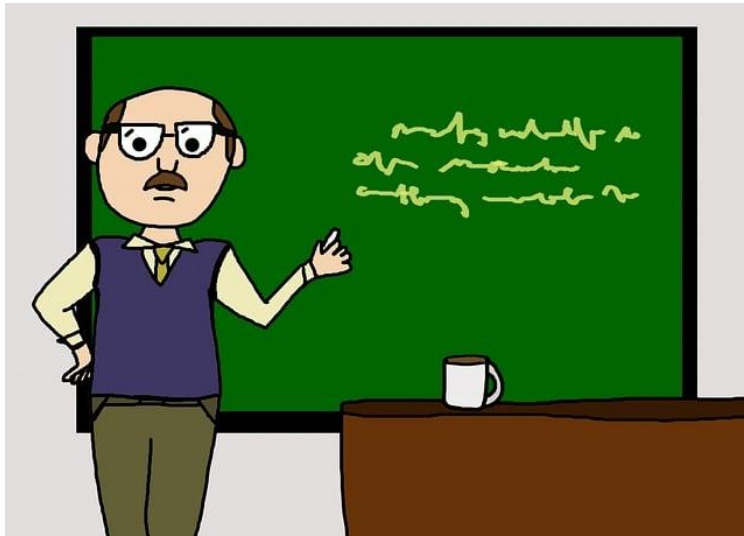




Le cours 2026



Nom : DESNOUAILLES

Age : 58 ans

INSA : depuis 2008

Employeur : WORLDLINE

Métier : Software Engineer

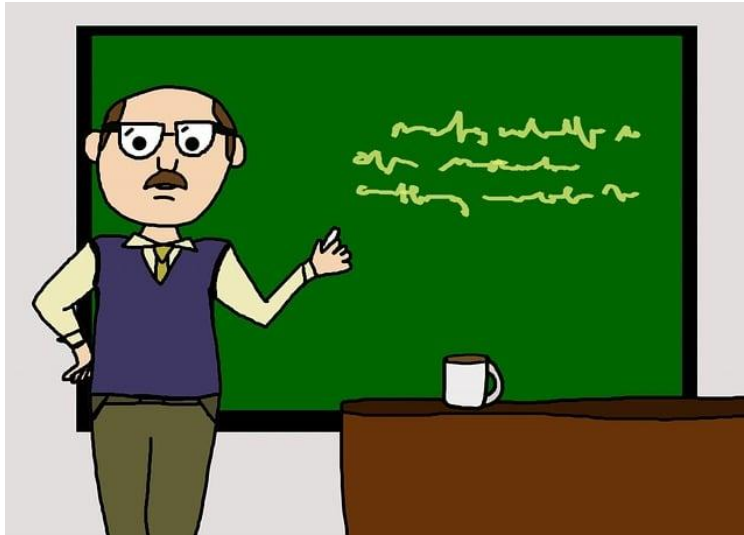
Domaine : ATM

Actif au Comité Social Etablissement

Délégué Syndical Central

QUALIOPFI formateur syndical pour Adulte

- **Modélisations des données**
 - Modèles sémantiques : **Entité-Association**
 - Modèles logiques : Hiérarchiques, réseaux, relationnels
- **Le modèle relationnel**
 - concepts, dépendances fonctionnelles, règles d'intégrité, formes normales
 - J'apprends à faire des ensembles et à compter
- **Le langage SQL (Structured Query Language)**
 - définition de données, manipulation de données, contrôle des données, les vues.
- **NoSQL », NewSQL et évolutions.**
 - Triangle de Brewer
 - Concilier performance, volume, évolutivité et flexibilité dans un contexte distribué.



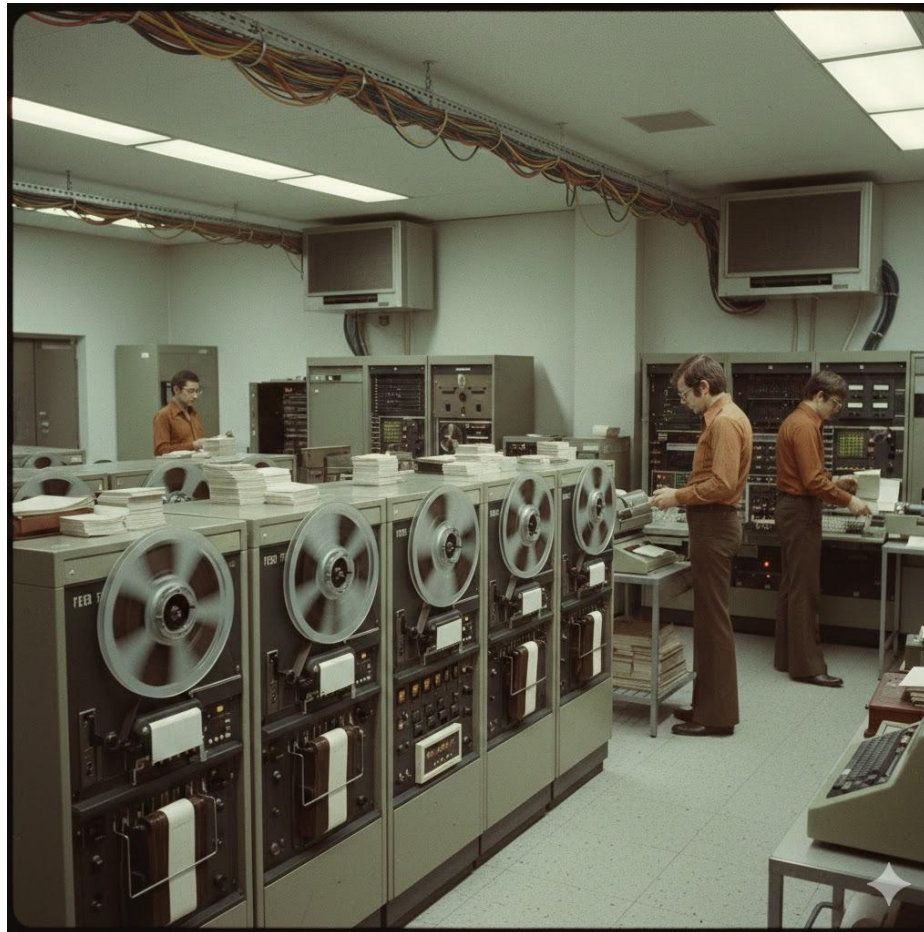
- 6 ou 7 Cours et 6 Travaux Dirigés
- Pas d'examen final
 - Des évaluations
 - Des TP notés

Tests « surprises » ou non d'apprentissage,
Les TP sont majeurs : mise en oeuvre de la compréhension



Généralités

Qu'est-ce qu'une « Base de Données (BD) » ?



Qu'est-ce qu'une « Base de Données (BD) » ?



« Base de Données» Késako ?

- Une **donnée** :

c'est un fait, une notion **représentée** sous une forme **conventionnelle** convenant à une communication, une interprétation ou un traitement soit par un être humain, soit par des moyens informatiques.

- Une **Information**:

c'est le signifiant que l'on attache ou que l'on peut déduire d'un ensemble de données ou de certaines associations entre données. C'est la partie utilisable de la donnée.

- Une **Base de Données**

Elle est faite pour enregistrer des données au sein d'un système d'information.

c'est une organisation cohérente de données permanentes et accessibles par des utilisateurs concurrents. Ces utilisateurs ont, par définition, des vues différentes et des objectifs différents.

- La **base de données** est un élément central dans le **Système d'Information** (SI) d'une entreprise. Elle permet d'obtenir les éléments nécessaires à la prise de décision.

«Système d'information» Késako ?

- Pour toute décision :
 - Évaluer les avantages et inconvénients,
 - Connaître au mieux tous les éléments favorisant la prise de décisions.
- Objectif : Fournir au décideur l'information pertinente à la prise de décision
- Rôle du SI :
 - Recueillir,
 - Mémoriser,
 - Véhiculer,
 - Fournir,les informations liées au fonctionnement d'une organisation.

Articulation SI, BD, SGBD ?

Système d'information - Bases de Données - Systèmes de Gestion de Bases de Données

- La BD représente la partie centrale du SI (mémorisation et exploitation des informations).
- Une BD représente un ensemble de données mémorisé qui est utilisé par de nombreuses personnes et dont l'organisation est régie par un **modèle de données**.
- Une BD c'est un ensemble de données qui :
 - Modélise des objets du monde réel ou virtuel,
 - Sert de support à une application informatique.
- Cet ensemble doit :
 - Vérifier un certain nombre de propriétés,
 - Être interrogeable par le contenu, selon n'importe quel critère.

Le lien ?

- **SGBD** : un Système de Gestion de Base de Données (**DBMS**) est un ensemble de logiciels qui facilite la création et l'utilisation de bases de données.
- Les données sont définies, administrées et gérées en utilisant des langages fondés sur des **modèles de données** pour :
 - Structurer,
 - Décrire l'information,
 - Mémoriser (stocker),
 - Manipuler (lire, écrire, supprimer),
 - Traiter (mettre à jour, analyser, extraire)
 - Protéger,
- des collections de données, tout en assurant leur **confidentialité** et leur **intégrité**

Une BD doit satisfaire 5 critères :

1. **Bonne représentation du monde réel :**

- Elle doit être une image fidèle de la réalité,
- Les informations doivent être fiables et à jour.

2. **Non redondance de l'information :**

- Une information doit être située physiquement à une seule place.

3. **Indépendance des programmes d'application par rapport aux données :**

- La BD est l'image du réel, les programmes sont construits après la structure.

4. **Sécurité et confidentialité des données :**

- La sécurité physique des données doit être assurée (sauvegarde)
- L'accès à une donnée ne doit être autorisée que pour les personnes habilitées.

5. **Performances des applications et des requêtes :**

- Les réponses doivent être aussi rapides que possible,
- Une organisation spécifique peut être définie pour améliorer les temps de traitement.

Ces critères sont repris dans les 4 critères techniques

Atomicité

Chaque instruction d'une transaction (lecture, écriture, mise à jour ou suppression de données) est traitée comme une seule unité. Soit l'instruction est exécutée dans son intégralité, soit elle ne l'est pas du tout. Cette propriété empêche que des données soient perdues ou corrompues

Cohérence

Les transactions ne modifient les tables que de façon prédéfinie et prévisible. La cohérence transactionnelle veille à ce qu'une corruption ou une erreur dans les données n'ait pas de conséquences inattendues sur l'intégrité d'une table.

Isolation

Lorsque plusieurs utilisateurs lisent et écrivent dans la même table simultanément, l'isolement de leurs transactions garantit que les transactions effectuées simultanément ne s'impactent pas les unes les autres. Les requêtes sont exécutées comme si elles étaient successives alors qu'elles sont en réalité simultanées.

Durabilité

Les modifications apportées à vos données par les transactions réussies sont enregistrées, même en cas de défaillance du système.

Approche ACID

Différentes Approches historiques...



Les bases de données : une histoire de plus d'un demi-siècle

- 1959 Première Standardisation du modèle réseau **CODASYL**
(*Conférence On Data Systems Languages*)



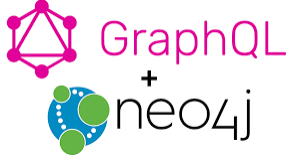


- 1970 : **TED CODD** crée le **modèle relationnel** pour IBM repris dans deux projets de recherches majeurs :
 - **INGRES** (Berkeley California) qui devient **POSTGRES** (logiciel libre!) qui devient **ILLUSTRA** racheté par **INFORMIX** qui redevient **POSTGRE SQL** intégré dans les solutions **CosmoDB**...
 - **System R** pour **IDM** qui devient **DB2** qui inspire **ORACLE** (Unix).



<http://scihi.org/codd-relational-database-model/>

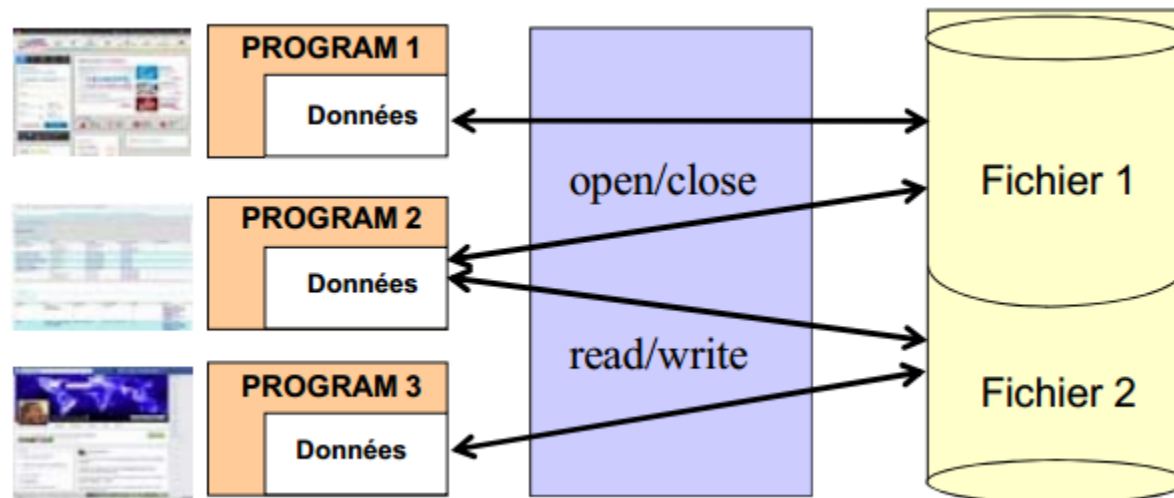


- **1976** : Peter Chen (UCLA/ IBM) définit le modèle **Entité-Association** (E/A) ou Entity Relationship (E/R)
- **1980** maturations des technologies relationnelles :
 - Standardisation des langages : **SQL** (Structured Query Language).
- **1990**
 - technologie **relationnelle** avec support de distribution et parallélisme
 - Emergence de « **ODMG** » Object Database Management Group (1991)

 - Emergence de la modélisation **UML** Unified Modeling Language (1995).

 - Emergence des outils Neo.


Les bases de données : depuis le milieu du XXème siècle

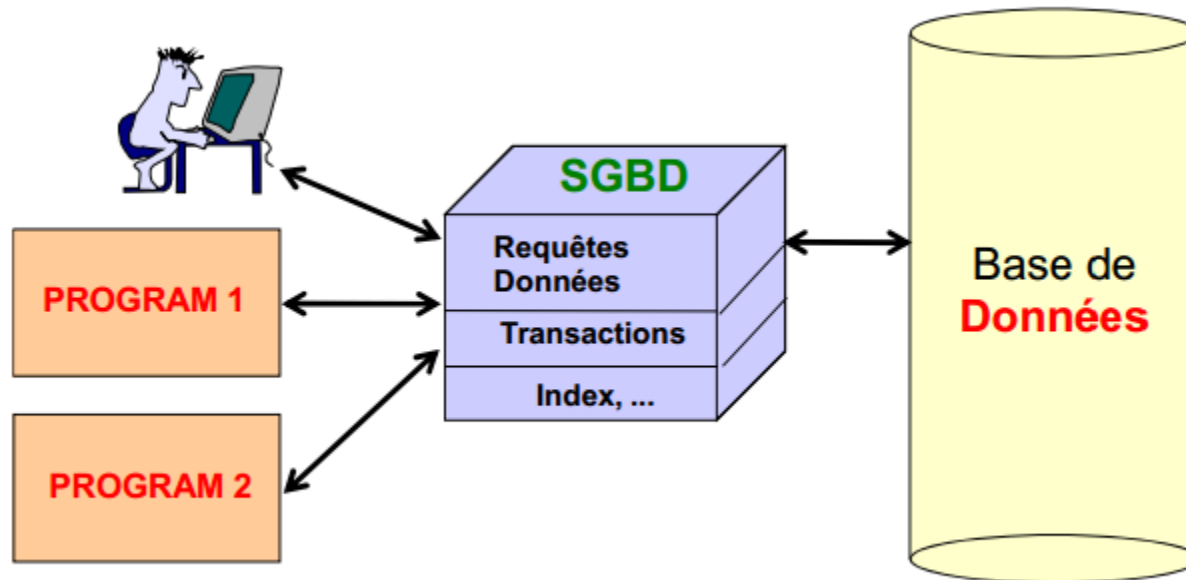
Les fichiers jusque dans les années 1960

- **4 opérations simples** ouvrir/fermer et lire/écrire
- Des méthodes d'accès (indexées, hachées, directes, ...)
- utilisation simultanée difficile par plusieurs programmes
- La gestion de données structurées dans des fichiers représente un coût important de développement.



Les bases de données : depuis le milieu du XXème siècle

Approche SGBD

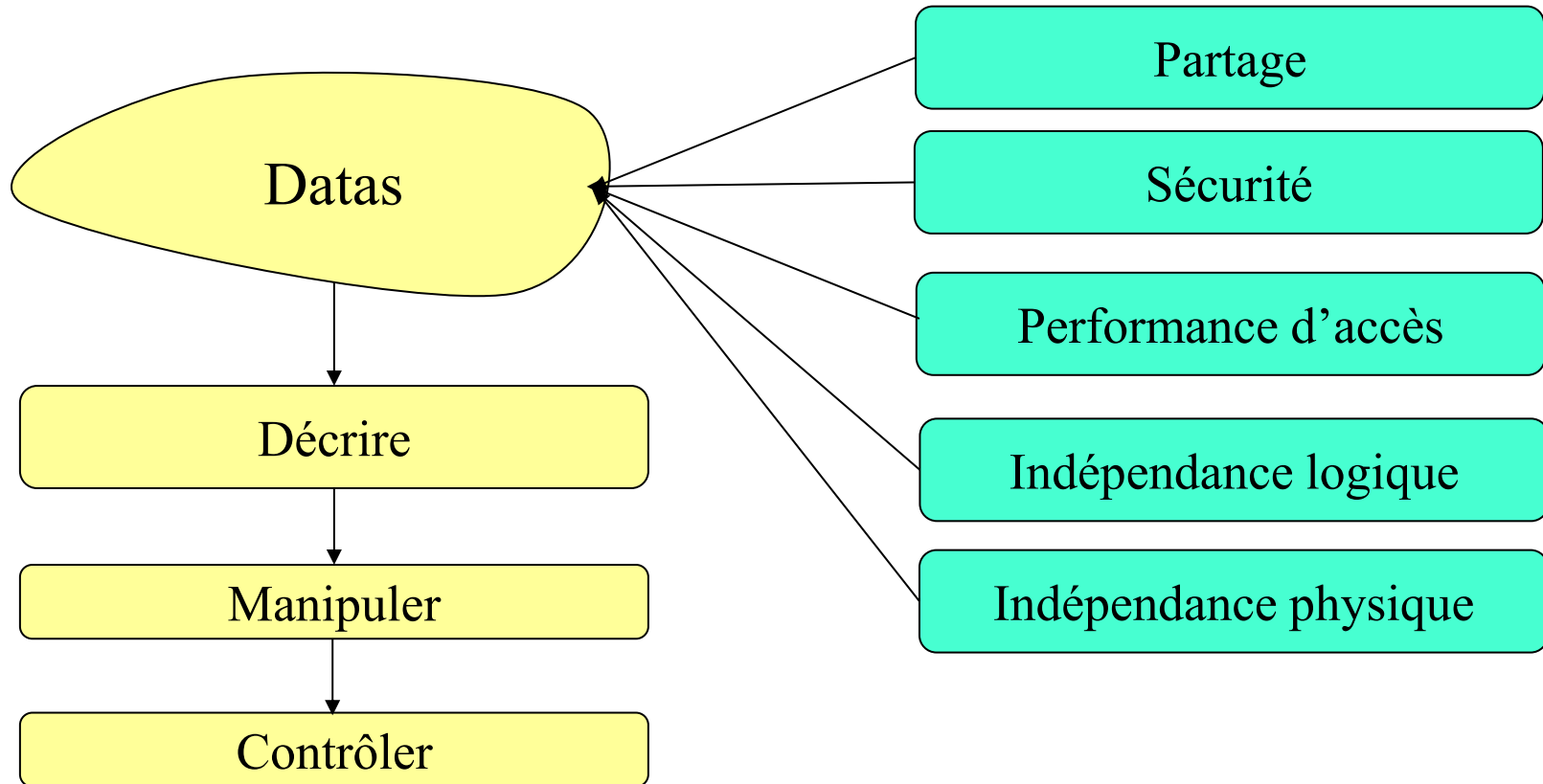


- structuration des données à travers un schéma de données
- Indépendance entre les programmes et la gestion des données
- Données partagées
- contrôle de cohérence (logique/physique à travers les schémas et les transactions).
- Performance d'accès élevée

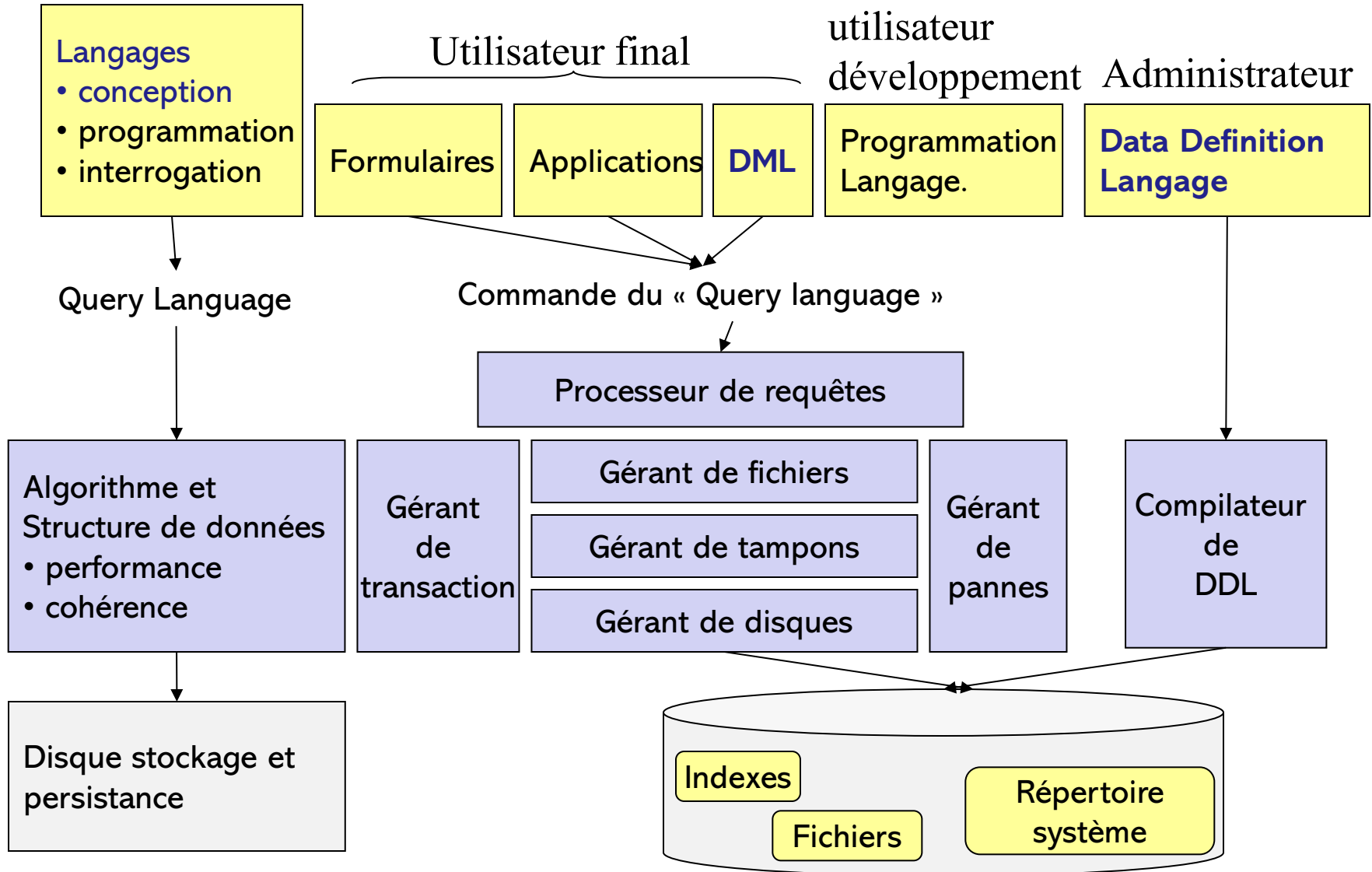
Approche SGBD



Principe d'un SGBD



ARCHITECTURE D'UN SGBD (DBS) EN 3 NIVEAUX

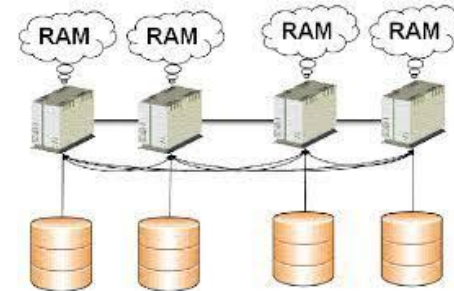


Les outils des SGBD

- **Décrire : Langages de définition de données (LDD)**
 - conception haut niveau; aperçu des données et des traitements.
 - pour définir les schémas externes (vues), logiques et physiques
- **Manipuler : Langage de manipulation de données (LMD)**
 - langage déclaratif pour interroger (requêtes ou request) et mettre à jour :
 - dire QUOI sans dire COMMENT.
 - exemple : Quels sont les noms des produits ayant un prix < 100€.
 - Langage **Autonome** (ex: SQL) ou **Intégré** dans un langage de programmation (ex: dans une API JDBC)
- **Contrôler les données : Langage de contrôle des données (LCD ou)**
 - exemple : le prix doit être compris entre 1 et 100€ lors de l'insertion dans la base.

Les outils de gestion des SGBD

- **Partage :**
 - contrôle des accès concurrents des transactions à la base
 - assure la cohérence de la base.
- **Sécurité**
 - reprise sur panne
 - gère un journal des évènements
- **Performance d'accès**
 - index (hachages, arbres balancés,...)
 - clusterisation
- **Indépendance logique**
 - vues différentes des mêmes données pour des application différentes
 - modification des schémas logiques sans impact sur les applications .
- **Indépendance physique**
 - modification des structure de stockage sans impact sur les applications.



Définitions des langages d'interface SGBD

- **Langages de conception (modèle Entité-Association, UML) :** utilisation pour la conception « haut niveau » aperçu des données et des traitements.
- **Langage de la base de données (SQL, algèbre relationnel, N.O. SQL...)**
 - langages déclaratifs : spécifier QUOI et non comment.
 - Utilisation pour la définition de schémas, d 'interrogation, de mise à jour et d 'administration.
- **Langage de programmation comme JAVA, JSP, PHP,**
 - Langages avec une interface SQL, exemple JDBC.
 - Utilisation pour la programmation d'applications avec accès aux Données.
- **Langage de programmation de bas niveau : C, C++,Visual Basic,**
 - mise en œuvre d'un accès au SGBD via une bibliothèque implémentant les opérateurs physiques, spécifiques de la base.

- Fin 1990 Emergences de nouveaux domaines
 - Entrepôts de données,
 - Décisionnel (SAP Business Object. – TerraData – IA)
 - Besoins autour du web : Moteurs de recherche, multimédias, mobiles



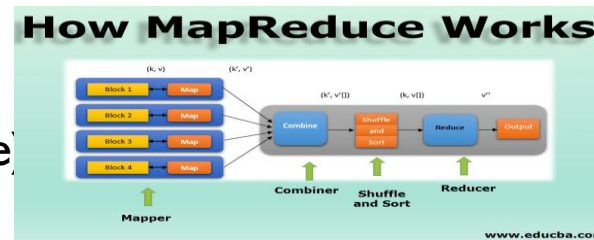
- **Années 2000 et 2010**

- Apparition de XML et de nouvelles architectures (ex : P2P)



- **NoSQL: Not Only SQL**

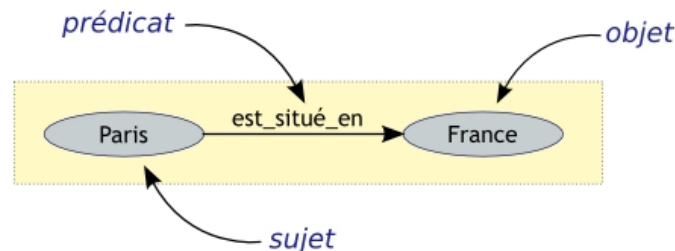
- hbase
- MongoDB
- Kibana hadoop
- MapReduce (Google)



- **RDF (Ressource Data Framework : web sémantique), ...**

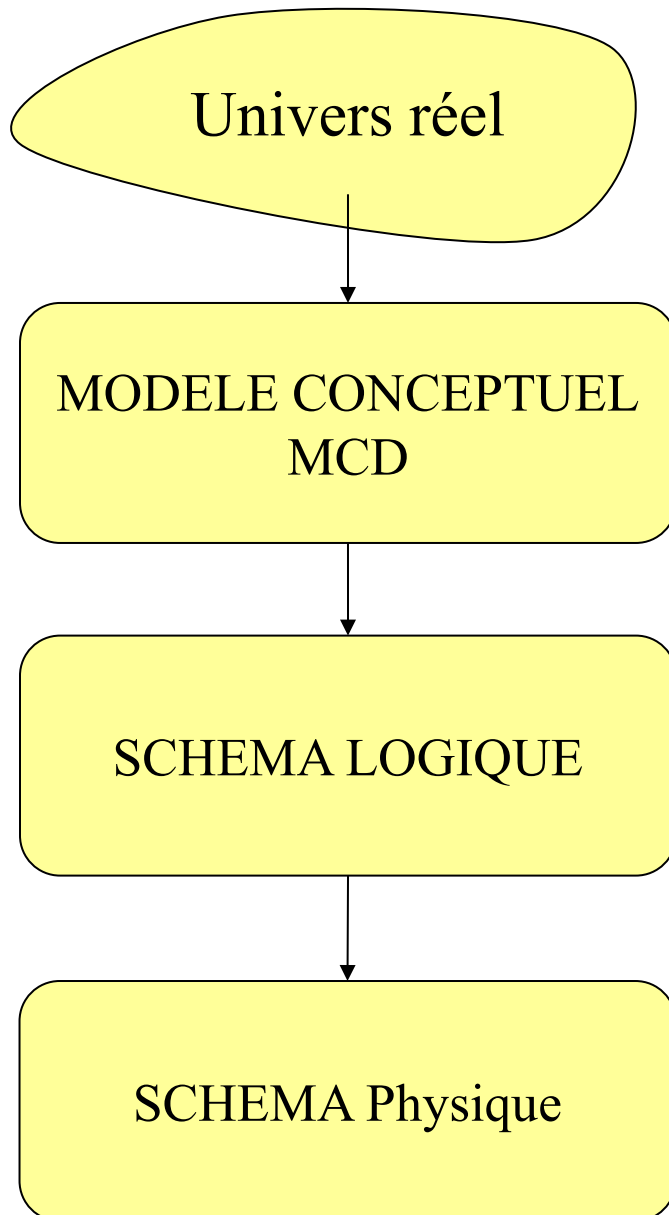


- **Cosmo DB, et autres.**





Modèle entité- Association



Modèles sémantiques /conceptuels

- Modèles orientés « conception »
- **Entité-Association**,
- Merise
- UML....

Modèles de BD /logiques

- Hiérarchique,
- Réseau
- Relationnel ...

Les modèles de BD sont souvent trop limités pour représenter directement le monde réel.

- **Conceptuel : le modèle entité Association E/A (E/R)**

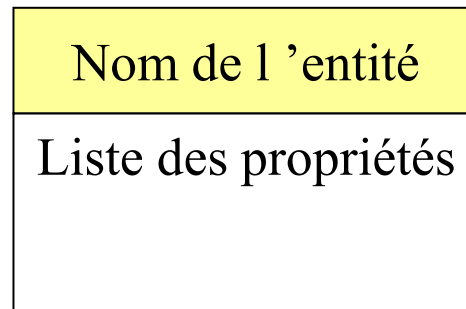
Il s'agit du formalisme graphique retenu par l'ISO ([International Organization for Standardization](#))

pour décrire l'aspect conceptuel des données à l'aide d'entités et d'associations sous forme de schémas conceptuels.



• **Conceptuel : le modèle entité Association E/A (E/R)**

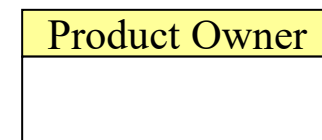
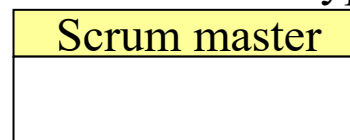
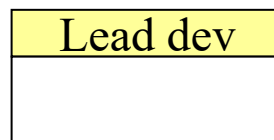
- **Entité** :représentation d ’objet matériel ou immatériel, exemple : un employé, un projet, un bulletin de paie.



- Les **propriétés** sont les données élémentaires relatives à une entité, par exemple : numéro d ’employé, nom, prénom, etc. Elles sont aussi appelées **attributs** ou caractéristiques de cette entité.

- **Type d ’entité** : correspond à un regroupement d’entités

- Ex : les entités suivantes sont des instances du type d’entité EMPLOYE



- **Conceptuel : le modèle entité Association E/A (E/R)**

Nom de l'entité
Identifiant, Propriété2, Propriété3

- **l'identifiant**

- propriété

ou

- groupe de propriétés qui sert à identifier une occurrence de cette entité de façon unique.

- **Conceptuel : le modèle entité Association E/A (E/R)**



- **Les associations**

- représente le lien entre les entités.
- Elles peuvent avoir des propriétés !

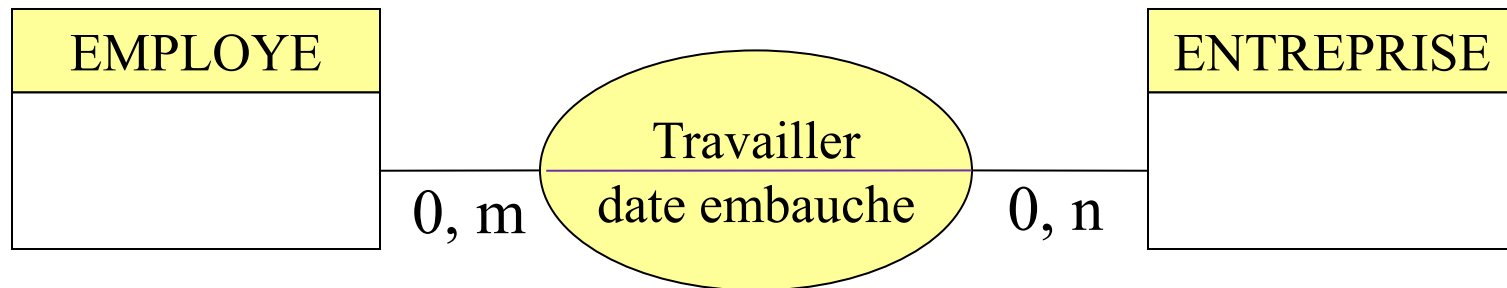
Le modèle Entité / Association a pour **objectif** :

- la détection d 'erreur de conception avant développement,
- la traduction automatique dans un modèle logique,
- Il correspond à la partie modélisation de données dans UML

(Langage de modélisation unifié).

- **Conceptuel : le modèle entité Association E/A (E/R)**

– *Les cardinalités:*



- Un/une employé peut travailler dans 0 à m entreprises.
 - A l'inverse, une entreprise peut employer entre 0 et n employés.
- La cardinalité d'une association pour une entité est constituée d'une borne **minimale** et d'une borne **maximale** :

- **Conceptuel : le modèle entité Association E/A (E/R)**

- Borne des Cardinalités :

- **Minimale** : nombre minimum de fois qu'une occurrence de l'entité participe aux occurrences de l'association, généralement 0 ou 1

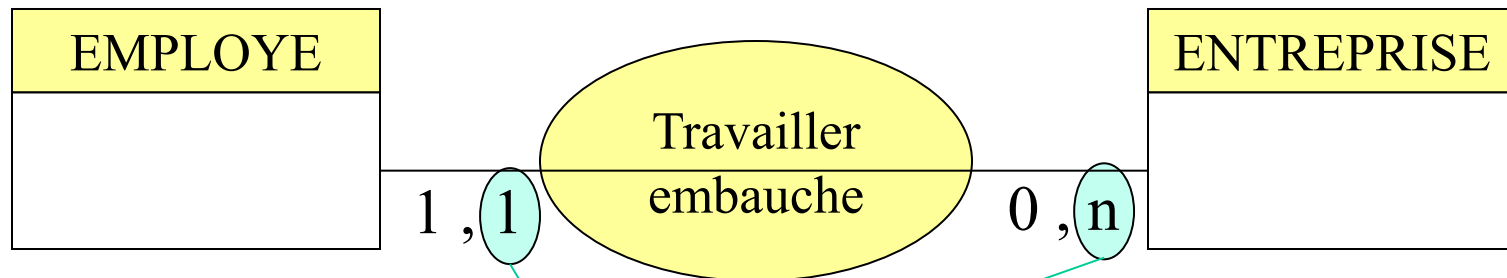
- ➔ la borne minimale exprime les contraintes d'intégrité

- **Maximale** : nombre maximum de fois qu'une occurrence de l'entité participe aux occurrences de l'association, généralement 1 ou n

- ➔ la borne maximale est nécessaire pour la conception de la base de donnée.

- **Conceptuel : le modèle entité Association E/A (E/R)**

– *Des cardinalités vers les Liens:*



En notant les cardinalités, nous déterminons 3 profils de liens :

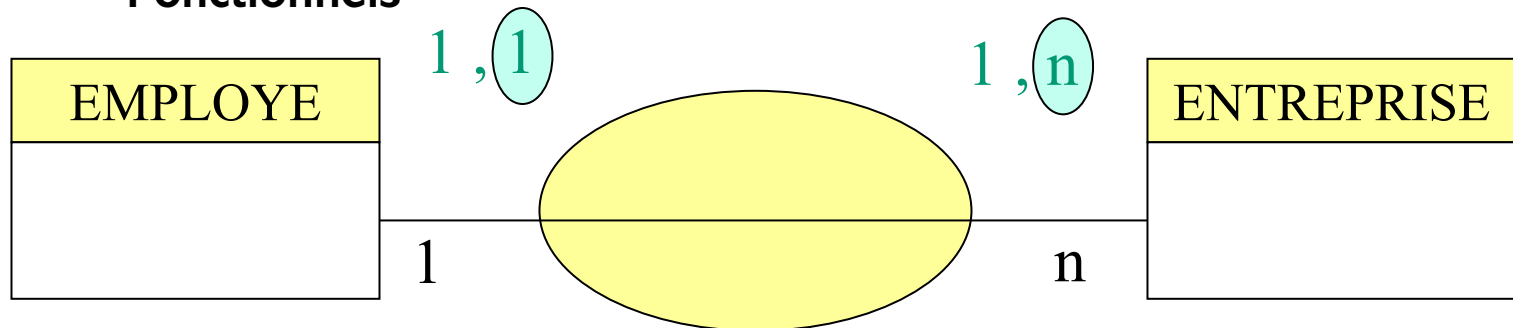
- » fonctionnel 1:n
- » hiérarchique n:1
- » maillé n:m

- **Conceptuel : le modèle entité Association E/A (E/R)**

Les Liens : représente le type de lien entre les entités.

fonctionnel 1:n hiérarchique n:1 maillé n:m

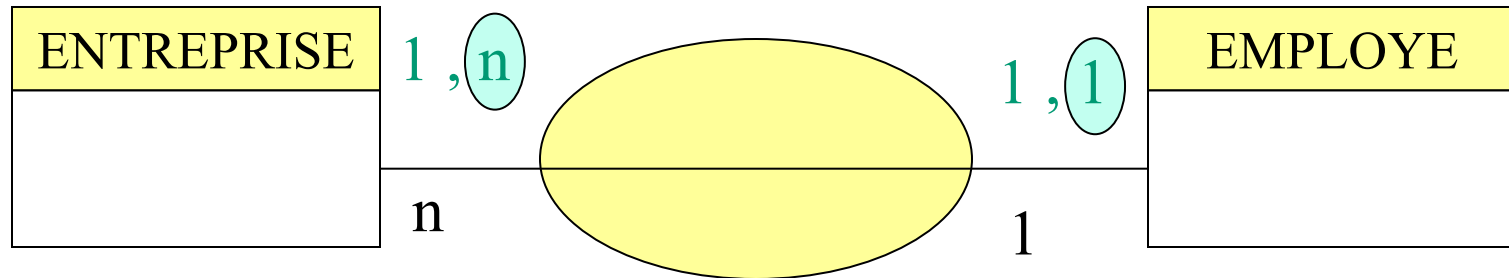
- **Fonctionnels**



Une instance de EMPLOYE ne peut être associée qu'à une instance de ENTREPRISE dans cet exemple.

- **Conceptuel : le modèle entité Association E/A (E/R)**

- **Liens hiérarchiques**
- représente le lien entre les entités.

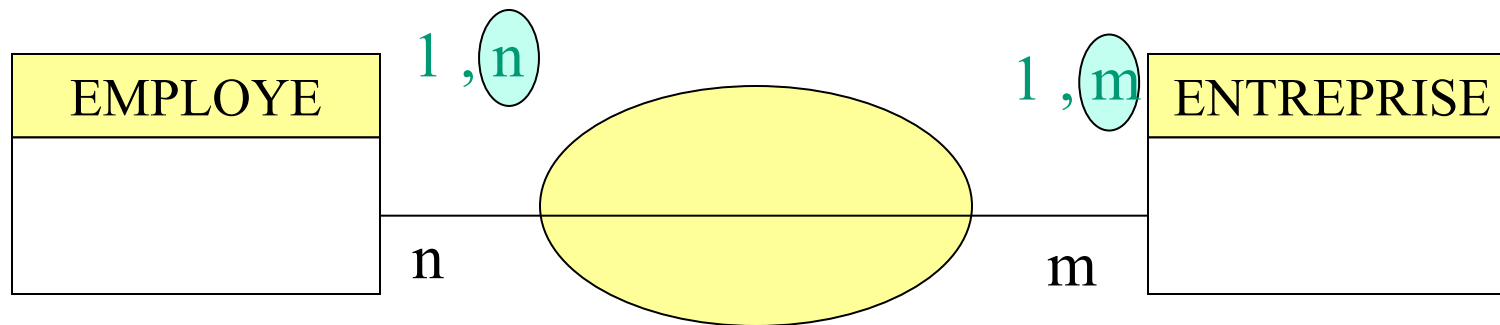


- Une instance de ENTREPRISE peut être associée à plusieurs instances de EMPLOYE dans cet exemple.

- **Conceptuel : le modèle entité Association E/A (E/R)**

- **Les liens maillés**

- représente le lien entre les entités.



- Une instance de EMPLOYE peut être travailler dans plusieurs ENTREPRISE.

- **Résumé de l'objectif de ce modèle E/A:**

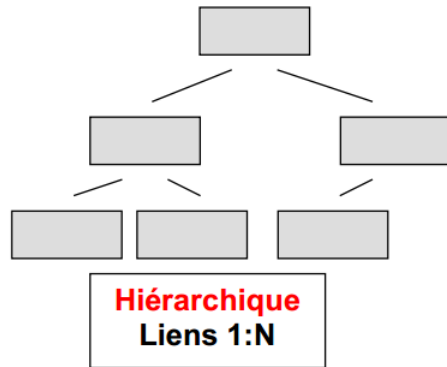
- la détection d'erreur de conception avant développement
 - la traduction automatique dans un modèle logique
 - Il est supporté par les outils de type CASE /Rational ROSE.
 - *Il correspond à la partie modélisation de données dans UML .*



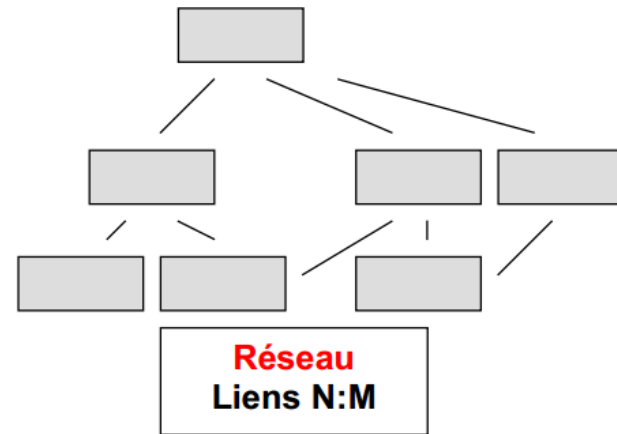
Modèles - passage du conceptuel au logique

- **Schémas logiques et Modèles de bases de données**
- **Les déclinaisons des schémas de conceptuels vers logiques**

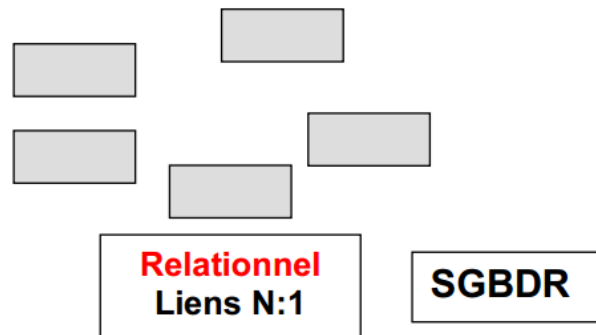
Le modèle arbre



Le modèle graphe



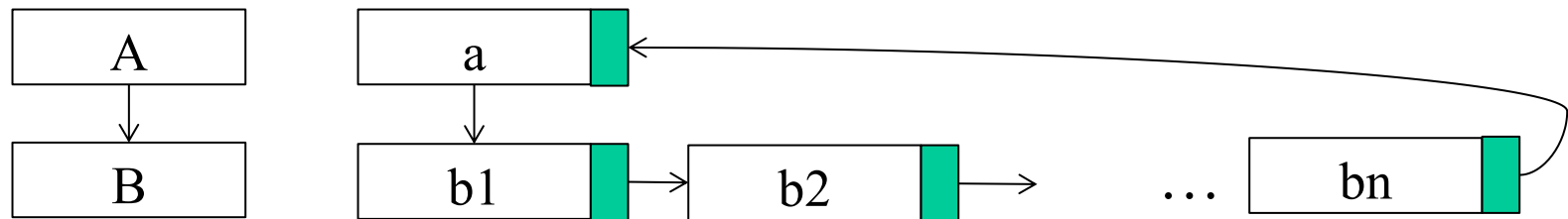
Le modèle relationnel



• Schéma logique : Graphe, hiérarchique, Réseau, Relationnel

Modèles issus de GRAPHES

- Modèle logique organisé sous forme de « **graphe** »
 - Accès aux données par navigation ou adressage par liens de chaînage.
 - Implémentation d'un lien par une liste circulaire



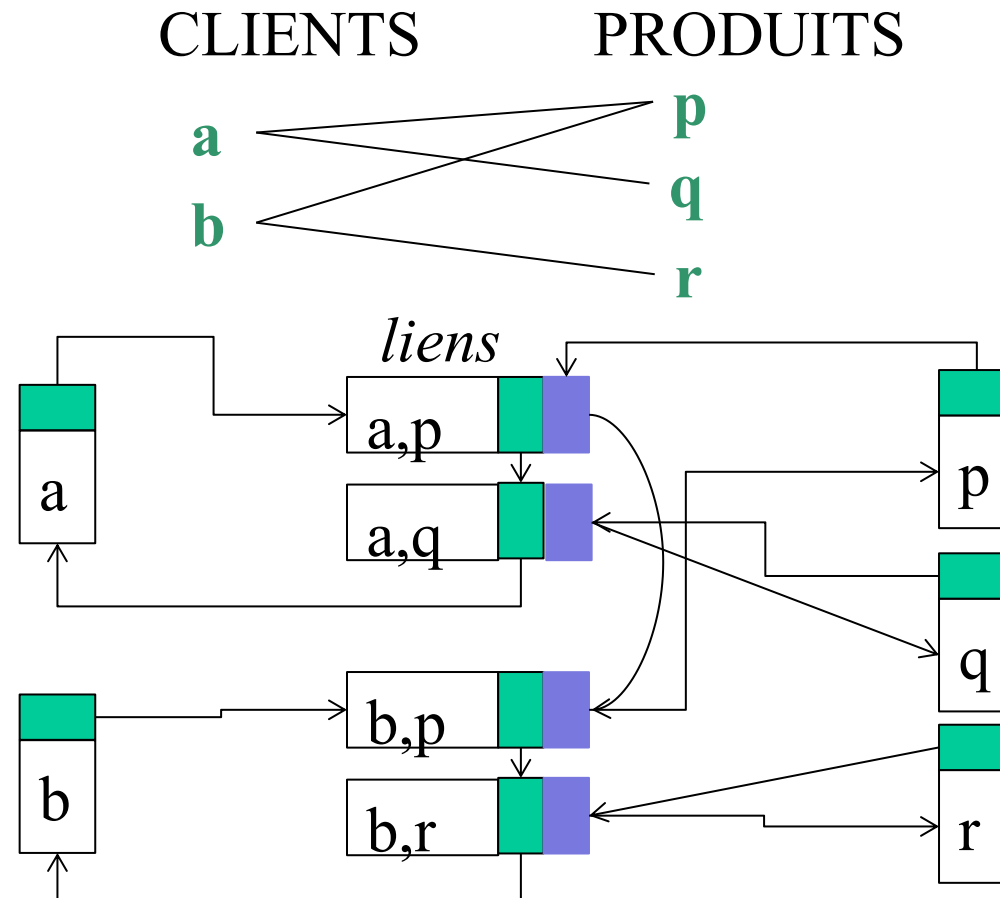
- Modèle **réseau** (CODASYL): schéma avec nœud et arc : « diagramme de Bachman »



• **Schéma logique** : Graphe, hiérarchique, Réseau, Relationnel

Modèles issus de GRAPHERS

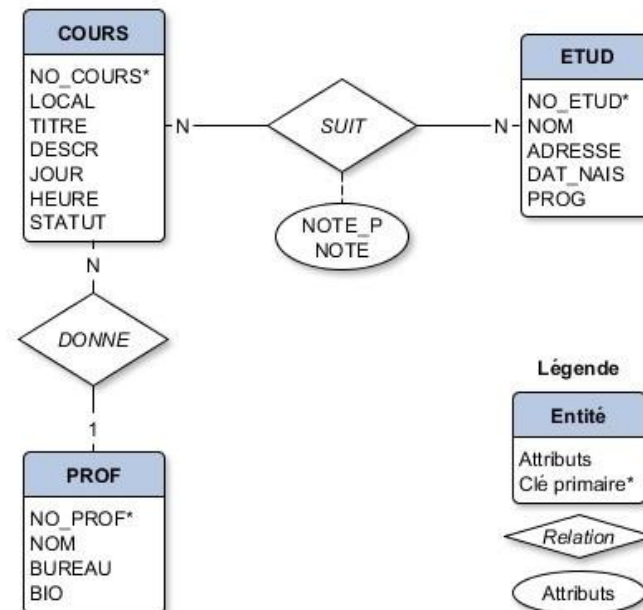
- modèle Réseau : schéma d'une association N:M par 2 liens
CODASYL.



• Schéma logique : Modèles

Modèle Relationnel

- imaginé par CODD
- Fondé sur la notion mathématique de Relation
- Données organisées en tables avec adressage relatif
- Accès par adressage relatif géré par le SGBD.
- Stratégie d'accès gérée par le SGBD

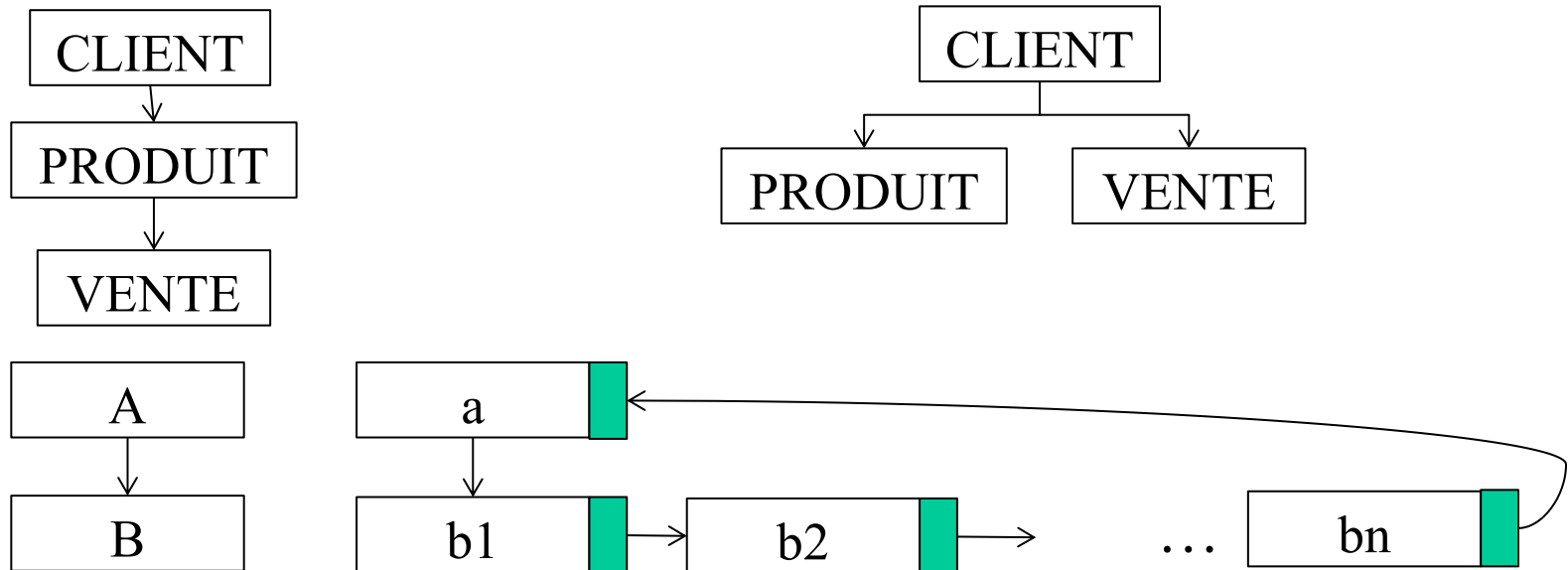


• Schéma logique : Modèles

Modèles issus de GRAPHERS (hiérarchiques et réseaux)

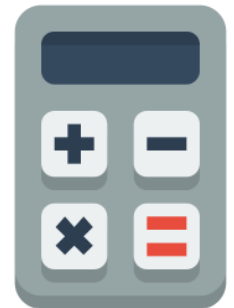
– Modèle hiérarchique :

- schémas avec nœud (segment) et arc (lien hiérarchique).
- Plusieurs choix possibles d'arborescences
- Dissymétrie des traitements selon le choix de la racine, ex : lister les produits achetés par le client a ou lister les clients qui ont achetés le produit b2...
- Adapté aux organisations à structure arborescente





Modèle logique Relationnel et algèbre



Exemple de cours généré avec GPT-4o « donnez moi un exemple d'algèbre relationnel »

L'algèbre relationnelle est un langage formel utilisé pour manipuler des relations dans une base de données relationnelle.

Voici un exemple simple utilisant une base de données fictive sur des étudiants et des cours.

Imaginons que nous avons les deux relations suivantes :

1.Étudiants (id, nom, age)

id	nom	age
1	Alice	20
2	Bob	22
3	Charlie	21

2.Cours (id, titre, id_etudiant)

id	titre	id_etudiant
1	Mathématiques	1
2	Physique	2
3	Informatique	1
4	Chimie	3

Exemple d'opération en algèbre relationnelle

Nous voulons trouver le nom des étudiants qui suivent le cours de "Mathématiques". Voici comment nous pourrions procéder :

1.Sélectionner les cours intitulés "Mathématiques" :

Utilisons l'opération de sélection (σ) : Cours_Mathematiques= σ titre='Mathematiques'(Cours)

Ce qui nous donne la relation :

id	titre	id_etudiant
1	Mathématiques	1

Initiations aux bases de données -48- modèle Relationnel

- *Exemple de cours généré avec GPT-4o « donnez moi un exemple d'algèbre relationnel »*

2. Joindre avec la relation Étudiants :

Ensuite, nous utilisons l'opération de jointure (\bowtie) pour associer les cours avec les étudiants :

Resultat=Cours_Mathematiques \bowtie Etudiants

Cela nous donnera une relation qui contient les informations des étudiants qui suivent "Mathématiques":

id	nom	age	id	titre	id etudiant
1	Alice	20	1	Mathématiques	1

3. Projection pour obtenir uniquement le nom des étudiants :

Enfin, nous appliquons l'opération de projection (π) pour obtenir seulement les noms des étudiants :

Noms_Etudiants= π nom(Resultat)

Ce qui donne le résultat final :

Nom
Alice

Conclusion

Cet exemple illustre comment utiliser les opérations de l'algèbre relationnelle pour interroger une base de données. Les opérations de sélection, de jointure et de projection permettent d'extraire des informations spécifiques d'une ou plusieurs relations.

Schéma logique : Modèle Relationnel

- 1970, CODD présente le modèle relationnel : schéma logique représenté par des RELATIONS
- **LE « SCHÉMA RELATIONNEL »**
 - C'est l'ensemble des RELATIONS qui modélisent le monde réel
 - Les relations représentent les **entités** du monde réel (des personnes, des objets, etc.) ou les **associations** entre ces entités.
 - Passage d'un schéma conceptuel E-A à un schéma logique relationnel
 - Une **entité** est représentée par la relation :
 - **nom_de_l'entité (liste des attributs de l'entité)**
 - une **association** M:N est représentée par la relation :
 - **nom_de_l'association (liste des identifiants des entités participantes, liste des attributs de l'association)**

Exemples :

- | | |
|--|-------------|
| ➤ CLIENT (<u>IdCli</u> , nom, ville) | Entité |
| ➤ PRODUIT(<u>IdPro</u> , nom, prix, qstock) | Entité |
| ➤ VENTE (<u>IdCli</u> , <u>IdPro</u> , <u>date</u> , qte) | Association |

- **Passage d'un schéma E/A (E/R) à un schéma relationnel**

- En définissant les **entités**

- Nom (**clé1,....,cléN,attribut1, attribut2, attribut3,....., attributM**)

- Ou**

- Nom (clé1,....,cléN,attribut1, attribut2, attribut3,....., attributM)

- En définissant les **associations N,M**

- NomAssociation (**liste des identifiants des entités participantes,**
liste des attributs de l'association)

LES AVANTAGES DU MODELE RELATIONNEL LOGIQUE

- Simplicité de présentation par tables
- Opérations algébriques relationnelles
 - algèbre relationnelle (plus, moins, divise, mulitplie, etc...)
 - langage « assertionnel » ou spécialisé
- Indépendance physique
 - optimisation des accès
 - stratégie d'accès déterminée par le système
- Indépendance logique
 - concept de VUES
- Maintien de l'intégrité
 - contraintes d'intégrité définies au niveau du schéma

LES AVANTAGES DU MODELE RELATIONNEL LOGIQUE

- **Domaine**
 - Ensemble de valeurs atomiques d 'un type sémantique :
 - $NOM_SITE = \{ATOMIUM, TOUR_EFFEL, STATUE_LIBERTE, \dots\}$
 - ensemble des valeurs possibles
- **Relation**
 - Sous ensemble du produit cartésien de plusieurs domaines
- **N-uplets**
 - un élément d 'une relation est un n-uplet de valeurs (tuple in english)
- **Attributs**
 - son nom doit être porteur de sens
 - différent du nom de domaine
 - plusieurs attributs peuvent avoir le même nom de domaine
- **Schéma d 'une relation**

- Exemple de définition de quatre entités relationnelles
 - Station (**id**, nom, capacité, lieu, région, tarif)
 - Activité (**idStation**, libellé, prix)
 - Client (**id**, nom, prénom, ville, région, solde)
 - Séjour (**id**, *idClient*, *idStation*, début, fin, nbPlaces)
- Entité exprimée sous forme de table
 - exemple de la table Station

id	nom	capacité	lieu	région	tarif
va	Venusa	350	Guadeloupe	Antilles	1200
fa	Farniente	200	Seychelles	Océan Indien	1500
sa	Santalba	150	Martinique	Antilles	2000
pa	Passac	400	Alpes	Europe	1000

- Les Opérateurs de base de l'algèbre relationnel

- La restriction ou sélection: σ (unaire)



- La projection: π (unaire)



- Le produit cartésien: \times (binaire)

- L'union: \cup (binaire)

- L'intersection \cap (binaire)

- La division $/$ (binaire)

- La différence: $-$ (binaire)

- L'appartenance \in et l'exclusion \notin

- Opérateurs dérivés

- *Jointure* : \bowtie , la composition d'un produit cartésien et d'une sélection.

• Restriction/sélection

– But :

- sélectionner certains N-uplets
- Réduire la taille verticalement

– Contraintes

- Unaire
- Spécifier une condition

– Exemple : stations aux Antilles

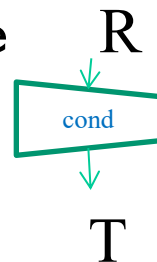
– Station_antilles <- $\sigma_{\text{région}='Antilles'}$ (Station)

id	nom	capacité	lieu	région	tarif
va	Venusa	350	Guadeloupe	Antilles	1200
fa	Farniente	200	Seychelles	Océan Indien	1500
sa	Santalba	150	Martinique	Antilles	2000
pa	Passac	400	Alpes	Europe	1000

id	nom	capacité	lieu	région	tarif
va	Venusa	350	Guadeloupe	Antilles	1200
sa	Santalba	150	Martinique	Antilles	2000

– Notation textuelle $T \leftarrow \sigma_{\text{cond}}(R)$

– Notation graphique



• Projection (unaire)

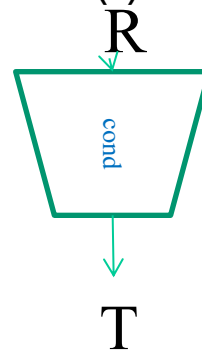
- But:
 - sélectionner certains attributs
 - Réduire la taille horizontalement
 - Supprime les doublons
- Contraintes
 - Unaire
 - Spécifier une liste d'attributs

id	nom	capacité	lieu	région	tarif
va	Venusa	350	Guadeloupe	Antilles	1200
fa	Farniente	200	Seychelles	Océan Indien	1500
sa	Santalba	150	Martinique	Antilles	2000
pa	Passac	400	Alpes	Europe	1000

Exemple : région des noms

- Station_Region <- π nom,région (Station)
- Notation textuelle T <- π condition (R)
- Notation graphique

nom	région
Venusa	Antilles
Farniente	Océan Indien
Santalba	Antilles
Passac	Europe



- exemple π region(Station)

- **Union** (Supprime les doublons)

StationAsie (id, nom, capacité, lieu, région, tarif)

id	nom	capacité	lieu	région	tarif
ma	MangaCity	50	Tokyo	Asie	1800
fa	Farniente	200	Seychelles	Océan Indien	1500

StationEurope (id, nom, capacité, lieu, région, tarif)

id	nom	capacité	lieu	région	tarif
pa	Passac	400	Alpes	Europe	1000
vt	Val Thorens	350	Alpes	Europe	1200

– **Station_AE** = StationAsie U StationEurope

id	nom	capacité	lieu	région	tarif
pa	Passac	400	Alpes	Europe	1000
vt	Val Thorens	350	Alpes	Europe	1200
ma	MangaCity	50	Tokyo	Asie	1800
fa	Farniente	200	Seychelles	Océan Indien	1500

- Intersection

Station_Mer (id, nom, capacité, lieu, région, tarif)

id	nom	capacité	lieu	région	tarif
va	Venusa	350	Guadeloupe	Antilles	1200
fa	Farniente	200	Seychelles	Océan Indien	1500
sa	Santalba	150	Martinique	Antilles	2000

Station_Antilles(id, nom, capacité, lieu, région, tarif)

id	nom	capacité	lieu	région	tarif
va	Venusa	350	Guadeloupe	Antilles	1200
sa	Santalba	150	Martinique	Antilles	2000
fo	FortAventure	50	Martinique	Antilles	2200

StationMer_Antilles = Station_Antilles \cap Station_Mer

id	nom	capacité	lieu	région	tarif
va	Venusa	350	Guadeloupe	Antilles	1200
sa	Santalba	150	Martinique	Antilles	2000

- **Produit cartésien**

Station (id, nom, capacité, lieu, région, tarif)

id	nom	capacité	lieu	région	tarif
va	Venusa	350	Guadeloupe	Antilles	1200
fa	Farniente	200	Seychelles	Océan Indien	1500
sa	Santalba	150	Martinique	Antilles	2000
pa	Passac	400	Alpes	Europe	1000

Activité (libellé, prix)

libellé	prix
randonnée	350
Plongée	150

Activité_Station (libelle, prix, id, nom, capacité, lieu, région, tarif)

= **Activité (libellé, prix) X Station (id, nom, capacité, lieu, région, tarif)**

Produit cartésien - Résultat

Activité_Station (**id**, nom, capacité, lieu, région, tarif, **libelle**, prix) =
Station(**id**, nom, capacité, lieu, région, tarif) X Activité (**libellé**, prix)

id	nom	capacité	lieu	région	tarif	libellé	prix
va	Venusa	350	Guadeloupe	Antilles	1200	Randonnée	350
va	Venusa	350	Guadeloupe	Antilles	1200	plongée	150
fa	Farniente	200	Seychelles	Océan Indien	1500	Randonnée	350
fa	Farniente	200	Seychelles	Océan Indien	1500	plongée	150
sa	Santalba	150	Martinique	Antilles	2000	Randonnée	350
sa	Santalba	150	Martinique	Antilles	2000	plongée	150
pa	Passac	400	Alpes	Europe	1000	Randonnée	350
pa	Passac	400	Alpes	Europe	1000	plongée	150

- Différence

Station_Mer (id, nom, capacité, lieu, région, tarif)

id	Nom	Capacité	Lieu	région	tarif
va	Venusa	350	Guadeloupe	Antilles	1200
fa	Farniente	200	Seychelles	Océan Indien	1500
Sa	Santalba	150	Martinique	Antilles	2000

Station_Antilles(id, nom, capacité, lieu, région, tarif)

id	nom	capacité	lieu	région	tarif
va	Venusa	350	Guadeloupe	Antilles	1200
sa	Santalba	150	Martinique	Antilles	2000
fo	FortAventure	50	Martinique	Antilles	2200

Station_MerSaufAntilles = Station_Mer - Station_Antilles

id	nom	capacité	lieu	région	tarif
fa	Farniente	200	Seychelles	Océan Indien	1500

Station_AntillesSaufMer = Station_Antilles - Station_Mer

id	nom	capacité	lieu	région	tarif
fo	FortAventure	50	Martinique	Antilles	2200

- **Division** (Élimine les doublons)

Station_Nom (nom, capacité, lieu)

Nom	Capacité	Lieu
Venusa	350	Guadeloupe
Venusa	350	Seychelles
Santalba	150	Martinique
Madinia	50	Martinique

Station_Capacité (capacité, lieu)

capacité	lieu
350	Guadeloupe
350	Seychelles

Station_Divise = Station_Nom / Station_Capacité

nom
Venusa

- Jointure

Station_Nom (nom, capacité)

Nom	Capacité
Venusa	350
Farniente	200
Santalba	400

Region_Lieu (region, lieu)

Région	lieu
Antilles	Guadeloupe
Océan Indien	Seychelles
Antilles	Martinique

Station_Jointe = Station_Nom \bowtie Region_Lieu (capacité < 400 et lieu \neq Martinique) ou σ capacité < 400 et lieu \neq Martinique (Station_Nom X Region_Lieu)

Nom	Capacité	Lieu	Région
Venusa	350	Guadeloupe	Antilles
Venusa	350	Seychelles	Océan Indien
Farniente	200	Guadeloupe	Antilles
Farniente	200	Seychelles	Océan Indien

- Jointure conservant une « clé étrangère » et l'intégrité fonctionnelle :

Station_Nom (nom, capacité)

Nom	Capacité
Venusa	350
Farniente	200
Santalba	400

Capacité_Lieu (capacité, lieu)

capacité	lieu
350	Guadeloupe
400	Seychelles
200	Martinique

Station_Jointe = Station_Nom \bowtie Capacité_Lieu (capacité < 400 et lieu != Martinique) OU σ capacité < 400 et lieu != Martinique (Station_Nom X Capacité_Lieu)

Nom	Capacité	Lieu
Venusa	350	Guadeloupe

- Clé d'une relation

- primaire :

- attribut (ou groupe d'attributs) qui détermine tous les autres
 - PRODUIT (no_produit, nom, prix_HT)
 - une clé détermine de manière unique une n-uplet
 - une relation peut posséder plusieurs clés **candidates**
 - *si nom de produit est unique, nom peut devenir la clé primaire.*

- étrangère ou clé secondaire :

- attribut (ou groupe d'attributs) qui fait référence à la clé primaire d'une autre relation
 - CATEGORIE(no_categ, designation, tva)
 - PRODUIT(no_produit, nom, marque, no_categ, prixHT)
 - > no_categ est clé étrangère dans PRODUIT, c'est la clé primaire de CATEGORIE

- **Normalisation**

- Afin de réduire les doublons.
- Définition des formes normales ajoutés au schéma relationnel :
 - **1ère forme : 1FN** : une relation est en 1FN si tout attribut est atomique à savoir non décomposable.

ELEVE(**n_eleve**, nom_prenom, liste_notes) devient en 1FN :

- ELEVE(**n_eleve**, nom, prenom)
- NOTE(**n_eleve**, matiere, note)

- **Définition des formes normales** ajoutés au schéma relationnel :
 - **2ème forme : 2FN** : une relation est en 2FN si :
 - elle est en 1FN
 - tout attribut n'appartenant pas à la clé ne dépend pas que d'une partie de la clé **ou**
 - aucun attribut ne faisant pas partie de la clé primaire ne doit dépendre que d'une partie de la clé primaire.

COMMANDE(date, n_cli, num_produit, qte, prix_unitaireHT)

Le prix_unitaire HT est dépendant de n°produit : partie de la clé !

Manière de le mettre en 2FN

- COMMANDE (date, n_cli, num_produit, qte)
- PRODUIT(num_produit, prix_unitaire_HT)

! Ne pas respecter la 2FN entraîne des redondances. Cela gaspille de l'espace de stockage, et pose aussi le problème de la mise à jour des données.

- **Définition des formes normales ajoutés au schéma relationnel :**
 - **3ème forme : 3FN** : une relation est en 2FN si :
 - elle est en 2FN
 - si tout attribut n'appartenant pas à la clé primaire ne dépend pas d'un attribut non clé

Ou

 - aucun attribut ne faisant pas partie de la clé primaire ne doit dépendre d'une partie des autres attributs ne faisant pas non plus partie de la clé primaire.

PERSONNE (id_personne, civilité, nom, prenom, sexe) devient en 3FN :

- PERSONNE (id_personne , civilité, nom, prenom)
- CIVILITE (civilité, sexe)

- **Intégrité**

- de domaine

- les valeurs d'une colonne de relation doivent appartenir au domaine correspondant (*liste de valeurs possibles*)

- contrôle des valeurs des attributs
 - contrôle entre valeurs des attributs

- de clé

- les valeurs de clés primaires doivent être uniques

- uniques
 - non nulles
- } => unicité de clé
} => unicité des n-uplets

- référentielle

- les valeurs de clés étrangères sont nulles ou sont des valeurs de la clé primaires auxquelles elles font référence

- relations indépendantes

Séjour (*id, idClient, idStation, début, fin, nbPlaces*)



Le langage SQL - 1er

Le SQL (Structured Query Language) est un langage standardisé utilisé pour interagir avec des bases de données relationnelles. Il permet aux utilisateurs de créer, lire, mettre à jour et supprimer des données (opérations désignées sous l'acronyme CRUD : Create, Read, Update, Delete). Voici un aperçu des principales fonctionnalités et concepts du SQL :

1. Création de bases de données et de tables

- **CREATE DATABASE** : pour créer une nouvelle base de données.
- **CREATE TABLE** : pour définir une nouvelle table avec des colonnes de données spécifiques.

SQL - CREATE DATABASE ExempleDB;

```
CREATE TABLE Utilisateurs ( Id INT PRIMARY KEY, Nom VARCHAR(100), Email VARCHAR(100) );
```

2. Insertion de données

- **INSERT INTO** : pour ajouter des nouvelles lignes dans une table.

SQL - INSERT INTO **Utilisateurs** (Id, Nom, Email) VALUES (1, 'Alice', 'alice@example.com');

3. Lecture de données

- **SELECT** : pour interroger et récupérer des données à partir d'une table.

SQL - SELECT * FROM **Utilisateurs**; -- Récupère toutes les colonnes
SELECT Nom, Email FROM **Utilisateurs** WHERE Id = 1; -- Récupère des colonnes spécifiques avec une condition

4. Mise à jour de données

- **UPDATE** : pour modifier des données existantes dans une table.

SQL - UPDATE **Utilisateurs** SET Email = 'alice_new@example.com' WHERE Id = 1;

5. Suppression de données

- **DELETE** : pour supprimer des lignes d'une table.

SQL - DELETE FROM **Utilisateurs** WHERE Id = 1;

6. Clauses et filtres

- Les clauses comme **WHERE**, **ORDER BY**, **GROUP BY**, et **HAVING** permettent de filtrer et de structurer les résultats des requêtes.

SQL - SELECT Nom FROM **Utilisateurs** WHERE Email LIKE '%@example.com'
ORDER BY Nom;

7. Jointures

- Les jointures (**JOIN**) permettent de combiner des lignes de deux ou plusieurs tables en fonction d'une condition liée.

SQL - SELECT Utilisateurs.Nom, Commandes.Montant FROM **Utilisateurs**
JOIN **Commandes** ON Utilisateurs.Id = Commandes.UtilisateurId;

8. Indexation

- Les index améliorent la vitesse des requêtes sur des colonnes spécifiques, mais peuvent ralentir l'insertion ou la mise à jour des données.

9. Transactions

- Les transactions permettent d'exécuter des ensembles d'instructions SQL de manière atomique, garantissant que toutes les opérations réussissent ou échouent ensemble.

SQL -BEGIN TRANSACTION; -- Instructions **SQL COMMIT;** ou **ROLLBACK** en cas d'erreur

10. Fonctions et procédures

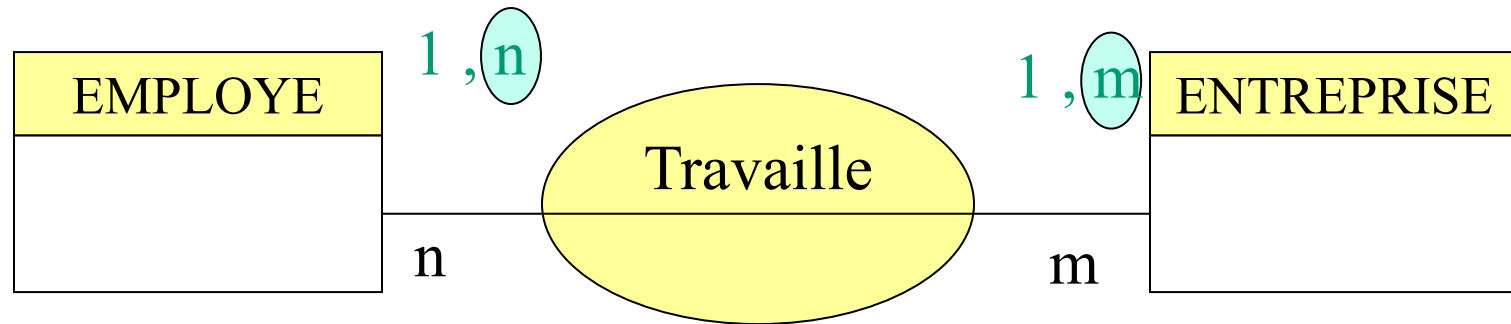
- SQL permet également créer des fonctions et des procédures stockées pour automatiser des tâches spécifiques.

Conclusion : SQL est un langage puissant et flexible qui est essentiel pour la gestion des données dans les systèmes de bases de données relationnelles. Maîtriser SQL est indispensable pour les développeurs, les analystes de données et les administrateurs de bases de données.



Règles de traduction du modèle Conceptuel en modèle Relationnel

- **Rappel entité Association E/A (E/R)**
 - Les Entités, les liens et les cardinalités



TRADUCTION du MCD vers un LMD Relationnel.

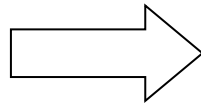
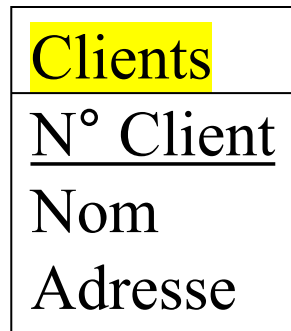
- Pour traduire un Modèle Conceptuel de Données en Modèle Relationnel de Données il suffit d'appliquer **5 règles**.
- **Notations** : on dit qu'une association binaire (entre deux entités) ou réflexive est de type :
 - 1 : 1 (un à un) si aucune des deux cardinalités maximales n'est de n,
 - 1 : n (un à plusieurs) si une des deux cardinalités maximales est de n,
 - n : m (plusieurs à plusieurs) si les deux cardinalités maximales sont de n.
- Un schéma relationnel ne peut pas faire la différence entre 0,n et 1,n.
Par contre, il peut la faire entre 0,1 et 1,1.

TRADUCTION du MCD vers un MLD Relationnel.

- Règle n°1 :

Toute entité devient une table dans laquelle les attributs deviennent les colonnes.

L'identifiant de l'entité constitue la clé primaire de la table.



Clients(N° Client, Nom, Adresse)

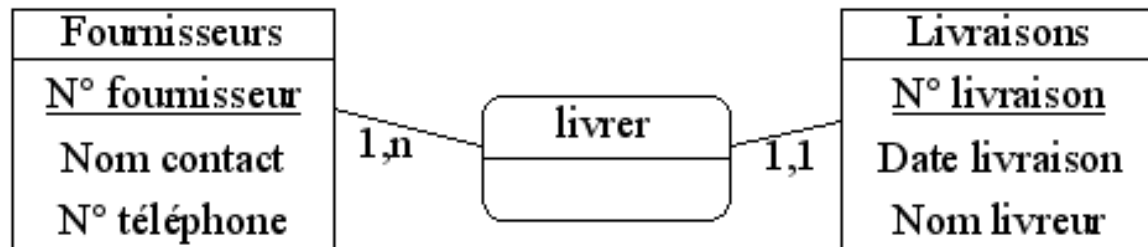
TRADUCTION du MCD vers un MLD Relationnel..

Règle n°2 :

Une association binaire de type 1 : n disparaît au profit d'une clé étrangère dans la table côté 0, 1 ou 1,1 qui fait référence à la clé primaire de l'autre table.

Cette clé étrangère ne peut pas recevoir la valeur vide ou NULL, si la cardinalité est 1, 1.

Exemple :



L'association **livrer** de la figure précédente se traduit par :

Fournisseurs(N° fournisseur, nom contact, n° téléphone),

Livraisons(N° livraison, date livraison, nom livreur, #N° fournisseur (non vide)).

L'association "de type 1 : n ne devrait pas contenir d'attributs, mais dans ce cas, ils glissent vers la table côté 1:1.

TRADUCTION du MCD vers un MLD Relationnel..

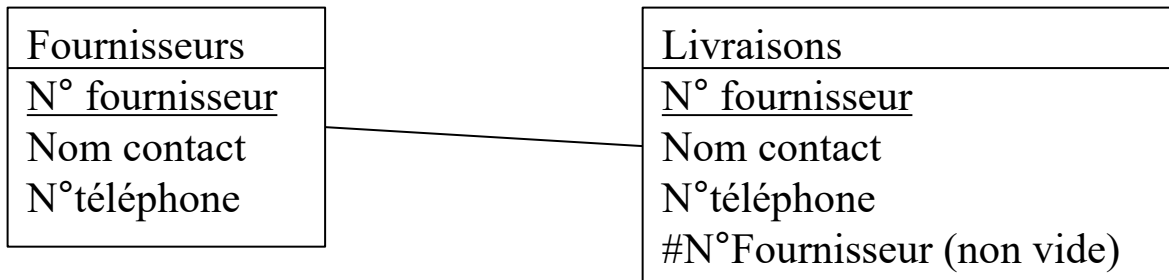
Autre écriture

Exemple :

Fournisseurs(N° fournisseur, nom contact, n° téléphone),

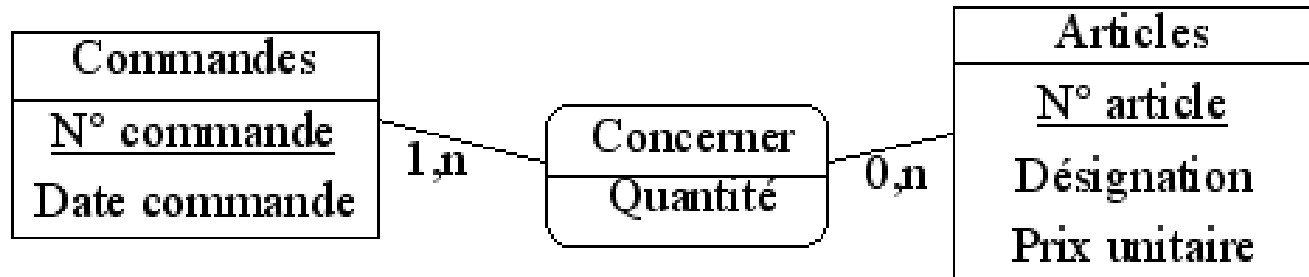
Livraisons(N° livraison, date livraison, nom livreur, #N° fournisseur (non vide)).

Peut s'écrire :



TRADUCTION du MCD vers un MLD Relationnel.

- Règle n°3 :
- une association binaire de type n : m devient une table supplémentaire (parfois appelée **table de jonction**, table de **jointure** ou table **d'association**) dont la clé primaire est composée de deux clés étrangères (qui référencent les deux clés primaires des deux tables en association). Les attributs de l'association deviennent des colonnes de cette nouvelle table
- Exemple



- *Autre écriture*

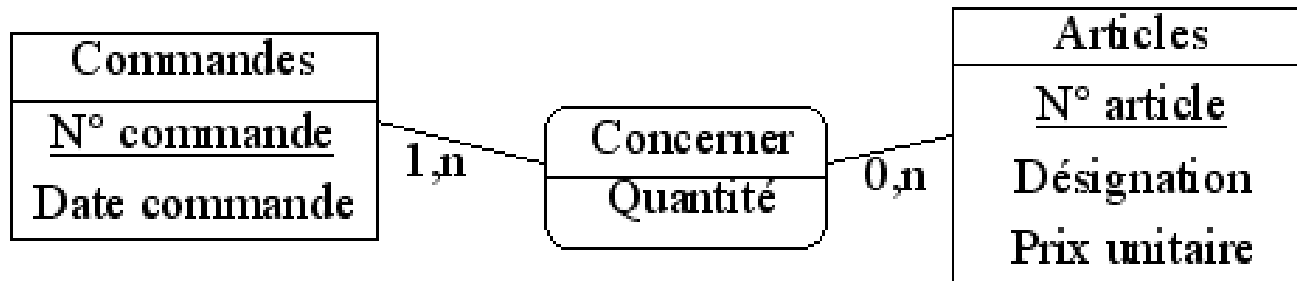
Commandes(N° commande, date_commande),

Articles (N° Article, désignation, Prix unitaire).

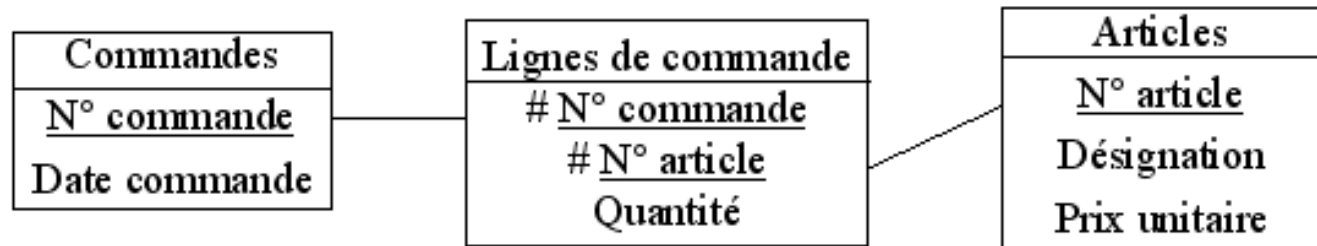
Ligne_de_commande(#N° commande, #N° Article, Quantité)

TRADUCTION du MCD vers un MLD Relationnel.

- **Règle n°3** : table de jonction, table de jointure ou table d'association)
Exemple



- Traduction



- *Autre écriture*

Commandes(N° commande, date_commande),

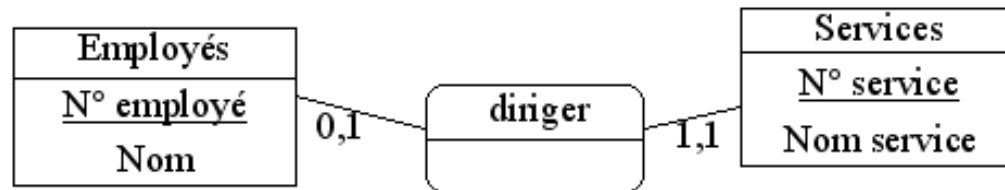
Articles (N° Article, désignation, Prix unitaire).

Ligne_de_commande(#N° commande, #N° Article, Quantité)

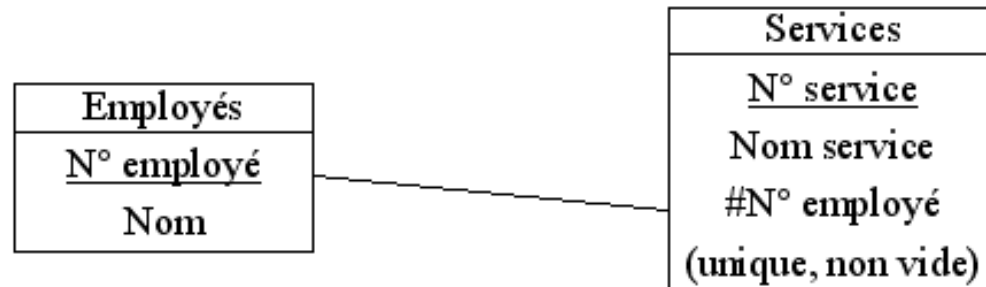
TRADUCTION du MCD vers un MLD Relationnel.

Règle n°4 : une association binaire de type 1 : 1 est traduite comme une association binaire de type 1 : n sauf que la clé étrangère se voit imposer une **contrainte d'unicité** (que des valeurs distinctes) en plus d'une éventuelle contrainte « non vide » ou de non vacuité.

Il devrait y avoir au moins un côté de cardinalité 0,1. C'est alors dans la table du côté opposé que doit aller la clé étrangère. Exemple



Traduction



Employés (N°employé, nom),

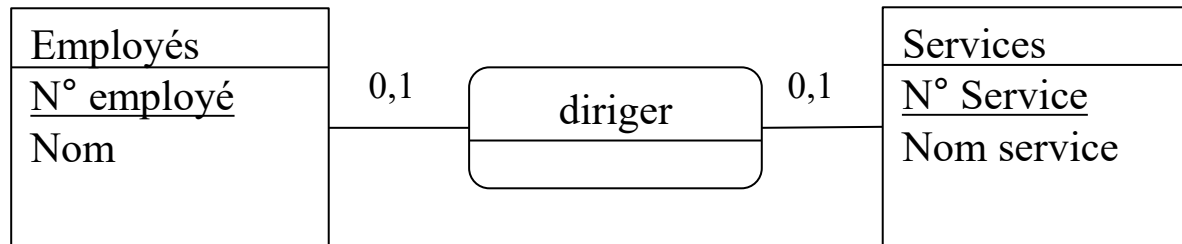
Service (N°Service, Nom_service, #N°employé (unique, non vide)).

TRADUCTION du MCD vers un MLD Relationnel.

Règle n°4 –

Suite : Si les deux côtés sont de cardinalité 0,1 alors la clé étrangère peut être placée indifféremment dans l'une des deux tables

Exemple



Traduction

Employés(N°employé, nom)

Service(N°Service, Nom_service, #N°employé(unique,non vide))

Ou

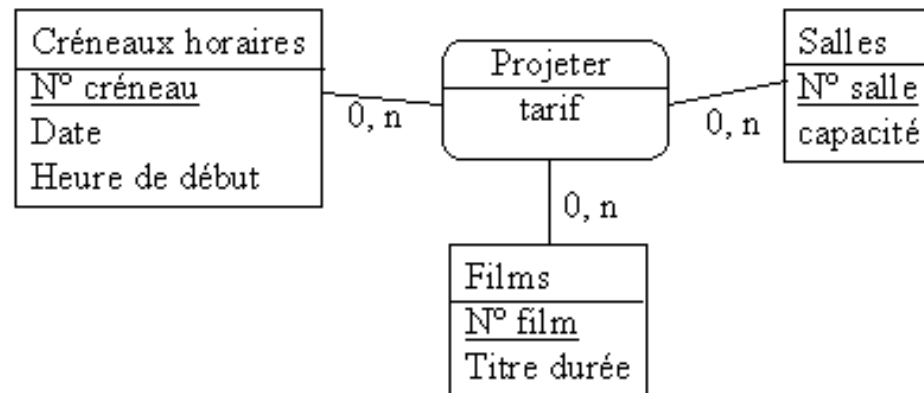
Employés(N°employé, nom, #N°Service(unique,non vide))

Service(N°Service, Nom_service)

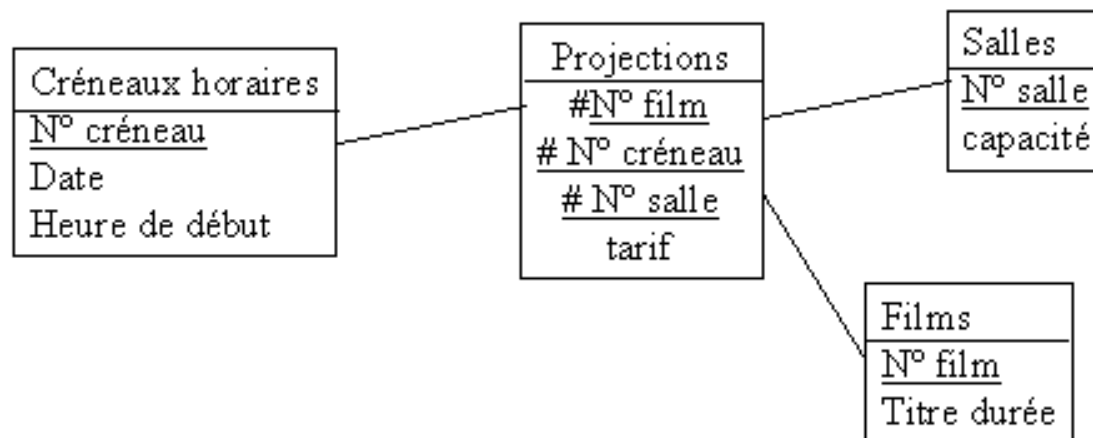
TRADUCTION du MCD vers un MLD Relationnel.

Règle n°5 : une association non binaire est traduite par une table supplémentaire dont la clé primaire est composée d'autant de clés étrangères que d'entités en association. Les attributs de l'association deviennent des colonnes de cette nouvelle table.

Exemple :



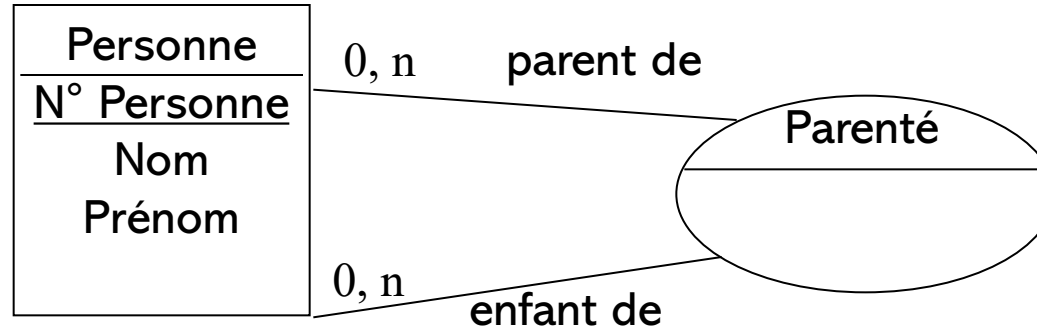
Traduction



TRADUCTION du MCD vers un MLD Relationnel.

cas de la relation réflexive : Dans certains cas, elle peut se traduire par une relation porteuse de deux attributs, duplications de l'identifiant de l'objet, et toutes deux renommées.

Exemple



Traduction

- Personne(N° Personne, Nom, Prénom),
- Parenté (N° Parent, N°Enfant).

TRADUCTION du MCD vers un MLD Relationnel. de l'algèbre vers le SQL

– Selection ou restriction : σ

- `SELECT * FROM R WHERE Condition (dateEpreuve >= 01/10/04)`

NoEt	matiere	dateEpreuve	note
12	Maths	04/10/04	12
17	Géographie	07/10/04	9
12	Géographie	07/10/04	15
19	Maths	03/09/04	13
23	Droit	08/09/04	8
17	Droit	08/09/04	14



NoEt	matiere	dateEpreuve	note
12	Maths	04/10/04	12
17	Géographie	07/10/04	9
12	Géographie	07/10/04	15

– Projection : π

- `SELECT DISTINCT(attribut) FROM LesRésultats WHERE Condition (NoEt > 16)`

NoEt	matiere	dateEpreuve	note
12	Maths	04/10/04	12
17	Géographie	07/10/04	9
12	Géographie	07/10/04	15
19	Maths	03/09/04	13
23	Droit	08/09/04	8
17	Droit	08/09/04	14



NoEt
17
23

TRADUCTION du MCD vers un LMD Relationnel.

– Renommage d 'une relation – exemple sur le produit cartésien

- un alias peut désigner une relation, il est valide pour la requête courante et permet de simplifier la lecture d 'une expression

- ex :

```
SELECT DISTINCT ( NoEt, nomEnseignant)
```

```
FROM LesInscription R1,LesEnseignants R2
```

```
WHERE R1.matiere = R2.matiere
```

NoEt	matiere
12	Maths
17	Géographie
12	Géographie
19	Maths
23	Droit
17	Droit
12	Droit
17	Maths

matiere	nomEnseignant
Maths	Dupond
Géographie	Durand
Droit	Leblanc



NoEt	nomEnseignant
12	Dupond
17	Durand
12	Durand
19	Dupond
23	Leblanc
17	Leblanc
12	Leblanc
17	Dupond

TRADUCTION du MCD vers un LMD Relationnel.

– Produit cartésien : \times

- SELECT * FROM R,S

- ex :

```
SELECT DISTINCT ( R1.NoEt, R2.NoEt)
```

```
FROM LesResultats R1, LesResultats R2
```

NoEt	matiere	dateEpreuve	note
12	Maths	04/10/04	12
17	Géographie	07/10/04	9
12	Géographie	07/10/04	15
19	Maths	03/09/04	13
23	Droit	08/09/04	8
17	Droit	08/09/04	14



NoEt	NoEt
12	12
12	17
12	19
12	23
17	12
17	17
17	19
17	23
19	12
19	17
19	19
19	23
23	12
23	17
23	19
23	23

TRADUCTION du MCD vers un LMD Relationnel.

– Jointure : ⋈

- SELECT * FROM R,S WHERE Condtion

- ex :

```
SELECT DISTINCT (NoEt, nomEnseignant)
```

```
FROM LesInscriptions, LesEnseignants
```

```
WHERE LesInscriptions.matiere = LesEnseignants. matiere
```

NoEt	matiere
12	Maths
17	Géographie
12	Géographie
19	Maths
23	Droit
17	Droit
12	Droit
17	Maths

matiere	nomEnseignant
Maths	Dupond
Géographie	Durand
Droit	Leblanc



NoEt	nomEnseignant
12	Dupond
17	Durand
12	Durand
19	Dupond
23	Leblanc
17	Leblanc
12	Leblanc
17	Dupond

TRADUCTION du MCD vers un LMD Relationnel.

- Traduction de l'appartenance \in

```
SELECT * FROM R
```

```
WHERE R.attribut IN ( Select attribut FROM S WHERE C )
```

- Traduction de l'exclusion \notin

```
SELECT * FROM R
```

```
WHERE R.attribut NOT IN ( Select attribut FROM S WHERE C )
```

TRADUCTION du MCD vers un LMD Relationnel.

– Traduction de l'Union U

```
SELECT * FROM R WHERE Condition
```

```
UNION
```

```
SELECT * FROM S WHERE Condition
```

– Exemple :

```
SELECT NoEt FROM LesInscriptions WHERE matiere='Maths '
```

```
UNION
```

```
SELECT NoEt FROM LesInscriptions WHERE matiere = 'Géographie'
```

NoEt	matiere
12	Maths
17	Géographie
12	Géographie
19	Maths
23	Droit
17	Droit
12	Droit
17	Maths



NoEt
12
17
19

TRADUCTION du MCD vers un LMD Relationnel.

– Traduction de l'intersection \cap

```
SELECT * FROM R WHERE Condition
```

```
INTERSECT
```

```
SELECT * FROM S WHERE Condition
```

– Exemple :

```
SELECT NoEt FROM LesInscriptions WHERE matiere='Maths '
```

```
INTERSECT
```

```
SELECT NoEt FROM LesInscriptions WHERE matiere = 'Géographie'
```

NoEt	matiere
12	Maths
17	Géographie
12	Géographie
19	Maths
23	Droit
17	Droit
12	Droit
17	Maths



NoEt
12
17

TRADUCTION du MCD vers un LMD Relationnel.

– Traduction de la différence : opérateur –

SELECT * FROM R

MINUS

SELECT * FROM S

Exemple :

SELECT NoEt FROM LesInscriptions WHERE matiere='Droit'

MINUS

SELECT NoEt FROM LesResultats WHERE matiere = 'Droit'

NoEt	matiere
12	Maths
17	Géographie
12	Géographie
19	Maths
23	Droit
17	Droit
12	Droit
17	Maths

NoEt	matiere	dateEpreuve	note
12	Maths	04/10/04	12
17	Géographie	07/10/04	9
12	Géographie	07/10/04	15
19	Maths	03/09/04	13
23	Droit	08/09/04	8
17	Droit	08/09/04	14



NoEt
12

- **Division** (Élimine les doublons)

Station_Nom (nom, capacité, lieu)

Nom	Capacité	Lieu
Venusa	350	Guadeloupe
Venusa	350	Seychelles
Santalba	150	Martinique
Madinia	50	Martinique

Station_Capacité (capacité, lieu)

capacité	lieu
350	Guadeloupe
350	Seychelles

Station_Divise = Station_Nom / Station_Capacité

nom
Venusa

TRADUCTION du MCD vers un LMD Relationnel.

– Traduction de la division : La division / (binaire)

Pas d'équivalent en SQL !

La division de R/S devient

SELECT A FROM R

MINUS

(SELECT * FROM R WHERE A,B IN (
 (SELECT R.A, S.B FROM R,S

MINUS

SELECT A,B FROM R)

)

Nom	Capacité	Lieu
Venusa	350	Guadeloupe
Venusa	350	Seychelles
Santalba	150	Martinique
Madinia	50	Martinique

capacité	lieu
350	Guadeloupe
350	Seychelles

nom
Venusa

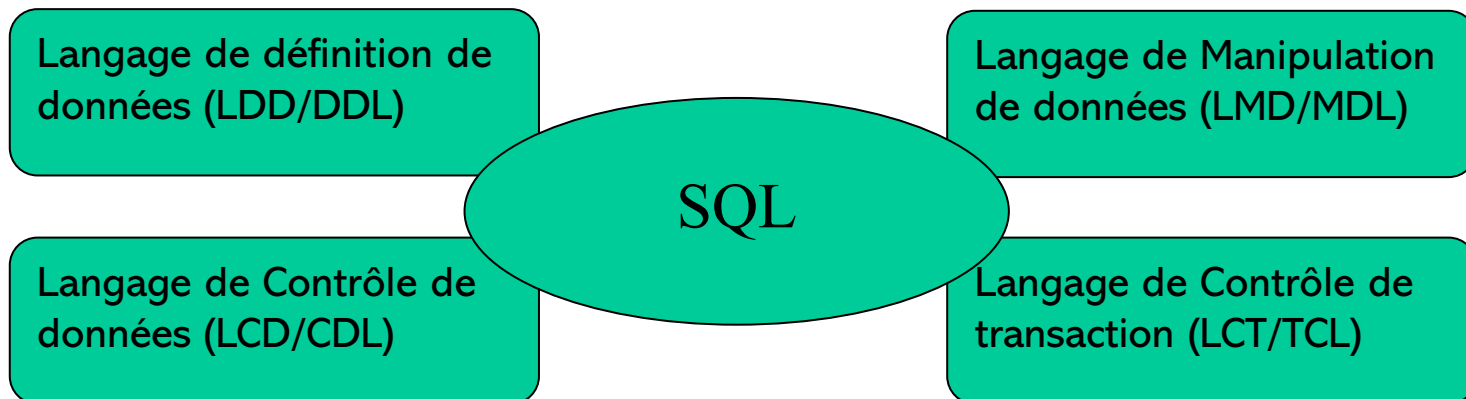
MINUS -> « résultat minoré de ».



Reprenons le SQL

- **Structured Query Langage (IBM)**

- Issu du langage QUEL, **SEQUEL** commercialisé par ORACLE
- Inclut une grande partie des possibilités de l'algèbre relationnel
- langage interprété : interface d'utilisation (MySQL, Squirrel, SQLylog, Dbeaver, MSAccess,...)
- langage intégré dans un programme (C, JAVA) : interface de programmation
- procédures exécutées dans le SGBDR Intégrés mais exécutable de l'extérieur par les interfaces d'utilisation ou de programmation.



- **Structured Query Language**

- **Les commandes SQL**

- Définition de données (**LDD**)

- CREATE

- DROP

- ALTER

- Manipulation de données (**LMD**) – (CRUD : Create, Read, Update, Delete).

- SELECT

- INSERT

- UPDATE

- DELETE

- Contrôle de transactions (**LCT**)

- Accès concurrents

- COMMIT

- ROLLBACK

- Droits d'accès

- GRANT / REVOKE

--LDD – CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX.

SQL- CREATE TABLE Utilisateurs (ID INT PRIMARY KEY, Nom VARCHAR(100),
Email VARCHAR(100));

CREATE TABLE table (

-- définition des colonnes

Nom et type de colonne [NOT NULL [UNIQUE]] [DEFAULT valeur]
[PRIMARY KEY] [REFERENCES table] [CHECK condition] ,... ,

-- contraintes de table

[PRIMARY KEY (liste de colonnes)],

[UNIQUE (liste de colonnes)] ,... ,

[FOREIGN KEY (liste de colonnes) REFERENCES table

[ON DELETE {RESTRICT | CASCADE | SET NULL}]

[ON UPDATE {RESTRICT | CASCADE | SET NULL}] ,... ,

[CHECK condition] ,...

)

Extrait de la norme SQL 2

-- Exemple de LDD--

```
CREATE TABLE vente
(
    IdCli    CHAR(4) NOT NULL          REFERENCES client      ,
    IdPro    CHAR(6) NOT NULL,
    date     DATE   NOT NULL,
    qte      SMALLINT                CHECK (qte BETWEEN 1 AND 10),
-- contrainte de table
PRIMARY KEY (IdCli, IdPro, date)      ,
FOREIGN KEY (IdPro) REFERENCES produit
--ON DELETE CASCADE ON UPDATE CASCADE --SQL2 ONLY
)
```

Autres exemples : ALTER TABLE, DROP TABLE, CREATE INDEX.

- **ALTER TABLE** permet de modifier la table

ALTER TABLE client ADD COLUMN telephone CHAR(16)

- **DROP TABLE** détruit immédiatement l'accès la table

DROP TABLE client

- **CREATE INDEX** permet d'accélérer les recherches et de créer des indexes multi-colonnes

CREATE [UNIQUE] INDEX index

ON table (colonne [ASC | DESC], ...)

l'option **UNIQUE** assure l'unicité de la clé

l'option **ASC**(endant) ou **DESC**(endant) permet de connaître l'ordre de création de l'indexe et donc de la recherche.

- **Types de données 1**

- **chaînes de caractère**

- **CHAR(n)** chaîne constante ou n est la longueur maximale
 - **VARCHAR(n)** chaîne variable ou n est la longueur maximale

- **entiers**

- **SMALLINT** sur 2 octets -32.768 à +32.767
 - **INTEGER** sur 4 octets -2.147.483.648 à +2.147.483.647

- **décimaux**

- **NUMERIC** => (n,m) le nombre de décimal doit être exactement m
 - **DECIMAL(n,m)**

- **Types de données 2**

- Numériques à virgule flottante
 - **REAL** (SIMPLE PRECISION) au moins 7 chiffres significatifs
 - **FLOAT** (DOUBLE PRECISION) au moins 15 chiffres significatifs
- Types temporels (<https://www.data-transitionnumerique.com/sql-date/>)
 - **DATE** : YYYYMMDD - jour : 2 chiffres, mois : 2 chiffres, année : 4 chiffres
 - **TIME** : HHMMSS,MMM - heures, minutes, secondes, millièmes:
 - **TIMESTAMP** : heures, minutes, secondes : ex : **1677657600**
 - Wed Mar 01 2023 09:00:00 UTC+0100
 - **INTERVAL** : intervalle de temps (INTERVAL 10 YEARS)
- Valeur NULL
 - colonne non renseignée et donc vide d'information. La valeur n'est pas zéro, c'est une absence de valeur

- **Contraintes d'intégrité**

- NOT NULL valeur null impossible
- UNIQUE unicité d'un attribut
- PRIMARY KEY clé primaire
- FOREIGN KEY clé étrangère
- CHECK plage ou liste de valeurs
- Toute opération violant une des contraintes sera rejetée
- Le système garantit l'intégrité des données
- Les contraintes d'intégrité peuvent être ajoutées ou supprimées :
 - sur une table
 - sur une colonne

Exemple

```
ALTER TABLE SALARIE  
DROP CONSTRAINT NOM_UNIQUE  
ADD CONSTRAINT SAL_MIN CHECK(SAL > 1000)  
RENAME CONSTRAINT NOM1 TO NOM2
```

--Manipulation de données (MLD) –

SELECT selection de lignes ou colonnes

```
SELECT P.prix FROM produit P WHERE P.idPro = 'p1'
```

SELECT	[DISTINCT] liste d'attributs, expressions, agrégat
FROM	liste de tables ou vues
WHERE	clause de recherche : qualifications / prédicats
GROUP BY	liste d'attributs de partion.
HAVING	qualification de groupe (agrégat)
ORDER BY	liste de colonnes [ASC DESC]

SELECT
INSERT
UPDATE
DELETE

SELECT

- Contrairement à l'algèbre relationnel, **SQL n'élimine pas les doublons** : pour cela, il faut spécifier **DISTINCT** !

SELECT DISTINCT client_id, produit FROM commandes;

Chaque combinaison unique de client_id et produit sera retournée, ce qui signifie que si un client a commandé le même produit plusieurs fois, cela n'apparaîtra qu'une seule fois dans les résultats.

- les opérations arithmétiques sont disponibles (+, -, *, /) pour les opérateurs d 'agrégats COUNT, MIN, MAX, SUM, AVG.
- L 'étoile permet de lister tous les attributs

SELECT * FROM commandes;

- Peut utiliser un tri du résultat ORDER
- Peut découper le résultat selon certains critères GROUP BY
- Peut restreinte le résultat selon certains critères HAVING

SELECT
INSERT
UPDATE
DELETE

INSERT ajout de lignes,

```
INSERT INTO client (idPro, nom, ville) VALUES  
(‘c42’,‘Duchemin’,‘Bourges’)
```

Mais

Attention

```
INSERT INTO client VALUES (‘c42’,‘Duchemin’,‘Bourges’)
```

⇒ Résultat aléatoire.

Rejouer une deuxième fois

```
INSERT INTO client (idPro, nom, ville) VALUES  
(‘c42’,‘Duchemin’,‘Bourges’)
```

Implique une erreur idPro devant être unique.

SELECT
INSERT
UPDATE
DELETE

UPDATE et DELETE

UPDATE mise à jour de lignes,

```
UPDATE TABLE produit SET P.prix= P.prix*1,21 WHERE P.idPro='p2'
```

DELETE suppression de lignes.

```
DELETE FROM produit WHERE P.idPro='p2'
```

SELECT
INSERT
UPDATE
DELETE

- La clause **WHERE** : les **conditions** de recherche
 - une condition de recherche est spécifiée par un **prédicat**.
SELECT produit FROM TABLE WHERE prix < 1,45
 - Les prédicats composés
 - composé de plusieurs prédicats simples articulés par :
 - **AND**
SELECT produit FROM TABLE WHERE prix < 1,45 AND idPro <= 100
 - **OR**
SELECT produit FROM TABLE WHERE prix >= 1,45 OR idPro > 100
 - **NOT**
SELECT produit FROM TABLE WHERE NOT (idPro=100)
 - Les **qualifications ou prédicats** simples :
 - (=, <, >, <=, >=)

- **WHERE** : condition de recherche

- Les **qualifications ou prédicats** simples :

- **LIKE** ‘%toto%’ ou LIKE ‘toto%’, ou LIKE ‘%toto’

contient commence par finit par

SELECT produit FROM TABLE WHERE produit LIKE ‘stylo%’

- **BETWEEN** : colonne entre deux valeurs

SELECT prix FROM TABLE WHERE prix BETWEEN 1.10 AND 1.90

- **IS NULL** : dont la valeur est inconnue

- **IN** : colonne dans une liste :

SELECT marque FROM TABLE WHERE marque IN (‘LENOVO’, ‘APPLE’, ‘DELL’)

- **EXIST** : au moins une valeur définie

SELECT nom FROM clients c

WHERE EXISTS (SELECT 1 FROM commandes

WHERE cmd.client_id = c.id)

- **WHERE** : condition de recherche

- Les **qualifications ou prédicats** simples :

ALL : prédicat vrai pour tous

SELECT nom, prix FROM produits

WHERE prix > **ALL** (SELECT prix_vente FROM ventes)

ANY : prédicat vrai pour au moins un

SELECT nom, salaire FROM employes

WHERE salaire > **ANY** (SELECT budget FROM departements)

- Les opérateurs **d'agrégats**

- Pas d'équivalent en algèbre relationnelle,
- Permettent d'effectuer des calculs sur des groupes
- Principaux opérateurs :

COUNT: nombre de valeurs

```
SELECT departement, COUNT(*) FROM employes GROUP BY departement
```

```
SELECT departement, COUNT(*) AS nombre_employes  
FROM employes GROUP BY departement
```

MIN : minimum des valeurs,

```
SELECT nom, prix FROM produits
```

```
WHERE prix > (SELECT MIN(prix) FROM produits);
```

MAX : maximum des valeurs,

```
SELECT nom, prix FROM produits
```

```
WHERE prix < (SELECT MAX(prix) FROM produits)
```

- Les opérateurs **d'agrégats**

- Pas d'équivalent en algèbre relationnelle,
- Permettent d'effectuer des calculs sur des groupes de n-uplets
- Principaux opérateurs :

SUM : somme des valeurs,

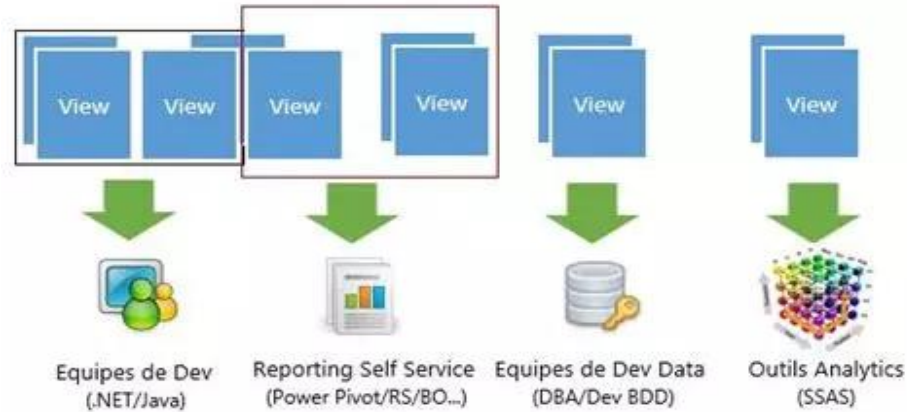
```
SELECT vendeur_id, SUM(montant) AS total_ventes FROM ventes  
GROUP BY vendeur_id  
HAVING SUM(montant) > 1000.00
```

AVG : moyenne des valeurs.

```
SELECT etudiant_id, AVG(note) AS moyenne_notes FROM notes  
GROUP BY etudiant_id HAVING AVG (note) > 15.00;
```

- Le tri du résultat d'un SELECT
 - **ORDER BY** : spécifie les colonnes qui vont définir le critère de tri dans l'ordre des colonnes
SELECT nom, prix FROM produits **ORDER BY** prix (par défaut ASC)
 - **ASC** (croissant) , **DESC** (décroissant): l'ordre de tri
ORDER BY peut être précisé par ASC ou DESC.
SELECT nom, prix FROM produits ORDER BY prix **ASC (défaut)**
SELECT nom, prix FROM produits ORDER BY prix **DESC**
 - **HAVING** Spécifie une condition de restriction de groupe nécessite la présence de l'opérateur **GROUP BY**
SELECT etudiant_id, AVG(note) AS moyenne_notes FROM notes
GROUP BY etudiant_id **HAVING** AVG (note) > 15.00

• LES VUES



*

Avantages des vues

1. Simplification des requêtes : Les vues permettent aux utilisateurs de travailler avec des ensembles de données plus simples sans se soucier des détails sous-jacents.
2. Sécurité : Les vues peuvent restreindre l'accès à certaines colonnes ou lignes d'une table. Cela empêche les utilisateurs d'accéder à des données sensibles.
3. Encapsulation de la logique : En regroupant des requêtes complexes, vous pouvez encapsuler la logique dans une vue et réduire le risque d'erreurs lors de la rédaction de requêtes.
4. Synthèse des données : Elles peuvent servir à agréger des données, par exemple pour fournir des sommes ou des moyennes.

• LES VUES

Une vue en SQL est un objet de base de données qui représente une requête stockée. Elle permet de créer une sorte de "table virtuelle" en utilisant le résultat d'une requête SELECT. Contrairement à une table physique, une vue ne stocke pas les données ; elle ne fait que sélectionner et présenter les données d'autres tables.

Une vue est une vision partielle ou particulière des données d'une ou plusieurs tables de la base.

Création **CREATE VIEW** MYVUE_R (col1, col2...) AS SELECT ... FROM R....

1/ **CREATE TABLE** employes (

id INT PRIMARY KEY,

nom VARCHAR(100),

poste VARCHAR(100),

salaire DECIMAL(10, 2),

date_embauche DATE);

2/ **CREATE VIEW** vue_employes

AS SELECT nom, poste FROM employes;

- LES VUES

- Le SELECT sur une vue **ne peut pas utiliser la clause ORDER BY.**

```
SELECT * FROM MYVUE_R
```

- Suppression de vue

```
DROP VIEW MYVUE_R
```

- **LCD : Langage de contrôle des données**
 - *Accès concurrents*
 - **COMMIT** : valider la transaction
 - **ROLLBACK** : la transaction est annulée
 - Droits d'accès
 - **GRANT** : accorder des privilèges à d'autres utilisateurs ou à tous
 - **REVOKE** : supprimer des privilèges à d'autres utilisateurs
 - pour les droits de SELECT, INSERT, UPDATE, DELETE
 - Exemple : On accorde à Morgan le droit de sélectionner et d'insérer des lignes dans la relation Etudiant. Un utilisateur ayant reçu ce privilège peut le transmettre à son tour.

GRANT SELECT ON Etudiant TO Morgan With Grant Option

ou à tous : **GRANT SELECT ON Etudiant TO PUBLIC With Grant Option**

et la suppression :

REVOKE privilège ON Etudiant FROM Tarik

- **Rappel SQL**
 - **Langage Définition de Données LDD**
 - CREATE, DROP, ALTER
 - **Langage Manipulation de données LMD**
 - SELECT, INSERT, UPDATE, DELETE
 - clause WHERE prédicat :
 - =, <>, <, >, <=, >=, LIKE %toto%, BETWEEN val1 AND val2
 - IS NULL, IN, EXIST, ALL, ANY
 - GROUP BY, ORDER BY, HAVING
 - OR, AND, NOT
 - **Langage Contrôle de Données LCD**
 - Commit, Rollback, Grant, Revoke

- **Rappel SQL**
 - **Langage Définition de Données LDD**
 - CREATE, DROP, ALTER
 - **Langage Manipulation de données LMD**
 - SELECT, INSERT, UPDATE, DELETE (CRUD)
 - clause WHERE prédicat :
=, <>, <, >, <=, >=, LIKE %toto%, BETWEEN val1 AND val2
IS NULL, IN, EXIST, ALL, ANY
GROUP BY, ORDER BY, HAVING
OR, AND, NOT
 - **Langage Contrôle de Données LCD**
 - Commit, Rollback, Grant, Revoke



Aperçu de NO-SQL

- La réponse est
- Un effet de mode
 - Le Big Data : un tournevis sonique
 - Le Big Data : une solution UNIVERSELLE
 - Sous le label NoSQL, tout est mis ...
 - Le Big Data c 'est nouveau, depuis plus de 20 ans...
- **DÉFINITION** NOSQL : Not Only Structured Query Langage
 - la fin d'un standard entre les constructeurs
- **UNE RÉALITÉ PLUS NUANCÉE**
 - un fichier est déjà une base de données NoSQL...
 - Ne replace pas SQL mais le complète
 - Une base NOSQL, pourquoi pas, mais **quel est le besoin ?**

42



- Pourquoi faire évoluer les modèles existants
 - Les critiques du modèle relationnel
 - **Manque d'évolutivité**
 - les besoins et les données évoluent mais pas le modèle.
 - L'intégration de données est devenu un secteur d'activité à part entière de l'entreprise.
 - **Manque d'efficacité** La masse des données est multiplié par deux tous les 6 mois (acteurs de type réseaux sociaux,...)
 - informatique décisionnelle pour l'analyse de grands jeux de données (banque et désormais industrie)
 - Big Data pour la gestion et l'analyse de masses de données.

- **Une réalité plus nuancée**

- Une avancée avec redis, riak, elastic search ou mongodb.... mais une abondance de projets en cours de développement
- Une perte d'universalité : chaque projet est dédié par profil d'application...fini la standardisation.
- schémas relationnels vs nosql : besoin encore plus fort de réfléchir au modèle pour optimiser les indexes : les indexes sont à la base de la performance. La réflexion de conception doit restreindre les indexes pour maximiser les performances et minimiser les volumes des indexes. Le maintien des indexes est couteux en traitement.
- Nosql et l'heure du choix: besoin encore plus fort de réfléchir au besoin pour partir sur le bon projet NoSql : orienté GIS, document, détection temps réelle,...

- Schémas des besoins dans une banque :
 - le **système opérationnel** :
 - mémorise toutes les informations sur les clients, leurs différents comptes et les opérations sur ces comptes : débit, crédit, virement, change, ...
 - déclenche des actions (envoi de relevés, ...)
 - le **système décisionnel** conserve sur une période (mensuelle, annuelle)
 - le nombre d'opérations, leur montant moyen, la moyenne des soldes sur les différents comptes, ... sur une durée historique de une ou plusieurs années.
 - Il dispose également d'informations (très, trop) personnelles sur le client mais conforme à la RGPD ...

- Schémas des besoins dans une banque :
 - **système opérationnel** vs le **système décisionnel**

Exemple Pour préparer un nouveau produit d'assurance vie, donner le nombre de client pouvant être ciblés sur des produits de niveau de risque 4 sur 9

- Chercher les revenus par tranches d 'âges par mois depuis 2014
- Donner le montant moyen du solde courant sur le compte depuis 5 ans,
- Donner le montant moyen des consommations par mois et par an.
- Comparer le type d'achats par région et le mode de souscription
- -> **ciblage client**

Constat lié au besoin : le SQL peut répondre à ce besoin, mais les requêtes sont complexes à écrire et peu lisibles. le temps de calcul peut s 'avérer élevé et nuire au système opérationnel



Le dernier cours

- **Suivre l'évolution des données**
 - **Apparition** de grandes plateformes autour du Web, moteurs de recherche, base de documents type cloud, réseaux sociaux,...
 - **Volume** considérable lié aux données du web (vidéos, photosb,..) et aux capteurs (outils connectés,-Linky,voiture,...-, rapport d'activité, surveillance réseau,...)
 - Données **évolutives** sans structure connue à l'avance.

- **Suivre l'évolution des besoins en gestion de données**
 - répartir les données pour tenir les performances malgré la masse de données.
 - Nécessité de gérer des données hétérogènes et évolutives

- NoSQL est arrivé à répondre aux règles des « 3V »
 - Volume
 - Vitesse
 - Variété
- Les 4 grandes familles de modèles NoSQL
 - Clés-Valeurs
 - Colonnes
 - Documents
 - Graphes

– **Modèle à appliquer**

• **Modèle relationnel**

- Assurer l'intégrité dans un contexte multi-utilisateurs et de système opérationnel : gérer les achats de produits, les clients, les portefeuilles.
- Modèle basé sur des tables (entités) et des jointures (relations) avec l'application des formes normales.

• **Modèle en étoile**

- permettre d'effectuer des analyses efficaces pour produire des rapports en vue de prise de décision
- Modèle basé sur la notion de fait (quantité que l'on souhaite analyser) et de dimension (axe d'analyse)
- Exemple
 - » Fait : souscription du produit à un certain prix et pour une certaine quantité.
 - » Dimension : temporelle (toujours vrai) mais aussi spatiale, client, vendeur, type de produit,....

• **Exemple de modèle en étoile**

- FAITACHAT(refjour,refclient, refvendeur, refproduit, refprix)
- JOURS (numjour, nomjour, datevaleur, ...)
- CLIENT (numclient, nom, prenom, refadresse, codepostal,nb_enfant...)
- VENDEURS (numvendeur,nom, prenom, adresse, codepostal, nb_enfant)
- PRODUIT(numproduit, appellation, durée, numcatalogue, categorie, refrisque)

NOSQL

- 4 grandes familles modèles
 - Clés-Valeurs
 - Colonnes
 - Documents
 - Graphes

– Avant toute chose, le HASH MAP

Un **hash map**, ou carte de hachage, est une structure de données qui permet de stocker des paires clé-valeur. Cette structure utilise une fonction de hachage pour transformer les clés en indices dans un tableau, ce qui permet un accès rapide aux valeurs correspondantes. Au lieu de parcourir la structure de données pour trouver une valeur associée à une clé, le hash map permet une recherche en temps constant, en moyen.

Les langages de programmation comme

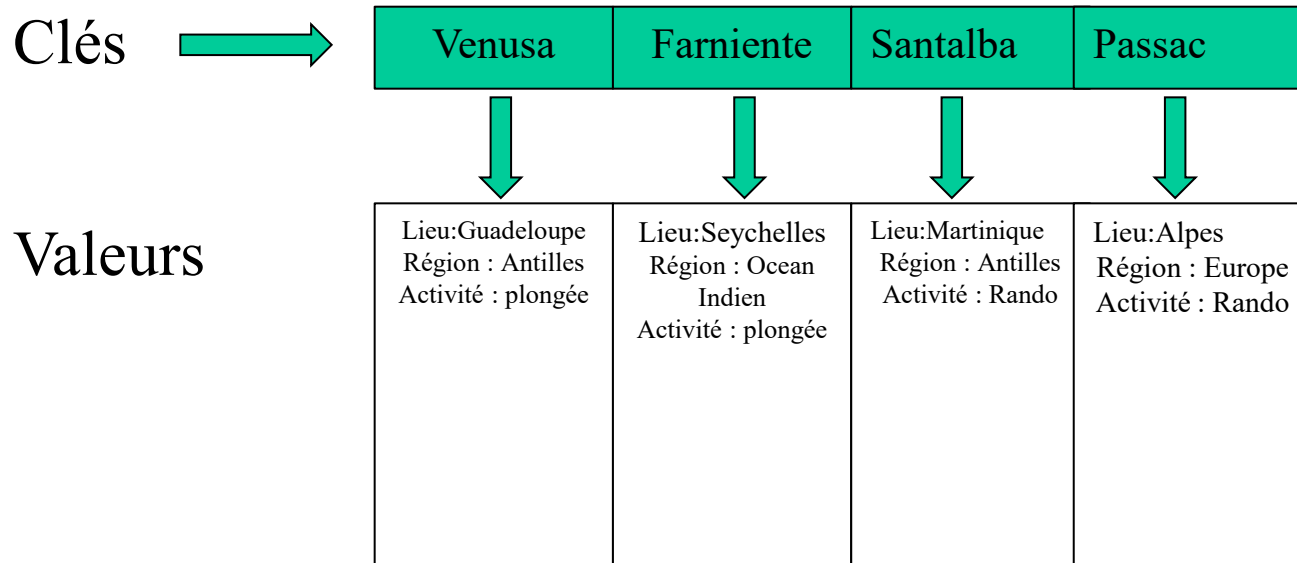
Java (avec **HashMap**),

Python (avec **dict**),

JavaScript (avec des objets)

ont des implémentations de hash maps intégrées.

- La famille de base **Clés-Valeurs** (Key/value)
 - Stockage **orienté Clé**
 - énorme HashMap - table de hachage distribuée
 - 4 Opérations possibles dites CRUD
 - » Create (clé,valeur)
 - » Read (clé)
 - » Update (clé,valeur)
 - » Delete (clé)



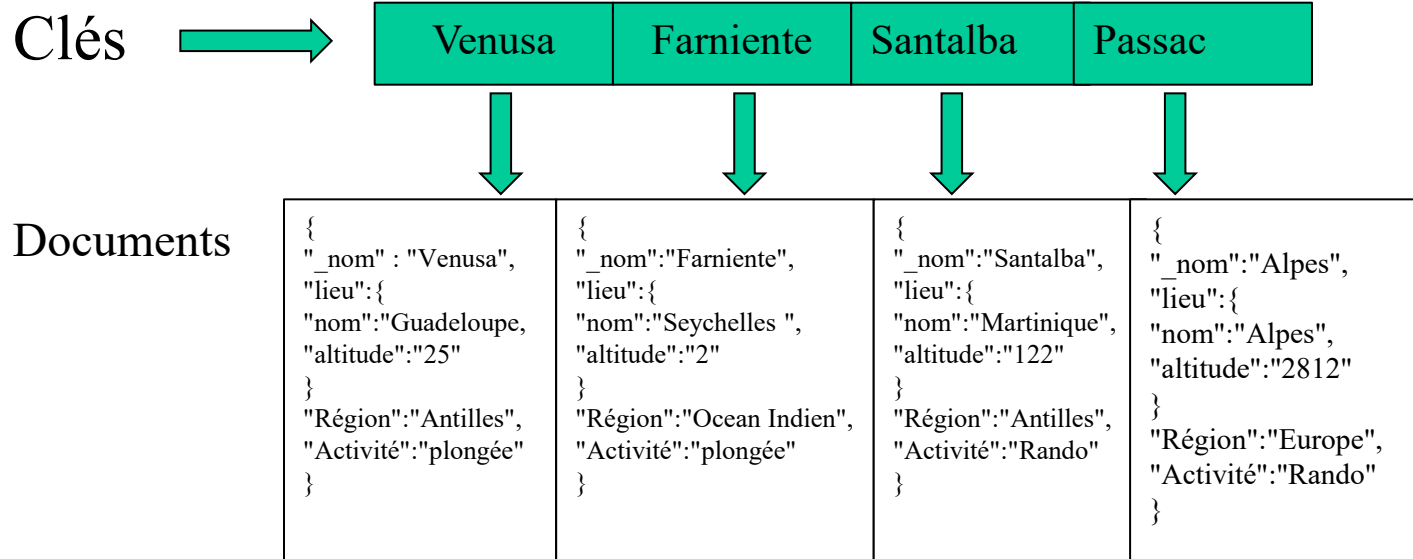
- La famille de base **Colonne** (Google Cloud BigTable, Hbase Apache ElasticSearch, SparkSQL)
 - Stockage **orienté Colonne**
 - Tables éclatées par attribut et identifiant et distribuées !!
 - Opérations d'agrégations et de corrélations entre tables
 - Exemple : Les activités en Guadeloupe (agrégation sur la clé **nom**)

Nom	Lieu
Venusa	Guadeloupe
Farniente	Seychelles
Santalba	Martinique
Passac	Alpe

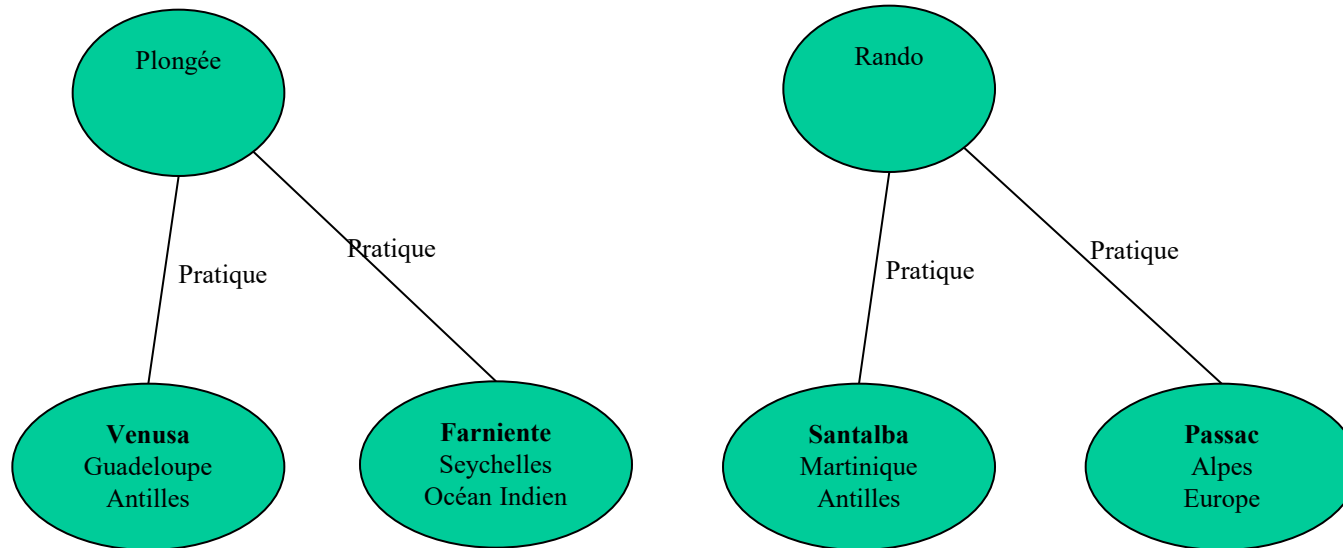
Nom	Region
Venusa	Antilles
Farniente	Océan Indien
Santalba	Antilles
Passac	Europê

Nom	Activité
Venusa	Plongée
Venusa	E-sport
Farniente	Plongée
Farniente	Char à voile
Santalba	Rando
Santalba	Hamac
Passac	Rando
Passac	Petanque

- La famille de base **Orientée Document** (MongoDB, Cassandra pour facebook, DynamoDB pour Amazon)
 - Stockage orienté clés/ sous clés dans le documents
 - Langage Riche (documents)
 - Exemple : nom des lieux (lieu.nom) situés à plus de 100 m d'altitude (lieu.altitude>100) et pratiquant l'activité rando (activité= "Rando")



- La famille de base **Orientée Graphe** (neo4j, flockdb twitter, MS Azure Cosmos DB)
 - Stockage orienté graphes
 - Réseaux, scoring d'information :



Base Relationnelle : ACID !

Atomicité : Une transaction doit être intègre : tout ou rien

Cohérence : Le contenu d'une base doit être cohérent avant et après la transaction

Isolation : Les modifications ne sont visibles qu'après validation

Durabilité : Une fois la transaction validée, l'état de la base est permanent (reprise, panne, etc...)

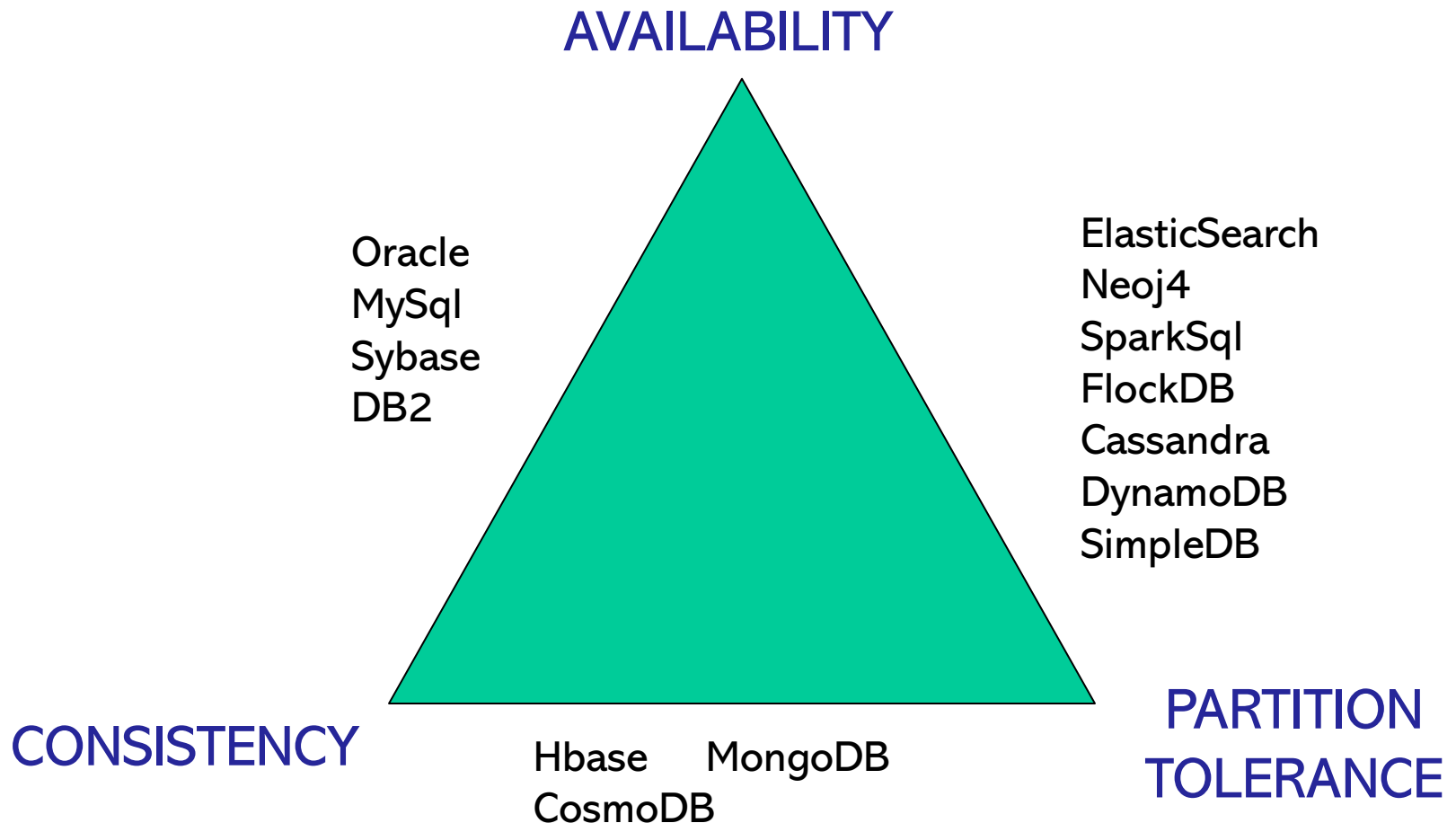
Base NoSQL

Consistency (Cohérence) : Une donnée n'a qu'un état visible quel que soit le nombre de réplicas

Availability (Disponibilité) : Tant que le système est actif, la donnée est disponible

Partition Tolerance (Distribution) : Quel que soit la distribution, toute requête fournit un résultat correct

Le triangle de Brewer ou CAP



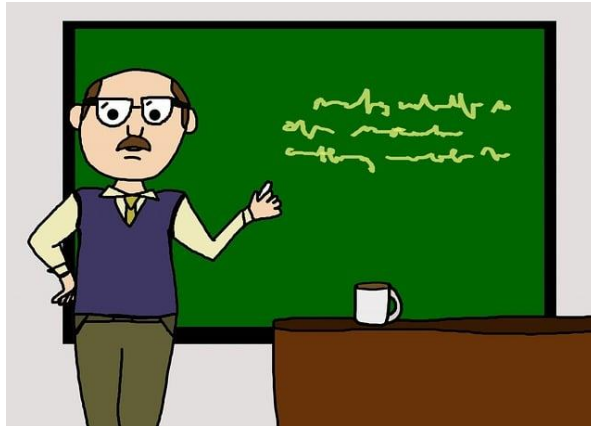
En conclusion

- Evolution des données
 - **Grandes plateformes** autour du Web, moteurs de recherche, bases cloud, réseaux sociaux,...
 - **Volume** considérable lié aux données du web (videos,medias..) et aux capteurs (outils connectés, rapport, surveillance réseau,...)
 - **Données évolutives** sans structure connue à l 'avance.
- Evolution des besoins en gestion de données
 - répartir les données pour tenir les performances malgré la masse de données.
 - Nécessité de gérer des données hétérogènes et évolutives

En conclusion

- SGBD Relationnelles traditionnelles ne sont pas adaptés en **performances, évolutivité, flexibilité pour le traitement de données évolutives à grand volume dans un contexte distribué.**
- Maturité du NoSql,
- Emergence du NewSQL depuis 2021: issu des SGBDR mais dédié au traitement transactionnel en ligne, cherche à obtenir les propriétés des systèmes NoSql tout en maintenant les propriétés ACID.

C'est fini....



Vous n'en avez plus besoin !



Merci à vous et
bonne suite.





Des compléments

Looping

Modélisation Conceptuelle de Données


Logiciel gratuit et libre d'utilisation - Université de Toulouse

Télécharger

Version 4.0 - 15/05/2022 -  zip contenant les versions 32 et 64 bits



 Contact

 Facebook

 Forum

Si vous souhaitez apporter une contribution, vous pouvez acquérir le livre ci-dessous dont les bénéfices seront intégralement reversés à ACTION CONTRE LA FAIM

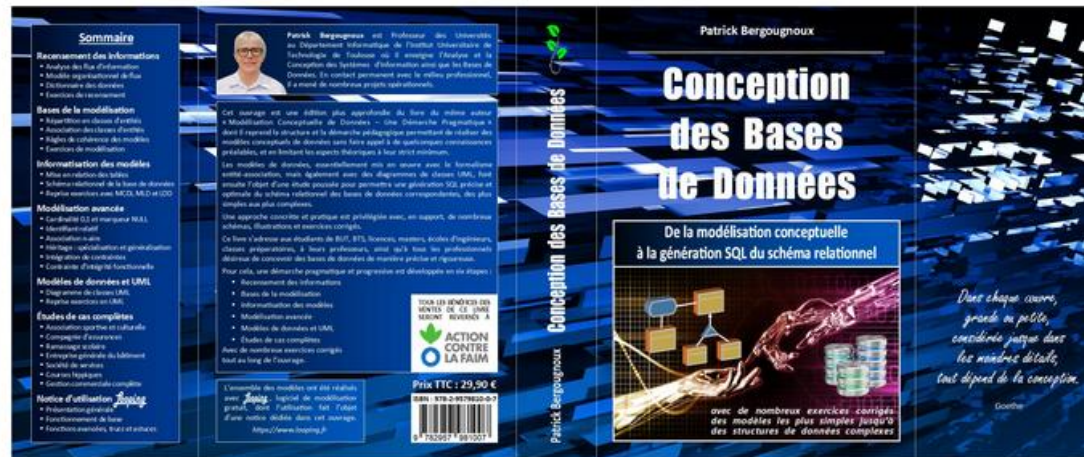


BIBLIOGRAPHIE

INSTALLATION

FONCTIONNALITES

NOUVEAUTES



Sommaire

- Recensement des informations
 - Analyse des flux d'information
 - Modèle organisationnel de flux
 - Déterminer des données
 - Exercices de recensement
- Basés de la modélisation
 - Association des données d'entrée
 - Association des données d'entrée
 - Diagramme de collaboration des modèles
 - Exercices de modélisation
- Information des modèles
 - Mise en relation des tables
 - Diagramme relationnel de la base de données
 - Diagramme relationnel de la base de données
 - Diagramme relationnel de la base de données
- Modélisation avancée
 - Coordonnées 3D et marqueur 3D
 - Modélisation 3D
 - Association 3D
 - Diagramme relationnel et généralisation
 - Diagramme relationnel de la base de données
 - Diagramme relationnel de la base de données
- Modèles de données et UML
 - Diagramme relationnel de la base de données
 - Diagramme relationnel de la base de données
- Études de cas complètes
 - Association des données d'entrée
 - Diagramme relationnel de la base de données
 - Diagramme relationnel de la base de données
 - Diagramme relationnel de la base de données
 - Diagramme relationnel de la base de données

Notice d'utilisation Looping

- Propriété intellectuelle
- Modélisation de flux
- Fonctionnalités avancées, flux et flux



Livre Papier
640 pages



Format Numérique
eBook Kindle



Encore des questions ?

Looping

Modélisation Conceptuelle de Données

Logiciel gratuit et libre d'utilisation - Université de Toulouse

Télécharger

Version 4.0 - 15/05/2022 - .zip contenant les versions 32 et 64 bits



Contact

Facebook

Forum

Si vous souhaitez apporter une contribution, vous pouvez acquérir le livre ci-dessous dont les bénéfices seront intégralement reversés à **ACTION CONTRE LA FAIM**



BIBLIOGRAPHIE INSTALLATION FONCTIONNALITES NOUVEAUTES

Conception des Bases de Données
Patrick Bergougnoux

De la modélisation conceptuelle à la génération SQL du schéma relationnel

Des choix courts, grands ou petits, considérés jusqu'aux moindres détails, tout dépend de la conception.

Prix TTC : 29,90 €

avec de nombreux exercices corrigés des modèles les plus complexes jusqu'à des structures de données complexes

Livre Papier
640 pages

Format Numérique
eBook Kindle

<https://www.looping-mcd.fr/>



- Logiciel **gratuit** et **libre d'utilisation**
- Compatible Windows 7 et supérieur, 32 et 64 bits
- Fonctionne également avec Wine sous macOS et Linux
- Installation par simple copie du fichier "Looping.exe" (moins de 8 Mo)
- Exécutable depuis tout support : disque local, serveur, clé USB, ...

Récupération d'un fichier

Looping.zip

2 versions 64 ou 32 bits