7SI = system d'information

E = entité

A = attribues

C= cardinalité

I = identifiant

R = relation ou associations

La démarche de travail :

Cerner le problème

– Identifier les acteurs

– Identifier les flux d'information

Définir le domaine de l'étude

Etablir le dictionnaire d'information

Etablir les diagrammes de flux

Etablir les modèles conceptuels

Etablir les modèles logiques

Etablir les modèles physiques

Mise en œuvre

Faire le dictionnaire de donnée en ds ( +1 p)

**Construction du MCD (exercice) (70)**:

CM3

4 cours :

-le modèle relationnel

-SQL

-la normalisation

-bases de données rapportées

**Le modèle relationnel**

Modèle logique associé aux SGBD relationnels

Objectif :

* Indépendance physique
* Traitement des redondances
* Langage de manipulation de données
* Standard

Définition :

* Domaine :

Ensemble fini ou infini de valeurs (les valeurs d'un domaine sont atomiques)

* Relation :

Sous ensemble du produit cartésien d'une liste de Domaines. On caractérise une Relation par son nom

La relation R sur les domaines D1,…,Dn est constitué :

* D'un entête, un ensemble fixé d'attributs
* D'un corps, ensemble tuples

Entête = schéma = intention

Corps = exertion

Exemple de relation :

Vin(identVin entier, NomVin : chaine de caractères, Region : DomaineViticole, Millésime : année)

(1, Hart Médoc; Bordeau, 2005)

(2, château neuf du pape, cote du rhone, 1999)

Vocabulaire :

|  |  |
| --- | --- |
| MR | SQL |
| Relation  Tuple  Attributs  Cardinalité  Degré  Clé patrimoine | Table  Ligne  Colonne  Nb de lignes  Nb de colonnes  Identifiant unique |

Contrainte d'intégrité :

* Clé primaire : ensemble d'attributs dont les valeurs permettent de Distinguer les tuples les uns des autres. Ou clé simple est constitué d'un seul attribut.

Une clé primaire est unique et irréductible.

Contraintes d'identités :

Elles imposent que toute relation possède une clé candidate et que tout attribut participant a cette clé ait une valeur non NULL.

Contrainte référentielle :

Contrainte portant sur une relation S, consistant a imposer que la valeur d'un groupe d'attribut apparaisse comme valeur de clé candidaté dans une autre relation R.

Transformation MEA🡪Modèle Relationnel

|  |  |
| --- | --- |
| Transformation d'une Entité | Transformation d'une association |
| Une entité 🡪 une relation  -les propriétés de l'entité deviennent les attributs de la relation  -l'identifiant devient clé primaire | Pour des associations (O ou 1-n)(O ou 1-n) l'association devient une relation R  Les attributs de r = les clés primaires des entités associées + propriétés de l'association.  Clé de R : l'union clés primaires des entités associés |

Pour les assiciation (1 1)( 0 ou 1-n)

0-n

1

A

E2

E1

E1🡪R1

E2🡪R2

Clé de R2 devient clé étrangere de R1

Les propriétés de l'assiociation migrent avec la clé

Pour manipuler le modèle relationnel, on utilise l'algèbre relationnelle

Algèbre relationnelle (AR):

* Introduit en 1970 : Cadd
* L'AR est constitué d'opérations agissant sur des relations et produisant d'autres relations.
* Restriction
* Projection
* Produit
* Union
* Intersection
* Difference
* Jointure
* Division

**I/Opération ensemblistes :**

* Sur des relations de meme schéma :
  + Union de 2 relations R et S est une relation T contenant l'ensemble des tuples de R et des tuples de S on note RUS
  + Intersection de 2 relations R et S est une relation T contenant l'ensemble des tuples qui appartiennent que 2 relations. On note R∩S
  + la différence de 2 relations R et S est une relation T qui contient l'ensemble des tuples de R qui n'appartiennent pas à S. on note R\S
* sur des relations n'ayant pas le meme schema :
  + le produit cartésien de 2 relations R et S est une relation T ayant pour attributs ke concaténation des attributs de R et de S et dont les tuples sont constitués de toutes les concaténations d'un tuple de R à un tuple de S. on note R X S

R1 {(1,a,bleu),(4,c,rouge),(18,v,vert)}

R2{(Eric, Dupont,432),(Florent, Dupuis,1515),(Louis, Durand, 2024)}

**II/Opération spécifiques :**

1. Affectation :

Sauvegarde du résultat d'une expression de recherche ou renommage d'une relation. On note R🡨T

1. Restriction de la relation R par une qualification Q est une relation T constituée de tous les tuples de R qui satisfait Q. on note 6Q R
2. Projection d'une relation R(A1,…,An) sur les attributs A1… Ap (avec p <=n) est une relation T de schema (A1…Ap) dont les tuples sont obtenus par elimination des attributs de R n'apportant pas à T et par suppréssion des doubles. On note πA1…Ap R

**III/les opérations de jointure :**

1. La ϴ jointure de R et S. selon une qualification Q est l'ensemble des tuples de R X S qui satisfait les qualification Q. on note R|X|S

Q

1. L'équi jointure de la relation R et S est un ϴ jointure avec pour critère de qualification un test d'égalité entre 2 attributs A et B. on note R |X| S

A=B

1. la jointure naturelle entre R et S est une equi jointure sur tous les attributs de R et S ayant le meme Nom, suivie d'une projection permettant de Ne garder qu'un suel de ces attributs de meme nom. Pon note R (X) S
2. la Division de R(A1…An) par la relation S(A1… Ap) (p<=n) est une relation T (Ap+1…An) formée de tous les tuples qui concaténés a chaque tuple de S donne toujours un tuple de R. on note R/S
3. opération de calcul
   1. compte(R) : dénomble le nombre de lignes de R
   2. SonneA(R) : fait la somme cumulée sur l'attribut A de Relation R
   3. Min, Max, MoyenneA(R) retourne le min, max, la moyenne de l'attributnA de la relation R.

Patient(N\*55, nomP, prenomP, telP, adresseP, villeP,#codeM)

Docteur(CodeD, nomD, prenomD, telD, adresseD, villeD)

Mutuelle(CodeM, nomM, telM, adresseM, villeM)

Hospitalise(N\*55,codeD, idS, codeA, date)

Service(idS, denomination, nom Hopital, telS, villeS)

Affectation(codeA, nomA, dangetosité)

1) donner le nom de tous les patients 🡪 (πnomP (Patient))

2) donner le nom des docteurs de reims 🡪 πnomD (6villeD=Reims(docteur))

3) donner les numéros de téléphone des services cardiaques de Reims 🡪 πtelS (6villeS=Reims (Service)a denomination= Cardiaque)

4) donner le nom des mutuelles contractés par des patients de reims 🡪 πNomM(6villeP=Reims(Patient |X| Mutuelle))

5) le nom des patients ayant été hospitalisé dans un service de neurologie par un docteur de Reims avant le 16/10/2015

R51🡨πcodeD (6villeD=Reims(Docteur))

R52🡨 πidS(6Denomination=Neurologie(Service))

R53🡨 (6date<16/10/2015(Hospitalisation))

R54🡨 ((R53 |X| R52)|X| R51)

R55🡨 πnomP(Patient |X| R54)

6) le nom des docteurs ayant hospitalisé un patient qui cotise une mutuelle qui a été contractée par des patients qui ont été hospitalisés a reims pour une affection grave.

R61🡨 πcadeA(6dangerosité = grave affectation)

R62🡨 πidS (6villeS=Reims Service)

R63🡨 πN\*55((Hospitalise |X| R62) |X| R61)

R64🡨 πcodeM(Patient |X|R63)

R65🡨 πn\*SS(Patient |X|R64)

R6 🡨πnomD(Docteur|X|(Hospitalise|X|R65)

**Structure Query Language (SQL)**

Standard

Historique:

1970 : codd, modele relationnel

1974 : sequel2(IBM)

1981 : oracle (SQL)

1986: SQL/86 normalisé

…

2011 : SQL5

SQL c'est :

* Un langage de manipulation de données
* Un langage de definition de données
* Un langage de contrôle de données

Une requete SQL peut etre :

* Interactive
* Incluse dans un programme d'application

🡪Manipulation de données:

Syntaxe minimale :

Select <Liste d'attributs projetés>

From <Listes de Relations>

On enrichit cette syntaxe a l'aide de Clauses qui permettent d'exprimer :

* Les projections
* Les restrictions
* Les jointures
* Les produits cartésiens
* Les agrégats
* Les tris
* …

La projection :

S'exprime via la liste des attributs projets

Les noms des docteurs

ΠnomD Docteur

Select NomD From Docteur

Nom prenom des patients

ΠnomP prenomP Patient

Select NomP PrenomP From Patient

Projection :

Si on souhaite projeter sur tous les attributs de la relation on utilise \*

Affection :

AR : Affection

SQL : Select \* From Affection

Note :

Contrairement à l'agèbre delationnel, il n'y a pas de suppression des doublons pour supprimer les doublons, on utilise **Distinct**

Expression d'une restriction : la restriction s'exprime en utilisant la clause **where**

Ex:

Les medecins de reims

AR: 6ville=Reims

SQL : Select \* From Docteur where VilleD like 'Reims'

Les mots clé suivant peuvent etre utilisés dans une qualification de restriction :

* **IN**
* **LIKE**
* **NOT**
* **NULL**
* **Opérateurs de comparaison arithmétique**
* **AND**
* **OR**
* **BETWEEN**
* **…**

Les medecins de reims ou de paris

Select \* From Docteur where villeD like 'Reims' OR villeD like 'Paris'

Select \* From Docteur where villeD IN ('Reims', 'Paris')

Le nom et prenom du patient 432

Select NomP, PrenomP From Patient where n\*SS=432

Le nom des mutuelles qui ne sont pas à Reims ni à Paris

Select nomMut From Mutuelle where Not (villeM IN('Reim','Paris'))

Expression d'un Tri :

L'expression d'un tri s'effectue par le mot cké ORDER BY

Exemple : Le nom des patients de Reims trié par ordre alphabétique

Select NomP from Patient where villeP like 'Reims' ORDER BY nomP

Présentation des colonnes

La présentation des colonnes s'exprime avec le mot clé AS

Select att1 AS nom1, att2 As nom2 From

Jointures

* Produit cartésien R1xR2

Select \* From R1 , R2

* **(!a ne pas utiliser !)** Jointure R1|X|R2

NB

Select \* from R1 R2 where R1.NB=R2.NB

**(a utiliser)** Jointure naturelle

Select \* from R1 naturalJoin R2 USING NR

Exemple :

Nombre d'hospitalisation du patient 413

Select Count(\*) from hospitalisation where n\*SS=413

Le N SS maximal d'un patient hospitalisé dans un service à Reims

**Attributs calculés**

COUNT() 🡪 denombrement

AVG() 🡪 moyenne

MAX() 🡪 maximum

MIN() 🡪 minimum

SUM() 🡪 somme cumulée

…

AR :

MAXnSS(hospitalisarion |X|(6villeS=Reims(service)))

SQL :

Select Max(NSS) from hospitalisation Natural Join Service USING idS where Service.VilleS like 'Reims'

Agrégats

Un agrégat est un partitionnement horizontal d'une relation selon les valeurs d'un groupe d'attributs suivi d'un regroupement par une fonction de calcul.

L'idée estde regrouper les données d'une table en sous table et de realiser sur ces sous tables des opérations de calcul.

On utilise la clause **Group By**

Exemple :

Donner pour chaque NSS le nombre d'hospotalisations

Select NSS COUNT(\*) from hospitalisation Group By NSS

Restriction sur regroupement:

On utilise la classe HAVING

Donner les NSS des patients ayant été hospitalisés plus de 2 fois

Select NSS, COUNT(\*) From hospitalisation Group By NSS HAVING C OUNT(\*)>2

Syntaxe maximale:

Select <liste d'attributs> From <liste de relation> soin <liste de relation> USING <liste attributs> where <restriction> Group By <liste attributs à partitionner> having <restriction sur regroupement> orderby <attribut>[ASC/DESC]… <attribut>[ASC/DESC]

Sous requetes:

Se placent derrière une clause where, une clause HAVING, et/ou plus rarement derrière une clause From

Une sous requete retourne :

* Soit une valeur simple (on utilise alors un teste d'égalité ou de comparaison)
* Soit une table composée d'attributs (on utilise les opérateurs IN, EXISTS)

Le mot clé EXISTS (Select…) est evalué à vrai si le resultat est de l'évaluation du select qui le suit est non vide

La requete "appelante" n'est évaluée que si le resultat de la sous requete est evaluée à vrai

La division :

La division de la relation R(A1…An) par la relation S (Ap+1…An) est la relation T (A1…Ap) formée de tous les tuples qui concaténés à chaque tuple de S donnent toujours un tuple de R.

SQL n'offre pas la possibilité d'exprimer le quantificateur "Ɐ". On peut utiliser le quantificateur "ⱻ".

La question "donner le NSS des Patients ayant été hospitalisés dans tous les services"

ΠNSS idShospitaliser/πidS Service

Donner les NSS de Patient pour lesquel il n'existe aucun service qui ne l'a pas accuilli

R(A,B)/S(B)=T(A)

={QꞒR(A)|ⱯbꞒS(B),(a,b)ꞒR(A,B)}

={QꞒR(A)|ⱻbꞒS(B),(Q,b)ꞒR(AB)}

/ /

Select A from R as R1 where not EXISTS (Select B from S where not EXISTS (select AB from R where R1.A=A AND S.B=B))

Opérations ensemblistes :

Union : on utilise UNION

Intersection : INTERSECT

Différence : EXCEPT

Les patients extérieurs à reims

Patient \6ville=Reims Patient

(Select \* from Patient)EXCEPT (select \* from Patient where villeP like 'Reims')

Mise à jour d'information:

* Insertion :
  + INSERT INTO <relation>[att1…attn] Values (Val1…Valn)
  + INSERT INTO <relation>[att1…attn] (Select…)
* Mises-à-jour
  + UPDATE <Relation> Set att1=val1,…,att2=val2 where <conditions>
* Suppression
  + OELETE from <relation> where <conditions>

1. Les noms des patients de Reims autant été hospitalisées dans un service ou des patients de Lyon ont été hospitalisées pour des affections grave
2. Le nom des docteurs de Reims hospitalisant des patients pour des affections bénignes
3. Le nom des mutuelles étant cotisés par des patients hospitalisés à paris dans des services qui accueillent des patients de Rouen

2) πnomD((6villeD=reimsdocteur)|X|((hospitalisation)|X|codeAff(πcodeAff(6dangerosité = BenignesAffection))))

Select nomD From docteur naturalJoin (Hospitalisation NaturalJoin affection using codeAff) using codeD where villeD like 'Reims' AND dangerosité like 'benigne'

Select nomD from docteur where villeD like 'Reims' AND codeD IN (Select codeD from hospitalisation where codeAff IN (Select CodeAff from affection where dangerosité like 'benigne'))

1) R11🡨πcodeAff(6dangerosité=graveaffection)

Select codeAff from affection where dangerosité like 'Grave'

R12🡨πnSS(6villeP=Lyon patient)

Select nSS from patient where villeP like 'Lyon'

R13🡨πidS((Hospitalisation |X|codeAff R11)|X|nSS R12)

Select idS from hospitalise where codeAff IN(RM) AND NSS IN(R12)

R14🡨πnSS(Hospitalise|X|idSR13)

Select nSS from hospitalise where idS IN (R13)

R1🡨πnomP cilleP=Reims 6Patient |X| R14

Select nomP from Patient where villeP like 'Reims' AND NSS IN (Select NSS from hospitalise where idS IN (Select idS from hospitalise where codeAff IN (Select code Aff from affection where dangerosité like 'Grave') AND NSS IN (Select NSS from Patient where VilleP like 'Lyon')))

**La normalisation**

Exemple :

Emprunt Ami(N°=DVD, Titre, Nom, Prénom, Tel, DateEmprunt)

* Anomalie d'insertion
* Anomalie de mise à jour
* Anomalie de suppression

Normalisation :

Objectifs :

* Supprimer les anomalies précédentes et éliminer les Redondances
* Normaliser une relation consiste à décomposer une relation comportant des anomalies en plusieurs relations sans anomalie
* La normalisation repose sur les liens sémantiques entre attributs

Définition :

Soit R une relation et X, Y 2 sous-ensembles d'attributs. On dit que x détermine Y (ou que Y dépend fonctionnellement de X).

Si et seulement si

Ɐr extension de R, Ɐ E1, E2 Ꞓ r²

On a

ΠX(E1)=πX(E2)=>πy(E1)=πy(E2)

On note X🡪Y

Exemple :

CP🡪Ville

Ville🡪Pays

Salaire🡨diplôme

Malus🡪montant contrat d'assurance

Propriété des dépendances fonctionnelle :

Pour une relation R, l'ensemble des DF peut être très grand

Objectifs : réduire cet ensemble à une taille manipulable

Règles de Armstrong : Soient X, Y, Z des groupes d'attributs de R.

-Réflexive : Si Y C X alors X🡪Y

-Augmentation : Si X🡪Y alors XZ🡪YZ

-Transitivité : Si X🡪Y et Y🡪Z alors X🡪Z

3 propriétés dérivées :

* Union :

Si X🡪Y et X🡪Z alors X🡪YZ

* Décomposition :

Si X🡪YZ alors X🡪Y et X🡪Z

* Pseudo transitivité :

Si X🡪Y et WY🡪Z alors WX🡪Z

Exercice :

1. Soit R(ABCDE)

Avec A🡪B AD🡪BD (par augmentation)

D🡪E BD🡪BE (par augmentation)

1. R(ABCDEF)

A🡪BC A🡪B

B🡪E A🡪C

CD🡪EF AD🡪ED

Montrer que AD🡪E AD🡪E

AD🡪D

**Fermeture d'un ens de DF**

Définition :

L'ens de toutes les DF qui peuvent être déduites d'un ens de DF S de départ d'une relation R est appelé fermeture de S et est notée S+

Comment calculer S+ ?

En utilisant les règles de Armstrong

A partir d'un ens de DF S de départ, on peut calculer d'autres DF.

Définition :

* L'ensemble des DF noté S+ obtenues après application des règles de Armstrong s'appelle la fermeture de S
* 2 ens de DF S1 et S2 sont équivalent ssi ils ont la meme fermeture (ie S1+ = S2+)

**Fermeture d'un attribut**

Savoir si une DF X🡪Y appartient à la fermeture de S (ens de DF de départ).

On calcule la fermeture de X, X+, si Y est dans X' alors X🡪Y est dans S+.

**Algorithme de saturation**

Données : X ens d'attributs

S ens de DF

Sortie : X+ fermeture de X

Algo : Res🡨X

Tant que Res évolue

Pour chaque DF Y🡪Z de S

Si Y Ꞓ Res

Res🡨Res U {Z}

X+🡨Res

Eco:

Soit R(ABCDEF)

Avec S={ A🡪BC

E🡪CF

B🡪E

CD🡪EF

}

Calculer AB+

AB+={A B C D E F}

Corollaire :

Pour un ensemble S de DF. Une X🡪Y Ꞓ S+ ssi Y est un sous ens de X+

Calcul de S+

* Il est facile d'en déduire une méthode pour calculer S+.

Soit R une relation de Schéma A1…An avec S un ensemble de DF.

On calcule la fermeture de tout groupe d'attribut X de l'ensemble des parties de (A1…An)

FASTIDIEUX !

Ø

{A1} {A2} {A3} {A4} {A5} {A1 A2} {A1 A3} {A1 A4} {A1 A5} {A1 A2 A3} {A1 A2 A4} {A1 A2 A5}

{A2 A3} {A2 A4} {A2 A5} {A1 A3 A4} {A1 A3 A5}

{A3 A4} {A3 A5} {A2 A3 A4} {A2 A3 A5}

{A4 A5} {A2 A4 A5}

{A1 A2 A3 A4} {A1 A2 A3 A4 A5}

{A1 A2 A3 A5}

{A1 A2 A4 A5}

{A1 A3 A4 A5}

**Retour sur la notion de clé**

Définition :

Une clé candidate d'une relation

R (A1…An) est un groupe d'attributs X

Tel que

* + Unicité X🡪A1…An
  + Irreductabilité Ɐ X' C X X[](https://commons.wikimedia.org/wiki/File:U%2B219B.svg?uselang=fr)A1…An

**Couverture irredondante**

Définition :

2 ens de DF sont équivalant ssi ils ont la meme fermeture.

Parmi l'ens des DF S de départ certaines DF sont redondantes d'autres découlent de celles exprimées

Objectif :

Construire un ens de DF équivalent à l'ens de DF de départ mais uniquement avec les DF les plus pertinentes.

A partir de cet ensemble, on pourra générer toutes les autres DF.

Soit R une relation, S un ens de DF de départ.

But : isoler les DF les plus pertinentes.

Définition :

Une DF est dite élémentaire si elle est de la forme X🡪A où A est un attribut unique et où il n'existe pas X' C X tel que X'🡪A

Le dépendant A ne contient qu'un seul attribut.

Le déterminant est irréductible.

A partir d'un ens de DF élémentaires SE, l'ensemble de toutes les DF élémentaires sont obtenues uniquement par transitivité.

Sont SE un ens de DF élémentaires, une DF élémentaire est dite redondante si elle se déduit des autres par transitivité.

Dans S {A🡪B, A🡪C, B🡪C}

A🡪C est redondante.

2 méthodes :

-**Directe**

**-Méthode exhaustive**

La DF élémentaire X🡪A est-elle redondante ?

On calcule X+ sur S\{X🡪A} est redondante.

Définition :

Une couverture irredondante d'un ensemble de DF S est un ensemble de DF noté Irr(S) tel que :

1. Toute DF élémentaire est dans S+
2. Aucune DF de Irr(S) n'est redondante

**Construction de Irr(S)**

Soit S un ens de DF initial, on simplifie : Ɐ S, X🡪 Y de S

R1 : Rendre le membre droit atomique

R2 : Rendre le membre gauche irréductible

R3 : Eliminer les DF redondantes de l'ensemble obtenu après R1 et R2

Pour R1 : on applique la décomposition

Pour R2 : on ne considère que les membres gauches ayant plus d'un attribut. On applique l'algorithme de saturation sur X\{un attribut composant X}. Si on retrouve le nombre droit de S alors X est réductible

Pour R3 : on utilise soit la méthode directe, soit la méthode exhaustive

Calcul de |rr(S) pour R(ABCD) avec S={ A🡪BC B🡪C A🡪B AB🡪C AC🡪D}

[R1]

A🡪BC A🡪B

A🡪C

SR1 : {A🡪B A🡪C B🡪C AB🡪C AC🡪D}

[R2]

* AB🡪C

AB🡪C

{AB\{B}}+

A+={A B C D}

On retrouve c donc AB🡪C est réductible et B peut être retiré de AB

{AB\{A}}+

B+={B,C}

On retrouve C

AB🡪C n'est pas irréductible et A peut être retiré de AB

On obtient AB🡪C

A🡪C

B🡪C

* AC🡪D

{AC\{A}}+

C+={C}

On ne retrouve pas D donc A ne peut être retiré

{AC\{C}}+

A+={A B C D} on retrouve D donc C peut être retiré

* A🡪D

SR2

{A🡪B A🡪C A🡪D B🡪C}

SR3

{A🡪B A🡪D B🡪C}

Irr(S)={A🡪B B🡪C A🡪D}

Normaliser :

Objectif : supprimer les anomalies

Une relation est dans une forme normale particulière

1NF C 2NF C 3NF

La procédure consiste à transformer une relation qui se trouve dans une forme normale particulière en une sous ensemble de relations dans une formel normale plus satisfaisante.

La procédure est réversible.

Théorème de décomposition :

Toute relation R (A1…An) est décomposable sous perte d'informations en R1 et R2 s'il existe une DF X🡪 G

R1=πX Y(R)

R2=πdelta\{y}(R)

Delta =(A1…An)

Démarche globale

🡪 calcul de la couverture irredondante

🡪calcul des clés candidates

* Normalisation 2NF
* Normalisation 3NF

Définition :

* Une relation est en première forme normale (1NF) si chacun des attributs qui la compose est atomique.
* Une relation R est en deuxième forme normale (2NF)

Ssi :

* R est 1nf
* Tout attributs n'appartiennent pas à une clé candidate ne dépend pas d'une partie d'une clé

R(ABCDE) AB🡪CDE R1 (A B D E)

B🡪C R2(B C)

* Une relation R est en troisième forme normale (3NF) ssi :
* R est 2NF
* Tout attribut non clé ne dépend pas d'un autre attribut non clé

R(ABCD) AB🡪CDE

D🡪E

R1(ABCD)

R2(DE)

**Données rapportées**

Étudiant

Diplome

Objet :

Répartir la Base de données sur un système reparti de gestion de base de données.

Autre morceau BD

Morceau BD

X ième morceau BD

Avantage de la répartition :

* Augmentation des performances
* Fiabilité/disponibilités
* Evolution/ extensible
* Partage
* Contrôle partagé
* Hétérogénéité

Inconvénients :

* Mise en œuvre très complexe
* Problème de contrôle distribué
* Support de communication très lent

Architecture :

* Centralisée
* Distribuée

Important :

Transparence pour l'utilisateur!

* + Localisation
  + Partitionnement
  + Duplication

Les "morceaux de la BD" sont obtenus par fragmentation :

* + Horizontale
  + Verticale
  + Mixte

🡪 Fragmentation (obtenir les fragments)

🡪 Allocation (déterminer ou doivent être localisé les fragments

🡪 Test

(Autres techniques :

* Fragmentation dynamique
* Allocation dynamique
* Cliché
* Duplication

)

**Gestion de la concurrence :**

Définition :

Une transaction est une unité de traitement séquentielle cohérente et fiable constituée d'une suite d'opération à réaliser sur la base de données. Appliqués à une base de données cohérente, une transaction restitue une base de données cohérente.

Opération sur les transactions :

* Start
* Commit
* Abort
* Opérations internes
* Lecture/écriture

Une transaction est active si elle a extrait son start mais de commit ni de abort

Propriété des transactions :

* Atomicité : toutes les opérations de la transaction sont exécutées ou aucune d'entre elles
* Cohérence : les transactions doivent préserver la cohérence de la BD
* Isolation : les mises à jour effectuées par une transaction ne sont visibles qu'une fois celle-ci validée.
* Durabilité : les mises à jour sont définitives.

Soit 2 transactions :

T1

T11 x🡨lire(x)

T12x🡨x+10

T13 Ecrire(X,x)

T2

T21y🡨lire(x)

T22y🡨y+20

T23Ecrire(X,y)

Perte de mises à jour :

X=0

On exécute T1 puis T2 ou T2 puis T1

On obtient X=30

X=0

On exécute T1 et T2 "en meme temps"

T11,t21,t12,t22,t13,t23

X=20

T11,t21,t12,t22,t13,t23

X=10

|  |  |
| --- | --- |
| T1 | T2 |
| X Ꞓ lire (A)  X Ꞓ X+1  Ecrire (A,X)  Y 🡨 lire (B)  Y🡨Y+1  Ecrire (B,Y) | Z 🡨 lire (A)  Z 🡨 Z+2  Ecrire (A,Z)  T 🡨 lire (B)  T 🡨 T\*2  Ecrire (B,T) |

A=B=0 contrainte de cohérence A=B

T1 puis T2 A=B=2

T2 puis T1 B=A=1

T1 // T2

Lecture non reproductible :

|  |  |
| --- | --- |
| T1 | T2 |
| X Ꞓ lire (A)  Afficher (X) | Y 🡨 lire A  Y 🡨 Y+1  Ecrire (A,Y) |
| X 🡨 lire (A)  Afficher (X) |

**Gestion des transactions**

Gestionnaire des transactions :

Réaliser les exécutions des opérations des transactions

Ordonnanceur :

Contrôle l'ordre des opérations des différentes transactions

Architecture :

Un site

Transactions

Gestionnaire de transaction

Gestionnaire de transaction

Ordonnanceur

Ordonnanceur

Gestionnaire de données

Données

Rôle de l'ordonnanceur :

Lorsqu'il reçoit une opération d'une transaction, il peut :

* + L'exécuter
  + Le rejeter
  + Le mettre en attente

Définition :

Une exécution d'un ensemble de transaction est dite sérialisable si elle donne pour chaque transaction participante, le meme résultat que l'exécution en série (séquentielle) de ces meme transactions.

Conflit d'opérations :

Les conflits pour plusieurs transactions on lieux sur des opérations (lecture/écriture \_\_ écriture/écriture) il faut donc connaitre l'ordonnancement des opérations.

Graphe de précédence :

G=(V,E)

V représente l'ensemble des transactions

E les relations de précédence par rapport à une donnée

Un arc de Ti à Tj avec le label A.

Indique que Ti accède à A avant Tj

Un graphe de précédence sans cycle caractérise une exécution sérialisable

Afin d'ordonner les opérations des différences transactions il est possible d'utiliser des verrous sur les données. 2 types de verrous : lecture et écriture. Un verrou en écriture est exclusif.

Soit l'exécution suivante :

T1 lock(X,E)

T2 lock(Y,E)

T2 lock(Z,E)

T2 Unlock(Y)

T1 lock(Y,E)

T1 unlock(X)

T2 lock(X,E)

T2 unlock(Z)

T2 unlock(X)

T3 lock (X,E)

T3 lock(Z,E)

T3 unlock (X)

T3 unlock (Z)

T1 unlock (Y)

* Tracer le cercle

X

T1

T2

Y

Z

X

T3

X

Verrouillage à 2 phases.

La technique du verrouillage à 2 phases consiste par une transaction à ne plus poser de verrou après en avoir relâché un

Verrouillage à 2 phases => sérialisable

Exo :

Proposer une exécution de 2 transactions sur 3 variables respectant le verrouillage à 2 phases

T1 lock (X, Y, Z)

T1 unlock (X)

T2 lock (X)

T1 unlock (Y)

T2 lock (Y)

T1 unlock (Z)

T2 lock (Z)

T2 unlock (X, Y, Z)

Propriétés du crontrole de concurrence

Sureté : toute exécution concurente de transaction est sérialisable

Vivacité : toute transaction se termine en temps fini

Soit les 2 transactions T1 et T2 : T1//T2

T1 lock (X,Y)

T2 lock (Y, X)

T2 unlock (X, Y)

T1 unlock (X, Y)

T1 lock X

T2 lock Y

T1 lock Y 🡪 attente

T2 lock X 🡪 attente

🡺Situation d'interblocage

Graphe des attentes :

G=(V,E)

V ensemble des transactions

E relations d'attente entre transactions

Circuit ! 🡪 interblocage

T2

T1