Introduction : SGBD/DBMS

Cours de D. Ploix
Université Évry Val d'Essonne
M1

références

Craig Mullins, Database Administration

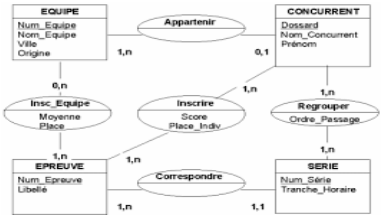
Site "TechTarget", DBMS

(SG)BD != BD

Utilisateurs

Applications

Données



Physique

Composants d’un SGBD

Composants *BD* :

- Modèles de données, données, applications connectées, utilisateurs, requêtes, indexes

Composants *SG* :

- Moteur, réseau, CPU, RAM, stockage, OS, progiciel SGBD, ...

Les composants DB sont *relativement* indépendants des choix de SG. Le cours ASGBD vise à identifier les impacts sur les composants de l'application (DB) des choix des systèmes de gestion des BD.

Familles de SGBD / db-engines.com

345 systems in ranking, September 2018

Rank Sep 2018	Rank Aug 2018	Rank Sep 2017	DBMS	Database Model	Score Sep 2018	Score Aug 2018	Score Sep 2017
1.	1.	1.	Oracle	Relational DBMS	1309.12	-2.91	-49.97
2.	2.	2.	MySQL	Relational DBMS	1180.48	-26.33	-132.13
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1051.28	-21.37	-161.26
4.	4.	4.	PostgreSQL	Relational DBMS	406.43	-11.07	+34.07
5.	5.	5.	MongoDB	Document store	358.79	+7.81	+26.06
6.	6.	6.	DB2	Relational DBMS	181.06	-0.78	-17.28
7.	↑ 8.	↑ 10.	Elasticsearch	Search engine	142.61	+4.49	+22.61
8.	↓ 7.	↑ 9.	Redis	Key-value store	140.94	+2.37	+20.54
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	133.39	+4.30	+4.58
10.	10.	↓ 8.	Cassandra	Wide column store	119.55	-0.02	-6.65
11.	11.	11.	SQLite	Relational DBMS	115.46	+1.73	+3.42
12.	12.	12.	Teradata	Relational DBMS	77.38	-0.02	-3.52
13.	13.	↑ 16.	Splunk	Search engine	74.03	+3.53	+11.45
14.	14.	↑ 18.	MariaDB	Relational DBMS	70.64	+2.34	+15.17
15.	15.	↓ 13.	Solr	Search engine	60.20	-1.69	-9.71
16.	↑ 18.	↑ 19.	Hive	Relational DBMS	59.63	+1.69	+11.02
17.	17.	↓ 15.	HBase	Wide column store	58.47	-0.33	-5.87
18.	↓ 16.	↓ 14.	SAP Adaptive Server	Relational DBMS	58.04	-2.39	-8.71
19.	19.	↓ 17.	FileMaker	Relational DBMS	55.30	-0.75	-5.69
20.	↑ 21.	↑ 22.	Amazon DynamoDB	Multi-model	53.34	+1.69	+15.52
21.	↓ 20.	↓ 20.	SAP HANA	Relational DBMS	52.73	+0.80	+4.40
22.	22.	↓ 21.	Neo4j	Graph DBMS	40.10	-0.83	+1.67
23.	23.	23.	Couchbase	Document store	34.55	+1.59	+1.44
24.	24.	24.	Memcached	Key-value store	31.54	-1.38	+2.60
25.	25.	↑ 27.	Microsoft Azure SQL Database	Relational DBMS	25.25	-0.85	+3.65
26.	26.	↓ 25.	Informix	Relational DBMS	24.91	-0.48	-2.93
27.	↑ 28.	↓ 26.	Vertica	Relational DBMS	20.36	+0.32	-1.65
28.	↓ 27.	↑ 30.	Firebird	Relational DBMS	20.25	-0.04	+2.38
29.	29.	↑ 37.	Microsoft Azure Cosmos DB	Multi-model	19.18	-0.35	+7.95
30.	30.	↓ 28.	CouchDB	Document store	18.64	+0.21	-2.35

Structure du cours

CM :

1) SGBD SQL sur base Oracle
2) SGBD NoSQL sur base Hbase

TD :

- Mise en contexte des sujets abordés en CM
- Mise en œuvre des SGBD Oracle et Big Data
  - construire les VM suivant les indications données sur la page du cours.

Schéma de données

Préliminaire : conception du modèle de base de données

- Transactionnel, Analytique, …
- Définissant :
  - Des tables contenant des données
  - Des requêtes sur les données
    - En lecture ou écriture
    - Intégrant des calculs
    - Nécessitant un contrôle d'accès

Du besoin à l'application

Analyse de la problématique → modélisation conceptuelle, décoréllée d'un contexte physique (logique).

- Si le contexte applicatif est transactionnel, on vérifiée avec attention le respect des formes normales, sources de 80 % des problèmes de performance.
- Si le contexte applicatif est décisionnel, on établi une modélisation décisionnelle (dénormalisée).

Passage à l'implémentation → création du schéma logique / physique

Construction d'une base de données *rappel sur les formes normales*

- Formes normales : un enregistrement est « en 1ère ssi tous les domaines sous-jacents ne contiennent que des données atomiques »

Exemple : mettre la table suivant en 1FN

Client	Jour 1	Jour 2	Montants
Ho David	Lundi		19,20 19,20
Ho David	Mercredi	Jeudi	-80
Ho Marian	Jeudi		100,00 150,00 -40,00

Construction d'une base de données
rappel sur les formes normales

Groupes multiples par séparation de lignes

Client	Jour 1	Jour 2	Montant
Ho David	Lundi		19,2
Ho David	Lundi		19,2
Ho David	Mercredi	Jeudi	-80
Ho Marina	Jeudi		100
Ho Marina	Jeudi		150
Ho Marina	Jeudi		-40

Transactions identiques via ID unique

ID achat	Client	Jour 1	Jour 2	Montant
1	Ho David	Lundi		19,2
2	Ho David	Lundi		19,2
3	Ho David	Mercredi	Jeudi	-80
4	Ho Marina	Jeudi		100
5	Ho Marina	Jeudi		150
6	Ho Marina	Jeudi		-40

Plusieurs éléments significatifs dans 1 seule colonne

ID achat	Nom Client	Prénom Client	Jour 1	Jour 2	Montant
1	Ho	David	Lundi		19,2
2	Ho	David	Lundi		19,2
3	Ho	David	Mercredi	Jeudi	-80
4	Ho	Marina	Jeudi		100
5	Ho	Marina	Jeudi		150
6	Ho	Marina	Jeudi		-40

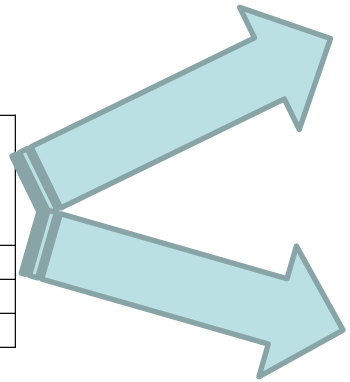
Plusieurs colonne représentant le même fait

ID achat	Nom Client	Prénom Client	Jour	Montant
1	Ho	David	Lundi	19,2
2	Ho	David	Lundi	19,2
3	Ho	David	Mercredi	-80
4	Ho	Marina	Jeudi	100
5	Ho	Marina	Jeudi	150
6	Ho	Marina	Jeudi	-40
7	Ho	David	Jeudi	-80

Construction d'une base de données *rappel sur les formes normales*

- Formes normales : un enregistrement est « en 2ème ssi 1ère + toutes les données n'étant pas des clés dépendent entièrement de clés »

ID Produit Clé primaire	ID Fournisseur clé primaire	Nom du fournisseur	Prix	Adresse
65	2	Lautre	59.99	Paris
73	2	Lautre	20.00	Paris
65	1	Leclerk	59.99	Lyon



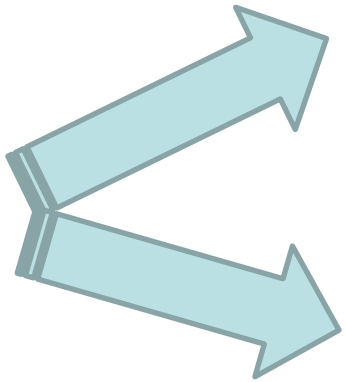
ID Fournisseur clé primaire	Nom du fournisseur	Adresse
1	Lautre	Paris
2	Leclerk	Lyon

ID Produit Clé primaire	ID Fournisseur clé primaire	Prix
65	2	59.99
73	2	20.00
65	1	59.99

Construction d'une base de données *rappel sur les formes normales*

- Formes normales : un enregistrement est « en 3ème ssi 2ème + tous les éléments hors clés primaire sont indépendants transitivement de celle si »

ID Produit Clé primaire	**Nom du fournisseur**	**Adresse**
1000	Toyota	Park Avenue
1001	Mitsubishi	Lincoln Street
1002	Toyota	Park Avenue



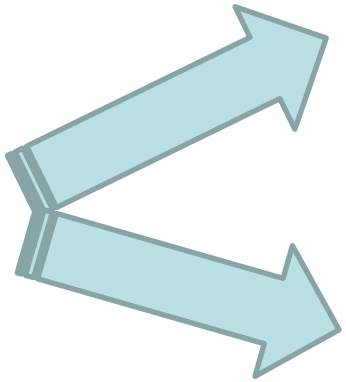
Nom du fournisseur	
clé primaire	**Adresse**
Toyota	Park Avenue
Mitsubishi	Lincoln Street

ID Produit	**Nom du**
clé primaire	**Fournisseur**
1000	Toyota
1001	Mitsubishi
1002	Toyota

Construction d'une base de données
rappel sur les formes normales

- Formes normales : un enregistrement est « en Boyce-Codd ssi 3ème + les clés sont les uniques sources de dépendance fonctionnelle »

Clé primaire			
étudiant	matière	enseignant	note
Jules	réseau	Petit	10
Jules	ASGBD	Ploix	12



clé primaire		
étudiant	matière	note
Jules	réseau	10
Jules	ASGBD	12

matière	enseignant
réseau	Petit
ASGBD	Ploix

Modèle logique/physique

Construction du modèle logique / physique à partir du modèle conceptuel

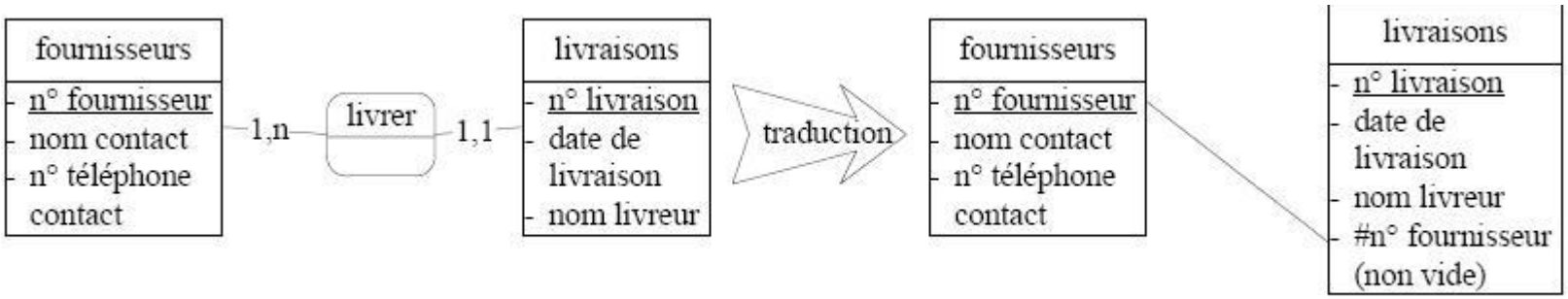
Toute entité devient une table dans laquelle les attributs deviennent les colonnes. L'identifiant de l'entité constitue alors la clé primaire de la table.

Gestion des PK / FK

- Si pas de clé primaire dans une entité, on construit une clé primaire « technique »
- Si une clé primaire complexe est référencée, on construit une clé primaire « technique » (optimisation de l'espace)

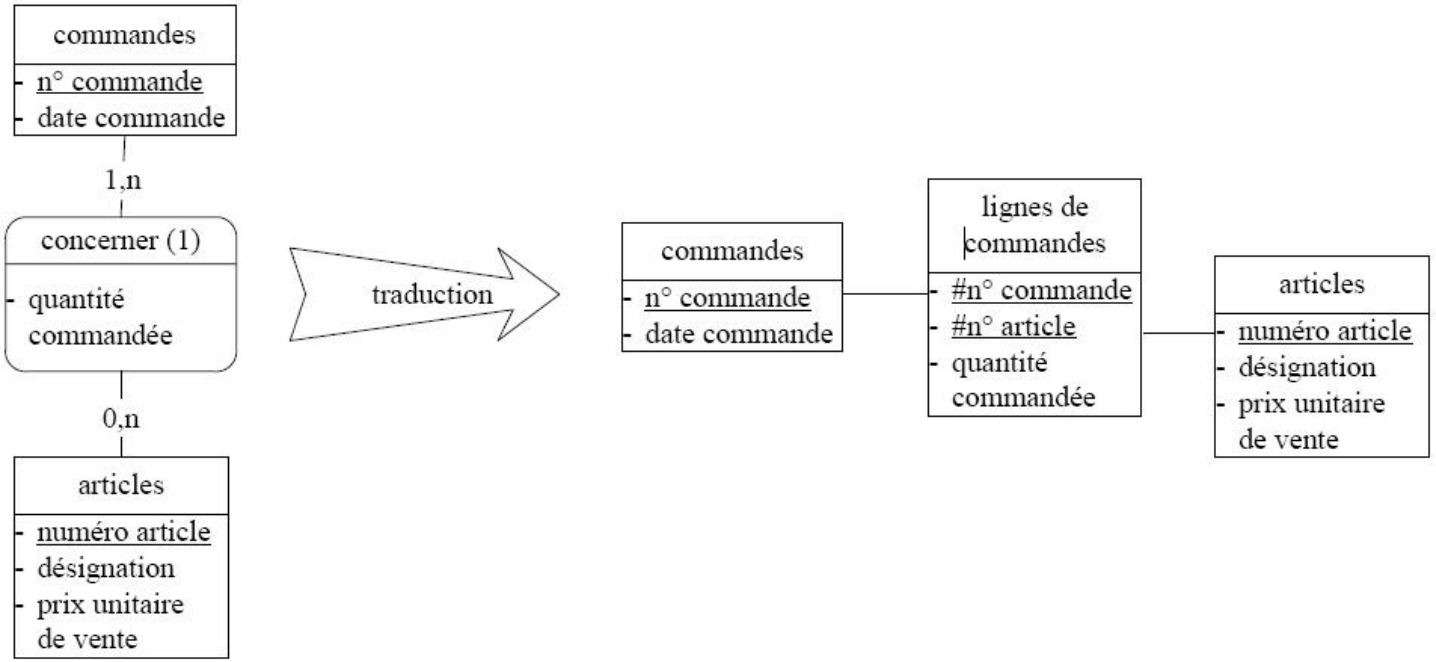
Construction du modèle logique / physique à partir du modèle conceptuel

Une association binaire de type 1 : n disparaît, au profit d'une clé étrangère dans la table coté 0,1 ou 1,1 qui référence la clé primaire de l'autre table. Cette clé étrangère ne peut pas recevoir la valeur vide si la cardinalité est 1,1



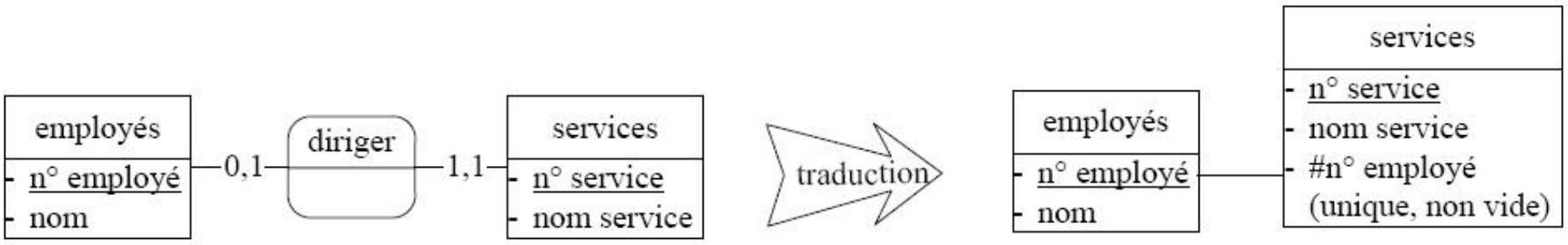
Construction du modèle logique / physique à partir du modèle conceptuel

Une association binaire de type n :m devient une table supplémentaire (table de jonction) dont la clé primaire est composée des deux clés étrangères.



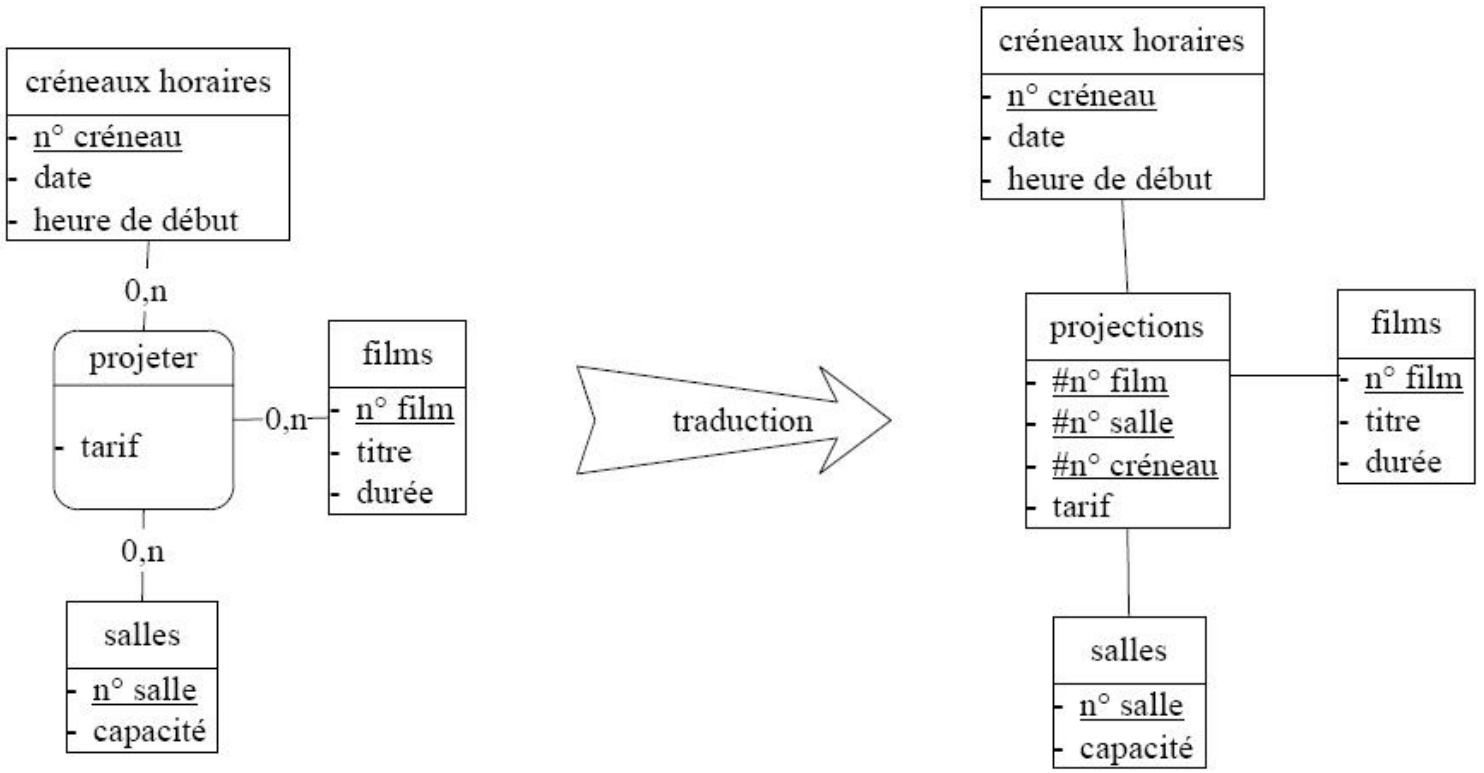
Construction du modèle logique / physique à partir du modèle conceptuel

Une association binaire de type 1 :1 est traduite comme une association binaire de type 1 :n sauf que la clé étrangère se voit imposer une contrainte d'unicité en plus d'une éventuelle contrainte de non vacuité (cette contrainte d'unicité impose à la colonne correspondante de ne prendre que des valeurs distinctes).



Construction du modèle logique / physique à partir du modèle conceptuel

Une association non binaire est traduite par une table supplémentaire dont la clé primaire est composée d'autant de clés étrangères que d'entité en association. Les attributs de l'association deviennent les colonnes de cette nouvelle table.



Construction du modèle physique *traduction des types de données*

Type OCL	**Type SQL2**	**Oracle**	**MySQL**
Boolean	BIT(1)	Non supporté (utiliser CHAR(1) + CHECK)	BIT, BOOL, BOOLEAN
Integer	INTEGER ou SMALLINT	NUMBER(n)	TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT
String	CHARACTER (CHAR) (n), CHARACTER VARYING (VARCHAR) (n)	VARCHAR2(n), LONG ou LONG VARCHAR (chaîne jusqu'à 2G), CLOB (chaîne jusqu'à 4G), NCLOB (chaîne pour caractères encodés sur plusieurs octets)	CHAR(n), VARCHAR(n), BLOB, TEXT
Real	NUMERIC(p,s) (précision exacte), DECIMAL(p,s), REAL, DOUBLE PRECISION, FLOAT(p)	NUMBER(p,s)	SQL2
Enum{v1,…vn}	CHARACTER (CHAR) ou VARCHER + CHECK … IN (v1,…,vn) (possibilité de création de domaine)	Domaine non supporté	ENUM
	DATE	DATE inclut TIME	SQL2
	TIME		SQL2
	TIMESTAMP		SQL2
	BIT(n), BIT VARYING(n)	RAW(n : max = 255), LONG RAW (binaire jusqu'à 2G), BLOB (binaire jusqu'à 4G)	SQL2
		BFILE (pointeur à un fichier externe)	

Installation d'une BD

Concepts de base : vocabulaire Oracle

- Logiques :
  - Utilisateurs = qui est connecté
    - A des droits sur des objets présents dans des schémas
  - Schéma = ce qui est stocké
    - ensemble d'objets (tables, vues, indexes, cluster…) qui appartiennent à un utilisateur et qui en porte le nom
- Physiques :
  - Data Blocks, Extents, et Segments
    - organisation des éléments physiques
  - Mémoire SGA et Processus serveurs
    - Contrôle et optimisation du fonctionnement d'Oracle
  - Listener
    - accessibilité depuis le réseau
- Faisant le lien
  - Tablespace = espace de stockage du pdv de ce qui est stocké
    - groupement de structures logiques
  - Profils de ressources,
  - …

Concepts de base
Structure Physique

- **Datafiles** :
  - Conteneur des données (structurés en blocks, …)
- **Redo Log files** :
  - Trace dynamique des modifications effectuées sur les données
- **Archive log files** :
  - Archive des traces des modifications effectuées
- **Control files** :
  - Description de la structure physique de la base

Concepts de base
Utilisateurs

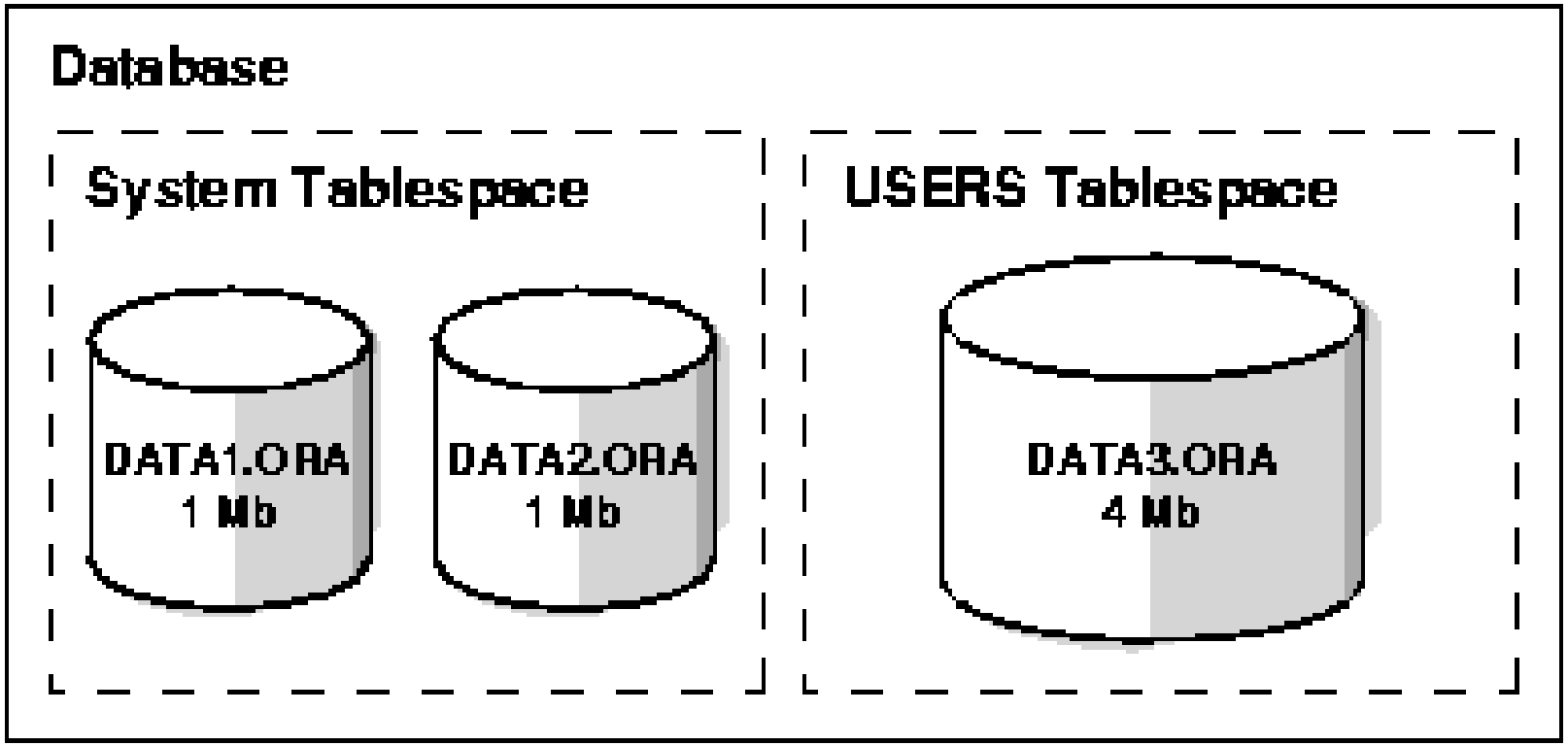
- Utilisateurs liés à l'OS :
  - Sous Unix : groupe dba (utilisateur *oracle*)
  - Sous Windows : groupe ORA_DBA
  - Peuvent se connecter « CONNECT / AS SYSDBA » sans saisir de mot de passe.
- Utilisateurs définis dans Oracle :
  - Utilisateurs avec le rôle DBA
  - Utilisateurs liés à l'application

L'utilisateur correspond à un rôle qui donne accès à des actions particulières sur tout ou partie des données !

Concepts de base
Stockage

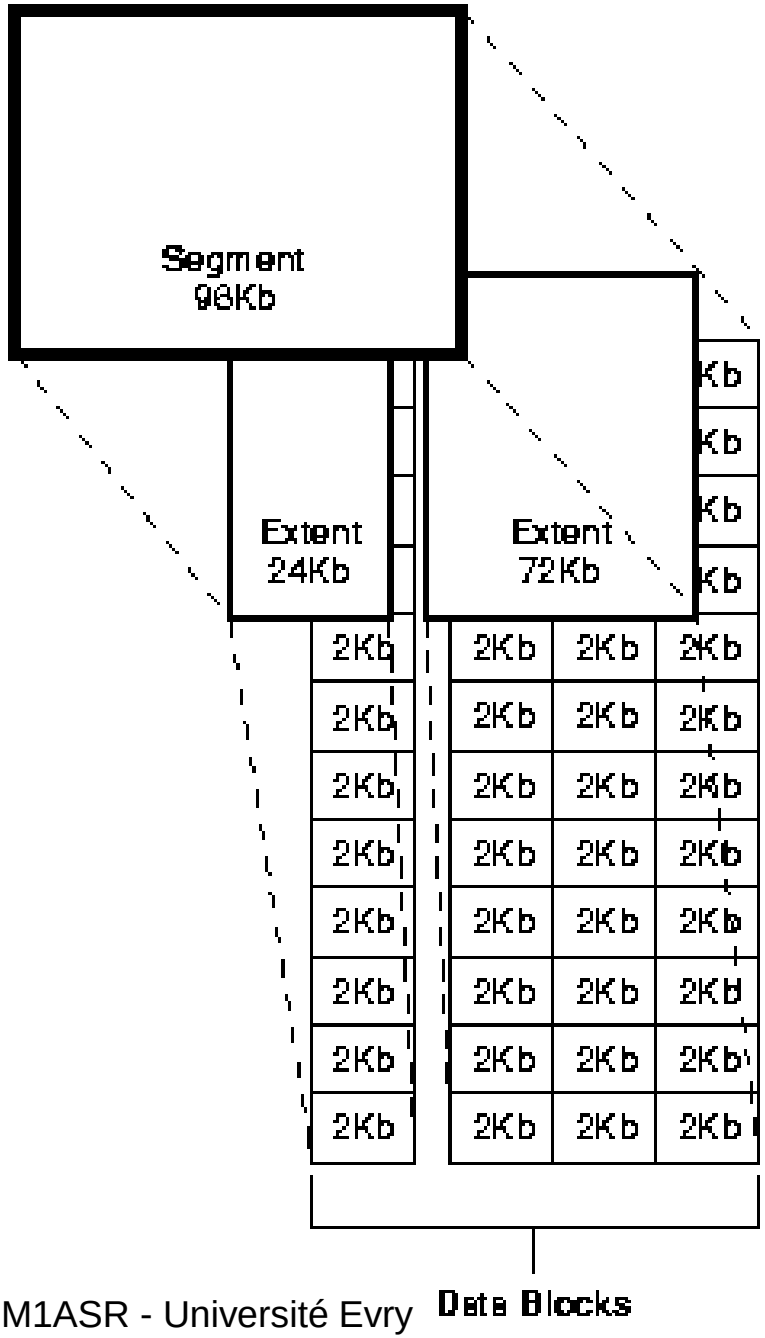
- Niveau Logique :
  - Structures (Table, index, …) et données
    - Correspondant à un usage
    - Appartenant à un utilisateur
- Niveau physique :
  - Fichiers gérés par un FS
- Niveau intermédiaire :
  - Un « espace de stockage » (tablespace) contenant des éléments logiques (applicatifs) et lié à des éléments physiques

Concepts de base
Tablespaces



Concepts de base Répartition physique

limitation max_extents : les extents correspondent à une fragmentation physique du stockage



Calcul de la taille nécessaire au stockage

Deux méthodes :

- Estimations large via des règles de calcul
- Estimation via des outils :
  - Dans les outils de conception (Toad, …)
  - Dans Oracle (DBMS_SPACE)

Caractéristique **pct_free** des tables et indexes :

Correspond au % de chaque data block laissé libre en cas de modification des enregistrements.

Best Practice :

Pour les indexes dont les valeurs indexées changent rarement : 2

Pour les tables dont les valeurs changent rarement : 2

Pour les tables dont les valeurs changent fréquement : 10 à 30

Il faut toujours laisser un peu (2 min) car Oracle à besoin de cet espace dans certain cas.

Calcul de la taille nécessaire au stockage règles de calcul d'une estimation globale

Permet l'estimation globale en amont de la création de la base de données

La somme des tailles des attributs donnent la taille maximale d'un enregistrement

Le nombre d'enregistrement à considérer est :

- Le nombre initial (à la création)
- Le taux de croissance doit permettre, si capacity planning il y a, d'optimiser l'occupation d'espace

Table ETUDIANT

Enregistrement	Type	Taille Max	Taille Moyenne
NOM	VARCHAR2	100	10
PRENOM	VARCHARS	100	10
ID_ETUDIANT	NUMBER	10	10

Taille Enregistrement	30	
Nombre Initial	1 000 000	
Taille Table	30 000 000	(30 Mo)
Croissance 10 % par an, taille à 5 ans :	48 315 300	(50 Mo)
Prise en compte du pct_free de 10 % :	55 Mo	

Calcul de la taille nécessaire au stockage
outils Oracle de calcul de la taille d'une table

```
SET SERVEROUTPUT ON;

DECLARE
v_used_bytes NUMBER(10);
v_Allocated_Bytes NUMBER(10);
v_type sys.create_table_cost_columns;
BEGIN
v_Type := sys.create_table_cost_columns(
sys.create_table_cost_colinfo('NUMBER',9),
sys.create_table_cost_colinfo('VARCHAR2',50),
sys.create_table_cost_colinfo('VARCHAR2',15),
sys.create_table_cost_colinfo('DATE',NULL),
sys.create_table_cost_colinfo('DATE',NULL)
);

DBMS_SPACE.CREATE_TABLE_COST('USERS',v_Type,10000,7,v_used_Bytes,v_Allocated_Bytes);

DBMS_OUTPUT.PUT_LINE('Used Bytes: ' || TO_CHAR(v_used_Bytes));
DBMS_OUTPUT.PUT_LINE('Allocated Bytes: ' || TO_CHAR(v_Allocated_Bytes));

END;
```

Used Bytes: 696 320 ➔ taille utilisée par 10 000 enregistrements de la table (théorique de 900 000)
Allocated Bytes: 720 896 ➔ taille occupée (% free à 7), surcoût de 3,5 %.

Calcul de la taille nécessaire au stockage outils Oracle de calcul de la taille d'un indexe

```
declare
        calc_used_bytes NUMBER;
        calc_alloc_bytes NUMBER;
begin
        DBMS_SPACE.CREATE_INDEX_COST (
          ddl => 'create index BOOK_CAT on BOOKSHELF (CategoryName) tablespace
      BOOKS_INDEX',
           used_bytes => calc_used_bytes,
          alloc_bytes => calc_alloc_bytes
        );
        DBMS_OUTPUT.PUT_LINE('Used bytes = '||calc_used_bytes);
        DBMS_OUTPUT.PUT_LINE('Allocated bytes = '||calc_alloc_bytes);
end;
```

Imprimera la taille nécessaire au stockage de l'indexe sur la colonne CategoryName de la table BOOKSHELF dans le tablespace BOOKS_INDEX

Utile, mais uniquement si la base existe déjà et qu'il y a déjà des données dans la table… donc en cas d'ajout d'un indexe sur une table existante

Quelques particularités des tables Oracle

Spécifications du modèle de stockage des données (caractéristiques des tables) :

- Indication du taux de remplissage des blocks de données : influence l'espace disque utilisé
  - PCTFREE 10 : % préservé pour des données future
  - PCTUSED 40 : % d'occupation pour d'être remis dans le pool des blocks libres
  - PCTFREE + PCTUSED ≤ 100
- Taille de bloc, taille initiale (initial) et de progression (next)
- Initrans et maxtrans pour indiquer le type d'accès à l'objet…
- Logging ou nologging ?

Quelques particularités des tables Oracle

Contraintes de validité des colonnes :

- Champ type CONSTRAINT <nom> (NOT NULL | check(<expression>))
- Constraint <nom> [primary|foreign] key (champ(s)) [deferrable];
- Reference (champ) [deferrable];

Concepts de base relatifs aux schéma de données

Liens entre bases de données :

- Database links

Éléments de stockage des données :

- Tables et index-organized tables
- Clusters de tables
- Indexes et index types
- Dimensions
- Views
- Materialized views et materialized view logs
- Object tables, object types, et object views

Concepts de base relatifs aux schéma de données

Opérateurs internes :

- Sequences
- Synonyms
- Operators

Concepts de base relatifs aux schéma de données

Calculs intégrés dans la base via :

- Du PLSQL :
  - Stored functions, procedures, et packages
  - Et triggers
- Du Java :
  - Java classes, Java resources, et Java sources
- Des librairies externes :
  - External procedure libraries

Mais aussi …

Après la conception logique (fonctionnelle) et le travail sur les tables, le DBA va pouvoir :

- Identifier les index à ajouter et leur type
- Identifier les tables à partitionner
- Identifier les contentions à gérer
- Identifier la segmentation des objets et percevoir les TBS
- Identifier les paramètres à spécialiser pour l'instance Oracle (Vx)
- Identifier les paramètres à spécialiser pour les sessions Oracle (Vx)
- Identifier les tables à « suivre » de près
- Identifier les futures entretiens et leur rythme
- Cibler les exports, en plus des sauvegardes

Concepts de base Oracle vs MySQL et SQL Server

- Une instance Oracle ⇔ un serveur MySQL
- Les objets d'une applications sont
  - Dans les schémas des utilisateurs Oracle de l'application et répartis dans les différents tablespaces eux-mêmes constitués de fichiers de données.
  - Dans une « base de données » (MySQL – un répertoires+fichiers / SQL Serveur 1 à N fichiers DBF)
  - Les utilisateurs ont des droits sur les objets de la/des bases gérées par le serveur.
- SQL Serveur est intrinsèquement lié au système (Windows) – alors que les deux autres sont OS indépendants.
- Le client contact
  - Un Listener Oracle qui va lui indiquer la localisation du serveur
  - Un moteur installé sur une machine (MySQL / SQLServer)

Étapes de la Création d'une système de base de données Oracle

1. Établir les caractéristiques de la base
2. Évaluation du matériel du serveur
3. Installation du logiciel Oracle (serveur et clients)
4. Créer et ouvrir la base de données
5. Sauvegarder la base de données
6. Créer les utilisateurs système
7. Implémenter la structure de la base
8. Sauvegarder la base de données fonctionnelle
9. Optimiser les performances de la base

Étape 1 : établir les caractéristiques de l'instance

- Structure logique globale de la base de données (application) :
  - Que va contenir chacun des tablespace, …
  - Ensemble des objets à définir (users, tables, indexes, …)
- Structure fonctionnelle globale de l'instance :
  - Transactionnelle (ie web, gestion, …)
  - Analytique (ie BO, …)
  - Spatiale (ie ArcView, …)
- Structure physique globale de la base :
  - Où sont placés les datafiles
- Stratégie de sauvegarde :
  - Mise en place des archivelog, …

Étape 2 :Préparation du déploiement : OFA : Oracle Flexible Architecture

- Normalise l'arborescence de déploiement des composants Oracle
  - Convention de nommage des répertoires
    - product : les binaires du moteur par version, les éléments de configuration et de log des listeners dans network/admin et network/log.
    - admin : configuration par base de données (adhoc, arch, adump, create, exp, logbook, pfile),
    - oradata : les fichiers de données
  - Séparation des fichiers d'usage (IO) différents sur différentes partitions, différents médias.

Étape 3.1 : Composants Oracle

- Installation des clients
  - Copie/référencement des drivers JDBC
  - Installation des clients pour tora/toad/…
- Installation du serveur
  - Serveur Oracle
  - Services Net
  - Services d'administration (*managment*)

Étape 3.2 : Composants Oracle Entreprise

- Gestionnaire de sécurité
  - Cryptage des données stockées
  - Cryptage des données en circulation (SSL)
  - Authentification sécurisée & SSO
- Partitionnement des données
- Gestion des données spatiales pour GIS
- Gestion de données transactionnelles (OLAP) ou analytique (Data Mining)

Étape 4 : créer et ouvrir la base

- Via le DCA (Database Creation Assistant)
  - Oracle décide de la plus part des paramètres
  - Une structure par défaut est mise en place…
- À la mano…
  - Permet la customisation de l'ensemble des éléments
  - On part toujours de fichiers types, …
- Via des outils propriétaires
  - Les grandes entreprises ont besoin d'harmoniser les structures et organisation des bases de données Oracle

Étape 5 : sauvegarder la base de données

- Test des procédures de sauvegarde
- Sauvegarde « de référence » avant le déploiement applicatif
- Mise en place des procédures liées à la stratégie de sauvegarde

Étape 6 : créer les espaces de stockage

Création des structures logiques (TableSpaces) et physiques (datafiles) prévues

Étape 7 : créer les utilisateurs système

- Mise en place de la stratégie de sécurité décidée pour la base :
  - Création des rôles :
    - Administrateur
    - Responsables de la structure
    - Responsables des valeurs
    - Pouvant accéder aux données
  - Création des utilisateurs
    - Affectation de leur rôle
    - Affectation des ressources

Étape 8 : implémenter la structure de la base

Création des objets applicatifs de la base

Étape 8 : sauvegarder la base de données fonctionnelle

La sauvegarde complète d'une base dont la structure vient de changer doit être automatique (de l'ordre du reflex)…

(voir) il peut même être préférable de faire également une sauvegarde avant les modifications… tout dépend du contexte

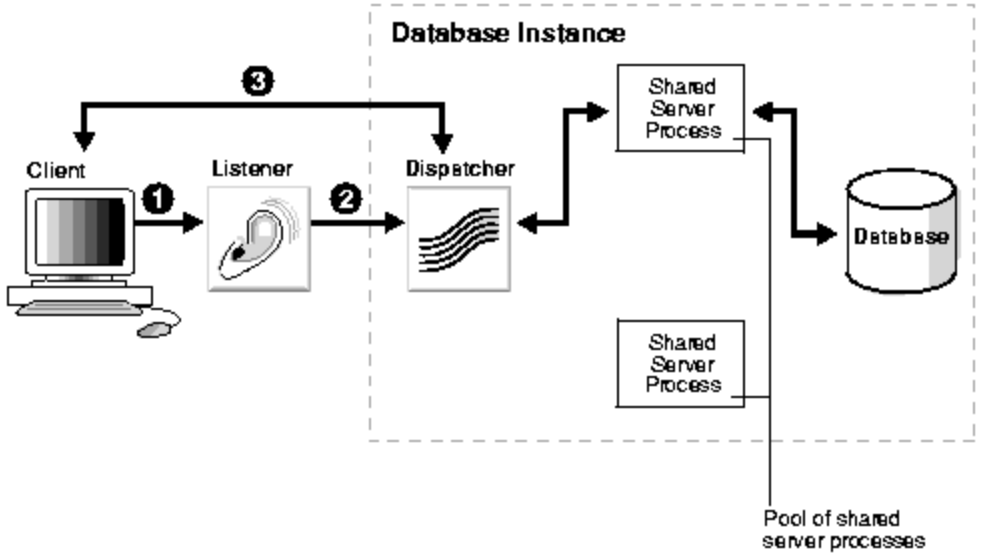
Étape 9 : optimiser les performances de la base

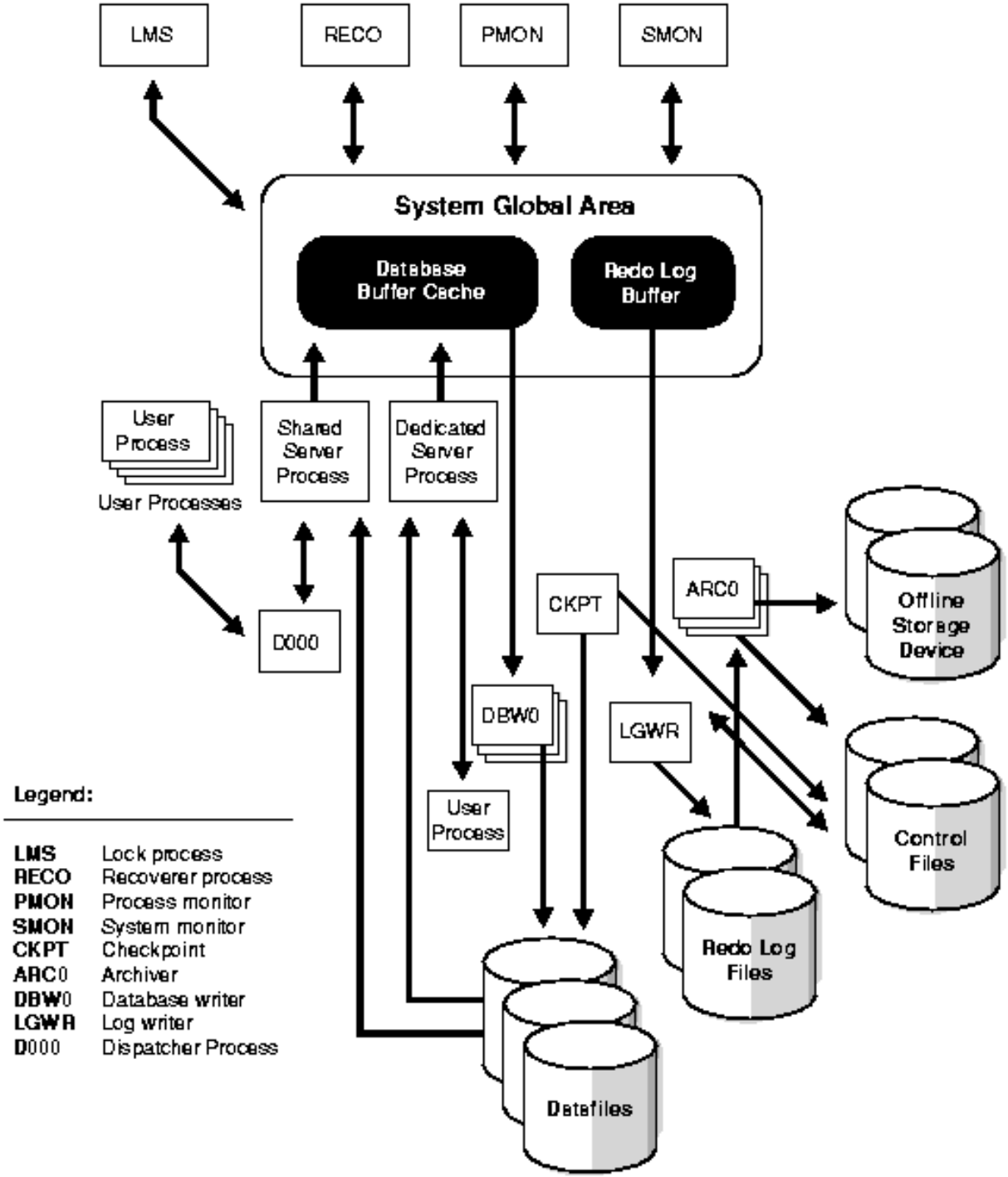
Le processus d'optimisation demande une base en fonctionnement : il se base sur un outillage faisant une analyse dynamique des traitements *réellement* effectués sur la base…

Fonctionnement du moteur

Concepts de base
Services de connexion

- Listener – dispatcher – base de données
  - À l'écoute du réseau pour donner accès à une base
  - N instances possibles sur une machine
    - Versions multiples d'oracle
    - Différentes configurations
  - Bascule automatique
    - Client load balancing
    - Connection load balancing
    - Connect time failover
    - Transparent application failover




LMS
RECO
PMON
SMON
System Global Area
Database Buffer Cache
Redo Log Buffer
User Process
User Processes
Shared Server Process
Dedicated Server Process
D000
CKPT
ARC0
Offline Storage Device
DBW0
LGWR
User Process
Control Files
Redo Log Files
Datafiles
Legend:
LMS Lock process
RECO Recoverer process
PMON Process monitor
SMON System monitor
CKPT Checkpoint
ARC0 Archiver
DBW0 Database writer
LGWR Log writer
D000 Dispatcher Process

Gestion de la performance

Performance ?

- De quelles performance parle-t'on ?
- Quels sont les éléments actifs et quels sont leurs rôles respectifs ?
- Quels sont les mesures de performance significatives sur ces éléments ?
- Quels sont les optimisations possibles, comment les mettre en œuvre ?

Critères de performance…(1/3)

- La notion de performance d'un système indique sa capacité à répondre dans les délais prévus aux questions prévues dans le contexte prévu → la question essentielles est :

  « qu'est-ce qui est prévu ? »…

Critères de performance… (2/3)

Questions principales :

- Combien d'utilisateurs sont prévus ?
- Comment les utilisateurs interagiront-ils ?
- Où sont positionnés les systèmes ?
- Quelle est la vitesse du réseau ?
- Quelle quantité de données seront accédées par les utilisateurs et quelle proportion l'est en principalement en lecture seule ?
- Quel est le temps de réponse exigé par les utilisateurs ?
- Le système doit-il être disponible 24/24 ?
- Tous les changements doivent-ils être répercutés en temps réel ?

Critères de performance… (3/3)

Questions plus liées à un impact budgétaire et de complexité que de tunning :

- Quelle sera la taille de la base de données ?
- Quelle est le format de sortie des transactions ?
- Quelles sont les conditions de disponibilité ?
- Les qualifications (RH) sont-elles disponibles pour construire et administrer et optimiser cette application ?
- Quels sorte de compromis sera édicté par des contraintes budgétaires ?

Quoi regarder ?

1) Code applicatif :

- **Design de la base de données**
- **Codage de l'application**
  - **Efficience du code**
  - **Bonne ordonnancement des taches**

2) l'instance de la base :

- **Réglage des paramètres de mémoire et de buffers**
- **Élimination des blocages**
- **Surveillance des IO sur les données**

3) l'infrastructure sous jacente

- **Système d'exploitation**
- **Ressources matérielles (contention des IO)**

indexes

Rappels sur les indexes
Principes

Espace disque est paginé en pages de 4-8 KO

- Appelés aussi blocs

Une page est une unité de transfert entre disque et RAM

Les pages sont structurées en fichiers

- séquentiels ou directs ou non-ordonnés
- hachés
- Arbres B
- autres

Rappels sur les indexes

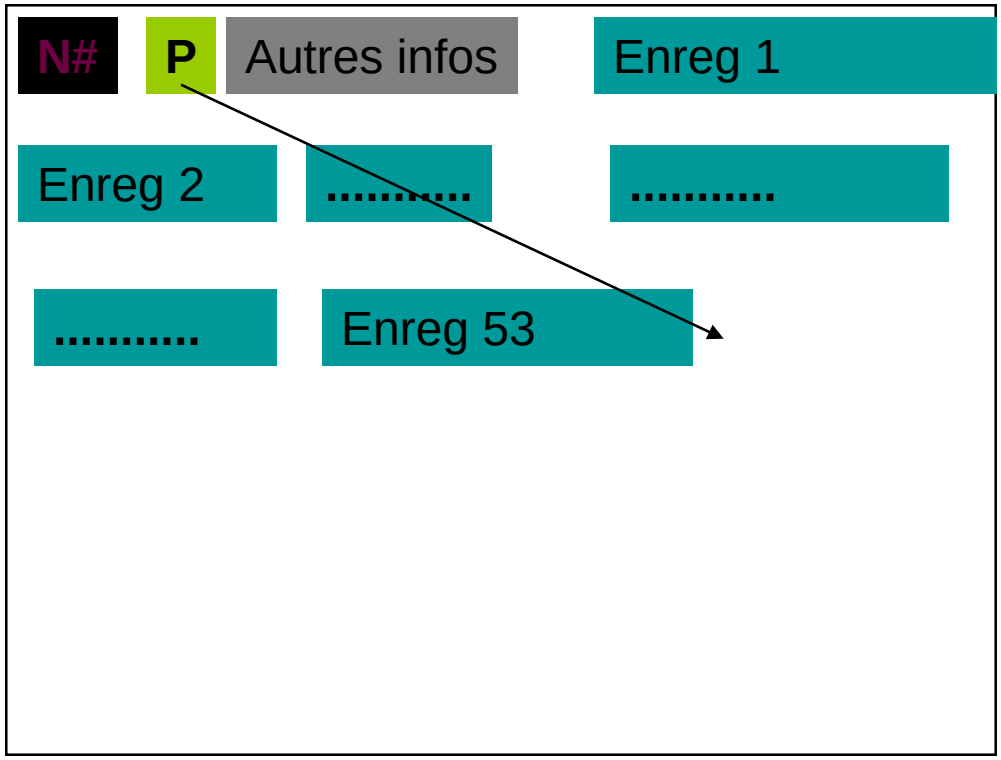
Les indexes sont des fichiers qui contiennent les paires

attribut(s) d'accès -> numéro de page

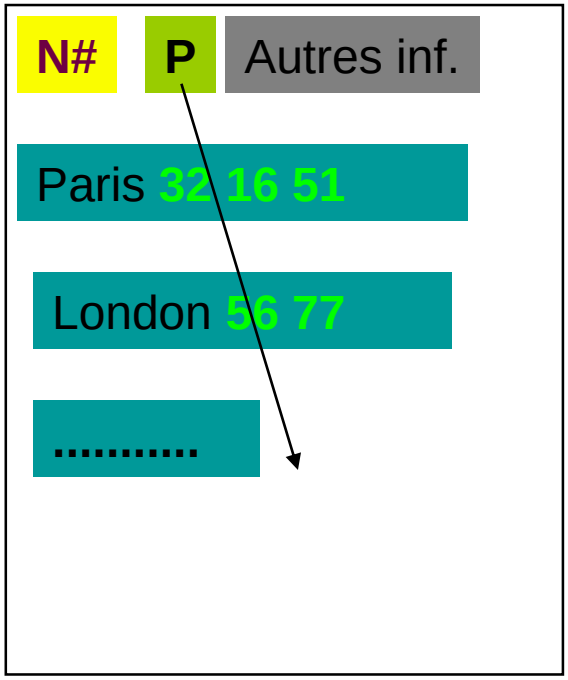
les indexes peuvent être

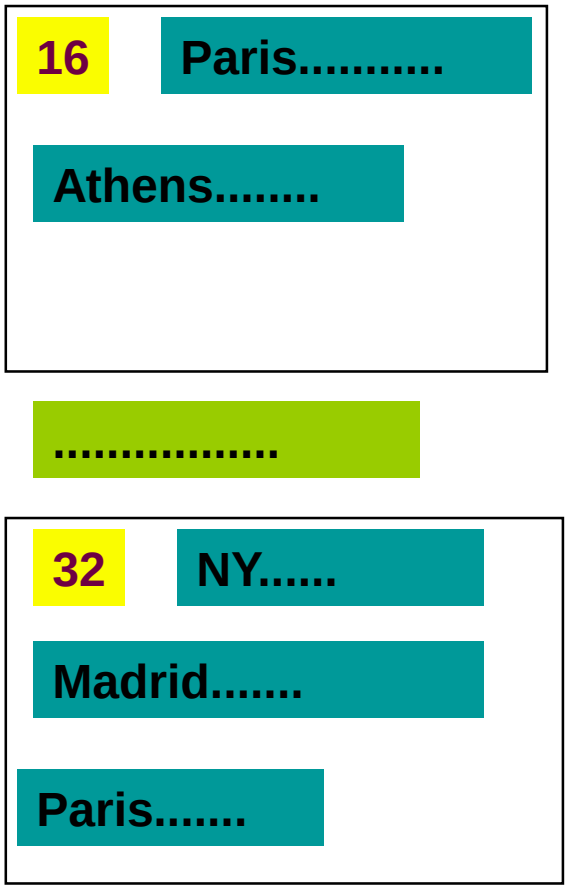
- séquentiels ou directs ou non-ordonnés
- hachés
- arbres B
- Binaires,
- autres

Structure d'une page



Page d'index typique





Accélération de 10:1000 fois selon la capacité d'une page d'index

Fichiers ordonnés

Arbre-B est le plus populaire (tout SGBD)

- En fait l'arbre B^+ lié (linked B^+ - tree de Lehma & Yao)
  - Pour le parcours séquentiel et l'accès concurrent + efficace
    - Loquets individuels sur chaque page parcourue

Une page contient d>>1 paires en ordre lexicographique de clés

- clé - enreg.
- ou clé-pointeur de page avec l'enreg.

Une page qui devient pleine éclate en deux pages semi-pleines

La clé d'éclatement est dupliquée dans la page de niveau supérieur avec les pointeurs vers les nouvelles pages

Arbre B(+)

Un Arbre B d'ordre **m** est un arbre tel que :

1/ Chaque nœud contient k clés triées avec :

- m <= k <= 2m (nœuds non racine)
- 1 <= k <= 2m (nœuds racine)

2/ Chaque chemin de la racine à une feuille est de même longueur **h** (hauteur)

3/ Un nœud est soit :

- Terminal (feuille)
- possède **k+1** fils

Les clés du ième fils ont des valeurs comprises entre les valeurs des (i-1)ème et ième clés du père.

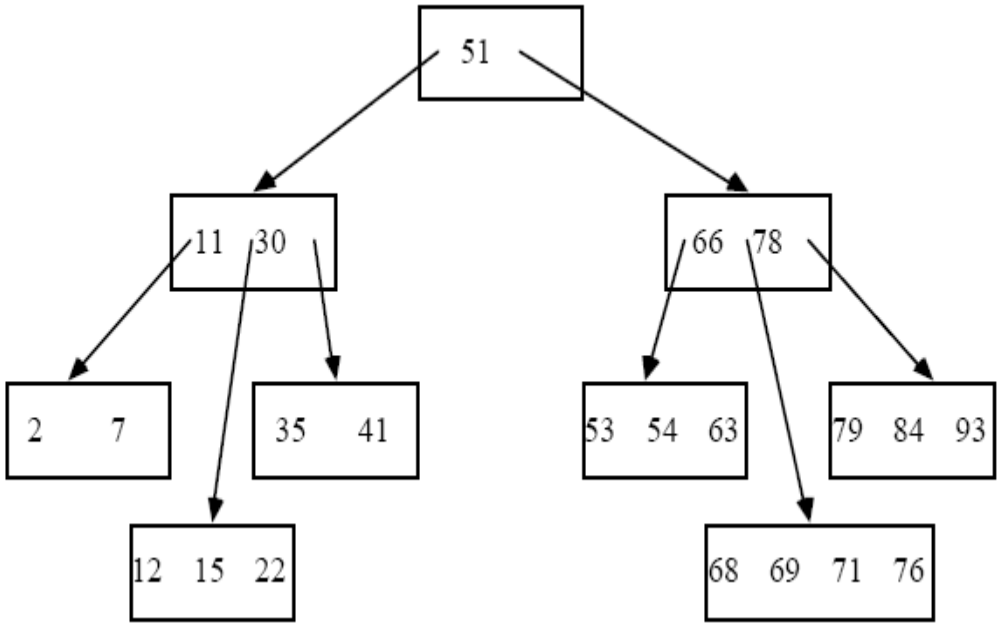
4/ Un arbre B+ est un arbre B dont les feuilles contiennent toutes les clés.

Exemple

Arbre B d'ordre 2

-> Chaque nœud, sauf la racine, contient **k** clés avec 2 <= **k** <= 4.

-> la racine contient **k** clés avec 1 <= **k** <= 4.



Indexes B(+)

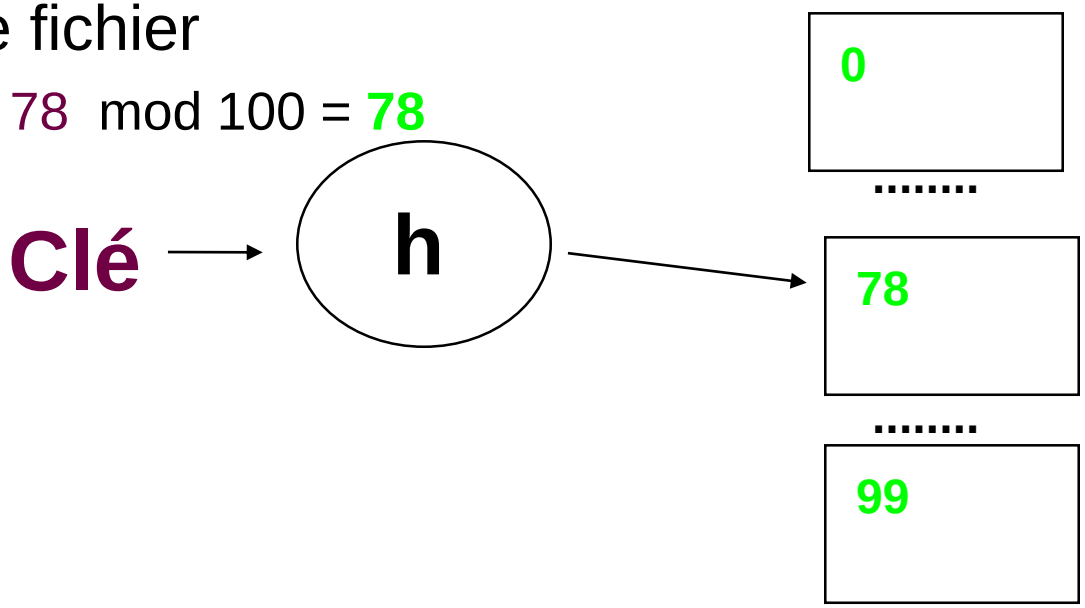
Un index (hash, B+, …) doit avoir une sélectivité (« plage » de clés) de 15 à 20 % maximum pour être intéressant. Si elle est moindre (+ de 20%, alors il est trop coûteux).

L'utilisation de ce type d'indexe est adapté dans le cas d'une forte cardinalité, d'un nombre important de valeurs et de requête de sélectivité réduite (si la requête ramène plus de 5% des données l'indexe ne sera pas systématiquement utilisé / approprié). Donc, particulièrement pour les traitements de type OLTP.

Accès haché

Le n# de page est obtenu par calcul à partir de la clé primaire de fichier

12 34 56 78 mod 100 = **78**



Adapté à des cardinalité moyenne et une bonne répartition afin de ne pas avoir de pages par code de hash trop importante.

Facilite l'accès à une donnée ponctuelle mais pas adaptée à la recherche de plage de données

Index BITMAP

À Chaque valeur possible pour le champ indexé une table de bit correspondant à la présence de la valeur dans l'enregistrement est généré.

À la table trois clés d'indexes seront générés :

Paris : 10001

Montreuil : 01000

Evry : 00110

ID	Ville
1	Paris
2	Montreuil
3	Evry
4	Evry
5	Paris

Donc facilité de calcul (or, and). Mais la mise à jours dans un contexte OLTP génère rapidement des points de blocages sur l'accès à la ressource modifiée.

Donc, adapté à des valeurs à cardinalité réduite et à faible évolutivité ➔ OLAP.

Optimisation du modèle

Principes de l'optimisation du modèle

- Les modélisations sont adapté aux algorithmes implémentés dans les moteurs de SGBD
  - Entité / Relation :
    - adapté au calcul ensembliste *transactionnel*
      - Limitation des redondances,
      - Utilisation de jointures,
      - Cohérence (intégrité) fonctionnelle
- Représente 95 % des bases de données en place

Schéma de base de données optimal ?

Optimisation par la dénormalisation ?

Non si

- Le système peut fonctionner correctement sans dénormalisation, ou
- Les performances du système ne sont pas plus acceptables après la dénormalisation, ou
- Le système sera moins fiable après dénormalisation.

Dénormalisation : précautions d'usage et avantages

Précautions nécessaires :

- Essayer d'avoir une forme denormalisée mise à jours à partir d'une forme normalisée (double l'espace nécessaire),
- Utilisation de triggers ou de programmes pour le maintient automatique de la synchronisation des redondances,
- Utilisation de contraintes et de triggers pour le maintient de la cohérence relative à la redondance

Avantages :

- Optimisation du temps de calcul

Dénormalisation : tables préjointes

Condition :

- Ne contient pas de colonnes redondantes,
- Doit uniquement contenir les colonnes nécessaires au calcul,
- Doit être créées périodiquement via une requête SQL sur les tables normalisées (possible de l'émuler via du code si le SGBD ne les proposent pas en natif).

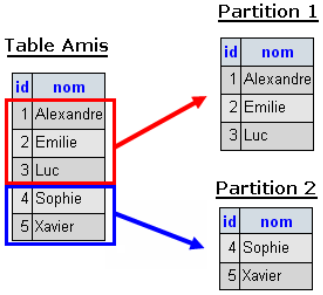
Avantage : pré-calcul des jointures

Désavantage : maintient de la synchronisation

Cette dénormalisation existe via les « vues matérialisées » (Oracle) ou « vues indexées » (SQL Server) dont l'objet est de stocker le résultat de la requête définissant la vue dans une table interne. La définition de la vue matérialisée inclus alors les conditions de mise à jours de la table.

Dénormalisation : tables miroirs / scindées / groupées

Scission Horizontale : split de la table en tranches de données…



- Mise en place du « partitionnement » des tables. Très avantageux pour des grandes tables puisque cela équivaut à construire N tables contenant chacune une partie des données ➔ gestion distinctes des indexes, réduction des collisions, …

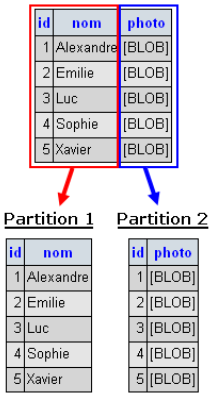
Difficultés :

- trouver la clé de partitionnement la plus pertinente par sa séparation en partitions de taille aussi égales que possibles.
- Deviens très complexe à mettre en œuvre dans le cas de modèles composés d'un grand nombre de tables.

La plus part des SGBD offrent la fonctionnalité. Le partitionnement est alors soit

- Par rapport à une plage (range partitionning) : définition des bornes des différentes partitions (ensemble des données par années, …)
- Par liste : applicable à des attributs à faible cardinalité (répartition de la population par taille, …)
- Par clé de hash :
  - » Soit via la définition d'une fonction de calcul de clé
  - » Soit via l'utilisation d'un calcul de clé de hash laissé au SGBD

Scission Verticale :



- Des colonnes sont placées dans N tables qui partagent la clé primaire…
- Des colonnes sont scindées en plusieurs colonnes afin d'éviter les champs de grande taille (qui souvent mal gérés physiquement par les DBMS).
- Utilisation de vues pour redonner la vision externe d'une seule table.
- Oracle propose le mécanisme de « cluster » de table permettant à des tables qui partagent une même clé primaire et sont souvent consultées en mêmes temps d'être stockée dans les mêmes datablock ce qui en accélèrent les IO.
- Cas d'usage : gestion des BLOB par oracle ==> mieux vaux avoir 2 x 4000 au lieu de 8000...

Vue matérialisé : exemple

```
CREATE MATERIALIZED VIEW cust_mth_sales_mv
BUILD IMMEDIATE
REFRESH FAST ON DEMAND
ENABLE QUERY REWRITE AS
SELECT s.time_id, s.prod_id, SUM(s.quantity_sold), SUM(s.amount_sold),
       p.prod_name, t.calendar_month_name, COUNT(*),
       COUNT(s.quantity_sold), COUNT(s.amount_sold)
FROM sales s, products p, times t
WHERE  s.time_id = t.time_id AND s.prod_id = p.prod_id
GROUP BY t.calendar_month_name, s.prod_id, p.prod_name, s.time_id;
```

```
BEGIN
  DBMS_REFRESH.make(
    name                 => 'SH.MINUTE_REFRESH',
    list                 => '',
    next_date            => SYSDATE,
    interval             => '/*1:Mins*/ SYSDATE + 1/(60*24)',
    implicit_destroy     => FALSE,
    lax                  => FALSE, (permet chg group)
    job                  => 0,
    rollback_seg         => NULL,
    push_deferred_rpc    => TRUE,
    refresh_after_errors => TRUE,
    purge_option         => NULL,
    parallelism          => NULL,
    heap_size            => NULL);
END;
/
```

```
BEGIN
  DBMS_REFRESH.add(
    name => 'SCOTT.MINUTE_REFRESH',
    list => 'SCOTT.EMP_MV',
    lax  => TRUE);
END;
/

EXEC DBMS_MVIEW.refresh('EMP_MV');
```

Dénormalisation : tables de reporting

Condition :

- Une colonne par colonne du rapport
- Séquencement physique propre
- Ne pas recomposer des champs (1FN)

Permet le pré-calcul (batch) des rapports

Utilisé dans les outils/projets analytiques (par exemple, reportServer de la suite Microsoft)

- Via les mats views...

Dénormalisation : tables miroirs / scindées / groupées

Tables Miroir :

- Demande, comme pour les tables pré-jointes, la mise en place d'une synchronisation d'une table « maitre » vers une table répliquée.
- Permet d'avoir différents besoins (analytiques/transactionnels) sur une même base via la spécialisation des tables par usage
- Peut être géré via les vues matérialisées (ou équivalent)

Regroupement des tables ayant des relations 1 – 1 voir 1 – N dans une table unique mais doit dans ce dernier cas être analysé (cf formes normales).

Dénormalisation : données redondantes et groupes répétés

La redondance des données si elle permet d'éviter des jointures systématiques et si

- Peut de colonnes sont nécessaires,
- Les colonnes sont stables rarement modifiées,
- Utilisées par beaucoup d'utilisateurs ou par peut d'utilisateurs critiques.

Les groupes répétés (supprime une table d'association) si :

- Les données sont rarement voir jamais agrégées,
- Les données se présentent statistiquement selon un pattern correct
- Le nombre d'occurrence est stable,
- Les données sont accédées collectivement,
- Le pattern d'insertion ou d'effacement est prédictible.

Cas d'usage :

- Dimensions d'une base BI

Dénormalisation : stockage des données calculées/dérivées

Pré-calcul à condition que :

- Les données sources soient relativement stables,
- Le coût de calcul des dérivation est élevé,
- L'utilisation des tables sources permet un recalcul rapide en cas de modification.

Cas d'usage :

- Datamarts des bases BI

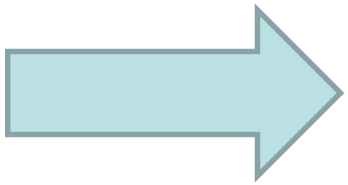
Dénormalisation : parcours de hiérarchie

Construction d'une table de pré-calcul des niveaux hiérarchiques

- Contient un enregistrement par relation hiérarchique, quel qu'en soit le niveau.
- Peut intégrer des indications supplémentaires comme le statut de feuille…

Demande une génération par programme car impossible en SQL.

ID	IDPARENT	DESCRIPTION
1	null	Produits frais
2	1	Produits laitiers
3	1	Légumes
4	1	Fuits
5	2	Yaourts
6	2	Fromages
7	6	Camembert
8	4	Orange
9	8	Orange sanguines
10	4	Bananes
11	10	Bananes Planteur



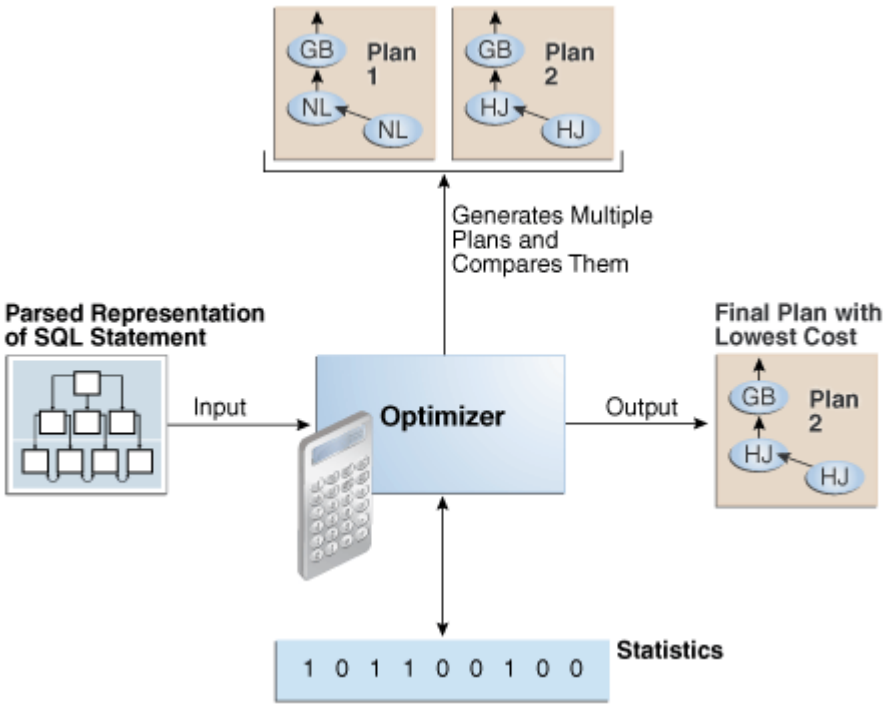
ID	ID_N1	ID_N2	ID_N3	DESCRIPTION
1	null	null	null	Produits Frais
2	1	null	null	Produits laitiers
3	1	null	null	Légumes
4	1	null	null	Fuits
5	1	2	null	Yaourts
6	1	2	null	Fromages
7	1	2	6	Camembert
8	1	4	null	Orange
9	1	4	8	Orange sanguines
10	1	4	null	Bananes
11	1	4	10	Bananes Planteur

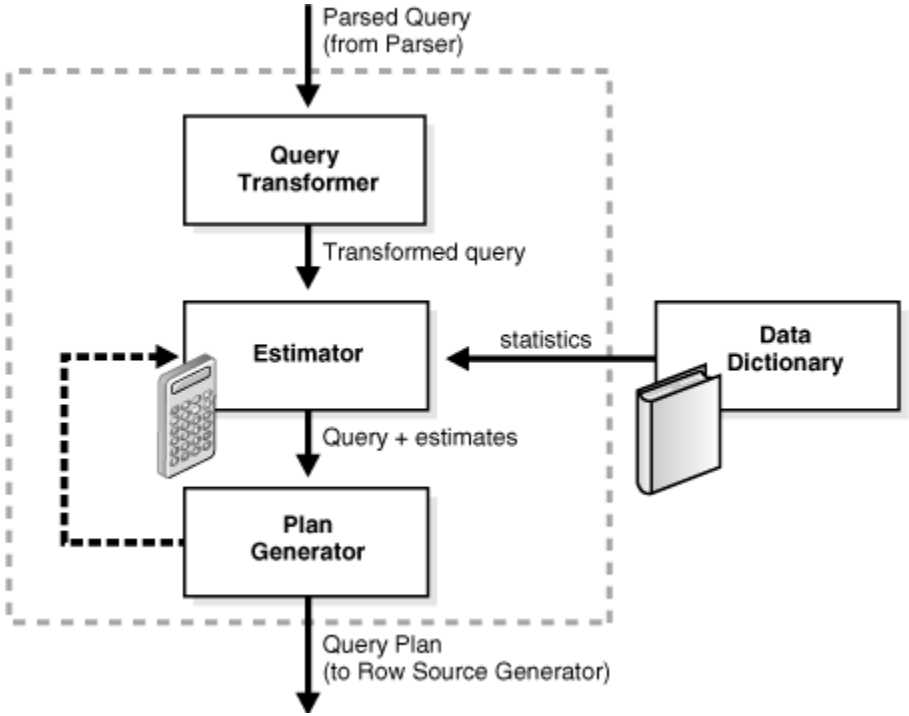
Dénormalisation : résumé

Dénormalisation	**A utiliser lorsque**
Tables préjointes	Le coût des jointures est prohibitif
Reporting	Des rapports spécifiques et critiques sont nécessaires
Miroir	Des données sont accédées simultanément par # env.
Scission	Des groupes distincts accèdent à des éléments distincts
Combinaison	Consolidation de relation 1-1 ou 1-N dans une table unique
Redondance	Réduction du nombre de jointure nécessaires
Répétition de groupes	Réduction du nombre d'I/O et (potentiellement) de l'espace de stockage nécessaire
Dérivation	Transforme le calcul dynamique en calcul batch
Speed table	Gain en efficacité de parcours des hiérarchies

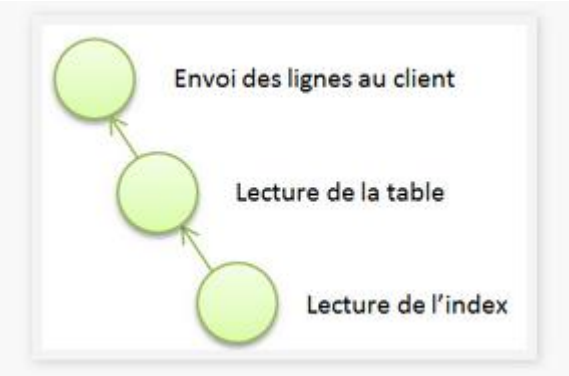
Analyse de la performance

Optimizer Oracle



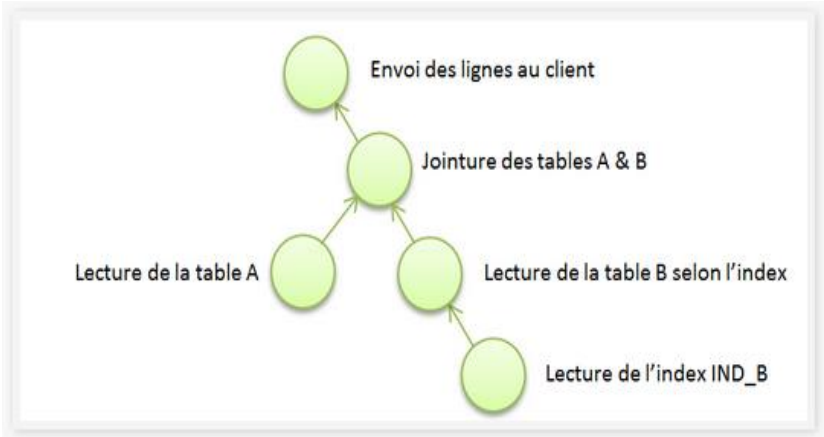


Plan d'executions

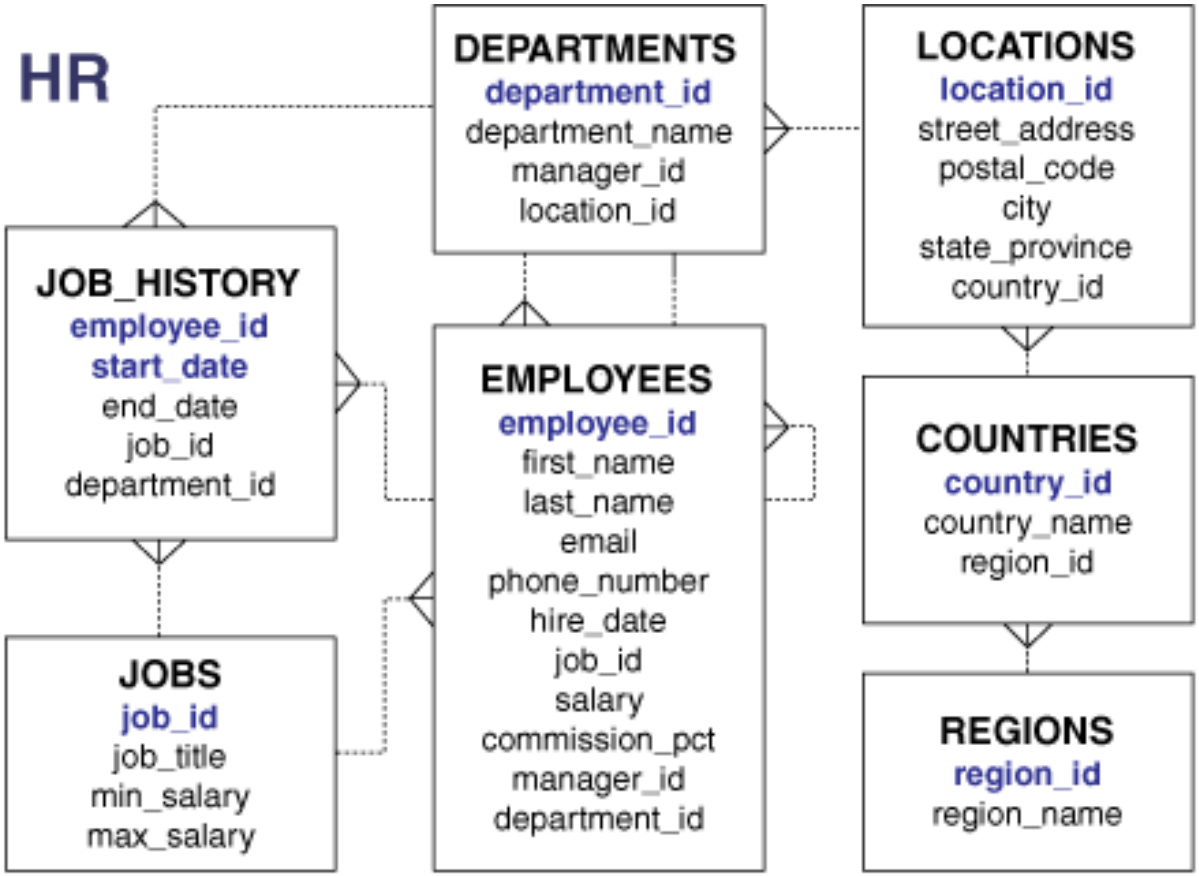


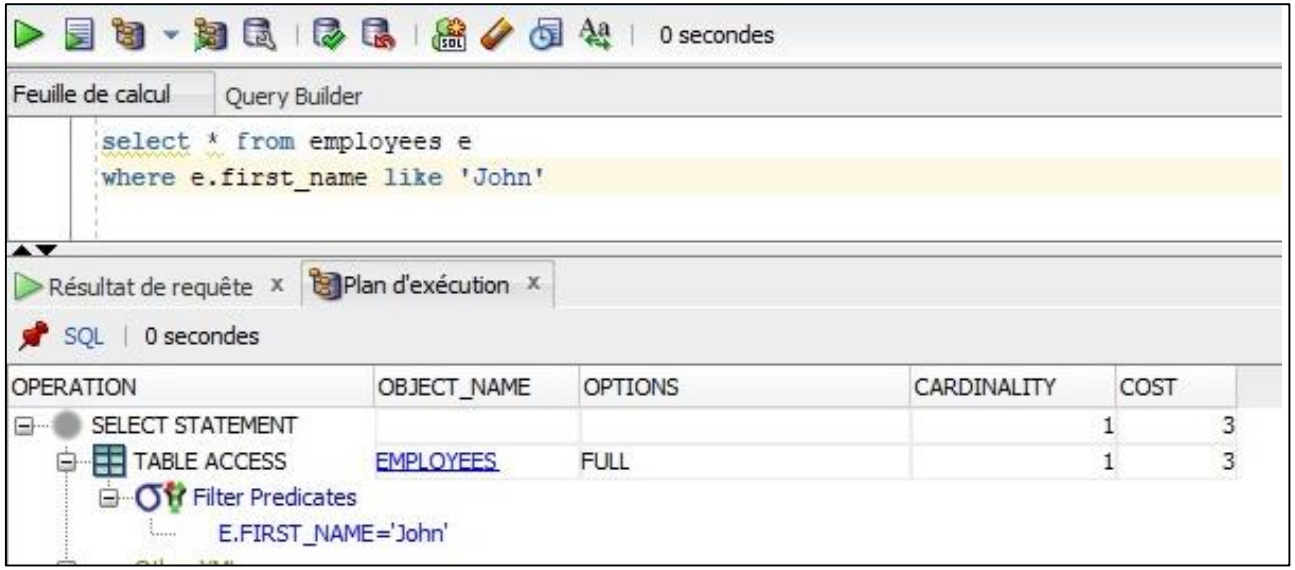

```
1  ----------------------------------------------------
2  | Id| Operation                    | Name          |
3  ----------------------------------------------------
4  | 0 | SELECT STATEMENT             |               |
5  | 1 |  TABLE ACCESS BY INDEX ROWID| CONSULTANT     |
6  | 2 |   INDEX RANGE SCAN           | CONSULTANT_PK |
7  ----------------------------------------------------
```

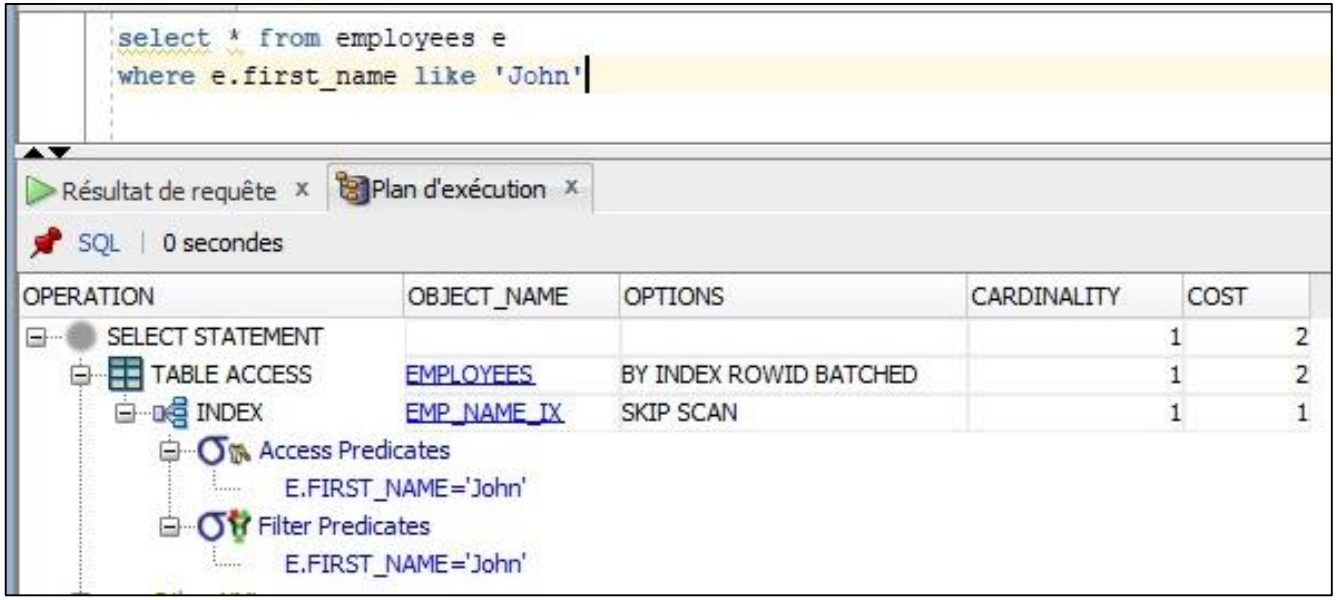
```sql
SELECT DEBUT
FROM CONSULTANT C
INNER JOIN EN_MISSION M on C.ID_CONSULTANT=M.ID_CONSULTANT
WHERE C.PRENOM='Luke';
```



Plan d'exécutions (par l'exemple)

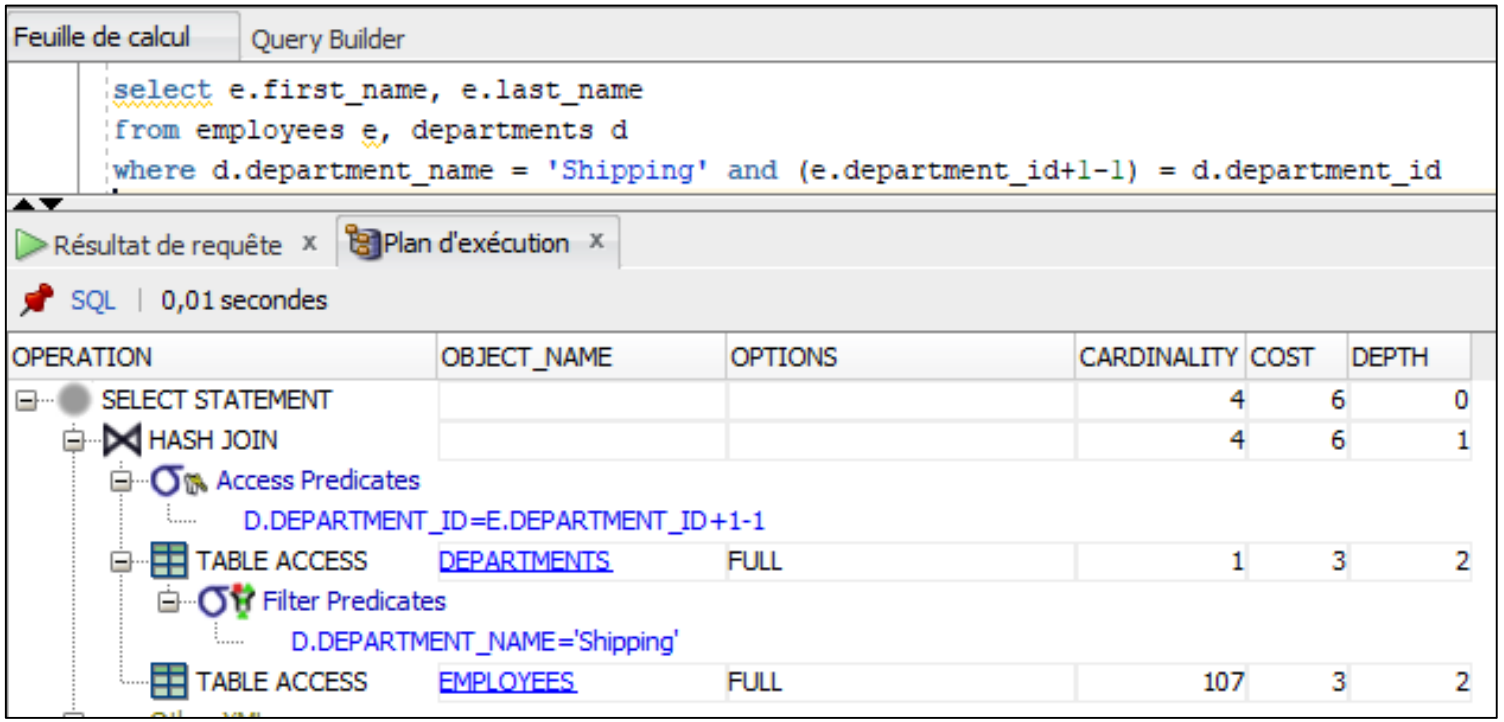



0 secondes
Feuille de calcul
Query Builder
select * from employees e
where e.first_name like 'John'
Résultat de requête
Plan d'exécution
SQL | 0 secondes
OPERATION
OBJECT_NAME
OPTIONS
CARDINALITY
COST
SELECT STATEMENT
1
3
TABLE ACCESS
EMPLOYEES
FULL
1
3
Filter Predicates
E.FIRST_NAME='John'


select * from employees e
where e.first_name like 'John'
Résultat de requête
Plan d'exécution
SQL | 0 secondes
OPERATION
OBJECT_NAME
OPTIONS
CARDINALITY
COST
SELECT STATEMENT
1
2
TABLE ACCESS
EMPLOYEES
BY INDEX ROWID BATCHED
1
2
INDEX
EMP_NAME_IX
SKIP SCAN
1
1
Access Predicates
E.FIRST_NAME='John'
Filter Predicates
E.FIRST_NAME='John'

Plan d'exécutions

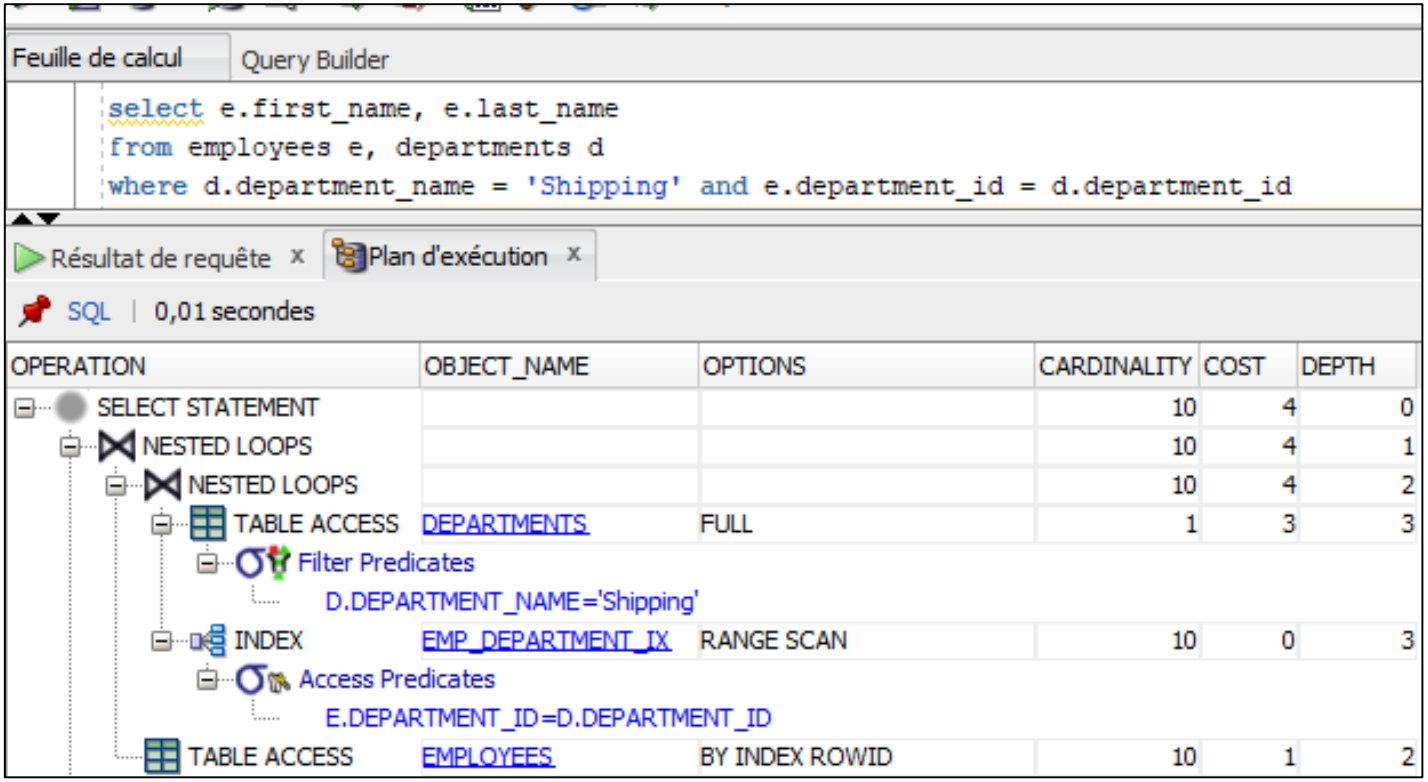
Membres d'un département



Aucun index utilisé

Plan d'exécutions

Membres d'un département



L'utilisation de l'index sur l'ID département est possible.

Plan d'exécutions

Membres d'un département

Feuille de calcul | Query Builder

```
select e.first_name, e.last_name
from employees e, departments d
where d.department_name = 'Shipping' and e.department_id = d.department_id
```

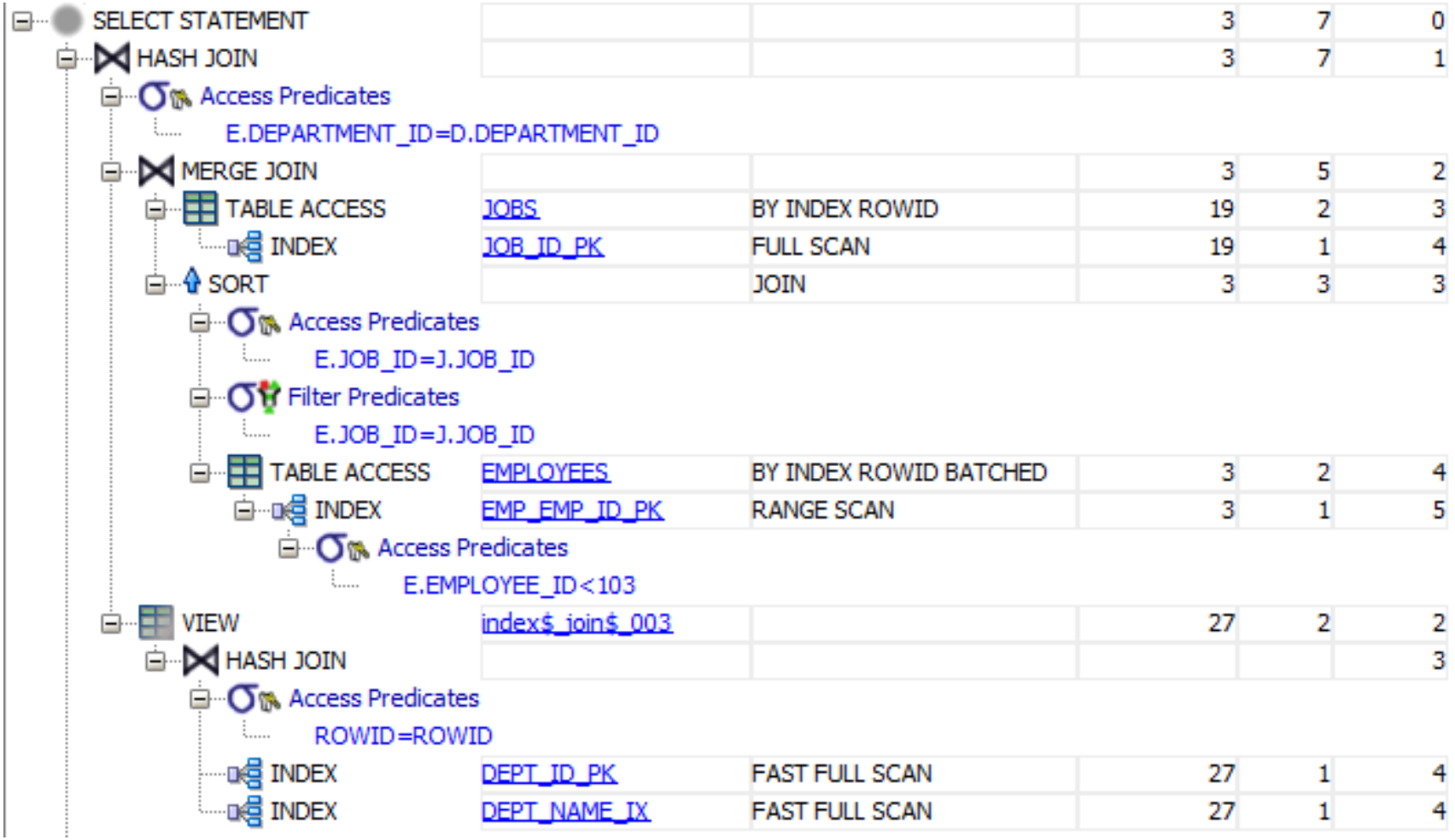
Résultat de requête x | Plan d'exécution x

SQL | 0 secondes

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	DEPTH
SELECT STATEMENT			10	3	0
NESTED LOOPS			10	3	1
NESTED LOOPS			10	3	2
TABLE ACCESS	DEPARTMENTS	BY INDEX ROWID BATCHED	1	2	3
INDEX	DEPT_NAME_IX	RANGE SCAN	1	1	4
Access Predicates					
D.DEPARTMENT_NAME='Shipping'					
INDEX	EMP_DEPARTMENT_IX	RANGE SCAN	10	0	3
Access Predicates					
E.DEPARTMENT_ID=D.DEPARTMENT_ID					
TABLE ACCESS	EMPLOYEES	BY INDEX ROWID	10	1	2

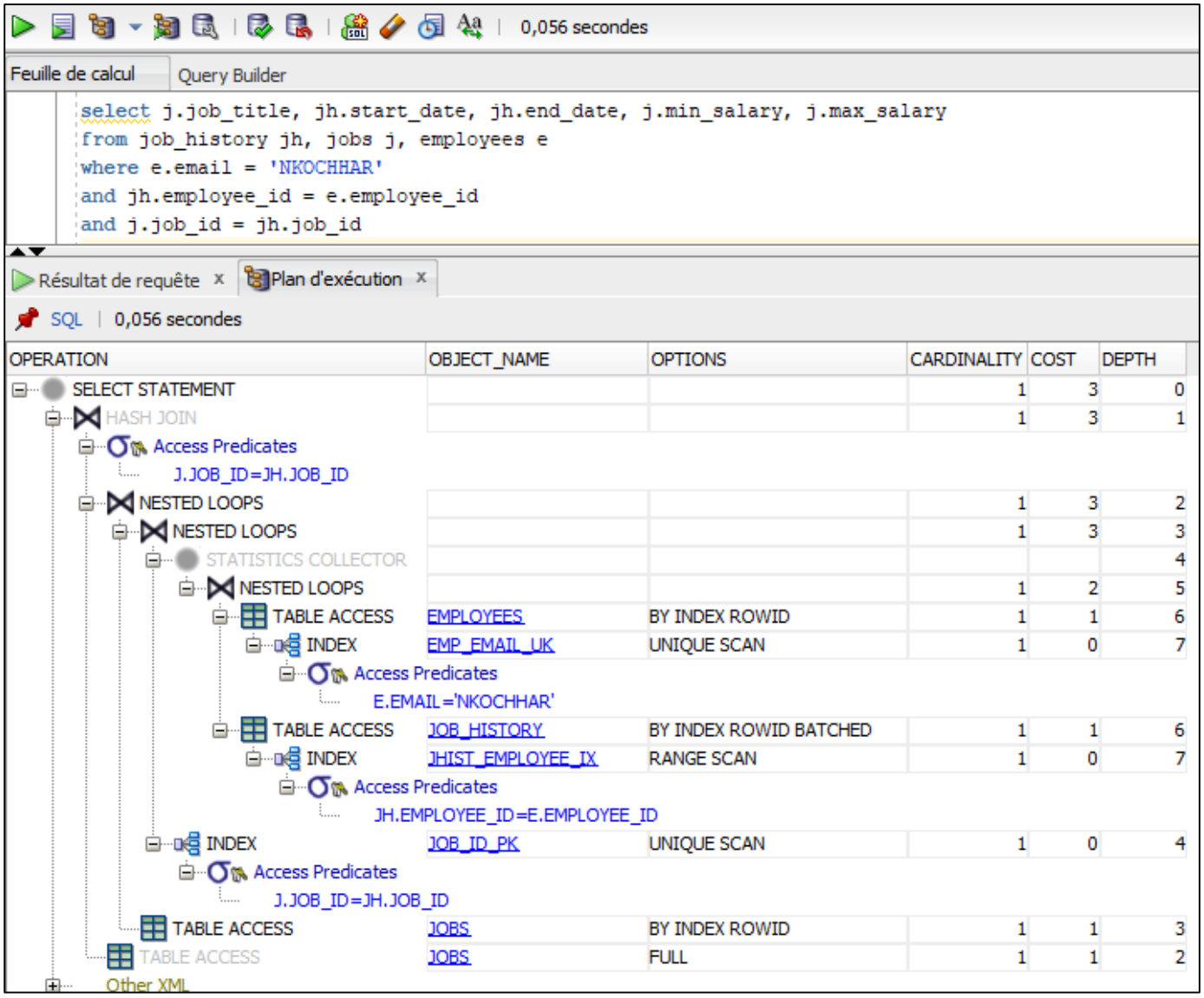
Index placé sur le nom du département

```
5 - SELECT e.employee_id, j.job_title, e.salary, d.department_name
  -   FROM employees e, jobs j, departments d
1 -   WHERE  e.employee_id < 103                    } 3
2 -     AND e.job_id = j.job_id                    } 4
        AND e.department_id = d.department_id;
```



Plan d'exécutions

Historique des jobs d'un employé



0,056 secondes

Feuille de calcul | Query Builder

```
select j.job_title, jh.start_date, jh.end_date, j.min_salary, j.max_salary
from job_history jh, jobs j, employees e
where e.email = 'NKOCHHAR'
and jh.employee_id = e.employee_id
and j.job_id = jh.job_id
```

Résultat de requête x | Plan d'exécution x

SQL | 0,056 secondes

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	DEPTH
SELECT STATEMENT			1	3	0
HASH JOIN			1	3	1
Access Predicates					
J.JOB_ID=JH.JOB_ID					
NESTED LOOPS			1	3	2
NESTED LOOPS			1	3	3
STATISTICS COLLECTOR					4
NESTED LOOPS			1	2	5
TABLE ACCESS	EMPLOYEES	BY INDEX ROWID	1	1	6
INDEX	EMP_EMAIL_UK	UNIQUE SCAN	1	0	7
Access Predicates					
E.EMAIL='NKOCHHAR'					
TABLE ACCESS	JOB_HISTORY	BY INDEX ROWID BATCHED	1	1	6
INDEX	JHIST_EMPLOYEE_IX	RANGE SCAN	1	0	7
Access Predicates					
JH.EMPLOYEE_ID=E.EMPLOYEE_ID					
INDEX	JOB_ID_PK	UNIQUE SCAN	1	0	4
Access Predicates					
J.JOB_ID=JH.JOB_ID					
TABLE ACCESS	JOBS	BY INDEX ROWID	1	1	3
TABLE ACCESS	JOBS	FULL	1	1	2
Other XML					

Plan d'exécutions

Somme des ventes faites à des clients dont le nom commence par A

```
select count(s.amount_sold) from sales s, CUSTOMERS c where c.cust_first_name like 'A%' and s.cust_id = c.cust_id;
```

Résultat de requête | Plan d'exécution

SQL | 0,054 secondes

OPERATION	OBJECT_NAME	OPTIONS	...	...	...	CARDINALITY	COST
SELECT STATEMENT						1	939
SORT		AGGREGATE				1	
HASH JOIN						178304	939
Access Predicates							
S.CUST_ID=C.CUST_ID							
TABLE ACCESS	CUSTOMERS	FULL				1370	423
Filter Predicates							
C.CUST_FIRST_NAME LIKE 'A%'							
PARTITION RANGE		ALL	1	...	4	918843	514
TABLE ACCESS	SALES	FULL	1	...	4	918843	514
Other XML							

Aucun index disponible

Plan d'exécutions

Somme des ventes faites à des clients dont le nom commence par A

```
select count(s.amount_sold) from sales s, CUSTOMERS c where c.cust_first_name like 'A%' and s.cust_id = c.cust_id;
```

Résultat de requête | Plan d'exécution

SQL | 0 secondes

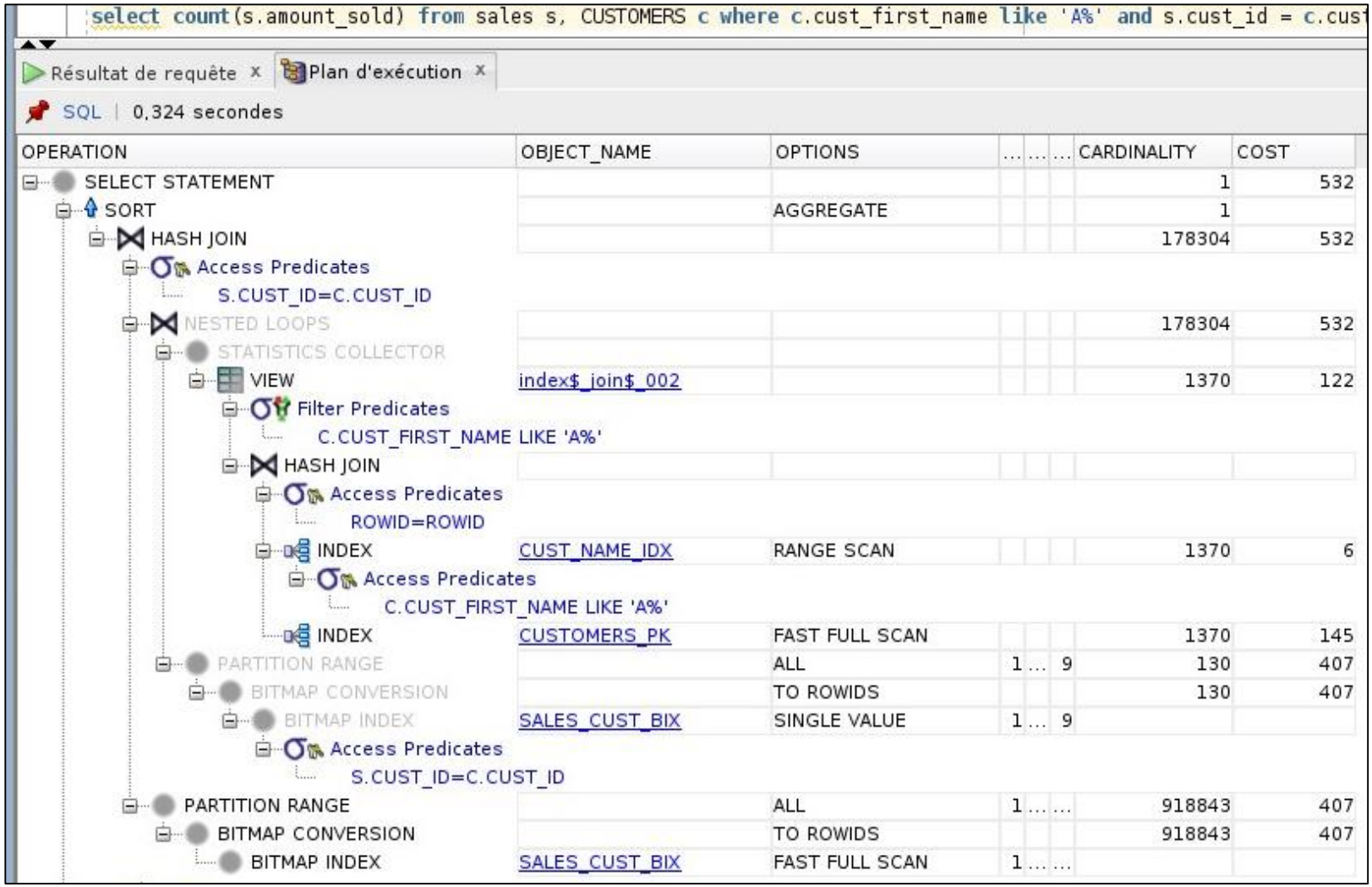
OPERATION	OBJECT_NAME	OPTIONS	...	...	...	CARDINALITY	COST
SELECT STATEMENT						1	832
SORT		AGGREGATE				1	
HASH JOIN						178304	832
Access Predicates							
S.CUST_ID=C.CUST_ID							
NESTED LOOPS						178304	832
STATISTICS COLLECTOR							
TABLE ACCESS	CUSTOMERS	FULL				1370	423
Filter Predicates							
C.CUST_FIRST_NAME LIKE 'A%'							
PARTITION RANGE		ALL	1	...	6	130	407
BITMAP CONVERSION		TO ROWIDS				130	407
BITMAP INDEX	SALES_CUST_BIX	SINGLE VALUE	1	...	6		
Access Predicates							
S.CUST_ID=C.CUST_ID							
PARTITION RANGE		ALL	1	...	9	918843	407
BITMAP CONVERSION		TO ROWIDS				918843	407
BITMAP INDEX	SALES_CUST_BIX	FAST FULL SCAN	1	...	9		

Index bitmap sur l'ID des clients dans la table des ventes

Gain : 12 %

Plan d'exécutions

Somme des ventes faites à des clients dont le nom commence par A



```
select count(s.amount_sold) from sales s, CUSTOMERS c where c.cust_first_name like 'A%' and s.cust_id = c.cust
```

Résultat de requête | Plan d'exécution

SQL | 0,324 secondes

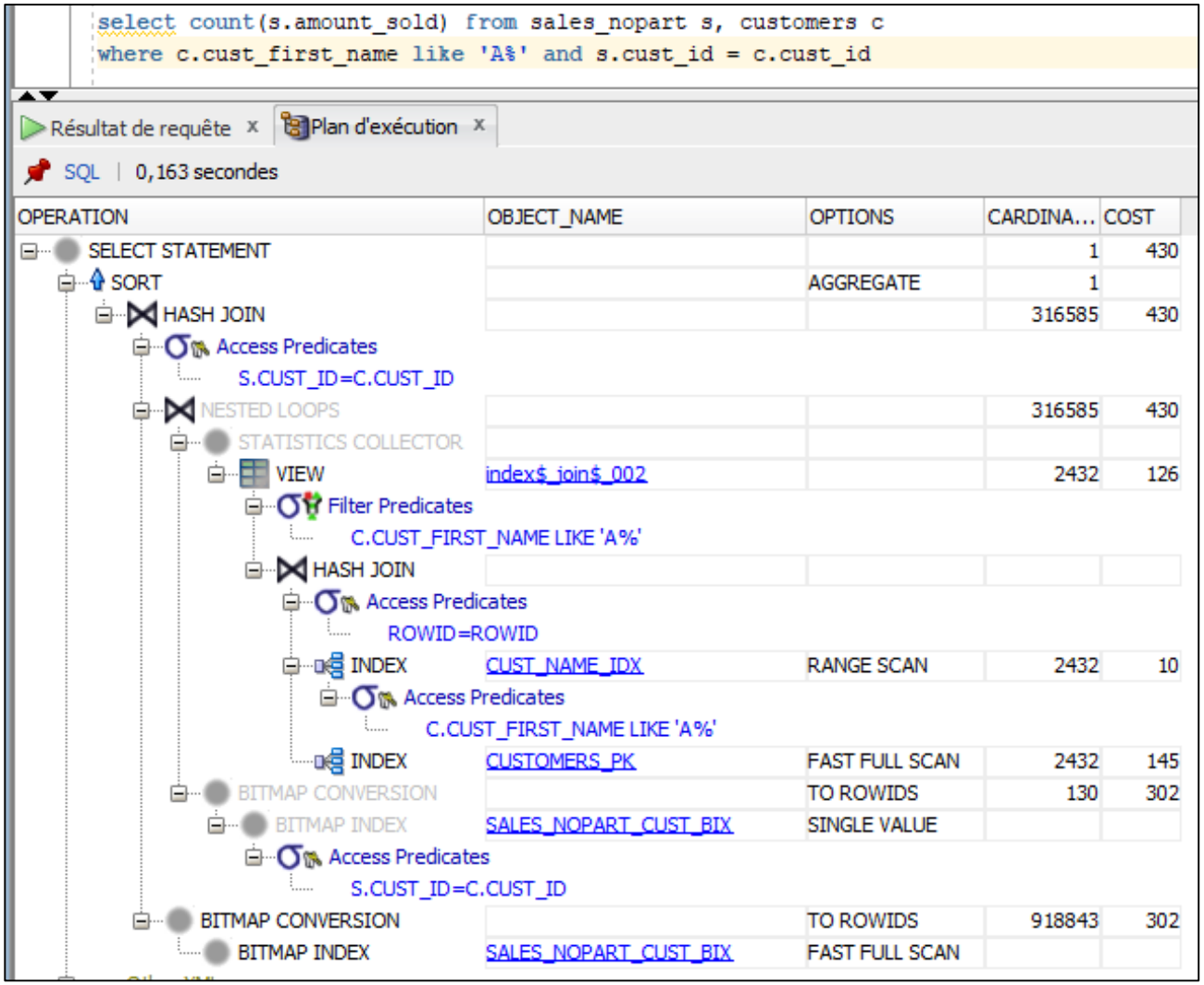
OPERATION	OBJECT_NAME	OPTIONS	...	...	...	CARDINALITY	COST
SELECT STATEMENT						1	532
SORT		AGGREGATE				1	
HASH JOIN						178304	532
Access Predicates							
S.CUST_ID=C.CUST_ID							
NESTED LOOPS						178304	532
STATISTICS COLLECTOR							
VIEW	index$_join$_002					1370	122
Filter Predicates							
C.CUST_FIRST_NAME LIKE 'A%'							
HASH JOIN							
Access Predicates							
ROWID=ROWID							
INDEX	CUST_NAME_IDX	RANGE SCAN				1370	6
Access Predicates							
C.CUST_FIRST_NAME LIKE 'A%'							
INDEX	CUSTOMERS_PK	FAST FULL SCAN				1370	145
PARTITION RANGE		ALL	1	...	9	130	407
BITMAP CONVERSION		TO ROWIDS				130	407
BITMAP INDEX	SALES_CUST_BIX	SINGLE VALUE	1	...	9		
Access Predicates							
S.CUST_ID=C.CUST_ID							
PARTITION RANGE		ALL	1	...	...	918843	407
BITMAP CONVERSION		TO ROWIDS				918843	407
BITMAP INDEX	SALES_CUST_BIX	FAST FULL SCAN	1	...	...		

Index sur le nom client dans la table des clients

Gain total : 44 %

Plan d'exécutions

Somme des ventes faites à des clients dont le nom commence par A



```
select count(s.amount_sold) from sales_nopart s, customers c
where c.cust_first_name like 'A%' and s.cust_id = c.cust_id
```

Résultat de requête | Plan d'exécution

SQL | 0,163 secondes

OPERATION	OBJECT_NAME	OPTIONS	CARDINA...	COST
SELECT STATEMENT			1	430
SORT		AGGREGATE	1	
HASH JOIN			316585	430
Access Predicates				
S.CUST_ID=C.CUST_ID				
NESTED LOOPS			316585	430
STATISTICS COLLECTOR				
VIEW	index$_join$_002		2432	126
Filter Predicates				
C.CUST_FIRST_NAME LIKE 'A%'				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
INDEX	CUST_NAME_IDX	RANGE SCAN	2432	10
Access Predicates				
C.CUST_FIRST_NAME LIKE 'A%'				
INDEX	CUSTOMERS_PK	FAST FULL SCAN	2432	145
BITMAP CONVERSION		TO ROWIDS	130	302
BITMAP INDEX	SALES_NOPART_CUST_BIX	SINGLE VALUE		
Access Predicates				
S.CUST_ID=C.CUST_ID				
BITMAP CONVERSION		TO ROWIDS	918843	302
BITMAP INDEX	SALES_NOPART_CUST_BIX	FAST FULL SCAN		

Changement de modèle : Table ventes non partitionnée

Gain total : 55%

```
SQL Text    : select count(s.amount_sold) from sales_nopart s, customers c
              where c.cust_first_name like 'A%' and s.cust_id = c.cust_id

-------------------------------------------------------------------------------
FINDINGS SECTION (1 finding)
-------------------------------------------------------------------------------

1- Index Finding (see explain plans section below)
--------------------------------------------------
  Le plan d'exécution de cette instruction peut être amélioré en créant un ou
  plusieurs index.

  Recommendation (estimated benefit: 64.22%)
  ------------------------------------------
  - Envisagez d'exécuter Access Advisor pour améliorer la conception du
    schéma physique ou de créer l'index recommandé.
    create index SH.IDX$$_00970001 on SH.CUSTOMERS("CUST_FIRST_NAME","CUST_ID")
    ;

  - Envisagez d'exécuter Access Advisor pour améliorer la conception du
    schéma physique ou de créer l'index recommandé.
    create index SH.IDX$$_00970002 on SH.SALES_NOPART("CUST_ID");

  Rationale
  ---------
    La création des index recommandés améliore de façon considérable le plan
    d'exécution de cette instruction. Il pourrait cependant être préférable
    d'exécuter "Access Advisor" en utilisant une charge globale SQL
    représentative contrairement à une seule instruction. Ceci permettra
    d'obtenir des recommandations d'index complètes prenant en compte le coût
    de maintenance des index et de la consommation d'espace supplémentaire.
```

```
1- Original
-----------
Plan hash value: 1184928097

-----------------------------------------------------------------------------------------
| Id  | Operation           | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |               |     1 |    17 |  1654   (1)| 00:00:01 |
|   1 |  SORT AGGREGATE     |               |     1 |    17 |            |          |
|*  2 |   HASH JOIN         |               |   316K|  5255K|  1654   (1)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| CUSTOMERS     |  2432 | 29184 |   423   (1)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| SALES_NOPART  |   918K|  4486K|  1229   (1)| 00:00:01 |
-----------------------------------------------------------------------------------------

2- Using New Indices
--------------------
Plan hash value: 1444798750

-----------------------------------------------------------------------------------------
| Id  | Operation           | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |               |     1 |    17 |   592   (2)| 00:00:01 |
|   1 |  SORT AGGREGATE     |               |     1 |    17 |            |          |
|*  2 |   HASH JOIN         |               |   316K|  5255K|   592   (2)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN | IDX$$_00970001|  2432 | 29184 |    10   (0)| 00:00:01 |
|   4 |    INDEX FAST FULL SCAN| IDX$$_00970002|   918K|  4486K|   579   (1)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

Avec les indexes positionnés :

```
-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                     |     1 |    17 |   430   (1)| 00:00:01 |
|   1 |  SORT AGGREGATE              |                     |     1 |    17 |            |          |
|*  2 |   HASH JOIN                  |                     |   316K|  5255K|   430   (1)| 00:00:01 |
|*  3 |    VIEW                      | index$_join$_002    |  2432 | 29184 |   126   (0)| 00:00:01 |
|*  4 |     HASH JOIN                |                     |       |       |            |          |
|*  5 |      INDEX RANGE SCAN        | CUST_NAME_IDX       |  2432 | 29184 |    10   (0)| 00:00:01 |
|   6 |      INDEX FAST FULL SCAN    | CUSTOMERS_PK        |  2432 | 29184 |   145   (0)| 00:00:01 |
|   7 |    BITMAP CONVERSION TO ROWIDS|                    |   918K|  4486K|   302   (0)| 00:00:01 |
|   8 |     BITMAP INDEX FAST FULL SCAN| SALES_NOPART_CUST_BIX|     |       |            |          |
-----------------------------------------------------------------------------------------
```

Rappel : jointures

Les jointures représentent le croisement de plusieurs tables dans une requête.

Les jointures « internes » (ou naturelles, ou par défaut), ont comme effet qu'a un enregistrement d'une table, correspond naturellement un enregistrement de l'autre.

Par exemple : je veux l'ensemble des employés et leur département d'appartenance.

```
SELECT ENAME, DNAME FROM EMP E, DEPT D
    WHERE D.DEPTNO = E.DEPTNO
```

retournera tous les employés et leur département d'appartenance.

Mais que ce passe-t'il si des employés ne sont pas encore dans un département ? il n'apparaissent pas dans les résultats de la requête.

Si on veux voir ces employés, il faut faire des jointures externes.

Rappel : jointures

Les jointures externes vont prendre en considération tous les enregistrements d'une table (dite directrice) même ceux qui n'ont pas de correspondants avec la table jointe. Il y a alors trois sortes de jointures externes : gauche, droite, toutes selon que ce sont les engistrements de la table de gauche, de droite ou des deux.

Par exemple :

Si on veux tous les employés même ceux qui n'ont pas de département :

Select ename, dname **from** emp e **left outer join** dept d **on** e.deptno = d.deptno;

Si on veux tous les départements même ceux qui n'ont pas d'employés :

Select ename, dname **from** emp e **right outer join** dept d **on** e.deptno = d.deptno;

Si on veux les deux cas de figure :

Select ename, dname **from** emp e **full outer join** dept d **on** e.deptno = d.deptno;

Rappel : jointures

Oracle permet de spécifier l'attribut utilisé pour réaliser la jointure via le suffixe (+).

Par exemple :

```
SELECT * FROM EMP E, BONUS B
WHERE E.ENAME = B.ENAME(+) AND E.COMM = B.COMM;
```

va récupérer tous les employés qui ont reçu une comm dont le montant existe dans la table des bonus même s'ils leur nom n'a pas été enregistrés dans la table des bonus.

Peux aussi s'écrire :

```
SELECT * FROM EMP E LEFT OUTER JOIN BONUS B
ON E.ENAME = B.ENAME
WHERE E.COMM = B.COMM
```

Optimiseur : Indexes et jointures externes

Cas des jointures externes, la table non-externe (celle qui n'est pas suivie du (+)) est la table directrice.

Pour l'utilisation des index, plusieurs cas sont possibles :

Les clauses WHERE ne portant pas sur la jointure mais sur la table non-externe sont appliquées AVANT la jointure et les éventuels index existants peuvent être utilisés.

Les clauses WHERE ne portant pas sur la jointure mais sur la table externe et qui sont suivies par un (+) sont appliquées AVANT la jointure et les éventuels index existants peuvent être utilisés.

Les clauses WHERE ne portant pas sur la jointure mais sur la table externe et sans le (+) sont appliquées APRES la jointure et les éventuels index existants ne peuvent pas être utilisés (Oracle 8).

Optimiseur : Indexes et jointures externes

Par exemple :

SELECT DNAME, ENAME FROM DEPT, EMP WHERE DEPT.DEPTNO = EMP.DEPTNO(+) AND DNAME = 'ACCOUNTING'

CREATE INDEX IDEPT1 ON DEPT(DNAME)

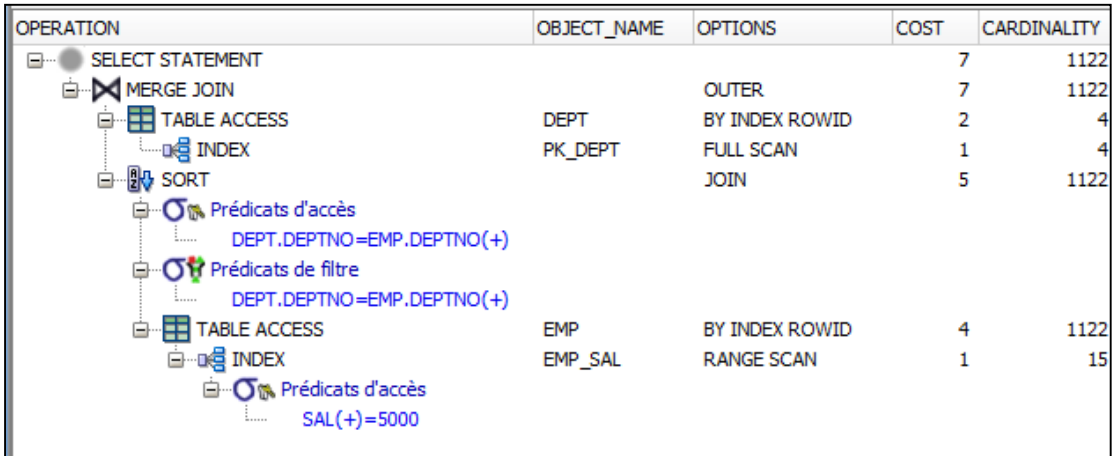
La table directrice est DEPT et l'index sur DNAME sert à y accéder.

Dans le cas :

SELECT DNAME, ENAME FROM DEPT, EMP WHERE DEPT.DEPTNO = EMP.DEPTNO(+) AND SAL(+) = 5000

CREATE INDEX EMP_SAL ON EMP (SAL)

La table directrice est toujours DEPT et un FTS est utilisé pour y accéder. La condition sur SAL est appliquée AVANT (grace au (+)) et l'index EMP_SAL est utilisé pour accéder à EMP.



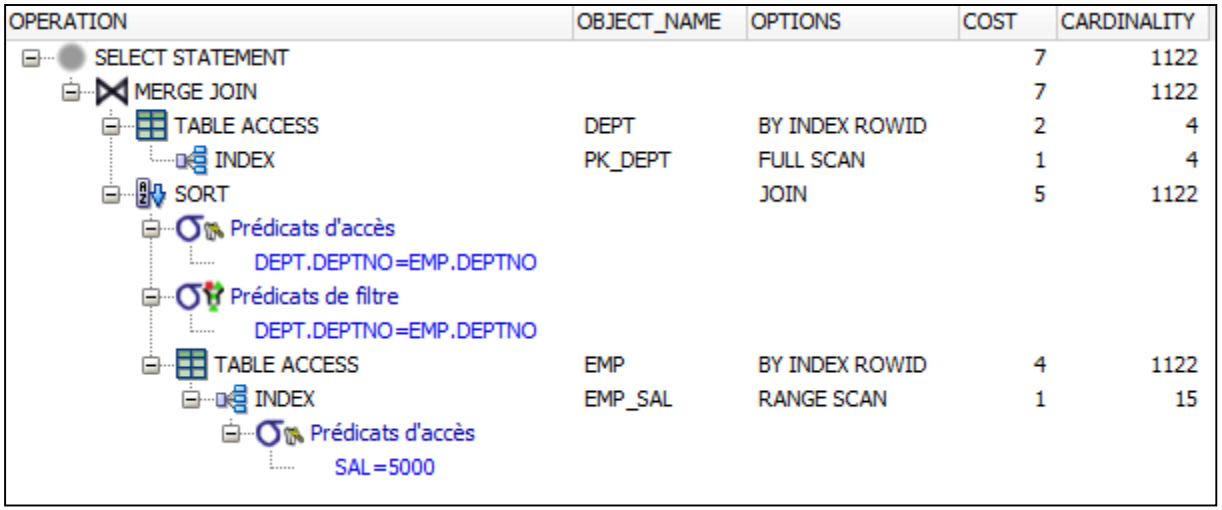
Optimiseur : Indexes et jointures externes

Egalement :

SELECT DNAME, ENAME FROM DEPT, EMP WHERE DEPT.DEPTNO = EMP.DEPTNO (+) AND SAL = 5000

CREATE INDEX EMP_SAL ON EMP (SAL)

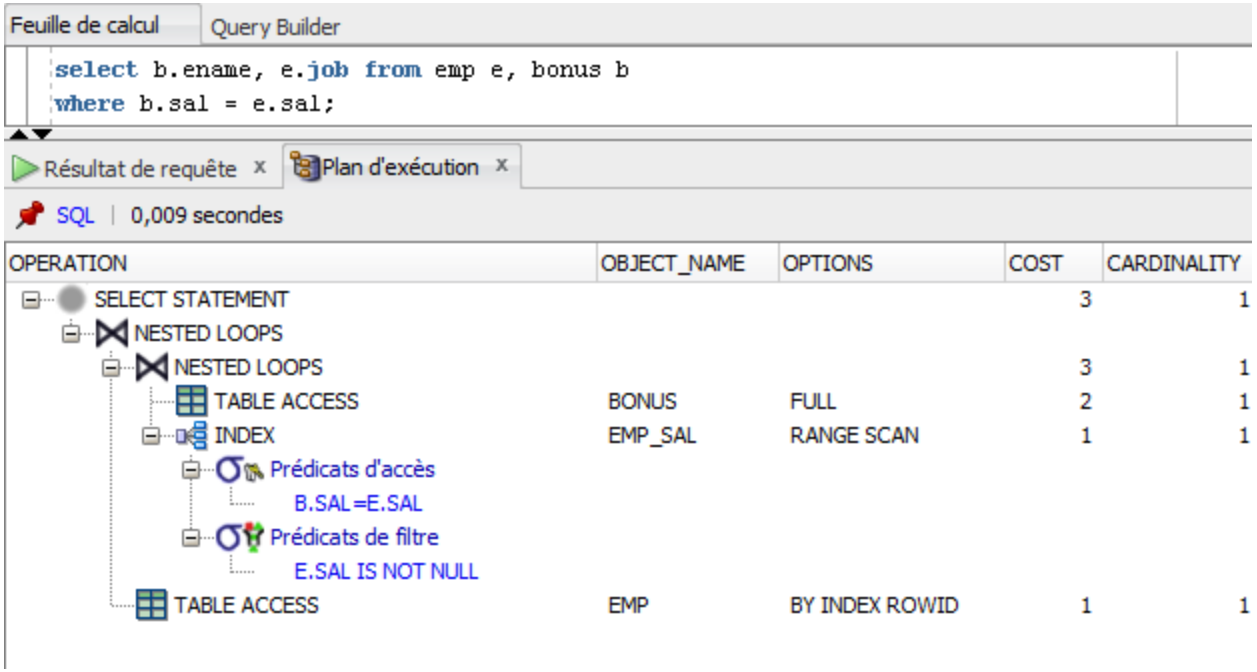
Idem que pour précédemment, et l'index est utilisé.



Méthodes de jointures de l'optimiser : Nested Loop

Méthode des boucles imbriquées :

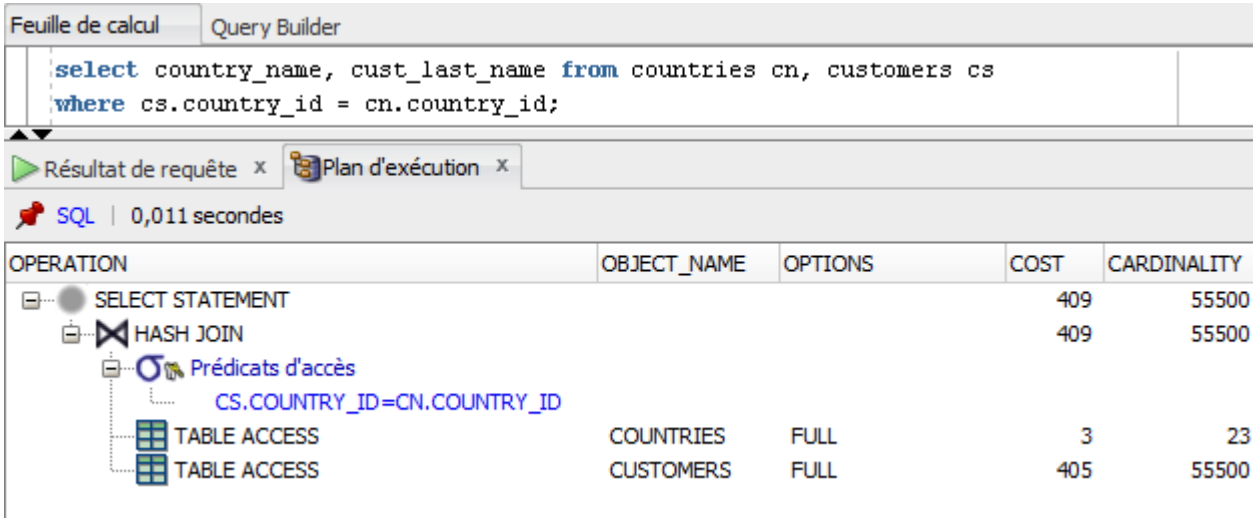
- Efficace pour des ensembles réduits de données
- Requière que la table interne soit dirigée par (dépende de) la table externe pour éviter que les même enregistrements soient récupérés par plusieurs itérations.



Méthodes de jointures de l'optimiser : Hash Join

Méthode de la jointure par table de hash :

- Efficace pour des ensembles importants de données
- Efficace si une partie importante de la plus petite de source de données fait parti de la jointure.
- L'optimiser construit une table de clés de hash en mémoire à partir de la table de données source la plus petite puis l'utilise pour parcourir la plus grande source de données.



Méthodes de jointures de l'optimiser : Sort Merge Join

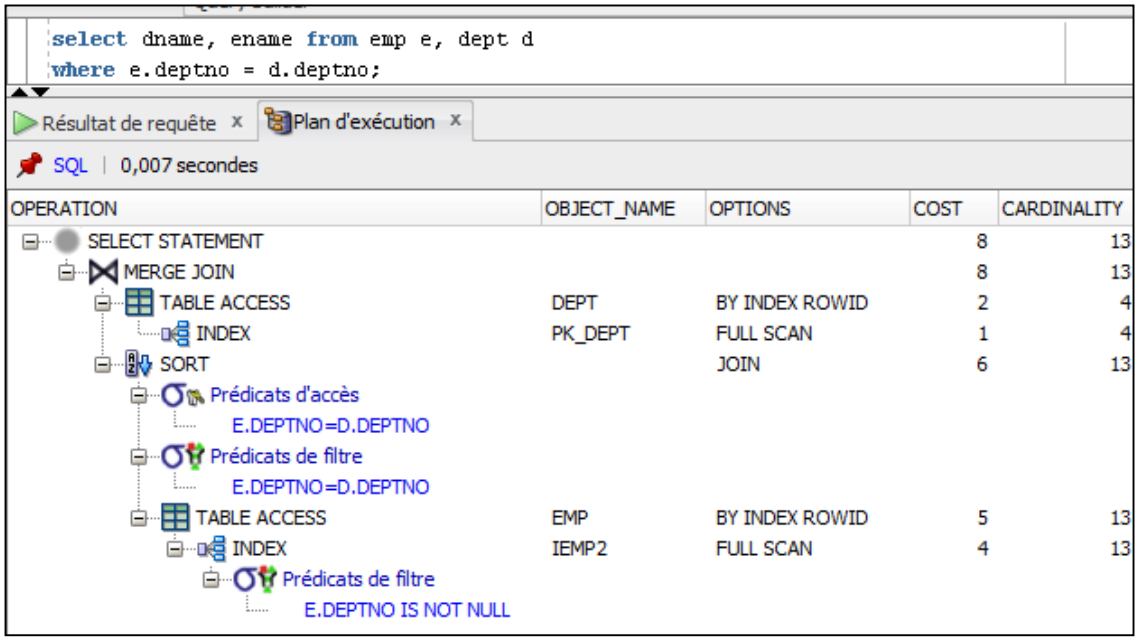
La méthode de la jointure par tri fusion (sort merge) est souvent moins efficace que la méthode par Hash sauf si :

- Les lignes sont déjà triées ET pas d'opération de tris

Elle consiste en deux étapes :

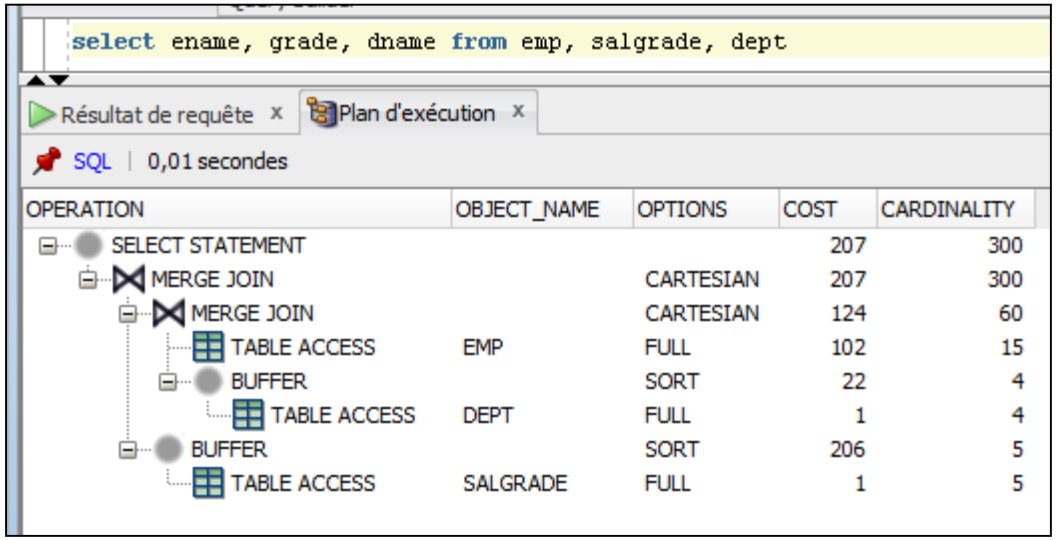
- Tris par rapport à la clé
- Fusion des ensembles triés

Et sera également plus efficace que les boucles imbriquées sur des ensembles important de données et lorsque les conditions de jointures sont des inégalitées (>, <, …).



Méthode de jointure de l'optimiser : produit cartésien

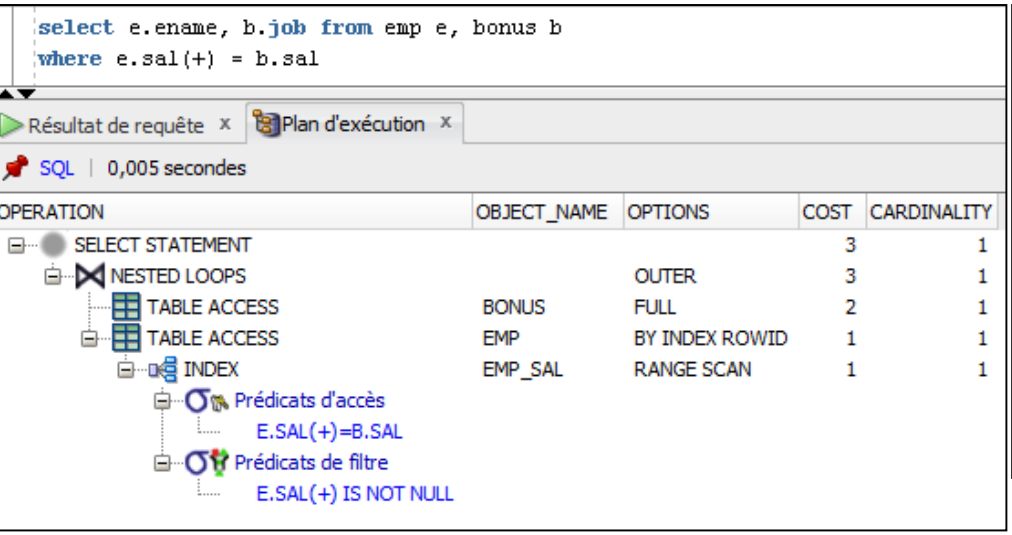
Utilisé lorsqu'il n'y a pas de condition de jointure.



OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			207	300
MERGE JOIN		CARTESIAN	207	300
MERGE JOIN		CARTESIAN	124	60
TABLE ACCESS	EMP	FULL	102	15
BUFFER		SORT	22	4
TABLE ACCESS	DEPT	FULL	1	4
BUFFER		SORT	206	5
TABLE ACCESS	SALGRADE	FULL	1	5

Méthodes de jointures de l'optimiser : jointures externes

Dans le cas des jointures externes, les tables directrices sont explicites et l'optimiser n'a plus la possibilité de choisir l'ensemble de données le plus efficace pour décider des tables directrice des jointures.

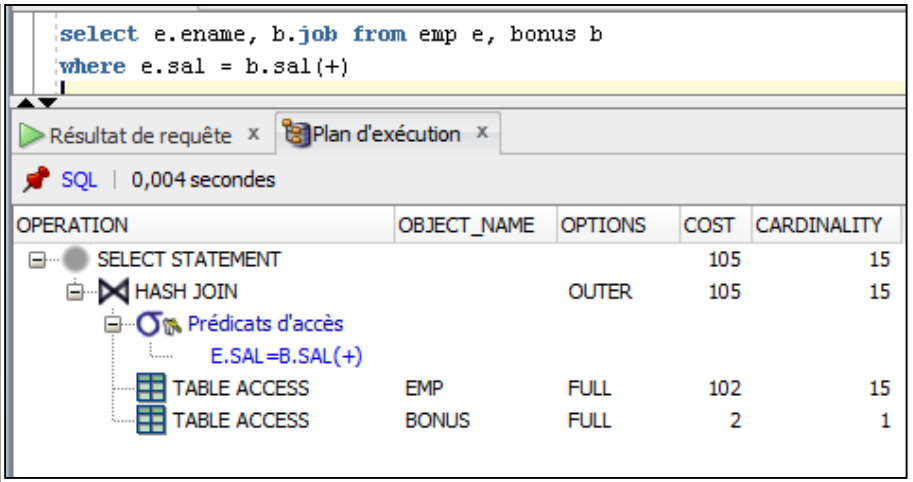


```
select e.ename, b.job from emp e, bonus b
where e.sal(+) = b.sal
```

Résultat de requête | Plan d'exécution

SQL | 0,005 secondes

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			3	1
NESTED LOOPS		OUTER	3	1
TABLE ACCESS	BONUS	FULL	2	1
TABLE ACCESS	EMP	BY INDEX ROWID	1	1
INDEX	EMP_SAL	RANGE SCAN	1	1
Prédicats d'accès				
E.SAL(+)=B.SAL				
Prédicats de filtre				
E.SAL(+) IS NOT NULL				



```
select e.ename, b.job from emp e, bonus b
where e.sal = b.sal(+)
```

Résultat de requête | Plan d'exécution

SQL | 0,004 secondes

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			105	15
HASH JOIN		OUTER	105	15
Prédicats d'accès				
E.SAL=B.SAL(+)				
TABLE ACCESS	EMP	FULL	102	15
TABLE ACCESS	BONUS	FULL	2	1

Éléments critiques de la performance de l'application

- « *Si les DBA peuvent identifier les points et problèmes qui sont les causes de ralentissement, les développeurs peuvent les résoudre.* »

- 80 % des gains de performances sont dues à une optimisation du code de l'application
  - Par son modèle,
  - Par le codage des requêtes SQL.

- Mais, la performance est **dynamique** et la performance d'une requête dans un contexte (base de dev) ne sera pas représentatif d'une performance dans un autre (prod) ni dans la durée. Le rôle du DBA est d'apporter au développeur la vue sur le comportement dynamique de son application.

Outillage de la mesure de la performance

- Mise en place de statistiques dynamiques sur le fonctionnement de la base :
  - Observation du fonctionnement des process internes de la base,
  - Observation du fonctionnement des applications (SQL) et du modèle de données (accès aux tables/indexes/enregistrements).
- Demande un temps d'exécution minimal pour être pertinents,
- Processus récurent « d'administration » de la base de données.

Analyse des statistiques : les causes possible de perte de performance

Les attentes (wait) sont la cause des pertes (ou baisses de performances).

Elles sont classifiées selon leur cause principale :

- **Actions d'administration** : commandes réalisées par un utilisateur privilégié qui induit une attente des autres utilisateurs (la reconstruction d'un indexe par exemple).
- **Application** : tous les "défauts" induits par le modèle qui induisent des blocages (mise en attente) du fait de l'accès concurent sur une donnée pour une cause explicite ou implicite (modification)
- **Architecture cluster** : cach global, synchronisation des noeuds, …
- **Actions liées à un commit** : évennements d'attentes liées au redolog
- **Accès concurrent** par beaucoup de sessions vers une même ressource (parsing SQL, verrou sur des buffers, …)
- **Configuration** : construction de ressources (log, logfile, buffer, …) de taille insuffisante
- **Idle** : sessions inactives
- **Réseau** : attente de données envoyée par le réseau
- **Scheduler** : attente liée à une priorisation du manager de ressources
- **System I/O** : attente sur les I/O système pour les processus en arrière plan (autre que ceux liés à la gestion des données utilisateur).
- **User I/O** : attente de lecture de bloc (processus SMON, or MMON).

Tuning MySQL

- Approche différente puisque 2 moteurs coexistent : MyISAM et InnoDB
  - MyISAM, moteur par défaut, stocke les tables sous la forme de 3 fichiers : 1 pour le format de la table, 1 pour les données, 1 pour les indexes. Indexation textuelle plus performante, insertion, lecture et modification plus efficiente dans la plus part des cas. Mais : pas de dépendance fonctionnelle entre table (contraintes sur les FK), lock sur l'ensemble de la table en cas d'insertion.
  - InnoDB, stocke les tables dans un espace de stockage comparable à un TableSpace Oracle. Permet le lock au niveau des enregistrements et les contraintes sur les clés etrangères.
- Table de choix :

Critère	Moteur
Est-ce que la table va être beaucoup plus modifiée (RWUD) que sélectionnée (Transaction Safe)	InnoDB
Recherche en texte plein	MyISAM
Modélisation et contrôle relationnel	InnoDB
Espace de stockage ou RAM limitée	MyISAM
Dans le doute	MyISAM

Sécurisation SGBD

Types de changements

Autour de la base

Modification du matériel,

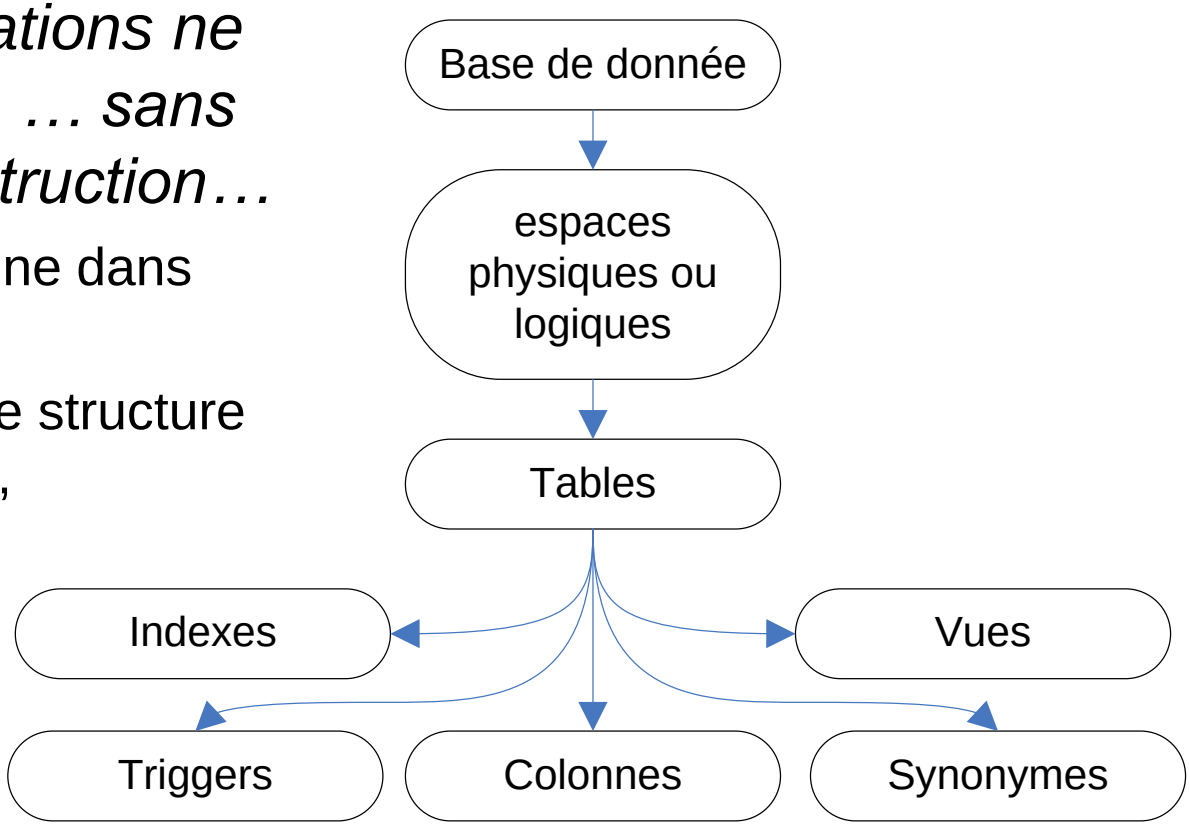
Modification/cohérence de l'analyse (MCD, MLD, MPD),

Modification de l'application utilisatrice…

Modification d'une base de donnée

Toutes les modifications ne sont pas possibles … sans destruction/reconstruction…

- Ajout de colonne dans une table,
- Modification de structure de tablespace,
- ….



Modification d'une base de données
Ce que ALTER ne peut généralement pas…

Pour la base :

- Changer son nom,
- Déplacer un objet d'une base à un autre,

Pour sa structure physique :

- Changer le nombre de partitions ou de fichiers de données d'un tablespace,

Pour les tables :

- Déplacer d'un tablespace à un autre,
- Réarranger l'ordre des colonnes,
- Effacer une colonne, en changer le type ou la taille,
- Ajouter une colonne non nulle ou entre deux existantes,

Pour les vues :

- Ajouter ou effacer une colonne, changer la commande SELECT,

Pour les indexes :

- L'unicité, le clustering, l'ordre

Modifier le contenu d'un trigger

Changer une clé de Hash.

Modification d'une base

Exemple de l'ajout d'une colonne entre deux existantes

1. Retrouver la définition actuelle de la table,
2. Retrouver la définition de toute vue qui la référence,
3. Retrouver la définition de tout index qui la référence,
4. Retrouver la définition de tout trigger sur la table,
5. Capturer toutes les contraintes appliquées sur la table et mesurer l'impact de l'effacement de la table,
6. Retrouver tous les grants sur la table,
7. Retrouver tous les programmes qui la référence,
8. Sauvegarder les données de la base,
9. Effacer la table, ce qui inclus également tous les indexes, trigger et vues liés,
10. Recréer la table avec la nouvelle colonne,
11. Recharger les données,
12. Recréer les contraintes (clés étrangères) effacées,
13. Recréer les triggers, indexes et vues,
14. Recréer les grants,
15. Examiner en détail toute requête applicative qui référence la table.

Problèmes sur une base ?

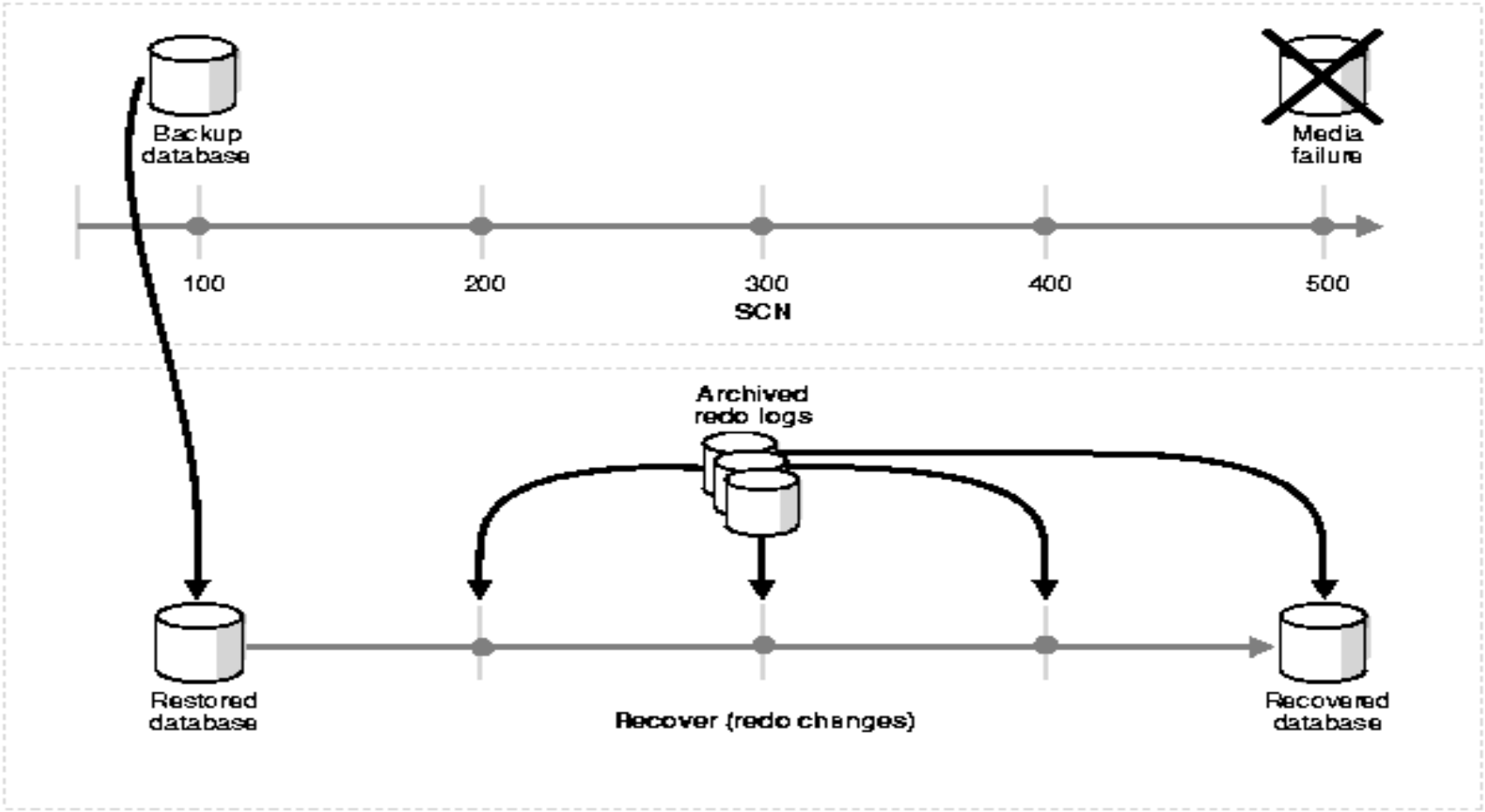
- Disfonctionnement d'un média physique
  - Un disque casse,
  - Une CPU casse, …
- Erreur de manipulation
  - Un utilisateur effectue une opération malheureuse,
- Corruption de l'état de l'instance
  - Suite à un arrêt brutal
- Erreur de paramétrage
  - Toutes les extensions (extens) sont utilisées
- Disfonctionnement d'un processus Oracle
  - RMON est là pour surveiller
- Problèmes réseaux…

Problèmes sur une base !

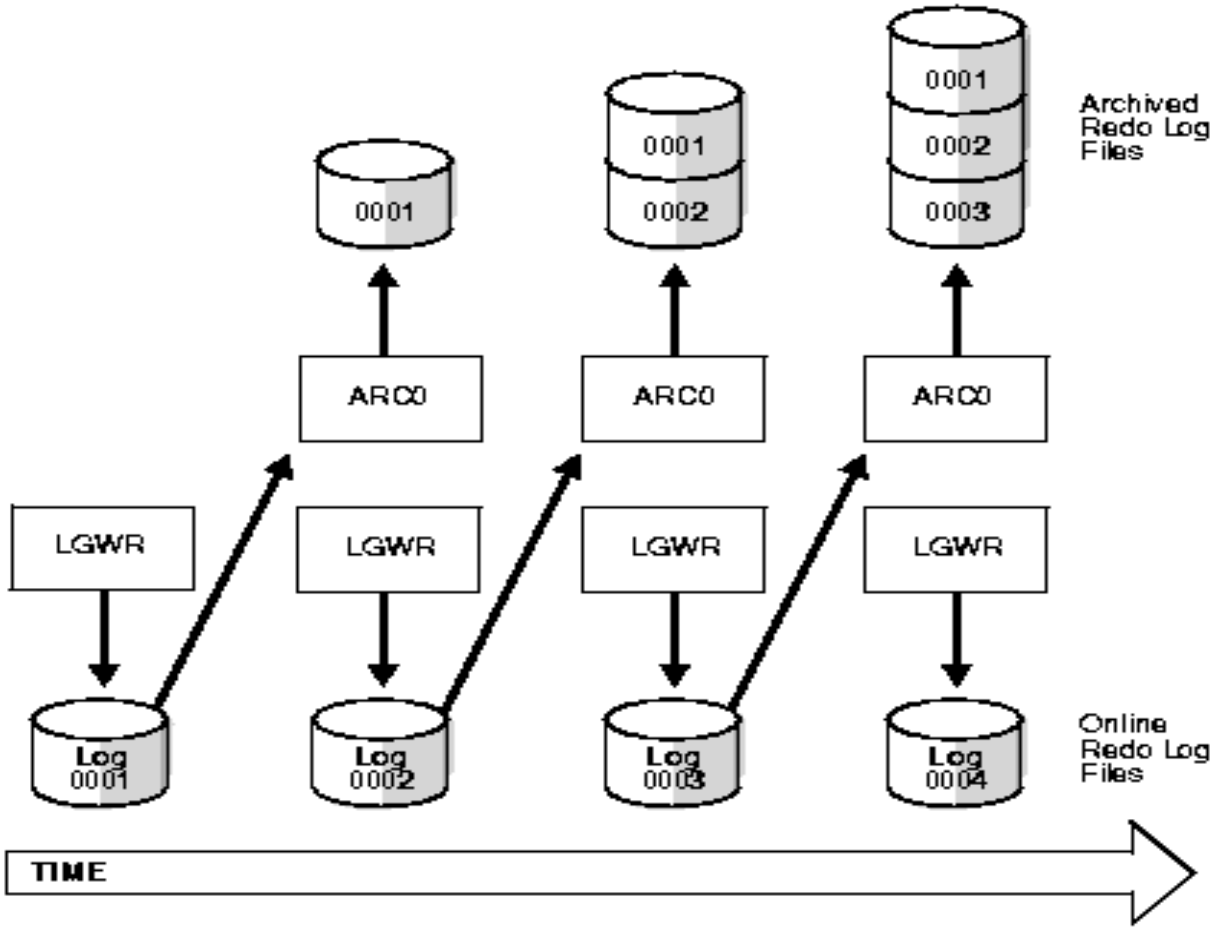
Chaque type de problème amène sa solution **préventive** :

- Matériel : Sauvegarde + Dataguard + Cluster
- Manipulation : Sauvegarde + instances de développement
- Corruption de l'état de l'instance : Sauvegarde + gestion de l'instance par le DBA
- Erreur de paramétrage : actions ponctuelles
- Disfonctionnement Processus : Prise en charge par PMON
- Problèmes réseau : Prise en charge par PMON et RMON

Sauvegarde / Restauration



Sauvegarde / Restauration



Sauvegarde / Restauration

Méthodologie générale de la sauvegarde

- À chaud (on-line) :
  - Minimise la durée de la sauvegarde
  - Peut provoquer des lock selon les DBMS
  - Nécessiter la mise en œuvre de méthodologie spécifiques à chaque DBMS
- À froid (off-line) :
  - Minimise la durée de la restauration
  - Nécessite un arrêt de la base
  - Est effectuée par copie de l'ensemble des éléments physiques de la base
  - Est possible avec tous les DBMS

Sauvegarde / Restauration

Stratégie : souvent une combinaison de sauvegarde à chaud et à froid

- Sauvegarde à froid une fois tous les N jours
- Sauvegarde à chaud plus régulièrement

On joue alors sur le mode et la durée de rétention :

- Redondance : combien de copie à un temps t donné,
- Durée : pendant combien de temps une sauvegarde doit-elle être conservée

Sauvegarde / Restauration

Nombre de bandes nécessaires ?

- Méthode GFS : Grand-père, Père, Fils :
  - Fils = Une sauvegarde différentielle ou FULL journalière
  - Père = Une sauvegarde FULL par semaine
  - Grand-père = Une sauvegarde FULL (père) par mois
- Rotation des bandes (sur 1 ans) :
  - Un jeu de bande est utilisé pour le fils (4, 5 ou 6),
  - Un jeu de bande est utilisé pour le père (4),
  - Un jeu de bandes est utilisée pour le grand-père (12)

Sauvegarde / **Restauration**

Selon le problème :

Revenir au dernier état disponible

- Restauration de l'ensemble des sauvegardes disponibles (FULL + Archives Log)

Revenir au dernier état stable

- Restauration du dernier FULL disponible
- Analyse des Archives Log pour ne pas reproduire l'erreur.

Sauvegarde / Restauration

Des solutions intégrées existent :

- Intégrée (RMAN) pour Oracle,
- Intégrée pour SQL Server,
- Plus à la main pour MySQL,

L'intégration des solutions est souvent ce qui *différentiait* les solutions propriétaires des solutions libres… ce qui tend à ne plus être vrai.

Sécurisation par clustering

L’idée est d’avoir :

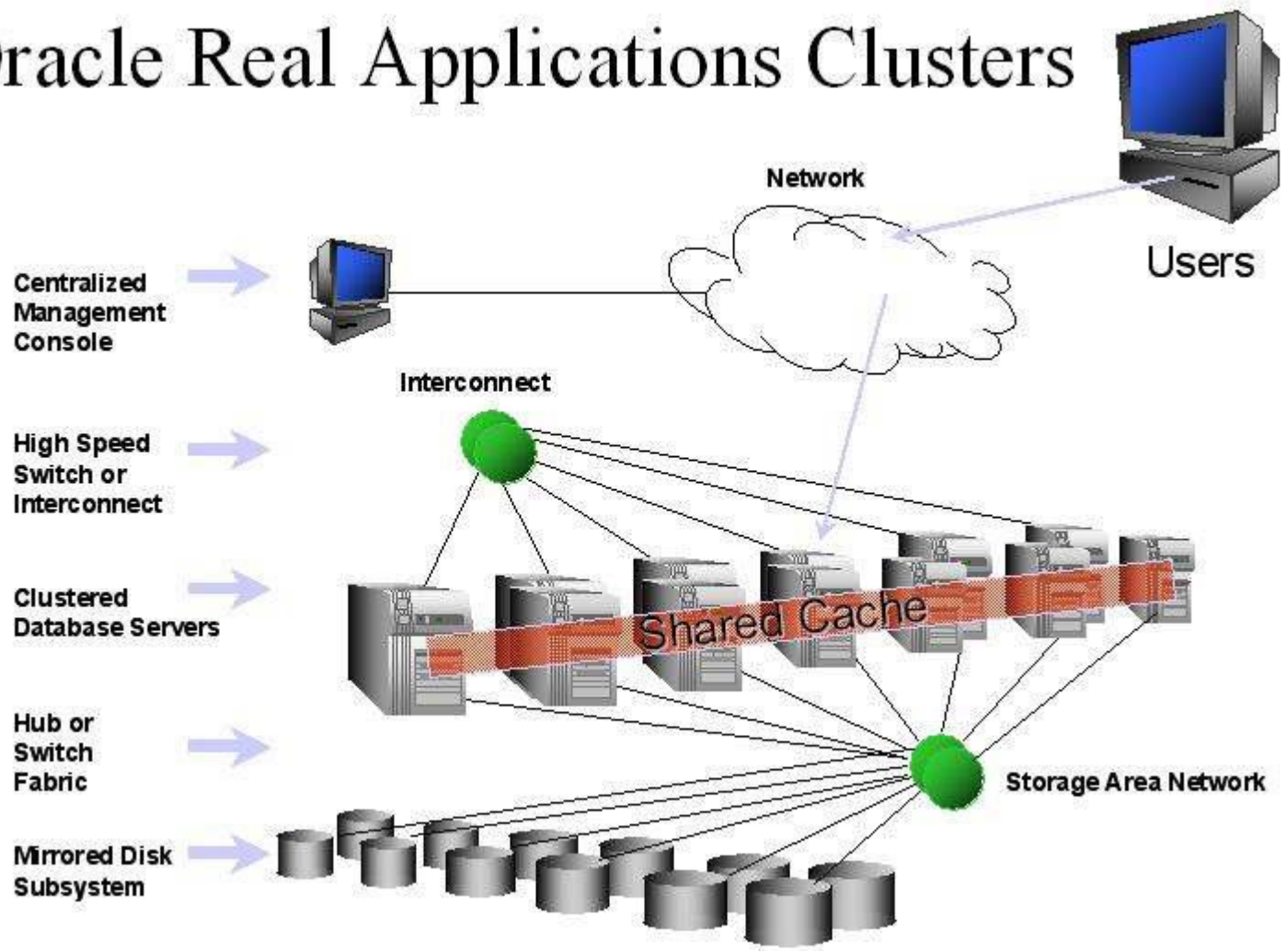
- Une sécurisation de la puissance de calcul
- Une sécurisation des données
- Les deux

Les solutions dépendent des DBMS mais se regroupent sous le terme de Cluster

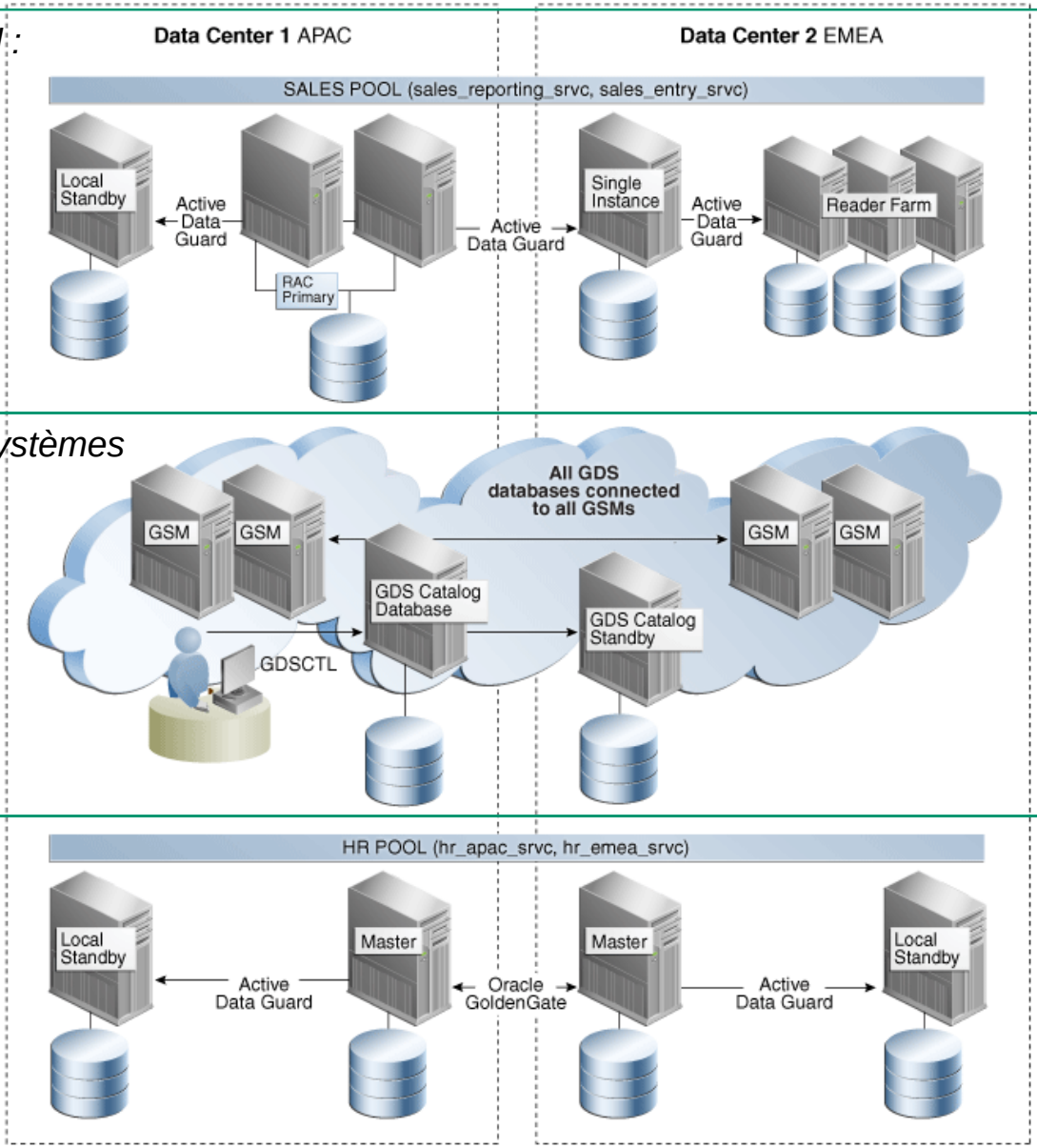
- Un élément dirige les requêtes vers un ensemble de ressources les plus disponibles

Cluster actif/actif

Les données ne sont stockées qu'à un seul endroit. L'accès au disque (partagé) est réalisé via des FS spécialisés (ASM, VxSF, …) permettant l'accès à un même fichier à n systèmes actifs. Les modifications physiques sont réalisées par un seul nœud et communiqué aux autres nœuds.



Système commercial : haute disponibilité haute performance



Production RW Locale
Production RO Distante
Standby Locale
Standby Distante

Administration des systèmes

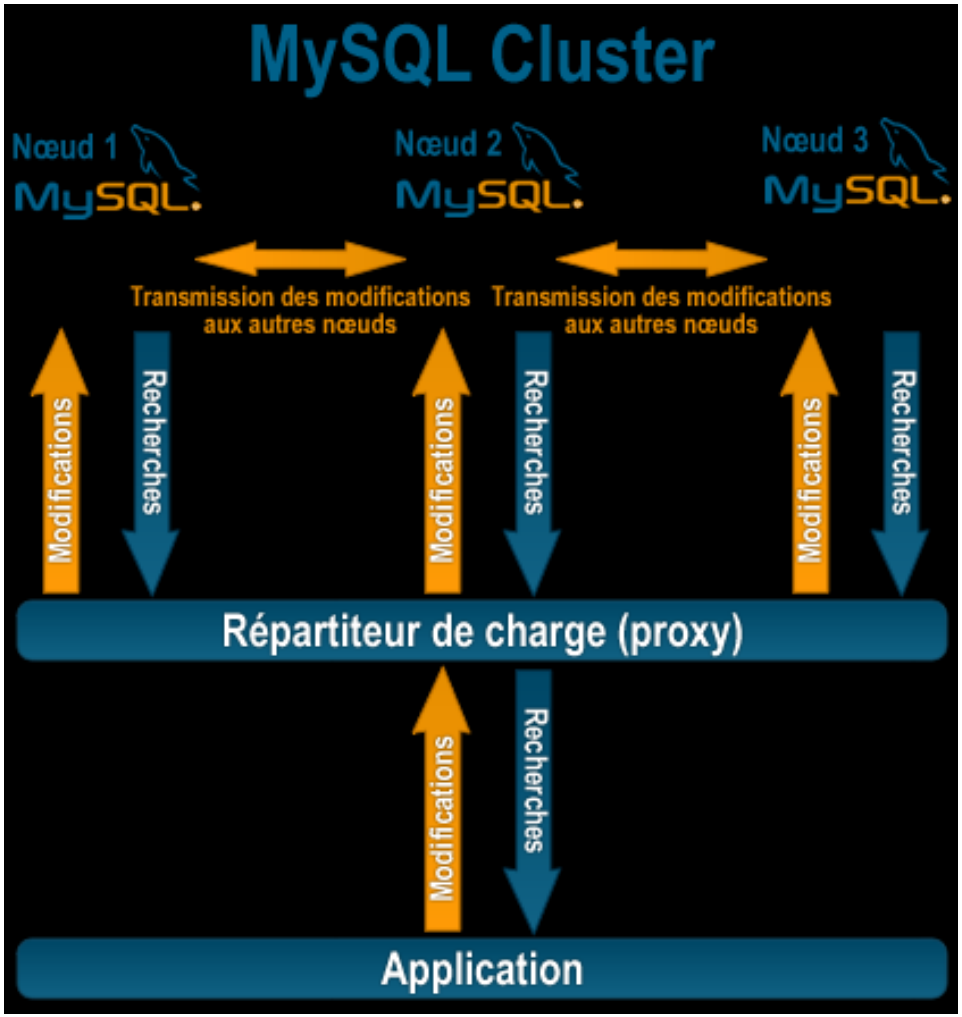
Global Service Management
Global Data Service

Système RH : haute disponibilité

Production RW Répartie
Standby sur les deux DC

Cluster actif / actif

Chaque nœud du cluster stocke l'intégralité des données.
Les modifications sont transmises à tous les autres nœuds pour prise en compte dans leurs propres fichiers.



Cluster actif/actif

Fonctionne via une VIP (Virtual IP) gérée par le gestionnaire de clusteurisation (RAC / répartiteur de charge) pour adresser les nœuds via une seule adresse.

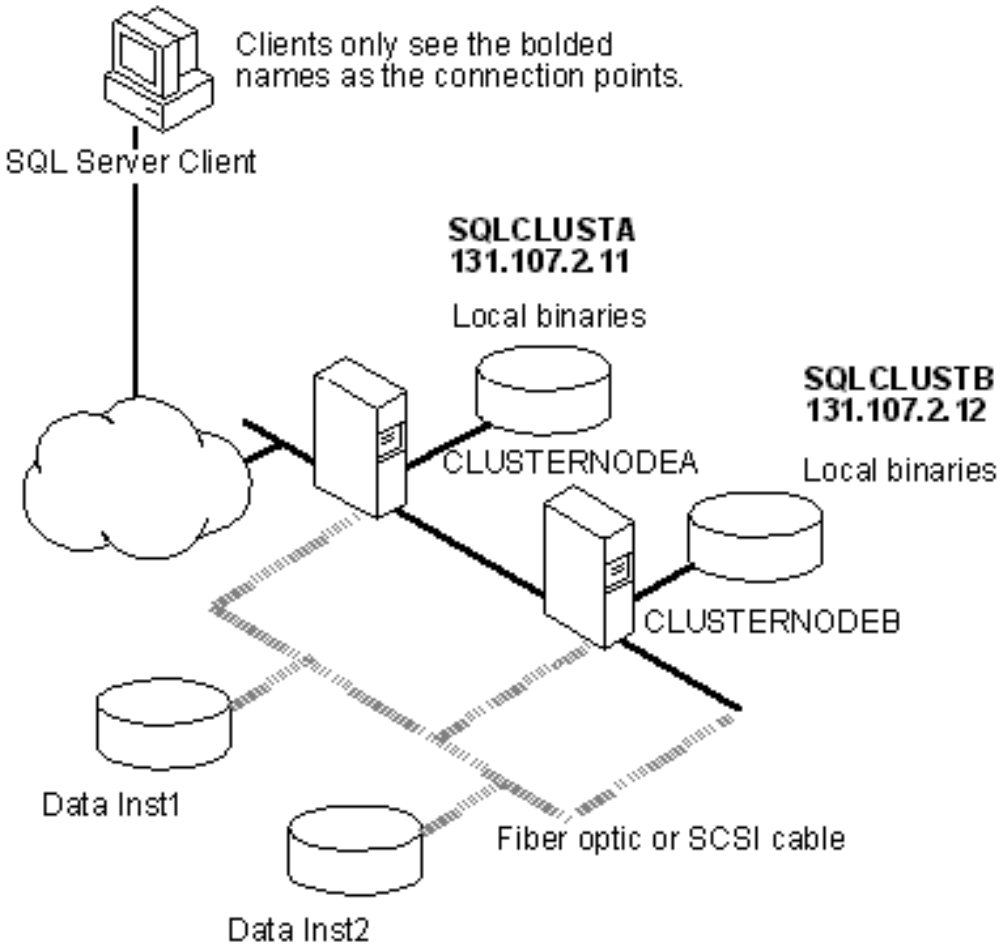
Travaille en mode « mémoire globale partagée »

- Induit un temps de communication important dans le cas de modification des données,
- Pour RAC : Induit une utilisation des compteurs « par plage » qui ne garanti plus la continuité « historique » des numéros fournis (sauf à réduire la plage à 1) : chaque nœuds a « une plage » qu'il peut utiliser et se synchronise avec les autres lorsqu'il a tout utilisé.

Permet la « spécialisation » des nœuds en fonction de leur architecture sous jacente (châssis optimisé pour le calcul, pour les transations, …).

Cluster actif/passif
Failover Cluster SQL Server

Un seul nœud est actif à un moment donné. En cas de défaillance du nœud, un autre nœud devient actif sur les mêmes données.
Demande une couche OS de clusterisation gérant le nœud qui « voit » et peut « modifier » les données.
Existe aussi sous Unix avec des outils comme (par exemple) Veritas, PowerHA, … qui gèrent au niveau OS les « bascules » des stockages et des logiciels.



Architecture de réplication des données

Gère la distribution en temps réel / différé des modifications sur les bases.

Se base sur le principe de « nœud maître » (dans lequel sera effectué les modification) et de « nœuds esclaves » qui sont informés des modifications

Permet la mise en place de nœuds en lecture seule mais aussi la bascule de responsabilité maître / esclave entre les nœuds

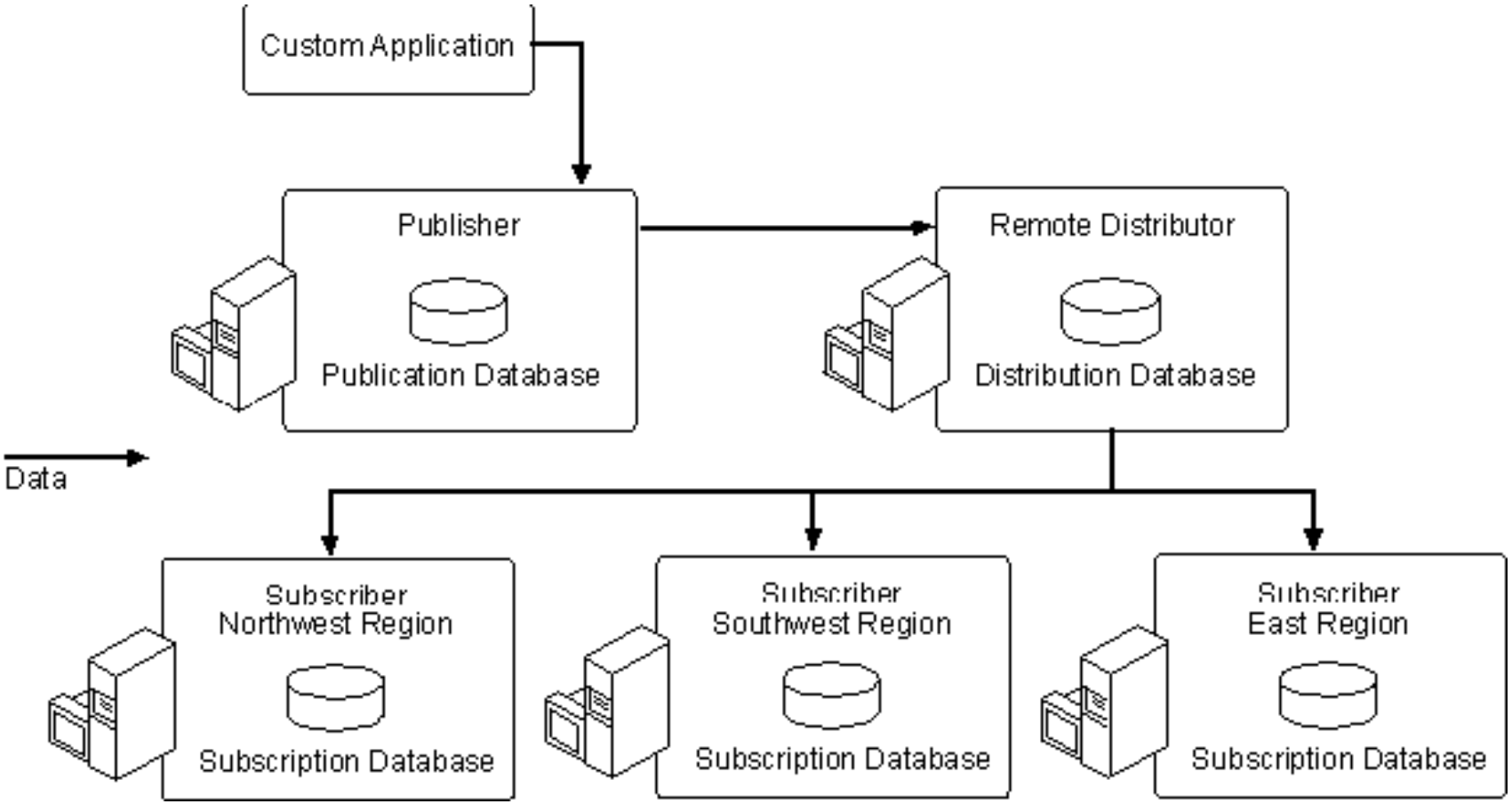
- Pour gérer des incidents (perte de base)
  - Attention, le retour au nominal est coûteux car il demande l'arrêt de la base maître le temps de reconstruire l'esclave !
- Pour gérer la proximité (service devant être rendu « au plus prêt » des utilisateurs/administrateurs)

Dataguard Oracle

Réplication des données d'un serveur principal vers un (ou plusieurs) serveur secondaires via un processus LGWR (écriture / transmission des log). Chaque serveur ayant un stockage en propre et pouvant être situé dans des datacentres différents.

- **Protection garantie** : la transaction n'est validée que lorsqu'une base secondaire à inscrit l'opération dans son log (PROTECTION),
- **Protection instantanée** : la transaction est validée sans attente et transmise immédiatement (AVAILABILITY),
- **Protection différée** : la transaction est transmise plus tard (PERFORMANCE).

Réplication SQL Server (1/2)



Réplication SQL Serveur (2/2)

Mode snapshot :

- Une image des données est distribuée

Mode transaction :

- Une image initiale est distribuée,
- Puis uniquement les modifications.

Mode merge :

- Une image est préparée,
- Le transfert et l'application peut être différée.