EPITA

Bases de données Apprentissage AppIng2 1er semestre

Introduction aux bases de données

Qu’est une base de données ?

Approche par fichiers

Avant les systèmes de gestion des bases de données

- Les premiers embryons de bases de données remontent aux années 1950 pour faciliter les traitements bancaires et d'assurances
- Pendant près de 20 ans, les données étaient stockées dans des fichiers « plats »
- Le principal langage de traitement de fichiers était le COBOL

Le fichier

- C'est un objet stocké sur un support inaltérable
- Il est constitué d'octets mis bout à bout
- Il comporte une marque de fin
- La données stockée dans le fichier est à l'appréciation du développeur
- La structure du fichier peut être fortement formatée ou très libre
- La cohérence des données présentes dans le fichier ne peut être garantie que par le programme accédant au fichier

Les fichiers structurés

- Un fichier structuré est un fichier qui contient des informations écrites selon un schéma imposé
- Ce schéma imposé est généralement de type champs à largeur fixe
- Chaque groupe d'octets correspond soit à un champ, soit à un enregistrement
- La position d'un enregistrement peut donc être connue à l'avance grâce à son offset

Le fichier séquentiel indexé

- Ce type de fichier a été créé pour permettre un accès rapide à l'information présente
- L'indexation permet de se placer directement à une position référencée dans le fichier
- Cette notion est très présente dans un langage comme le Cobol
- Il implique une structure de données connue à l'avance
- Un ou plusieurs champs servent d'index. Cet index est maintenu dans un fichier à part, permettant le pointage vers les enregistrements présents dans le fichier à accéder
- L'indexation est faite sur une clé primaire, c'est-à-dire comportant une et une seule fois une valeur de clé

Forces et faiblesses

- Points forts :
  - Le fichier est la structure la plus « légère » en manipulation par programme
  - Tout système d'exploitation sait gérer des fichiers
- Points faibles :
  - Une fois la structure établie, il est très difficile de la modifier
  - Une modification de la structure peut entraîner très rapidement des modifications en profondeur du programme écrit pour l'accès au fichier
  - La cohérence des données doit être maintenue par le programme
  - La concurrence d'accès à la donnée doit être écrite par le développeur
  - La fiabilité du fichier repose sur la fiabilité du système d'exploitation
  - Des mécanismes d'auto-correction doivent être mis en place par le développeur pour palier à d'éventuels dysfonctionnements du système sous-jacent

Approche par modèle relationnel

La base de données

Naissance du concept de SGBD

- Apparition du langage SQL
- Séparation nette entre le socle physique supportant les données, et les opérations de traitement
- Conceptualisation des données
- Invention de la normalisation
- Forme normale de Boyce-Codd
- Concept de Système de Gestion de Bases de Données

La base de données

- C'est l'endroit où on stocke la donnée
- Le stockage de la donnée implique la persistance de l'information
- La base de données est inerte
- La base de données nécessite un outil pour évoluer dans le temps

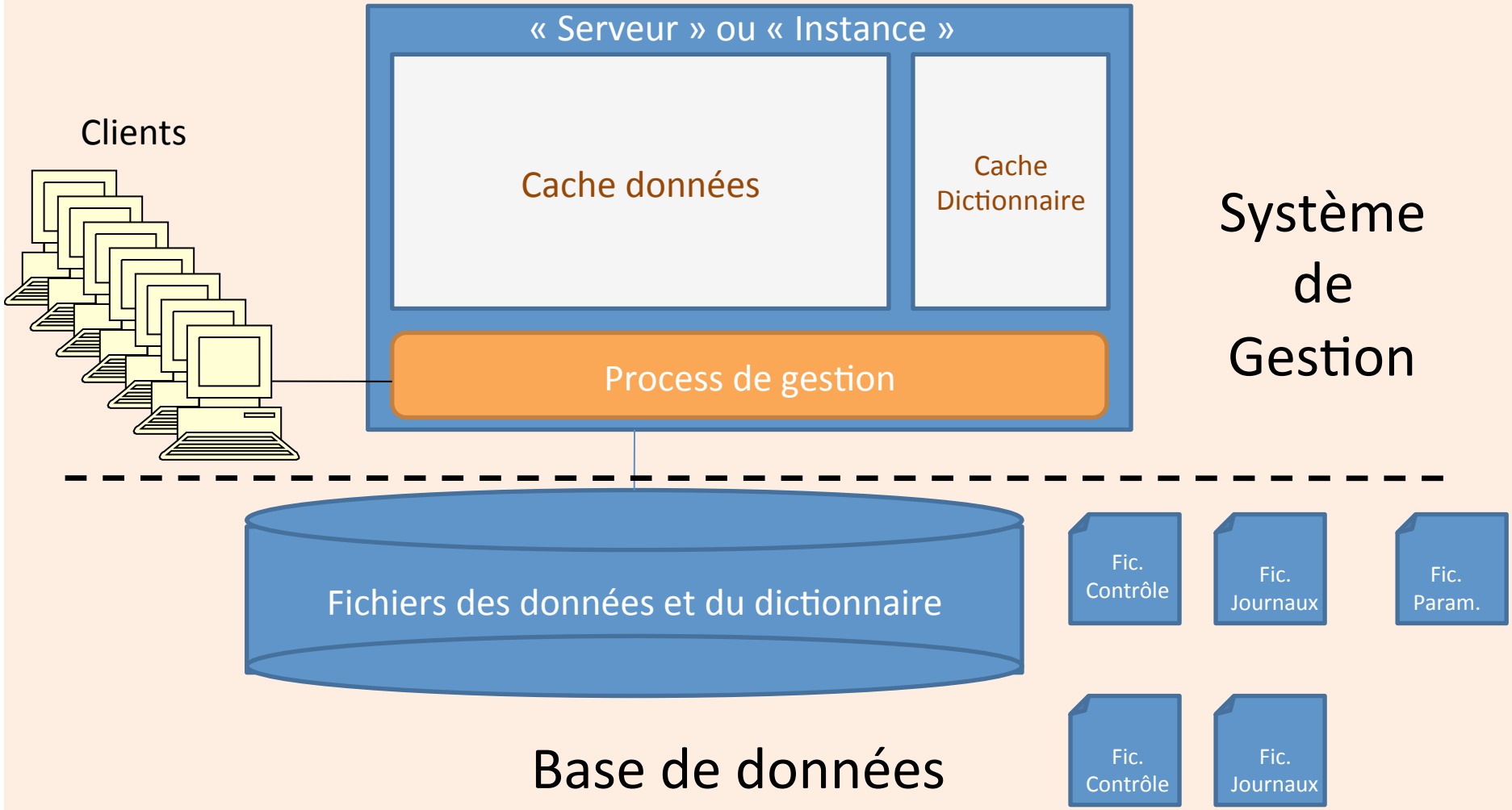
Gestion des bases de données

- Une base de données est gérée par un système
- Il s'agit du Système de Gestion de Bases de Données, ou SGBD
- Le système de gestion pourvoit à l'interface entre la donnée et l'utilisateur
- Le système de gestion permet la mise à jour de la donnée

Propriétés ACID

- Atomicité : Une transaction est atomique, si elle peut être intégralement validée ou intégralement annulée
- Cohérence : Une transaction ne peut laisser une base de données en état incohérent
- Isolation : Une transaction ne peut voir aucune autre transaction en cours d'exécution
- Durabilité : Une fois la transaction validée avec succès, les résultats de celle-ci ne disparaîtront pas

Schéma simplifié d'un SGBD



Le modèle relationnel

- Il pose l'existence des objets suivants :
  - Les entités
  - Les relations
  - Les associations
- Les entités définissent le stockage ultérieur des données
- Les relations définissent les dépendances des entités entre elles
- Les associations définissent les dépendances des entités entre elles et ajoutent des attributs à la relation de base

Traduction du modèle relationnel dans le monde des bases de données

- Les entités se traduisent par des tables
- Les relations définissent les clés primaires et les clés étrangères entre les tables
- Les relations peuvent aussi définir des tables intermédiaires entre les tables mises en relation
- Elles peuvent définir de nouvelles informations internes dans les tables, nécessaires à leur existence

La table

- Elle est constituée de :
  - Colonnes
  - Parfois de sous-tables
  - Contraintes d'intégrité des données
  - Triggers
  - Clés primaires
  - Index

Les règles de gestion → Contraintes d'intégrité

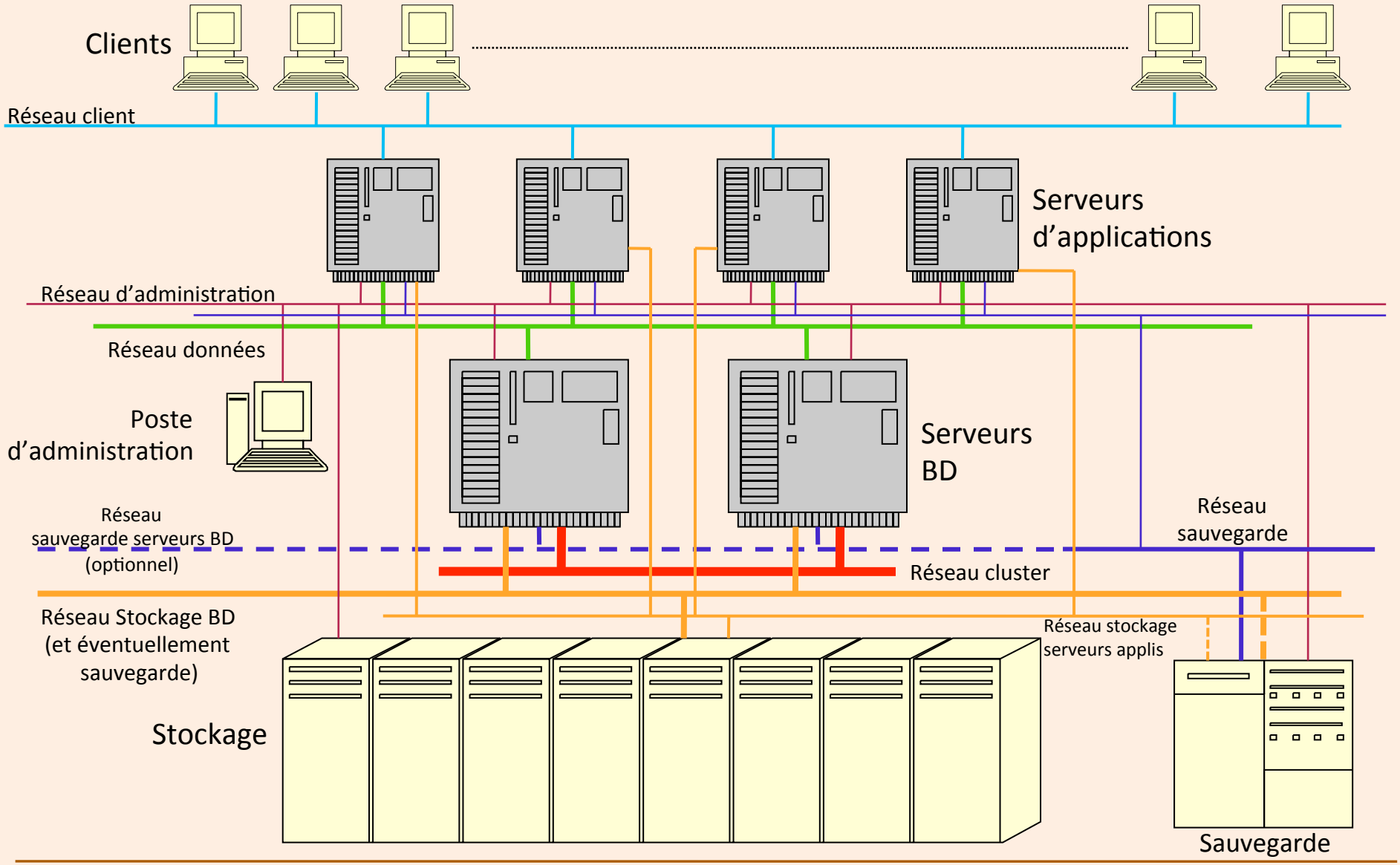
- Nul, non-nul : Une colonne peut accepter ou non la valeur nulle
- La clé primaire : C'est une clé qui définit de manière unique une ligne dans la table. Les éléments définissant la clé primaire sont forcément uniques et ne peuvent comporter de valeur nulle
- Liste de valeurs autorisées : Seules les valeurs de la liste peuvent entrer dans la colonne
- La clé étrangère : Elle s'applique à une ou plusieurs colonnes d'une table, et s'appuie sur la clé primaire d'une autre table
- Le trigger : Il s'agit d'un programme, parfois complexe, qui permet de mettre en place une règle de gestion particulière non gérée par le noyau de base de données pour les données de la table

La relation

- Elle est établie au travers des clés des tables
- Une liaison naturelle entre deux tables consiste à lier une colonne ou un groupe de colonnes entre ces deux tables
- La table-maître contient la clé primaire
- La table liée l'est au travers d'une clé étrangère référençant la clé primaire

Le socle physique de support des bases de données

Le socle physique



Le stockage, élément central de la chaîne

Le stockage

- … est composé de :
  - Baie(s) de disques (enclosures),
  - Disques en RAID (0, 1, 5, 10, 50, sécurité),
  - Câblage interne cuivre (SCSI),
  - Câblage interne fibre (Fibre Channel),
  - Cache mémoire,
  - Operating System dédié,
  - Pour certains : batteries de sauvegarde en cas de coupure de courant.

Le stockage

- Plusieurs types de stockages existent :
  - Stockage interne serveur
  - NAS : Network Appliance Storage
  - SAN : Storage Area Network

Le stockage – forces faiblesses des différents types

Type de stockage	Forces	Faiblesses
Interne	- Peu coûteux - Pas d'encombrement	- Limité en capacité du fait du nombre de slots disques - Architecture non évolutive - Non séparation du stockage et du serveur - Mutualisation impossible
NAS	- Mutualisé - Pas de limitation en terme de disques physiques - Evolution possible - Coût moyen - Possibilité de « mirrorer » le stockage	- Technologie réseau classique (bande passante limitée) - Système de fichiers propriétaire empêchant la création aisée de « raw devices » - Volumineux - Grosse consommation électrique - Nécessite une salle sécurisée
SAN	- Mutualisé - Pas de limitation en terme de disques physiques - Réseau dédié - Possibilité de mutualiser les sauvegardes - Bande passante quasi-illimitée - Possibilité de « mirrorer » le stockage	- Cher !!! - Volumineux - Grosse consommation électrique - Nécessite une salle sécurisée

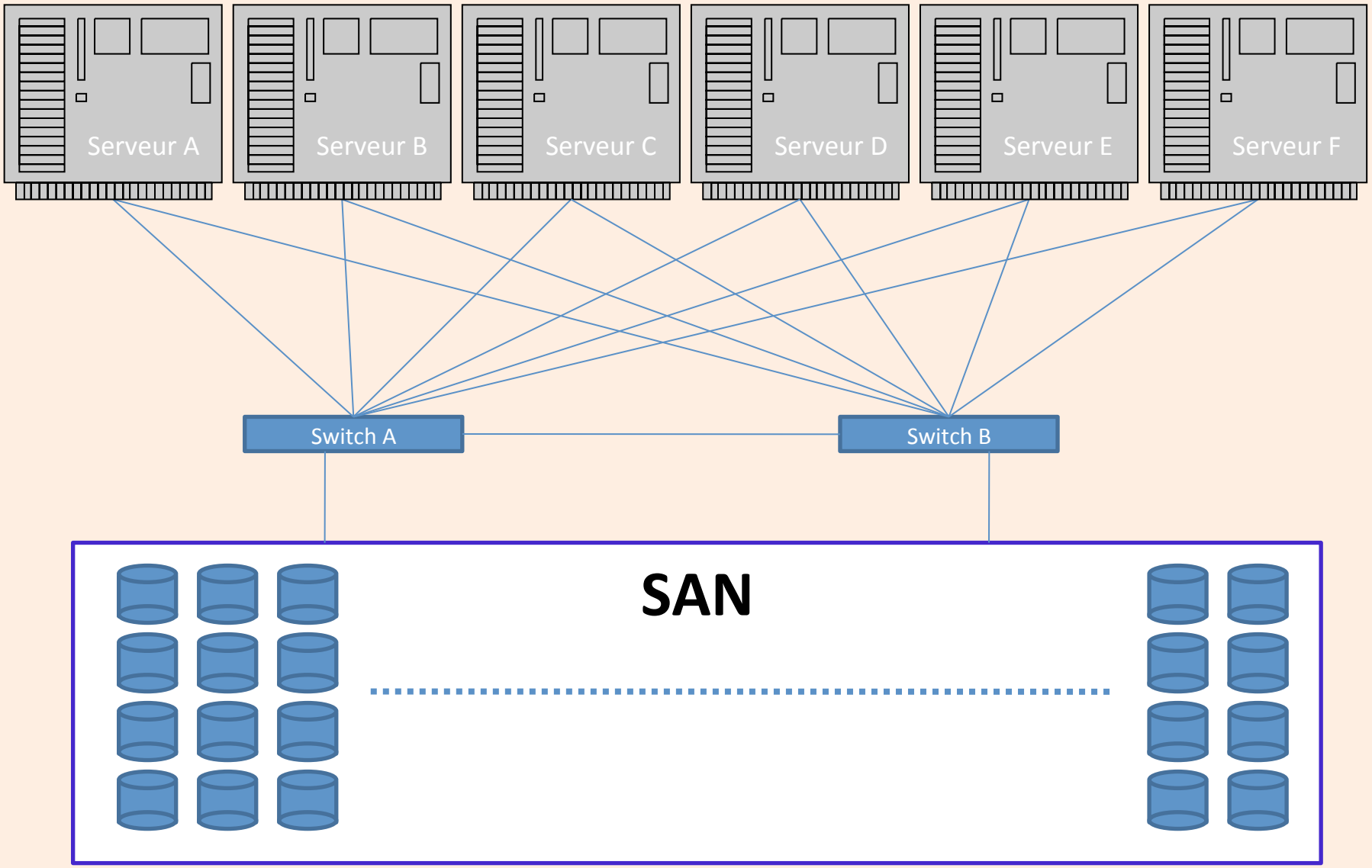
SAN IBM DS8700

SAN Oracle-Sun Exadata

HP – XP 24000

EMC Symmetrix DMX-4

Exemple d'architecture autour d'un SAN



Considérations physiques liées au stockage

- Le nombre de disques donne la bande passante physique « en défaut de cache »
- Le cache doit être dimensionné pour accélérer suffisamment les traitements dits OLTP
- Le type du stockage doit être pris en compte au moment de la constitution des unités logiques
- Le nombre de serveurs connectés et leur activité doivent être tenus en compte dans le dimensionnement d'un stockage

Vocabulaire (1/3)

- **E/S ou I/O** : Entrée/Sortie sur un périphérique
- **I/O/s** : I/O par seconde
- **I/O length** : Taille d'une I/O exprimée en Kio, Mio
- **I/O Wait** : Attente I/O « active » et synchrone forçant une interruption de traitement tant que l'I/O n'est pas résolue
- **Bande passante ou I/O Throughput** : Nombre d'I/O/s maximum et débit maximum en octets/seconde ou encore combinaison des deux facteurs
- **SAN switch** : Equipement réseau spécifique au SAN assurant la liaison entre le SAN et les serveurs connectés, un autre SAN, un système de sauvegarde
- **FC ou Fibre Channel** : Fibre optique assurant la transmission de l'information, qu'on peut aussi, par extension, nommer le bus de données
- **RAID ou Redondant Array of Inexpensive Disks** : Architecture hardware ou software de redondance des données au travers des disques
- **SAN Fabric** : Trio SAN + FC + switches

Vocabulaire (2/3)

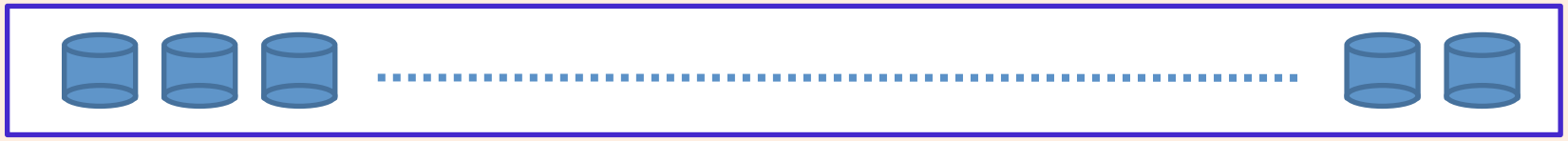
- **SCSI ou Small Computer System Interface** : Ensemble de standards de communication entre un ordinateur et des périphériques
- **iSCSI ou Internet SCSI** : Protocole employé pour connecter des périphériques tels que des SANs avec des serveurs. La couche SCSI est encapsulée dans le protocole TCP/IP et se présente sous la forme de clients (serveurs) et d'un serveur (SAN)
- **LUN ou Logical Unit Number** : Pointeur vers un espace de stockage. Désigne aussi l'espace de stockage par extension
- **WWN ou World Wide Name** : Identifiant unique d'un LUN sur l'ensemble d'un réseau SAN
- **Zoning** : Découpage d'un SAN en plusieurs « zones » indépendantes
- **Latence (I/O)** : Temps nécessaire pour acquérir une I/O
- **SSD ou Solid-State Drive** : Support de stockage de type mémoire-flash. Extrêmement rapide. Défaut actuel : Le support s'altère dans le temps et les « cellules » meurent, créant des trous dans la matrice au fil du temps. Coût très élevé
- **HDD ou Hard-Disk Drive** : Disque dur composé de plateaux et de bras de lecture/ écriture. Bien moins rapide que le SSD, mais plus fiable dans le temps. Coût réduit. Défaut majeur : Pièces mécaniques = Fragilité de fait

Vocabulaire (3/3) – Types les plus courants de RAID

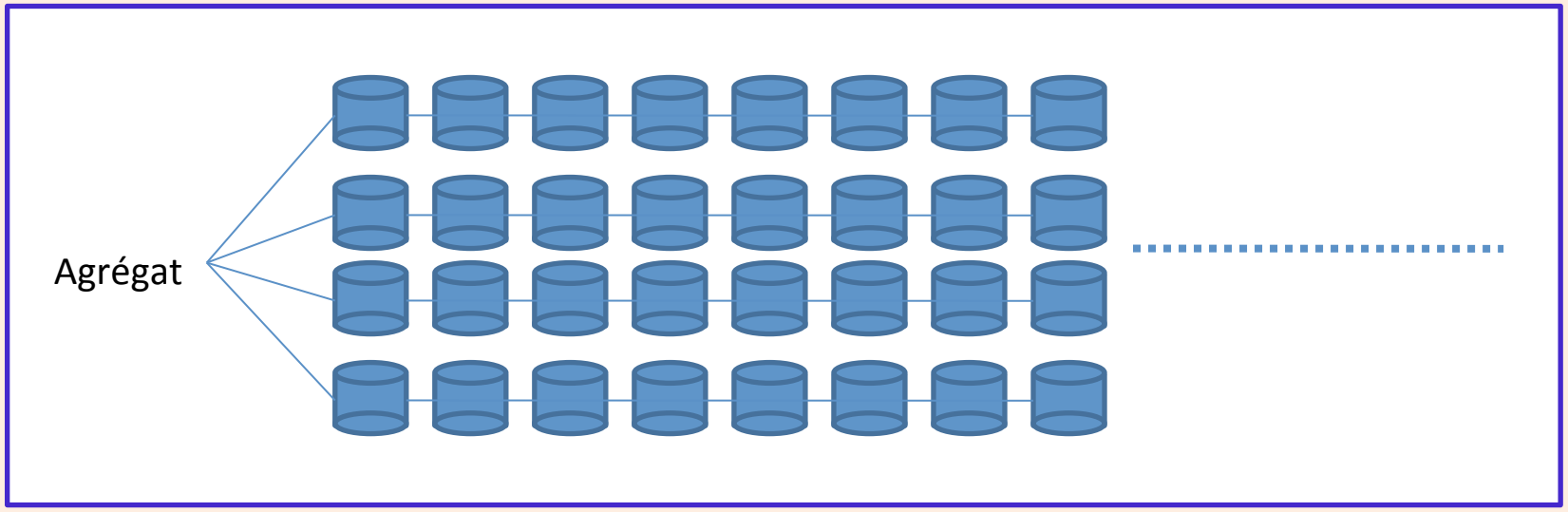
- **RAID 0** : Disk striping → Réunion de plusieurs disques dans le but de faire un espace unique de stockage permettant l'écriture parallèle sur l'ensemble des disques du RAID. Aucune sécurité des données en cas de perte d'un disque
- **RAID 1** : Disk mirroring → Réunion de 2 disques ou plus, chacun étant le miroir de l'autre. Les écritures se font simultanément sur les 2 disques, les lectures se font selon le disque répondant le plus vite
- **RAID 5** : Agrégat de disques avec parité répartie au travers de tous les disques. La perte d'un disque n'entraîne pas la perte des données. Permet le remplacement à chaud d'un disque
- **RAID 6** : RAID 5 évolué. La parité est multiple permettant la possible perte de *n* disques si la parité est multipliée par *n*
- **RAID 0 + 1** : Priorité donnée au RAID 0. La perte d'un disque entraîne la perte d'un élément complet de miroir immédiatement
- **RAID 1 + 0** : Priorité donnée au RAID 1. Plus grande fiabilité que le précédent. Il faut perdre tous les disques d'un miroir pour que le miroir soit perdu

Différents principes de fonctionnement des SANs

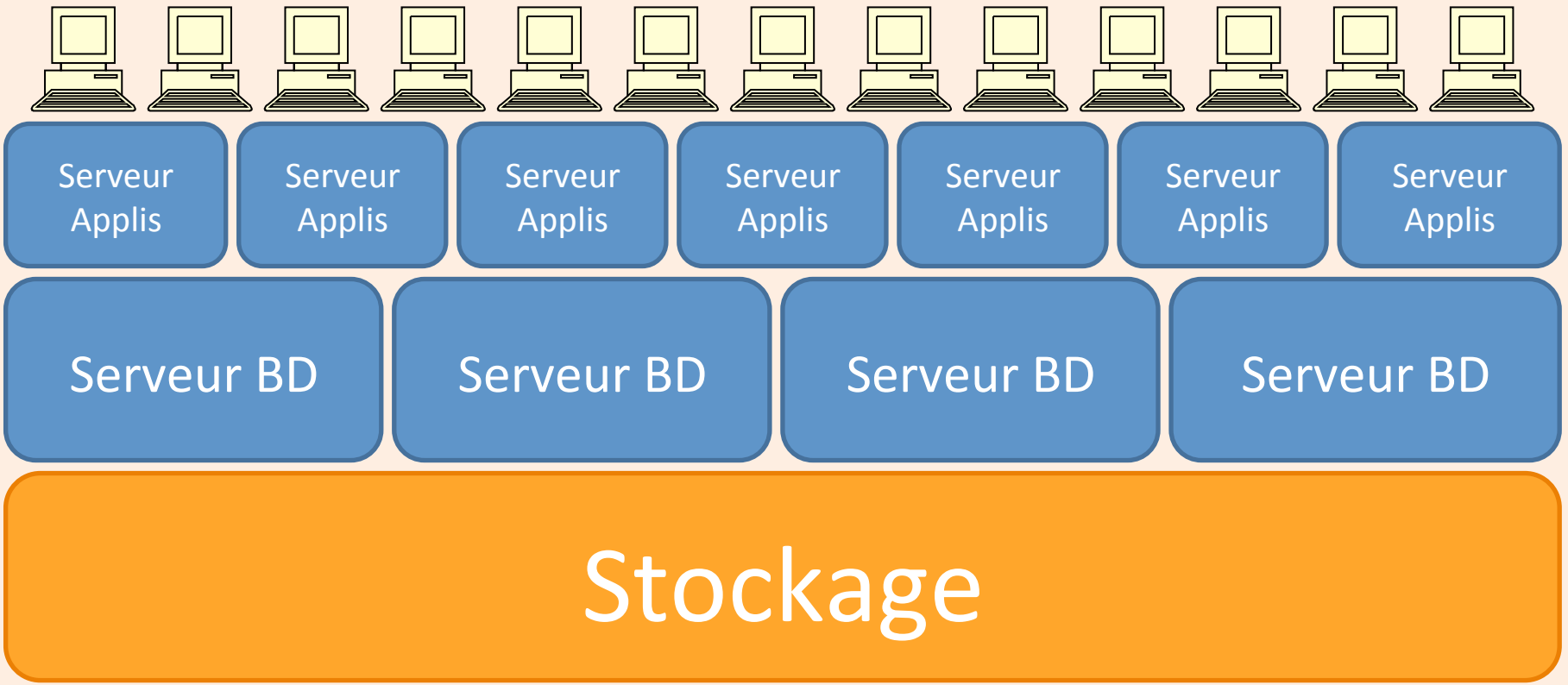
- Deux architectures majeures existent :
  - RAID 6 évolué



  - RAID 5 par grappes



De la performance du stockage à la performance des requêtes clientes



Principes de répartition des données sur le stockage

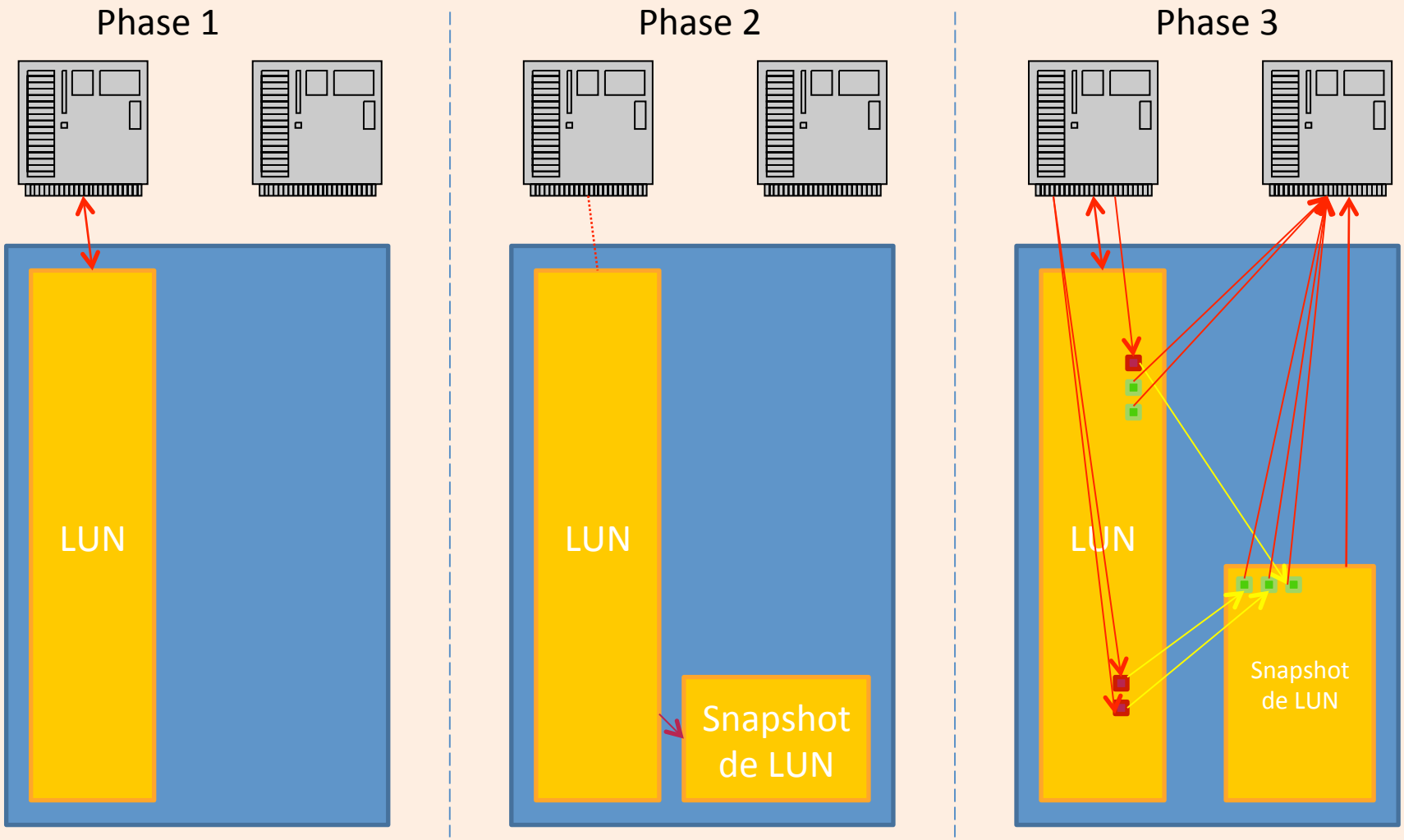
- Il est préférable de disposer d'agrégats contenant le plus grand nombre de disques possible afin de maximiser le flux I/O
- Les journaux de transaction doivent se trouver sur les « axes » les plus rapides en écriture
- Les fichiers de base de données doivent se trouver sur les axes les plus sûrs
- La répartition des fichiers de la base de données doit répondre à la fréquence d'utilisation des données

Architectures complexes

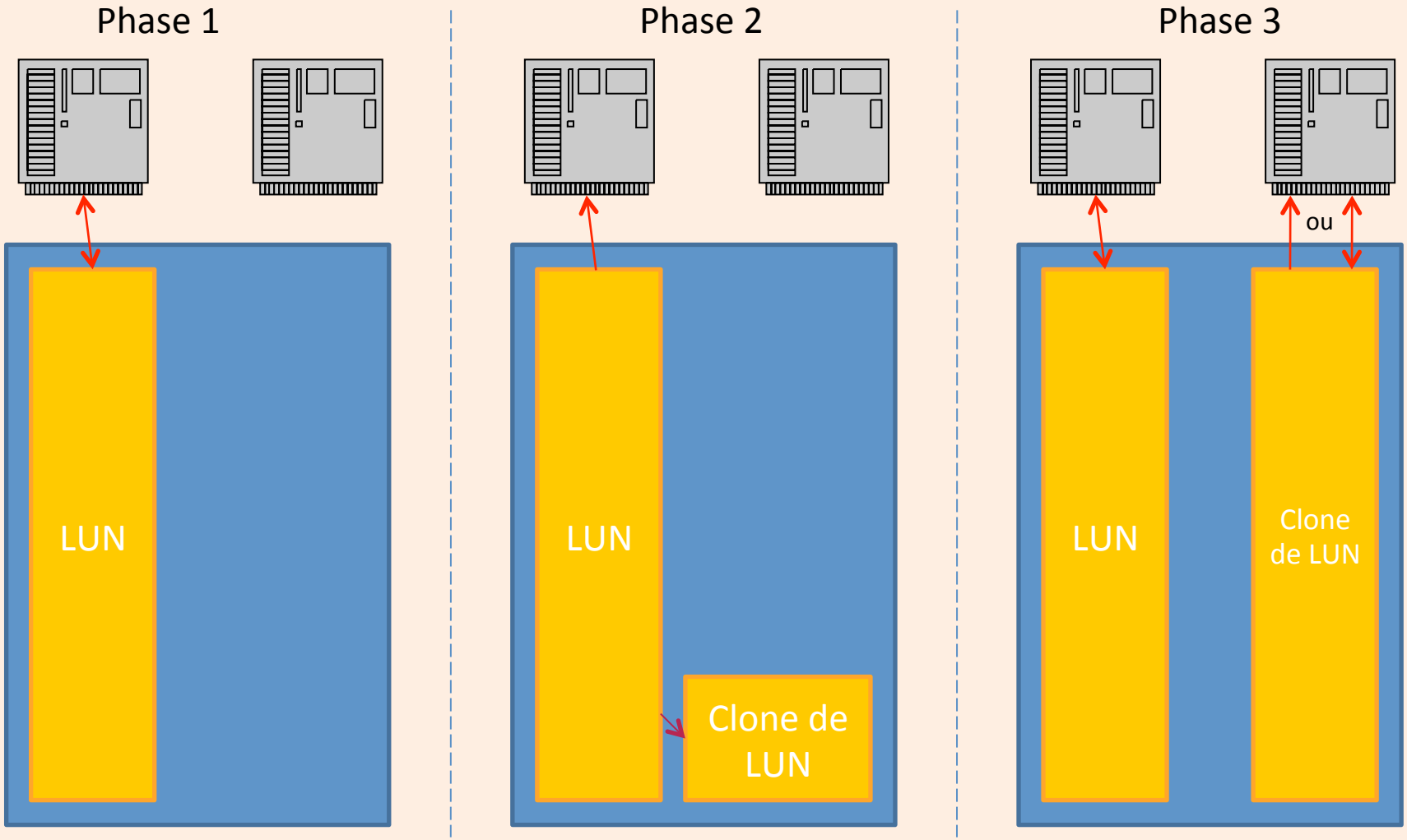
- Réplication de tout ou partie d'un SAN
- Principe de « Continuous Access »
- SAN SSD au-dessus d'un SAN HDD (rôle de cache géant du premier sur le second)

Quelques mécanismes avancés spécifiques aux stockages, couramment employés

Le snapshot de LUN



Le clone de LUN



Le snapclone de LUN

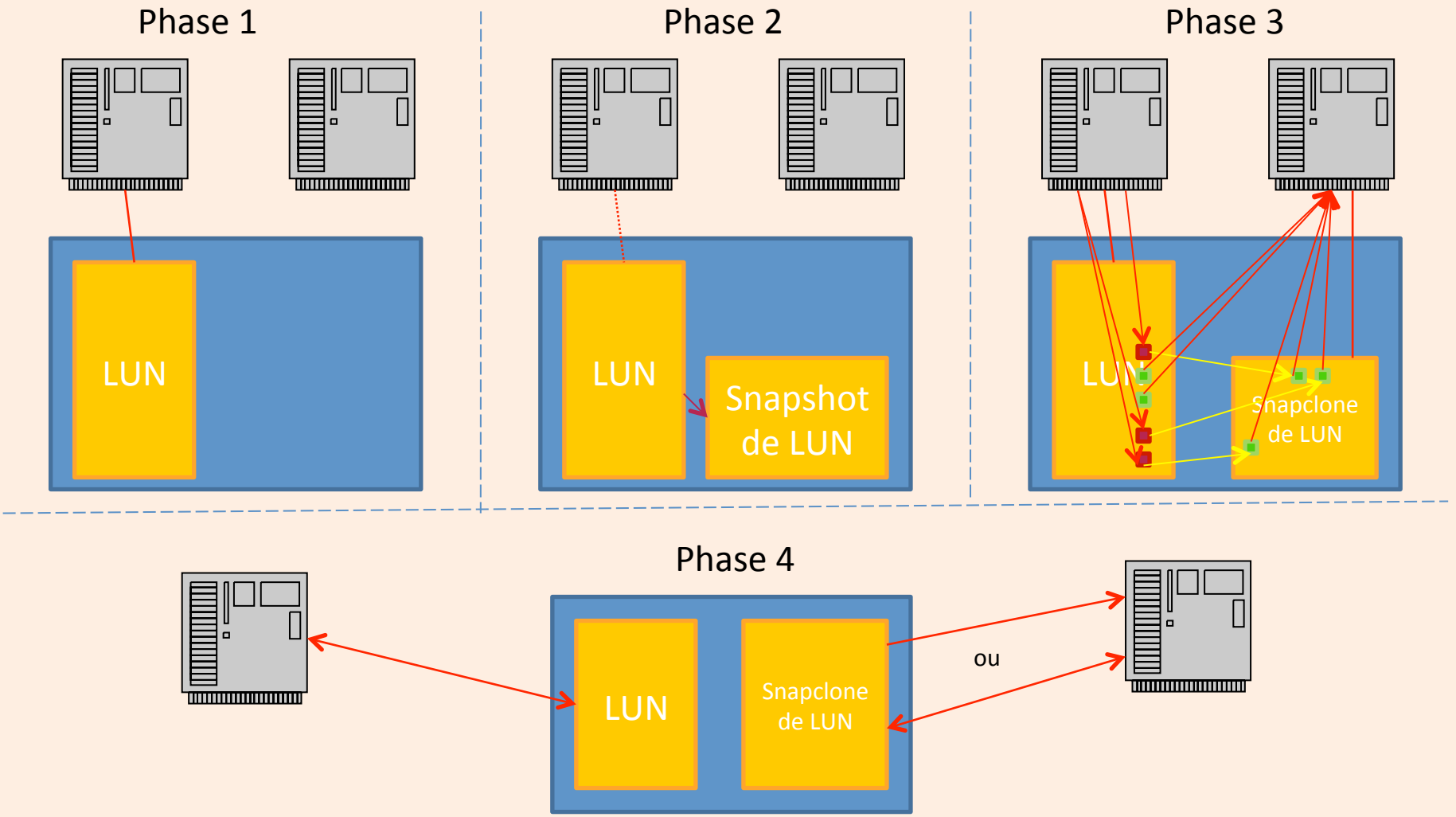


Tableau des différents types de copie d'un LUN

Type de LUN	Forces	Faiblesses
Snapshot	Peu de place sur le stockage. Peu d'I/Os pour le constituer, donc peu de charge induite sur le stockage.	La lecture du snapshot entre en concurrence directe avec les opérations du LUN « copié ». Il en résulte une possible baisse de performance du LUN original. Seule la lecture est possible.
Clone	Ne vient pas en concurrence avec le LUN « copié ». Le LUN cloné peut être disponible en lecture/écriture si besoin.	Prend l'espace strict correspondant au LUN original. Durant toute sa constitution le LUN principal n'est disponible qu'en lecture.
Snapclone	Ne vient en concurrence du LUN copié que dans sa phase de constitution. Laisse le LUN principal totalement disponible en lecture/écriture Ne dégrade pas les performance du LUN original une fois constitué. Le LUN cloné peut être disponible en lecture/écriture si besoin.	Prend l'espace strict correspondant au LUN original.

La sécurité des données – La sauvegarde

- On ne stocke pas des données sans les sauvegarder
- La sauvegarde impacte les performances de la base de données
- Plusieurs types de sauvegardes existent
- Le type de sauvegarde dépend du mode de fonctionnement de la base de données

Les deux types de sauvegarde

- La sauvegarde à froid : Elle oblige à la fermeture de la base de données qui ne peut être réouverte qu'à l'issue de la sauvegarde
- La sauvegarde à chaud : La base de données continue à évoluer durant toute la sauvegarde. La sauvegarde est malgré tout cohérente à la fin de son traitement

Types de sauvegardes – forces/faiblesses

Type de sauvegarde	Forces	Faiblesses
Concepteur du SGBD, utilisation des outils proposés (RMAN, pg_dump, dump database…)	Déclenchement indépendant du système d'exploitation	**Peut induire une charge I/O concurrente** sur les transactions en cours et de ce fait **pénaliser les performances**
SAN (Snapshot, snapclone)	**Limites I/Os très élevées de la baie de stockage** avec utilisation des ressources dédiées de la baie sans sacrifier aux performances du SGBD Utilisation des ressources matérielles de la baie	Nécessite une procédure système/stockage La restauration peut être plus complexe à gérer

Principes de répartition des données sur un SAN

- Fortement dépendant de l'architecture du SAN
- Plus on répartit les fichiers sur un grand nombre de disques, plus on augmente les performances
- Les types de fichiers doivent être soigneusement répartis pour ne pas risquer de concurrence de fonctionnement de la baie (longueur des I/Os par exemple)

Le serveur : « Moteur » du SGBD

Gamme IBM Power Server

Gamme HP Integrity Servers

Gamme Oracle-Sun

Eléments physiques de performance d'un serveur

- Mémoire
- CPU
- Cartes réseau
- Cartes stockage
- Cartes réseau faible latence (mise en cluster)
- Cartes-mères

Eléments logiciels de performance d'un serveur

- Système d'exploitation, calibrage :
  - Gestion mémoire
  - Gestion des sémaphores
  - Gestion du cache système
  - Buffers divers
  - IPC
  - …
- Pilotes périphériques :
  - Disposer des bonnes versions !!
  - Préférer les pilotes constructeurs aux pilotes génériques du système d'exploitation

Calibrage serveur en fonction d'une utilisation SGBDR

- Choix du matériel (PC-like ou autre)
- Choix de l'architecture (64 bits généralement)
- … bis (Itanium, Opteron/Xeon)
- Choix du système d'exploitation
- CPU en nombre et cœurs en nombre
- Mémoire suffisamment dimensionnée
- Arrêt de tous les démons et services inutiles
- Swap de petite taille
- Paramétrage système adapté à l'utilisation

Du socle physique au SGBD

Interdépendance du SGBD et du socle physique

- Le socle physique pourvoit au :
  - Stockage
  - Puissance de traitement délivrée par le serveur
  - Accès à l'information via le serveur
- Le SGBD :
  - S'appuie sur le serveur pour traiter les données
- Une configuration optimum d'un SGBD avec le socle physique doit être réalisée en fonction de l'application-métier

Quel SGBDR pour quelle utilisation ?

MySQL™ (Oracle - Sun)

- Bases de données de petite à moyenne volumétrie (< 400 Go)
- « Gratuit » pour une utilisation hors-entreprise
- Payant lorsqu'il s'agit de déployer de vraies productions et disposer d'un vrai support
- Cluster intégrant un « maître » en lecture/écriture et des « esclaves » en lecture
- Propriétés ACID du noyau « non » de base (InnoDB)
- Rapidité affichée surtout avec les tables ISAM ne nécessitant pas de transactionnel et donc pas de gestion de la concurrence d'accès
- Principalement utilisée pour les sites WEB personnels et professionnels non-critiques ou pour les applications ne nécessitant pas trop de transactions concurrentielles
- Déployable sur un choix important de systèmes d'exploitation
- Utilisation de InnoDB indispensable pour ajouter les capacités transactionnelles et concurrentielles

PostgreSQL

- Bases de données petite à grande volumétrie
- « Gratuit » pour une utilisation hors-entreprise
- Payant pour disposer d'un support
- Payant pour l'utilisation en cluster
- Totalement adhérent aux propriétés ACID
- Moteur de bases de données comparé à Oracle RDBMS en terme de performance et souplesse
- Déployable sur un choix important de systèmes d'exploitation
- Très en pointe dans le domaine des S.I.G.

Sybase™ ASE (Adaptive Server Enterprise)

- Logiciel commercial
- Bases de données de petite à grande volumétrie
- Gestion interne de bloc avec classement des enregistrements (Index Organized Tables) nativement
- Présent sur de nombreuses plateformes
- Essentiellement dédié à l'OLTP, même si certaines entreprises l'utilisent pour du décisionnel
- Totalement adhérent aux propriétés ACID
- Capacité au clustering
- Gestion simplifiée de l'administration
- Réplication des données

Sybase™ IQ

- Logiciel commercial
- Entièrement dédié au décisionnel
- Les blocs contiennent des colonnes et les colonnes sont systématiquement indexées
- A utiliser sur des bases de particulièrement grande volumétrie pour en tirer le potentiel
- Totalement adhérent aux propriétés ACID
- Présent sur de nombreuses plateformes
- Peut être utilisé en cluster

Oracle™ RDBMS

- Logiciel commercial
- Moteur polyvalent (OLTP, Décisionnel, Hybride)
- De petite à très grande volumétrie
- Très grande souplesse de paramétrage… rendant ce paramétrage complexe
- Totalement adhérent aux propriétés ACID
- Présent sur un grand nombre de plateformes
- Capacité de mise en cluster très avancées (leader sur le marché)
- Capacité de gestion de sites de secours très avancées
- Réplication de très haut niveau

IBM™ DB2™ UDB™

- Logiciel commercial
- Moteur polyvalent (OLTP, Décisionnel, Hybride)
- De petite à très grande volumétrie
- Très grande souplesse de paramétrage… rendant ce paramétrage complexe
- Totalement adhérent aux propriétés ACID
- Présent sur un grand nombre de plateformes
- Capacité de mise en cluster sous la forme de bases de données réparties
- Capacité de gestion de sites de secours
- Réplication

Microsoft™ SQL Server™

- Logiciel commercial
- Naissance avec la scission en version 5 de Sybase. Microsoft décide de faire sa propre évolution du serveur « SQL Server », nom du SGBD Sybase à l'époque
- Bases de données de petite à grande volumétrie
- Gestion interne de bloc avec classement des enregistrements (Index Organized Tables) nativement
- Présent UNIQUEMENT sur Windows
- Essentiellement dédié à l'OLTP, même si certaines entreprises l'utilisent pour du décisionnel
- Totalement adhérent aux propriétés ACID
- Capacité au clustering
- Gestion simplifiée de l'administration
- Réplication des données

Conclusion

- Les SGBDs sont très dépendants de l'architecture physique les hébergeant
- Lors de la phase de conception d'une nouvelle application sur une nouvelle architecture, la bonne connaissance de la nature, la quantité et du flux de données traités, permettra de tailler l'architecture physique afin que celle-ci réponde aux exigences de service de l'application
- S'il s'agit d'une évolution applicative, les données de performance issues de la production actuelle doivent permettre d'anticiper sur la puissance demandée par la future application. Il en résultera peut-être une revue de l'architecture physique : évolution ou remplacement
- L'architecture physique ne doit pas être négligée dans la conception d'une application sous peine que celle-ci ne puisse jamais rendre le service pour lequel elle est conçue

Le modèle de données

Le modèle conceptuel

- Il est proche de l'humain
- Il n'est pas accroché à un moteur de bases de données particulier
- Il s'appuie sur le concept entité-relation

Entité-Relation

- Une entité représente une collection de données
- Une relation est un lien entre plusieurs collections de données
- Il se fonde sur la Forme Normale de Boyce-Codd en particulier, et surtout sur la Forme Normale Domaine Clef

Forme Normale

- C'est une forme, une structuration de la donnée
- Cette forme est normée : elle obéit à des règles
- Il s'agit donc de normaliser la donnée
- La normalisation est limitée physiquement à une base de données…
- … plus précisément à un « schéma »
- Les formes normales telles que définie dabs le modèle relationnel sont au nombre de 6

Rôle des formes normales

- Eviter les anomalies transactionnelles :
  - Redondance des données
  - Anomalies de lecture
  - Anomalies d'écriture
  - Contre-performance

1ère Forme normale - 1FN (source Wikipedia)

- Relation dont tous les attributs :
  - contiennent une valeur atomique (les valeurs ne peuvent pas être divisées en plusieurs sous-valeurs dépendant également individuellement de la clé primaire)
  - contiennent des valeurs non répétitives (le cas contraire consiste à mettre une liste dans un seul attribut).
  - sont constants dans le temps (utiliser par exemple la date de naissance plutôt que l'âge).
- Le non respect de deux premières conditions de la 1FN rend la recherche parmi les données plus lente parce qu'il faut analyser le contenu des attributs. La troisième condition quant à elle évite qu'on doive régulièrement mettre à jour les données.

2ème Forme normale - 2FN (source Wikipedia)

- Respecte la deuxième forme normale, la relation respectant la première forme normale et dont :
  - Tout attribut ne composant pas un identifiant dépend d'un identifiant.
- Le non respect de la 2FN entraine une redondance des données qui encombrent alors inutilement la mémoire et l'espace disque.

3ème Forme normale - 3FN (source Wikipedia)

- Respecte la troisième forme normale, la relation respectant la seconde forme normale et dont :
  - Tout attribut ne composant pas un identifiant dépend directement d'un identifiant.
- Le non respect de la 3FN peut également entrainer une redondance des données.

Forme normale de Boyce-Code - FNBC (source Wikipedia)

- Respecte la forme normale de Boyce-Codd, la relation respectant la troisième forme normale et dont :
  - tous les attributs non-clé ne sont pas source de dépendance fonctionnelle (DF) vers une partie de la clé
- Le non respect de la 2FN, 3FN et la FNBC entraîne de la redondance. Une même information étant répétée un nombre considérable de fois.

4ème Forme normale - 4FN (source Wikipedia)

- Pour toute relation de dimension n en forme normale de Boyce-Codd, les relations de dimension n-1 construites sur sa collection doivent avoir un sens. Il ne doit pas être possible de reconstituer les occurrences de la relation de dimension n par jointure de deux relations de dimension n-1. Cette normalisation conduit parfois à décomposer une relation complexe en deux relations plus simples.

5ème Forme normale - 5FN (source Wikipedia)

- Pour toute relation de dimension n (avec n supérieur à 2) en quatrième forme normale, il ne doit pas être possible de retrouver l'ensemble de ses occurrences par jointure sur les occurrences des relations partielles prises deux à deux. Cette normalisation conduit parfois à décomposer une relation complexe en plusieurs relations plus simples.
- Le non respect de la 4FN et 5FN entraîne de la perte de données et les informations manquent de précision.

6ème Forme normale - 6FN (source Wikipedia)

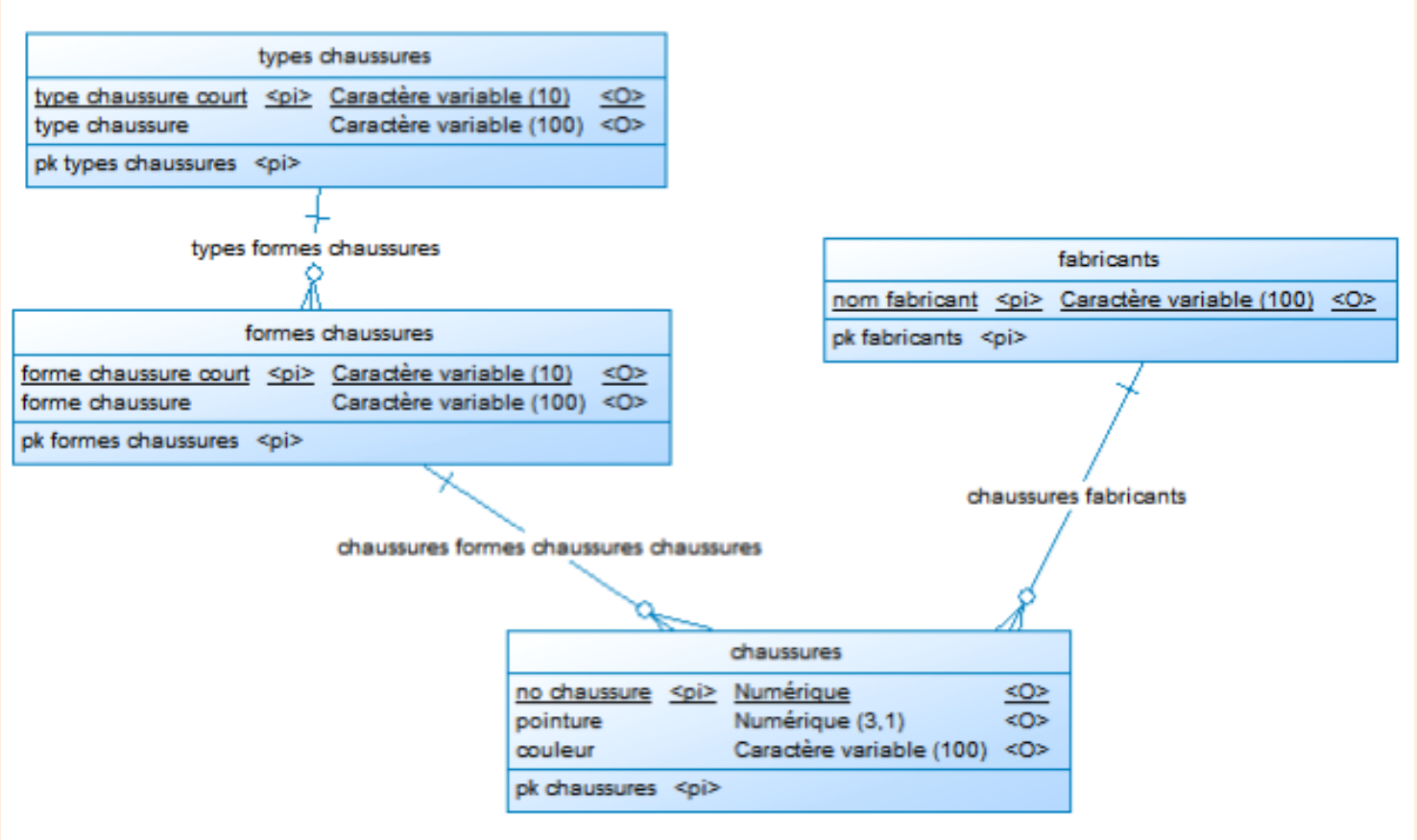
- Une relation est en 6FN si et seulement si, quelle que soit la dépendance de jointure à laquelle elle satisfait, cette dépendance est triviale
- Toute relation en 6FN est aussi de forme 5FN
- Cette 6ème forme normale a pour vocation de traiter les données de nature temporelle
- Une donnée évolue dans le temps et nécessite la conservation d'un historique afin de toujours présenter une cohérence lors de son interrogation « dans le temps »

Forme normale de domaine-clef - FNDC (source Wikipedia)

- Une relation est en FNDC si et seulement si toutes les contraintes sont la conséquence logique des contraintes de domaines et des contraintes de clefs qui s'appliquent à la relation.

Explication du modèle de données et de PowerAMC

Exemple de modèle conceptuel de données (cherchez l'erreur)

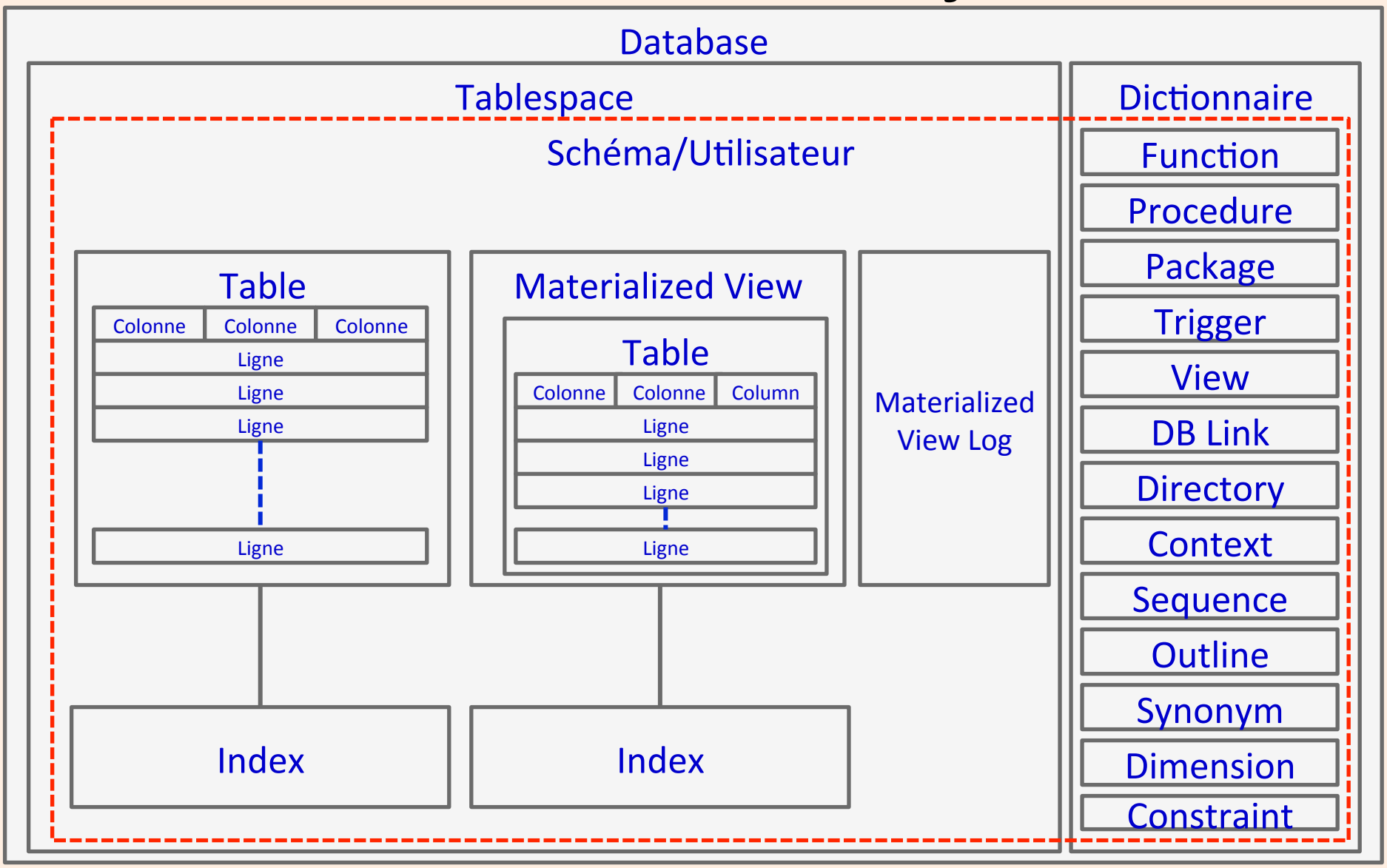


Oracle RDBMS
Les objets

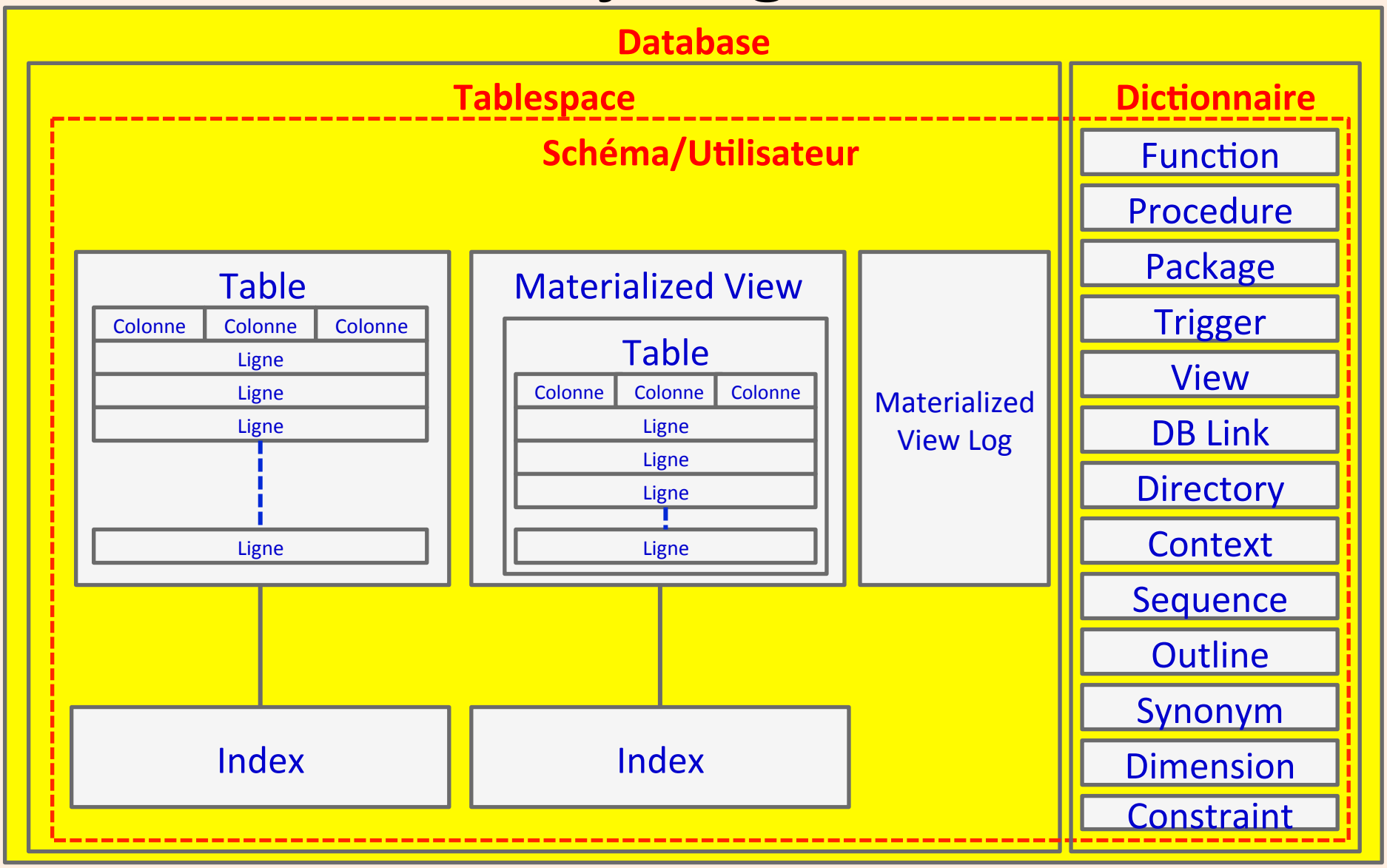
De l'importance de bien connaître les objets disponibles

- On ne « réinvente pas la roue » !
- Les objets contiennent des fonctionnalités natives
- Ils sont gérés par le noyau et bénéficient des propriétés ACID de celui-ci
- Ils permettent de simplifier considérablement le code applicatif tout en l'optimisant

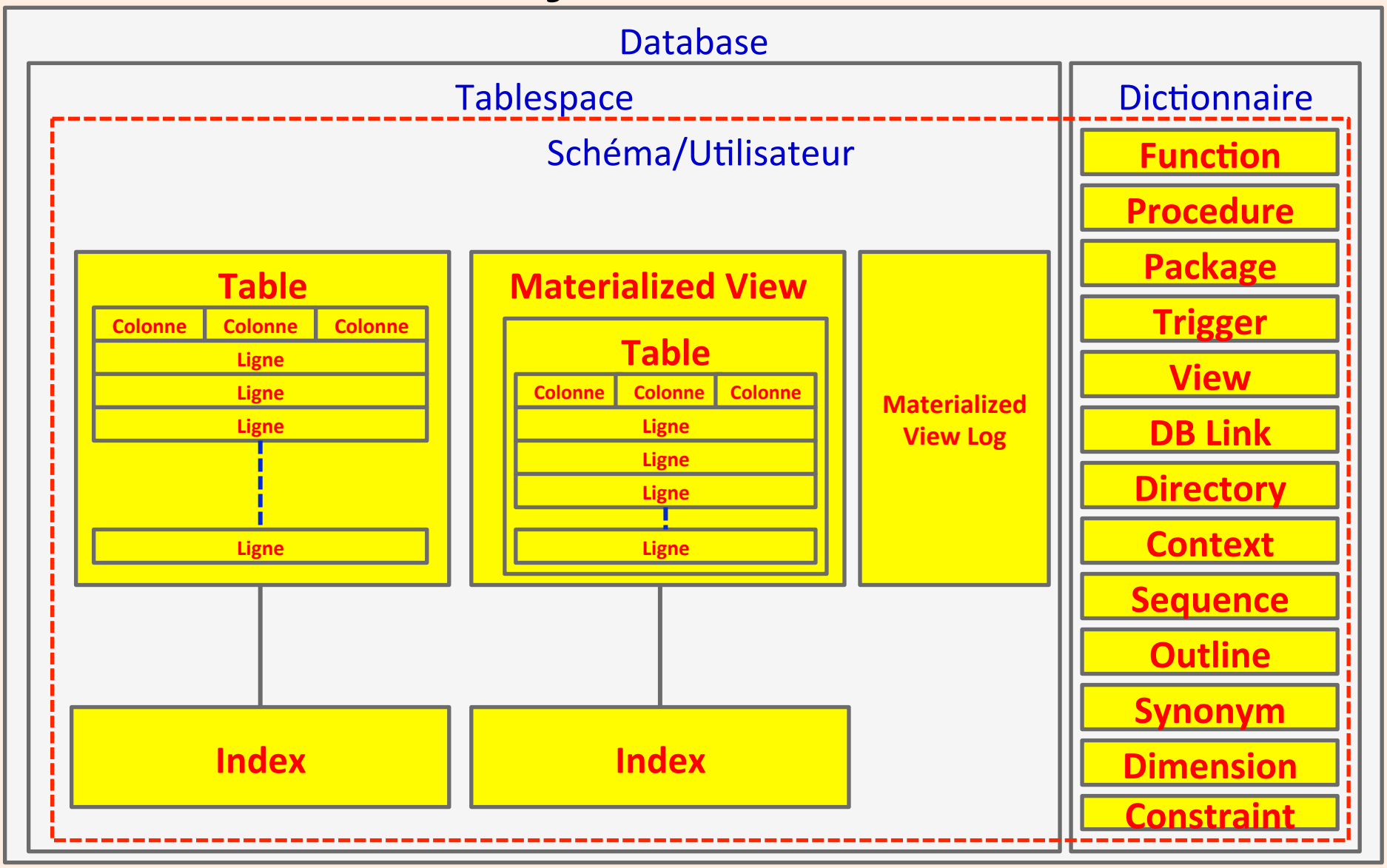
Hiérarchie des objets



Les objets globaux



Les objets du schéma



La base de données

- Manipulations sur la base de données :

```
CREATE DATABASE
ALTER DATABASE
DROP DATABASE
```

Exemple de création d'une base de données

```
CREATE DATABASE "EPITA"
MAXINSTANCES 8
MAXLOGHISTORY 1
MAXLOGFILES 32
MAXLOGMEMBERS 3
MAXDATAFILES 1024
DATAFILE SIZE 2170M   AUTOEXTEND ON NEXT  10240K MAXSIZE UNLIMITED
EXTENT MANAGEMENT LOCAL
SYSAUX DATAFILE SIZE 4070M   AUTOEXTEND ON NEXT  10240K MAXSIZE UNLIMITED
SMALLFILE DEFAULT TEMPORARY TABLESPACE TEMPORARY TEMPFILE  SIZE 670040064
   AUTOEXTEND ON NEXT  640K MAXSIZE UNLIMITED, SIZE 10500M EXTENT MANAGEMENT
   LOCAL  UNIFORM SIZE 1048576
SMALLFILE UNDO TABLESPACE "UNDOTBS1" DATAFILE SIZE 16000M   AUTOEXTEND ON NEXT
   5120K MAXSIZE UNLIMITED
CHARACTER SET WE8ISO8859P1
NATIONAL CHARACTER SET AL16UTF16
LOGFILE GROUP 1  SIZE 1250M,
GROUP 2  SIZE 1250M,
GROUP 3  SIZE 1250M,
GROUP 4  SIZE 1250M
USER SYS IDENTIFIED BY manager1 USER SYSTEM IDENTIFIED BY manager1;
```

Le Tablespace

- Le tablespace est le contenant principal :

```
CREATE TABLESPACE
ALTER TABLESPACE
DROP TABLESPACE
```

Exemple de création d'un tablespace

```
CREATE SMALLFILE TABLESPACE
  USERS LOGGING DATAFILE SIZE 5M
  AUTOEXTEND ON NEXT  1280K
  MAXSIZE UNLIMITED EXTENT
  MANAGEMENT LOCAL SEGMENT SPACE
  MANAGEMENT  AUTO;
```

Le Schéma/Utilisateur

- L'utilisateur (schéma) :

```
CREATE USER
ALTER USER
DROP USER
```

Exemple de création d'un schéma/ utilisateur

```
CREATE USER TOPAD IDENTIFIED BY
  TOPAD
DEFAULT TABLESPACE DATA
TEMPORARY TABLESPACE TEMP
QUOTA UNLIMITED ON DATA;
```

La table

- La table est le principal contenant des données :

```
CREATE TABLE
ALTER TABLE
DROP TABLE
```

Exemple de création d'une table

```
create table CALLS  (
   CALL_ID                    NUMBER(21)     not null,
   PHONE_NUM                  NUMBER(20,0)   not null,
   START_CALL_DTE             DATE           not null,
   START_TRANSIT_CALL_DTE     DATE           not null,
   CALL_COUNTRY_CODE          NUMBER(3,0)    not null,
   CALL_NUM                   NUMBER(20,0)   not null
)
STORAGE(MINEXTENTS 10)
TABLESPACE callsmanagement;
```

L'index

- L'index est l'objet qui permet d'accéder aux données de la table sans nécessiter de balayer l'ensemble de la table :

```
CREATE INDEX
ALTER INDEX
DROP INDEX
```

Exemple de création d'un index

```
CREATE UNIQUE INDEX pk_calls ON
calls(call_id)
TABLESPACE callsmanagementidx;
```

Les contraintes d'intégrité

- Les principales sont :
  - Primary key
  - Unique key (Voir l'exemple précédent)
  - Foreign key
  - Not null
  - Check (Liste de valeurs)

Syntaxe :

```
ALTER TABLE ADD CONSTRAINT
ALTER TABLE DROP CONSTRAINT
```

Exemple d'ajout d'une contrainte

```
ALTER TABLE calls ADD CONSTRAINT
pk_calls PRIMARY KEY
(call_id)
USING INDEX;
```

Le trigger

- Le trigger est une contrainte d'intégrité personnalisée. Si les contraintes standard ne suffisent pas, la création d'un trigger devient nécessaire :

```
CREATE TRIGGER BEFORE/AFTER/INSTEAD OF
  INSERT/UPDATE [OF column,...]/DELETE
  REFERENCING OLD AS ... NEW AS ...
  PARENT AS ...[FOR EACH ROW] ON
  <table>
BEGIN
  ...
END;
```

Quel est le mode de fonctionnement d'un trigger ?

- Le trigger est défini pour une table, une montée de version applicative, un changement de structure d'objet, une action au niveau base de données
- Pour une table :
  - Il peut être déclenché sur un ordre insert, update ou delete
  - Sur chaque ligne ou une fois toutes les lignes modifiées
  - Il peut être déclenché avant, après ou « à la place de » (en remplacement de l'ordre)
- Il peut avoir besoin de connaître les valeurs présentes « avant » pour pouvoir les utiliser dans le code
- Deux variables spécifiques existent : OLD et NEW
- A la fin du processus les valeurs ont été modifiées à la convenance du développeur

Points importants à retenir lorsqu'on utilise des triggers

- Le trigger est écrit en PL/SQL et n'est pas natif
- Attention à la possible cascade de triggers entre objets mis à jour !
- Attention aux effets non voulus pouvant être déclenchés par des triggers possédant des clauses « pragma »

Exemple de trigger

```
create or replace trigger load_keep_cache after startup on database
declare
stmt1 varchar2(1000);
cur_tbl varchar2(70);
compte number;
cursor cs is select tbl from keep_cache_tbl;
begin
for i in cs loop
   cur_tbl := i.tbl;
   stmt1 := 'SELECT /*+ FULL(' || cur_tbl || ') CACHE(' || cur_tbl ||
   ') */ COUNT(1) FROM ' || cur_tbl;
   execute immediate stmt1 INTO compte;
   insert into log_start_keep_cache(stmt,dte,cpte)
   values(stmt1,sysdate,compte);
end loop;
commit;
end;
/
```

La procédure

- La procédure est un programme exécuté côté serveur directement par le noyau de base de données. Elle ne retourne aucune valeur :

```
CREATE PROCEDURE AS
BEGIN
END;
```

Exemple de procédure

```
create or replace procedure load_keep_cache as
stmt1 varchar2(1000);
cur_tbl varchar2(70);
compte number;
cursor cs is select tbl from keep_cache_tbl;
begin
for i in cs loop
   cur_tbl := i.tbl;
   stmt1 := 'SELECT /*+ FULL(' || cur_tbl || ') CACHE(' || cur_tbl ||
   ') */ COUNT(1) FROM ' || cur_tbl;
   execute immediate stmt1 INTO compte;
   dbms_output.put_line('Compte ' || cur_tbl || ' : ' || compte);
   insert into log_start_keep_cache(stmt,dte,cpte)
   values(stmt1,sysdate,compte);
end loop;
commit;
end;
/
```

La fonction

- La fonction est un programme exécuté côté serveur directement par le noyau de base de données. Elle retourne une valeur :

```
CREATE FUNCTION RETURN AS
BEGIN
END;
```

Exemple de fonction

```
create or replace function load_keep_cache return number as
stmt1 varchar2(1000);
cur_tbl varchar2(70);
compte number;
compte_glob number;
cursor cs is select tbl from keep_cache_tbl;
begin
compte_glob := 0;
for i in cs loop
    cur_tbl := i.tbl;
    stmt1 := 'SELECT /*+ FULL(' || cur_tbl || ') CACHE(' || cur_tbl || ') */
    COUNT(1) FROM ' || cur_tbl;
    execute immediate stmt1 INTO compte;
    dbms_output.put_line('Compte ' || cur_tbl || ' : ' || compte);
    insert into log_start_keep_cache(stmt,dte,cpte) values(stmt1,sysdate,compte);
    compte_glob := compte_glob + compte;
end loop;
commit;
return(compte_glob);
end;
/
```

Le package

- Le package est un ensemble de procédure(s) et/ou fonction(s)
- Le package est constitué d'un « signature » et d'un « corps » (body)

```
CREATE PACKAGE <package name> AS
<set of proc/func signatures>
END <package name>;
CREATE PACKAGE BODY <package name> AS
<set of proc/func>
END <package name>;
```

Exemple de package

```
create or replace package manage_cache as
    procedure load_keep_cache;
    function last_date_cache_loaded return date;
end manage_cache;

create or replace package body manage_cache as

    procedure load_keep_cache as
      stmt1 varchar2(1000);
      cur_tbl varchar2(70);
      compte number;
      compte_glob number;
      cursor cs is select tbl from keep_cache_tbl;
      begin
            compte_glob := 0;
            for i in cs loop
               cur_tbl := i.tbl;
               stmt1 := 'SELECT /*+ FULL(' ||
               cur_tbl || ') CACHE(' || cur_tbl ||
               ') */ COUNT(1) FROM ' || cur_tbl;
               execute immediate stmt1 INTO compte;
               dbms_output.put_line('Compte ' ||
               cur_tbl || ' : ' || compte);
               insert into
               log_start_keep_cache(stmt,dte,cpte)
               values(stmt1,sysdate,compte);
               compte_glob := compte_glob +
               compte;
            end loop;
            commit;
            return(compte_glob);
      end;

    function last_date_cache_loaded return date as
      date_cache_loaded date;
      begin
          compte := 0;
          select max(dte) into date_cache_loaded
          from log_start_keep_cache;
          return(date_cache_loaded);
      end;
end manage_cache;
/
```

Oracle RDBMS
SQL – PL/SQL

Introduction au SQL

Le SQL

- ... est l'acronyme pour "Structured Query Language"
- Utilise des mots courants pour exprimer des requêtes sur les bases de données
- Et plus ou moins riche fonction du noyau de bases de données
- Est un ensemble d'instructions classées en deux groupes principaux :
  - Data Definition Language (DDL), ou langage de définition des données
  - Data Manipulation Language (DML), ou langage de manipulation des données

Data Definition Language (DDL)

- Toutes les instructions pour créer :
  - Une base de données
  - Un tablespace
  - Une table
  - Un index...
- ... Pour réaliser des opérations globales :
  - Effacement total du contenu de tables
  - Reconstruction d'index
  - Manipulation des partitions de tables/index
  - Déplacement de tables...

Data Manipulation Language

- Toutes les instructions pour :
  - Insérer des données dans les tables
  - Mettre à jour les données dans les tables
  - Effacer les données des tables
  - Sélectionner les données

Data Definition Language (DDL)

Les instructions

- Aller voir à l'URL http://tahiti.oracle.com pour le détail de toutes les instructions pour la version 11gR2 d'Oracle 11gR2, et plus particulièrement le livre : "SQL Language Reference"
  http://download.oracle.com/docs/cd/E11882_01/server.112/e10593/sqlqr01001.htm#i99574

Data Manipulation Language (DML)

Les instructions

- ... sont :
  - SELECT
  - INSERT
  - UPDATE
  - DELETE

SELECT

- L'instruction principale utilisée après s'être connecté à une base de données
- Trouver les données que l'on veut utiliser
- Présenter les données dans une forme compréhensible pour l'être humain

SELECT – syntaxe (1)

```
SELECT [DISTINCT] <col1>,...
FROM <table1>
JOIN <table2> ON <condition>
JOIN ...
WHERE <condition>
GROUP BY <liste de colonnes>
HAVING <condition>
ORDER BY <liste de colonnes>
```

SELECT – syntaxe (2)

```
SELECT ...
UNION [ALL]/MINUS/
 INTERSECT
SELECT
...
```

SELECT – syntaxe (3)

```
SELECT /*+hint */ ...

SELECT ... FROM
  (SELECT ... FROM ...) t1

SELECT ... WITH ...
```

et bien plus encore. Voir ce lien :

http://download.oracle.com/docs/cd/E11882_01/server.112/e10592/statements_10002.htm#SQLRF01702

SELECT – syntaxe pour le PL/SQL

```
SELECT <col1>,...
INTO <var1>,...
FROM ...
WHERE ...

SELECT ... FOR UPDATE
```

INSERT – syntaxe

```
INSERT INTO
  <table>(<col1>,...)
VALUES(<litteral>,...)

INSERT INTO <table>(col1,...)
  SELECT ...
INSERT INTO ... RETURNING
  <var1>,...
```

UPDATE – syntaxe

```
UPDATE <table>
SET <col1> = <litteral1>
...
WHERE
...

UPDATE <table>
SET (<col1>,...) =
(SELECT <col1a>,... FROM ... WHERE ...)
WHERE
...

UPDATE ... RETURNING <var1>,...
```

DELETE – syntaxe

```
DELETE FROM <table> WHERE
  <condition>

DELETE FROM ... RETURNING
  <var1>,...
```

Types SQL

Types Nombres

- **NUMBER [(précision[,décimales])]**: Nombre compris entre 1.0 X 10^{-130} et 1.0 X 10^{126}
- **FLOAT[(precision)]**: Sous-type de nombre sans notion de décimales. Le nombre de décimales est directement interprêté des données
- **BINARY_FLOAT**: Nombre flottant codé sur 32-bits en simple précision
- **BINARY_DOUBLE**: Nombre flottant codé sur 64-bits en double précision

Types Caractères

- **CHAR[(size [BYTE|CHAR])]**: Chaîne de caractères de longueur fixe. Longueur par défaut : 1
- **NCHAR[(size [BYTE|CHAR])]**: Chaîne de caractères de longueur fixe et de type “national” (National character set). Longueur par défaut : 1
- **VARCHAR2[(size [BYTE|CHAR])]**: Chaîne de caractères de longueur variable. Longueur par défaut : 1
- **NVARCHAR2[(size [BYTE|CHAR])]**: Chaîne de caractères de longueur variable et de type “national” (National character set). Longueur par défaut : 1

Types Date/Heure

- **DATE**: Date comprise entre -4712/01/01 et 9999/01/01 contenant optionnellement des heures, minutes et secondes sans fraction de seconde
- **TIMESTAMP[(précision en fraction de seconde)] [WITH [LOCAL] TIME ZONE]**: Identique au type “date” et doté d’une précision en fraction de seconde. “With time zone” implique la gestion du fuseau horaire pour la partie “heures”. “Local” indique que l’heure est délivrée en fonction du fuseau horaire de la session
- **INTERVAL YEAR[(précision année)] TO MONTH**: Période de temps stockée en années et mois avec une notion de précision sur l’année de 1 à 9 chiffres
- **INTERVAL DAY[(précision année)] TO SECOND [(Précision en fraction de seconde)]**: Période stockée en jours et secondes avec une précision de 1 à 9 chiffres sur la partie “jour” et une précision de 0 à 9 chiffres sur les fractions de seconde

Types “Large Object”

- **BLOB**: ... Binary Large OBject. Taille maximum
  (4 Go - 1) x (Taille du bloc de données)
- **CLOB**: ... Character Large OBject contenant une chaîne de caractères mono ou multi-octets. Taille maximum
  (4 Go - 1) x (Taille du bloc de données)
- **NCLOB**: ... Character Large OBject contenant des caractères Unicode utilisant le “National character set”. Taille maximum
  (4 Go - 1) x (Taille du bloc de données)
- **BFILE**: Un pointeur vers un fichier binaire stocké hors de la base de données. Taille maximum du fichier : 4 Go

Types Row Id

- **ROWID**: Chaîne codées en “Base 64” représentant l’adresse unique d’une ligne dans une table
- **UROWID[(taille)]**: Identique au type ROWID, mais concernant les tables organisées sous forme d’index. La taille maximum et la valeur défaut sont 4 000 octets.

Types Long et Raw

- **LONG**: Données de type caractère de longueur variable pouvant atteindre 2 Go. Type fourni par Oracle pour des questions de compatibilité
- **LONG RAW**: Données de type binaire pouvant atteindre une longueur maximum de 2 Go
- **RAW(taille)**: Données de type binaire de longueur maximum de 2 000 octets. La taille doit être précisé.

Opérateurs SQL

Opérateurs SQL (1)

- “+”, “-” unaire : Indicateur de signe
- “+”, “-”, “*”, ”/” binaire : Addition, Soustraction, Multiplication, Division
- “||” : Concaténation
- PRIOR : Opérateur hiérarchique dans une requête contenant “CONNECT BY”
- CONNECT_BY_ROOT unaire : Opérateur hiérarchique excluant les conditions “CONNECT BY” et “START WITH”

Opérateurs SQL (2)

- UNION: Toutes les lignes distinctes des requêtes liées
- UNION ALL: Identique à UNION mais en conservant les lignes dupliquées
- INTERSECT: Toutes les lignes distinctes contenues dans les deux requêtes
- MINUS: Toutes les lignes de la 1ère requête qui ne sont pas contenues dans la seconde

Fonctions SQL

Principales fonctions numériques (1)

- ABS(n): Valeur absolue de n
- ACOS(n): Arc-cosinus de n (radians)
- ASIN(n): Arc-sinus de n (radians)
- ATAN(n): Arc-tangente de n (radians)
- CEIL(n): Valeur entière suivant n (fractional)
- COS(n): Cosinus de n (radians)
- COSH(n): Cosinus-hyperbolique de n (radians)
- EXP(n): Exponentielle de n
- FLOOR(n): Valeur entière précédant n (fractional)
- LN(n): Logarithme naturel de n
- LOG(n): Logarithme base 10 de n

Principales fonctions numériques (2)

- MOD(m,n): Reste de la division de m par n (FLOOR est utilisée)
- POWER(m,n): m puissance n
- REMAINDER(m,n): Reste de la division de m par n (ROUND est utilisée)
- ROUND(m,n): Arrondi de m à une précision de n décimales
- SIGN(n): 1 → positif, 0 → zéro, -1 → négatif
- SIN(n): Sinus de n (radians)
- SINH(n): Sinus-hyperbolique de n (radians)
- SQRT(n): Racine carrée de n positif ou zéro. Une valeur négative renvoit “NaN”
- TAN(n): Tangente de n (radians)
- TANH(n): Tangente-hyperbolique de n (radians)
- TRUNC(m,n): FLOOR(m) à une précision de n décimales

Principales fonctions de manipulation de chaînes de caractères

- CHR(n): Caractère issu du code n
- CONCAT(s1,s2): Concaténation de s1 avec s2
- LOWER(s): minuscules de s
- LTRIM(s): s sans les espaces de début de chaîne
- REPLACE(c,s,r): Recherche de s dans c et remplacé par r
- RTRIM(s): s sans les espaces de fin de chaîne
- SUBSTR(s,m,n): Sous-chaîne de s à partir de la position m avec n caractères de longueur. Si m est zéro il est traité comme 1
- TRIM(s): Identique à RTRIM(LTRIM(s))
- UPPER(s): MAJUSCULES de s
- ASCII(c): Code ASCII de c
- INSTR(s1,s2[,n[,m]]): Retourne la première position de s2 dans s1 en démarrant la recherche au caractère n et à l'occurrence m de s2
- LENGTH(s): Longueur en caractères de s

Fonctions date/heure principales

- ADD_MONTHS(d,n): Ajout de n mois à d (date)
- CURRENT_DATE: Retourne la date grégorienne courante
- CURRENT_TIMESTAMP[(n)]: Retourne la date grégorienne courante "exacte"avec une précision de n décimales
- EXTRACT(*période* FROM d): Retourne la période de d. La période peut être year, month, day, hour, minute, second, timezone_hour, timezone_minute, timezone_region, timezone_abbr
- MONTHS_BETWEEN(d1,d2): Nombre de mois entre d1 et d2
- ROUND(d,fmt): Arrondi de la date d selon le format fmt
- SYSDATE: Retourne la date courante du serveur
- TO_CHAR(d[,f[,n]]): Convertit une date en chaîne de caractères dans un format spécifié avec un paramètre NLS
- TO_TIMESTAMP(s[,f[,n]]): Convertit une chaîne de caractères en date dans un format spécifié avec un paramètre NLS
- TRUNC(d,f): Tronque une date d avec le format f

Principales fonctions de conversion

- CONVERT(s,dc,sc): Convertit s dans le jeu de caractères “dc” à partir du jeu de caractères source “sc”
- TO_CHAR(ns): Convertit la chaîne “ns”dans le jeu de caractère principal de la base de données
- TO_CHAR(d,f,n): Convertit une date dans un format avec un jeu de caractères choisi
- TO_CHAR(n,f,nls): Convertit n en chaîne de caractères dans un format avec un choix de jeu de caractères
- TO_DATE(s,f,n): Convertit une chaîne de caractères formatée selon f et issue d’un jeu de caractères en date
- TO_NUMBER(s,f,n): Convertit une chaîne de caractères formatée selon f et issue d’un jeu de caractères en nombre
- TO_TIMESTAMP(s,f,n): Convertit une chaîne de caractères formatée selon f et issue d’un jeu de caractères en date

Principales fonctions d'agrégats

- AVG(expr): Moyenne d'une colonne
- COUNT(expr): Nombre de lignes d'une requête
- MAX(expr): Valeur maximum d'une colonne d'une requête
- MIN(expr): Valeur minimum d'une colonne d'une requête
- SUM(expr): Somme d'une colonne d'une requête

Le PL/SQL

Qu'est le PL/SQL?

- C'est un langage structuré
- Standard
- Portable
- Intégré à la base de données
- Avec des performances élevées de traitement
- Langage fortement intégré
- Compilable côté serveur

Structure générique du PL/SQL

```
DECLARE
   …
BEGIN
   DECLARE
      …
   BEGIN
      DECLARE
         …
      BEGIN


      END;
   END;

END;
```

Bloc PL/SQL non nommé

```
declare
   <définition des variables>
begin
   <Code PL/SQL>
end;
/
```

PL/SQL dans une fonction ou une procédure ou un déclencheur

```
create or replace {procedure/
  function/trigger} <nom>
  [(variables)] [uniquement pour
  les fonctions : return variable]
  as
   <définition des variables>
begin
   <Code PL/SQL>
end;
/
```

Variables bind et code (1)

- Une “bind variable” est :
  - Une variable définie dans le programme
  - Hors du PL/SQL (et du SQL)
  - Une variable utilisée pour passer des arguments à du code PL/SQL ou SQL
  - Utilisée pour la réutilisation du code (Optimisation)
- Une variable de code est :
  - A l’intérieur du code
  - Utilisée comme dans tout autre langage
  - Prend le rôle de “bind variable” quand elle est utilisée dans du PL/SQL dans une instruction SQL

Variables bind et code (2)

Variables bind (Exemple avec SQL*Plus):

```
variable name varchar2(100)
begin
   select upper(t1_name) from t1
  into :name;
   insert into t2(t2_name)
  values(:name);
   commit;
end;
/
```

Variables bind et code (3)

Variables de code :

```
declare
  name varchar2(100);
begin
  select upper(t1_name) from t1 into name;
  insert into t2(t2_name) values(name);
  commit;
end;
/
```

Portée des variables dans le code

```
DECLARE
  …
BEGIN
    DECLARE
      …
    BEGIN
        DECLARE
          …
        BEGIN


        END;
    END;
END;
```

- Chaque variable déclarée est visible dans son propre bloc
- Pour voir une variable globale il est nécessaire de faire précéder le nom de la variable avec le nom de la procédure/fonction ou le nom du bloc

Variables dans la signature d'une procédure/fonction/déclencheur

```
create or replace {procedure/function/
  trigger} <nom> (<nom de la variable> [IN|
  OUT|IN OUT] <type de la variable> [DEFAULT
  valeur],...)
```

IN: Le défaut. La variable est passée avec une valeur, sans valeur de retour attendue

OUT: La variable est passée vide et le code retourne une valeur dans cette variable

IN OUT: La variable est passée avec une valeur et est retournée avec une valeur identique ou différente (suite à transformation dans le code)

DEFAULT valeur : Si aucune valeur n'est passée dans une variable, la valeur par défaut est assignée à la variable en question

Appel d'une procédure/fonction

```
declare
  res varchar2(100);
begin
  proc1(
    var2 => :val2,
    var1 => :val1
  ); -- Nommage de la variable sans prendre
  en considération l'ordre propre à la
  signature  res := func1(:val3,:val4); --
  Utilisation de l'ordre naturel des
  variables dans la signature
end;
/
```

Programmes imbriqués PL/SQL

```
PROCEDURE p1
  …
BEGIN
   PROCEDURE p1_sp1
     …
   BEGIN
     FUNCTION p1_sp1_sf1
       …
     BEGIN


     END;
   END;

END;
```

Character Set

- Le character set, ou jeu de caractères, dépend directement du character set de la base
- Attention en codant à l'extérieur de la base de données à bien respecter le character set !!

Principaux opérateurs

- ":=" : Opérateur d'assignation
- "%" : Indicateur d'attribut
- "@" : Indicateur de connexion externe
- ":" : Indicateur de variable hôte (ou bind)
- "**": Opérateur d'exponentiation
- "<>, !=, ^=, ~=": Opérateur "différent de"
- "||": Opérateur de concaténation
- "=>": Opérateur d'association pour la position des variables dans l'appel de fonction/procédure
- "..": Opérateur d'intervalle de valeurs
- "--": Opérateur de commentaire mono-ligne
- "/* */": Opérateur de commentaire multi-lignes
- "<< >>": Nommage explicite de bloc (pour le nommage de variables globales)

Littéraux

- Nombre : 12, 12.345, 3.1415926535f, 44D, NULL
- Chaîne : 'Un exemple de texte', q'!autre chose', NULL
- Intervalle de temps: INTERVAL '42-7' YEAR TO MONTH, INTERVAL '-4' MONTH, NULL
- Booléen : TRUE, FALSE, NULL
- Date: TO_DATE(‘14-JUN-2010','DD-MON-YYYY')

Délimiteur “point-virgule”

- Le délimiteur “point-virgule” est le délimiteur de fin d’instruction

```
BEGIN
   a := 10;
   b := 20;
END;

BEGIN a := 10; b := 20; END;
```

Conditions et séquences

- `if <condition> then <bloc> elsif <condition> else <bloc> end if;`
- ```
  case <expression>
  when <result a> then <bloc>
  when <result b> then <bloc>
  ...
  else <bloc>
  end case;
  ```
- ```
  case when <condition a> then <bloc>
  when <condition b> then <bloc>
  ...
  else <bloc>
  end case;
  ```
- `a := case ... end;`
- `goto <label>`
- `null;`

Itérations

- `loop exit when <condition>; <bloc> end loop;`
- `loop <bloc condition contenant exit; > end loop;`
- `loop <bloc> exit when <condition>; end loop;`
- `while <condition> loop <bloc> end loop;`
- `for <variable de boucle> in <valeur début>..<valeur fin> loop <bloc> end loop;`
- `for <pointeur de curseur> in <requête selest> loop <bloc> end loop;`
- `for <pointeur de curseur> in <déclaration de curseur> loop <bloc> end loop;`

Types PL/SQL

Principaux types PL/SQL

- PLS_INTEGER or BINARY_INTEGER: -2^{32} to $2^{32} - 1$
- BINARY_FLOAT: Nombre décimal en simple précision
- BINARY_DOUBLE: Nombre décimal en double précision
- NUMBER: Nombre à décimales fixes ou variables compris entre $1x10^{-130}$ to $1.0x10^{126}$
- CHAR, VARCHAR2: Identique au type SQL avec 32767 caractères maximum
- RAW: Identique au type SQL avec 32767 octets maximum
- NCHAR, NVARCHAR2: Identique au type SQL avec 32767 caractères maximum
- LONG and LONG RAW: Identique au type SQL avec 32760 octets maximum
- ROWID and UROWID: Identique au SQL
- BOOLEAN: True ou False ou Null
- DATE, TIMESTAMP, TIMESTAMP WITH TIMEZONE, INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECONDS: Identique au SQL
- BFILE: Identique au SQL
- BLOB, CLOB, NCLOB: Identique au SQL. Longueur maximum de 8 To à 128 To

Fonctions PL/SQL

Fonctions PL/SQL

- Toutes les fonctions du SQL à l'exception de :
  - Agrégats
  - Analytiques
  - Collection
  - Data mining
  - Encodage et décodage
  - Modèle
  - Référence à objets
  - XML
  - Certaines fonctions de conversion
  - Certaines fonctions particulières (voir manuel)
- Deux fonctions particulières existent :
  - SQLCODE: Retourne le dernier code erreur SQL
  - SQLERRM: Retourne le dernier message erreur SQL

Gestion des erreurs PL/SQL

Gestion des erreurs PL/SQL

- Keyword: EXCEPTION → Début du bloc de gestion
- Keyword: WHEN <valeur erreur> THEN → Branchement vers la gestion d'erreur
- OTHERS: Si aucune autre condition n'a été exécutée, exécution par défaut
- RAISE <nom d'exception>: Retourne l'exception à la procédure parente
- RAISE <nom package.nom exception>: Retourne l'exception contenue dans la package à la procédure parente
- RAISE_APPLICATION_ERROR(nombre, message, conservation de la pile d'erreurs): Définit une erreur utilisateur avec un nombre compris entre -20005 et -20999 ainsi qu'un message. La conservation de la pile d'erreur est possible

Exceptions PL/SQL Nommées

Nom de l'exception/Erreur Oracle/SQLCODE	Description
CURSOR_ALREADY_OPEN/ORA-06511/-6511	Tentative d'ouverture d'un curseur déjà ouvert
DUP_VAL_ON_INDEX/ORA-00001/-1	Violation d'une clé unique d'un index unique
INVALID_CURSOR/ORA-01001/-1001	Le curseur n'existe pas
INVALID_NUMBER/ORA-01722/-1722	La chaîne de caractères n'est pas un nombre
LOGIN_DENIED/ORA-01017/-1017	Utilisateur/mot de passe incorrects
NO_DATA_FOUND/ORA-01403/+100	Le dernier enregistrement du curseur est passé
NOT_LOGGED_ON/ORA-01012/-1012	Vous devez vous connecter
PROGRAM_ERROR/ORA-06501/-6501	Contacter le Support Oracle !!
STORAGE_ERROR/ORA-06500/-6500	Mémoire insuffisante
TIMEOUT_ON_RESOURCE/ORA-00051/-51	Délai d'attente dépassé sur une ressource interne
TOO_MANY_ROWS/ORA-01422/-1422	SELECT INTO ne doit retourner qu'une ligne
TRANSACTION_BACKED_OUT/ORA-00061/-61	Problème avec une transaction distante
VALUE_ERROR/ORA-06502/-6502	Problème durant une conversion, une troncature…
ZERO_DIVIDE/ORA-01476/-1476	Interdit !

ANNEXE

Oracle RDBMS
Architecture détaillée

Pourquoi connaître le fonctionnement du noyau ?

- Connaître l'architecture du noyau facilite le développement sur le SGBDR
- Connaître le fonctionnement peut permettre d'en tirer profit
- Connaître les fonctionnalités afin d'éviter de re-développer ce qui existe déjà
- Optimiser les développements en fonction du noyau et ne pas « faire faire » au serveur d'applications ce que le serveur de bases de données sait très bien faire
- Ne pas donner de tâches au noyau de bases de données qui doivent revenir au serveur d'applications

Eléments de vocabulaire propre à Oracle - 1

- RDBMS : Relational DataBase Management System (SGBDR en français)
- Instance : … ou instance de base de données. Contexte mémoire et process d'une base de données
- Base de données : Couche physique englobant l'ensemble des fichiers
- SID : System IDentifier. Identifiant unique d'une instance de base de données Oracle sur un serveur spécifique. Cet identifiant a, entre autres, pour rôle la réservation de la mémoire partagée au niveau du système

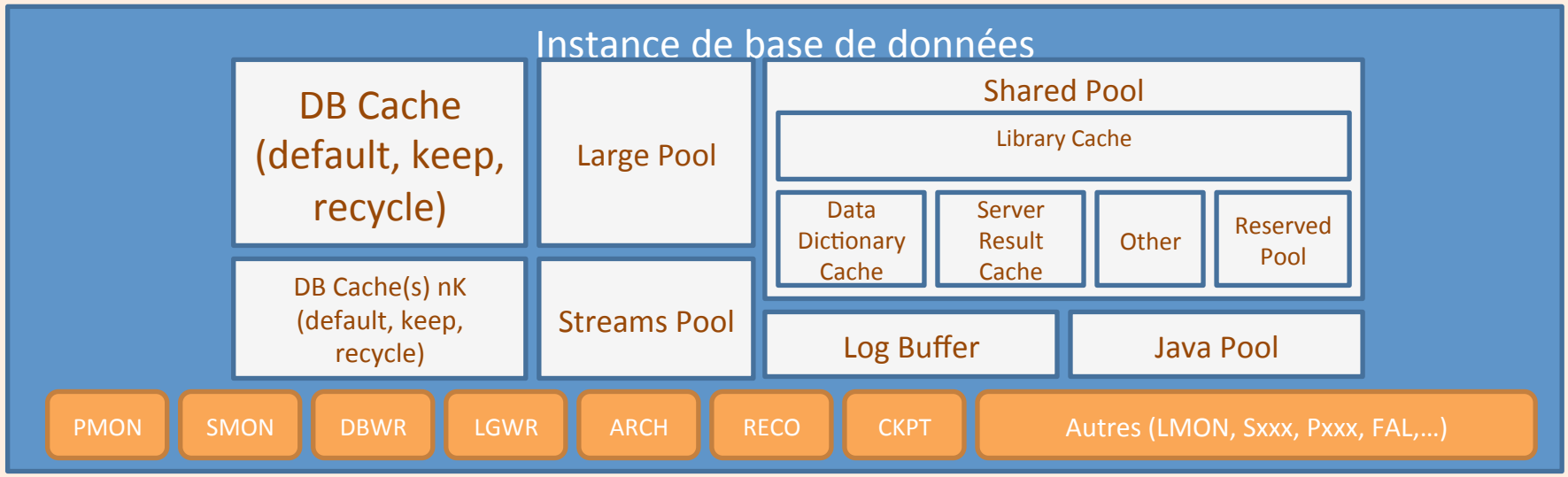
Eléments de vocabulaire propre à Oracle - 2

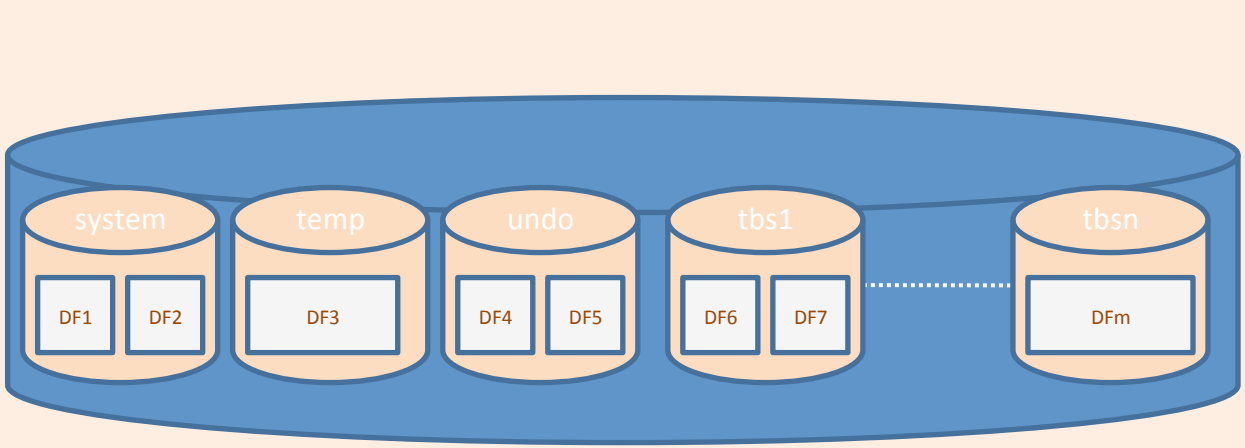
- Tablespace : Ensemble de fichiers regroupés dans un espace logique, dans lequel les objets sont stockés
- Datafile : Sous-partie d'un tablespace
- Schéma : Contenant logique des objets d'une application
- Utilisateur : Se confond avec le schéma. Il représente la partie « droits » d'accès

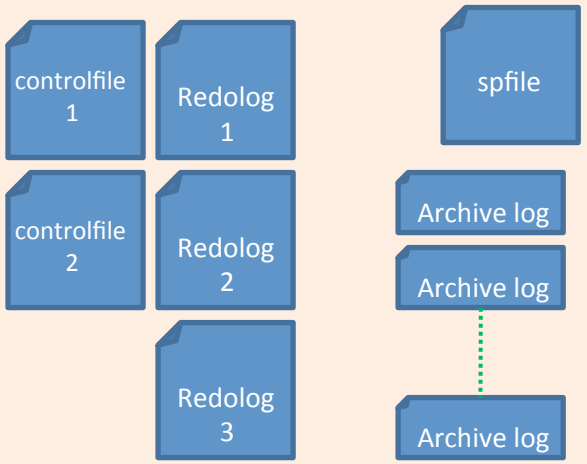
Eléments de vocabulaire propre à Oracle - 3

- Net Services : Protocole réseau propriétaire de communication entre les différents acteurs d'une architecture Oracle RDBMS
- Listener : Process dont le rôle est de permettre la mise en relation entre une demande d'un client et une instance de base de données
- Dataguard : Environnement pourvoyant aux PRAs (Plan de Reprise d'Activité)
- RAC : Real Application Cluster. Ensemble d'instances de base de données liées entre elles et partageant STRICTEMENT la même base de données
- Incarnation : Axe temps en cours pour une base de données. En cas de restauration/recover incomplète, un nouvel axe temps est créé pour cette base de données et de fait l'incarnation de celle-ci change

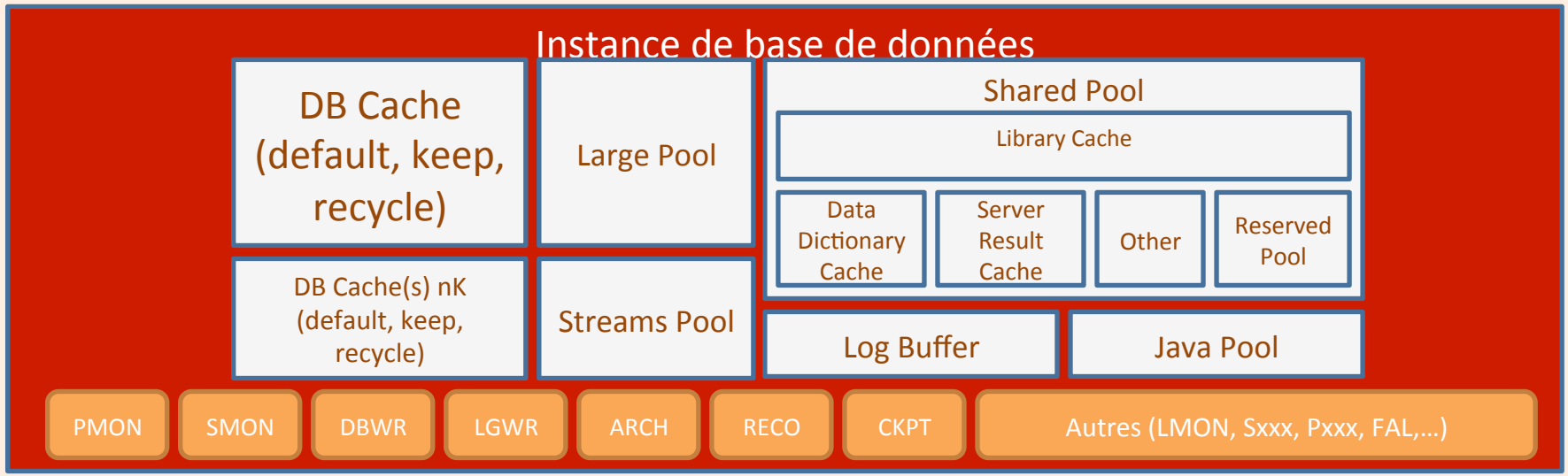
Architecture détaillée

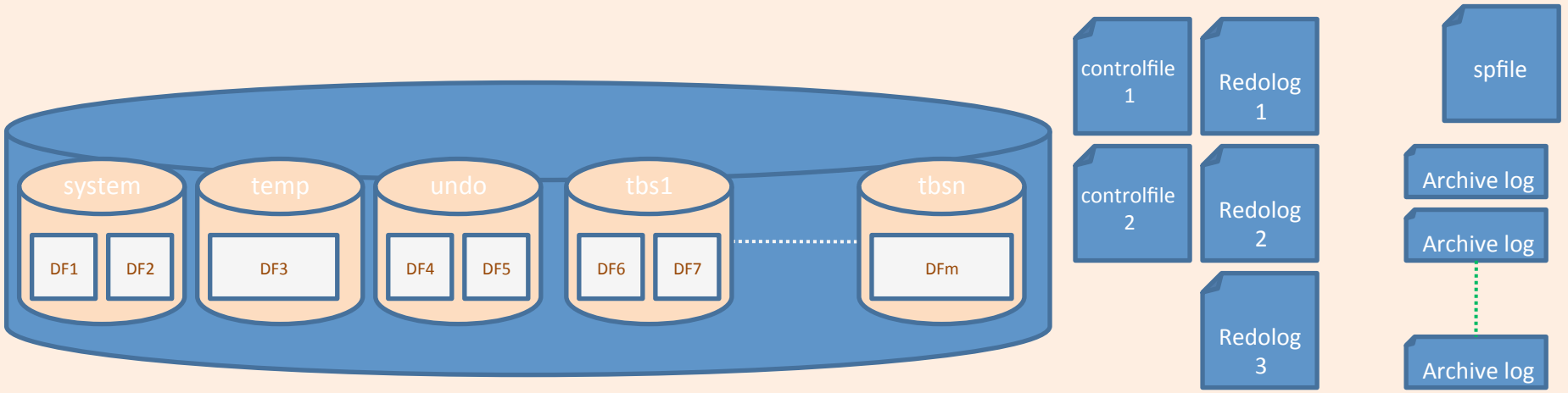




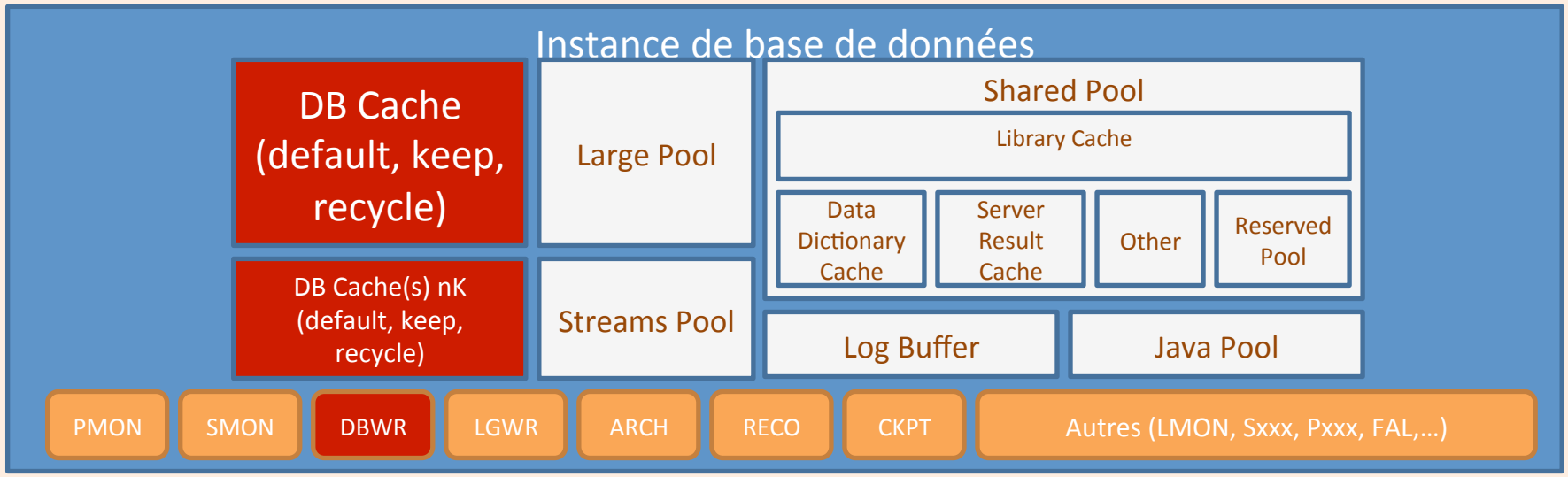


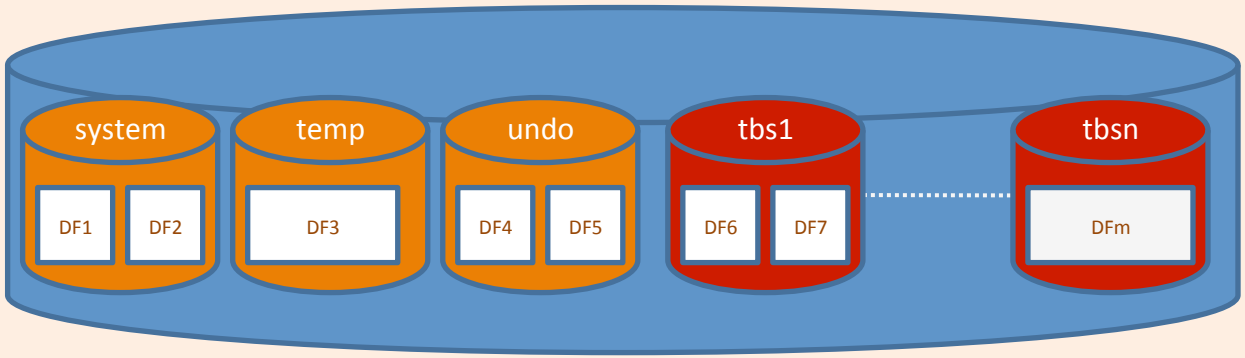
L'instance de base de données

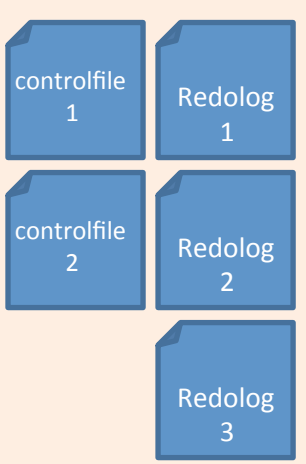


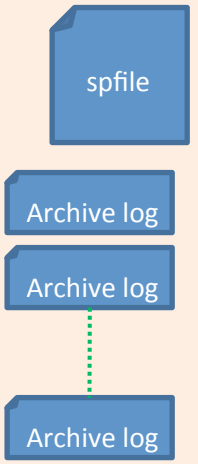


Le Cache de Données

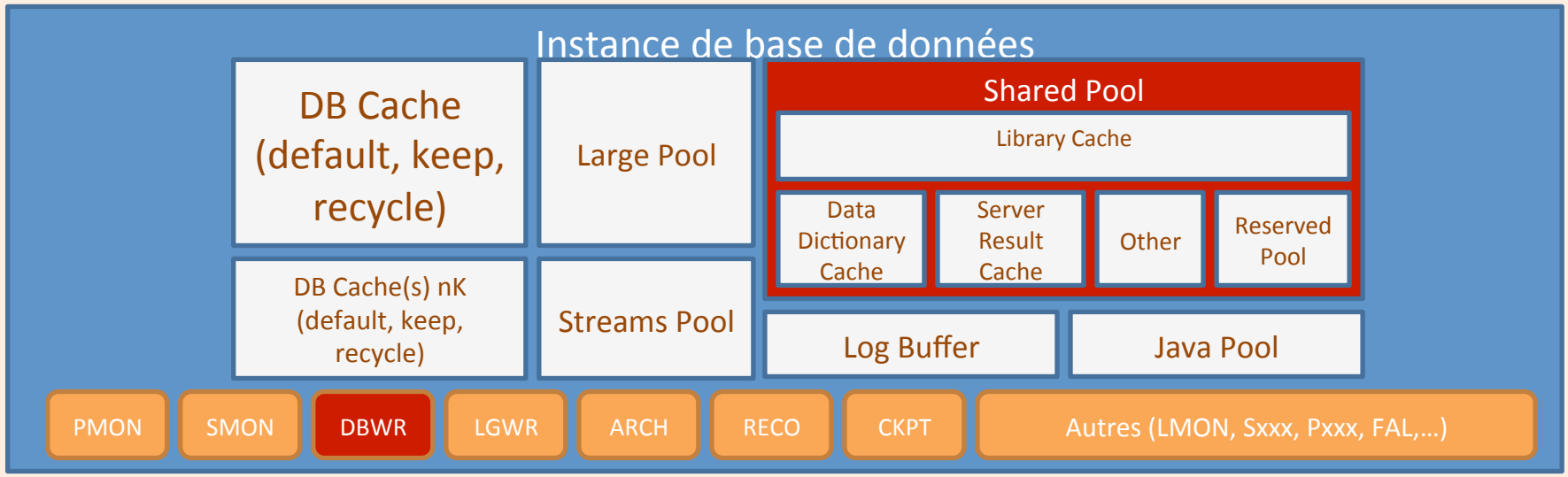


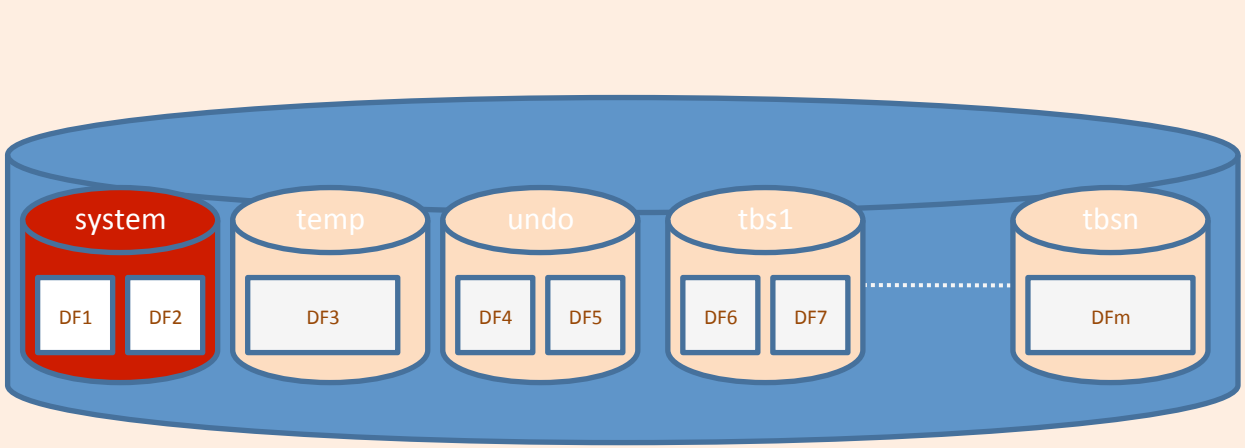


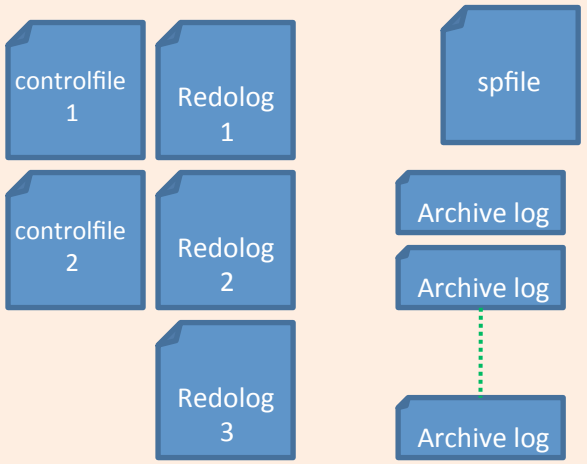




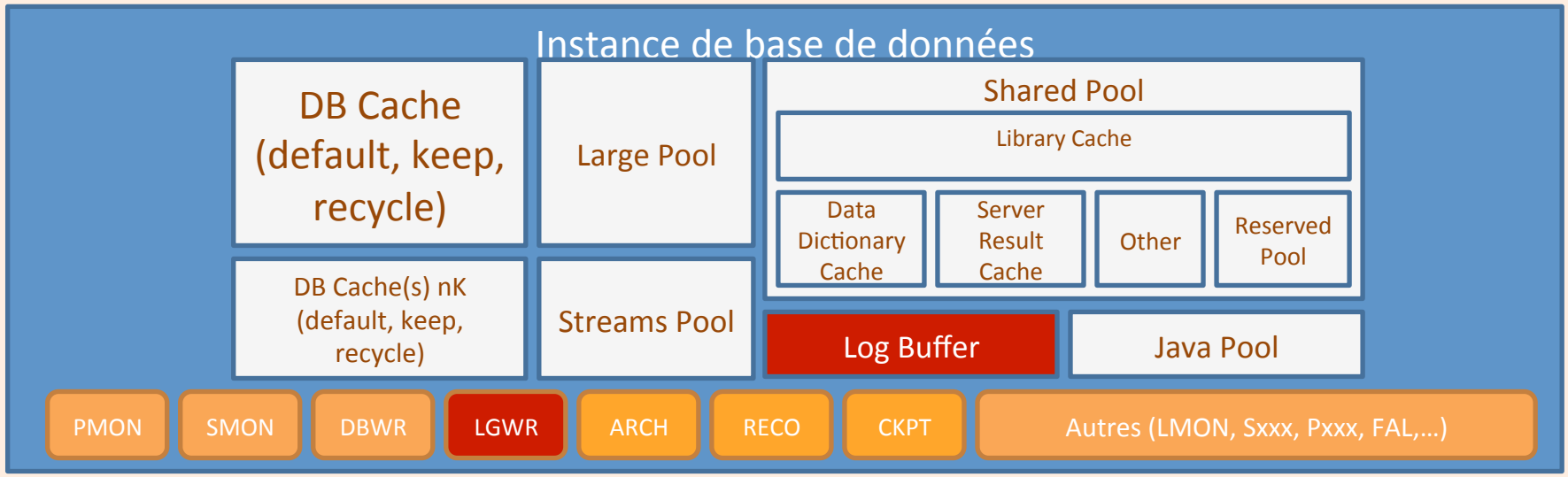
La zone de gestion de l'instance

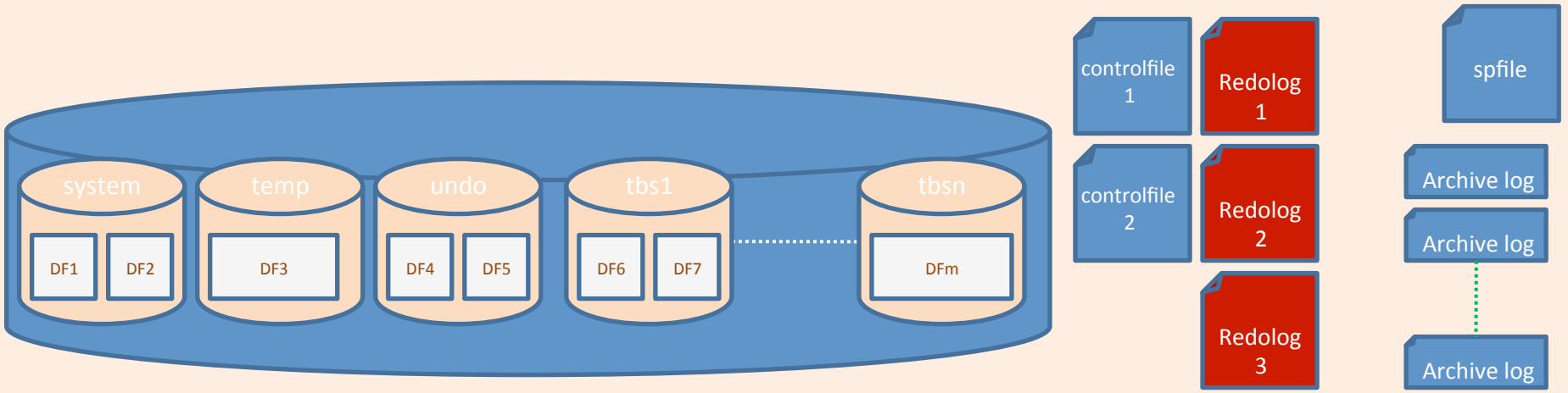




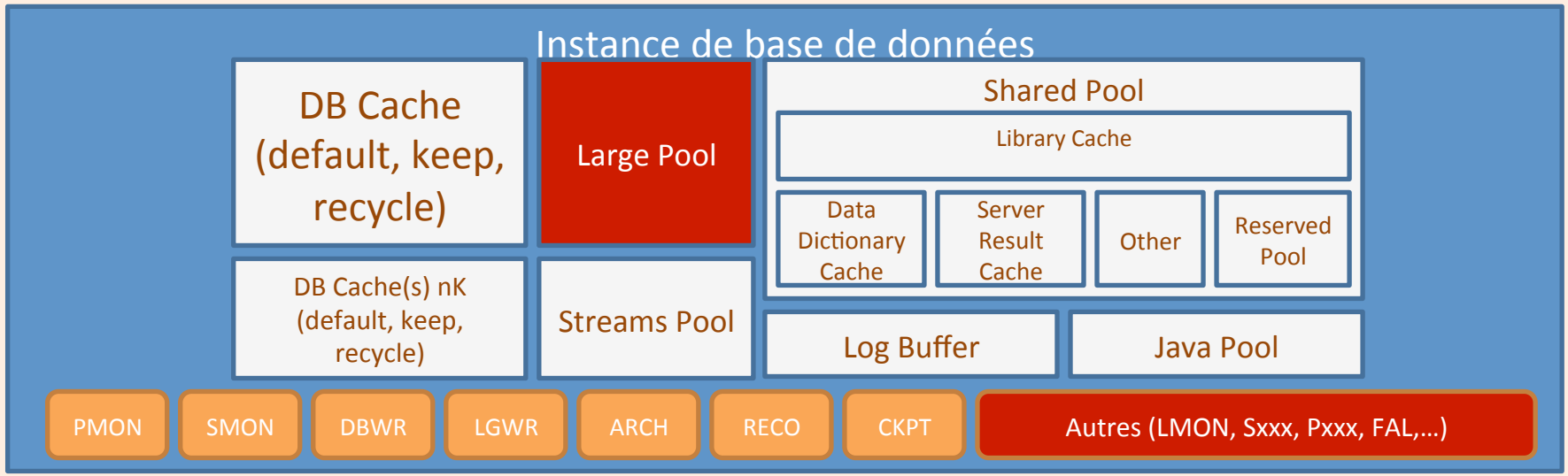


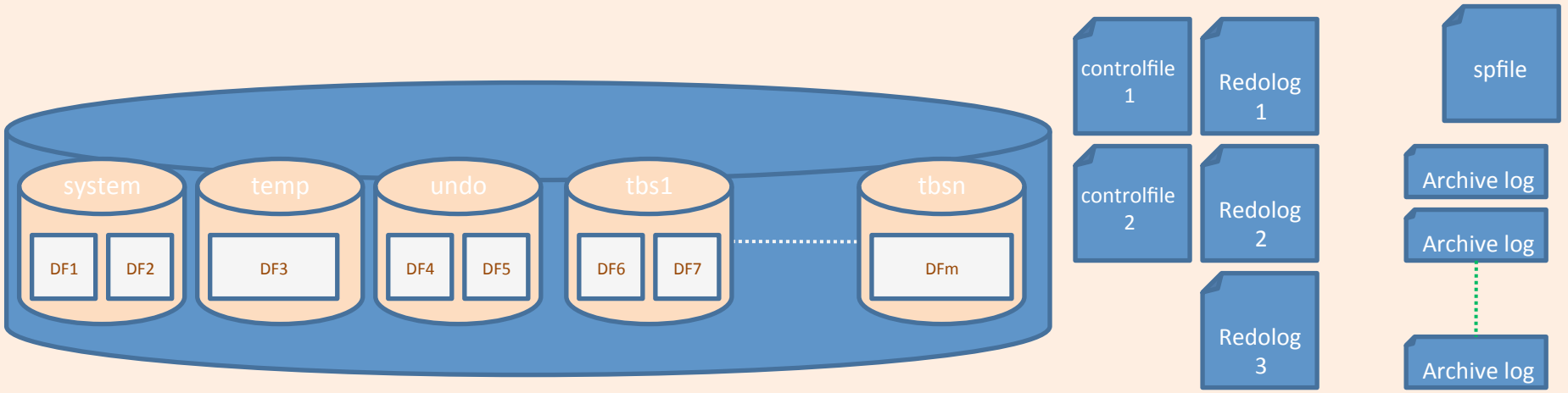
Le redolog buffer



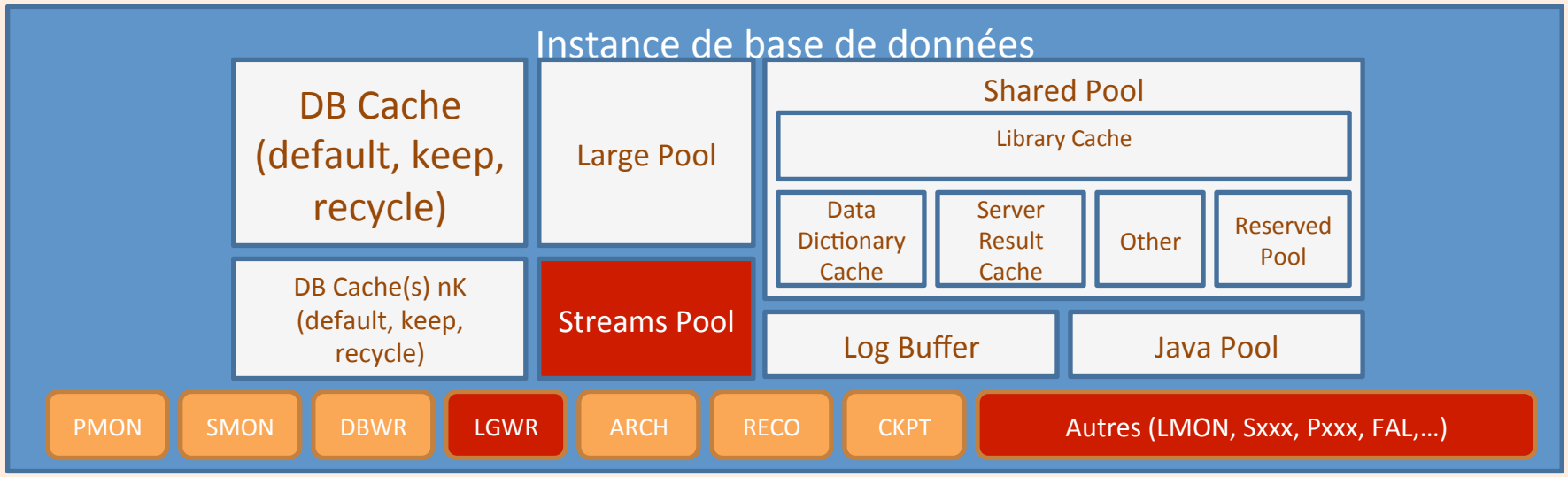


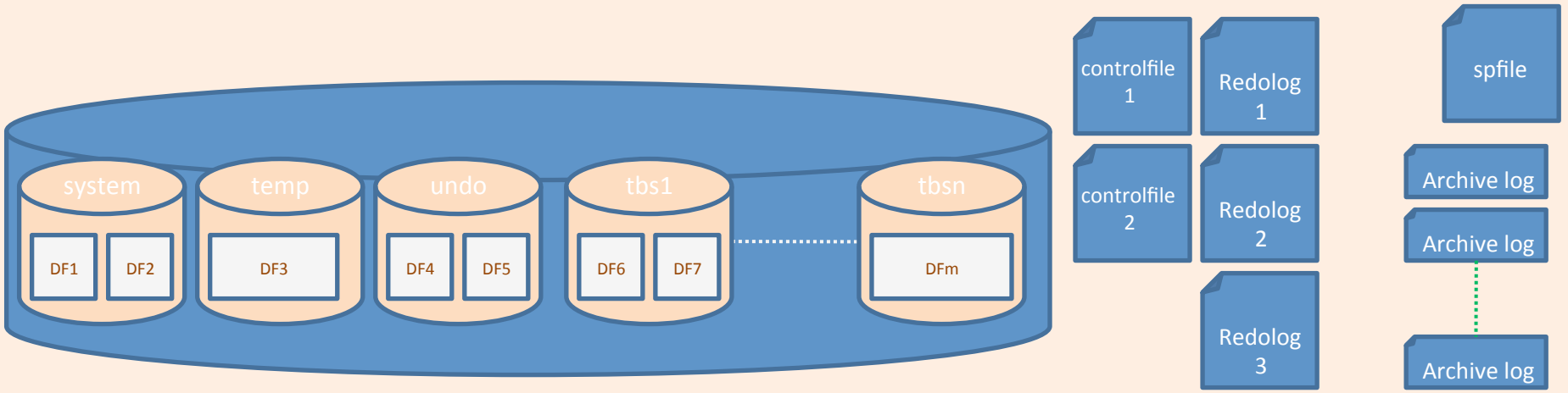
La Large Pool



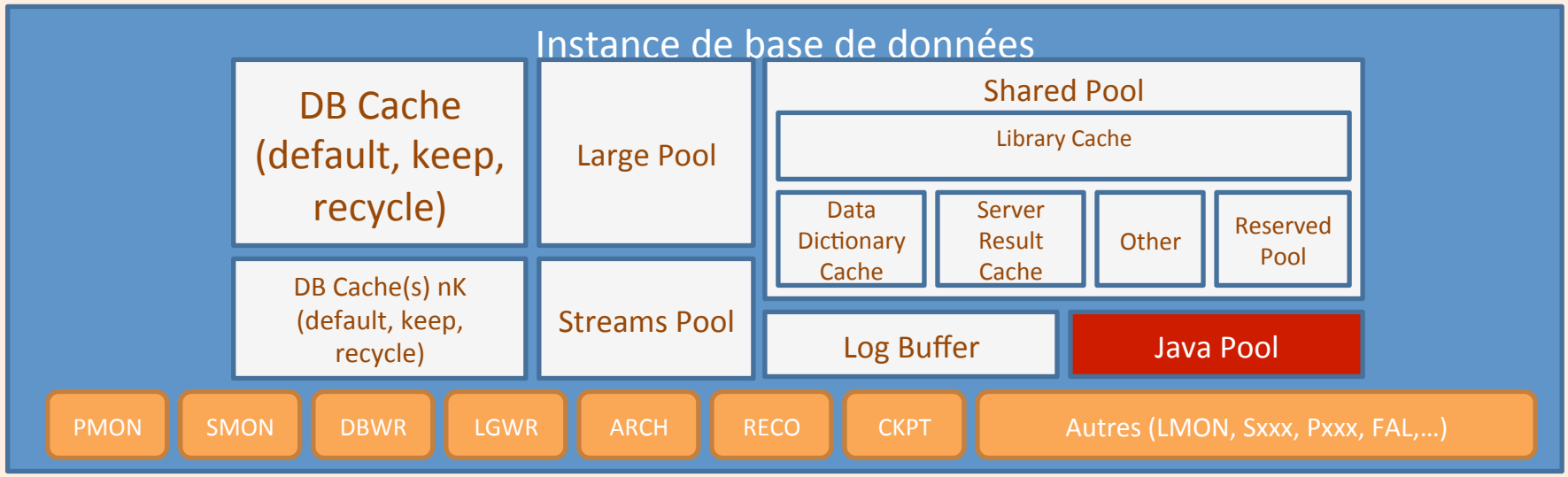


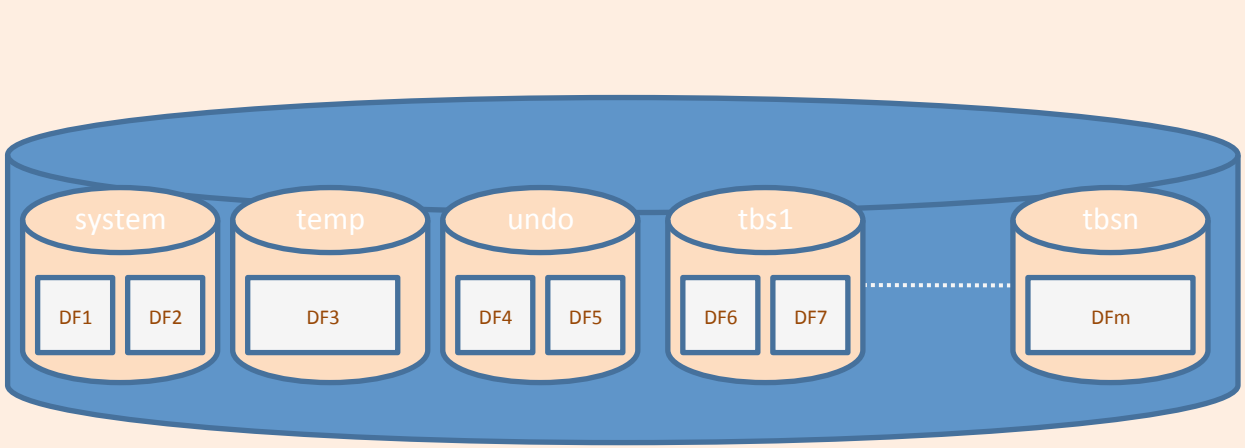
La Streams Pool

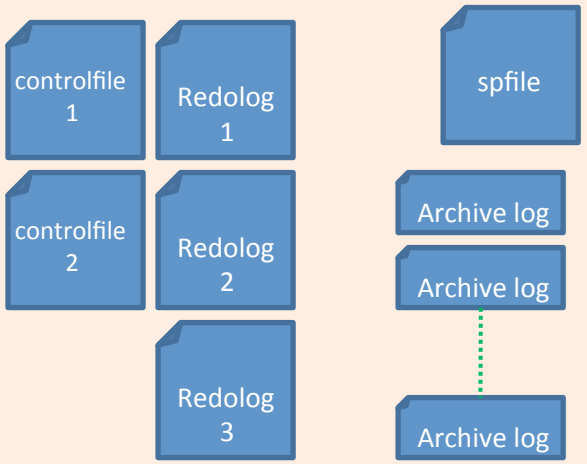




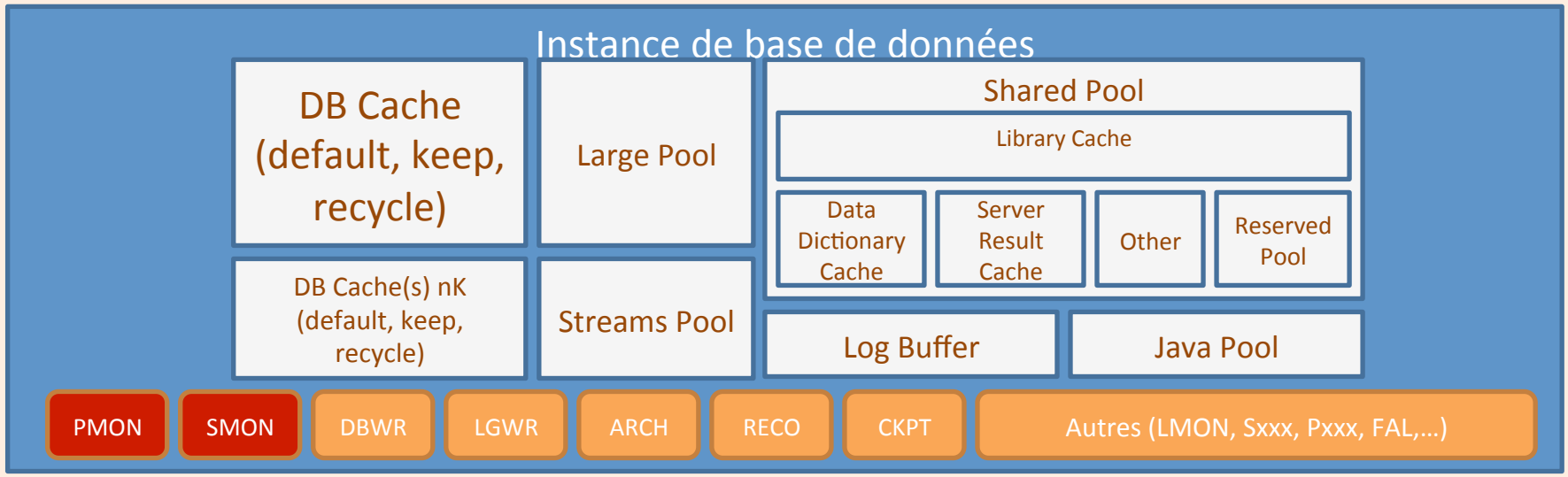
La Java Pool

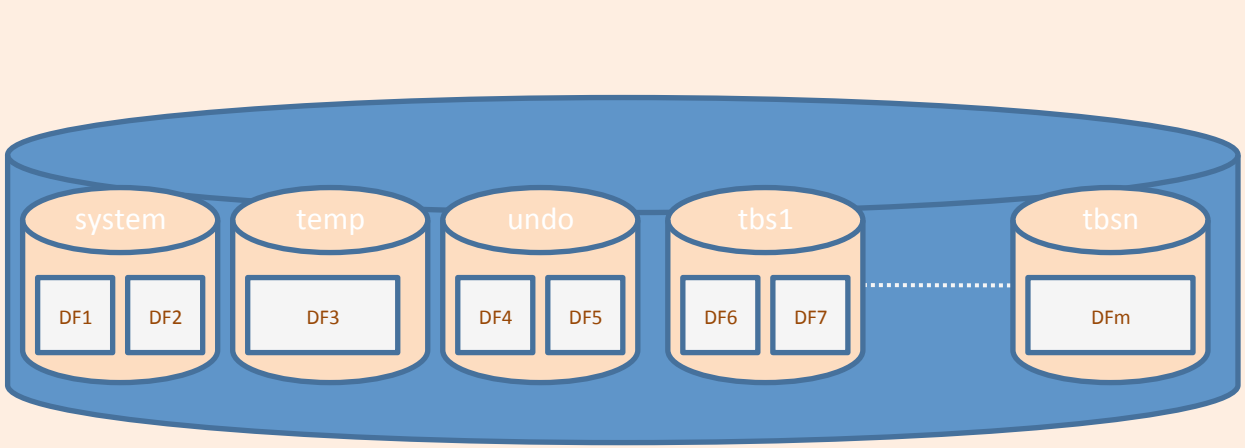


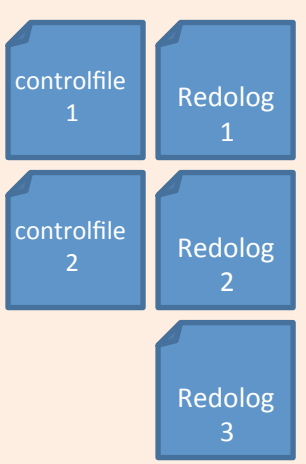


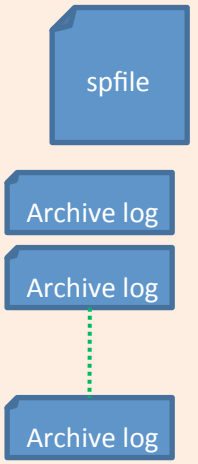


Les Process de monitoring

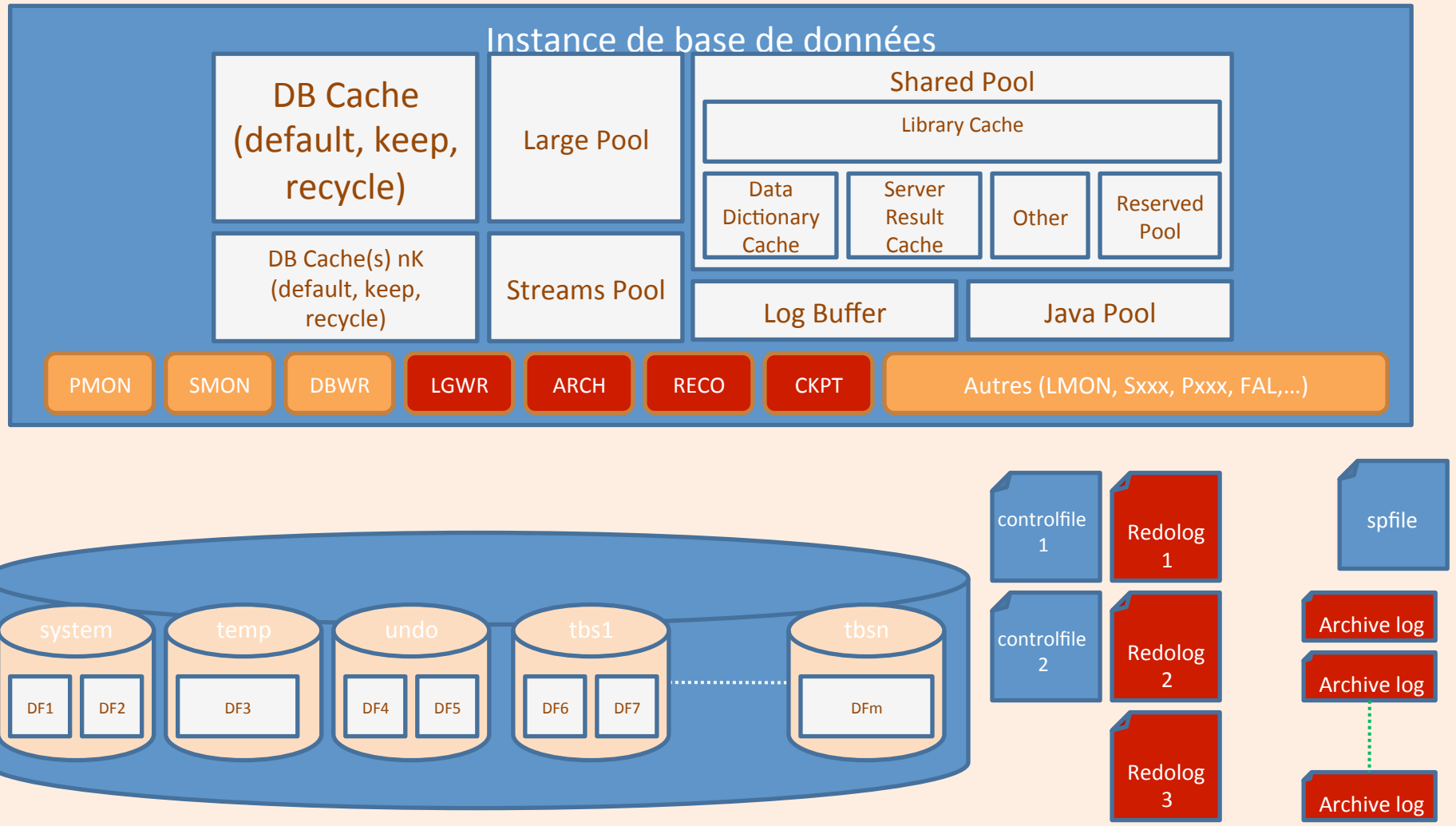








Les Process de gestion liés au recover



Modes de fonctionnement du noyau Oracle

NOARCHIVELOG

- C'est le mode par défaut de fonctionnement du noyau
- Dans ce mode, aucun fichier de redolog ne fait l'objet de sauvegarde lors de son « switch »
- La base de données ne peut être sauvegardée qu'à froid
- Pas de possibilité de reprise jusqu'à l'instant du crash en cas de recours à une sauvegarde

ARCHIVELOG

- Ce mode est à paramétrer spécifiquement
- Chaque redolog, dès qu'il est plein, fait l'objet d'une sauvegarde « archive »
- Les sauvegardes « à chaud » deviennent possibles
- Les sauvegardes « à chaud » s'appuient sur la sauvegarde des fichiers dans un état non consistant auquel est ajoutée une sauvegarde spécifique des archives, dont principalement l'archive de fin de sauvegarde que l'on génère tout spécialement pour avoir un checkpoint, donc un point de reprise cohérent

NOLOGGING/LOGGING

- Le mode « NOLOGGING » spécifie, pour les objets pour lesquels il est défini, que les transactions effectuées ne génèrent pas d'information de redologs
- Ces objets, en cas de besoin de reprise, ne sont pas reconstituables, ne disposant pas d'informations pouvant les rendre à nouveau consistants
- Ce mode est employé pour accélérer les mises à jour lors de lourds traitements de chargement par exemple
- Le mode « LOGGING » doit être la règle si l'on veut être capables de reconstituer la base de données en cas d'incident d'exploitation
- Le mode « NOLOGGING » ne doit être employé que dans des cas bien particuliers

FORCE LOGGING

- Lorsque ce mode est appliqué à un objet (BDD → Tables), il prend le pas sur les instructions de « NOLOGGING » qui peuvent être demandées par les applications ou les utilisateurs
- Ce mode de fonctionnement peut être décidé au niveau de l'exploitation pour s'assurer la capacité à reconstruire la base de données à tout moment, quand bien même les décisions fonctionnelles préconisent de travailler sans enregistrer les opérations de redologs

SUPPLEMENTAL LOG

- Ce mode implique un enregistrement d'informations plus important dans les redologs
- Il est employé dans les problématiques de réplication afin d'assurer une consistance entre les différents maître et esclaves de l'architecture de réplication
- Il permet un traitement beaucoup plus avancé de l'information pour les réplicats
- Il donne des possibilités de reprise en cas de crash bien plus élevées pour les réplicats
- Il est plus coûteux en stockage d'archives et en I/Os
- A n'employer que dans le cas de réplication !!

Fondamentaux du noyau

Niveaux de sérialisation transactionnelle de l'instance

Niveau d'isolation	Dirty Read	Lecture non « répétable »	Lecture fantôme
Read uncommited	Possible	Possible	Possible
Read commited	Impossible	Possible	Possible
Repeatable read	Impossible	Impossible	Possible
Serializable	Impossible	Impossible	Impossible

Le SCN

- Le SCN : System Change Number
- Il progresse tout au long de la vie de l'instance
- Il permet de sérialiser facilement les transactions
- Il donne la cohérence des objets entre eux
- Il permet la reconstitution intégrale d'une base de données à partir de ses transactions
- Il est la clé de voûte de plusieurs mécanismes :
  - Réplication
  - Dataguard
  - Message Queuing
  - Restaure/Recover

Les transactions

- Une transaction est un ensemble fini d'opérations ayant un début et une fin
- La fin d'une transaction peut être sa validation
- … ou son annulation
- Le moteur de base de données doit avoir un repère pour identifier les transactions
- Une transaction générant des informations temporaires (tant que non validées), doit disposer d'un espace permettant l'annulation

Les transactions : Redologs

- Ils sont la colonne vertébrale de l'instance
- Toutes les transactions sont écrites même si elles ne sont pas terminées
- Ils permettent le « fast-commit »
- Ils sont écrits de manière à pouvoir permettre une sérialisation des transactions afin de pouvoir facilement rejouer celles-ci en cas de besoin
- Leur numéro interne s'incrémente à chaque nouvelle ouverture (rotation de redologs)
- Ils sont en nombre fini et sont employés de manière circulaire
- Ils ne peuvent être réemployés si les opérations qu'ils contiennent n'ont pas été écrites en base de données avant leur réouverture (checkpoint)
- Ils sont écrits en continu
- Le SCN prend une place centrale dans leur écriture et sérialisation

Le Checkpoint

- … est le mécanisme qui « ordonne » au noyau d'écrire tous les blocs « commités » ou non dans les fichiers de la base de données
- Il assure la durabilité de la base de données
- Il est en relation directe avec les redologs et surtout leur rotation
- Sans checkpoint, pas de cohérence possible
- Sans checkpoint, l'instance se fige si elle n'a plus de fichier de redologs « libres »
- Le process responsable du checkpoint est CKPT
- Un checkpoint intervient dans les conditions suivantes :
  - Ordre explicite donné au noyau
  - Arrivée en bout de redolog
  - Demande de rotation de redolog
  - Archivage de redolog
  - Demande de fermeture de la base de données (arrêt d'instance)

Les transactions : Undos

- Toute transaction génère des modifications
- Tant qu'une transaction n'est pas terminée, les modifications peuvent être annulées
- Les Undos stockent les valeurs de bloc « avant » modification
- En cas de Rollback, ce sont ces valeurs qui sont rétablies
- Les Undos peuvent contenir plusieurs valeurs différentes pour un même enregistrement, si nécessaire. On appelle ce phénomène le « clonage » de bloc

La reprise sur crash : Redologs

- La raison d'être principale des redologs est la reprise sur crash
- Ils pourvoient à la remise en état de la base de données en cas de crash n'ayant endommagé aucun fichier de la base
- Ils permettent le redémarrage de l'instance en cas d'arrêt brutal lié à un « abort »
- Ce mécanisme fonctionne quel que soit le mode de fonctionnement de la base : ARCHIVELOG, NOARCHIVELOG
- Le Checkpoint a un rôle essentiel pour que ce mécanisme puisse fonctionner

Le recover : Archivelogs

- Ce sont les pendant des redologs
- Chaque redolog, une fois qu'il est rempli, est recopié sous forme de redolog
- Il n'y a pas de limite au nombre d'archivelogs stockés
- Le filesystem où sont stockés les archivelogs doit être surveillé avec attention afin de ne pas être saturé
- Le moyen le plus sûr de récupérer de l'espace dans le filesystem contenant les redologs est la sauvegarde de ceux-ci sur bande, et leur purge une fois déposés sur bande
- S'ils sont stockés depuis la date de création de la base de données avec une première sauvegarde complète de la base au moment de sa création, il permettent de revenir à n'importe quel point dans le temps de la base de données

Le recover : Sauvegarde « à chaud »

- Une sauvegarde à chaud consiste à sauvegarder les fichiers de la base de données dans leur état inconsistants
- Elle comprend au minimum les fichiers d'archives des redologs avant début de sauvegarde, ainsi que l'archive générée en fin de sauvegarde
- L'ensemble de ces fichiers, y compris les redologs dits « courants », permettent une remise en état d'une base de données au travers d'une sauvegarde, jusqu'au dernier commit enregistré séquentiellement dans les redologs avant le crash

La gestion du DB Cache

- Le DB Cache est constitué sous la forme d'une file de blocs unique classés de MRU (Most Recent Used) à LRU (Least Recent Used)
- Les blocs les plus anciens peuvent être recyclés pour permettre à des blocs demandés par les transactions de monter en mémoire
- Le process DBWR est à l'origine de l'écriture des blocs, parfois sur ordre de CKPT
- Il peut être géré par plusieurs process DBWR (DBW0 à DBWn)
- Il est par essence, volatile, et nécessite d'être écrit régulièrement dans les fichiers pour les blocs modifiés
- Il peut contenir (depuis la 11gR2) les résultats des requêtes s'il est explicitement demandé de les y stocker
- Il contient des blocs de même taille
- Il ne peut contenir un mix entre blocs de tailles différentes
- Dans un tel cas, il existe autant de caches que de tailles de blocs différentes

Automatic Storage Management

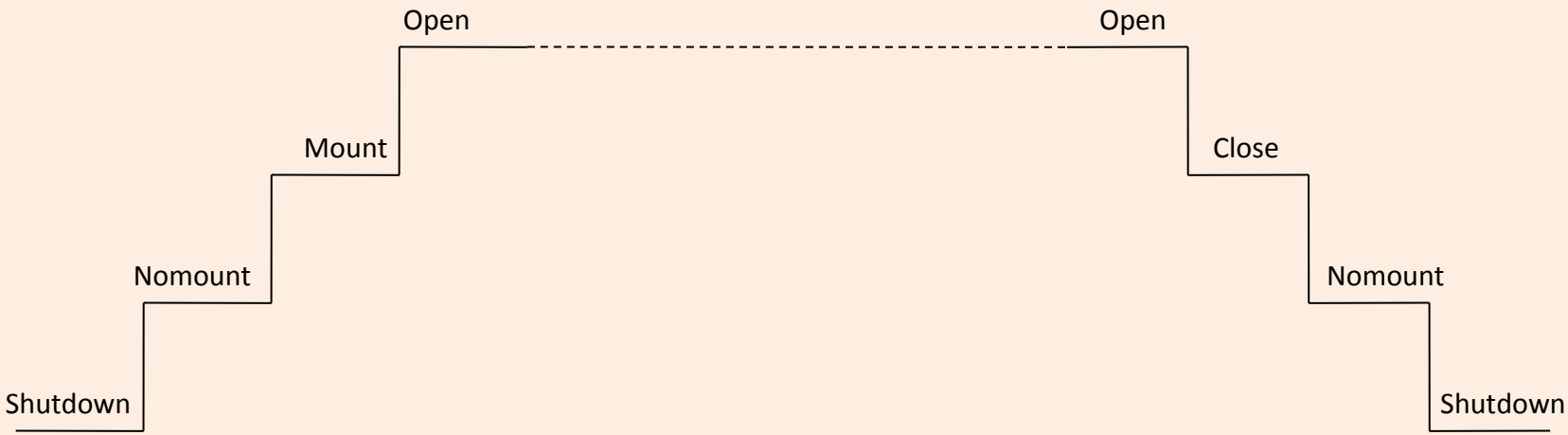
- Il s'agit d'un mode de stockage développé par Oracle, dédié aux bases de données Oracle
- Il nécessite des devices vierges présentés par le stockage
- Il permet d'ajouter/retirer dynamiquement des LUNs sans arrêt de production
- Il permet de répartir aisément le stockage sur plusieurs SAN
- Il optimise les traitements avec le stockage (longueur des I/Os…)
- Il est indispensable pour les bases de données en cluster

Les fichiers de la base de données

- Durant toute la vie de l'instance, à moins qu'ils soient en lecture seule, les fichiers de la base de données sont constamment inconsistants
- Les écritures régulières des blocs modifiés en mémoire vers les fichiers, permettent de maintenir ceux-ci dans un état proche de la cohérence

Démarrage/arrêt d'une instance Oracle

Principe de démarrage/arrêt d'une instance de base de données



Types d'arrêts possibles et leur action sur l'instance et la BDD

Opérations de BDD	Normal	Transactional	Immediate	Abort
Permettre les connexions de nouveaux utilisateurs	Non	Non	Non	Non
Attente jusqu'à ce que les sessions courantes terminent	Oui	Non	Non	Non
Attente jusqu'à ce que les transactions courantes terminent	Oui	Oui	Non	Non
Faire un checkpoint et fermer les fichiers ouverts	Oui	Oui	Oui	Non