
5   END;
```

Signification des abréviations

- API Application Program Interface
- BD Base de Données
- BDE Base de Données d'Extraction
- BDT Base de Données de Transformation
- BI Business Intelligence
- CSV Comma Separated Values
- DM Data Mart
- DW Data Warehouse
- ETL Extraction, Transformation, Loading
- LDD Langage de Définition de Données
- OID Object Identifier
- OS Operating Système (Système d'Exploitation)
- RO Relationnel-Objet
- SGBD Système de Gestion de Bases de Données
- SGBDR Système de Gestion de Bases de Données Relationnelles
- SGBDRO Système de Gestion de Bases de Données Relationnelles-Objets
- SQL Structured Query Language
- XML eXtensible Markup Language

Bibliographie

[(Goglin, 2001)] Goglin J.-F. (2001, 1998). *La construction du datawarehouse : du datamart au dataweb*. Hermes, 2ème édition.

[(Inmon, 2002)] Inmon W.-H. (2002, 1990). *Building the data warehouse*. Wiley Publishing, 3rd edition.

[(Kimbal, Ross, 2003)] Kimball R., Ross M. (2003). *Entrepôts de données : guide pratique de modélisation dimensionnelle*. Vuibert.

[(Kimball, Caserta, 2004)] Kimball R., Caserta J. (2004). *The Data Warehouse ETL Toolkit*. Wiley Publishing.

[(Kimball, Ross, 2008)] Kimball R., Ross M. (2008, 2002). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Wiley Publishing, second edition.

[(Kimball et al., 2008)] Kimball R., Ross M., Thornthwaite W., Mundy J., Becker B. (2008, 1998). *The Data Warehouse Lifecycle Toolkit*. Wiley Publishing, second edition.

[Abbey01] Abbey Michael, Corez Michael, Abramson Ian, *Oracle 9i : Notions fondamentales*, CampusPress, 2001.

Webographie

[(Smile, 2012)] Smile (2012, 2006). *Décisionnel : le meilleur des solutions open source*. *http://www.smile.fr/Livres-blancs/ERP-et-decisionnel/Le-decisionnel-open-source* .

[w_loria.fr/~roegel(1)] Roegel Denis, *Oracle : SQL*, *http://www.loria.fr/~roegel/cours/iut/oracle-sql.pdf* , 1999.