Modélisation et contraintes dans les bases de données

BDD Avancées

Nicolas Travers
DUT Informatique – 2° année

CNAM
nicolas.travers (at) cnam.fr

Plan

Objectifs du cours

- Comprendre le fonctionnement d'un SGBD : *Système d'information*
- Intégrer les intéractions avec le SGBD : *SQL*, *Processus*, *Programme*, *Concurrence*
- Evolution vers le NoSQL : *MongoDB*

Bibliographie

Ouvrages en français

1. Carrez C. *Des structures aux bases de données*. Masson.
2. Gardarin G. *Maîtriser les bases de données : modèles et langages*. Eyrolles.
3. Date C.J. *Introduction aux bases de données*. Vuibert.
4. Akoka J. et Comyn-Wattiau I. *Conception des bases de données relationnelles*. Vuibert Informatique.
5. Bales, D.K. *Java programming with Oracle JDBC*. O'Reilly, 2002.
6. Bizoi, R. *PL/SQL pour Oracle 10g*, Eyrolles. 2006.

Bibliographie

Ouvrages en anglais

1. R. Ramakrishnan et J. Gehrke. *Database Management Systems*. MacGraw Hill.
2. R. Elmasri, S.B. Navathe. *Fundamentals of Database Systems*. Addison Wesley.
3. Ullman J.D. and Widom J. *A First Course in Database Systems*. Prentice Hall.
4. H. Garcia-Molina, J.D. Ullman, J. Widom. *Database Systems : The Complete Book*. Prentice Hall.
5. Garcia-Molina H., Ullman J. and Widom J. *Implementation of Database Systems*. Prentice Hall.
6. Ullman J.D. *Principles of Database and Knowledge-Base Systems*. Computer Science Press.
7. Abiteboul S., Hull R., Vianu V. *Foundations of Databases*. Addison Wesley.

Bibliographie

Le standard SQL

1. Date C.J., *A Guide to the SQL Standard*, Addison-Wesley.

Quelques systèmes

1. Oracle (actuellement 12c),
2. DB2 (IBM),
3. SQL Server (Microsoft),
4. MySQL (SUN / Oracle),
5. PostgreSQL,
6. SQLite

Applications des bases de données

1. **Applications "classiques" :**
   - SRH : Salaire, Personnel,...
   - Commercial : Stock, vente, clients, réservation,...
   - Documentaire : Bibliothèque, tutoriaux
2. **Applications "modernes" :**
   - Documents électroniques : journaux
   - Web : commerce électronique, serveurs Web, blogs
   - Génie logiciel : gestion de programmes, manuels, ...
   - Documentation technique : plans, dessins, ...
   - Bases de données spatiales : cartes routières, systèmes de guidage GPS
   - Bases de données multimédia : archives audiovisuelles, imagerie médicale, ...

Problème central : comment stocker et manipuler les données ?

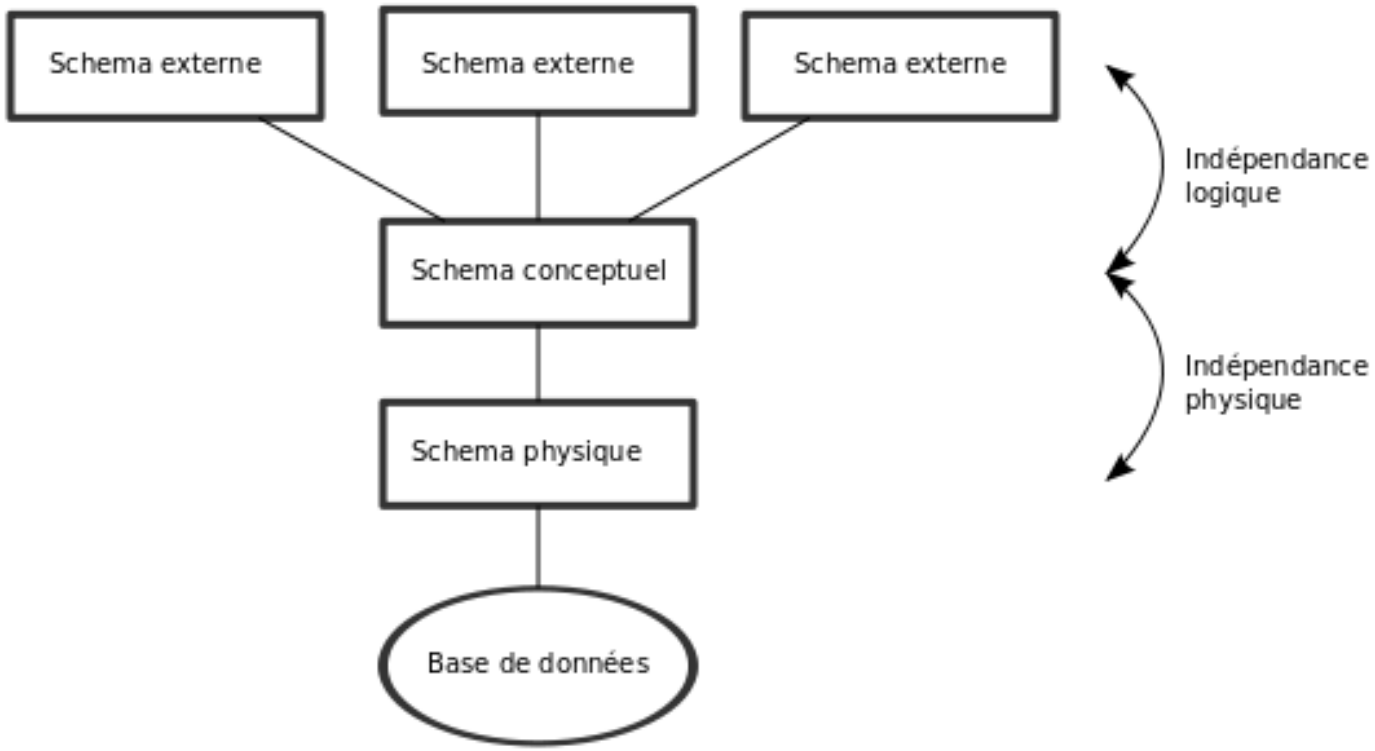
Une base de données est

- un grand ensemble de données
- structurées et
- mémorisées sur un support permanent

Un système de gestion de bases de données (SGBD) est

- un logiciel de haut niveau d'abstraction
- permet de
  - manipuler ces informations
  - gérer le stockage
  - gérer la concurrence

Niveaux d'architecture d'un SGBD : ANSI-SPARC (1975)



Architecture d'un SGBD : Niveau externe

Vues utilisateurs

Exemple : base(s) de données de l'ECP :

1. **Vue de la planification des salles :** pour chaque cours
   - Noms des enseignants
   - Horaires et salles
2. **Vue de la paye :** pour chaque enseignant
   - Nom, prénom, adresse, indice, nombre d'heures
3. **Vue du service de scolarité :** pour chaque étudiant
   - Nom, prénom, adresse, no étudiant, inscriptions aux cours, résultats

Architecture d'un SGBD : Niveau Logique

- Définition de la structure des données :
  Langage de Description de Données (LDD)
- Consultation et mise à jour des données :
  Langages de Requêtes (LR) et
  Langage de Manipulation de Données (LMD)

Architecture d'un SGBD : Niveau Physique

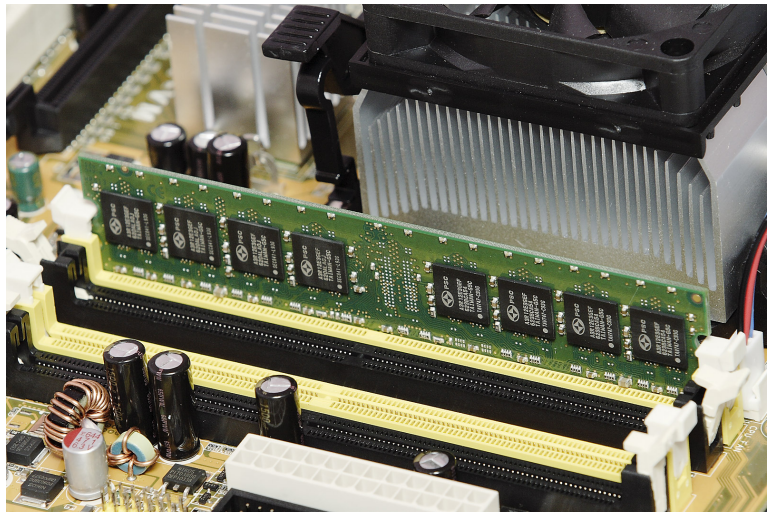
Organisation du Stockage

- Gestion des **fichiers** (Disque Dur)
- Gestion du **cache** (RAM)
- Gestion des fichiers de **logs**
- Gestion du **stockage** du niveau logique
- **Optimisation** des requêtes

Disque Dur

Temps d'accès : $\sim$9 ms

Random Access Memory



Temps d'accès : 15×10^{-6} ms

Architecture d'un SGBD : Thèmes abordés

- **Niveau externe**
  - Application (PL/SQL, ODBC)
- **Niveau logique**
  - Modélisation / Conception / Contraintes
  - Algèbre Relationnelle / Interrogation
  - Concurrence
- **Niveau physique**
  - Choix de stockage
  - Indexation
  - Optimisation de requêtes

  $\Rightarrow$ Cours de Bases de Données Avancées (M2 OSY)

En résumé

On veut gérer un grand volume de données

- structurées (ou semi-structurées),
- persistantes (stockage non volatile),
- cohérentes,
- fiables (protégées contres les pannes) et
- partagées entre utilisateurs et applications
- indépendamment de leur organisation physique

Historique des modèles SGBD

À chaque génération correspond un modèle logique

< 60	S.G.F. (e.g. COBOL)	
mi-60	HIÉRARCHIQUE IMS (IBM)	navigationnel
	RÉSEAU (CODASYL)	navigationnel
73-80	RELATIONNEL	déclaratif
mi-80	RELATIONNEL	explosion sur micro
Fin 80	ORIENTÉ-OBJET	navig. + déclaratif
	RELATIONNEL ETENDU	nouvelles applications
	DATALOG (SGBD déductifs)	pas encore de marché
Fin 90	XML	navig. + déclaratif
05-	NoSQL	micro-traitement
		+ distribution

La gestion du SGBD

- **Le concepteur**
  - évalue les besoins de l'application
  - conçoit le schéma logique de la base
- **L'administrateur du SGBD**
  - installe le système et créé la base
  - conçoit le schéma physique
  - fait des réglages fins (*tuning*)
  - gère avec le concepteur l'évolution de la base (nouveaux besoins, utilisateurs)
- **L'éditeur du SGBD**
  - fournit le système et les outils

Objectifs de la Modélisation

- Meilleure compréhension du système
  - Abstraction du problème
  - Se concentre sur l'essentiel (pas de détails)
  - Visualisation du système simple
- Conception progressive
  - Raffinement du modèle successif
  - Découpage en vues

Objectifs de la Modélisation

- Meilleure compréhension du système
  - Abstraction du problème
  - Se concentre sur l'essentiel (pas de détails)
  - Visualisation du système simple
- Conception progressive
  - Raffinement du modèle successif
  - Découpage en vues

$\Rightarrow$ Génération des structures de données

Les risques d'une mauvaise conception
Niveau applicatif

- Application volumineuse
  - Trop de relations / contraintes
- Informations de l'application incomplètes
  - Spécification incomplète → patch du modèle
- Application non-évolutive
  - Specification fermée → restructuration de la BD

Les risques d'une mauvaise conception
Niveau système

- Perte d'informations
  - Modèle non normalisé
- Performances
  - Volume de données non prise en compte
  - Fréquence des requêtes imprévues
  - Stockage de l'information inadapté
- Coût de réorganisation élevé
  - Mauvais choix de stockage

Les risques d'une mauvaise conception
Niveau utilisateur/Concepteur

- Sécurité de l'information
  - Vues utilisateurs non définies
- Interrogation complexe
  - Pas d'abstraction de la BD
- Peu d'évolutivité de la BD
  - Pas de schéma
  - Historique d'évolution inexistant

Méthode de conception d'une BDD

Connaissance informelle

(Texte descriptif des besoins)

Analyse des besoins

(Spécification des vues, Description formelle)

Schéma Conceptuel

(Intégration des vues)

Schéma Logique

Transformation et normalisation

Schéma Physique

Évaluation et optimisation

Connaissance Informelle

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- "*Notre compagnie d'aviation a besoin de définir des avions, des vols avec destinations, des passagers et compte client, un équipage et leur attribution, des bagages et leur suivi.*"
- Défini par le client
- Peu/Pas de connaissance des nécessités opérationnelles

Analyse des Besoins

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- La compagnie d'aviation est définie par :
  - `Avions`, *avec un matricule, un modèle, une capacité ;*
  - `Aéroports`, *avec un nom, un pays, un code d'aéroport, formant le point de départ et le point d'arrivée d'un vol ;*
  - `Vols`, *reliant deux aéroports destinations, avec date de départ, une distance et une durée, ainsi qu'un avion associé ;*
  - `Personnels`, *avec nom, prénom, date de naissance, métier et matricule permettant de composer un équipage d'un vol comportant un pilote, un co-pilote, un officier mécanicien, deux chefs de cabine, et au moins une hotesse ou stewart ;*
  - `Clients` *avec nom, prénom, date de naissance, identifiant, compte de fidélité ;*
  - `Passagers` *qui lie un client à un vol, avec un ou plusieurs numéro de bagage, cout du billet ;*
- Définition des vues :
  - `Aiguilleur du ciel` (consultation/ajout/modification Vols)
  - `Admin compagnie` (consultation vols, édition d'équipage)
  - `Personnel` (consultation planning de vol)
  - `Client` (consultation vols, achat billet)

Analyse des Besoins

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- Spécification définie par un échange entre le concepteur et le client
- Informations atomiques (attributs)
- Estimation quantitatives des données
- Domaines de valeurs des attributs
- Dépendances fonctionnelles
- Contraintes

Schéma conceptuel - Entité/Association

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

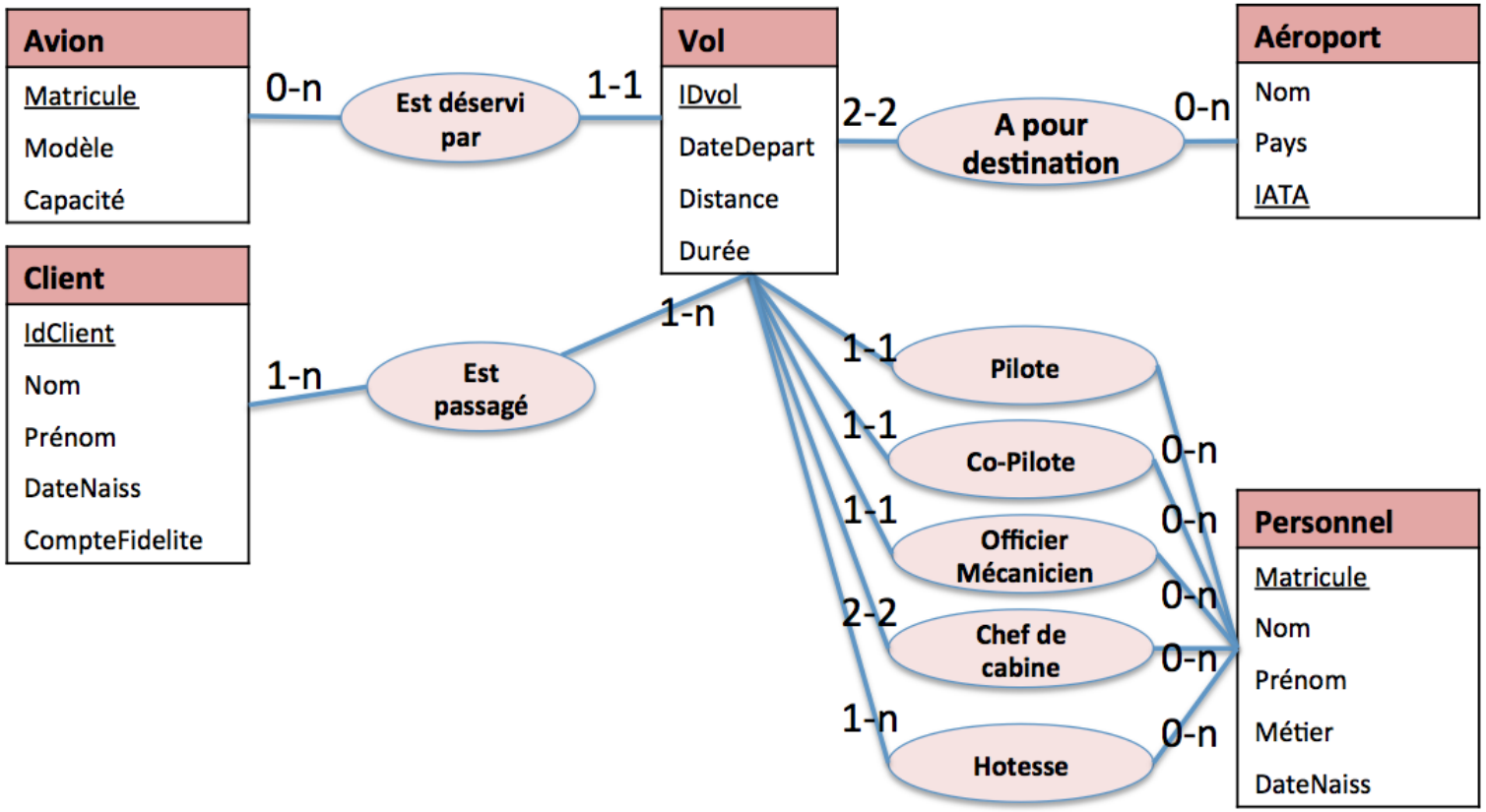


Schéma conceptuel - Pourquoi ?

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- Modèle Entité/Association
- Intuitif : Simple à définir/comprendre
- Très utilisé
- Formalise la BDD, évite les erreurs d'intégration
- ⚠ Modélisation non-unique
- ⚠ Pas d'intégration des traitements

Schéma conceptuel - Définitions

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- **Entité** : objet abstrait ou concret (*Avion*, *Equipage*)
- **Attribut** : propriété d'une entité (*Nom*, *DateNaiss*, *Capacité*)
- **Clé/Identifiant** : 1 attribut (ou +) dont la (les) valeur(s) identifient de façon unique chaque occurrence d'une entité (*Matricule*, *IdClient*)
- **Association** : lien/relation entre entités (*Avion - Destinations*)
  - 2 ou + entités
  - Une entité vers elle-même

Schéma conceptuel - Cardinalités

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- Sur chaque lien d'une association
- Précise pour chaque occurence de l'entité, le nombre d'association du même type
  - 0 non obligatoire
  - 1 au moins un / maximum un
  - X au moins X / maximum X
  - n plus d'une fois
- Ex : `Aéroport 2-2 Vol`
  2 Aéroports pour un vol, min et max.
- Ex : `Client 1-n Vol`
  1 client prend au moins un vol, et peut être plus d'une fois.

Schéma Logique - E/A vers Relationnel

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- **Avion**(Matricule INT, Modele VARCHAR, Capacité INT)
- **Aeroport**(IATA CHAR(3), Nom VARCHAR, Pays VARCHAR)
- **Personnel**(Matricule INT, Nom VARCHAR, Prenom VARCHAR, DateNaiss DATE, Metier VARCHAR)
- **Vol**(IDVol INT, Depart# CHAR(3), Arrivee# CHAR(3), DateDepart DATETIME, Distance INT, Duree TIME, Avion# INT, Pilote# INT, Co-Pilote# INT, Mecanicien# INT, ChefCabine1# INT, ChefCabine2# INT)
- **Hotesse**(IDVol# INT, Hotesse# INT)
- **Client**(IDClient INT, Nom VARCHAR, Prenom VARCHAR, DateNaiss DATE, CompteFidelité INT)
- **Passager**(IDVol# INT, IDClient# INT, Prix FLOAT, NbBagage INT)

Schéma Logique - Clés

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- Clé primaire : 1 attribut (ou +) dont la (les) valeur(s) identifient de façon unique un n-uplet
  - Matricule ou Nom, Prénom,
  - IDClient ou Nom, Prénom,
  - IDVol ou Depart, Arrivee, DateHeure
- Clé étrangère : Référence vers une clé primaire Depart#, IDClient# ou Pilote#
  - La valeur doit exister dans la table référencée (Insertion)
  - Les n-uplets doivent être supprimés en cas de suppression de la clé référencée
  - Traduction du lien Entité/Association

Schéma Logique - Relation

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- Base de données : Ensemble de Relations
- Relation : Nom + Schéma + n-uplets
- Schéma : Ensemble fini d'attributs différents
- Attribut : Domaine de valeur (INT, VARCHAR, FLOAT...)
- N-uplet (*Tuples*) : Instance d'une relation prenant pour chaque attribut une valeur du domaine lié

Schéma Logique - Concept ⇒ Relation

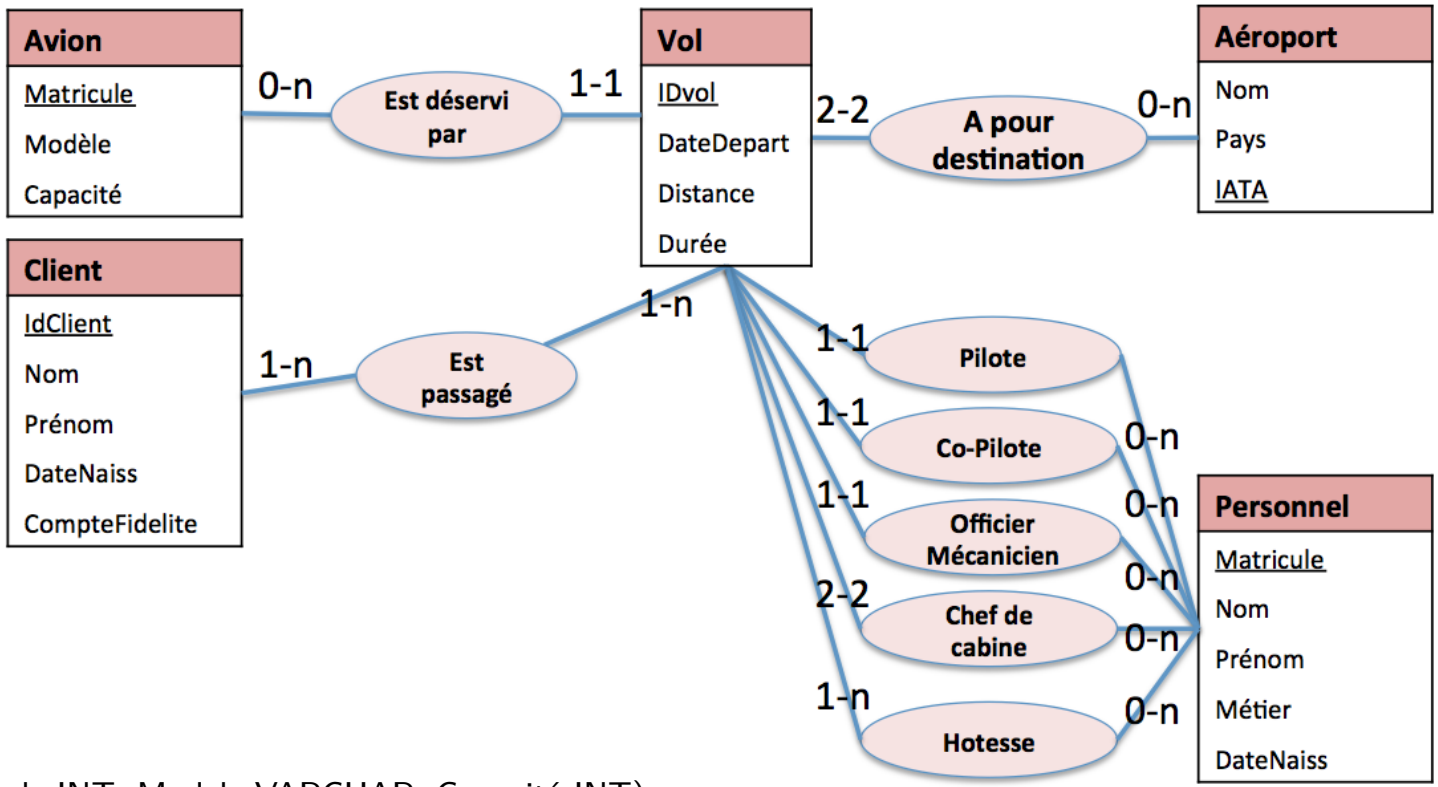
Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- Schéma conceptuel non formel
- Traduction :
  - Entité ⇒ Relation avec clé primaire (*Avion*, *Aéroport*)
  - Association

x-n/x-m ⇒ Relation, avec pour clé primaire l'ensemble des deux clés étrangères + Attributs association (*Passager*, *Hotesse*)

x-1/x-n ⇒ Clé étrangère (côté n) est importée dans la relation (côté 1) + Attribut association (*Vol*)

Concept ⇒ Relation



Avion(Matricule INT, Modele VARCHAR, Capacité INT)

Aeroport(IATA CHAR(3), Nom VARCHAR, Pays VARCHAR)

Personnel(Matricule INT, Nom VARCHAR, Prenom VARCHAR, DateNaiss DATE, Metier VARCHAR)

Vol(IDVOL INT, Depart# CHAR(3), Arrivee# CHAR(3), DateDepart DATETIME, Distance INT, Duree TIME, Avion# INT,Pilote# INT, Co-Pilote# INT, Mecanicien# INT, ChefCabine1# INT, ChefCabine2#)

Hotesse(IDVol# INT, Hotesse# INT)

Client(IDClient INT, Nom VARCHAR, Prenom VARCHAR, DateNaiss DATE, CompteFidelité INT)

Passager(IDVol# INT, IDClient# INT, Prix FLOAT, NbBagage INT)

Schéma Physique - Logique vers Physique

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

```
CREATE TABLE ‘Passager‘ (
    ‘IDVol‘ BIGINT NOT NULL,
    ‘IDClient‘ BIGINT NOT NULL,
    ‘Prix‘ FLOAT NOT NULL,
    ‘NbBagages‘ TINYINT NOT NULL DEFAULT 1,
    PRIMARY KEY ‘pk_Passager‘ (‘IDVol‘, ‘IDClient‘),
    CONSTRAINT ‘fk_PassagerVol‘ FOREIGN KEY (‘IDVol‘)
        REFERENCES ‘Vol‘ (‘IDVol‘)
        ON DELETE CASCADE,
    CONSTRAINT ‘fk_PassagerClient‘
        FOREIGN KEY (‘IDClient‘)
        REFERENCES ‘Client‘ (‘IDClient‘)
) STORAGE (INITIAL 80K NEXT 160K PCTINCREASE
10) PCTFREE 10
```

Schéma Physique - Choix de stockage

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- Choix de stockage dépend du SGBD utilisé :
  - MySQL ⇒ ENGINE
  - Oracle ⇒ Cluster/Partition/Extensions/PCTFREE...
  - SQLServer ⇒ IAM/Cluster
  - DB2 ⇒ Compression
- Typage des données
- Encodage des caractères
- Spécifications des index et contraintes
- Anti-quote ‘ non obligatoire (Mots réservés par le système)

Schéma Physique - Typage des données

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- Chaque type à son encodage et sa taille (octets)
- Types petits → Table petite
- Types grands → Plus de précision ou domaine plus grand
  - TINYINT : 1 octet, 0 à 255
  - BIGINT : 8 octets, 0 à 10^{19}
  - FLOAT : 8 octets, 24 décimales
  - NUMBER(M,D) : 1 à 22 octets, D décimales
  - DATE : 3 octets
  - DATETIME : 8 octets
  - TIMESTAMP : 4 octets
  - YEAR : 1 octet
  - VARCHAR(N) : N+1 octets
  - ENUM(...) : 1 octet (Sous MySQL)

Schéma Physique - Encodage des caractères

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- *Charset* : UTF8, ISO8859-1, latin1, UTF16, cp1252, ASCII...
- Stockage des caractères dépend de la langue
  - Accents
  - Caractères de contrôle (espace, passage à la ligne...)
  - Sensibilité à la casse
  - Taille d'encodage
- Difficulté de conversion

Schéma Physique - Dictionnaire de Données

Connaissance informelle
↓
Analyse des besoins
↓
Schéma Conceptuel
↓
Schéma Logique
↓
Schéma Physique

- Zone de la base de données de métadonnées
- Stocke toutes les informations :
  - Bases de données
  - Tables (+ indexes, statistiques, clés primaires/étrangères)
  - Procédures et Triggers
  - Administration (Utilisateurs, Droits...)
- Utilisé en permanence (vérification requêtes, optimisation et droits d'accès)

Théorie de la Normalisation

- **Objectif :** Bien concevoir son schéma relationnel
  - Éviter les redondances d'information
  - Supprimer les anomalies de mises à jour
- **Concept :**
  - Dépendances fonctionnelles
  - Formes normales
- **Méthode :** Décomposer les relations en sous-ensembles normalisés
  (Représentation à l'aide des attributs)

Normalisation : Exemple

- *Non normalisée :*
  Vol(IDVol, Hotesse, Depart, Arrivee, DateHeure, Distance,
  Duree, Avion, Pilote, Co-Pilote, Mecanicien,
  ChefCabine1, ChefCabine2)

- *Normalisée :*
  Vol(IDVol, Depart, Arrivee, DateHeure, Distance, Duree,
  Avion, Pilote, Co-Pilote, Mecanicien,
  ChefCabine1, ChefCabine2)
  Hotesse(IDVol, Hotesse)

Dépendances Fonctionnelles (DF)

- **Définition :**
  - Relation $\texttt{R(A}_1\texttt{, A}_2\texttt{ ... A}_n\texttt{)}$
  - X et Y des sous-ensembles de A_1, A_2 ...A_n

  $\Rightarrow$ $X \rightarrow Y$ (X détermine Y) ssi toute valeur de X détermine une valeur unique de Y
- **Exemples :**
  - IDClient $\rightarrow$ Nom, Prénom
  - IATA $\rightarrow$ Nom, Pays
  - IDVol $\rightarrow$ Depart, Arrivée, DateHeure, Avion, Pilote
  - Depart, Arrivée, DateHeure $\rightarrow$ Avion, Pilote

Décomposition sans perte d'une relation

- **Théorème :**
  - Soient R(A_1,A_2,...A_n,X,B_1,B_2,...B_m) une relation, et X $\rightarrow$ B_1, B_2, ... B_n une DF
  - La décomposition de R en R1(A_1, A_2, ... A_n, **X**) et R2(**X**, B_1, B_2, ... B_m) est sans perte d'information
- **Exemple :**
  - Voiture(numImm, Marque, Type, Couleur) avec Type $\rightarrow$ Marque
  - Sans perte :
    - Véhicule (NumImm, Type, Couleur)
    - Modèle (Type, Marque)
  - Avec perte :
    - Véhicule (NumImm, Type)
    - Modèle (Type, Marque, Couleur)

Formes Normales

- Définir des règles de décomposition de relations
  - Préserve les Dépendances Fonctionnelles
  - Objets et associations atomiques

→ 3 niveaux de formes normales

1° Forme Normale : 1FN

- Tout attribut doit avoir une valeur atomique

Personnel	Matricule	Nom	Métier
	1	Dupont	*Pilote, Officier Mécanicien*
	2	Dupond	Pilote

⇒

Personnel	Matricule	Nom
	1	Dupont
	2	Dupond

Profession	Matricule	Métier
	1	Pilote
	1	Officier Mécanicien
	2	Pilote

2° Forme Normale : 2FN

- Une relation est en 2ème forme normale ssi :
  - Elle est en 1FN
  - Tout attribut non clé ne dépend pas d'une partie de la clé
- Exemple :
  Vol(IDVol, *Depart, Arrivee, DateHeure*, Distance, Duree, Avion, Pilote, Co-Pilote, Mecanicien, ChefCabine1, ChefCabine2)

2° Forme Normale : 2FN

- Une relation est en 2ème forme normale ssi :
  - Elle est en 1FN
  - Tout attribut non clé ne dépend pas d'une partie de la clé
- Exemple :
  Vol(IDVol, *Depart, Arrivee, DateHeure*, Distance, Duree, Avion, Pilote, Co-Pilote, Mecanicien, ChefCabine1, ChefCabine2)

⇒ **Vol**(IDVol, Depart, Arrivee, DateHeure, Distance, Duree, Avion, Pilote, Co-Pilote, Mecanicien, ChefCabine1, ChefCabine2)

3° Forme Normale : 3FN

- Une relation est en 3ème forme normale ssi :
  - Elle est en 2ème forme normale
  - Tout attribut non clé ne dépend pas d'un attribut non clé

→ Pas de perte d'information

→ Représentation canonique du monde réel

- Exemple :
  Vol(IDVol, *Depart, Arrivee, DateHeure*, Distance, Duree, Avion, Pilote, Co-Pilote, Mecanicien, ChefCabine1, ChefCabine2)

3° Forme Normale : 3FN

- Une relation est en 3ème forme normale ssi :
  - Elle est en 2ème forme normale
  - Tout attribut non clé ne dépend pas d'un attribut non clé

→ Pas de perte d'information

→ Représentation canonique du monde réel

- Exemple :
  Vol(IDVol, *Depart, Arrivee, DateHeure*, Distance, Duree, Avion, Pilote, Co-Pilote, Mecanicien, ChefCabine1, ChefCabine2)

⇒ **Vol**(IDVol, Avion, Pilote, Co-Pilote, Mecanicien, ChefCabine1, ChefCabine2)
VolTrajet(IDVol, Depart, Arrivee, DateHeure)
Trajet(Depart, Arrivee, Distance)

TrajetDuree(Avion, Distance, Duree)

3° Forme Normale : 3FN

- Une relation est en 3ème forme normale ssi :
  - Elle est en 2ème forme normale
  - Tout attribut non clé ne dépend pas d'un attribut non clé

→ Pas de perte d'information

→ Représentation canonique du monde réel

- Exemple :
  Vol(IDVol, *Depart, Arrivee, DateHeure*, Distance, Duree, Avion, Pilote, Co-Pilote, Mecanicien, ChefCabine1, ChefCabine2)

⇒ **Vol**(IDVol, Avion, Pilote, Co-Pilote, Mecanicien, ChefCabine1, ChefCabine2)
VolTrajet(IDVol, Depart, Arrivee, DateHeure)
Trajet(Depart, Arrivee, Distance)

TrajetDuree(Avion, Distance, Duree)

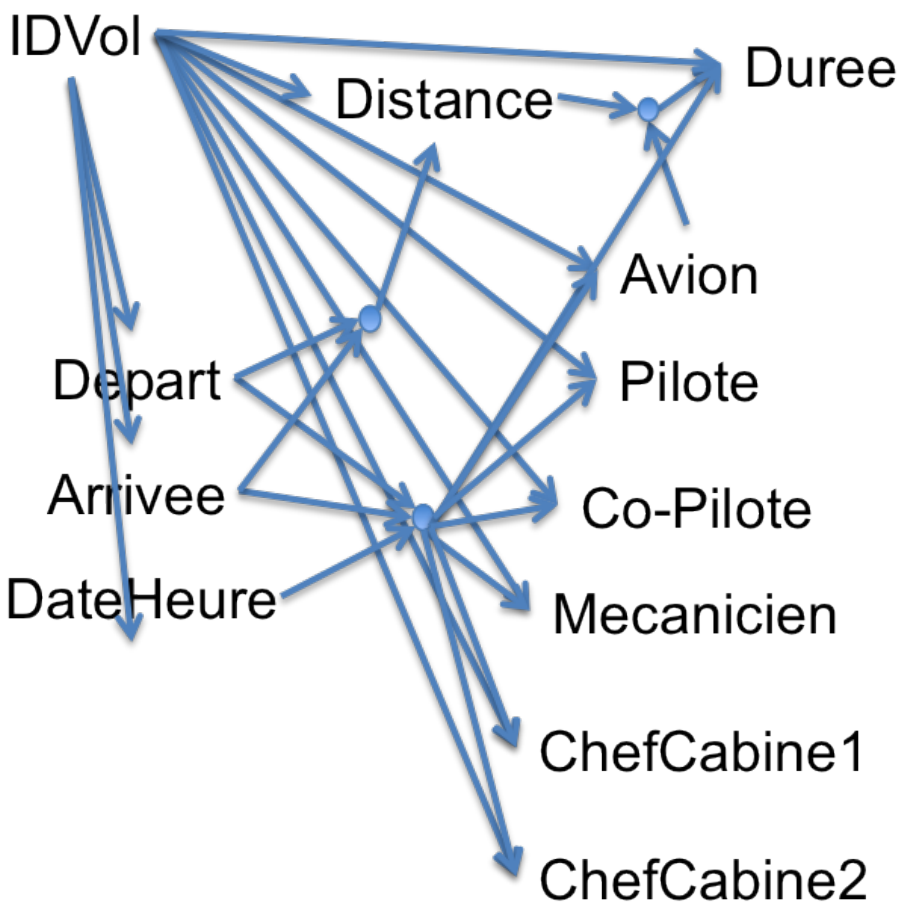
⚠ Base de données parfois moins optimisée

Algorithme de décomposition en 3FN

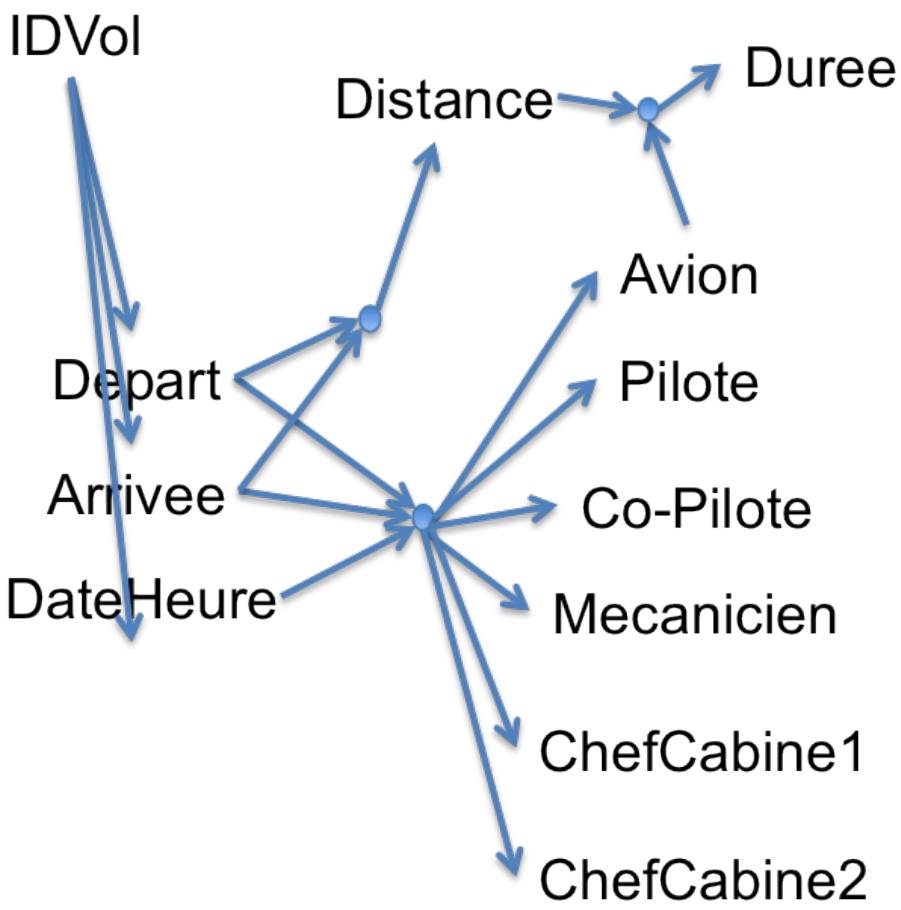
1. Créer un graphe de Dépendances Fonctionnelles
2. Calculer sa couverture minimale (minimum de DF, le reste peut être déduit)
3. Si ensemble d'attributs isolés $\rightarrow$ Relation dont tous les attributs sont clés
4. Rechercher le plus grand ensemble X d'attributs qui détermine d'autres attributs A1, ..., An
   - Relation R (X, A1, ..., An)
   - Supprimer les DF (X $\rightarrow$ A1, ..., X $\rightarrow$ An) du graphe
   - Supprimer les attributs isolés du graphe
5. Si le graphe n'est pas vide, répéter à partir de **4**

Algorithme 3FN : Exemple

1. Graphe de DF

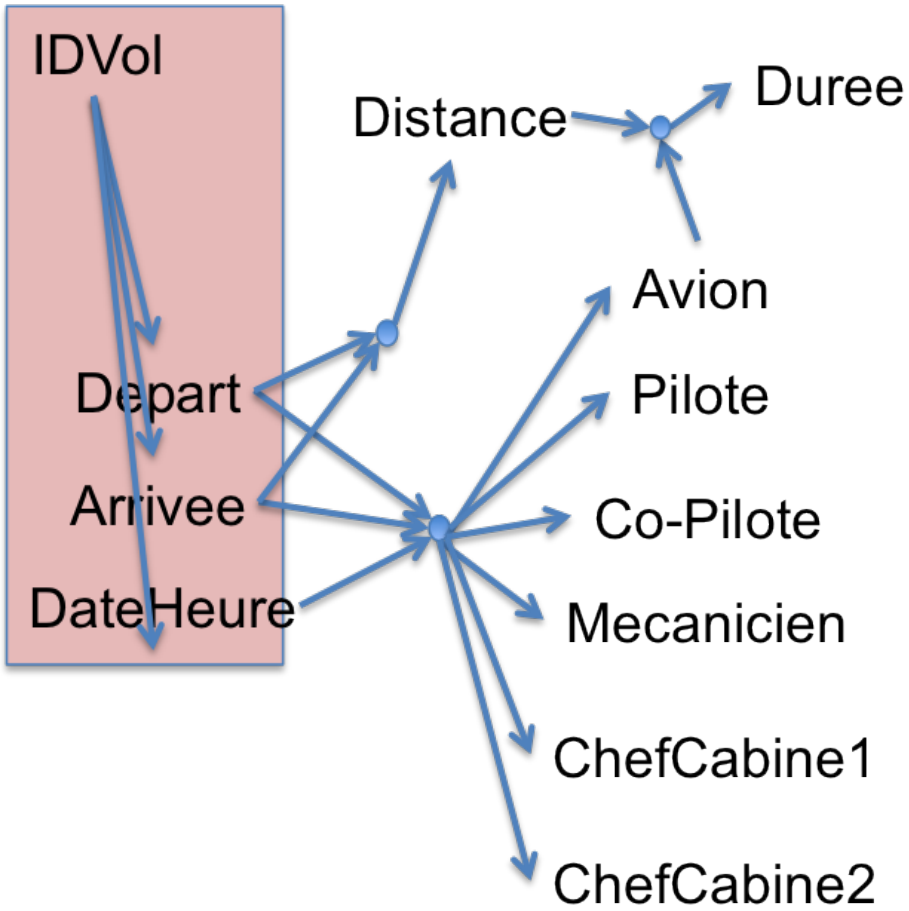


Algorithme 3FN : Exemple



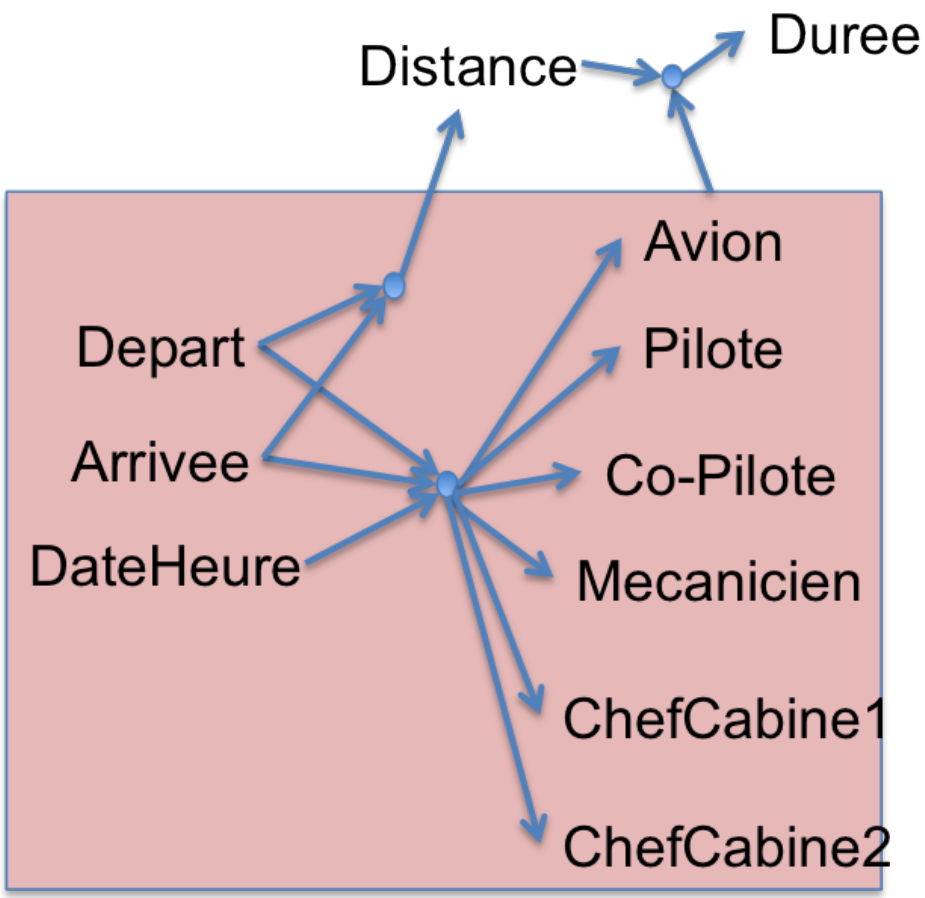
1. Graphe de DF
2. Couverture minimale

Algorithme 3FN : Exemple



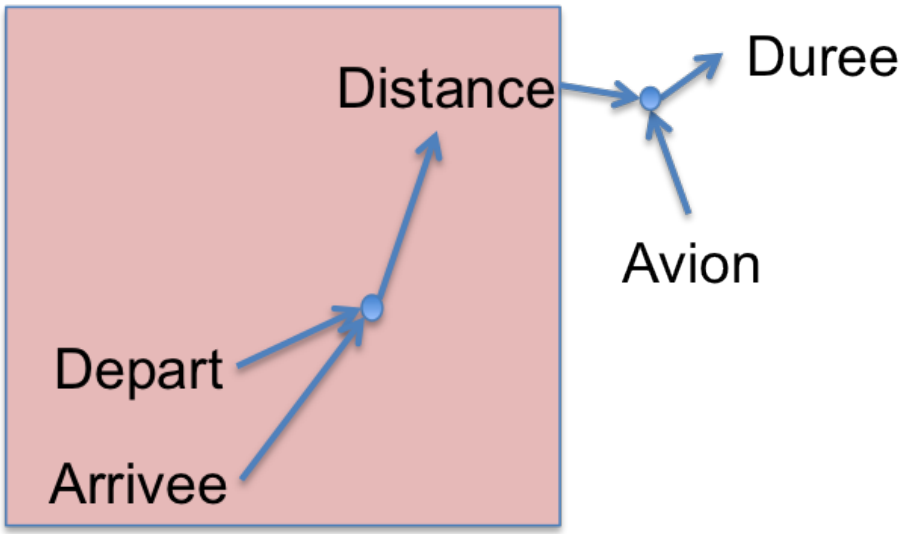
1. Graphe de DF
2. Couverture minimale
3. Vol(IDVol, Depart, Arrivee, DateHeure)

Algorithme 3FN : Exemple



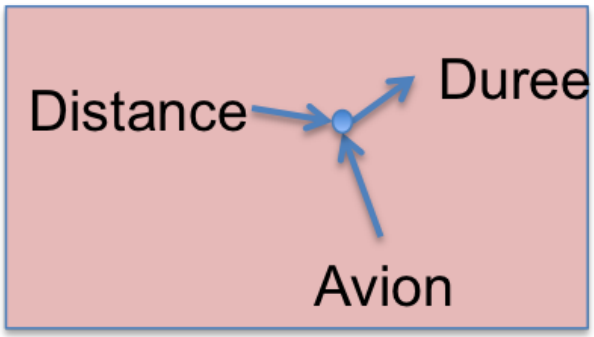
1. Graphe de DF
2. Couverture minimale
3. Vol(IDVol, Depart, Arrivee, DateHeure)
4. VolInfo(Depart, Arrivee, DateHeure, Avion, Pilote, Co-Pilote, Mecanicien, ChefCabine1, ChefCabine2)

Algorithme 3FN : Exemple



1. Graphe de DF
2. Couverture minimale
3. Vol(IDVol, Depart, Arrivee, DateHeure)
4. VolInfo(Depart, Arrivee, DateHeure, Avion, Pilote, Co-Pilote, Mecanicien, ChefCabine1, ChefCabine2)
5. Distance(Depart, Arrivee, Distance)

Algorithme 3FN : Exemple



1. Graphe de DF
2. Couverture minimale
3. Vol(IDVol, Depart, Arrivee, DateHeure)
4. VolInfo(Depart, Arrivee, DateHeure, Avion, Pilote, Co-Pilote, Mecanicien, ChefCabine1, ChefCabine2)
5. Distance(Depart, Arrivee, Distance)
6. Duree(Avion, Distance, Duree)

Contraintes sur les données

- Besoin d'exprimer des contraintes sur les données :
  - Types d'informations (entiers, texte, date)
  - Valeurs (nulles, bornes, unicité)

Types de données

Types numériques

Type	Taille	Interval de valeurs
TINYINT	1 octet	0 à 255 ou -128 à 127
SMALLINT	2 octets	-32768 à 32767 ou 0 à 65535
MEDIUMINT	3 octets	-8388608 à 8388607 ou 0 à 16777215
INT, INTEGER	4 octets	-2147483648 à 2147483647 ou 0 à 4294967295
BIGINT	8 octets	$\sim 10^{19}$
FLOAT(p)	4 octets if $0 \leq p \leq 24$, 8 octets if $25 \leq p \leq 53$	nombre de décimales
FLOAT	8 octets	Par defaut, 53 décimales
DOUBLE(p), REAL	8 octets	
DECIMAL(M,D), NUM(M,D)	Variable	
BIT(M)	$\sim (M+7)/8$ octets	

Types Dates et Heures

Type	Taille
DATE	3 octets
TIME	3 octets
DATETIME	8 octets
TIMESTAMP	4 octets
YEAR	1 octet

Types textuel

Type	Taille
CHAR(M)	M octets ou ou M x 2 octets, $0 \leq M \leq 255$
BINARY(M)	M octets, $0 \leq M \leq 255$
VARCHAR(M)	M + 1o (0-255 o), M + 2o (> 255 o)
BLOB, TEXT	L + 2 octets, where $L \leq 215$
ENUM('v1','v2',...)	1o (255 valeurs), 2o (65 535 valeurs)
SET('v1','v2',...)	1, 2, 3, 4, or 8 octets (64 valeurs maximum)

Types de données - signées / non-signées

- Signée / *Signed* :
  - Prendre en compte les valeurs négatives
  - Le premier bit de l'encodage désigne le 'signe'

1	0	0	0	0	0	0	1

$\Rightarrow$ -1

  - 7 bits pour la valeur
- Non-signée / *unsigned* :
  - 8 bits pour la valeur

1	0	0	0	0	0	0	1

$\Rightarrow$ 129

Contraintes de domaines

- Valeurs `NULL` :
  - Valeur non définie
  - `NULL` $\neq 0$
  - Economie de place de stockage
  - Peut être utile pour une clé étrangère non spécifiée (ex : ChefCabine2 = `NULL`)
- Valeurs `UNIQUE` :
  - Unicité de chaque valeur de l'attribut
  - *Primary Key* = *UNIQUE* + index
- Contrainte sémantique avec `CHECK` sur un n-uplet :
  - 1 Attribut : `CHECK(Matricule>1000 AND Matricule<9000)`
  - 2 Attributs : `CHECK(Pilote != CoPilote)`

Améliorer la rapidité de la Base de Données

- Table $\rightarrow$ stockage sur le disque (pages de données)
- $\nearrow$ n-uplets, $\nearrow$ place (nb pages), $\nearrow$ temps de traitement

$\Rightarrow$ Besoin d'accès rapides

Améliorer la rapidité de la Base de Données

- Table $\rightarrow$ stockage sur le disque (pages de données)
- $\nearrow$ n-uplets, $\nearrow$ place (nb pages), $\nearrow$ temps de traitement

$\Rightarrow$ Besoin d'accès rapides

- Optimisation de requêtes complexe, besoin de bonnes pratiques (ne répond pas à tout)

Indexation

- La plupart des requêtes ne demandent qu'une partie des données
  $\rightarrow$ Accès à un fragment des données du disque
- **Index** : Information d'emplacement d'une donnée
  - Associé à un attribut (ou plusieurs) d'une table
  - Fonctionne dès que l'attribut est demandé dans la requête (`WHERE`)

Indexation

- La plupart des requêtes ne demandent qu'une partie des données
  $\rightarrow$ Accès à un fragment des données du disque
- **Index** : Information d'emplacement d'une donnée
  - Associé à un attribut (ou plusieurs) d'une table
  - Fonctionne dès que l'attribut est demandé dans la requête (`WHERE`)
- Index de type arborescent (BTree)
  - Accès à une donnée
  - Utile sur des données très diverses (ex : clé primaire, unique, noms)

Indexation

- La plupart des requêtes ne demandent qu'une partie des données
  $\rightarrow$ Accès à un fragment des données du disque
- **Index** : Information d'emplacement d'une donnée
  - Associé à un attribut (ou plusieurs) d'une table
  - Fonctionne dès que l'attribut est demandé dans la requête (`WHERE`)
- Index de type arborescent (BTree)
  - Accès à une donnée
  - Utile sur des données très diverses (ex : clé primaire, unique, noms)
- Index de type table de hachage (HASH)
  - Accès à un ensemble de données identiques
  - Utile sur des données peu diverses (ex : année, pays, service)

Index : Principe

- Un index est stocké dans un autre fichier $\mathcal{I}$
- Besoin d'une clé d'accès $\mathcal{K}$ (un ou plusieurs attributs)
- Besoin d'un pointeur sur les données (ROWID) : @
- Caractéristiques :
  - $\mathcal{I}$ ne contient pas les données
  - A chaque $\mathcal{K}$ correspond un ou plusieurs ROWIDS @
  - $\mathcal{I}$ est ordonné sur les valeurs de $\mathcal{K}$
  - Entrée de $\mathcal{I}$: ($\mathcal{K}$, @)
  - Si $\mathcal{I}$ plus gros qu'une page, on indexe de la même manière chaque page de $\mathcal{I}$

Arbre B+

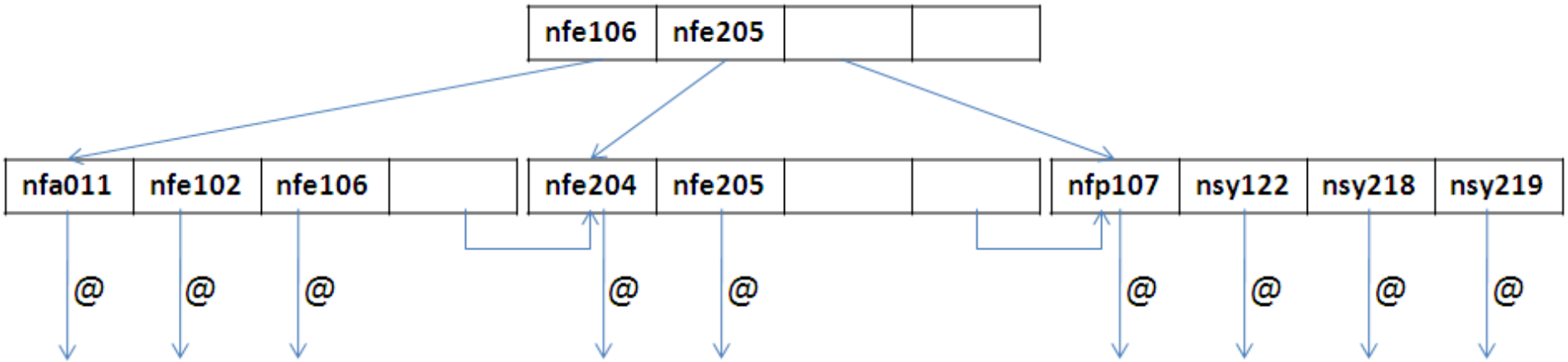
- *Balanced Tree* : Arbre équilibré
- Implanté dans tous les SGBD relationnels (index par défaut)
- Les feuilles de l'arbre contiennent les pointeurs
- Principe des Arbres Binaires de Recherche

⇒ **Optimise les requêtes d'égalité**
(avec peu de valeurs, et les inéquations)

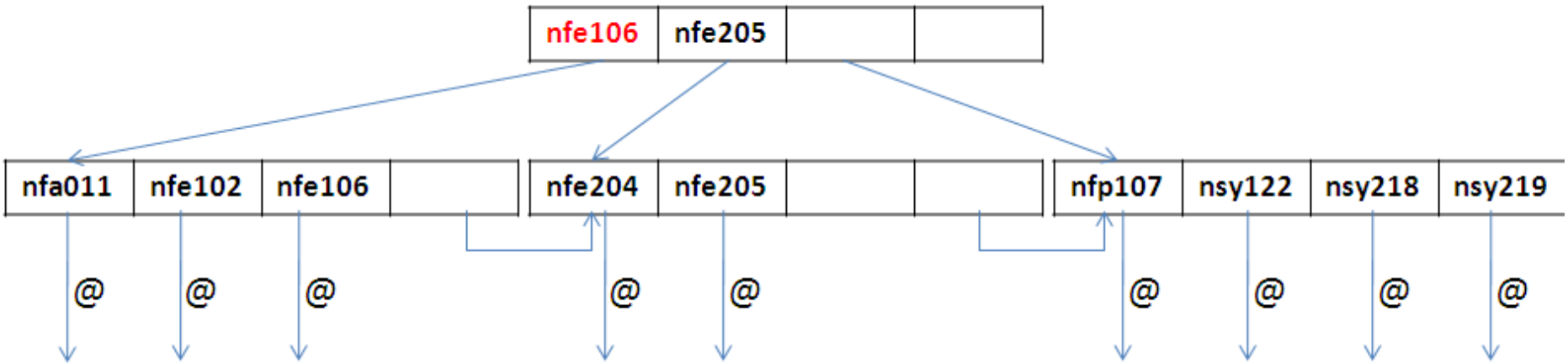
Arbre B+ : Recherche

- Recherche récursive de $\mathcal{K}$ à partir de la racine
- Soit C_1 à C_n les valeurs des clés de la page
  1. Si $\kappa \leq C_1$, recherche sur le noeud référencé P_1
  2. Si $\kappa > C_n$, recherche sur le noeud référencé P_{n+1}
  3. Si $C_i < \kappa \leq C_{i+1}$, recherche sur le noeud référencé P_{i+1}

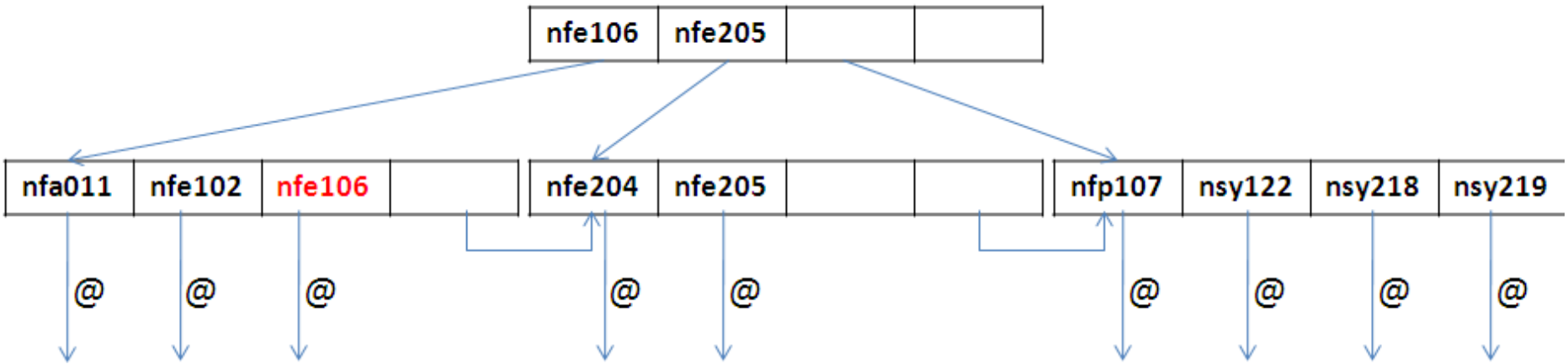
Arbre B+ : Recherche de *nfe106*



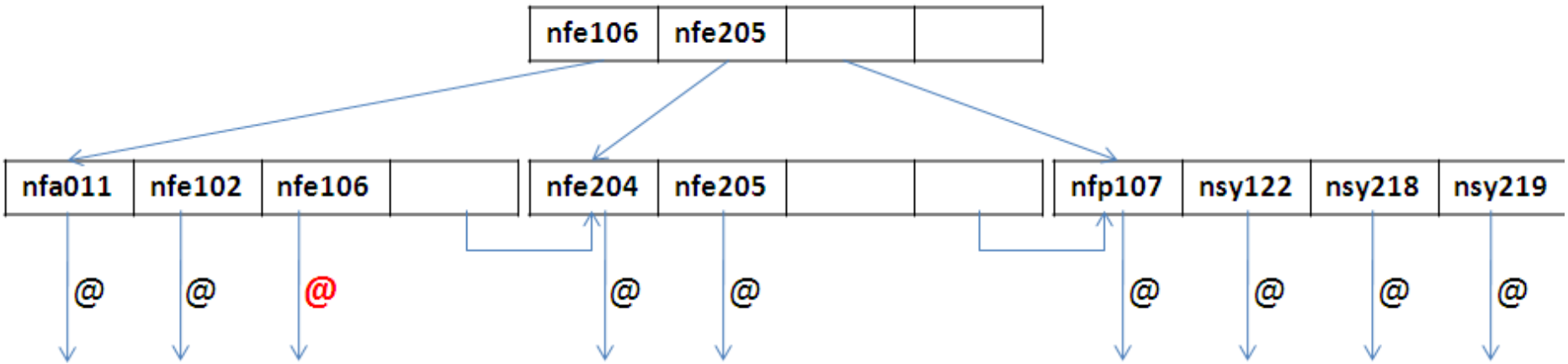
Arbre B+ : Recherche de *nfe106*



Arbre B+ : Recherche de *nfe106*



Arbre B+ : Recherche de *nfe106*



Creation d'un Index de type BTree

- Index par défaut d'un SGBD
- Btree :
  CREATE INDEX *Passager_client* ON *Passager* (*IDClient*) ;

Hachage

- Partitionnement des données
- Très peu de place en mémoire
  - Fonction de hachage
  - Table d'adresses de partitions
- Collisions : plusieurs clés $\mathcal{K}$ par partitions
- Divise le temps de parcours par le nombre de partitions (en moyenne)

Hachage

- Partitionnement des données
- Très peu de place en mémoire
  - Fonction de hachage
  - Table d'adresses de partitions
- Collisions : plusieurs clés $\mathcal{K}$ par partitions
- Divise le temps de parcours par le nombre de partitions (en moyenne)

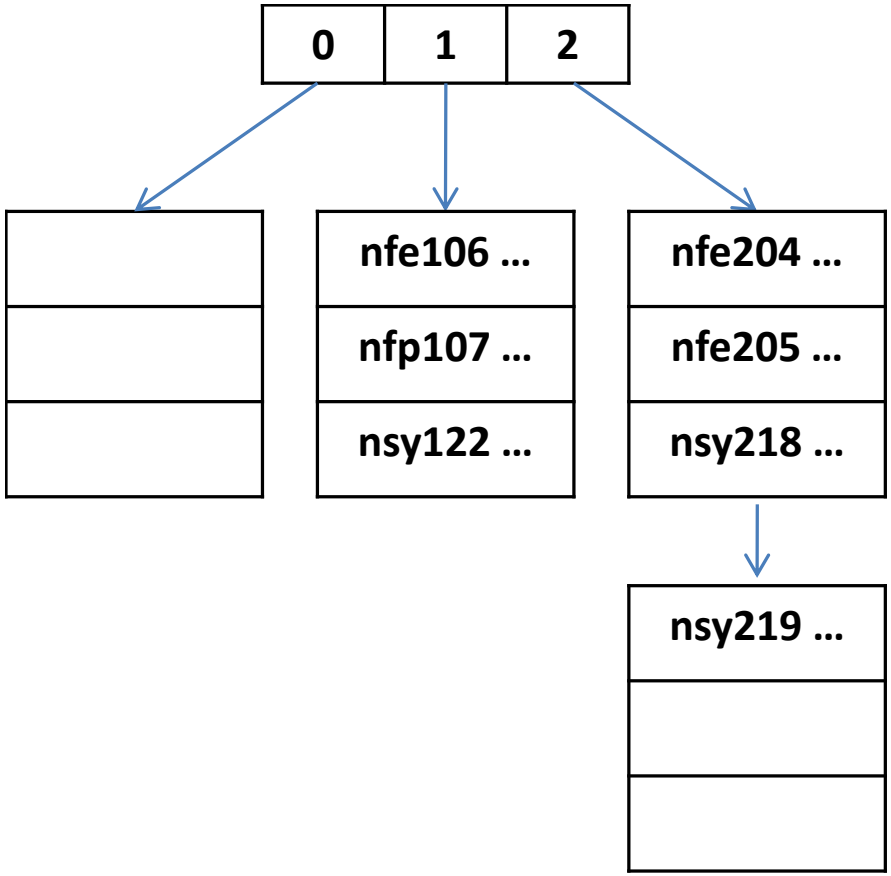
$\Rightarrow$ **Optimise les requêtes d'égalité**.
(Avec de nombreuses valeurs identiques)

Hachage : Caractéristiques

- **Fonction de hachage** : $h(\mathcal{K}) = \alpha$
  - $1 \leq \alpha \leq \beta$
  - $\beta =$ Nb de partitions
  - h est une fonction uniforme[1] : $P(h(\kappa) = \alpha) = \frac{1}{\beta}$
  - Si $\phi(\mathcal{K}) = \alpha$, alors on ajoute la clé $\mathcal{K}$ (+ données) à la page α
- **Table de hachage** T en MC[2]
  - $T[\alpha] =$ Adresse du bloc α

1. Par défaut, *ORA_HASH* sous `Oracle`
`http://docs.oracle.com/cd/B28359_01/server.111/b28286/functions112.htm`
2. Mémoire Centrale

Hachage : Exemple



La fonction de hachage prend le 4° caractère.

Partitionnement

- Plusieurs typles de partitionnement :
  - Hachage
  - Intervalles de valeurs
  - Liste de valeurs
  - Composition des précédentes
- Exemple de création :
  CREATE TABLE Vol (...)
  PARTITION BY HASH (Depart) PARTITIONS 100 ;