

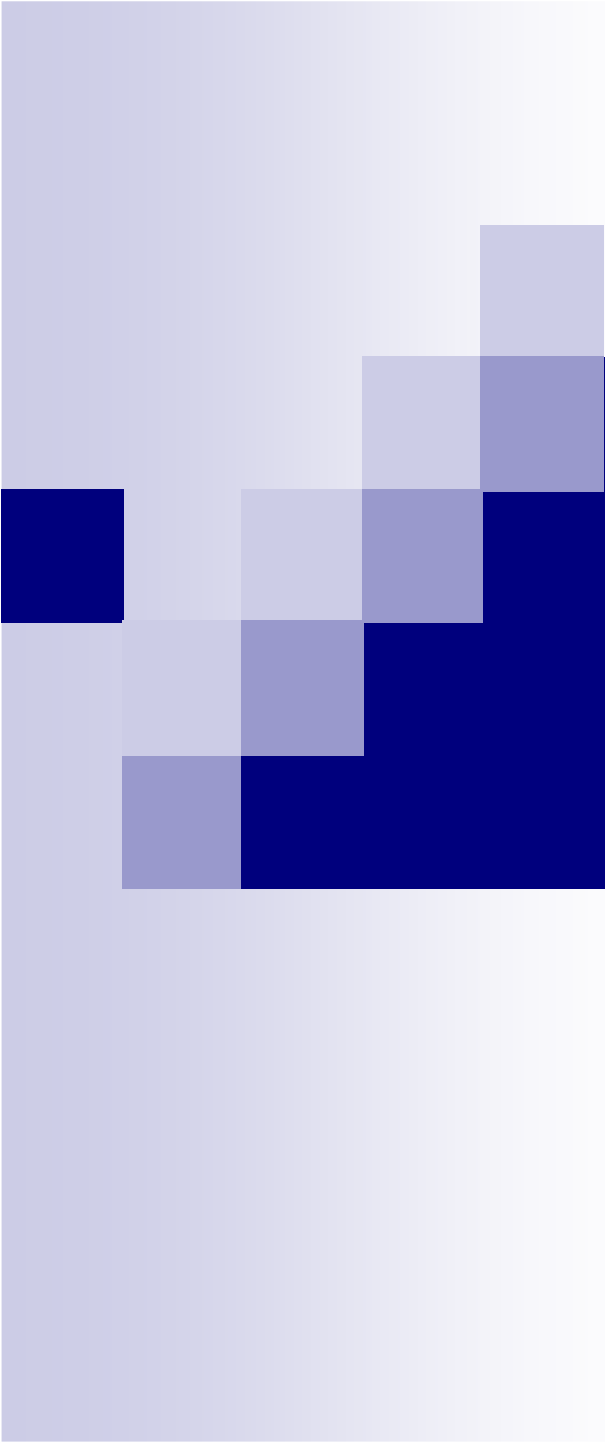


Organisation du projet BD



Dates importantes

- 8 janvier : lancement
- 13 janvier : rapport de modélisation
- 16 janvier : rapport contraintes
- 21 janvier : rapport transactions
- 29 janvier : remise du rapport final + code
- 30 janvier: soutenances



Rappels et compléments techniques pour le projet BD



Plan

- Contraintes
- Triggers
- Transactions
- JDBC
- Gestion des dates
- Tests
- Soutenances
- Compléments divers sur Oracle



Contraintes (rappels)

- Cf. TP en début d'année
- Possibilité de retarder la vérification d'une contrainte jusqu'au moment de la validation d'une transaction (commit)
 - Ajouter DEFERRABLE derrière la déclaration
 - Préciser ensuite si la vérification est initialement différée ou pas : INITIALLY DEFERRED (ou IMMEDIATE)
 - Par défaut, une contrainte est déclarée NOT DEFERRABLE INITIALLY IMMEDIATE
 - On peut modifier au sein d'une transaction le mode d'une contrainte (si elle a été déclarée DEFERRABLE) : SET CONSTRAINT <nom> <IMMEDIATE | DEFERRED>;



Triggers : Rappels

- Cf. TP en début d'année
- Un trigger surveille :
 - Une seule table
 - Un ou plusieurs événements (insert/update/delete) dans la table
 - À un instant donné (avant ou après l'exécution de la requête)
 - En mode lignes ou état
- Un trigger peut effectuer des vérifications liées à plusieurs contraintes
- La vérification d'une contrainte peut nécessiter plusieurs triggers
- Problème des tables mutantes : il n'est pas possible de manipuler une table dans le corps d'un déclencheur (en mode lignes) qui porte sur cette table



Triggers : erreurs fréquentes (1/2)

- Mode lignes
 - Sémantique
 - Tables mutantes
- Confusion sur la chronologie (before/after)
- Penser à la taille du résultat d'une requête d'interrogation en PL/SQL
 - Possibilité d'obtenir un ensemble vide => prévoir l'exception NO_DATA_FOUND
 - Possibilité d'obtenir plusieurs tuples =>
 - Utiliser un opérateur d'agrégation (count)
 - ... ou un (ou plusieurs) curseur(s) si les tuples doivent être analysés individuellement



Triggers : erreurs fréquentes (2/2)

- Oubli de surveiller certaines tables pour la vérification d'une contrainte
- Exemple :
 - « *Un pilote ne peut être affecté que sur un avion pour lequel il est qualifié.* »
 - Il faut surveiller la table AFFECTATIONNP ... mais aussi la table QUALIFICATION



Triggers : Compléments (1/2)

- Ordre d'exécution théorique
 - ☐ Déclencheurs en mode état before
 - ☐ Analyse de toutes les lignes affectées par la requête
 - ☐ Déclencheurs en mode lignes before
 - ☐ Verrouillage, modifications, vérification des contraintes d'intégrité
 - ☐ Déclencheurs en mode lignes after
 - ☐ Vérification des contraintes différées
 - ☐ Déclencheurs en mode état after



Triggers : Compléments (2/2)

- Désactivation d'un trigger
 - ALTER TRIGGER nomDeclencheur DISABLE;
 - ALTER TABLE nomTable DISABLE ALL TRIGGERS;
- Réactivation
 - ALTER TRIGGER nomDeclencheur ENABLE;
 - ALTER TABLE nomTable ENABLE ALL TRIGGERS;
- Attention : la désactivation temporaire d'un trigger est déconseillée. À n'utiliser qu'en dernier recours.



Transactions (rappels)

Niveaux d'isolation

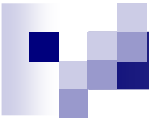
- Seulement deux niveaux d'isolation sous Oracle :
READ_COMMITTED et SERIALIZABLE
- READ_COMMITTED
 - Permet seulement de lire des modifications validées (pas de lectures sales)
 - Lectures non reproductibles et fantômes possibles
- SERIALIZABLE
 - Synchronisation des données visibles par une session uniquement lorsque la transaction locale se termine (commit/rollback)
 - Lectures non reproductibles et fantômes impossibles
 - Niveau d'isolation supérieur mais performances restreintes



Transactions (rappels)

Niveaux d'isolation

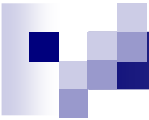
- Oracle définit également un mode « lecture seule »
 - pour les transactions qui ne modifient pas la base.
 - On obtient ainsi un niveau d'isolation équivalent à `SERIALIZABLE`.
- La transaction doit commencer par l'instruction suivante : `SET TRANSACTION READ ONLY;`
- Attention :
 - Certaines opérations sont interdites au sein d'une transaction en lecture seule
 - Ne pas oublier de terminer la transaction (`commit` ou `rollback`)



Transactions

Verrouillage

- Par défaut, Oracle utilise des verrous (exclusifs) uniquement pour les écritures
- ... mais peut quand même détecter des séquences qui sont non-conformes au niveau d'isolation
SERIALIZABLE
 - *Oracle generates an error when a serializable transaction tries to update or delete data modified by a transaction that commits after the serializable transaction began*
 - Code d'erreur 8177 retourné suite à une opération non sérialisable => il faut annuler la transaction (et éventuellement essayer de la relancer)



Transactions

Verrouillage

- Dans certains cas, il peut être nécessaire de verrouiller des tuples lors d'une lecture au sein d'une transaction.
 - Pour cela, on peut utiliser FOR UPDATE à la fin d'une requête d'interrogation (SELECT)
- Les verrous sont automatiquement relâchés à la fin d'une transaction



Transactions

Niveaux d'isolation et verrouillage

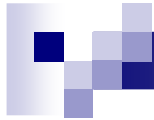
- Pour plus de détails, voir la documentation Oracle :
 - Oracle 11g Database concepts, Chapter 13 (Data concurrency and consistency)
 - Introduction to data concurrency and consistency
 - How Oracle manages data concurrency and consistency
 - Oracle 11g Application developer's guide



Transactions

Intégration avec les triggers

- Les requêtes SQL exécutés par un trigger ont la même vue de l'état de la base que la transaction englobante
- Un trigger ne peut pas prendre de décision sur la terminaison de la transaction englobante
 - Il peut seulement accepter ou refuser (en renvoyant une erreur) l'opération sur la base
 - En fonction du succès ou de l'échec d'une opération (ou d'un ensemble d'opérations), le code applicatif (écrit en PL/SQL ou Java dans le cadre du projet) décide de valider (commit) ou d'annuler (rollback) la transaction



JDBC

- L'essentiel à déjà été vu en TP
- Méthodes importantes pour les interactions avec la base via l'interface statement
 - executeQuery pour les requêtes d'interrogation (qui retournent un ensemble de tuples)
 - executeUpdate pour les requêtes d'insertion, modification, suppression de tuples (et aussi instructions LDD)
 - execute
 - pour les instructions LDD (create, drop, alter)
 - Pour l'appel de fonctions et procédures stockées (via l'interface CallableStatement)



JDBC

Erreurs fréquentes

- Confusion entre `executeQuery` et `executeUpdate`
- Réutilisation d'un statement pour exécuter une nouvelle requête alors qu'on a encore besoin du `resultSet` associé à la requête précédente
- Mauvaise gestion des exceptions
 - Exceptions simplement propagées ou attrapées mais sans véritable traitement
 - Oubli de libérer les ressources devenues inutiles (`resultset`, `statement`, `connection`)



Gestion des dates

- Dans la BD, utiliser le type SQL « date »
- Dans l'application :
 - À la frontière avec la base : `java.sql.Date` (sous classe de `java.util.Date`)
 - Dans la logique « métier » : `java.util.Calendar`
- Documentation
 - PL/SQL
 - Arithmétique sur les dates :
http://www.oraFAQ.com/wiki/SQL_FAQ (sections 7 et 8)
 - Voir aussi documentation de `to_date` et `to_string`
 - Java : cf. javadoc (Sun/Oracle)



Tests

- Aspect primordial
- Couverture
 - Vérifier que le comportement du système est conforme aux spécifications, dans tous les cas
- Automatisation
 - Objectif : simplifier l'analyse des résultats et faciliter les tests de non-régression
 - Un bon jeu de tests doit seulement produire un message de résultat : succès ou erreur (+ détails éventuels)
 - Code PL/SQL et/ou Java



Exemple de méthodologie de test


Tentative d'insertion de tuples (corrects) :

- 1) Initialiser/vérifier l'état de départ de la base
=> récupérer le nombre de tuples (n_i)
- 2) Insertion des tuples
=> incrémenter un compteur (c)
- 3) Récupérer le nombre final (n_f) de tuples dans la base
- 4) Comparer ($n_i + c$) à n_f



Soutenances / Démonstrations

- Préparer des scénarios pertinents pour illustrer :
 - les fonctionnalités implantées
 - le bon fonctionnement du système
- Pour chaque scénario de test, décrire clairement :
 - l'état initial
 - l'état final attendu
 - le résultat effectivement obtenu
- Essayer d'automatiser au maximum le lancement des démonstrations
- ... mais prévoir un mode de saisie interactif minimaliste pour permettre de lancer d'autres requêtes selon les demandes des enseignants



Oracle SQL Developer

Description

- Environnement intégré pour le développement et le test de code SQL et PL/SQL
- Disponible sur toutes les plateformes (basé sur Java)
- Connexion JDBC avec base de données
- Liens
 - Description des fonctionnalités :
<http://www.oracle.com/technetwork/developer-tools/sql-developer/what-is-sqldev-093866.html>
 - Documentation :
http://download.oracle.com/docs/cd/E12151_01/index.htm
 - Téléchargement : <http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html>



Oracle SQL Developer Installation

- Sur vos machines personnelles
 - Récupérer la version packagée pour votre système
 - <http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html>
 - Nécessite la création (gratuite) d'un compte OTN
- Sur les machines de l'UFR
 - Pas encore installé par défaut sur les PC/serveurs
 - Récupérer la bonne version (200-300Mo)
 - Copier et extraire l'archive sur votre compte
 - Lancement
 - Sous Windows : `sqldeveloper.exe`
 - Sous Unix : `sh sqldeveloper.sh`



Connexion avec Oracle sur im2ag-oracle

■ Rappel sur JDBC

- URL 11g: `jdbc:oracle:thin:@im2ag-oracle.e.ujf-grenoble.fr:1521:ufrima`
(pwd: bd2012 / 2013)
- Revoir TP JDBC pour l'établissement d'une connexion de manière programmatique
- Les pilotes Oracle 11g :
<http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html>



Connexion avec Oracle sur im2ag-oracle

- Depuis vos machines personnelles
 - Via le VPN
 - URL JDBC inchangée
 - ... mais peut poser des problèmes dans certains cas
 - Via un tunnel SSH
 - Redirection d'un port local (par exemple 1521) vers le serveur JDBC
 - Création du tunnel : cf. fichier README-tunnel_ssh.txt
 - Nécessite de modifier l'URL de connexion :
jdbc:oracle:thin@localhost:<numéro de port local>:ufrima



Documentation Oracle 9.2

- Site Oracle

- <http://www.oracle.com/technetwork/database/enterprise-edition/documentation/database-093888.html>

- Code d'erreurs

- http://tahiti.oracle.com/pls/db92/db92.error_search?search=



Commandes utiles pour les affichages d' Oracle

■ Sous sqlplus :

- ☐ set echo {on | off} : affichage des commandes lors de leur exécution
- ☐ set feedback {on | off} : affichage du nombre de tuples impactés par une requête
- ☐ set serveroutput {on | off} : affichage des traces applicatives (DBMS_OUTPUT.PUT_LINE)



Commandes utiles pour les affichages d' Oracle (2)

- Dans le code PL/SQL :
 - Paquetage DBMS_OUTPUT
 - Permet d' écrire sur la console
 - DBMS_OUTPUT.ENABLE : activation du paquetage
 - DBMS_OUTPUT.PUT_LINE : affichage
 - Exemple 1 : DBMS_OUTPUT.PUT_LINE('du texte');
 - Exemple 2: DBMS_OUTPUT.PUT_LINE('valeur de la variable : ' || myvar || ' et date courante : ' || SYSDATE);