Bases de données

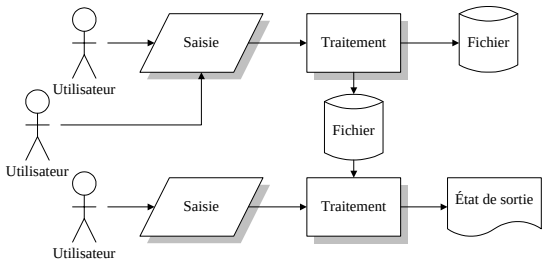
Master/Magister Informatique et Statistique
Année 2006-2007
Jérôme Darmont

http://eric.univ-lyon2.fr/~jdarmont/

Plan du cours

- Introduction
- Modèle UML
- Modèle relationnel
- Langage SQL

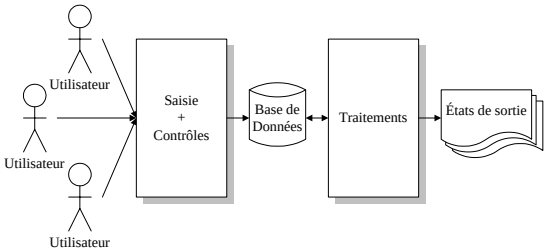
Organisation en fichiers



Limites des systèmes à fichiers

- Particularisation de la saisie et des traitements en fonction des fichiers
  ⇒ un ou plusieurs programmes par fichier
- Contrôle en différé des données
  ⇒ augmentation des délais et du risque d'erreur
- Particularisation des fichiers en fonction des traitements
  ⇒ grande redondance des données

Organisation base de données



Avantages de l'organisation BD

- Uniformisation de la saisie et standardisation des traitements (ex. tous les résultats de consultation sous forme de listes et de tableaux)
- Contrôle immédiat de la validité des données
- Partage de données entre plusieurs traitements
  ⇒ limitation de la redondance des données

Définitions

- Base de données (BD) : Collection de données cohérentes et structurées
- Système de Gestion de Bases de Données (SGBD) : Logiciel(s) assurant structuration, stockage, maintenance, mise à jour et consultation des données d'une BD

Propriétés de l'organisation BD

- Usage multiple des données
- Accès facile, rapide, protégé, souple, puissant
- Coût réduit de stockage, de mise à jour et de saisie
- Disponibilité, exactitude, cohérence et protection des données ; non redondance
- Évolution aisée et protection de l'investissement de programmation
- Indépendance des données et des programmes
- Conception *a priori*

Objectifs des SGBD

- Indépendance physique : un remaniement de l'organisation physique des données n'entraîne pas de modification dans les programmes d'application (traitements)
- Indépendance logique : un remaniement de l'organisation logique des fichiers (ex. nouvelle rubrique) n'entraîne pas de modification dans les programmes d'application non concernés

Objectifs des SGBD

- Manipulation facile des données : un utilisateur non-informaticien doit pouvoir manipuler simplement les données (interrogation et mise à jour)
- Administration facile des données : un SGBD doit fournir des outils pour décrire les données, permettre le suivi de ces structures et autoriser leur évolution (tâche de l'administrateur de BD)

Objectifs des SGBD

- Efficacité des accès aux données : garantie d'un bon débit (nombre de transactions exécutées par seconde) et d'un bon temps de réponse (temps d'attente moyen pour une transaction)
- Redondance contrôlée des données : diminution du volume de stockage, pas de mise à jour multiple ni d'incohérence

Objectifs des SGBD

- Cohérence des données : ex. L'âge d'une personne doit être un nombre entier positif. Le SGBD doit veiller à ce que les applications respectent cette règle (contrainte d'intégrité).
- Partage des données : utilisation simultanée des données par différentes applications
- Sécurité des données : les données doivent être protégées contre les accès non autorisés ou en cas de panne

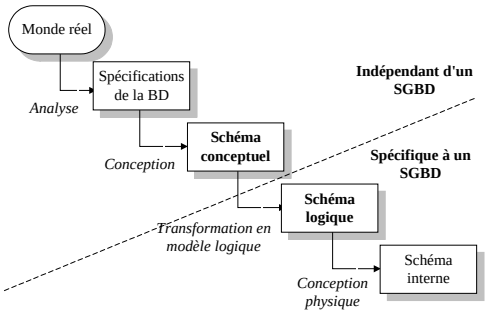
Fonctions des SGBD

- Description des données : Langage de Définition de Données (LDD)
- Recherche des données
- Mise à jour des données
- Transformation des données

} Langage de Manipulation de Données (LMD)

- Contrôle de l'intégrité des données
- Gestion de transactions et sécurité

Processus de conception d'une BD



Plan du cours

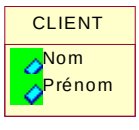
- ✓ Introduction
- Modèle UML
- Modèle relationnel
- Langage SQL

Généralités

- UML : *Unified Modeling Language*
- Ensemble de formalismes graphiques pour la modélisation orientée objet (analyse)
- Auteurs : Rumbaugh, Booch, Jacobson
- Standard de l'OMG (*Object Management Group*) depuis 1997, soutenu par de nombreux éditeurs de logiciels
- Mise en œuvre d'une BD : transformation d'un diagramme de classes UML en schéma logique

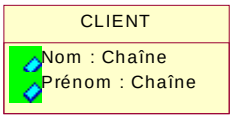
Classes et attributs

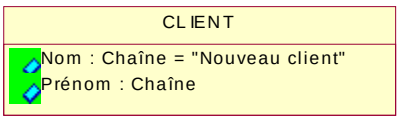
- Classe : Groupe d'entités du monde réel ayant les mêmes caractéristiques et le même comportement (ex. CLIENT)
- Attribut : Propriété de la classe (ex. Nom et Prénom du client)



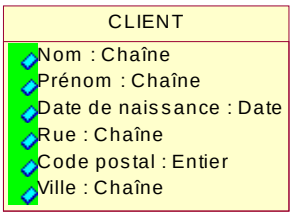
Types d'attribut

- Type d'attribut :
  - Nombre entier
  - Nombre réel
  - Chaîne de caractères
  - Date
- Valeur par défaut (=)





Exemple de classe avec ses attributs



Instances

- Classe : ex. CLIENT
- Instances (objets) de la classe CLIENT : les clients
  - Albert Dupont
  - James West
  - Marie Martin
  - Gaston Durand
  - ...

Identifiants

- Liste des clients

Nom	*Prénom*	*Date de Naissance*	*Etc.*
Dupont	Albert	01/06/70	...
West	James	03/09/63	...
Martin	Marie	05/06/78	...
Durand	Gaston	15/11/80	...
Titgoutte	Justine	28/02/75	...
Dupont	Noémie	18/09/57	...
Dupont	Albert	23/05/33	...

Problème : Comment distinguer les Dupont ?

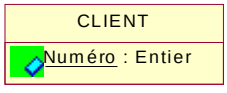
Identifiants

- Solution : Ajouter un attribut Numéro de client !

Numéro	*Nom*	*Prénom*	*Date de Naissance*
1110	Dupont	Albert	01/06/70
2002	West	James	03/09/63
3333	Martin	Marie	05/06/78
4042	Durand	Gaston	05/11/80
5552	Titgoutte	Justine	28/02/75
6789	Dupont	Noémie	18/09/57
7000	Dupont	Albert	23/05/33

Identifiants

- Le numéro de client est un attribut identifiant. Un identifiant caractérise de façon unique les instances d'une classe.
- NB : Dans le paradigme objet (OO), un objet est déjà identifié par son OID (*Object IDentifier*). La finalité de notre utilisation d'UML n'étant pas OO, nous ajouterons un attribut identifiant.
- Convention graphique :
  NB : Ne pas confondre avec les attributs de classe UML dont c'est la notation usuelle



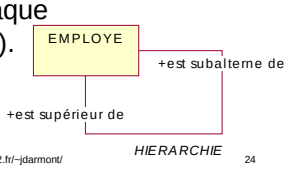
Associations

- Définition : liaison perçue entre des classes ex. Les clients commandent des produits.

- Les classes CLIENT et PRODUIT peuvent être qualifiées de participantes à l'association COMMANDE.
- Degré ou arité d'une association : nombre de classes participantes.
  En général : associations binaires (de degré 2).

Associations récursives et rôles

- Association récursive : une même instance de classe peut jouer plusieurs rôles dans la même association (ex. employés et supérieurs hiérarchiques).
- Rôle : fonction de chaque classe participante (+).

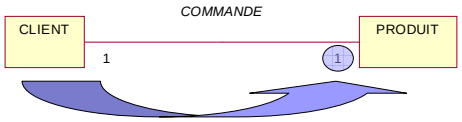


Multiplicité (ou cardinalité)

- Définition : Indicateur qui montre combien d'instances de la classe considérée peuvent être liées à une instance de l'autre classe participant à l'association
  - 1 Un et un seul
  - 0..1 Zéro ou un
  - 0..* ou * Zéro ou plus
  - 1..* Un ou plus
  - M..N De M à N (M, N entiers)

Associations « 1-1 »

- ex. Un client donné ne commande qu'un seul produit. Un produit donné n'est commandé que par un seul client.



Lire "Un client commande multiplicité (1) produit(s)".

Associations « 1-N »

- ex. Un client donné commande plusieurs produits. Un produit donné n'est commandé que par un seul client.

NB : La multiplicité un à plusieurs (1..*) peut aussi être zéro à plusieurs (0..* ou *).

Associations « 0 ou 1-N »

- ex. Un client donné commande plusieurs produits. Un produit donné est commandé au maximum par un client, mais peut ne pas être commandé.

NB : La multiplicité un à plusieurs (1..*) peut aussi être zéro à plusieurs (0..* ou *).

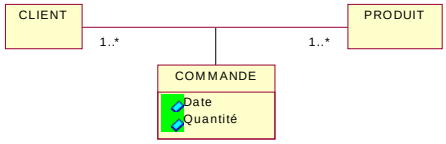
Associations « M-N »

- ex. Un client donné commande plusieurs produits. Un produit donné est commandé par plusieurs clients.

NB : Les multiplicités un à plusieurs (1..*) peuvent aussi être zéro à plusieurs (0..* ou *).

Classes-associations

- Dans une association M-N, il est possible de caractériser l'association par des attributs (qui par définition doivent appartenir à une classe).
  ex. Une commande est passée à une Date donnée et concerne une Quantité de produit fixée.



Exemple complet

Notes d'examen : Spécifications

- Les étudiants sont caractérisés par un numéro unique, leur nom, prénom, date de naissance, rue, code postal et ville.
- Ils passent des épreuves et obtiennent une note pour chacune.
- Les épreuves sont caractérisées par un code, ainsi que la date et le lieu auxquels elles se déroulent.

Exemple complet

Notes d'examen : Spécifications (suite)

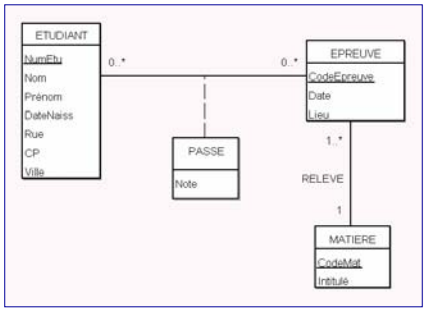
- Chaque épreuve relève d'une matière unique (mais une matière donnée peut donner lieu à plusieurs épreuves).
- Les matières sont caractérisées par un code et un intitulé.

Exemple complet

- Marche à suivre pour produire un diagramme de classes UML :

1. Identifier les classes.
2. Identifier les associations entre les classes.
3. Identifier les attributs de chaque classe et de chaque classe-association.
4. Évaluer la multiplicité des associations.

Exemple complet



Plan du cours

- ✓ Introduction
- ✓ Modèle UML
- Modèle relationnel
- Langage SQL

Généralités

- Le modèle relationnel est un modèle logique associé aux SGBD relationnels (ex. Oracle, SQL Server, DB2, MySQL, Access…).
- Objectifs du modèle relationnel :
  - Indépendance physique
  - Traitement du problème de redondance des données
  - LMD non procéduraux (faciles à utiliser)
  - Devenir un standard

Généralités

Caractéristiques des systèmes relationnels :

- Langages d'interrogation puissants et déclaratifs
- Accès orienté valeur
- Grande simplicité, absence de considérations physiques
- Description du schéma très réduite
- LDD intégré au LMD
- Grande dynamique de structure
- Optimisation de requêtes
- Utilisation interactive ou à partir d'un langage hôte

Relations et attributs

- Une relation R est un ensemble d'attributs $\{A_1, A_2, \ldots, A_n\}$.
  ex. La relation EPREUVE est l'ensemble des attributs {CodeEpreuve, Date, Lieu}
- Chaque attribut A_i prend ses valeurs dans un domaine $dom(A_i)$.
  ex. Note $\in$ [0, 20]
  Lieu $\in$ {'Amphi 136', 'Amphi 236', 'Salle 201', 'Salle 301', …}

N-uplets

- Un n-uplet t est un ensemble de valeurs $t = \langle V_1, V_2, \ldots, V_n \rangle$ où $V_i \in dom(A_i)$ ou V_i est la valeur nulle (NULL).
  ex. <'InfoS2', '30-06-2006', 'Amphi 136'> est un n-uplet de la relation EPREUVE.
- Notation : R (A_1, A_2, …, A_n)
  ex. EPREUVE (CodeEpreuve, Date, Lieu)

Contraintes d'intégrité

- Clé primaire : Ensemble d'attributs dont les valeurs permettent de distinguer les n-uplets les uns des autres (notion d'identifiant).
  ex. CodeEpreuve est clé primaire de la relation EPREUVE.
- Clé étrangère : Attribut qui est clé primaire d'une autre relation.
  ex. Connaître la matière dont relève chaque épreuve
  ⇒ ajout de l'attribut CodeMat à la relation EPREUVE

Contraintes d'intégrité

- Notations : Clés primaires soulignées, clés étrangères postfixées par le caractère #.
  ex. EPREUVE (CodeEpreuve, Date, Lieu, CodeMat#)
- Contraintes de domaine : Les attributs doivent respecter une condition logique.
  ex. Note ≥ 0 ET Note ≤ 20

Traduction UML-relationnel

1. Chaque classe devient une relation. Les attributs de la classe deviennent attributs de la relation. L'identifiant de la classe devient clé primaire de la relation.

   ex. ETUDIANT (NumEtu, Nom, Prénom, DateNaiss, Rue, CP, Ville)

Traduction UML-relationnel

2. Chaque association 1-1 est prise en compte en incluant la clé primaire d'une des relations comme clé étrangère dans l'autre relation.

   ex. Si un étudiant peut posséder une (et une seule) carte CUMUL, on aura :

   CARTE (NumCarte, Crédit, …)
   ETUDIANT (NumEtu, Nom, Prénom, DateNaiss, Rue, CP, Ville, NumCarte#)

Traduction UML-relationnel

3. Chaque association 1-N est prise en compte en incluant la clé primaire de la relation dont la multiplicité maximale est 1 comme clé étrangère dans l'autre relation.

   ex.
   EPREUVE (CodeEpreuve, Date, Lieu, CodeMat#)
   MATIERE (CodeMat, Intitulé)

Traduction UML-relationnel

4. Chaque association M-N est prise en compte en créant une nouvelle relation dont la clé primaire et la concaténation des clés primaires des relations participantes. Les attributs de la classe-association sont insérés dans cette nouvelle relation si nécessaire.

   ex. PASSE (NumEtu#, CodeEpreuve#, Note)

Traduction UML-relationnel

- Schéma relationnel complet de l'exemple

  ETUDIANT (NumEtu, Nom, Prénom, DateNaiss, Rue, CP, Ville)

  EPREUVE (CodeEpreuve, Date, Lieu, CodeMat#)

  MATIERE (CodeMat, Intitulé)

  PASSE (NumEtu#, CodeEpreuve#, Note)

Problème de la redondance

- [En dehors des clés étrangères]

ex. Soit la relation COMMANDE_PRODUIT.

NumProd	*Quantité*	*NumFour*	*Adresse*
101	300	901	Quai des brumes
104	1000	902	Quai Claude Bernard
112	78	904	Quai des Marans
103	250	901	Quai des brumes

Cette relation présente différentes anomalies.

Anomalies liées à la redondance

- Anomalies de modification : Si l'on souhaite mettre à jour l'adresse d'un fournisseur, il faut le faire pour tous les n-uplets concernés.
- Anomalies d'insertion : Pour ajouter un fournisseur nouveau, il faut obligatoirement fournir des valeurs pour NumProd et Quantité.
- Anomalies de suppression : Ex. La suppression du produit 104 fait perdre toutes les informations concernant le fournisseur 902.

Normalisation

Objectifs :

- Suppression des problèmes de mise à jour
- Minimisation de l'espace de stockage (élimination des redondances)

Dépendances fonctionnelles

- Soit R (X, Y, Z) une relation où X, Y, et Z sont des ensembles d'attributs. Z peut être vide.
- Définition : *Y dépend fonctionnellement de X* ($X \rightarrow Y$) si c'est toujours la même valeur de Y qui est associée à X dans la relation R.
- ex. PRODUIT (NumProd, Dési, Prix)
  DF possibles : NumProd $\rightarrow$ Dési
  Dési $\rightarrow$ Prix

Propriétés des DF

Règles d'inférence d'Armstrong

- Réflexivité :
  Si $Y \subseteq X$ alors $X \rightarrow Y$.
- Augmentation :
  Si $W \subseteq Z$ et $X \rightarrow Y$ alors $X, Z \rightarrow Y, W$.
- Transitivité :
  Si $X \rightarrow Y$ et $Y \rightarrow Z$ alors $X \rightarrow Z$.

Propriétés des DF

- Pseudo-transitivité :
  Si $X \rightarrow Y$ et $Y, Z \rightarrow T$ alors $X, Z \rightarrow T$.
- Union :
  Si $X \rightarrow Y$ et $X \rightarrow Z$ alors $X \rightarrow Y, Z$.
- Décomposition :
  Si $Z \subseteq Y$ et $X \rightarrow Y$ alors $X \rightarrow Z$.

- NB : La notation X, Y signifie $X \cup Y$.

Première forme normale

- *Une relation est en 1FN si tout attribut n'est pas décomposable.*

- ex. Les relations PERSONNE (Nom, Prénoms, Age) et DEPARTEMENT (Nom, Adresse, Tel) ne sont pas en 1FN si les attributs Prénoms et Adresse peuvent être du type [Jean, Paul] ou [Rue de Marseille, Lyon].

Deuxième forme normale

- *Une relation est en 2FN si :*
  - *elle est en 1FN ;*
  - *tout attribut non clé primaire est dépendant de la clé primaire entière.*
- ex. La relation CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville) est en 2FN.

Deuxième forme normale

- ex. La relation COMMANDE_PRODUIT (NumProd, Quantité, NumFour, Ville) n'est pas en 2FN car on a NumProd, NumFour → Qté et NumFour → Ville.
- La décomposition suivante donne deux relations en 2FN :

  COMMANDE (NumProd, NumFour, Quantité)
  FOURNISSEUR (NumFour, Ville).

Troisième forme normale

- *Une relation est en 3FN si :*
  - *elle est en 2FN ;*
  - *il n'existe aucune DF entre deux attributs non clé primaire.*
- ex. La relation MUSIQUE (NoChanson, NoChanteur, Nom) avec les DF NoChanson → NoChanteur, NoChanteur → Nom et NoChanson → Nom est en 2FN, mais pas en 3FN.

Troisième forme normale

- Anomalies de mise à jour sur la relation MUSIQUE : il n'est pas possible d'introduire un nouveau chanteur sans préciser la chanson correspondante.
- La décomposition suivante donne deux relations en 3FN qui permettent de retrouver (par transitivité) toutes les DF :

  R1 (NoChanson, NoChanteur) ;
  R2 (NoChanteur, Nom).

Algèbre relationnelle

- Ensemble d'opérateurs qui s'appliquent aux relations
- Résultat : nouvelle relation qui peut à son tour être manipulée

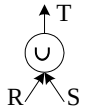
⇒ L'algèbre relationnelle permet d'effectuer des recherches dans les relations.

Opérateurs ensemblistes

- Union : T = R ∪ S ou T = UNION (R, S)
  R et S doivent avoir même schéma.

  ex. R et S sont les relations PRODUIT de deux sociétés qui fusionnent et veulent unifier leur catalogue.

  Notation graphique :

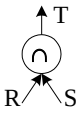


Opérateurs ensemblistes

- Intersection : T = R ∩ S ou
  T = INTERSECT (R, S)
  R et S doivent avoir même schéma.

  ex. Permet de trouver les produits communs aux catalogues de deux sociétés.

  Notation graphique :

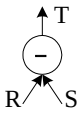


Opérateurs ensemblistes

- Différence : T = R - S ou T = MINUS (R, S)
  R et S doivent avoir même schéma.

  ex. Permet de retirer les produits de la relation S existant dans la relation R.

  Notation graphique :

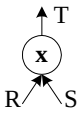


Opérateurs ensemblistes

- Produit cartésien : T = R x S
  ou T = PRODUCT (R, S)

  Associe chaque n-uplet de R à chaque n-uplet de S.

  Notation graphique :



Produit cartésien

ex.

NumProd	*Dési*
0	P1
1	P2

X

NumFour	*RaisonSoc*
10	F1
20	F2
30	F3

=

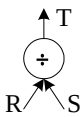
NumProd	*Dési*	*NumFour*	*RaisonSoc*
0	P1	10	F1
1	P2	10	F1
0	P1	20	F2
1	P2	20	F2
0	P1	30	F3
1	P2	30	F3

Opérateurs ensemblistes

- Division : T = R ÷ S ou T = DIVISION (R, S)

  R (A_1, A_2, …, A_n) S (A_{p+1}, …, A_n)
  T (A_1, A_2, …, A_p) contient tous les n-uplets tels que leur concaténation à chacun des n-uplets de S donne toujours un n-uplet de R.

  Notation graphique :



Division

ex.

NumCli	*Date*	*Quantité*
1	22/09/99	1
1	22/09/99	5
1	10/10/99	2
2	15/10/99	9
3	22/09/99	1
3	10/10/99	2

÷

Date	*Quantité*
22/09/99	1
10/10/99	2

=

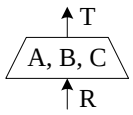
NumCli
1
3

Opérateurs spécifiques

- Projection : T = Π <A, B, C> (R)
  ou T = PROJECT (R / A, B, C)

  T ne contient que les attributs A, B et C de R.
  ex. Noms et prénoms des clients.

  Notation graphique :

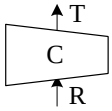


Opérateurs spécifiques

- Restriction : T = σ <C> (R)
  ou T = RESTRICT (R / C)

  T ne contient que les attributs de R qui satisfont la condition C.
  ex. C = Clients qui habitent Lyon.

  Notation graphique :

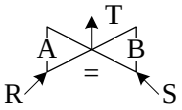


Opérateurs spécifiques

- Jointure naturelle : T = R >< S
  ou T = JOIN (R, S)

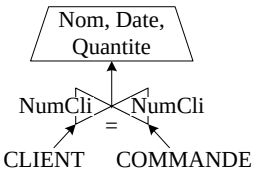
  Produit cartésien R x S et restriction A = B sur les attributs A ∈ R et B ∈ S.

  Notation graphique :



Exemple de requête*

Commandes avec le nom du client et pas seulement son numéro



*Requête : enchaînement d'opérations

Exemple de requête

Décomposition des opérations

CL

NumCli	*Nom*
1	C1
2	C2
3	C3

X

CO

NumCli	*Date*	*Quantité*
1	22/09/99	1
3	22/09/99	5
3	22/09/99	2

Exemple de requête

=

CL.NumCli	*Nom*	*CO.NumCli*	*Date*	*Quantité*
1	C1	1	22/09/99	1
2	C2	1	22/09/99	1
3	C3	1	22/09/99	1
1	C1	3	22/09/99	5
2	C2	3	22/09/99	5
3	C3	3	22/09/99	5
1	C1	3	22/09/99	2
2	C2	3	22/09/99	2
3	C3	3	22/09/99	2

Exemple de requête

CL >< CO

CL.NumCli	*Nom*	*CO.NumCli*	*Date*	*Quantité*
1	C1	1	22/09/99	1
3	C3	3	22/09/99	5
3	C3	3	22/09/99	2

Nom	*Date*	*Quantité*
C1	22/09/99	1
C3	22/09/99	5
C3	22/09/99	2

Π <Nom, Date, Quantité> (CL >< CO)

(Projection sur les attributs Nom, Date, Quantité)

Classification des SGBD relationnels

- Niveau 1 : Systèmes non relationnels. Supportent uniquement la structure tabulaire.
- Niveau 2 : Systèmes relationnellement minimaux. Permettent les opérations de sélection, projection et jointure.
- Niveau 3 : Systèmes relationnellement complets. Toutes les opérations de l'algèbre relationnelle.
- Niveau 4 : Systèmes relationnellement pleins. Permettent la définition des contraintes d'intégrité.

Plan du cours

- ✓ Introduction
- ✓ Modèle UML
- ✓ Modèle relationnel
- Langage SQL

Généralités

- SQL : *Structured Query Language,* issu de SEQUEL (*Structured English as a QUery Language*)
- SQL permet la définition, la manipulation et le contrôle d'une base de données relationnelle. Il se base sur l'algèbre relationnelle.
- SQL est un standard depuis 1986.
- Nous adoptons dans ce chapitre la syntaxe du SQL d'Oracle (très proche de la norme).

Types de données principaux

- NUMBER(n) : nombre entier à n chiffres
- NUMBER(n, m) : nombre réel à n chiffres au total (virgule comprise) et m chiffres après la virgule
- VARCHAR(n) : chaîne de caractères de taille n
- DATE : date au format 'JJ-MM-AAAA'

Contraintes d'intégrité

- Mot clé CONSTRAINT
- Identification par un nom de contrainte
- Clé primaire : PRIMARY KEY (clé)
- Clé étrangère : FOREIGN KEY (clé) REFERENCES table(attribut)
- Contrainte de domaine : CHECK (condition)

Définition des données

ex.
```
CREATE TABLE Client (   NumCli NUMBER(8),
                        Nom VARCHAR(1000),
                        DateNaiss DATE,
                        Salaire NUMBER(8,2),
                        NumEmp NUMBER(3),

CONSTRAINT Cle_pri PRIMARY KEY (NumCli),
CONSTRAINT Cle_etr FOREIGN KEY (NumEmp)
   REFERENCES Employeur(NumEmp),
CONSTRAINT Sal_ok CHECK (Salaire >= 1286.09)
                                        );
```


Modifications structurelles

- Ajout d'attributs
  ALTER TABLE nom_table ADD (attribut TYPE, …);
  ex. ALTER TABLE Client ADD (tel NUMBER(8));
- Modifications d'attributs
  ALTER TABLE nom_table MODIFY (attribut TYPE, …);
  ex. ALTER TABLE Client MODIFY (tel NUMBER(10));
- Suppression d'attributs
  ALTER TABLE nom_table DROP COLUMN attribut, ...;
  ex. ALTER TABLE Client DROP COLUMN tel;

Modifications structurelles

- Ajout de contrainte
  ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte définition_contrainte;
  ex. ALTER TABLE Client
  ADD CONSTRAINT sal_ok CHECK (salaire > 0);
- Suppression de contrainte
  ALTER TABLE nom_table DROP CONSTRAINT nom_contrainte;
  ex. ALTER TABLE Client
  DROP CONSTRAINT sal_ok;

Index

- Définition : Structure de données physique permettant d'accélérer les accès aux données
- Exemple :
  CREATE INDEX Idx_etu ON Etudiant (Nom);
- NB : La clé primaire d'une relation est automatiquement indexée.

Mise à jour des données

- Ajout d'un n-uplet
  ex. INSERT INTO Produit
  VALUES (400, 'Nouveau produit', 78.90);
- Modification de la valeur d'un attribut
  ex. UPDATE Etudiant SET Nom='Dudule'
  WHERE NumEtu = 333333;
- Suppression de n-uplets
  ex. DELETE FROM Etudiant
  WHERE Ville = 'Lyon';

Interrogation des données

- Par l'exemple, sur la base ETUDIANTS

  ETUDIANT (NumEtu, Nom, Prénom, DateNaiss, Rue, CP, Ville)

  EPREUVE (CodeEpreuve, Date, Lieu, CodeMat#)

  MATIERE (CodeMat, Intitulé)

  PASSE (NumEtu#, CodeEpreuve#, Note)

Étoile, tri et champs calculés

- Tous les n-uplets d'une table : étoile (*)
  ex. SELECT * FROM Etudiant;
- Tri du résultat
  ex. *Par ordre alphabétique [inverse] de nom*
  SELECT * FROM Etudiant
  ORDER BY Nom [DESC];
- Champs calculés
  ex. *Transformation de notes sur 20 en notes sur 40*
  SELECT Note * 2 FROM Passe;

Projection et restriction

- Projection
  ex. *Noms et Prénoms des étudiants, uniquement (pas les autres attributs)*
  SELECT Nom, Prénom FROM Etudiant;
- Suppression des doublons
  ex. SELECT DISTINCT Nom FROM Etudiant;
- Restriction
  ex. *Étudiants qui habitent à Lyon*
  SELECT * FROM Etudiant
  WHERE Ville = 'Lyon';

Opérateurs de restriction

ex. *Épreuves se déroulant après le 01/01/2006*
SELECT * FROM Epreuve
WHERE Date >= '01-01-2006';

ex. *Notes comprises entre 10 et 20*
SELECT * FROM Passe
WHERE Note BETWEEN 10 AND 20;

ex. *Notes indéterminées (sans valeur)*
SELECT * FROM Passe
WHERE Note IS NULL;

Opérateurs de restriction

ex. *Étudiants habitant une ville dont le nom se termine par* sur-Saône

SELECT * FROM Etudiant
WHERE Ville LIKE '%sur-Saône';

'sur-Saône%' ⇒ commence par *sur-Saône*
'%sur%' ⇒ contient le mot *sur*

Opérateurs de restriction

ex. *Prénoms des étudiants dont le nom est Dupont, Durand ou Martin*

SELECT Prénom FROM Etudiant
WHERE Nom IN ('Dupont', 'Durand', 'Martin');

NB : Possibilité d'utiliser la négation pour tous ces prédicats : NOT BETWEEN, NOT NULL, NOT LIKE, NOT IN.

Fonctions d'agrégat

- Elles opèrent sur un ensemble de valeurs.
- AVG(), VARIANCE(), STDDEV() : moyenne, variance et écart-type des valeurs
- SUM() : somme des valeurs
- MIN(), MAX() : valeur minimum, valeur maximum
- COUNT() : nombre de valeurs
- ex. *Moyenne des notes*
  SELECT AVG(Note) FROM Passe;

Fonction COUNT et opérateur DISTINCT

ex. *Nombre total de notes*
SELECT COUNT(*) FROM Passe;
SELECT COUNT(NumEtu) FROM Passe;

ex. *Nombre d'étudiants notés*
SELECT COUNT(DISTINCT NumEtu)
FROM Passe;

Exemple

Table PASSE

NumEtu	*CodeEpreuve*	*Note*
1000	InfoS2	13.0
3000	EcoS1	12.5
3000	InfoS1	15.0

COUNT(NumEtu) ⇒ Résultat = 3
COUNT(DISTINCT NumEtu) ⇒ Résultat = 2

Jointure

ex. *Liste des notes avec le nom des étudiants*

SELECT Nom, Prénom, Note
FROM Etudiant, Passe
WHERE Etudiant.NumEtu = Passe.NumEtu;

Jointure

ex. *Idem avec le numéro d'étudiant en plus*

SELECT E.NumEtu, Nom, Prénom, Note
FROM Etudiant E, Passe P
WHERE E.NumEtu = P.NumEtu
ORDER BY Nom, Prénom;

NB : Utilisation d'alias (E et P) pour alléger l'écriture + tri par nom et prénom.

Jointure

Jointure exprimée avec le prédicat IN

ex. *Notes des épreuves passées le 23 septembre 2006*

SELECT Note FROM Passe
WHERE CodeEpreuve IN (
    SELECT CodeEpreuve FROM Epreuve
    WHERE Date = '23-09-2006');

NB : Il est possible d'imbriquer des requêtes.

Prédicats d'existence

- Prédicats EXISTS / NOT EXISTS

ex. *Étudiants qui ont passé au moins une épreuve [n'ont passé aucune épreuve]*

SELECT * FROM Etudiant E
WHERE [NOT] EXISTS (
    SELECT * FROM Passe P
    WHERE E.NumEtu = P.NumEtu);

Prédicats de dénombrement

- Prédicats ALL / ANY

ex. *Numéros des étudiants qui ont obtenu au moins une note supérieure à chacune [à au moins une] des notes obtenues par l'étudiant client n° 1000.*

```
SELECT DISTINCT NumEtu FROM Passe
WHERE Note > ALL [ANY] (
    SELECT Note FROM Passe
    WHERE NumEtu = 1000      );
```


Groupement

ex. *Moyenne de chaque étudiant*

```
SELECT NumEtu, AVG(Note)
FROM Passe
GROUP BY NumEtu;
```

ex. *Nombre de notes par étudiant*

```
SELECT NumEtu, COUNT(Note)
FROM Passe
GROUP BY NumEtu;
```


Groupement

ex. *Note moyenne pour les étudiants ayant passé moins de 5 épreuves*

```
SELECT NumEtu, AVG(Note)
FROM Passe
GROUP BY NumEtu
HAVING COUNT(*) < 5;
```

Attention : La clause HAVING ne s'utilise qu'avec GROUP BY.

NB : HAVING : évaluation de condition sur un résultat de groupement (*a posteriori*)

≠ WHERE : évaluation de condition *a priori*

Opérations ensemblistes

INTERSECT, MINUS, UNION

ex. *Code des épreuves ayant soit lieu dans l'amphi 136, soit ayant été passées par l'étudiant n° 2222*

```
SELECT CodeEpreuve FROM Epreuve
WHERE Lieu = 'Amphi 136'
UNION
SELECT CodeEpreuve FROM Passe
WHERE NumEtu = 2222;
```


Tutoriel SQL

Pour approfondir SQL en ligne…

http://eric.univ-lyon2.fr/~jdarmont/tutoriel-sql/

Plan du cours

- ✓ Introduction
- ✓ Modèle UML
- ✓ Modèle relationnel
- ✓ Langage SQL