SGBD Avancés

Nicolas Anciaux et Philippe Pucheral

Objectifs du cours

- Comprendre le fonctionnement interne d'un SGBD
  - Suffisamment pour comprendre une doc. technique
  - Pour déduire le comportement au niveau des performances
  - Pour être capable de régler un SGBD
- Illustrations du fonctionnement interne de SGBD existants
  - SGBD commercial → le noyau d'*Oracle*
  - SGBD open source → le noyau de *MySQL*
- Appréhender certaines thématiques très actuelles
  - Engouement actuel pour la sécurité dans les SGBD
  - Evolution vers la mémoire Flash
  - …

Menu …

- Survol des SGBD – architecture générale d'*Oracle* : 1 cours
- Fonctionnement interne des SGBD : 5 cours
  - Transactions – isolation, atomicité, durabilité, répartition
  - Modèle de stockage et d'indexation – structures de données
  - Modèle d'exécution – algorithmes, allocation mémoire
  - Optimisation – coûts des opérations, choix du meilleur plan
  - Sécurité – droits d'accès, chiffrement, tiers de confiance

    (NB: pour chaque item, description de l'implantation dans *Oracle*)
- Fonctionnement interne de *MySQL* : 1 cours ½
  - exposés sur le noyau de *MySQL* (6 sujets)
    → Lecture de doc. technique [en anglais], [analyse de code C,C++]
- Examen : ½ cours
  - 1h30, tous documents autorisés

Sujets d'exposés : MySQL/InnoDB internals

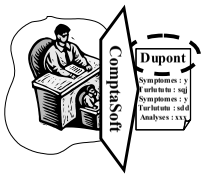
- Sujet 1 – MySQL/InnoDB internals – The Query Processor
- Sujet 2 – MySQL/InnoDB internals – Transactions & Concurrency
- Sujet 3 – MySQL/InnoDB internals – Logging & Recovery
- Sujet 4 – MySQL/InnoDB internals – Memory Management
- Sujet 5 – MySQL/InnoDB internals – Storage & Indexes
- Sujet 6 – MySQL/InnoDB internals – Physical Deletion
- http://www-smis.inria.fr/~anciaux/SGBD/ISTY%202007-2008/Exposes/
- Planning
  - Dans les deux dernières séances
  - Présentation orale par groupe de x

Rappels sur les SGBD

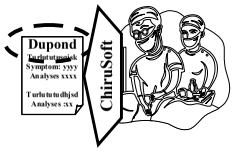
Plan du cours

- SGBD vs. système de gestion de fichiers
- Revue des apports majeurs d'un SGBD
  - Description canonique des données
  - Indépendance logique/physique
  - Langage de manipulation
  - Gestion des vues
  - Optimisation des questions
  - Gestion de la cohérence
  - Gestion des pannes
  - Concurrence d'accès
  - Gestion de la confidentialité
  - Standards
- Architecture globale d'un SGBD
  - Introduction des cours suivants
- Architecture d'Oracle

Un système d'information sans SGBD (1)

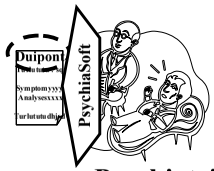


Comptabilité



Chirurgie

Consultations



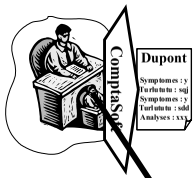
Psychiatrie

Plusieurs applications
- ➔ plusieurs formats et langages
- ➔ difficulté de maintenance
- ➔ difficulté d'interopérabilité

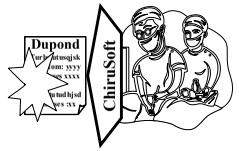
Redondance des données
- ➔ incohérence

Pas de facilité d'interrogation
- ➔ toute question doit être prévue et programmée
- ➔ coût de développement élevé
- ➔ redondance de code
- ➔ difficulté d'évolution

Un système d'information sans SGBD (2)



Comptabilité



Chirurgie

Consultations

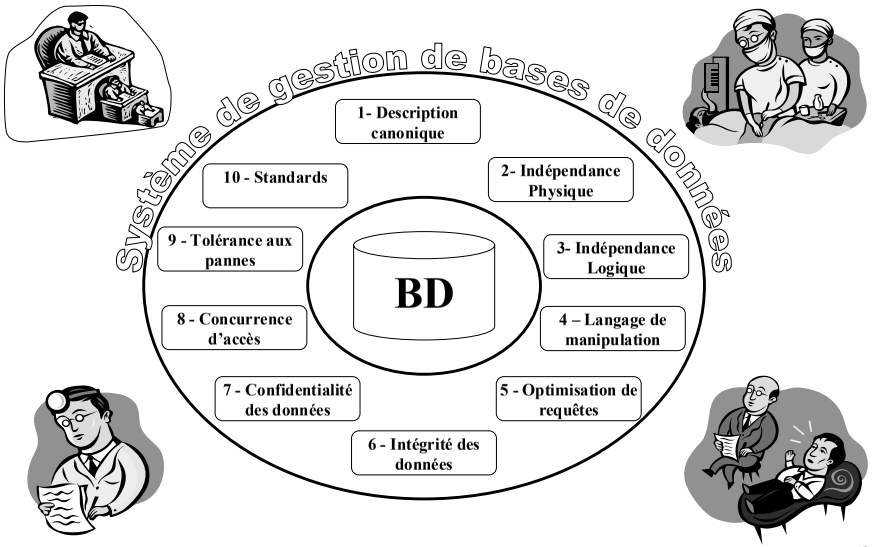
Psychiatrie

Gestion simpliste des droits
- ➔violation de la confidentialité

Gestion simpliste du parallélisme
- ➔blocage du système ou bien incohérence en cas d 'accès simultanés

Pas de tolérance aux fautes
- ➔incohérence en cas de panne
- ➔perte de données

L'approche Bases de données



I - Description canonique des données

- Description cohérente, unique et centralisée des données manipulées par l'ensemble des applications constituant le système d'information.

— Perception globale du système d'information

=> augmentation du niveau d'informatisation

=> nouveaux traitements (aide à la décision, analyse de données, …)

— Factorisation de la description des données et de leur comportement (contraintes d'intégrité …)

— Elimination de la redondance

=> redondance = source d'incohérence

=> redondance système reste nécessaire pour : fiabilité, performance de consultation, disponibilité en environnement réparti ou mobile

Exemple : modélisation Relationnelle

Docteurs

Id-D	Nom	Prénom
1	Dupont	Pierre
2	Durand	Paul
3	Masse	Jean
….	……..	……

Visites

Id-D	Id-P	Id-V	Date	Prix
1	2	1	15 juin	250
1	1	2	12 août	180
2	2	3	13 juillet	350
2	3	4	1 mars	250

Prescriptions

Id-V	Ligne	Id-M	Posologie
1	1	12	1 par jour
1	2	5	10 gouttes
2	1	8	2 par jour
2	2	12	1 par jour
2	3	3	2 gouttes
….	….	….	……

Patients

Id-P	Nom	Prénom	Ville
1	Lebeau	Jacques	Paris
2	Troger	Zoe	Evry
3	Doe	John	Paris
4	Perry	Paule	Valenton
….	…….	…….	…….

Médicaments

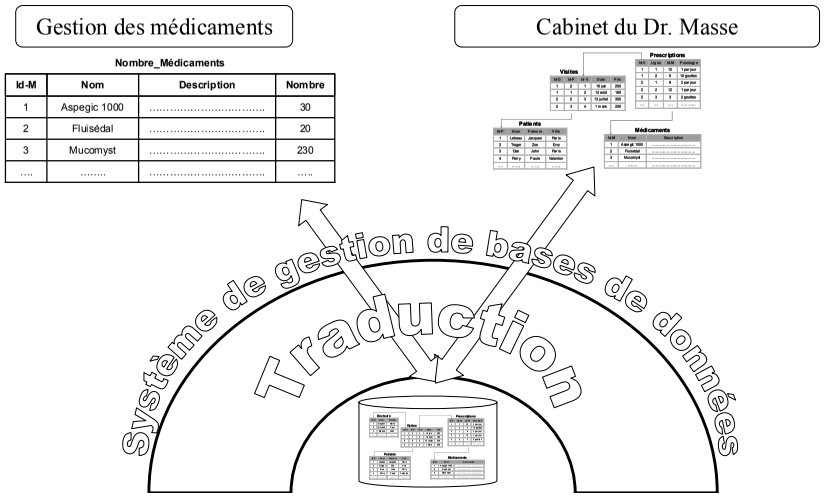
Id-M	Nom	Description
1	Aspegic 1000	…….
2	Fluisédal	…….
3	Mucomyst	…….
….	……..	…….

II - Indépendance Physique

- Indépendance des programmes d'applications vis à vis du modèle physique des données
- Bénéfices
  - Écriture des applications par des non-spécialistes des fichiers et des structures de stockage;
  - Possibilité de modifier les structures de stockage (fichiers, index, chemins d'accès, …) sans modifier les programmes;
  - Meilleure portabilité des applications et indépendance vis à vis du matériel.

III - Indépendance Logique

- Les appli. peuvent définir des **vues logiques** de la BD



Avantages de l'indépendance logique

- Possibilité pour chaque application **d'ignorer** les besoins des autres (bien que partageant la même BD)
- Possibilité **d'évolution de la base de données** sans réécriture des applications
  - Ajout/renommage de champs, ajout de relation
- Possibilité **d'intégrer des applications existantes** sans modifier les autres
- Possibilité de limiter les conséquences du partage : **Données confidentielles**

IV - Manipulation aisée

- La manipulation se fait via un langage **déclaratif**
  - La question déclare l'objectif sans décrire la méthode
  - Le langage suit une norme commune à tous les SGBD
  - **SQL : Structured Query Langage**
- Syntaxe (aperçu !)

Select	<Liste de champs ou de calculs à afficher>
From	<Liste de relations mises en jeu>
Where	<Liste de prédicats à satisfaire>
Group By	<Groupement éventuel sur un ou plusieurs champs>
Order By	<Tri éventuel sur un ou plusieurs champs>

Exemple de question SQL (1)

- Nom et description des médicaments de type aspirine

Select Nom, Description
From Médicaments
Where Type = 'Aspirine'

En algèbre:
$\Pi_{\text{Nom, Description}} (\sigma_{\text{Type=aspirine}} (\text{Médicaments}))$

Exemple de question SQL (2)

- Patients parisiens ayant effectués une visite le 15 juin

```
Select   Patients.Nom, Patients.Prénom
From     Patients, Visites
Where    Patients.Id-P = Visites.Id-P
     and      Patients.Ville = 'Paris'
     and      Visites.Date = '15 juin'
```

En algèbre:

$\Pi_{\text{Nom, Prénom}} (\sigma_{\text{Ville=paris et date = 15 juin}} (\text{Patients} \bowtie_{\text{Id-P=Id-P}} \text{Visites}))$

Exemple de question SQL (3)

- Dépenses effectuées par patient triées par ordre décroissant

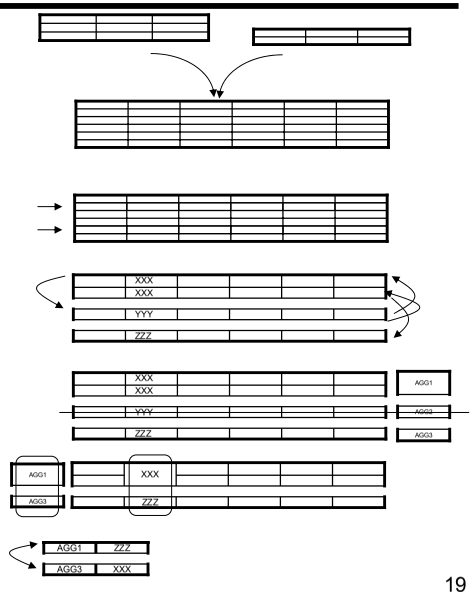
```
Select   Patients.Id-P, Patients.Nom, sum(Prix)
From     Patients, Visites
Where    Patients.Id-P = Visites.Id-P
GroupBy      Patients.Id-P, Patients.Nom
OrderBy      sum(Prix) desc
```

En algèbre:

$\gamma_{\text{Id-P, Nom, SUM(Prix)}} (\text{Patients} \bowtie_{\text{Id-P=Id-P}} \text{Visites})$

Évaluation « sémantique » d'une requête SQL

1. FROM
   Réalise le produit cartésien des relations
2. WHERE
   Réalise restriction et jointures
3. GROUP BY
   Constitue les partitions (e.g., tri sur l'intitulé du groupe)
4. HAVING
   Restreint aux partitions désirées
5. SELECT
   Réaliser les projections/calculs finaux
6. ORDER BY
   Trier les tuples résultat



III' – Gestion des vues

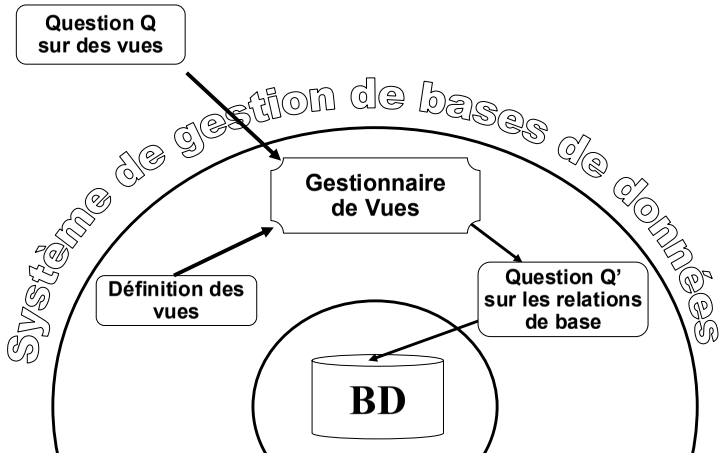
- Les vues permettent d'implémenter l'indépendance logique en créant des **objets virtuels**
- Vue = Question SQL stockée
- Le SGBD stocke la **définition** et non le résultat

- Exemple : la vue des patients parisiens

```
Create View Parisiens as (
        Select      Nom, Prénom
        From        Patients
        Where       Patients.Ville = 'Paris' )
```

Les vues : des relations virtuelles !

Le SGBD **transforme** la question sur les vues en question sur les relations de base



Les vues : Mise à jour

- Non définie si la répercussion de la mise à jour vers la base de données est ambiguë…
  - Ajouter un tuple à la vue « nombre de médicaments » ?
- Restrictions SQL (norme):
  - Pas de distinct, d'agrégats, ni d'expression de calcul
  - La vue contient les clés et les attributs « not null »
  - Il y a une seule table dans le « from »
  - Certains SGBDs supportent plus de mises à jour
- Clause « With check option »
  - Le SGBD vérifie que les tuples insérés ou mis à jour correspondent à la définition de la vue

Les vues : Les instantanés (snapshot)

- Instantané, Snapshot, vue concrète, vue matérialisée
  - Résultat matérialisé sur le disque
  - Accessible seulement en lecture
  - Peut être réactualisé
- Exemple
  - **create snapshot** Nombre_Médicaments **as**
    Select Id-M, Nom, Description, count(*)
    From Médicaments M, Prescriptions P
    Where M.Id-M = P.Id-M
    refresh every day
- Objectif principal : la performance

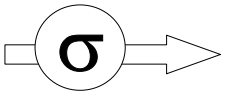
V –Optimisation automatique

- Traduction **automatique** des questions déclaratives en programmes impératifs :
  - ➔ Utilisation de l'algèbre relationnelle
- Optimisation **automatique** des questions
  - ➔ exploitation des propriétés (commutativité, distributivité …) des opérateurs de l'algèbre
  - ➔ Gestion centralisée des chemins d'accès (index, hachages, …)
  - ➔ Techniques d'optimisation poussées
- Économie de l'astuce des programmeurs
  - Milliers d'heures d'écriture et de maintenance de logiciels

Sélection

Patients

Id-P	Nom	Prénom	Ville
1	Lebeau	Jacques	Paris
2	Troger	Zoe	Evry
3	Doe	John	Paris
4	Perry	Paule	Valenton



Patients

Id-P	Nom	Prénom	Ville
1	Lebeau	Jacques	Paris
2	Troger	Zoe	Evry
3	Doe	John	Paris
4	Perry	Paule	Valenton

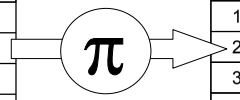
Patients de la ville de Paris, noté en algèbre:

$\sigma_{\text{Ville=paris}}$ **(Patients)**

Projection

Patients

Id-P	Nom	Prénom	Ville
1	Lebeau	Jacques	Paris
2	Troger	Zoe	Evry
3	Doe	John	Paris
4	Perry	Paule	Valenton



Patients

Id-P	Nom	Prénom	Ville
1	Lebeau	Jacques	Paris
2	Troger	Zoe	Evry
3	Doe	John	Paris
4	Perry	Paule	Valenton

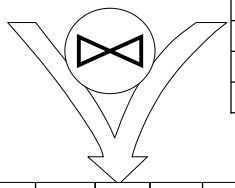
Nom et prénom des patients, noté en algèbre:

$\Pi_{\text{Nom, Prénom}}$ **(Patients)**

Jointure

Patients

Id-P	Nom	Prénom	Ville
1	Lebeau	Jacques	Paris
2	Troger	Zoe	Evry
3	Doe	John	Paris
4	Perry	Paule	Valenton



Visites

Id-D	Id-P	Id-V	Date	Prix
1	2	1	15 juin	250
1	1	2	12 août	180
2	2	3	13 juillet	350
2	3	4	1 mars	250

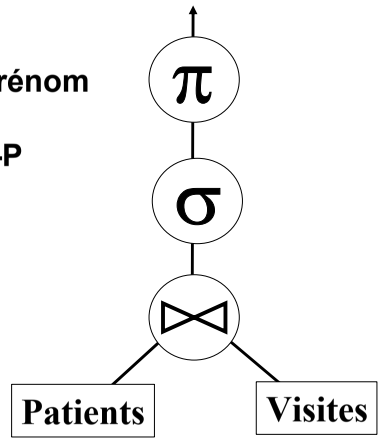
Id-P	Nom	Prénom	Ville	Id-D	Id-P	Id-V	Date	Prix
1	Lebeau	Jacques	Paris	1	1	2	12 août	180
2	Troger	Zoe	Evry	1	2	1	15 juin	250
2	Troger	Zoe	Evry	2	2	3	13 juillet	350
3	Doe	John	Paris	2	3	4	1 mars	250

Patients et leurs visites, noté en algèbre:

Patients $\bowtie_{\text{Id-P=Id-P}}$ **Visites**

Exemple de plan d'exécution

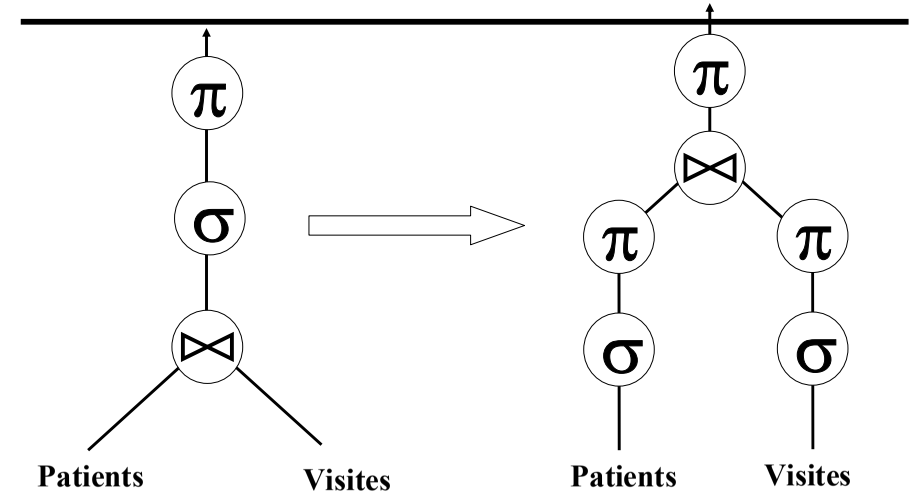
Select Patients.Nom, Patients.Prénom
From Patients, Visites
Where Patients.Id-P = Visites.Id-P
and Patients.Ville = 'Paris'
and Visites.Date = '15 juin'



En algèbre:

$\Pi_{\text{Nom, Prénom}}$ ($\sigma_{\text{Ville=paris et date = 15 juin}}$ (Patients $\bowtie_{\text{Id-P=Id-P}}$ Visites))

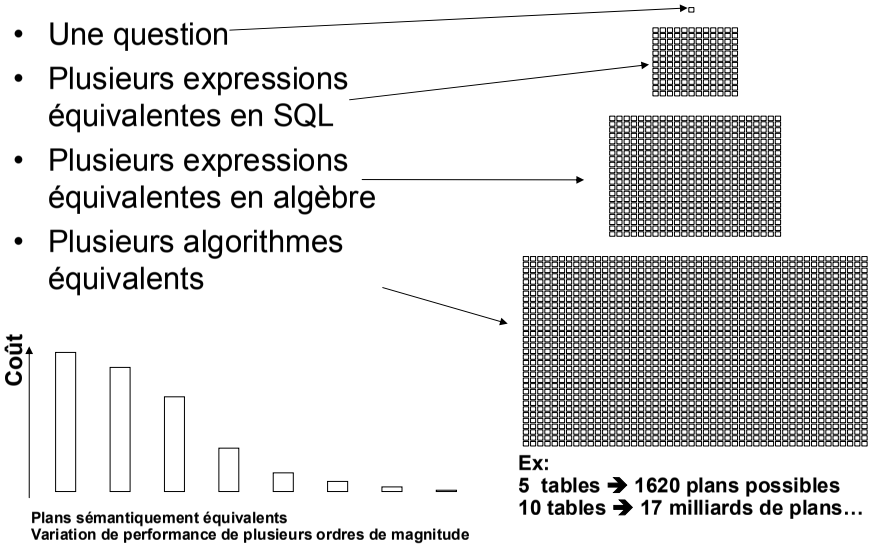
Plan d'exécution optimisé



$\Pi_{Nom, Prénom}$ ($\Pi_{Nom, Prénom, Id\text{-}P}$ $\Pi_{Id\text{-}P}$ ($\sigma_{Ville=paris}$ ($\sigma_{date = 15\ juin}$ (Patients)) (Visites)) $\bowtie_{Id\text{-}P=Id\text{-}P}$

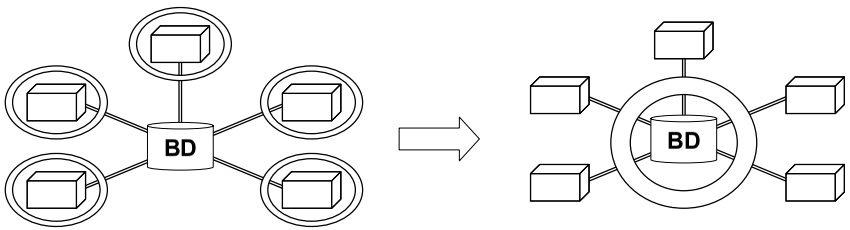
Optimisation réellement nécessaire ?

- Une question
- Plusieurs expressions équivalentes en SQL
- Plusieurs expressions équivalentes en algèbre
- Plusieurs algorithmes équivalents



VI – Intégrité sémantique

- Contrainte d'intégrité :
  - propriété sémantique que doivent respecter les données afin d'assurer la cohérence de la base.
- Objectif : Détecter les mises à jour erronées et réagir
  - simplification du code des applications
  - sécurité renforcée par l'automatisation
  - évolutivité des contraintes
  - Cohérence globale des contraintes



Typologie des contraintes d'intégrité (1)

- Contraintes de domaine (mono-attribut)
  - Contrôle de types : ex: Nom alphabétique
  - Contrôle de valeurs : ex: Salaire mensuel entre 1 et 10 K€
  - Non Nullité : ex: le Nom d'un patient doit être renseigné
- Contraintes multi-attributs mono-tuple
  - Relations entre données élémentaires : PrixVente > PrixAchat
  - Relations temporelles : le salaire d'un employé ne peut pas décroître
- Contraintes multi-tuples mono-table
  - Unicité : le nro insee détermine un patient unique
  - Contrainte agrégative : salaire du PDG = max(salaire)

Typologie des contraintes d'intégrité (2)

- Contraintes multi-tuples multi-tables
  - Contrainte d'intégrité référentielle : une visite doit être liée à un médecin et un patient existants
  - Un électeur doit être inscrit sur au plus une liste électorale
  - Contrainte agrégative : la somme des quantités vendues doit être inférieure ou égale aux quantités produites
  - Le médicament X ne doit pas être prescrit en même temps que Y si une contre-indication est référencée dans le Vidal
- Problème complexe
  - Nécessite un langage de déclaration et un mécanisme de vérification
  - Les SGBD commerciaux supportent généralement peu de contraintes (par rapport à la norme SQL2)
    - Principalement CI de domaine, unicité, référentielle

Contraintes d'intégrité : Exemple

- Contraintes d'intégrité référentielles

Docteurs

Id-D	Nom	Prénom
1	Dupont	Pierre
2	Durand	Paul
3	Masse	Jean
....	.....	.....

Visites

Id-D	Id-P	Id-V	Date	Prix
1	2	1	15 juin	250
1	1	2	12 août	180
2	2	3	13 juillet	350
2	3	4	1 mars	250

Prescriptions

Id-V	Ligne	Id-M	Posologie
1	1	12	1 par jour
1	2	5	10 gouttes
2	1	8	2 par jour
2	2	12	1 par jour
2	3	3	2 gouttes
....	....	....	.....

- Vérification lors de l'insertion, la suppression, la modification
- Propagation des suppressions/modifications en cascade possible (on delete/update cascade)

Contraintes d'intégrité : Syntaxe

create table <nom de table> (
<attribut> <domaine> **[<contrainte d'attribut>],** (mono-attribut)
<attribut> <domaine> **[<contrainte d'attribut>], …**
[<contrainte de relation>]) (mono ou multi-attributs)

- Différent types de contraintes :
  - Non nullité : **not null**
  - Unicité : **unique**
  - Vérification : **check <formule>**
  - Clé primaire : **primary key**
  - Contrainte d'intégrité référentielle : **references <relation> (<attribut>)**
    - **on delete / on update ➔ cascade, set null, set default**
- On peut nommer les contraintes

Déclencheurs : Définition

- Définition : Déclencheurs ou Triggers
  - Règle E – C – A
  - Événement – [Condition] – Action
- Lorsque l'évènement se produit
  - Insert / Update / Delete pour une relation donnée
- si la condition est remplie
  - Prédicat SQL optionnel
- alors exécuter l'action
  - Code à exécuter (ex. PL/SQL sous Oracle)
  - Pour chaque tuple concerné ou une fois pour l'événement

Déclencheurs : Objectifs

- Objectif : une base de données 'active'
  - valider les données entrées
  - créer un audit de la base de données
  - dériver des données additionnelles
  - maintenir des règles d'intégrité complexes
  - implanter des règles métier
  - supporter des alertes (envoi de e-mails par exemple)
- Gains
  - simplification du code des applications
  - sécurité renforcée par l'automatisation
  - les déclencheurs sont stockées dans la base
  - Cohérence globale des déclencheurs

Déclencheurs : Syntaxe (dans Oracle)

Create trigger <nom de trigger>
before | after — permet d 'indiquer quand le trigger va être exécuté
insert | delete | update [of <attributs>] — indique l'événement déclencheur
on <relation> — indique le nom de la table qui doit être surveillée
[referencing old as <var>, new as <var>] — en SQL3, Oracle utilise :new et :old
for each row — précise si l'action est exécuté 1 fois par tuple touché ou pour toute la table
[when <condition>] — permet d 'indiquer une condition pour l'exécution du trigger
DECLARE — déclaration de variables pour le bloc PL/SQL
BEGIN — bloc PL/SQL contenant le code de l'action à exécuter
  <PL/SQL bloc> — dans SLQ3, on peut indiquer une suite de commande SQL
END

- La syntaxe diffère légèrement suivant le SGBD…

Déclencheurs : Exemples simples

```
Create trigger calcul_TTC after insert on Vente
Begin
   update vente set Prix_TTC = Prix_HT*1.206
End ;
```

```
Create trigger ModifCommande after update on Commande
For each row
Begin
   if :new.qte< :old.qte
   then raise_application_error(-9996,' La quantité ne peut diminuer');
End ;
```

```
Create trigger ModifCommande after update on Commande
For each row
When (new.qte< old.qte)
Begin
   raise_application_error(-9996, ,' La quantité ne peut diminuer');
End ;
```

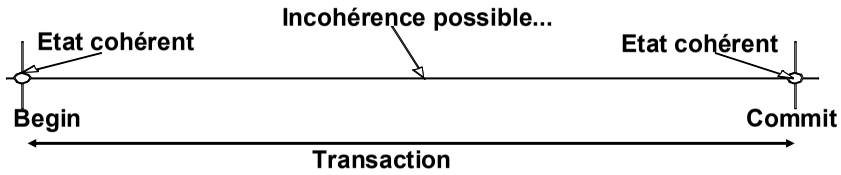

Déclencheurs : Remarques finales

- Cascade de triggers
  - l'action d'un trigger peut déclencher d'autres triggers
- Interactions avec les contraintes
  - l'action d'un trigger peut causer la vérification des contraintes
  - les actions des contraintes référentielles peuvent déclencher des triggers (delete cascade, update cascade)
- Mécanisme très (trop ?) puissant
  - Cascade 'infinie'
  - Tables en 'mutation'
    - ➔ Usage limité

VII – Tolérance aux pannes

- Motivations
  - Transaction Failure : Contraintes d'intégrité, Annulation
  - System Failure : Panne de courant, Crash serveur
  - Media Failure : Perte du disque
  - Communication Failure : Défaillance du réseau
- Objectifs
  - Assurer l'Atomicité des transactions
  - Garantir la Durabilité des effets des transactions commises
- Moyens
  - Journal mémorisant les états successifs des données
  - Mécanismes de reprise

Transaction



Begin
  CEpargne = CEpargne - 3000
  CCourant = CCourant + 3000
Commit T1

Atomicité et Durabilité

ATOMICITE

Begin
  CEpargne = CEpargne - 3000
  CCourant = CCourant + 3000
Commit T1

Panne

➔ **Annuler le débit !!**

DURABILITE

Begin
  CEpargne = CEpargne - 3000
  CCourant = CCourant + 3000
Commit T1

Crash disque

➔ **S'assurer que le virement a été fait !**

VIII: Accès concurrents aux données

— Objectif : assurer *l'Isolation* des transaction, c.à.d que différentes applications partageant les mêmes données doivent pouvoir s'ignorer et travailler de manière asynchrone.

— Le SGBD garantit la *sérialisabilité* des accès: l'effet d'une exécution simultanée de transactions doit être le même que celui d'une exécution séquentielle.

< T1 || T2 …|| Tn > ≡ < T1; T2; … Tn >

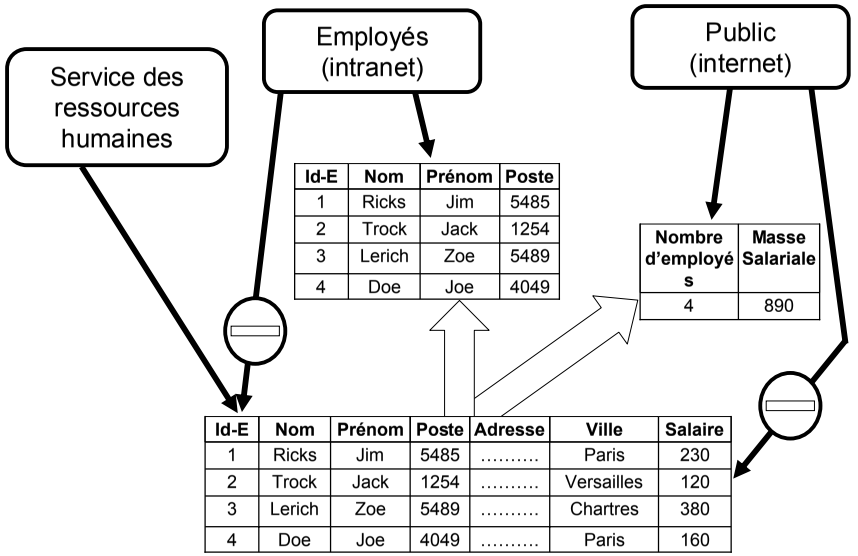
— Les transactions exécutées en parallèle ne doivent pas entrer en conflit lecture-écriture ou écriture-écriture, afin d'éviter :

- des pertes de mises à jour
- des introductions d'incohérence
- des lectures non reproductibles

IX – Confidentialité

- **Objectif : Protéger les données de la BD contre des accès non autorisés**
- Deux niveaux
  - Connexion restreinte aux **usagers répertoriés** (identification/authentification)
  - **Privilèges** d'accès aux objets de la base
- Usagers : Usager, rôles
- Objets : Relation, **Vue**, autres objets (procédures, etc.)

Puissance des droits SGBD



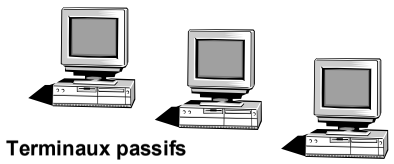
X - Standardisation

- L'approche bases de données est basée sur plusieurs standards
  - Langage SQL (SQL1, SQL2, SQL3)
  - Communication SQL CLI (ODBC / JDBC)
  - Transactions (X/Open DTP, OSI-TP)
- Force des standards
  - Portabilité
  - Interopérabilité
  - Applications multisources…

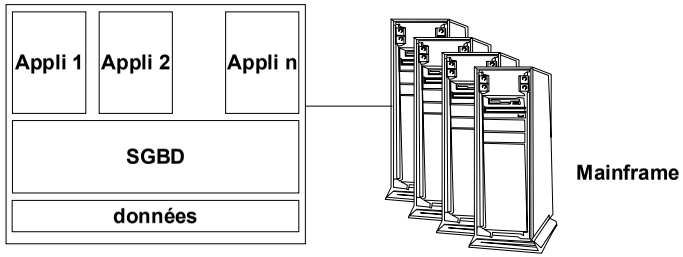
Architecture des SGBD

- Les architectures physiques de SGBD sont liées au mode de répartition
  - BD centralisée
  - BD client/serveur
  - BD 3-tiers
  - BD client/multiserveurs
  - BD répartie
  - BD hétérogène
  - BD mobile
- Eléments de vocabulaire …

Historiquement : architecture centralisée



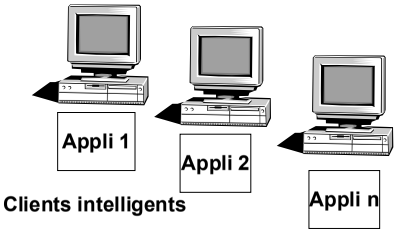
- Des terminaux clients
  - sans intelligence, passifs
- Un réseau
- Un ordinateur central
  - grande puissance (‘mainframe’)
  - Maintient la base et les applis



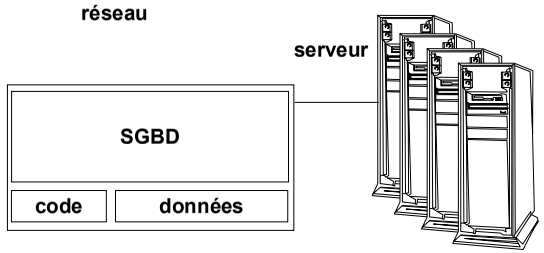
Exemple d’instance de cette architecture?

le minitel ☺

Architecture client serveur



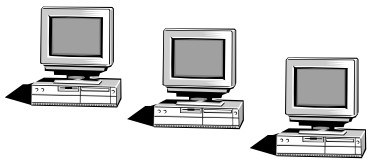
- Des clients intelligents
  - Font tourner les applications
- Un réseau
- Un serveur
  - Maintient la base



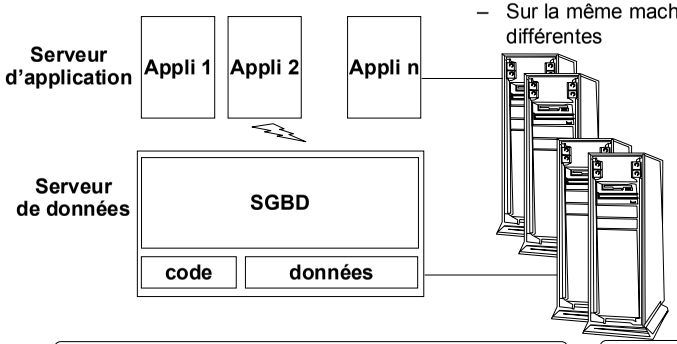
Exemple d’instance de cette architecture?

Buffon

Architecture 3-tiers



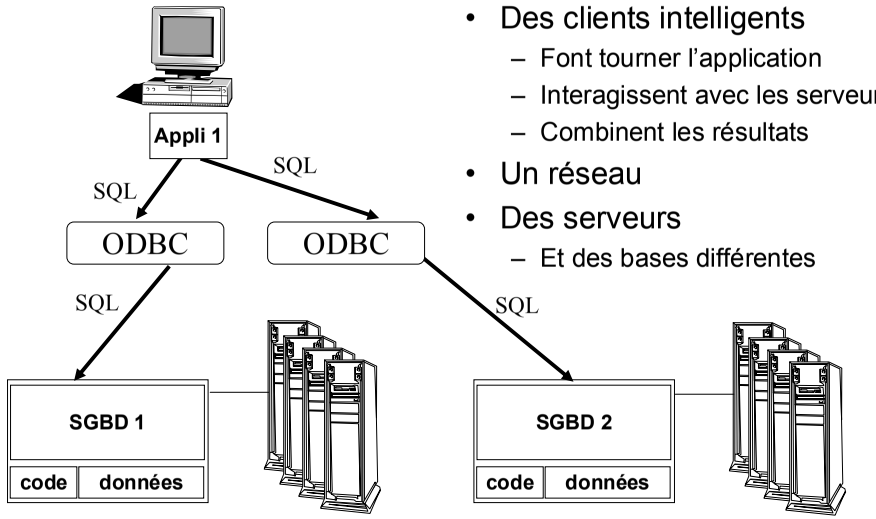
- Des clients
- Un réseau
- Un serveur d’application
  - Exécute le code applicatif
- Un serveur de données
  - Maintient la base
  - Sur la même machine ou des machines différentes



Exemple d’instance de cette architecture?

Appli Web

Architecture Client Multiserveurs

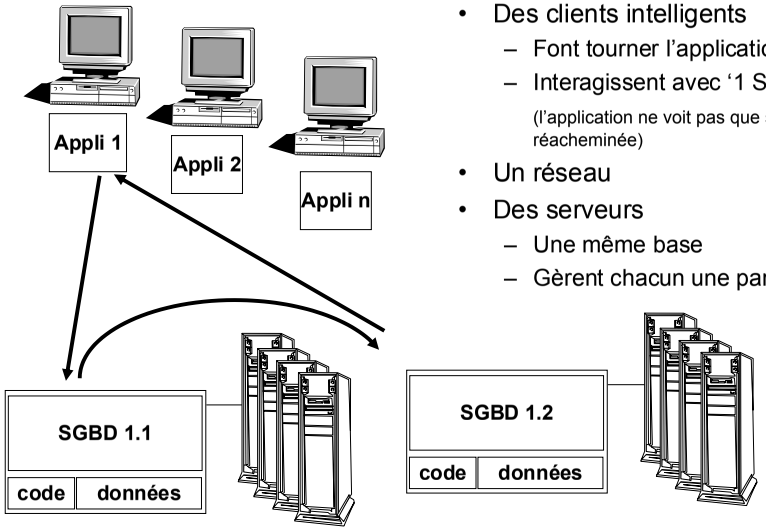


- Des clients intelligents
  - Font tourner l’application
  - Interagissent avec les serveurs
  - Combinent les résultats
- Un réseau
- Des serveurs
  - Et des bases différentes

Exemple d’instance de cette architecture?

Réservation d’un voyage

Architecture répartie



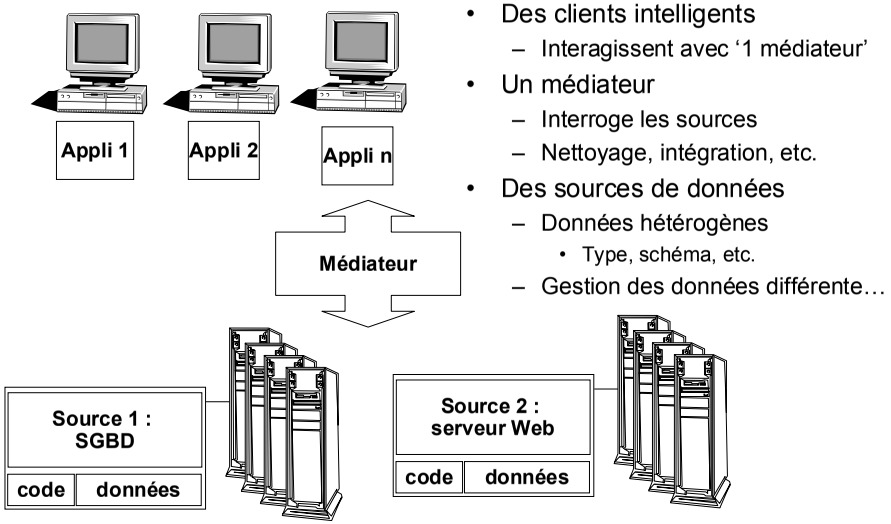
- Des clients intelligents
  - Font tourner l'application
  - Interagissent avec '1 SGBD'

    (l'application ne voit pas que sa requête est réacheminée)
- Un réseau
- Des serveurs
  - Une même base
  - Gèrent chacun une partition

Exemple d'instance de cette architecture?

Agences d'une société

Architecture hétérogène



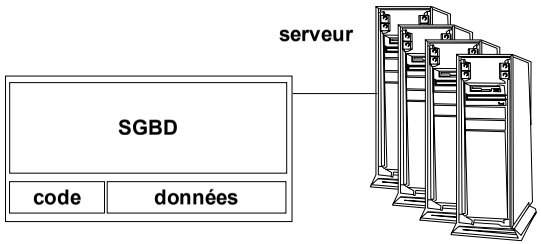
- Des clients intelligents
  - Interagissent avec '1 médiateur'
- Un médiateur
  - Interroge les sources
  - Nettoyage, intégration, etc.
- Des sources de données
  - Données hétérogènes
    - Type, schéma, etc.
  - Gestion des données différente…

Exemple d'instance de cette architecture?

Kelkoo

Architecture mobile

- Des clients mobiles intelligents
  - Portion SGBD répliquée
  - Interagissent avec SGBD externe
  - Fréquemment déconnectés
- Un réseau sans fil
- Un serveur
- Un protocole de synchronisation



Exemple d'instance de cette architecture?

Représentant de commerce

Applications traditionnelles des SGBD

- OLTP (On Line Transaction Processing)
  - Cible des SGBD depuis leur existence
  - Banques, réservation en ligne ...
  - Très grand nombre de transactions en parallèle
  - Transactions simples
- OLAP (On Line Analytical Processing)
  - Entrepôts de données, DataCube, Data Mining …
  - Faible nombre de transactions
  - Transactions très complexes

Nouvelles problématiques (1)

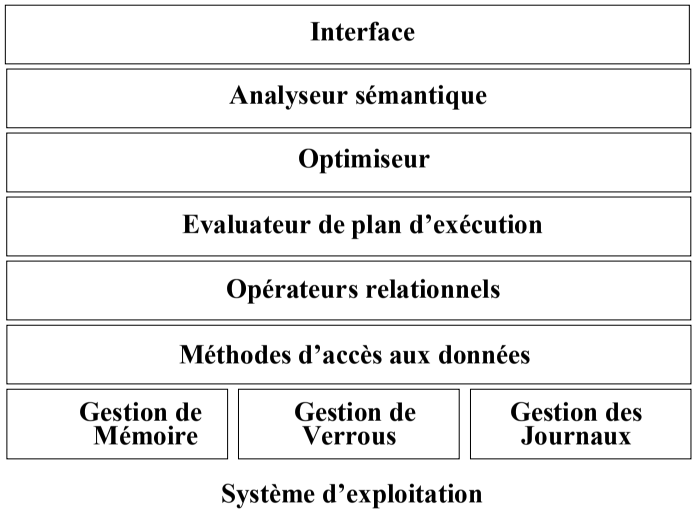
- Gestion de données complexes
  - Semi-structurées : stockage, indexation, interrogation de documents XML
  - Non-structurées : recherche par le contenu, index multidimensionnels, relations spatiales et temporelles
- Systèmes de médiation
  - De données : intégration de schémas, requêtes de médiation, interrogation large échelle (requêtes continues, personnalisation, critères approximatifs, etc)
  - De programmes : construction de workflows, optimisation des flux, grilles de calcul
- Entrepôts de données et fouille
  - Rafraîchissement, requêtes multi-dimensionnelles (cubes)
  - Recherche de règles associatives, de dépendances fonctionnelles, time series …

Nouvelles problématiques (2)

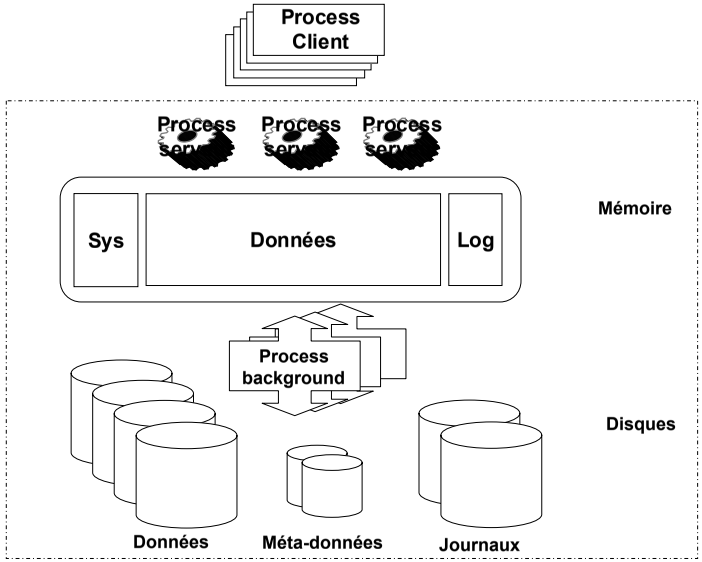
- Mobilité
  - BD de localisation, BD embarquées, cohérence des traitements déconnectés, réseaux de capteurs …
- Sécurité des bases de données
  - Modèles de contrôle d'accès et d'usage, chiffrement de bases de données, anonymisation, hardware sécurisé, BD Hippocratiques, Private Information Retrieval …
- Architecture et performance
  - Grandes mémoires, caches, nouvelles mémoires persistantes, nouveaux processeurs …
  - SGBD auto-administrables
  - Encore et toujours : indexation, optimisation de requêtes, réplication, transactions, benchmarks, etc …

Et comment ça marche ?

Architecture fonctionnelle d'un moteur de SGBD



Architecture opérationnelle d'un moteur de SGBD

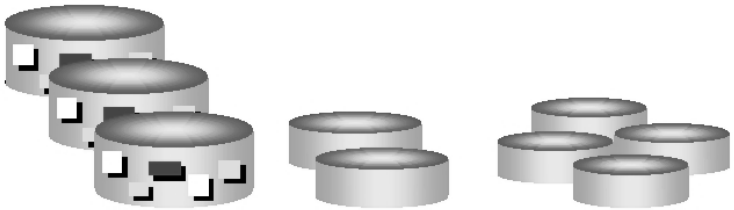


l'Architecture d'Oracle8

Certains slides empruntés à Pascale Borlat Salamet – Oracle France

ORACLE

La Base de Données Oracle



Fichiers de Données

Fichiers de Contrôle

Fichiers Redo Log

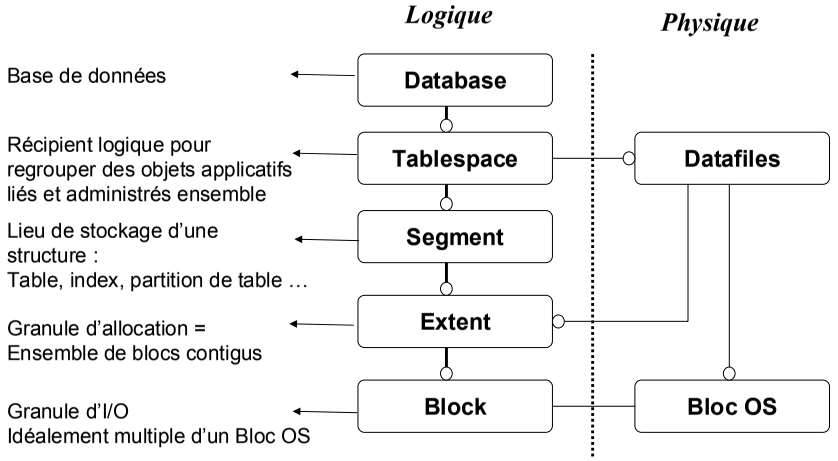
Contiennent les tables, index …

Contiennent les métadonnées, procédures stockées, triggers …

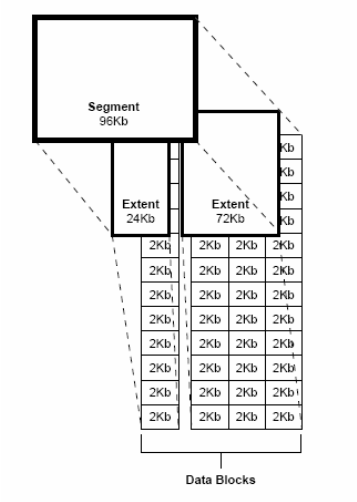
Contiennent les journaux

ORACLE

Organisation des données sur disque



Blocs, extents et segments



Instance Oracle

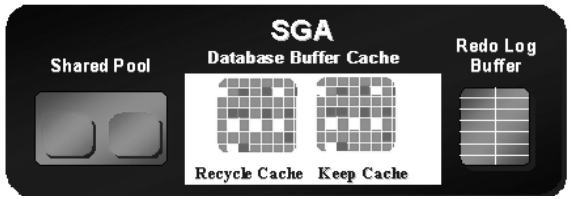
- 3 types de processus Oracle
  - Processus clients (exécutent les programmes d'application)
  - Processus serveurs (exécutent les commandes BD)
  - Processus background (prennent notamment en charge les I/O …)
- Espace mémoire
  - System Global Area (SGA) : partagée par tous les processus
  - Program Global Area (PGA) : privée à chaque processus du système
    - Exemple pour les processus serveurs
      - Private SQL Area : Structures runtime liées à l'exéc des statements SQL : curseurs, variables de session
      - SQL Work Area : mémoire de travail pour l'exécution des requêtes : zone de tri, Hash-Join, construction d'index bitmap …
  - Software Code Area (SCA) : zone de code logiciel
    - Stocke le code Oracle
- Une *Instance Oracle* :
  - 1 SGA + les processus background de cette SGA + leurs PGAs

Structure de la SGA (1)

- Shared Pool : Structures systèmes et méta-données
  - Structures système, procédures PL/SQL, dictionnaire et autre méta données liées à l'exécution de statements SQL (ex: plans d'exécution, verrous)
  - Paramétrage : shared_pool_size
- Buffer Cache : Stocke les blocs lus du disque
  - Sert de tampon pour lire/écrire dans la BD
  - Peut être décomposé en deux zones
    - Keep Buffer Cache : pour les blocs accédés très fréquemment
    - Recycle Buffer Cache : pour les blocs pouvant être recyclés rapidement
  - Paramétrage
    - db_cache_size, *db_keep_cache_size*, *db_recycle_cache_size*
- Redo log buffer : Stocke les enregistrements de journaux
  - Contient uniquement les données modifiées (et pas tout le bloc)
  - Utile pour la tolérance aux pannes
  - Paramétrage : log_buffer

Structure de la SGA (2)

- Et d'autres encore…
  - Specific block size caches : stocke les blocs de taille particulière
    - Le DBA peut stocker certaines tables (tablespace) dans des blocs de taille différente des autres tables
      - 2Ko, 4Ko, 8Ko, 16Ko, et 32Ko sont possibles
    - Les blocks de ces tables sont stockés dans ce cache accommodant leur taille particulière
    - Paramétrage : db_nk_cache_size (après Oracle9i)
  - Large pool : Données de backup et de restauration, données liées à Oracle XA
    - Paramétrage : large_pool_size (après Oracle8)
  - Java pool : Stocke des méthodes et des définitions de classes Java
    - Paramétrage : java_pool_size
  - …
- Vue simplifiée des Buffers du SGA



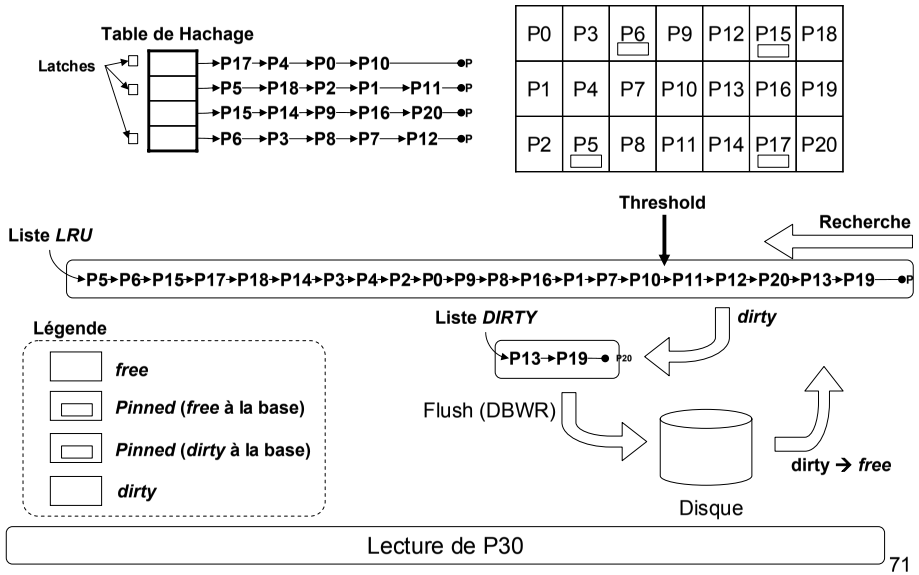
Fonctionnement du Buffer Cache

- Les blocs de la liste ont 3 états possibles
  - *free* : bloc libre
  - *pinned* : en cours d'utilisation → ne peut être ré-attribué
  - *dirty* : modifié → non consistant/disque
- Structures maintenues
  - Une table de Hachage
    - Mapping : adresse bloc sur disque → entrée du bloc en RAM
  - Une Liste *LRU* (mélangeant blocs *free, pinned, dirty*)
    - Chaînage des blocs du plus récemment au moins récemment utilisé
    - NB: utilisé = en consultation (SELECT) ou en modification (UPDATE)
  - Une Liste *DIRTY* (blocs *dirty*)
    - Reportée sur le disque par le processus DBWR (DataBase WRiter).

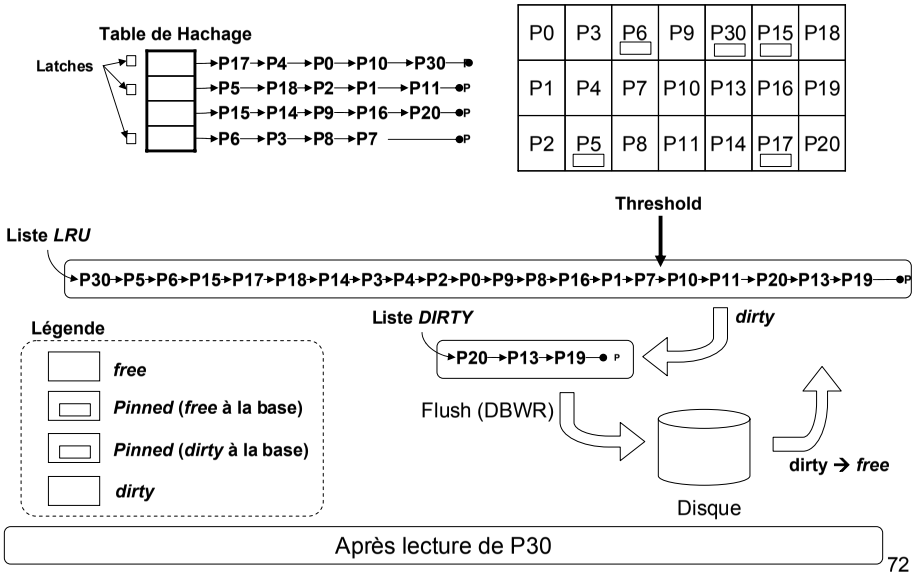
Fonctionnement du Buffer Cache (2)

- Algorithme : Demande de la page Pi du disque
  - H(Pi) → entrée RAM
  - Si la page est présente
    - Page cible trouvée...
  - Si la page est absente
    - Parcours liste *LRU* par la fin jusqu'à rencontrer une page *free*
      - Toute page *dirty* rencontrée est ajoutée à la liste dirty (sans la retirer de la liste *LRU*)
      - Si le *threshold* est rencontré
        - » Appel à BDWR pour reporter sur disque des pages de la liste *DIRTY* (flush pages *dirty* sur disque, les retire de la liste *DIRTY*, les passe à *free* dans LRU)
        - » Les pages reportées passent de *dirty* → *free*
        - » replace le curseur de recherche en fin de liste LRU …
    - Lit la nouvelle page du disque vers la page *free* ciblée
    - Modifie la table de hachage (retire la page *free* ciblée, ajoute la page lue)
    - Page cible trouvée…
  - Marquer la page cible comme *pinned*
  - Retourne la page
- NB : accès concurrents
  - Des « latch » (verrous courts) doivent être posés sur la table de hachage

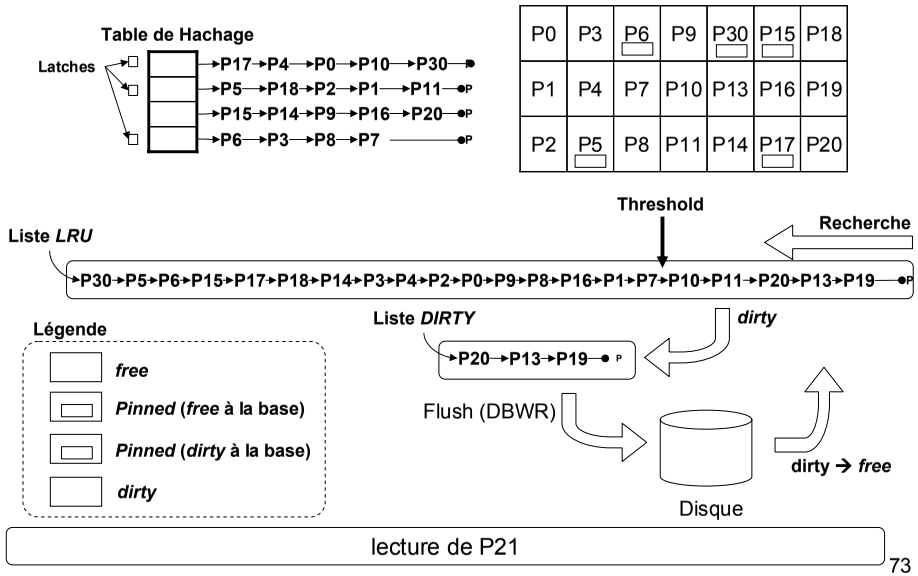
Fonctionnement du Buffer Cache (3)



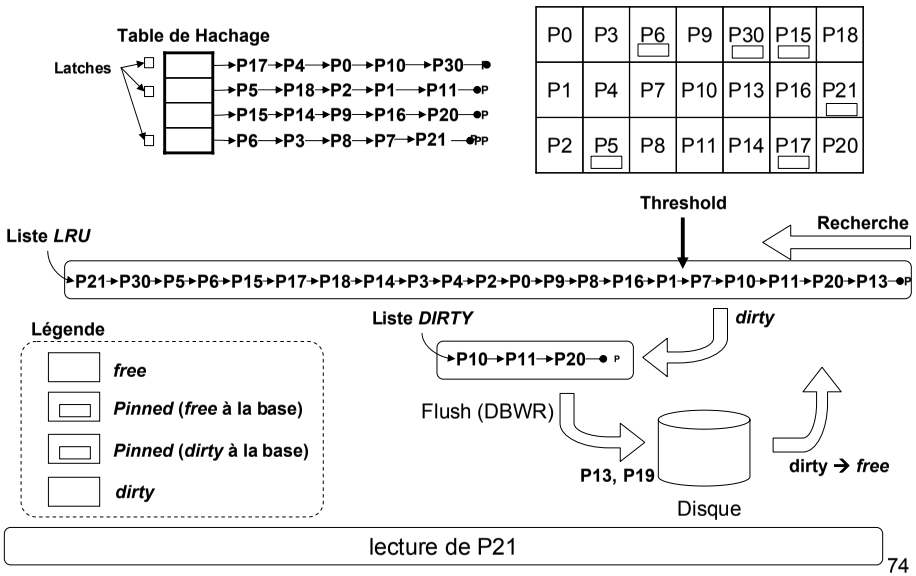
Fonctionnement du Buffer Cache (3)



Fonctionnement du Buffer Cache (3)



Fonctionnement du Buffer Cache (3)



Les Processus Background (1)

- Une instance est partagée par plusieurs utilisateurs
  - ➔ il faut centraliser les tâches (évite les problèmes de concurrence) et les factoriser pour ne pas gaspiller de ressources

Mais pourquoi en BACKGROUND ?

Pour optimiser les performances. Un système performant doit utiliser les ressources au maximum en évitant les pics (goulots d'étranglement)…

- Au démarrage de l'instance
  - PMON, SMON, DBWR, LGRW, CKPT, ARCH, RECO
  - Et d'autres processus liés aux calculs de statistiques, déclenchement d'alertes, etc

Les Processus Background (2)

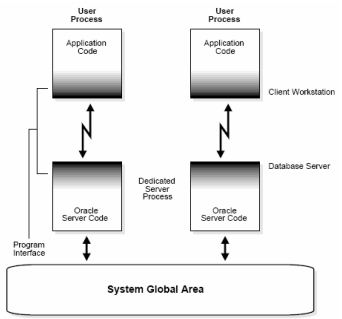
- DBWR (DataBase Writer)
  - Reporte les blocs modifiés (*dirty*) sur le disque et les libère dans le cache
    - son but est de conserver le cache propre
    - Dans l'ordre indiqué par la 'liste *DIRTY*' (voir slides précédents)
- LGRW (LoG Writer)
  - Responsable de l'écriture du Redo Buffer sur le disque
    - En fait, écriture sur des disques mirroir
  - Il s'exécute
    - à chaque COMMIT (Force-Log at Commit)
    - toutes les 3 secondes
    - quand le buffer est rempli au premier tiers
    - À chaque intervention du DBWR (WAL)

Les Processus Background (3)

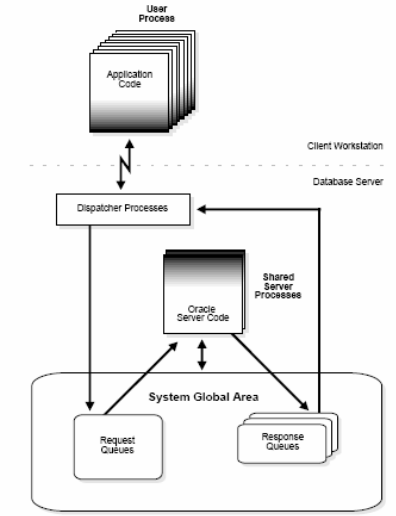
- CKPT (ChecKPoinT)
  - Gestion des *checkpoint*
  - Enregistre toute les 3 secondes dans le Control File un marqueur indiquant quelle portion du Redo Log est déjà reportée sur disque
- ARCH (ARCHiver)
  - Effectue l'archivage du fichier REDO LOG
- PMON (Processus MONitor)
  - Nettoie les connexions terminées de façon anormale
  - Défait les transactions après une 'transaction failure'
- SMON (System MONitor)
  - Assure la reprise d'une instance après une 'system failure'
- RECO (RECOverer)
  - Termine les transactions stoppées dans une phase de validation atomique suite à une panne (in-doubt transactions)
  - Présent uniquement dans une configuration distribuée

Processus serveur dédiés ou partagés

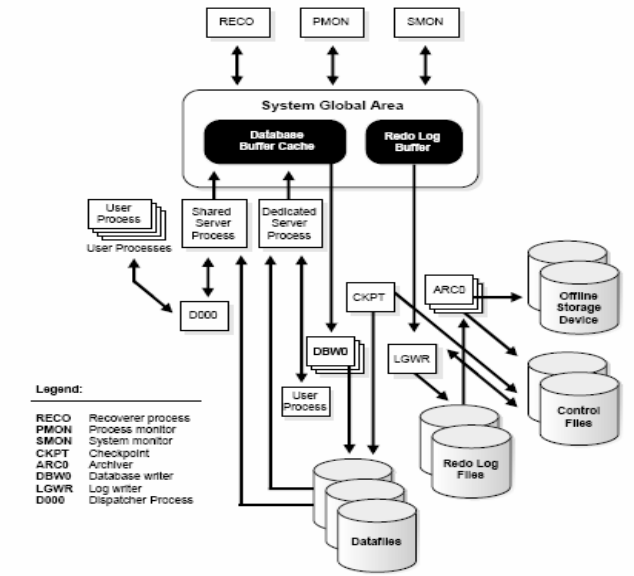
Serveurs dédiés



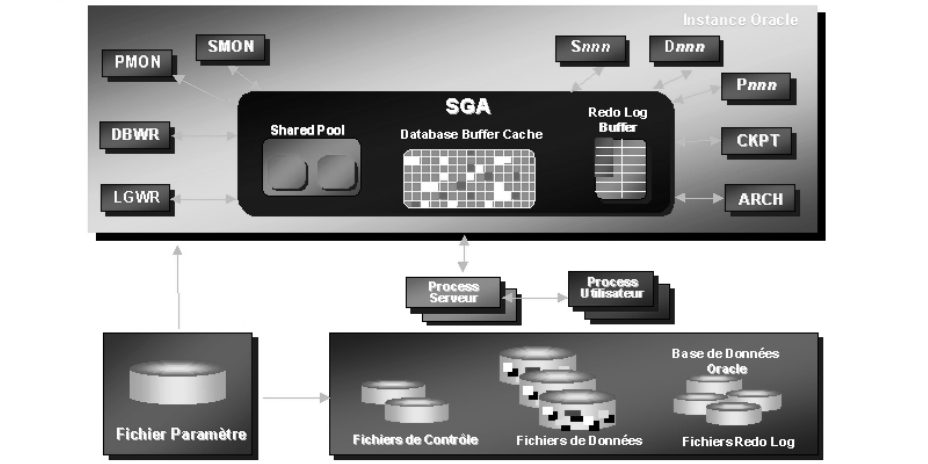
Serveurs partagés



Vue globale des processus background



Résumé



ORACLE

Bibliographie

- Le Noyau Oracle
  - Oracle8, Cours ENST de Pascale Borlat Salamet, Oracle France
    http://www.infres.enst.fr/people/saglio/bdas/00/orapbs/index.htm
  - Oracle9 et Oracle 10, Database Concepts
    www.lc.leidenuniv.nl/awcourse/oracle/server.920/a96524/toc.htm
    http://www-smis.inria.fr/~anciaux/Oracle10_Database_Concepts.pdf
  - Rapports Techniques divers
    - Triviadis : http://www.trivadis.com
    - Dbspecialists : http://www.dbspecialists.com/presentations/

- Livres Bases de Données
  - Database Management Systems
    Livre de R. Ramakrishnan et J. Gehrke
  - Bases de Données
    Livre de G. Gardarin, Editions Eyrolles, 2003
    http://georges.gardarin.free.fr/