# Principes d'utilisation des systèmes de gestion de bases de données

Cours 1

Ecole Informatique d'Orsay, M1

2005/2006

Emmanuel Waller

LRI

# Organisation du module

- site du cours
- M1 mentions Informatique et Miage
- prérequis  L Informatique ou Miage (précisions ult.)
- 11 x 2h cours + 2h TD (toujours en salle machine ; table + machine)
  + soutenance projet (4h) vendredi 16/12/5
  + examen (3h) vendredi 6/1/5 (session 1 : tous documents autorisés ; session 2 : aucun document)
- note du module :
  session 1 : (2 exa + proj) / 3 (à vérifier)
  session 2 :

Miage/Informatique

- équipe enseignante TD :
  Miachon, Nguyen, Senellart (Miage)
  Goasdoué, Waller (Informatique)

- délégués :
  élection TD2
  point avec enseignant 5 mn à chaque fin de cours et TD

# Documents cours 1

- Transparents du cours
- énoncé TD 1
- Projet partie 1
- Annales 2003/04
- Aide-mémoire machines
- Documentation en ligne

# Quoi ? Qui ? Quand ? Pourquoi ? Comment ?

Titre : "Principes d'utilisation des SGBD"

- Qu'est-ce que c'est ?
- Qui peut l'utiliser ?
- Dans quelles situation ?
- Pour résoudre quel type de problèmes ?
- Comment l'utiliser ?

# Les problèmes de base de données : exemple

# Contexte et problèmes BD

- Contexte :
- Vraie vie
- Programmeurs écrivent applications
- Rencontrent nombreux problèmes
- Il faut les gérer

# Exemple

- Application = situation à informatiser
- Réservation de places SNCF
- Clients effectuent réservations
- Employés gèrent trains
- Réservation = client, date, numéro train, numéro place
- Train = départ, arrivée, km, horaire, numéro train, ensemble des places libres, nombre total de places

# Problème 1 à gérer

- La situation :
- Client effectue sa réservation sur minitel (web, etc.)
- Il se déconnecte
- Il faut :
- Sa réservation ne doit pas disparaître (stockée)
- Son numéro de carte bleue doit disparaître après paiement (non stockée)

# Problème 2

- Situation :
- Quantité d'informations supérieure à mémoire vive
- Accès disque lent
- Il faut :
- Temps de réponse raisonnable

# Problème 3

- Situation :
- Réservation et numéro CB tapés par client
- Paiement effectué
- Panne de courant avant que réservation stockée
- Il faut :
- Paiement et réservation ne doivent pas être dissoci

# Problème 4

- Situation :
- Il reste une place dans le Paris-Saint Tropez de 17h05
- 2 clients se connectent simultanément
- Il faut :
- On accepte que l'un passe en premier et l'autre n'ait rien
- Mais pas que les deux paient pour le même siège

# Problème 5

- Situation :
- Un client veut modifier
- L'horaire d'un train
- Les réservation d'autres clients
- Il faut :
- Qu'il ne puisse pas
- Un employé doit pouvoir

## Problème 7

- Situation :
- Pour une réservation Paris-New York votre application doit dialoguer avec l'application homologue américaine
- Il faut :
- Tous les problèmes doivent être gérés en coopération entre les deux applications

# Problème 8

- Situation :
- Une fois votre application écrite : trop lente
- Pour accélérer : réorganiser informations sur le disque
- Il faut :
- Vous ne voulez pas réécrire toute votre application à chaque fois

# Problèmes 9 à 12

- Situation :
- Il y a des informations à manipuler par l'application
- Il faut :
- Les représenter (pb 9)
- Les organiser (pb 10)
- Les créer, modifier, détruire (pb 11)
- Les interroger (pb 12)
- De manière simple (maintenance, efficacité, non informaticien ; comparaison avec programmes Java)

# Vocabulaire

- Information = donnée
- Ex : Il y a un train Paris-Saint Tropez
- Ex : Jules a une place dans le Paris-Saint Tropez
- Informations = base de données
- Ces 12 problèmes sont appelés les "problèmes de base de données"

# Les 12 problèmes de base de données

- Persistance
- Grandes quantités
- Reprise sur panne
- Contrôle de concurrence
- Confidentialité
- Contraintes d'intégrité
- Répartition
- Indépendance des niveaux
- Modèle de données

- Conception
- Mise à jour
- Interrogation

# Vocabulaire

- Domaine des bases de données = étude des outils théoriques et pratiques pour résoudre ces problèmes (recherche, R & D, technologie, méthodologie, etc.)
- Définition : SGBD = logiciel pour gérer les problèmes de base de données
- Pour chaque problème, le SGBD fournit un ensemble d'outils (conceptuels et/ou pratiques) permettant au programmeur de gérer ce problème (peut être difficile)

# Vocabulaire

- On utilise le SGBD au moyen d'un ensemble d'ordres dits "de base de données; : le langage SQL
- Une application = des ordres de base de données (écrits par vous)
  - tapés directement au clavier (mode interactif)
  - lancés depuis un programme (mode programme)

# Vue d'ensemble (dessin tableau)

- Les deux modes d'utilisation d'un SGBD :
- Vous en train de taper devant votre clavier : mode interactif
- Gérer la partie base de données de votre application en écrivant les ordres de base de données nécessaires
- Session = les ordres SQL entre connexion et déconnexion
- Programme en train de s'exécuter : mode application
- Mêmes gestion et ordres de base de données qu'en interactif
- Lancés depuis un programme (C, Java, web, etc.)

# Vocabulaire : construire un SGBD

- pour chacun des 12 problèmes :
- étude théorique
- étude système et implantation
- ex : modèle, langage de requêtes

# Les modules BD à Orsay

- conception, modèle, ... : L2
- interrogation, mises à jour, conception avancée, contraintes, ..., introduction à la formalisation : L3
- utilisation : M1 (nous)
- introduction à la théorie : M1 “BD avancées”
- construction (stockage, grandes quantités, etc.) : M2Pro
- tous aspects avancés : M2Recherche

# Rappel : Quoi ? Qui ? Quand ? Pourquoi ? Comment ?

Titre : "Principes d'utilisation des SGBD"

- Qu'est-ce que c'est ?
- Qui peut l'utiliser ?
- Dans quelles situation ?
- Pour résoudre quel type de problèmes ?
- Comment l'utiliser ?

# Qui ?

- Programmeur d'application
- DBA
- Secrétaire
- PDG
- Etc.

# But du module

- Répondre à ces 5 questions au moyen de :
- réflexion fondamentale (adaptables : situations, durée)
- pratique (opérationnels)

# Autrement dit, il faut acquérir les compétences pour :

- "pourquoi" : comprendre les mécanismes au coeur de ces problèmes
- "quand" : détecter en pratique les situations dans lesquelles ces problèmes peuvent survenir
- "comment" : tirer le meilleur parti du SGBD pour les résoudre concrètement

# TD

- Réflexion : sur table
- Pratique : sur machine
- Logiciel : Oracle 8i/9i
- Mais tout le contenu du module est soigneusement indépendant d'un SGBD particulier (quasiment : variantes, syntaxe)

## Nous :

- Apprendre à détecter ces problèmes lors de l'écriture d'une application
- Pour les résoudre
- Comprendre leurs mécanismes
- Utiliser les outils du SGBD (concepts et ordres) pour concevoir et programmer notre solution pour notre application

# Nous (suite) :

- Pour une application donnée :
- Considérer chacun des 12 problèmes
- Détecter toutes ses occurrences dans l'application
- Programmer une solution pour chacune des occurrences
- Répartition : hors programme

## Contenu du module

- les problèmes de base de données (en mode interactif)
- le mode programme : PL/SQL, Java, C, Web

# Plan du cours 1

- logistique
  - organisation du module
  - documents cours 1
- introduction : le domaine des bases de données
  - du titre...
  - exemple
  - les problèmes de bases de données
  - vocabulaire
  - les modules de BD à Orsay
  - but du module
  - plan du module

  - plan du cours 1
- gérer les problèmes de base de données
  - rappels de L3
    * le modèle relationnel
    * conception
    * interrogation
    * mises à jour
    * contraintes d'intégrité, séquences
    * indépendance des niveaux et vues
    * utilisation concrète en TD
  - persistance
  - confidentialité
- présentation du projet

# Les problèmes de base de données

# Principe de cette partie

- On va considérer les problèmes BD un par un
- Pour chacun, on va :
- Comprendre oà est le problème
- Comprendre et utiliser les outils que fournit le SGBD :
- Les concepts pour approcher ce problème
- Les ordres de base de données pour programmer une solution concrète à une application

# Plan pbs BD

- Modèle de données
- Conception
- Mise à jour
- Interrogation
- Contraintes d'intégrité
- Indépendance des niveaux
- Persistance
- Confidentialité
- Reprise sur panne
- Contrôle de concurrence

- Grandes quantités
- Répartition

# Rappels de L3

## Remarque

- Les problèmes 9 et 10 (modèle et conception) ne se résolvent pas par des ordres de base de données
- Voyons-les maintenant

# Problèmes 9 et10

- Situation :
- Il y a des informations à manipuler par l'application
- Il faut :
- Les représenter (pb 9)
- Les organiser (pb 10)
- De manière simple (non informaticien ; comparaison avec types Java)

# Le modèle relationnel

# Brève intuition (rappel L3)

- Solution au problème 9. Utiliser SGBD : représenter nos informations dans son modèle
- Table (relation), colonnes (attributs), lignes (n-uplets), répétitions, types (entier, réel, chaîne, etc. : module non exhaustif, varchar2)
- Distinguer :
- Description des données : structure de la table (schéma)
- Données elles-mêmes : contenu de la table (instance)

# Le modèle relationnel

- Numéro train Destination
- 121 Paris
- 459 Saint Tropez
- 228 Rio de Janeiro
- 121 Paris
- 122 Paris
- 123 New York

## Le modèle relationnel

- Il existe une valeur spéciale, dite indéfinie, notée : NULL
- Permet de repésenter information incounne
- Base de données = ensemble de tables (schéma de la base, instance de la base)

# Compétences à acquérir

- Savoir représenter les informations de l'application par des tables avec leur contenu
- "Bonnes tables" : voir problème 10
- motivation : comparaison avec types Java

# Conception de schéma (Brève intuition, rappel L3)

- 
- Solution au problème 10 : organiser l'information en choisissant les "bonnes" tables
- critères : sans redondance (SR) et sans perte d'information (SPI)
- Lors de l'écriture de l'application :
- Faire la liste des informations pertinentes
- Les mettre dans des tables

# Compétences à acquérir

- Détecter redondances et pertes d'information
- Concevoir un schéma SR et SPI
- Le SGBD ne fournit pas d'outil pour la conception
- Hors programme (L3) : théorie (dépendances, formes normales)

# Plan pbs BD

- Modèle de données
- Conception
- Mise à jour
- Interrogation
- Contraintes d'intégrité
- Indépendance des niveaux
- Persistance
- Confidentialité
- Reprise sur panne
- Contrôle de concurrence

- Grandes quantités
- Répartition

# Rappel : problèmes 11 et 12

- Situation :
- Il y a des informations à manipuler par l'application
- Elles sont représentées dans des tables bien conçues
- Il faut :
- Les créer, modifier, détruire (pb 11)
- Les interroger (pb 12)
- De manière simple (maintenance, efficacité, non informaticien ; comparaison avec programmes Java)

# Le langage d'interrogation

- Solution au problème 12, ordres BD SQL
- Un ordre SQL d'interrogation est appelé une requête
- But : extraire des informations de la base de données
- Supposé connu : hors programme
- Compétences possédées : jointure simple

# Le langage de mise à jour

- Solution au problème 11
- Sous-ensemble de SQL, ordres de bases de données (comme toute la partie II)
- But :
- Créer, modifier, détruire (renommer)
- Tables :
- Structure
- Contenu

## Structure

- Créer, renommer, supprimer table
- Ajouter, modifier, supprimer
- Colonne
- Contrainte
- Options (documentation)
- Effet sur le contenu de la table (ou non)

## Contenu

- Insérer
- Modifier
- Supprimer
- Ensemble de lignes (zéro, une, plusieurs)

## Syntaxe

- Exemples
- TD, documentation
- Liste exhaustive des ordres de mise à jour (selon version SGBD)

# Exemples (1)

- create table piece (nom char(10), prix real)
- insert into piece values ('vis', 0.0)
- insert into piece values ('clou', 3.0)
- select * from piece
- select nom from piece where prix ¡ 1.0
- Indenter !!!

# Exemples (2)

- update piece set prix = prix + 3.0 where nom = 'vis'
- delete piece where nom = 'vis'
- select nom from piece where nom like 'c
- insert into piece values ('ecrou', null)
- select * from piece where prix is null

# Exemples (3)

- alter table piece add (num integer)
- rename piece to catalogue
- create table cheap(name, price) as select nom, prix from piece where age ¡= 4.0
- drop table piece

## Compétences à acquérir

- “Analyse” : savoir dire ce que fait un programme donné
- “Construction” : savoir écrire tout programme de transformation d’une base de données

# Plan pbs BD

- Modèle de données
- Conception
- Mise à jour
- Interrogation
- Contraintes d'intégrité
- Indépendance des niveaux
- Persistance
- Confidentialité
- Reprise sur panne
- Contrôle de concurrence

- Grandes quantités
- Répartition

# Contraintes : rappel du problème

- Situation :
- La manipulation des informations (humains, programmes) engendre des risque d'erreurs
- Il faut :
- Date ou kilométrage ne doivent pas être négatifs
- Pour un train donné : nombre de réservations + nombre de places libres = nombre total de places
- Cas général : il faut empêcher les valeurs
- Absurdes
- incohérentes entre elles

# Analyse du problème et outils SGBD

- Une mise à jour, humaine ou par programme, peut résulter en des données "incorrectes" (abmées, non "intgres")
- Outils fournis par le SGBD :
- Un langage (ordres SQL) de contraintes pour définir pour chaque application ce que le programmeur entend par "correctes"
- Détection automatique des mises à jour violant les contraintes déclarées
- Refus par le SGBD de ces mises à jour (ordre SQL de mise à jour sans effet)

# Le langage de contraintes (1)

- Obligation de renseigner une colonne
- Unicité d'une valeur dans une colonne
- Types énumérés
- Clef primaire
- Clef étrangre ("intégrité référentielle")
- Inclusion entre les valeurs de deux colonnes
- Vérification d'une formule arithmétique portant sur les lignes d'une table (chaque ligne indépendamment des autres)

# Le langage de contraintes (2)

- Exhaustivité du cours, possibilités exactes du langage selon le SGBD et la version ?
- Limitations :
- Comparaison avec :
- La logique du premier ordre
- Fonctions Java
- Voir aussi cours 3

# Exemples et ordres SQL

- create table t (
- a integer not null
- , b integer unique
- , c char(2)
- , primary key (a, c)
- , foreign key (b) references client(age)
- )
- alter table t add (check (a + b ¡ 12) constraint ew27)

# Séquences

- Situation :
- Nécessité en pratique de clefs numériques sans sémantique
- Outils fournis par le SGBD :
- un générateur de valeurs séquentielles (“numéros de séquence”) pour :
- Génération d’une nouvelle valeur
- Accès à la dernière valeur générée, pour coordonner les valeurs des clefs dans plusieurs : tables, lignes

# Ordres SQL (1)

- Création d'un générateur (appelé une "séquence") :
- Create sequence nom
- [ Increment by ( 1 — val ) ]
- [ start with val ]
- [ ( maxvalue v — no maxvalue) ]
- [ ( min value v — no minvalue) ]
- [ ( cycle — no cycle) ]
- [ ( cache val — 20) — no cache]

# Ordres SQL (2)

- Nocycle : arrêt quand max (min)
- Cache : prégénération de numéros de séquence (pas d'attente lors demande de valeur ; défaut $= 20$)
- Confidentialité (cours 2) : création nécessite privilège
- Quelques précisions et compléments dans documentation

# Ordres SQL (2)

- Utilisation d'une séquence :
- Dans select, insert, update
- Pseudo-colonne
- nom_sequence.nextval génère la prochaine valeur (première si premier appel) (ex : insert clef primaire)
- nom_sequence.currval : renvoie la valeur courante (= celle générée par le dernier nextval de la session) (ex : insert de clef étrangère)
- Utilisable simultanément par plusieurs utilisateurs (donc présence de "trous")

# Exemple

- Create sequence sgbd
- Start with 1000
- Incrment by 10
- Nomaxvalue
- Nocycle
- Insert into people values (sgbd.nextval, 'toto', 21)
- Insert into owner (sgbd.currval, 'ferrari')

# Compétences à acquérir

- "Analyse" : savoir prédire le comportement d'une séquence d'ordres SQL en présence de contraintes
- "Construction" :
- Savoir écrire les ordres SQL adéquats pour modéliser les contraintes d'un cahier des charges
- Connatre les possibilités du langage
- Motivation : comparaison avec une fonction Java vérifiant une contrainte

# Architecture du SGBD (dessin tableau)

- La bonne architecture pour gérer les problèmes BD :
- La base de données : données
- Le SGBD (programmes en train de s'exécuter) :
- Le serveur : unique, tourne en permanence
- Les clients (interactifs ou applications)
- En nombre quelconque
- Se (dé)connectent à volonté
- Non coordonnés

# Commande Unix (1)

- Le client Oracle s'appelle SQL*Plus
- Commande unix pour lancer un client SGBD : sqlplus
- unix¿ sqlplus
- unix¿ sqlplus waller/waller
- unix¿ sqlplus waller/waller@oradb

# Commande Unix (2)

- (Dessin tableau... écran, éditeur, unix waller, client sqlplus x 3 (waller, goasdoue, waller), serveur, disque, espace utilisateur)
- Compte Unix $=/=$ compte SGBD
- Déroulement :
- Taper ordres BD dans fichier unix
- Copié-collé dans fenêtre SQL*Plus

# Espaces utilisateurs

- (Dessin tableau... clients x 3 (waller, goasdoue, herault), serveur, BD (mêmes noms de table x 3))
- Disque :
- Espace unix
- Espace SGBD :
- Pour chaque utilisateur :
- Espace
- Tables

# Redirection des entrées/sorties

- (Dessin tableau... BD, serveur, client, fichiers x 2)
- Start (fichier unix : ordres SQL)
- Indépendant de la BD
- Spool (fichier unix)

# Rappel processus Unix

- ps -edf | grep sqlplus
- kill -9 ...

# Plan pbs BD

- Modèle de données
- Conception
- Mise à jour
- Interrogation
- Contraintes d'intégrité
- Indépendance des niveaux
- Persistance
- Confidentialité
- Reprise sur panne
- Contrôle de concurrence

- Grandes quantités
- Répartition

# Indépendance des niveaux : rappel du problème

- La situation :
- Il peut être nécessaire de réorganiser certaines parties de l'application pour cause de
- Performances
- évolutivité
- Il faut :
- Pouvoir le faire sans changer son comportement

# Les 3 niveaux des applications de bases de données

- "Logique" : tables relationnelles
- "Physique" : les détails de leur organisation physique sur le disque
- "Externe" : les programmes accédant aux tables relationnelles
- dessin

# Performances

- Performances d'une requêtes dépendent des détails du stockage sur le disque
- Il faut pouvoir modifier cette organisation sans modifier la structure des tables relationnelles
- Autrement dit : le niveau logique doit être indépendant du niveau physique

# Evolutivité

- Au cours du temps les fonctionnalités de l'application évoluent, en particulier les tables
- Mais les programmes utilisant les anciennes tables doivent toujours fonctionner avec le même comportement
- Le niveau externe doit être indépendant du niveau logique

# Outils fournis par le SGBD

- Le dictionnaire de données : indépendance des niveaux logique et physique
- Notion de vue : indépendance des niveaux externe et logique

# Le dictionnaire de données (1)

- Indépendance
- il stocke pour chaque table l'adresse du début de la liste chaînée des blocs physiques du disque qui contiennent les enregistrements de la table, etc.
- Conséquence du modèle de haut niveau (pb 9)
- Pour traduction logique vers physique

# Le dictionnaire de données (2)

- Répertorie informations concernant la base de données :
- Tables, utilisateurs, événements, etc.
- Stockées dans des... tables ! ("tables système")
- Select * from dictionary
- Select table_name from all_tables where owner = 'WALLER'
- Voir ligne de t nécessite privilège select sur t
- Accès par vues

# Notion de vue

- Vue :
- Outil du SGBD pour assurer indépendance entr eles niveaux externe et logique
- Permet aussi gérer certains aspects de :
- Grandes quantités (pb 2)
- Confidentialité (pb 5)
- Confort

# Exemple

- Situation :
- Billet ( client, destination, prix )
- Secrétaire SNCF Toto (catégorie lecture seule) gère uniquement les clients pour Saint Tropez
- Tape toujours la même requête (ex : gros quotient) :
- Select client
- From billet
- Where destination like 'St Trop

# Problèmes (1)

- Insupportable :
- Confort : il connaît très mal SQL
- Les tables de l'application
- Performances : elle est réexécutée à chaque fois (grandes quantités)
- Confidentialité : bien qu'il n'en ait pas besoin pour sa gestion, nécessite droits select sur
- Colonne prix de billet (cas général : toutes tables)
- Lignes de billet autres que Saint Tropez

# Problèmes (2)

- En cas de réorganisation des tables par le programmeur de l'application en :
- Voyage ( client, ntrain )
- Train ( ntrain, destination)
- La requête
- select client from billet where destination like 'St Tropne fonctionne plus
- Indépendance des niveaux

# Vocabulaire

- Une vue est un table virtuelle dont le contenu est défini par une requête
- Son contenu peut être au choix
- Recalculé
- Matérialisé (et maintenu)
- (Mises à jour : dépend des cas)

# Ordres SQL

- Create view mareq
- As
- Select client
- From billet
- Where destination like 'St Trop

# Conséquences (1)

- Toto tape simplement : Select * from mareq
- Confortable (plus de quotient)
- Résultat matérialisé (plus d'attente)
- Confidentialité respectée : il ne connaît que la table virtuelle mareq

# Conséquences (2)

- En cas de réorganisation des tables, le programmeur de l'application tape :
- Drop view mareq
- Create view mareq as select client from voyage, train where voyage.ntrain = train.ntrain and destination like 'St Trop
- La requête de Toto
- select * from mareq
- fonctionne toujours
- Toto n'a même pas besoin d'être prévenu

# En résum

- Le concept de vue permet de gérer des aspects de confort et de 3 problèmes de base de données :
- Indépendance des niveaux
- Confidentialité
- Grandes quantités
- utilisable par catégorie lecture seule

# Les 3 niveaux de la norme ANSI 78

- (Dessin tableau...)
- BD physique, serveur, clients interactifs et programmes
- Séparation des niveaux physique, logique, externe
- Vues
- Dictionnaire de données :
- Informations
- Traduction
- indépendance dans les quatre directions

# Compétences à acquérir

- "Analyse" :
- Vues : connatre le comportement d'une application en présence de vues
- "Construction" :
- Vues : définir et utiliser quand il faut et comme il faut
- Dictionnaire de données : savoir y récupérer les informations nécessaires

# Plan pbs BD

- Modèle de données
- Conception
- Mise à jour
- Interrogation
- Contraintes d'intégrité
- Indépendance des niveaux
- Persistance
- Confidentialité
- Reprise sur panne
- Contrôle de concurrence

- Grandes quantités
- Répartition

# Persistance : rappel du problème

- La situation :
- Un client effectue sa réservation
- Il se déconnecte
- Il faut :
- Sa réservation ne doit pas disparaître
- Son numéro de CB doit disparaître après paiement
- Cas général : certaines données (pas toutes) doivent être encore là (où ?...) après terminaison du processus client

# Persistance

- Problème BD 1
- Outils de gestion fournis par le SGBD :
- Notions (pour les ordres BD) de :
- Confirmation
- Annulation
- Règle :
- Une donnée persiste en fin de session si, et seulement si :
- Elle est dans une table
- Une confirmation a eu lieu
- Ordres de base de données (SQL)

- L3 : confirmation automatique et implicite ; pas d'annulation

# Exemple 1

- table T
- A B
- 1 2
- insert into t values(3,4)
- 1 2
- 3 4
- Confirmation
- Déconnexion
- Reconnexion
- 1 2
- 3 4

# Exemple 2

- table T
- A B
- 1 2
- insert into t values(3,4)
- A B
- 1 2
- 3 4
- Déconnexion sans confirmation
- Reconnexion
- A B
- 1 2

# Exemple 3

- table T
- A B
- 1 2
- insert into t values(3,4)
- A B
- 1 2
- 3 4
- Annulation
- Déconnexion
- Reconnexion
- A B
- 1 2

# Ordres de base de données fournis par le SGBD

- Confirmation : commit
- Annulation : rollback
- Questions :
- Y en a-t-il d'autres ?
- Confirmation et annulation : jusqu'où ?
- Réponse :
- TD 1 : expérimentation empirique
- Cours 2 : cas général

# Compétences à acquérir

- “Analyse” : savoir dire ce qui se trouve dans la base à la suite d’une séquence d’ordres BD
- ”Construction” : savoir écrire les ordres adéquats pour faire persister les données voulues
- Cas général sera vu au cours 2
- Motivation : comparaison avec les fichiers Java

# Plan pbs BD

- Modèle de données
- Conception
- Mise à jour
- Interrogation
- Contraintes d'intégrité
- Indépendance des niveaux
- Persistance
- Confidentialité
- Reprise sur panne
- Contrôle de concurrence

- Grandes quantités
- Répartition

# Confidentialité : rappel du problème

- La situation : Un client veut modifier
- L'horaire d'un train
- Les réservation d'autres clients
- Il faut :
- Un client ne doit pas pouvoir, mais un employé si
- Cas général :
- N'importe qui
- Ne doit pas pouvoir faire n'importe quoi
- Sur n'importe quelles données

# Analyse du problème

- Exemple : client toto exécute :
- Update train
- Set nom = 'toto'
- Where nom = 'titi' and dest = 'St Trop'
- Problème : Le triplet
- Utilisateur
- Action
- Objet
- Contredit le cahier des charges de la SNCF

# Protection

- Protéger l'entrée dans le système
- Une fois dedans, limiter les accès :
- Catégories de pouvoir
- Autorisations locales

# Confidentialité

- But :
- Protéger les données d'accès intempestifs
- Autoriser certaines accès
- Outils de gestion fournis par le SGBD :
- Notions de :
- Utilisateur (connexion que si déclaré)
- Catégories de pouvoirs
- Autorisations locales à chaque table
- Autorisation = (utilisateur, action, objet)
- Tout ce qui n'est pas autorisé est interdit

# Catégories de pouvoir

- Tout faire : "DBA"
- Mises à jour : "ressource"
- Lectures seules : "connect"
- Tout faire
- En particulier déclarer utilisateurs ("créer comptes")
- Ordre SQL :
- Grant dba to waller identified by waller
- Grant ressource to miagiste identified by miagiste
- Grant connect to pdg identified by pdg

# Catégories de pouvoir (2)

- Catégorie "Mise à jour"
- Actions autorisées :
- Création de nouvelles tables
- Sont alors possédées par soi-mme
- Modifications de tables existantes
- Tout sur ses propres tables
- Sur celles des autres si autorisation

# Catégories de pouvoir (3)

- Catégorie "Lecture seule"
- Actions autorisées :
- Requtes sur tables et vues existantes
- Tout sur ses propres vues
- Sur tables et vues des autres si autorisation
- Création de nouvelles vues
- Sont alors possédées par soi-mme

# Ordres SQL (1)

- Le problème 5 est résolu par le programmeur de l'application (tutu) qui tape les ordres SQL :
- Grant select
- On train
- To toto /* client */
- Grant select, update
- On réservation
- To titi /* employé */

# Ordres SQL (2)

- Revoke update
- On réservation
- From titi

# Ordres SQL (3)

- Tutu tape :
- Grant select
- On train
- To titi
- With grant option
- Titi tape :
- Grant select
- On tutu.train
- To tata

# Ordres SQL (4)

- Grant /* privilèges */
- all
- Parmi : select, insert, update, delete, alter, index
- On /* table */
- [ utilisateur . ] t
- To /* utilisateur(s) */
- toto
- public
- [ with grant option ]

# Notion de rôle

- = ensemble de couples (table, privilège) : autorisations
- Create rôle r
- Grant p
- On t
- To r
- Grant r
- To toto

# En général

- Privilège : sur Objet :
- Table
- Colonne
- Procédure
- Etc.
- Pouvoir :
- Création de table (ressource)
- Création de procédure
- Etc.

# Exemple

- create role sgbd_m1;
- grant create session to sgbd_m1;
- grant create table to sgbd_m1;
- grant create role to sgbd_m1;
- grant create trigger to sgbd_m1;
- grant create procedure to sgbd_m1;
- create user titi identified by titi;
- grant sgbdm1 to titi;
- commit;

# Compétences à acquérir

- "Analyse" : savoir dire ce que fait une séquence d'ordres SQL dans un contexte d'autorisations donné
- "Construction" : accorder les droits adéquats à une situation donnée
- Motivation : comparaison avec les fichiers Unix

# Plan pbs BD

- Modèle de données
- Conception
- Mise à jour
- Interrogation
- Contraintes d'intégrité
- Indépendance des niveaux
- Persistance
- Confidentialité
- Reprise sur panne
- Contrôle de concurrence

- Grandes quantités
- Répartition

# Plan du cours 1

- logistique
  - organisation du module
  - documents cours 1
- introduction : le domaine des bases de données
  - du titre...
  - exemple
  - les problèmes de bases de données
  - vocabulaire
  - les modules de BD à Orsay
  - but du module
  - plan du module

  - plan du cours 1
- gérer les problèmes de base de données
  - rappels de L3
    * le modèle relationnel
    * conception
    * interrogation
    * mises à jour
    * contraintes d'intégrité, séquences
    * indépendance des niveaux et vues
    * utilisation concrète en TD
  - persistance
  - confidentialité
- présentation du projet

# Présentation du projet

- Commerce électronique en contexte économique (une introduction)
- Gestion activités entreprise au-dessus SGBD
- Interaction commerciale entre entreprises par circulation de :
- Produits
- Argent

# Deux dimensions

- Interface minitel : Gestion tous problèmes BD "en solo"
- Microcosme économique
- Chaque entreprise :
- Achète
- Vend
- Commandes, livraisons, virements : électronique
- 2 banques par groupe de TD
- Toutes finances passent par banque
- Chambre de commerce : vue d'ensemble

# But du projet

- Comprendre et gérer les problèmes BD
- Préparer l'examen

# Partie 1

- A rendre : vendredi 23 septembre 2005 avant 9h à votre chargé de TD par mail 2 pages max fichier PDF obligatoire
- Contenu :
- Présentation générale, fonctionnalités, utilisateurs
- tout matériel cours 1
- Réfléchir pour TD 1 (demain) :
- Choix entreprise, banque, CCI
- Choix des binômes
- Enoncé exercice TD d'introduction au projet

# Prochain cours