Module Bases de Données, ESSI3
Université de Nice et Sophia-Antipolis

PRINCIPES DES SYSTÈMES DE GESTION DES BASES DE DONNÉES

Support de cours
Novembre 1993

N. Le Thanh

Table des matières

Table des matières

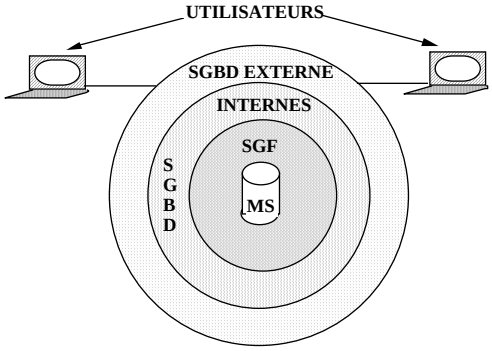
I- INTRODUCTION

- **Notion de SGBD**
- **Brève histoire des SGBD**
- **Objectifs et architectures des SGBDs**

I- INTRODUCTION

1- NOTION DE SGBD

• UN SGBD EST UN SYSTÈME PERMETTANT LE RANGEMENT, LA RECHERCHE, L'ASSEMBLAGE ET LA CONVERSION DES DONNÉES



• UNE BASE DE DONNÉE EST DÉFINIE COMME UN ENSEMBLE DE DONNÉES ASSOCIÉES À UNE MÊME APPLICATION GÉRÉ PAR UN SGBD

- 5 -

I- INTRODUCTION

caractéristiques

- **Description de données (nom, format, caractéristiques, ...) indépendant de leur utilisation (mise à jour, recherche)**
- **Protection des données contre tout incident ou accident**
- **Obtention d'une performance acceptable**

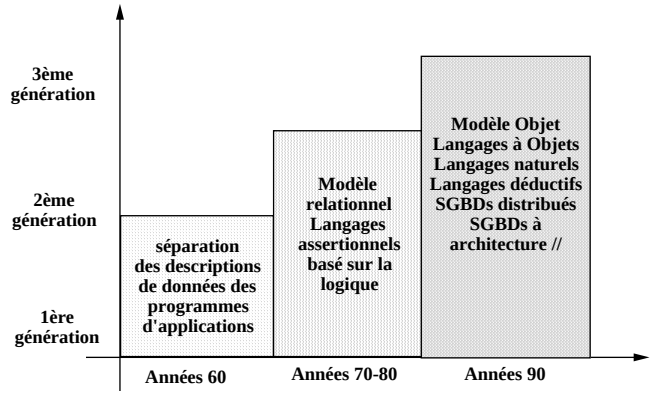
Architecture globale

- **COUCHE SGF : gestion des récipients de données sur mémoires secondaires**
- **COUCHE INTERNE :**
  - **gestion des données stockées dans les fichiers**
  - **placement, assemblage de ces données**
  - **gestion des liens entre données et des structures**
  - **gestion d'accès aux données**
- **COUCHE EXTERNE :**
  - **présentation des données aux programmes d'applications et aux d'usagers terminaux**
  - **langages permettant de contrôle de définition et de manipulation des données (analyse et interprétation des requêtes des usagers et mise en forme des données échangées avec le monde extérieur)**

- 6 -

I- INTRODUCTION

2- BRIÈVE HISTOIRE DE SGBD



- 7 -

I- INTRODUCTION

- **Avant 1960 : développement des SGF qui tendent essentiellement à gérer les mémoires secondaires comme :**
  - **• mémoires partageables**
  - **• mémoires banalisées**
  - **• mémoires directement adressables**
  - **• mémoires de capacité "infinie"**

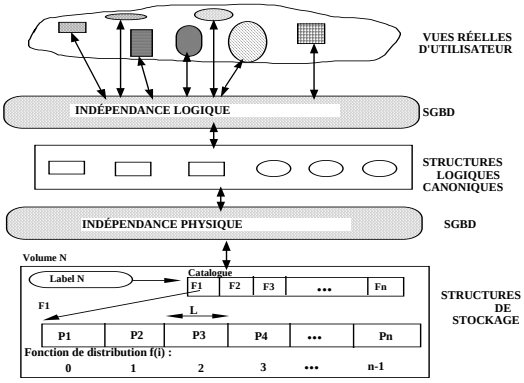
  —> SGF est le composant noyau de SGBD
- **Au milieu des années 60 : la naissance de la 1ère génération de SGBD :**
  - **• séparation de la description des données des programmes d'application**
  - **• langages d'accès navigationnels permettant de se déplacer dans les structures de type graohe et d'obtenir, un par un, des groupes de données appelées "articles"**
  - **• systèmes basés sur des modèles d'accès visant essentiellement à optimiser les méthodes de déplacement des données sur les mémoires secondaires afin de réduire les temps d'accès**
- **Les années 70-80 : l'introduction du modèle relationnel par Codd en 1970 a découlé une vague de recherche et de développement des SGBDs de deuxième génération : les SGBDs relationnels :**
  - **• enrichissement la couche externe de SGBD afin de simlifier l'accès aux données pour les utilisateurs externes**
  - **• les langages assertionnels basés sur la logique du 1er ordre permettant de spécifier les données indépendant des méthodes d'accès aux données**
- **Depuis du milieu des années 80 : plusieurs directions de recherche sur les modèles et sur les langages pour les SGBDs de la 3ème génération :**
  - **• modèles et langages à objets**
  - **• BDs déductives, langages naturels, ...**

- 8 -

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD
3.1- OBJECTIFS DES SGBD

• INDÉPENDANCE PHYSIQUE - INDÉPENDANCE LOGIQUE



- 9 -

I- INTRODUCTION

• INDÉPENDANCE PHYSIQUE

- **Les données élémentaires du monde réel sont assemblées pour décrire les entités et les associations entre entités directement perceptible dans le monde réel : entités, lien, actions, contraintes d'intégrité, ... Cette structure d'assemblage appartient aux concepteurs des applications**
- **La structure de stockage des données, par opposition, appartient au monde des informaticiens et n'a un sens que dans l'univers du système informatique : article, fichier, méthodes d'accès, ... Cet assemblage propre au monde informatique, doit être basé sur des considérations de technique et de performance**
- **La réalisation de l'indépendance des structures de stockage du monde physique aux structures de données du monde réel consiste à pouvoir définir l'assemblage des données élémentaires entre elles dans le système informatique indépendamment de l'assemblage réalisé dans le monde réel, en tenant compte seulement des critères de performance et de flexibilité**

• INDÉPENDANCE LOGIQUE

- **L'indépendance logique permet de définir dynamiquement des structures applicatives adaptées aux différentes classes d'utilisateurs indépendamment de la structure logique canonique de données dans SGBD**
- **Chaque classe d'utilisateur de voir les données comme elle souhaîte**
- **L'évolution de la vue de chaque classe ne remet pas en cause la structure canonique et inversement, la modification de la structure canonique ne perturbe pas les vues d'utilisateurs**

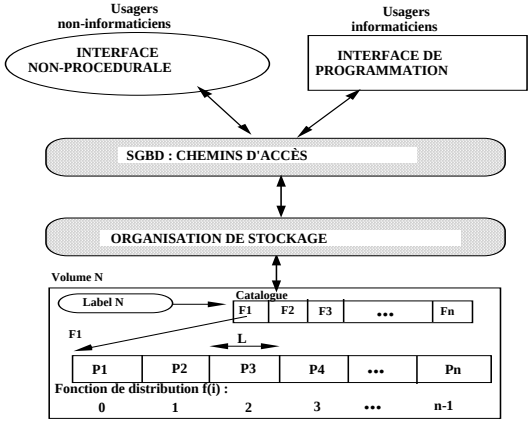
- 10 -

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD

3.1- OBJECTIFS DES SGBD

• INTERFACES BANALISÉES - EFFICACITÉ D'ACCÈS



I- INTRODUCTION

• INTERFACES BANALISÉES - EFFICACITÉ D'ACCÈS

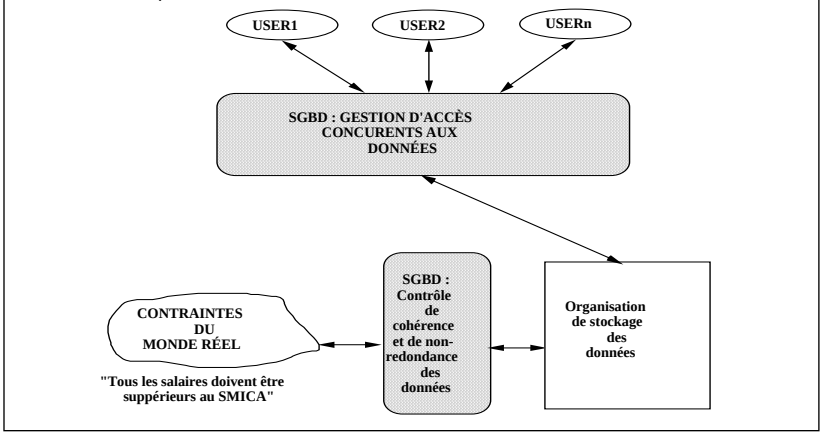
- **Les usagers non-informaticiens peuvent non seulement "voir" les données de leurs applications mais aussi manipuler ces données. Cette manipulation doit être basée sur des concepts d'interface non procédurales : l'utilisateur précise seulement les données qu'il veut chercher et manipuler mais non comment doit faire système pour accéder à ces données**
- **Pour pouvoir réaliser ces interfaces, les SGBDs doivent être disposés des moyens pour contruire des chemins d'accès efficaces aux données demandées. Ces moyens sont basés d'une part sur des modèles logiques de données, et d'autre part, sur une organisation spécifique, appelée niveau de chemins d'accès logiques, qui fait le pont entre un modèle logique de données et les organisations de stockages de données**
- **D'un autre côté, pour les informaticiens avertis, voire des programmeurs-systèmes, connaissant la structure de stockage des données, il doit être possible de fournir des langages de manipulation de données très efficaces permettant d'accès rapidement aux données le long des chemins d'accès définis dans la structure de stockage. Ces langages doivent être utilisables à partir de programmes classiques écrits en assembleur ou en langages de haut niveau**

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD

3.1- OBJECTIFS DES SGBD

• COHÉRENCES, NON-REDONDANCE ET PARTAGEABILITÉ DES DONNÉES



- 13 -

I- INTRODUCTION

• COHÉRENCES, NON-REDONDANCE ET PARTAGEABILITÉ DES DONNÉES

- **Dans le systèmes classiques à fichiers non intégrés, chaque application possède ses données propres. Ceci conduit généralement à de nombreuses duplications de données avec une perte de place en mémoire secondaire associée. En outre, il crée la difficulté pour la saisie et la maintenance de données. Dans l'approche de Bases de données, les données sont partagées entre plusieurs applications. Les fichiers plus ou moins redondants seront intégés dans un seul. Les SGBD doit veiller à la non duplication physique des données afin d'éviter les stockages et mise à jours multiples**

- **Chaque application réelle doit être modèlisée à partir un ensemble de règles, appelées contraintes d'intégrité, qui définissent la cohérence de données stockées vis-à-vis du monde réel. Le SGBD doit pouvoir représenter ces règles (par le modèle de données et des dispositifs spécifiques) et de vérifier automatiquement, à chaque mise à jour de données, que ces règles sont respectés**

- **Les données dans un SGBD sont par principe partageables entre plusieurs applications, voire plusieurs utilisateurs. La manipulation en simultanément des mêmes données par plusieurs utilisateurs peuvent introduire des erreurs et des incohérences sur les données. Le SGBD doit avoir des dispositifs adéquats permettant de gérer simultanémént plusieurs d'accès aux données des utilisateurs différents en tenant compte la cohérence et la performance d'accès**

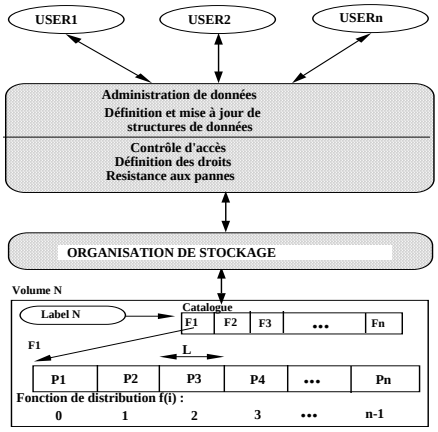
- 14 -

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD

3.1- OBJECTIFS DES SGBD

• ADMINISTRATION ET SÉCURITÉ DES DONNÉES



- 15 -

I- INTRODUCTION

• ADMINISTRATION ET SÉCURITÉ DES DONNÉES

- **Des fonctions essentielles des SGBDs sont la définition des structures des données et le suivi de leurs évolutions. Ces fonctions sont appelées administration des données. Afin de permettre un contrôle efficace des données, de résoudre les confits entre divers points de vue pas toujour cihérents, de pouvoir optimiser les accès aux données**

  --> Il est essentiel de répartir ces fonctions entre les mains d'un petit groupe de personnes hautement qualifiées.

  L'administration des données est ainsi souvent centraliée

- **Les données dans les Bases de données doivent être protégées contre les accès non autorisés ou mal intentionnés. Il doit exister des mécanismes adéquats pour autoriser, contrôler ou enlever les droits d'accès de n'importe quel usager à tout ensemble de données. Il est nécessaire que les SGBDs disposent des outils permettant de résister aux pannes logiques ou physiques**

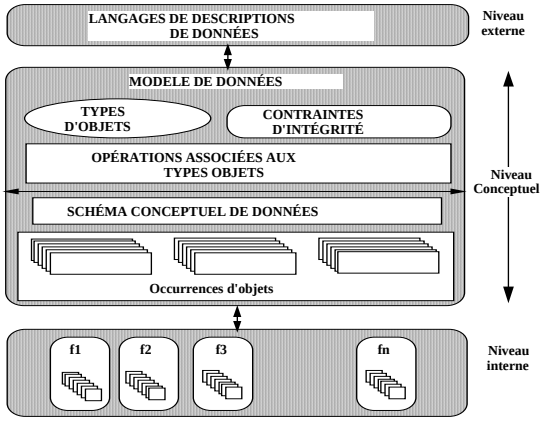
- 16 -

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD

3.2- CONCEPTS ET METHODES DE BASE

• DESCRIPTION DE DONNÉES



I- INTRODUCTION

• DESCRIPTION DES DONNÉES

- **TYPE D'OBJETS : Ensemble d'objets possédant des caractéristiques similaires et manipulables par des opérations identiques. Exemples : entier, réel, chaine, voiture, employé, ...**
- **OCCURRENCE D'OBJETS : Élément d'un ensemble d'objets**
- **MODÈLE DE DONNÉES : Ensemble de concepts, de règles de composition des concepts, des contraintes d'intégrité permettant de décrire et manipuler les types d'objets correspondant au monde réel**
- **LANGAGES DE DESCRIPTION DES DONNÉES : Langage supportant un modèle et permettant de décrire les données d'une base de données d'une manière assimilable par une machine**
- **SCHÉMA DE DONNÉES : Description des données issues des applications spécifiques et cohérentes**
- **NIVEAUX DE DESCRIPTION : Trois niveaux de description de données sont identifiés :**

  - NIVEAU CONCEPTUEL : Le niveau central permettant de définir les structures des données canoniques, à partir d'un modèle de données, correspondant aux certaines applications précises (types d'objets élémentaires, types d'objets composés, types d'objets d'association, ...)

  - NIVEAU INTERNE : Le niveau correspondant aus structures de stockage supportant les données. Il permet de décrire les données telles qu'elles sont stockées dans la machine (les fichiers, les chemins d'accès, ...)

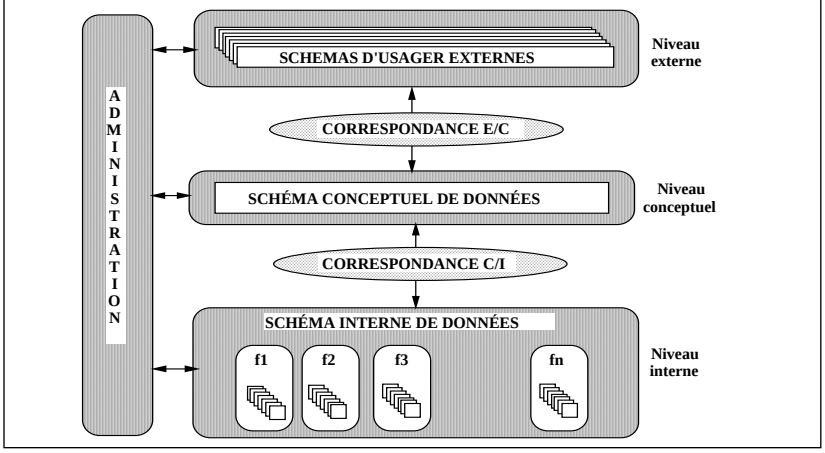
  - NIVEAU EXTERNE : Les schémas représentant les vues externes (multi-modèles) d'utilisateurs construits sur le schéma de données (modèle unique) canonique du niveau conceptuel

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD

3.2- CONCEPTS ET METHODES DE BASE

• TRANSFORMATION ET ADMINISTRATION DES DONNÉES



I- INTRODUCTION

• TRANSFORMATION ET ADMINISTRATION DES DONNÉES

- **L'existence de plusieurs niveaux de schémas de données gérés par un système pour décrire un même ensemble de données nécessite des outils systèmes assurant le passage entre niveaux d'une manière transparent vis-à-vis les utilisateurs. Ce sont les fonctions effectuant automatiquement la restruction d'occurrences de données conformes à un schéma en occcurences de données conformes à un autre schéma**
- **Dans un SGBD à trois niveaux de schémas, il existe deux transformations de données :**

  - la transformation conceptuelle/Interne permettant de faire passer des occurences de données depuis le format conceptuel au format interne et réciproquement

  - la transformation conceptuelle/Externe permettant de faire passer des occurences de données depuis le format conceptuel au format externe et réciproquement
- **La définition des différents schémas et des règles de transformation entre les niveaux est, en général, effectuée par les responsables, appelés administrateurs de données dont leurs rôles sont :**

  - administrateur de bases de données correspondant à la définition du schéma interne et des règles correspondance Interne/Conceptuel

  - administrateur d'entreprise correspondant à la définition du schéma conceptuel

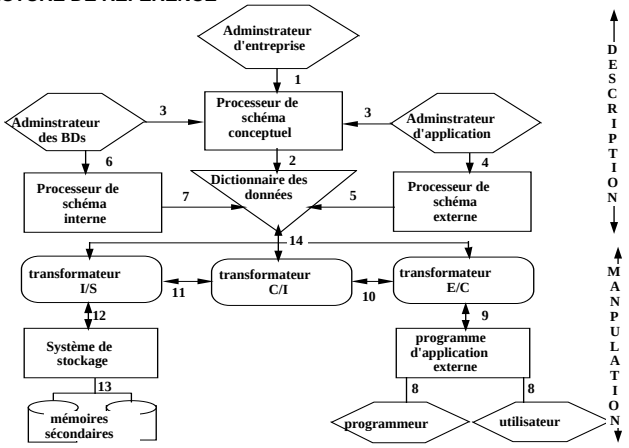
  - administrateur d'application correspondant à la définition du schéma externe et des règles correspondance Externe/Conceptuel
- **Les différents schémas et règles de transformation associés à une base de données sont stockés dans une structure, appelé dictionnaire de données. Le dictionnaire de données peut-être lui-même implanté dans le SGBD comme une base de données. Il constitute ainsi une méta-base : une base décrivant les autres bases**

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD

3.3- ARCHITECHTURE DES SGBDs

• ARCHITECTURE DE RÉFÉRENCE



I- INTRODUCTION

• ARCHITECTURE DE RÉFÉRENCE (ANSI/X3/SPARC)

- **L'architecture comporte de deux parties (à trois niveaux) :**
  - **la partie de description des données (autour du dictionnaire)**
  - **la partie de manipulation des données (accès, mise à jours)**

- **Les diverses interfaces :**

  1- Langage de description de données conceptuelles, format source:

  définition du scHéma conceptuel canonique avant la compilation

  2- Langage de description de données conceptuelles, format objet :

  résultat de la compilation du schéma conceptuel pour stocker dans la dictionnaire de données

  3- Description de données conceptuelles, format édition : consultation du schéma conceptuela a fin de définition des vues externes et les contraintes d'intégrité, ...

  4- Langages de description de données externes, format source : définition des vues externes à partir du schéma conceptuel

  5- Langages de description de données externes, format objet : les vues externes en code objet après la compilation

  6- Langages de description de données internes, format source : définition des schéma interne à partir du schéma conceptuel

I- INTRODUCTION

7- Langages de description de données internes, format objet : les contraintes d'intégrité, schémas internes après la compilation

8- Langages de manipulation de données externes, format source : interfaces de programme ou d'usagers pour manipuler des données

9- Langages de manipulation de données externes, format objet : programme après la compilation

10- Langages de manipulation de données conceptuelles, format objet transformation E/C en manipulation d'objets du schéma conceptuel

11- Langages de manipulation de données internes, format objet : transformation C/I en manipulation d'objets du schéma interne

12- Langages de manipulation de stockage, format objet : interface du système de stockage

13- Interface du mémoire secondaire : Entrées/sorties sur les supports

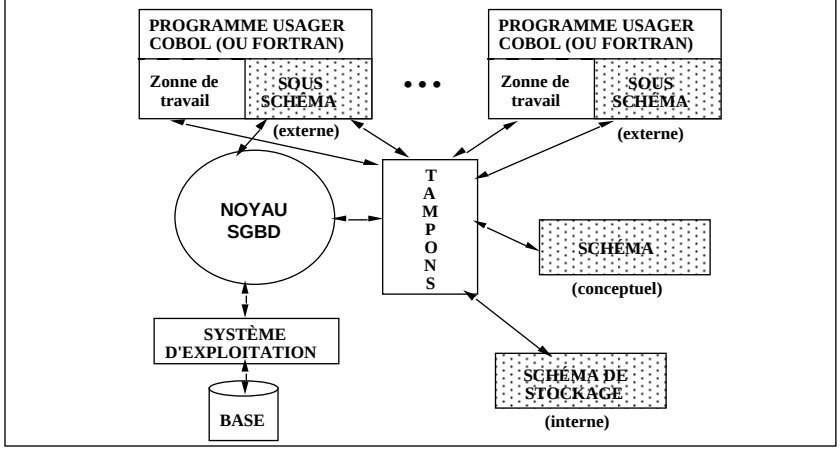
14- Interface d'accès au dictionnaire de données (pour les processeurs de transformation et d'accès)

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD

3.3- ARCHITECHTURE DES SGBDs

• ARCHITECTURE DU DBTG CODASYL (1971)



I- INTRODUCTION

• ARCHITECTURE DU DBTG CODASYL (1971)

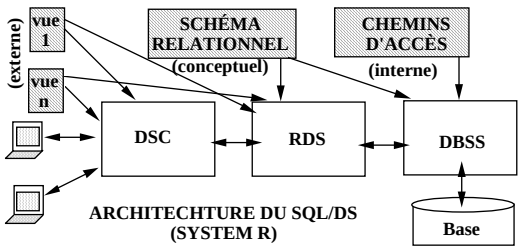
- **Le groupe de travail "Data Base Task Group" du comité CODASYL responsable du développement de COBOL a proposé des recommandations pour contruire un système de bases de données depuis 1971 : architecture et langage orienté COBOL**
- **L'architecture CODASYL s'articule autour du 'SCHÉMA' qui permet de définir les articles, leurs données et les lien entre ces articles (assimilé au schéma conceptuel)**
- **La structure de stockage de données est définie par le 'SCHÉMA DE STOCKAGE' qui correspond plus ou moins au schéma interne d'ANSI**
- **La notion de schéma externe est définie pour COBOL et appelée 'SOUS-CHÉMA'. Une proposition de 'sous schéma" pour le langage FORTRAN a été aussi faite**

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD

3.3- ARCHITECHTURE DES SGBDs

• ARCHITECTURE DES SGBD RELATIONNELS



ARCHITECHTURE DU SQL/DS
(SYSTEM R)

I- INTRODUCTION

• ARCHITECTURE DES SGBD RELATIONNELS

- **Le SYSTEM R est un prototype de recherche sur SGBD relationnel de l'IBM (années 70). Son architecture est divisée en trois composants principaux (Astrahan 76) :**

  - DATA SYSTEM CONTROL (DSC) : Ce composant assure les communications du système de base de données avac les autres programmes. Il contrôle l'initialisation et la terminaison du système et agit comme le superviseur du système. Il contrôle joue aussi le rôle d'un moniteur transactionnel qui contrôle plusieurs usagers simultanés

  - RELATIONAL DATA SYSTEM (RDS) : Ce composant translate les requêtes relationnelles en modules d'accès aux tables implanteées sur disques. Il gère les vues, détermine les meilleurs chemins d'accès et en général effectue la compilation des requêtes en provenant des usagers

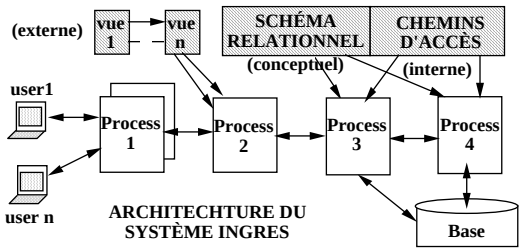
  - DATA BASE STORAGE SYSTEM (DBSS) : Ce composant effectue les accès demandés par les modules générés par RDS. Il gère également les allocations d'espace disques, les problèmes d'accès concurents et les reprises en cas de pannes. L'organisation des fichiers est VSAM

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD

3.3- ARCHITECHTURE DES SGBDs

• ARCHITECTURE DES SGBD RELATIONNELS



ARCHITECHTURE DU SYSTÈME INGRES

I- INTRODUCTION

• ARCHITECTURE DES SGBD RELATIONNELS

- **Le système INGRES (Stonebraker 76) réalisé à Berkeley et commercialisé sur les mini-ordinateur de Digital Equipment. INGRES est construit autour du système UNIX. Le SGBD se compose de quatre composants :**

  - PROCESS 1: qui est un moniteur interatif de terminaux. Il permet à l'usager devant un terminal de formuler, imprimer, modifier et lancer l'exécution des commandes INGRES. Il assure donc les communications avec les usagers

  - PROCESS 2: qui est un analyseur syntaxique permettant la modification des requêtes afin de supporter les vue, la protection et le contrôle de cohérence des donnéesn et enfin, le contrôle des accès concurents aux données

  - PROCESS 3 : qui est un Analyseur/Exécuteur des requêtes, permettant la décomposition d'une requêtes portant sur plusieurs tables en des sous-requêtes plus simples portant sur une table unique. Il accède également aux données afin de traiter les requêtes mono-table

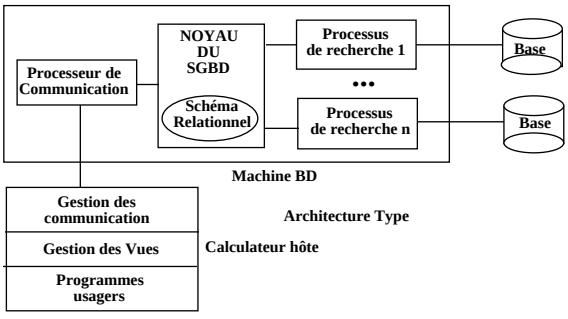
  - PROCESS 4 : qui se compose d'un certain nombre d'utilitaires d'administrateur de bases de données (création de tables, d'index, ...). Il prend également en charge les problème de résistance aux pannes et retardant les mises à jour jusqu'à la fin de la transaction

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD

3.3- ARCHITECHTURE DES SGBDs

• ARCHITECTURE DES MACHINES BASES DE DONNÉES



I- INTRODUCTION

• ARCHITECTURE DES MACHINES BASES DE DONNÉES

- **Une machine de Bases de données se compose en général de quatre types de modules :**

  - Des processeurs de recherche (Fitres) : qui sont chargés d'effctuer la sélectiondes données rechercheres anssi que leur enregistrement sur les disques (1 processeur par groupe de disques, par disque ou par piste

  - Le noyau du système de bases de données : Ce noyau est généralement relationnel. Il gère des chemins d'accès logiques aux données, les opérateurs relationnels, les accès concurents et la résistance aux pannes

  - La gestion des vues, de l'intégrité et de la sécurité : Une partie sur les machines hôtes, une partie sur la machine BD

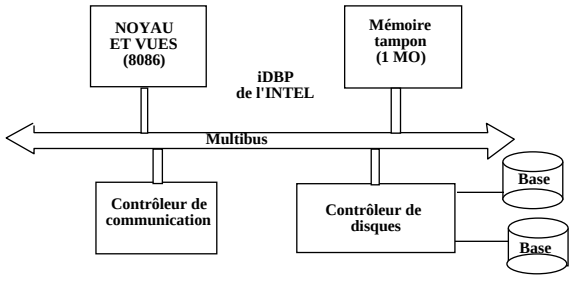
  - La gestion des communications : qui peut être réalisée sur un réseau de communication, un réseau de processeurs ou des canaux spécifiques

- 31 -

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD
3.3- ARCHITECHTURE DES SGBDs

• ARCHITECTURE DES MACHINES BASES DE DONNÉES

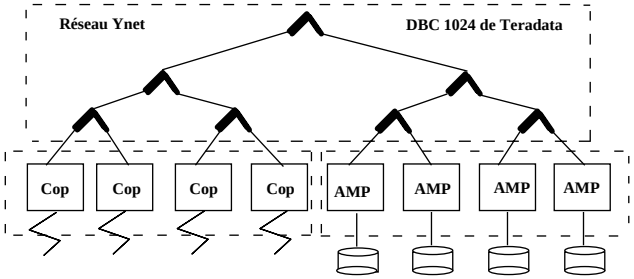


- 32 -

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD
3.3- ARCHITECHTURE DES SGBDs

• **ARCHITECTURE DES MACHINES BASES DE DONNÉES**



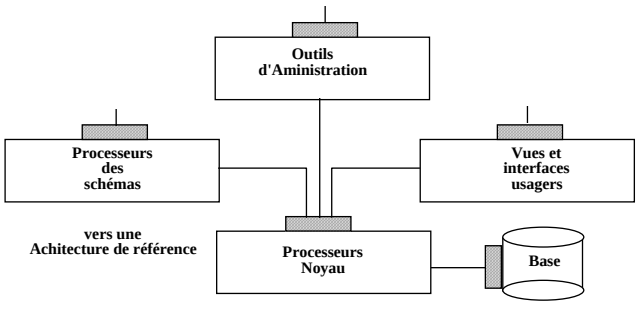
- **AMP : Noyau relationnel**
- **COP : Processeurs d'interface et de contrôle d'exécution**
- **Ynet : Réseau de processeurs de communication et de tri**

- 33 -

I- INTRODUCTION

3- OBJECTIFS ET ARCHITECTURE DES SGBD
3.3- ARCHITECHTURE DES SGBDs

• **ARCHITECTURE DES MACHINES BASES DE DONNÉES**



Proposée par CCA (Computer Corporation of America)

- 34 -

II- MODÈLE RELATIONNEL ET SES LANGAGES

- Concepts de base du Modèle relationnel
- Algèbre relationnelle
- Dépendances et formes normales
- Langages relationnels

II- MODÈLE RELATIONNEL ET SES LANGAGES

1- CONCEPTS DE BASE DU MODÈLE
1.1- STRUCTURES DE DONNÉES

- **DOMAINE : Un ensemble des valeurs atomiques**
  Exemple : - Domaine de numéros d'immatriculation VNI
  - Domaine VNOM
  - Domaine Puissance fiscales VPF
  - Domaine VCOULEUR

- **RELATION (TABLE) : Sous-ensemble du produit cartésien de n domaines non nécessairement distincts**
  Exemple : VOITURE $\subseteq$ VNI x VNOM x VNOM x VPF x VCOULEUR

- **ATTRIBUT (COLONNE) : Rôle d'un domaine dans une relation (nom unique dans une relation)**
  Exemple: Voiture (NV : VNI,
  Marque : VNOM,
  Type : VNOM,
  Puissance : VPF,
  Couleur : VCOULEUR)

II- MODÈLE RELATIONNEL ET SES LANGAGES

1- CONCEPTS DE BASE DU MODÈLE
1.1- STRUCTURES DE DONNÉES

EXEMPLE DE LA RELATION "VOITURE"

NV	Marque	Type	Puissance	Couleur
1970 AX 06	Renault	R19	7	Bleue
1971 AX 06	Peugeot	405	7	Rouge
1972 AX 06	Renault	Clio	5	Verte
1973 AX 06	Renault	Twingo	4	Blanche
1974 AX 06	Peugeot	605	9	Noire

II- MODÈLE RELATIONNEL ET SES LANGAGES

1- CONCEPTS DE BASE DU MODÈLE
1.2- CONTRAINTES D'INTÉGRITÉ

- **CLÉ PRIMAIRE :**

L'attribut ou l'ensemble d'attribut dont chaque valeur détermine, d'une manière unique, le tuple où elle appartient

- **DOMAINE PRIMAIRE :**

Tout domaine sur lequel est définie une clé primaire mono-attribut

- **CLE ÉTRANGÈRE :**

Tout attribut non clé défini sur un domaine primaire

- **CONTRAINTE DE RELATION :**

Tout n-uplet d'une relation est unique (non redondance)

II- MODÈLE RELATIONNEL ET SES LANGAGES

1- CONCEPTS DE BASE DU MODÈLE
1.2- CONTRAINTES D'INTÉGRITÉ

- **CONTRAINTE DE DOMAINE/ATRIBUT**

Il existe une et une seule clé-primaire dans un domaine primaire

Toute valeur d'un atribut appartient au domaine sur lequel défini cet attribut

La valeur de la clé primaire est unique et non nulle dans la relation

La valeur d'une clé étrangère est soit nulle soit égale à une valeur de l'attribut clé défini sur ce domaine

II- MODÈLE RELATIONNEL ET SES LANGAGES

2- ALGÈBRE RELATIONNELLE
2.1- OPÉRATIONS ENSEMBLISTES

- **UNION**

L'union de deux relations R et S unicompatibles (de même schéma) est une relation U = R ∪ S de même schéma contenant l'ensemble des tuples appartenant à R ou S ou aux deux relations

- **DIFFÉRENCE**

La différence de relation R par S (R - S) de même schéma est une relation D = R - S de même schéma contenant les tuples appartenant à R et n'appartenant pas à S

- **INTERSECTION**

L'intersection de deux relations R et S de même schéma est une relation I = R ∩ S de mêm schéma contenant l'ensemble des tuples appartenant aux deux relations
(remarque : I = R - (R - S) = S - (S - R))

- **PRODUIT CARTESIEN**

Le produit cartésien de deux relations (de schéma quelconque) R et S est une relation P = R x S, ayant pour attributs la concaténation de ceux de R et S et dont les tuples sont toutes les concaténations d'un tuple de R à un tuple de S.

II- MODÈLE RELATIONNEL ET SES LANGAGES

2- ALGÈBRE RELATIONNELLE

2.2- OPÉRATIONS RELATIONNELLES

- **PROJECTION**

La projection d'une relation R de schéma R(A1, A2, ..., An) sur les attributs Ai1, Ai2, .., Aip (avec ij ° ik et p < n) est une relation R'(Ai1, Ai2, .., Aip) = Proj (R / (Ai1, Ai2, ..Aip)) dont les tuples sont obtenus par élimination des valeurs des attributs de R n'appartenant pas à R' et par suppession des tuples en double

- **SÉLECTION**

La sélection de la relation R par une qualification Q est une relation R' = Sel (R /Q) de même schéma, dont les tuples sont ceux de R satisfaisant la qualification Q

- **JOINTURE**

La jointure de deux relations R et S selon une qualification Q quelconque est une relation R' de même schéma que celui du produit cartésien de R et S, notée R' = R x S (Q),

contenant les tuples de (R x S) satisfaisant la qualification Q

Cas particuliers : équi-jointure, jointure naturelle, autojointure

- **DIVISION**

La division de la relation R de schéma R(A1, A2, ..., An) par la sous-relations S de schéma S(Ap+1, .., An) est une relation Q de schéma Q(A1, .., Ap) formée de tous les tuples qui concaténés à chacun des tuples de S donne toujours un tuple de R

II- MODÈLE RELATIONNEL ET SES LANGAGES

3- DÉPENDANCES ET FORMES NORMALES

3.1- DÉCOMPOSITION D'UN SCHÉMA RELATIONNEL

- **DÉCOMPOSITION**

Une décomposition de la relation R de schéma R(A1, A2, .., An) est une remplacement de la relation R par une collection de relations R1, R2, ...Rm obtenues par des projections de R et telles que la jointure naturel R1 ⊗ R2 ⊗ .. ⊗ Rm est une relation ayant le même schéma que R

- **DÉCOMPOSITION SANS PERTE**

Une décomposition d'une relation R en R1, R2, .., Rm est dite sans perte si et seulement si, pour toute extension de R, R = R1 ⊗ R2 ⊗ .. ⊗ Rm

II- MODÈLE RELATIONNEL ET SES LANGAGES

3- DÉPENDANCES ET FORMES NORMALES
3.2- DÉPENDANCE FONCTIONNELLE

- **DÉPENDANCE FONCTIONNELLE**

Soient R(A1, A2, .., An) une relation et X, Y des sous-ensembles d'attributs de R. On dit que X détermine Y ou Y dépend fonctionnelement de X, et notée X Æ Y, si pour tout couple de tuples de R, si t1.X = t2.X on a aussi t1.Y = t2.Y

- **PROPRIÉTÉS**

Réflexivité :	$Y \subseteq X$	$\Rightarrow$	$X \rightarrow Y$
Augmentation :	$X \rightarrow Y$	$\Rightarrow$	$XZ \rightarrow YZ$
Transitivité :	$X \rightarrow Y$ et $Y \rightarrow Z$	$\Rightarrow$	$X \rightarrow Z$
Union :	$X \rightarrow Y$ et $X \rightarrow Z$	$\Rightarrow$	$X \rightarrow YZ$
Pseudo-transitivité :	$X \rightarrow Y$ et $WY \rightarrow Z$	$\Rightarrow$	$WX \rightarrow Z$
Décomposition :	$X \rightarrow Y$ et $Z \subseteq Y$	$\Rightarrow$	$X \rightarrow Z$

II- MODÈLE RELATIONNEL ET SES LANGAGES

3- DÉPENDANCES ET FORMES NORMALES
3.2- DÉPENDANCE FONCTIONNELLE

- **DÉPENDANCE FONCTIONNELLE ÉLÉMENTAIRE**

Toute dépendance fonctionnelle de la forme X Æ A, où A est un attribut unique non inclus dans X et où il n'existe pas X' appartenant dans X tel que X' Æ A

- **FERMETURE TRANSITIVE**

Soit F est un ensemble de dépendances fonctionnelles élémentaires sur un ensemble d'attributs. On appelle la fermeture transitive de F, notée F+, est l'union de F avec l'ensemble de toutes les dépendances engendrées de F par la transitivité

- **COUVERTURE MINIMALE**

Soit F est un ensemble de dépendances fonctionnelles élémentaires sur un ensemble d'attributs E. On appelle F la couverture minimale si et seulement si les deux propriétés suivantes sont vérifiées :

1) aucune dépendance dans F n'est redondant (F-f ° F)

2) Toute dépendance élémentaire sur E est dans F+

II- MODÈLE RELATIONNEL ET SES LANGAGES

3- DÉPENDANCES ET FORMES NORMALES

3.3-TROIS PREMIÈRES FORMES NORMALE

- **1ère FORME NORMALE : Une relation est en première forme normale si et seulement si tout attribut contient une valeur atomique**
- **2ème FORME NORMALE : Une relation est en deuxième forme normale si et seulement si elle est en 1ère forme normale et tout attribut n'appartenant pas à une clé ne dépend pas que d'une partie de cette clé**
- **3ème FORME NORMALE : Une relation est en troisième forme normale si et seulement si elle est en 2ère forme normale et tout attribut n'appartenant pas à une clé ne dépend pas d'un attribut non clé**
- **FORME NORMALE DE BOYCE-CODD : Une relation est en BCNF si et seulement si les seules dépendances fonctionnelles élémentaires sont celles dans lesquellees une clé détermine un attribut**
- **THÉORÈME DE DÉCOMPOSITION : Soit R(W) une relation, soit X, Y, Z trois ensembles d'attributs de W tels que $X \rightarrow Y \rightarrow Z$. La décomposition de R en deux relations R1(W1), R2(W2) telles que $W1 = (Y \cup Z)$, $W2 = (W - W1) \cup Y$ est une décomposition sans perte**

II- MODÈLE RELATIONNEL ET SES LANGAGES

3- DÉPENDANCES ET FORMES NORMALES

3.3-TROIS PREMIÈRES FORMES NORMALE

- **ALGORITHME DE DÉCOMPOSITION**

```
Procédure NORMALISER3 (DF, Atrributs) ;
Début
    C = " Couverture minimale des DF " ;
    " Editer les attribut isolés dans C ";
    RÉDUIRE (C) ;
    " Editer la relation composée de tous les attributs restants "
Fin;
Procédure REDUIRE (C) ;
Début
    Tantque " une DF n'inclut pas tous les attributs " faire
            " Recherche le plus grand ensemble d'attribut X tel que X  A1, X  A2, .., X  An ";
            " Editer les DF la relation R(X, A1, .., An) " ;
            " Eliminer le DF figurant dans R de C " ;
            " Eliminer les attributs isolés " ;
            RÉDUIRE (C) ;
    Fin tantque ;
Fin;
```

II- MODÈLE RELATIONNEL ET SES LANGAGES

3- DÉPENDANCES ET FORMES NORMALES

3.4- DÉPENDANCE MULTIVALUÉE ET 4ÈME FORME NORMALE

DÉPENDANCE MULTIVALUÉE

Soient R(A1, A2, .., An) une relation et X ey Y des sous ensembles d'attributs de R. On dit que X $\rightarrow\!\!\!\rightarrow$ Y est multidétermine Y ou il y a une dépendance multivaluée de Y sur X si, étant données des autres attributs Z = R - X - Y de la relation R

(X $\rightarrow\!\!\!\rightarrow$ Y) $\Leftrightarrow$ {(x,y,z) et (x,y',z') $\in$ R (x,y',z) et (x,y,z') $\in$ R}

PROPRIÉTÉS

1) Complémentation : (X $\rightarrow\!\!\!\rightarrow$ Y) ¼ (X $\rightarrow\!\!\!\rightarrow$ R - Y - X)

2) Augmentation : (X $\rightarrow\!\!\!\rightarrow$ Y) et (V $\subseteq$ W) $\Rightarrow$ (XW $\rightarrow\!\!\!\rightarrow$ YV)

3) Transitivité : (X $\rightarrow\!\!\!\rightarrow$ Y) et (Y $\rightarrow\!\!\!\rightarrow$ Z) $\Rightarrow$ (X $\rightarrow\!\!\!\rightarrow$ Z - Y)

4) Union : (X $\rightarrow\!\!\!\rightarrow$ Y) et (Y $\rightarrow\!\!\!\rightarrow$ Z) $\Rightarrow$ (X $\rightarrow\!\!\!\rightarrow$ YZ)

QUATRIEME FORME NORMALE

Une rélation est en quatrième forme normale si et seulement si les seules dépendances multivaluées élémentaires sont celles dans lesquelles une clé détermine un attribut

II- MODÈLE RELATIONNEL ET SES LANGAGES

3- DÉPENDANCES ET FORMES NORMALES

3.4- DÉPENDANCE MULTIVALUÉE ET 4ÈME FORME NORMALE

EXEMPLE : Soit la relation ETUDIANT

ÉTUDIANT	NE	COURS	SPORT
	10	BD	Tennis
	10	BD	Football
	20	BD	vélo
	20	MATH.	vélo

On déduit, par définition de dépendances multivaluées, que

NE $\rightarrow\!\!\!\rightarrow$ COURS **NE $\rightarrow\!\!\!\rightarrow$ SPORT**

ETUDIANT peut se décomposre en deux relations :

ET-COUR(NE, COURS) **et** **ET-SPORT(NE, SPORT)**

II- MODÈLE RELATIONNEL ET SES LANGAGES

3- DÉPENDANCES ET FORMES NORMALES

3.5- DÉPENDANCE DE JOINTURE ET 5ÈME FORME NORMALE

DÉPENDANCE DE JOINTURE

Soient R(A1, A2, .., An) une relation et X1, X2, .., Xm des sous ensembles d'attributs de R.

On dit qu'il existe une dépendance de jointure *[X1, X2, .., Xm] si et seulement si R est la jointure de ses projections sur X1, X2, .., Xm : R = R[X1] ⊗ R[X2] ⊗ .. ⊗ R[Xm]

CINQUIÈME FORME NORMALE

Un relation R est en forme normale si et seulement si toute dépendance de jointure est impliqué par les clés candidates de R

EXEMPLE : Dépendances de jointure entre

(BUVEUR, CRU), (BUVEUR, PRODUCTEUR) et (CRU, PRODUCTEUR)

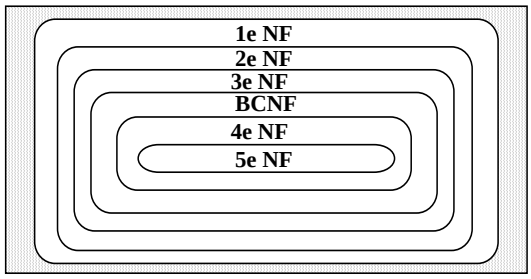
VINS	BUVEUR	CRU	PRODUCTEUR
	Jean	Chabris	Pierre
	Jean	Chabris	Alain
	Jean	Valrose	Alain
	Marc	Chabris	Alain

VINS se décompose en trois relations : R1(BUVEUR, CRU), R2 (BUVEUR, PRODUCTEUR) et R3(CRU, PRODUCTEUR)

II- MODÈLE RELATIONNEL ET SES LANGAGES

3- DÉPENDANCES ET FORMES NORMALES

3.6- PROCESSUS DE NORMALISATION



II- MODÈLE RELATIONNEL ET SES LANGAGES

4- LES LANGAGES
4.1- SQL

Interrogation	Mise à jour
SELECT	INSERT
FROM	INTO
WHERE	VALUES
GROUP BY	UPDATE
HAVING	SET
ORDER BY	DELETE
	WHERE

II- MODÈLE RELATIONNEL ET SES LANGAGES

4- LES LANGAGES
4.2- QUEL

Intérogation	Mise à jour
RETRIEVE	APPEND [TO] (<Att = val>*)
[INTO]	WHERE
WHERE	
	APPEND [TO] (<Att = val>*)
	WHERE
	REPLACE
	WHERE
	DELETE
	WHERE

II- MODÈLE RELATIONNEL ET SES LANGAGES

4- LES LANGAGES

4.3- QBE

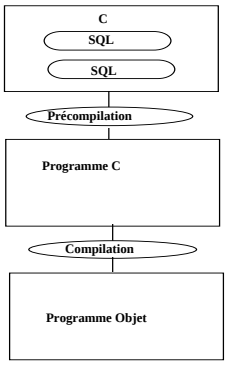
Intérogation **Mise à jour**

II- MODÈLE RELATIONNEL ET SES LANGAGES

4- LES LANGAGES

4.4- APPROCHES D'UTILISATION DEPUIS LES LANGAGES HÔTE

INTÉGRATION



EXTENSION

Etendre le langage hôte par :

- par l'introduction des types spécifique (relation, clé,...)

- par la possibilité de définir des variables de type relation

- par la possibilité de manipulation ces variables avec les qualifications logiques

- par introduction de la clause "FOR EACH" permettant de balayer les tuples associés à une variable

- par l'ajout des clauses de mise à jour

(Pascale / R)

III- PERFORMANCES ET STRUCTURES PHYSIQUES

- Introduction
- Quelques méthodes d'accès sélectifs

III- PERFORMANCES ET STRUCTURES PHYSIQUES

1- INTRODUCTION

- Performance = temps de réponse à une requête
- La base de données est stockée dans un support secondaire (disque), la performance dépend surtout l'organisation d'accès aux données
- Les critères d'accès aux données sont des critères sémantiques (accès par contenu)
- Par conséquent, l'organisation des chemins d'accès aux données doit être assurée par l'utilisateur de la base de données
- Tout SGBD dispose des moyens pour aider les utilisateurs à organiser les chemins d'accès aux données

III- PERFORMANCES ET STRUCTURES PHYSIQUES

1- INTRODUCTION

- **Deux méthodes utilisées :**

- les chemins d'accès sont intégrés dans des structures spécifiques de données (à priori) qui favorisent l'accès rapide (CLUSTER, ...)

- **intérêts : + réduction en général l'espace de stockage**
  + mécanisme simple
- **inconvenient : + structure figée, coût élevé si changement**
  + dépendence physique

- les chemins d'accès sont indépendantes (pospriori) de structures de données qui sont des structures favorisant l'accès rapide et contenant les pointeurs vers les données (INDEX)

- **intérêts : + structure dynamique et évolutive**
  + sécurité (indépendance physique)
- **inconvenients : +coût de stockage et de maintenance élevé**

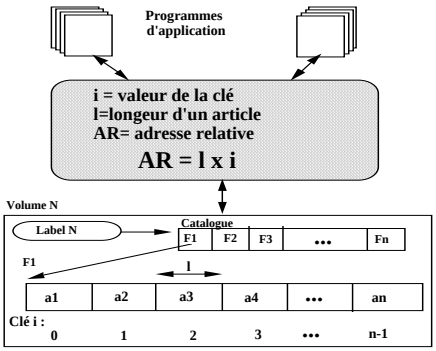
III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs
2.1- ORGANISATION D'ACCÈS DIRECT

- **CLÉ D'ARTICLE**

Identificateur d'un article permettant de sélectionner un article unique dans un fichier

- **ORGANISATION RELATIVE**



III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

2.1- ORGANISATION D'ACCÈS DIRECT

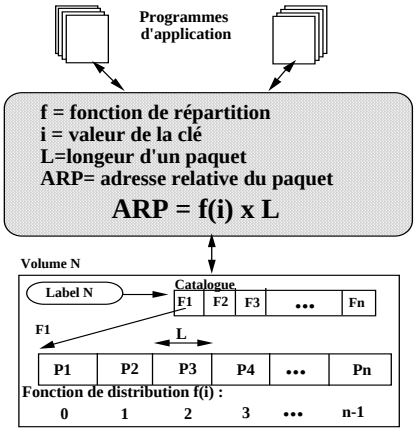
- **ORGANISATION RELATIVE (FICHIER RELATIF OU DIRECT)**

- **La taille du fichier doit être déterminée préalablement et la place pour le fichier doit être allouée dès sa création dans un l'espace consécultif sur le support**
- **La clé d'accès est l'ordre de l'article dans le fichier**
- **Les articles doivent être de même longueur l. Dans le cas contraire, on prend la taille maximale d'un article comme longueur**
- **Le calcul de l'adresse relative (AR) correspondant à la clé i est : AR = l x i**
- **AVANTAGES : • Simple • Performante**
  • Clé indépendante du contenu de l'article (l'ordre)
- **INCONVÉNIENTS : • perte de place importante • taille fixe des articles**
  • fichier statique (non évoluable en taille)
- **La méthode est adaptée à des fichiers composés des articles de taille fixe. La taille des fichiers doit est connue à priori.**

III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

2.2- ORGANISATION ALÉATOIRE



III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

2.2- ORGANISATION ALÉATOIRE (FICHIER ALÉATOIRE - HACHAGE - HASHED FILE)

- **Le fichier est organisé en un nombre de paquets (buckets) de taille fixe (L) (correspondant à une action d'E/S). Chaque paquet peut contenir plusieurs articles. La clé d'accès (k) est la valeur d'un rubrique de l'article**
- **La taille du fichier doit être déterminée préalablement et la place pour le fichier doit être allouée dès sa création dans un l'espace consécultif sur le support. Cette taille est calculée en fonction de la taille d'un paquet et le nombre de paquet**
- **La fonction de distribution (fonction d'achage) f applique sur la clé k et rend le numéro du paquet dans lequel sera rangé l'article. Le choix de cette fonction est très important pour l'obtention d'une distribution uniforme des articles dans les paquets. Plusieurs techniques possibles :**

  - le pliage consiste à choisir en combiner des bits de la clé (par ou exclusif par exemple)

  - les conversions de la clé en nombre entier ou flottant avec utilisation de la mantisse permettant d'obtenir également un numéro de paquet

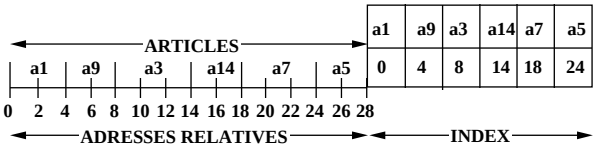
  - le modulo qui consiste à prendre pour numéro de paquet le reste de la division de la clé par le nombre de paquets
- **Le calcul de l'adresse relative du paquet (ARP) correspondant à la clé k est : ARP = L x f(k)**
- **Problème crutial : DÉBORDEMENT DE PAQUET**
- **AVANTAGE : dynamique, performante (sans débordements)**
- **INCONVÉNIENTS : risque de déborement, fichier de taille fixe**

III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

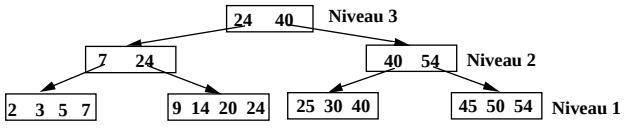
2.3- INDEX

- **INDEX : On appelle Index une "table de matière" permettant d'associer à chaque clé d'accès d'article l'adresse relative de cette article**



•INDEX HIÉRARCHISÉ

Un index à deux niveaux est un index trié divisé en paquets, possédant lui-même un index dont la clé d'accès à chaque paquet est la plus grande du paquet. Un index à n niveaux est un indes hiérarchisé à (n-1) niveaux possédant lui-même un index à un niveau



III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

2.3- INDEX

- **Le principe de base des organisations et méthodes d'accès indexés est d'associer à la clé d'accès d'un article son adresse relative dans un fichier à l'aide d'un ou plusieurs autres fichiers, appelés table de matière ou index**
- **Les étapes successives pour accèder à un article à l'aide de l'index sont :**

  - accès à l'index,

  - recherche sur la clé d'accès de l'article désiré afin d'obtenir l'adresse relative de l'article ou d'un paquet contenant l'article,

  - conversion de l'adresse relative en adresse absolue (physique)

  - accès à l'article (ou paquet d'articles) sur le support

  - transfert de l'article dans la zone du programme usager
- **L'index d'un fichier peut être trié. Cela permet d'une recherche dichotomique sur l'index**
- **Un index d'un fichier indexé peut contenir toutes les clés d'accès ou seulement certaines. Dans le premier cas on dit que l'index est dense et dans le seconde cas non-dense**
- **Dans le cas d'un index non dense, les articles du fichiers ainsi que l'index sont triés. Le fichier est lui-même divisé en paquets de taille fixe et chaque paquet correspond à une entrée en index contenant un doublet**

< max(les clés du paquet), adresse relative du paquet>

III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

2.4- STRUCTURE D'ARBRE BALANCÉ

• Un arbre balancé de degré m est un arbre où :

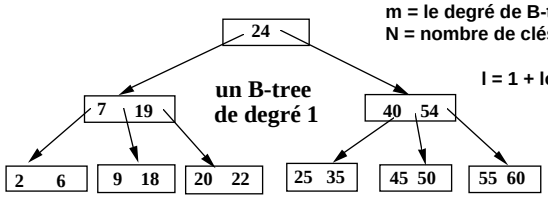
1- chaque noeud contient k clés triées dans l'ordre croissant, avec m Š k Š 2m, sauf la racine qui contient entre 1 et 2 m clés

2- un noeud est soit terminal, soit possède (k+1) fils; le ième fils possède des clés comprises entre la (i-1)ème et la ième clé du père

3- l'arbre est équilibré, c'est-à-dire que tous les noeuds terminaux sont au même niveau

l = le nombre de niveaux du B-tree
m = le degré de B-tree
N = nombre de clés à stocker dans le B-tree

l = 1 + logm+1((N+1)/2)



III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

2.4- STRUCTURE D'ARBRE BALANCÉ

- **Soient :** **- m le degré d'un arbre balancé**
  - l le nombre de niveaux de cette arbre
  - N le nombre minimum de clés stockées dans l'arbre

Dans le pire des cas si l'arbre est rempli au minimum, il existe:

- une clé à la racine, (niveau 1)
- deux branches avec m clés (niveau 2)
- (m+1) branches avec m clés ((l-2) niveaux)

Cela permet de déduire que : N = 1 + 2m (1 + (m+1) + (m+1)2 + ...+ (m+1)l-2)
= 1 + 2m (((m+1)l-1 -1)) / ((m+1)-1))
= 1 + 2((m+1)l-1 -1) = 2(m+1)l-1 -1

III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

2.4- STRUCTURE D'ARBRE BALANCÉ

- **Pour stocker N clé dans un arbre balancé d'ordre m, on a besoin au maximum de niveaux avec : l = 1 + log m+1 ((N + 1)/2)**

• avec N = 1 999 999 et m = 99, on a
l Š 1 + log100 1003 = 4 (4 opérations d'E/S maximum)

- **Deux manières d'utilisation de B-tree pour la réalisation des fichiers à index hiérarchisés :**

  - l'arbre balancé est utilisé pour les index uniquement : les noeuds feuilles contiennent toutes les clés d'accès aux articles et leur adresse relative correspondant, les feuilles intermédiaires contiennent les clés d'accès aux noeuds fils et leur adresse relative. Les articles sont stocker dans les fichiers classiques séparés du fichier d'index (série 3 de l'IBM)

  - l'arbre balancé est utilisé pour le fichier et index : les clés d'accès et les articles sont dans le même B-tree. Les clés sont dans les noeuds intermédiaires et les articles dans les noeuds feuilles (VSAM de l'IBM)

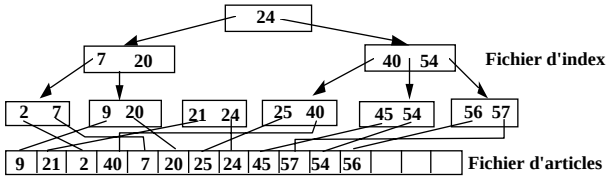
III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

2.5- ORGANISATIONS DE L'IBM

- **ORGANISATION DE SÉRIE 3 IBM**

1- Le fichier indexé est un fichier séquentiel, non trié
2- Le fichier d'index est trié, dense et sous-forme d'un B-tree



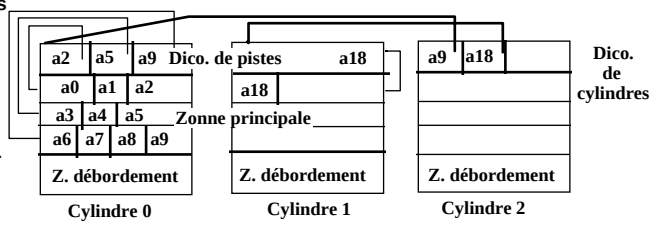
III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

2.5- ORGANISATIONS DE L'IBM

• ORGANISATION ISAM (Idexed Sequential Access Method)

1- Un seul fichier pour l'index et des articles
2- Les articles triés, l'index trié, non-dense
3- Le fichier comporte 3 zonnes logiques :
- zonne primaire : les articles à la première écriture dans de cylindres consécultifs dont certaines pistes sont réservées pour la zonne d'index et celle de débordement
- zonne de débordement où on transfère les articles lors des additions au fichier
- deux zonnes de débordement : une locale à chaque cylindre, et une globale dans 2 derniers cylindres
- zonne d'index : 1 index de pistes pour chaque cylindre (max(clé), adrpiste), 1 index de cylindres pour le fichier, 1 index global pour le débordement de l'index de cylindres)



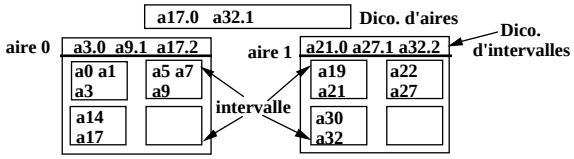
III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

2.5- ORGANISATIONS DE L'IBM

• ORGANISATION VSAM (Virtual Sequential Access Method)

1- Un seul fichier pour l'index et des articles
2- Les articles triés, l'index trié, non-dense sous forme B-tree
3- Le fichier est divisé en aires, dont
un aire = {intervalles} = {pistes de même cylindre ou de cylindres contigüs}
intervalle = partie de piste/{piste} correspondant à 1 E/S
4- même technique que celle d'ISAM en absence de la zonne de débordement car un intervalle ou un aire s'éclate automatiquement en deux s'il y a un débordement (B-tree)



III- PERFORMANCES ET STRUCTURES PHYSIQUES

2- Quelques méthode d'accès sélectifs

2.6- ORGANISATION DE CLUSTER

- **CLUSTER est une organisation de données permettant de regrouper physiquement de lignes de plusieurs tables ayant un lien logique entre elles**
- **les lignes sont regroupées suivant une même collonne dans un même bloc physique**
- **les clusters constituent une pré-jointure (jointure physique) sur disque**
- **CLUSTER entre deux relations département et enmployé sur la clé du département (clé de cluster)**

[Département]

1	d1	21	92	paris				

[Employé]

1	j	20	c1	21	90	20	2	1
2	a	25	c3	1	91	7	0	1
5	f	37	c2	1	87	8	0	1

IV- CONTRÔLE D'ACCÈS CONCURRENTS

- Concepts de base
- Exécutions sans conflit et sérialisables
- Contrôle d'exécution sérialisable : estampilles
- Contrôle d'exécution sérialisable : verrouillage à deux phases
- Contrôle d'exécution sérialisable : autres algorithmes

IV- CONTRÔLE D'ACCÈS CONCURRENTS

1- CONCEPTS DE BASE

1.1- Notion de cohérence

- **CONTRAINTE D'INTÉGRITÉ :** On appelle contrainte d'intégrité dans une base de données toute assertion qui doit être vérifiée par les données à des instants déterminées

- **Exemple :** Base de données du personnel d'une entreprise

  - Contraintes de domaine : "Tout employé doit avoir au moins 18 ans"
  - Contraintes de mise à jour : "Tout salaire ne peut être diminué"
  - Les dépendances : - fonctionnelles (clé primaire)
    - multivalués
    - d'inclusion (clé étrangère)
  - Contraintes arithmétiques, temporelle ... : Sol = Crédit - Débit

- **BASE DE DONNÉES COHÉRENTE :** Toute base de données dont l'insemble des contraintes d'intégrité (explicites ou implicites) est respecté par les données de la base est dit cohérente ou dans l'état cohérent

IV- CONTRÔLE D'ACCÈS CONCURRENTS

1- CONCEPTS DE BASE

1.2- Concept de transaction

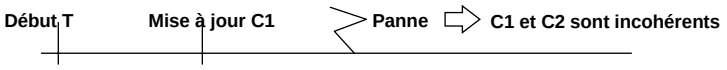
- **TRANSACTION : On appelle transaction l'unité de traitement séquentiel, exécutée pour le compte d'un utilisateur, appliquée à une base de données cohérente, restitue une base de données cohérente**

- **PROPRIÉTÉS :**

  - Une transaction est une unité logique atomique de traitement qui est soit complètement exécutée soit complètement abandonée

  - Si une transaction ne va pas à son terme pour une raison ou pour une autre, la base de données est restaurée dans l'état où elle se trouvait avant que la transaction ne démarre

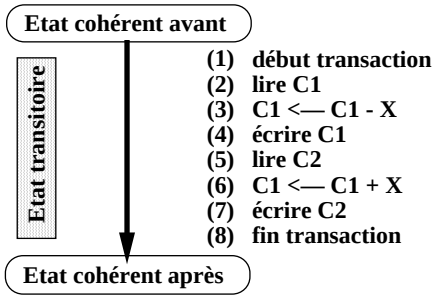
- **Exemple: Ecriture comptable à double : T crédite C2 et débite C1**



IV- CONTRÔLE D'ACCÈS CONCURRENTS

1- CONCEPTS DE BASE

1.2- Concept de transaction



IV- CONTRÔLE D'ACCÈS CONCURRENTS

1- CONCEPTS DE BASE

1.3- Vie d'une transaction

- **VIE SANS HISTOIRE**

  Une transaction s'exécute normalement jusqu'à la fin. Elle se termine par une instruction de validation de tous les effets qu'elle a produit sur la base de données.On dira que cette transaction est validée : toutes les modifications sur la base de données permettent la constitution de la nouvel état de cohérence de la base

- **UN ASSINSSINAT**

  Un évenement extérieur vient interrrompre l'exécution de la transaction de façon irrémédiable. Cet arrêt de la vie d'une transaction peut provenir soit d'une panne soit d'une action délibérée de la part du SGBD qui décide de supprimer telle ou telle transaction

- **UN SUICIDE**

  Aucours de son exécution, la transaction détecte certaines conditions qui font que la poursuite de son exécution s'avère impossible. Elle peut, dans ce cas, décider de se supprimer en sexécutant une instruction d'annulation

IV- CONTRÔLE D'ACCÈS CONCURRENTS

1- CONCEPTS DE BASE

1.4- Problèmes de concurrence d'accès

- **EXEMPLE ILLUSTRE**

  Soient T1 et T2 deux transactions qui s'intéressent à un même objet A. Etant donnée "lire" et écrire deux seules opérations possibles sur A, nous avons 4 cas possibles

  - **Lecture-Lecture : Aucune conflit : un même objet peut toujours être partagée en lecture**
  - **Ecriture-Ecriture : Ce cas peut induire des pertes de mises à jour :**

  T2 vient "écraser" par une autre écriture celle effectuée par T1

  sans tenir compte cette dernière

temps	T1	T2	A
t1	lire A	-	A = 10
t2	-	lire A	
t3	A:= A+10	-	
t4	-	-	
t5	-	A:= A+50	
t6	écrire A	-	A = 20
t7	-	écrire A	A = 60

IV- CONTRÔLE D'ACCÈS CONCURRENTS

1- CONCEPTS DE BASE

1.4- Problèmes de concurrence d'accès

- **EXEMPLE ILLUSTRE**

• Ecriture-Lecture : lectures impropre

T2 lit une valeur modifiée par T1 et ensuite T1 est annulée. On a une valeur "impropre" de A dans T2

temps	T1	T2	A
t1	lire A	-	A = 10
t2	A:=A+20	-	
t3	écrire A	-	A = 30
t4	-	lire A	
t5	Annulation	-	A = 10
t6	-	-	

• Lecture-Ecriture : lectures non reproductives : T1 modifie la valeur de A entre deux lectures de T2

temps	T1	T2	A
t1	lire A	-	A = 10
t2	A:=A+20	lire A	
t3	A:=A+10	-	
t4	écrire A	-	A=30
t5	-	-	
t6	-	lire A	

IV- CONTRÔLE D'ACCÈS CONCURRENTS

2- EXÉCUTIONS SANS CONFLIT ET SÉRIALISABLES

2.1- Concepts de base

- **CONTRÔLEUR (SCHEDULER) : Un contrôleur transactionnel est un module système chargé de contrôler les accès concurrent aux données**
- **GRANULE : On appelle granule l'unité de base de données dont l'accès est contrôlé individuellement par un contrôleur**
- **OPÉRATION : Une opération est une suite d'actions élémentaires accomplissant une fonction sur un granule en respectant sa cohérence interne**
- **RÉSULTAT DE L'OPÉRATION : Le résultat d'une opération est l'état du granule concerné après l'application de l'opération considérée anssi que les effets de bords provoqués par l'opération**
- **EXÉCUTION DE TRANSACTIONS (SCHEDULE - LOG) : Une exécution des transactions (T1, T2, ..., Tn) est une séquence d'actions élémentaires obtenues en interclassant les diverses actions des transactions concernant**

IV- CONTRÔLE D'ACCÈS CONCURRENTS

2- EXÉCUTIONS SANS CONFLIT ET SÉRIALISABLES

2.1- Concepts de base

- **Exemple**

T1	T2
Lire A → a1	Lire A → a2
a1 + 1 → a1	a2 * 2 → a2
Ecrire a1 → A	Ecrire a2 → A
Lire B → b1	Lire B → b2
b1 + 1 → b1	b1 * 2 → b2
Ecrire b1 → B	Ecrire b2 → B

EXÉCUTION 1	EXÉCUTION 2
T1: Lire A → a1	T2: Lire A → a2
T1: a1 + 1 → a1	T2: a2 * 2 → a2
T1: Ecrire a1 → A	T1: Lire A → a1
T2: Lire A → a2	T1: a1 + 1 → a1
T2: a2 * 2 → a2	T2: Ecrire a2 → A
T2: Ecrire a2 → A	T2: Lire B → b2
T1: Lire B → b1	T2: b1 * 2 → b2
T1: b1 + 1 → b1	T1: Ecrire a1 → A
T1: Ecrire b1 → B	T1: Lire B → b1
T2: Lire B → b2	T1: b1 + 1 → b1
T2: b1 * 2 → b2	T1: Ecrire b1 → B
T2: Ecrire b2 → B	T2: Ecrire b2 → B

IV- CONTRÔLE D'ACCÈS CONCURRENTS

2- EXÉCUTIONS SANS CONFLIT ET SÉRIALISABLES

2.2- Exécution sérialisable

- **SUCCESSION (SERIAL SCHEDULE)**

Une exécution E de T1, .., Tn est une succession s'il exite une permutation ' de (1, 2, ..,n) telle que E = < Tš(1), Tš(2), .., Tš(n) >

- **PROPRIÉTÉ**

Une succession est une exécution sans perte d'opératons ni inconsistances

- **EXÉCUTION SÉRIALISABLE (SERIALIZABLE SCHEDULE)**

Une exécution quelconque de T1, .., Tn est dit sérialisable si et seulement si elle donne pour chaque transaction participante le même résultat qu'une succession de T1, .., Tn

IV- CONTRÔLE D'ACCÈS CONCURRENTS

2- EXÉCUTIONS SANS CONFLIT ET SÉRIALISABLES

2.3- Propriétés des opérations

- **OPÉRATIONS COMPATIBLES : Deux opérations O1, O2 sont compatibles si et seulement si quel que soit l'exécution simultanée de O1 et O2, le résultat de cette exécution est le même que celui de l'exécution séquentielle de O1 suivie de O2 ou de O2 suivie O1**
- **PROPRIÉTÉ : Deux opérations travaillant sur deux granules différents sont toujours compatibles**
- **Note : En général, le résultat de <O1, O2> peut être différent de celui de <O2, O1>**
- **OPÉRATIONS PERMUTABLES : Deux opérations O1, O2 sont permutables si et seulement si toute exécution de O1 suivie O2 donne le même résultat que celle de O2 suivie O1**
- **Remarque**

  Il existe des opérations compatibles mais non permutables et inversement

IV- CONTRÔLE D'ACCÈS CONCURRENTS

2- EXÉCUTIONS SANS CONFLIT ET SÉRIALISABLES

2.3- Propriétés des opérations

- **EXEMPLE :**

O11- Lire A → a1 **a1 + 1 → a1** **Print a1**	**O21- Lire A → a2** **a2 * 2 → a2** **Ecrire a2 → A**
O12- Lire A → a1 **a1 + 1 → a1** **Ecrire a1 → A**	**O22- Lire A → a2** **a2 * 2 → a2** **Ecrire a2 → A**
O13- Lire A → a1 **a1 + 1 → a1** **Ecrire a1 → A**	**O23- Lire A → a2** **a2 + 10 → a2** **Ecrire a2 → A**

- O11 et O21 sont compatibles

- O13 et O23 sont permutables

- O12 et O22 ne sont pas compatibles ni permutables

IV- CONTRÔLE D'ACCÈS CONCURRENTS

2- EXÉCUTIONS SANS CONFLIT ET SÉRIALISABLES

2.4- Caractéristiques des exécutions sérialisables

- **TRANSFORMATIONS DE BASE**

- **Séparation d'opérations : La séparation d'opérations consiste à isoler des couples des opérations compatibles dans une éxécution et de remplacer chaque couple E(Oi, Oj) par la séquence donnant le même résultat : soit <Oi, Oj> soit <Oj, Oi>**
- **Permutation d'opérations : La permutation d'opérations consiste à isoler les couples des opérations permutables et de changer l'ordre d'exécution des opérations dans un même couple**

- **PROPRIÉTÉ**

  Les transformations de base : sélection d'opérations et permution d'opérations conservent le résultat d'une exécution

- **1e CONDITION SUFFISANTE (de l'exécution sérialisable)**

  Une condition suffisante pour qu'une exécution soit sérialisable est qu'elle puisse être transformée par séparation des opérations compatibles et permutation des opérations permutables en une succession des transactions composantes

IV- CONTRÔLE D'ACCÈS CONCURRENTS

2- EXÉCUTIONS SANS CONFLIT ET SÉRIALISABLES

2.4- Caractéristiques des exécutions sérialisables

- **EXEMPLE**

	O2 et O3 sont permutables. On obtient donc <T1, T2> :
O1:	**O1:**
T1: Lire A → a1	**T1: Lire A → a1**
T1: a1 + 1 → a1	**T1: a1 + 1 → a1**
T1: Ecrire a1 → A	**T1: Ecrire a1 → A**
O2:	**O3:**
T2: Lire A → a2	**T1: Lire B → b1**
T2: a2 * 2 → a2	**T2: b1 * 2 → b2**
T2: Ecrire a2 → A	**T1: Ecrire b1 → B**
O3:	**O2:**
T1: Lire B → b1	**T2: Lire A → a2**
T1: b1 + 1 → b1	**T2: a2 * 2 → a2**
T1: Ecrire b1 → B	**T2: Ecrire a2 → A**
O4:	**O4:**
T2: Lire B → b2	**T2: Lire B → b2**
T2: b1 * 2 → b2	**T2: b1 * 2 → b2**
T2: Ecrire b2 → B	**T2: Ecrire b2 → B**

L'exécution précédente est une exécution sérialisable

IV- CONTRÔLE D'ACCÈS CONCURRENTS

2- EXÉCUTIONS SANS CONFLIT ET SÉRIALISABLES

2.4- Caractéristiques des exécutions sérialisables

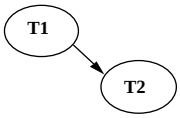
- **GRAPHE DE PRÉCÉDENCE : Soit S une exécution sans opération simultanée. On dit que Ti précéde Tj dans S, et noté Ti < Tj, si et seulement s'il existe deux opérations non permutables Oi et Oj telles que Oi est exécutée par Ti et Oj est exécutée par Tj**

- **2e CONDITION SUFFISANTES (de l'exécution sérialisable)**

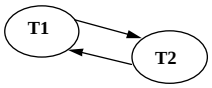
  Une condition suffisante pour qu'une exécution soit sérialisable est que le graphe de précédence associé soit sans circuit

- **EXEMPLE**

Sérialisable :
T1: a1 + 1 Æ a1
T2: a2 * 2 Æ a2
T1: b1 + 1 Æ b1
T2: b1 * 2 Æ b2



Non sérialisable :
T1: a1 + 1 Æ a1
T2: a2 * 2 Æ a2
T2: b1 * 2 Æ b2
T1: b1 + 1 Æ b1



IV- CONTRÔLE D'ACCÈS CONCURRENTS

3- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES D'ORDONANCEMENT INITIAL DES TRANSACTIONS PAR ESTAMPILLES

3.1- Concepts de base

- **MÉTHODE D'ORDONNANCEMENT INITIAL**

  Une première méthode pour empêcher la production d'une exécution non sérialisable consiste à ordoner les transaction lors leur lancement en exécution et à contrôler que l'accès conflituels aux granules s'effectuent dans l'ordre défini

- **ESTAMPILLE DE TRANSACTION**

  L'estampille d'une transaction est une valeur numérique affectée à la transaction au moment ce son lancement. Ce numéro est une référence unique pendant la vie de la transaction permettant d'ordonner cette transaction par rapport aux autres transactions

- **ESTAMPILLE DE GRANULE**

  L'estampille d'un granule est une valeur numérique associée à ce granule mémorisant l'estampille de la dernière transaction ayant opéré sur ce granule

IV- CONTRÔLE D'ACCÈS CONCURRENTS

3- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES D'ORDONANCEMENT INITIAL DES TRANSACTIONS PAR ESTAMPILLES

3.2- Algorithme d'ordonnacement total

- **PRINCIPE**

- Cet algorithme consiste à vérifier que les accès aux granules par les transactions s'effectuent bien dans l'ordre affecté au lancement et répéré par les estampilles de transactions

- Si deux transactions Ti d'estampille i et Tj d'estampille j avec i inférieur à j opèrent sur un même granule, le contrôleur vérifie si Ti précéde Tj sur ce granule. Dans le cas contraire, le contrôleur provoque la reprise d'une des deux transactions

- **ALGORITHME**

```
Procédure LIRE(Ti, g);
        Si      (E(g) Š i) alors
                "exécuter la lecture";
                E(g) = i;
        Sinon   ABORT;
        finsi;
Fin LIRE;
```

```
Procédure ECRIRE(Ti, g);
        Si      (E(g) Š i) alors
                "exécuter l'écriture";
                E(g) = i;
        Sinon   ABORT;
        finsi;
Fin ECRIRE;
```


IV- CONTRÔLE D'ACCÈS CONCURRENTS

3- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES D'ORDONANCEMENT INITIAL DES TRANSACTIONS PAR ESTAMPILLES

3.3- Algorithme d'ordonnacement partiel

- **PRINCIPE**

- Il est nécessaire d'ordonner les seules opérations NON PERMUTABLE sur les granules LIRE/ECRIRE - ECRIRE/LIRE - ECRIRE/ECRIRE

- L'algorithme partiel consiste à vérifier que les séquences des opérations non permutables s'effectuent bien dans l'ordre des estampilles affectées au lancement des transactions

- Il faut donc distinguer deux types d'estampilles associées aux granules : EL en lecture, EE en écriture

- Si une transaction Ti demande une lecture sur un granulle déjà demandé en écriture par une transaction Tj, alors i doit être supérieur à j

- Si une transaction Ti demande une écriture sur un granulle déjà demandé en écriture par une transaction Tj et en lecture par une transaction Tk, alors i doit être supérieur à j et à k

IV- CONTRÔLE D'ACCÈS CONCURRENTS

3- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES D'ORDONANCEMENT INITIAL DES TRANSACTIONS PAR ESTAMPILLES

3.3- Algorithme d'ordonnacement partiel

- **ALGORITHME**

```
Procédure LIRE(Ti, g);
    Si (EE(g) Š i) alors
        "exécuter la lecture";
        EL(g) = MAX(EL(g), i);
    Sinon ABORT; finsi;
Fin LIRE;
```

```
Procédure ECRIRE(Ti, g);
    Si ((EL(g) Š i) et (EE(g) Š i)) alors
        "exécuter l'écriture";
        E(g) = i;
    Sinon ABORT; finsi;
Fin ECRIRE;
```

- **RÈGLE D'ÉCRITURE DE THOMAS : "si EE(g) > i, on ignore l'écriture de Ti (écriture dépassée) mais ne supprime pas la transaction"**

```
Procédure ECRIRE(Ti, g);
    Si (EL(g) Š i)
        si (EE(g) Š i) alors
            "exécuter l'écriture";
            E(g) = i;
        finsi;
    Sinon ABORT; finsi;
Fin ECRIRE;
```


IV- CONTRÔLE D'ACCÈS CONCURRENTS

3- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES D'ORDONANCEMENT INITIAL DES TRANSACTIONS PAR ESTAMPILLES

3.4- Algorithme d'ordonnacement partiel multi-version

- **PRINCIPE**

- **Pour chaque granule g, le système mantient :**
  - **un ensemble d'estampilles écriture {EEi(g)} avac les valeurs associées {Vi(g)}**
  - **un ensemble d'estampille lecture {ELi(g)}**
- **L'algorithme partiel multi-version consiste à éviter la reprise de transaction en lecture de granulle.**
- **Si Ti demande une lecture sur g, il suffit de rechercher la version de valeur écrite par une transaction Tj immédiatement inférieure à i. Ainsi Ti précédera toutes les transactions d'estampilles supérieures écrivant sur le granule g, et suivra celle d'estampilles inférieures. Ti est correctement séquencée.**
- **Si une transaction Ti demande une écriture sur g, deux politiques sont possibles :**
  - **insérer la valeur écrire par Ti dans la liste des versions juste après celle d'estampille immédiatement inférieure, soit Vj(g). Ensuite reprendre la transaction Tk (k > i) ayant lu la valeur Vj(g) si elle exite**
  - **inserer la valeur écrire par Ti dans la liste des versions juste après celle d'estampille immédiatement inférieure, soit Vj(g), seulement dans le cas où Vj(g) n'est pas lue par une autre transaction Tk (k > i). Dans le cas contraire, reprendre Ti**

IV- CONTRÔLE D'ACCÈS CONCURRENTS

3- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES D'ORDONANCEMENT INITIAL DES TRANSACTIONS PAR ESTAMPILLES

3.4- Algorithme d'ordonnacement partiel multi-version

- **ALGORITHME**

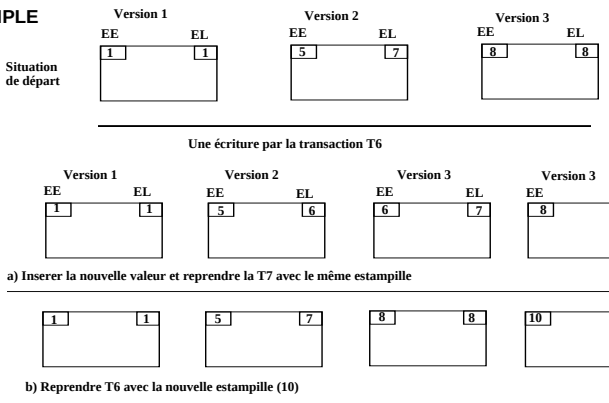
```
Procédure LIRE(Ti, g);
        j := "numéro de la dernière version de g" ;
        Tant que (EEj(g) > i) faire j := j - 1;  fin-tant-que;
        "exécuter la lecture de la version j de g";
        ELj(g) = MAX(ELj(g), i);
Fin LIRE;
Procédure ECRIRE(Ti, g);
        j := "numéro de la dernière version de g" ;
        Tant que (EEj(g) > i) faire j := j - 1; fin-tant-que;
        Si      (ELj(g) Š i) alors
                "exécuter l'écriture en inserant une version j+1 de g";
                EEj+1(g) = i;
        Sinon   ABORT;   finsi;
Fin ECRIRE;
```


IV- CONTRÔLE D'ACCÈS CONCURRENTS

3- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES D'ORDONANCEMENT INITIAL DES TRANSACTIONS PAR ESTAMPILLES

3.4- Algorithme d'ordonnacement partiel multi-version

- **EXEMPLE**



IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.1- Principes de base

- **STRATÉGIES**
- **Les stratégies de verrouillage consistent à éviter la génération d'exécution incorrecte en faisant attendre les transactions voulant exécuter des opérations conflictuelles sur le même granule**
- **Le premier objectif des algorithmes de verrouillage es de ne laisser s'exécuter simultanément ques des opérations compatibles**
- **MODE D'OPÉRATION**
- **Les modes d'opération sont des caractéristiques permettant de classer une opération et de déterminer ses compatibilités avec les autres opérations**
- **Les modes simples : m0=LIRE 1 0 m1=ECRIRE 0 0**
- **Les modes définis par le groupe DBTG CODASYL :**
  - **M1 = consultation non protégée 1 1 1 1 0 0**
  - **M2 = consultation protégée 1 1 0 0 0 0**
  - **M3 = mise à jour non protégée 1 0 1 0 0 0**
  - **M4 = mise à jour protégée 1 0 0 0 0 0**
  - **M5 = consultation exclusive0 0 0 0 0 0**
  - **M6 = mise à jour exclusive 0 0 0 0 0 0**

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.2- Protocole de verrouillage

- **DESCRIPTION**

- **Un protocole de verrouillage est un protocole d'accès à des granules partagés caractérisé par des demandes d'autorisation d'opérations et de signalements de fin d'opérations.**
- **Un protocole de verrouillage, en général, se compose de deux actions spéciales :**
  - **LOCK (g, M) permet à une transaction d'indiquer au contrôleur le début d'une opération correspondant au mode M sur le granule g**
  - **UNLOCK (g) permet à une transaction d'indiquer au contrôleur la fin d'une opération encours sur le granule g**
- **Le protocole de verrouillage nécessite donc l'exécution de deux actions supplémentaires LOCK et UNLOCK. Ce deux actions peuvent être automatiquement ajoutées par le contrôleur, chaque fois qu'il exécute une opération. Elles sont donc cachées aux utilisateurs finaux**
- **La plupart de système effectue le déverouillage soit à la fin de la transaction soit en des points intermédiaires où la base de données se trouve à l'état cohérent. Cela permet la mise en oeuvre des mécanismes de reprise en cas de pannes**

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.3- Algorithmes de verrouillage

- **PRINCIPES**

- **Mémoiriser les modes d'opérations en cours sur chaque granule (mettre des verrous)**

- **Pour cela, on associe, à chaque couple granule g et transaction Ti opérant sur ce granule, un vecteur de bits dont chacun représente le verrou d'un mode d'opération possible sur g par Ti**

$$A(g,i) = \begin{bmatrix} a1 \\ a2 \\ . \\ ak \end{bmatrix}$$

où : aj = 1 si le mode Mj est en cours d'exécution pour le compte de la transaction i sur le granule g, et 0 sinon

- **De manière simulaire, on peut présenter le mode M demandé lors de l'exécution d'une action LOCK (g,M) sur le granule g, par un vecteur de bits :**

$$M = \begin{bmatrix} m1 \\ m2 \\ . \\ mk \end{bmatrix}$$

où : mj = 1 si le mode Mj est demandé, et 0 sinon

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.3- Algorithmes de verrouillage

- **THÉOREME (de compatibilité)**

Les modes d'opérations demandées lors d'une primitive LOCK (g, M) exécutée par une transaction Ti sont compatibles avec les modes en cours d'exécution sur g par les autres transactions si et seulement si :

$$M \subset \neg \left(\neg C * \bigoplus_{k \neq i} A(g,k) \right)$$

où $\bigoplus$ **est la somme logique des vecteurs booléens**

$\neg$ **est la négation des matrices booléens**

C est la matrice de compatibilité

*** est le produit matriciel booléen (l'élément (i,j) de la matrice de produit est la somme booléen des multiplications des éléments de la lignes i^e de la matrice gauche et de j^e collonne de la matrice droite)**

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.3- Algorithmes de verrouillage

- **THÉOREME (de compatibilité)**

Preuve

- A(g,k) est un vecteur de bits dont le bit j est à 1 si et seulement si le mode Mj est en cours d'exécution sur g par la transaction k autre que Ti. ¬C est le complément de la matrice compatibilité.

- On a donc:

¬C * A(g,k) est un vecteur de bits dont le bit j est à 1 si et seulement si le mode Mj est incompatible avec un mode en cours d'exécution sur le granule g par une transaction autre que Ti.

- Finalement, ¬(¬C * A(g,k)) est donc un vecteur de bits représentant tous les modes d'opérations compatibles avec les modes en cours d'exécution sur le granule g par une transaction autre que Ti. Tout vecteur inclus dans ce dernier correspond donc à des modes compatibles avec ceux en cours d'exécution sur g par une transaction autre que Ti. (CQFD)

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.3- Algorithmes de verrouillage

- **ALGORITHME :**

Supposons que Q[g] est la file d'attente associée au granule g. Cette fille d'attente contient les demandes de verrouillage d'une transaction k en mode M : (k, M). On peut donc établir les algorithmes suivants :

```
Procédure LOCK (g, M) ;
    si      M ⊂ ¬ ( ¬C *    A(g,k))
                          k°i
    alors A(g,i) := A(g,i) + M ;
    sinon  " insérer (i,M) dans Q[g] " ;
           " bloquer la transaction Ti ";
    finsi;                          1
fin LOCK;
```

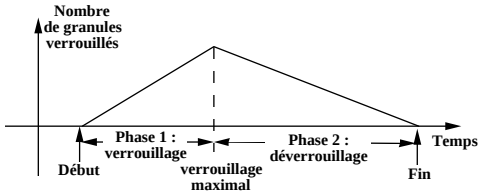
```
Procédure UNLOCK (g) ;
    A(g, i) := (0) ;
    Pour chaque (q,M') de Q[g] faire
       si  M' ⊂ ¬ ( ¬C *    A(g,k))
                          k°q
       alors  A(g,q) := A(g,q) + M ;
              "extraire (q, M') de Q[g]";
               "débloquer la transaction Tq";
       finsi;
    fin pour;
fin UNLOCK;
```

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.4- Algorithmes de verrouillage à deux phases

- **RESTRICTION DEUX PHASE : On appelle transaction deux-phases, toute transaction qui n'exécute pas de LOCK après avoir exécuté UNLOCK**



- **THÉORÈME : DE SÉRIALISABILITE**

  Toute exécution complète d'un ensemble de transaction deux-phases est sérialisable

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.4- Algorithmes de verrouillage à deux phases

- **Preuve : (par l'absurde)**
- **Une exécution sérialisable est une exécution dont le graphe de précédence ne possède pas de circuits.**
- **Considérons alors une exécution complète d'un ensemble :**

  [T1, T2, .., Tn] de transactions deux phases et supposons qu'il existe un circuit de précédence : Ti1 → Ti2,...→..., Tin → Ti1
- **Il découle immédiatement que :**

  - Ti2 "LOCK" un granule gi1 après que Ti1 "UNLOCK" ce granule gi1,

  - Ti3 "LOCK" un granule gi2 après que Ti2 "UNLOCK" ce granule gi2,

  - ...

  - Ti1 "LOCK" un granule gin après que Tin "UNLOCK" ce granule gin.
- **Chaque transaction Ti2, Ti3, ... étant deux-phase, elle exécute "UNLOCK" seulement lorsqu'elle a terminé tous ses "LOCK".**

  - De ce fait, Ti1 exécute "UNLOCK" sur gi1 avant d'exécuter "LOCK" sur gin. Par définition elle n'est donc pas une transaction deux phases. Ce qui est contraire aux hypothèses.

(CQFD)

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.4- Algorithmes de verrouillage à deux phases

- **RÈGLES D'UTILISATION DE "LOCK/UNLOCK"**

R1- Toute transaction doit exécuter LOCK avec le(s) mode(e) d'opération correct(s) et le granule choisi avant d'exécuter(une opération sur ce granule;

R2- Toute transaction doit exécuter UNLOCK sur le granule choisi plus ou moins longtemps après l'exécution de l'opération;

R3- Toute transaction ne peut exécuter LOCK après avoir exécuté UNLOCK

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

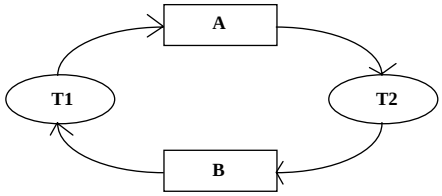
4.5- Problème de verrou mortel (interblocage)

- **INTERBLOCKAGE (Deadlock)**

  Le verrou mortel ou interblocage est la situation à laquelle on aboutit lorsque des granules ont été verrouillés dans un ordre tel qu'un groupe de transactions vérifie les deux propriétés suivantes :

  1- Chaque transaction du groupe est blocquée en attente d'un granule;

  2- L'exécution supposée de toutes les transactions n'appartenant pas au groupe ne permet pas de débloquer une des transactions du groupe



SOLUTIONS : la détection, la prévention

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

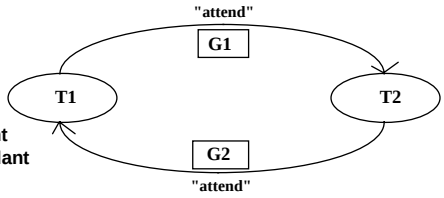
4.5- Problème de verrou mortel (interblocage)

- **GRAPHE DES ATTENTES (Murphy 68)**

  Le graphe des attentes est un graphe G(T, G) où

  - T est l'ensemble de transactions concurrentes [T1, T2, .., Tn] se partageant les granules G1, G2, .., Gm, et

  - G est la relation "attend" définie par : Tp "attend" Tq si et seulement si Tp attend le verrouillage d'un granule gi qui est déjà verrouillé par Tq



- **THÉORÈME (Murphy 68)**
  Il existe une situation d'interblocage dans une exécution E si et seulement si le graphe des attentes correspondant à E possède un circuit

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.5- Problème de verrou mortel (interblocage)

- **GRAPHE DES ALLOCATIONS (HOLT 72)**

  Le graphe des allocations est un graphe qui se compose de deux ensembles de sommets :

  - l'ensemble de transactions concurrentes [T1, T2, .., Tn] et

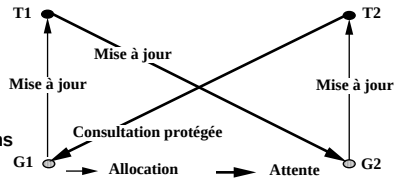
  - l'ensemble de granules G1, G2, .., Gm partagés.

- **Un arc relie le granule Gi à la transaction Tp si et seulement si Tp**

  a obtenu un verrouillage sur Gi dans un mode d'opération; l'arc est valué par les modes d'opérations alloués.

- **Un arc relie la transaction Tp au granule Gi si et seulement si Tp a demandé et n'a pas encore obtenu l'allocation de ce granule; l'arc est valué par les modes d'opérations demandé**

- **THÉORÈME DE VERROU MORTEL (HOLT 72)**
  Une condition nécessaire d'existence d'interblocage est la présence d'un circuit sur le graphe des allocations

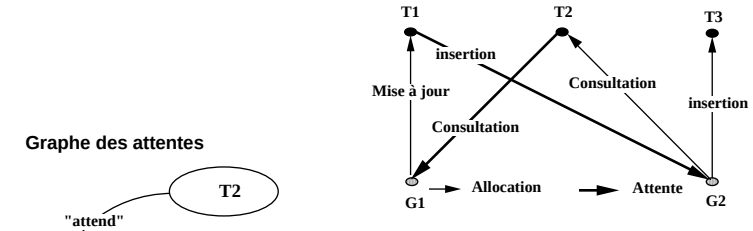


IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

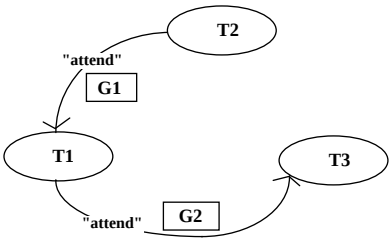
4.5- Problème de verrou mortel (interblocage)

- **EXEMPLE : graphe d'allocation avec circuit mais pas de verrou mortel**



Graphe d'allocation

Graphe des attentes



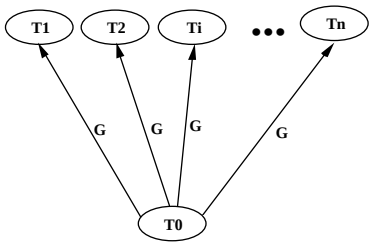
IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.6- Autres problèmes

- **PROBLÈME DE FAMINE (BLOCAGE PERMANENT)**

  Le problème de famine existe quand un groupe de transactions se coalise, en effectuant des opérations compatibles entre elles, contre une transaction individuelle qui désire effectuer une opération incompatible avec les précédentes. La transaction individuelle peut alors attendre indéfiniment



IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.6- Autres problèmes

- **PROBLÈME DES FANTÔMES (Eswaran 76)**

  Le problème des fantômes survient lorsqu'une entité est introduite dans la base de données et ne peut être traitée par une transaction en cours qui devrait logiquement la traiter

	Passager		Vol	
Nom	n° vol	n° siège	n° vol	nb passagers
Durant	100	10	100	4
Martin	100	5		
Durand	100	3		
Satre	100	14		

T1 : (1e partie) : lister la relation Passager (T1a)
T1 : (1e partie) : lister la relation Vol (T1b)
T2 : inserer dans Passager le tuple (Satre, 100, 14) et incrémenter le nombre de passagers du vol n° 100
- Hypothèse : Les transactions s'exécutent dans l'ordre E = (T1a, T2, T1b)

- E est une exécution valable mais le résultat de T1 est une liste de 3 nom alors que le nombre de passagers est 4. Satre est un "fantôme" dans cette exemple

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.7- Prévention des interblocages

- **PRINCIPE : La prévetion des interblocages consiste à supprimer l'une des conditions qui rend possible les situation de verrou mortel. On a de classes des techniques : préordonnancement des ressources (difficile à mettre en oeuvre) et préordonnancement des transactions**

- **METHODE DIE-WAIT : Quand une transaction Ti demande à verrouiller un granule qui est déjà verrouillé par une transaction Tj dans un mode incompatible, Ti attend Tj si et seulement si i < j (une transaction plus vieille attend une plus jeune). Dans le cas contraire Ti se suicide et elle sera reprise avec le même estampille (la jeune se suicide et la plus vielle ne mourras pas jamais)**

- **METHODE WOUND-WAIT : Quand une transaction Ti demande à verrouiller un granule qui est déjà verrouillé par une transaction Tj dans un mode incompatible, Ti attend Tj si et seulement si i > j (une transaction plus jeune attend une plus viellle). Dans le cas contraire Ti tue le Tj et elle sera reprise avec le même estampille (une plus vielle tue une plus jeune)**

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.8- Détection des interblocages

- **PRINCIPE : Un algorithme de détection de l'interblocage peut se déduire d'un algorithme de détection de circuits appliqué au graphe des attentes (ou des allocations). Soit N(k) le nombre de granules dont la transaction Tk attend le verrouillage :**

  1) sur le graphe des attentes G, un sommet est pendant si la transaction qu'il représente n'attend le verrouillage d'aucun granule. On peut réduire le graphe G en supprimer les sommets pendants (N(k) = 0)

  2) recaculer les N(k) sur le graphe G réduit : en comptant les demandes qui peuvent être satisfaites après la réduction et en décrémentant N(k) chaque fois que l'on compte une demande de transaction Tk.

  3) reviendre à 1) si les N(k) sont changés et G non vide, sinon 4)

  4) Si G est vide alors : Pas de circuits
  Sinon Circuit est détecté

IV- CONTRÔLE D'ACCÈS CONCURRENTS

4- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : ALGORITHMES DE VERROUILLAGE À DEUX PHASES

4.8- Détection des interblocages

- **ALGORITHME**

```
Booléenne Procedure SLOCK (gi, Q, K);
    "retourner VRAI si la demande Q de la transaction Tk, sur le granule gi, peut-être
    satisfaites compte tenu de l'état d'allocation des granules aux transactions présentes
    dans le graphe des attentes, et FAUX dans autres cas"
Fin SLOCK;
Booléenne Procédure DÉTECTER
    T = {liste des transaction Tj telque N(j) ° 0 } ;
    R = {liste des grannules verrouillés par les transactions de T} ;
    Pour "chaque entrée gi de R" faire
        Pour "chaque demande Q de Tk en attente de gi et non marquée" faire
            Si    SLOCK(gi, Q, k) = VRAI alors  "Marquer Q"; N(k) = N(k) - 1;
                  Si N(k) = 0 alors "Sortir Tk de T" ;
                                    "Ajouter le granules verrouillés par Tk à la liste R";
    Fin si;  Fin si;  Fin Pour; Fin Pour;
    Si T = Ø Alors    DETECTER = FAUX;  Sinon DETECTER = VRAI;  Fin si;
Fin DETECTER;
```

IV- CONTRÔLE D'ACCÈS CONCURRENTS

5- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : AUTRES ALGORITHMES

5.1- Ordonnancement par certification des lectures avant écriture (Kung 81)

- **PRINCIPES**

Au lieu d'ordonner les transactions par l'ordre de leur arrivée, la méthode de certification consiste à :

- considérer que chaque transaction se compose de deux étapes :
  - **une étape de lecture et calcul**
  - **une étape d'écriture réelle dans la base (commitment)**

- ordonner les transactions selon le moment où elles terminent la première étape de lecture et de calcul

- contrôler que les accès conflictuels aux granules s'effectuent bien dans l'ordre ainsi obtenu

IV- CONTRÔLE D'ACCÈS CONCURRENTS

5- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : AUTRES ALGORITHMES

5.1- Ordonnancement par certification des lectures avant écriture (Kung 81)

- **ALGORITHME**

- g est un granule, T est une transaction
- E(g) est l'estampille du granule g
- E(T, g) est l'estampille du granule g mémorisée lors de la lecture de g par T
- E(T) est l'estampille associée à la transaction T lors de l'ordonnancement

```
PROCÉDURE READ(T,g) ;
    "exécuter la lecture" ;
    ET (g) := E(g) ;
fin READ.
```

```
PROCÉDURE WRITE(T,g) ;
        Si E(T) > E(g) Alors "exécuter l'écriture" ;
        E(g) := E(T) ; fin si;
fin WRITE
```

```
PROCÉDURE CERTIFIER(T) ;
    Pour "chaque granule g lu par T" faire
        Si E(T,g) ° E(g) Alors ABORT(T); fin si;
        Si "il existe une transaction certifiée désirant écrire g"
        Alors ABORT(T); fin si;
    fin pour;
    E(T) := "horloge" ;
fin CERTIFIER.
```

IV- CONTRÔLE D'ACCÈS CONCURRENTS

5- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : AUTRES ALGORITHMES

5.1- Ordonnancement par certification des lectures avant écriture (Kung 81)

- **VADILITÉ DE L'ALGORITHME**

On peut démontrer que l'algorithme de certification permet :

- d'assurer que l'enregistrement des écritures est effectué dans l'odre de certification des transactions; les précédences WRITE/READ respectent donc l'ordre de certification des transaction

- d'assurer qu'une transaction ne peut lire les écritures d'une autre transaction que si la transaction écrivant a été certifiée et la transaction lisant ne l'a pas été, ceci par définition de CERTIFIER ; les précédences WRITE/READ respectent donc également l'ordre de certification des transactions

- d'assurer qu'une transaction certifiée avec succès ne peut avoir lu une entité ensuite écrite par une transaction certifiée avant elle ; les précédences WRITE/READ respectent donc également l'ordre de certification des transactions

- Finalement , toutes les opérations non permutables sont donc exécutées sur une même entité dans l'ordre de certification des transactions. Ceci assure l'absence de circuits dans le graphe de précédences et la correction de l'algorithme

IV- CONTRÔLE D'ACCÈS CONCURRENTS

5- CONTRÔLE D'EXÉCUTION SÉRIALISABLE : AUTRES ALGORITHMES

5.2- Ordonnancement au commitment de transaction (Viemont 82)

- **PRINCIPES**

- On peut remarquer que dans les systèmes actuels la fin de transaction est marquée par une action atomique (commit) qui provoque les écritures réelles dans la base. Il est possible alors de considérer deux estampilles de transactions :

- l'estampille initiale (heure de lancement)

- l'estampile finale (heure de commitment)

- A partir de ces deux estampilles, les conflits peuvent être résolus selon les pricipes suivants :

- conflit écriture-écriture par ordonnancement des écritures réelles au commitment à l'aide de la règle de THOMAS

- conflit lecture-écriture par reprise du lecture

- conflit écriture-lecture par attente du lecture jusqu'au commitment de l'écrivant

5.3- Détection de circuits dans le graphe de précédences

- coût élevé en taille du graphe (une transaction ne peut pas en être retirée aussitôt après la fin de da terminaison)