Université Internationale de Casablanca

LAUREATE INTERNATIONAL UNIVERSITIES

APPLIED INFORMATION TECHNOLOGY

SEMESTRE 3 FILIERE MANAGEMENT

Professeur : A. SAYOUTI

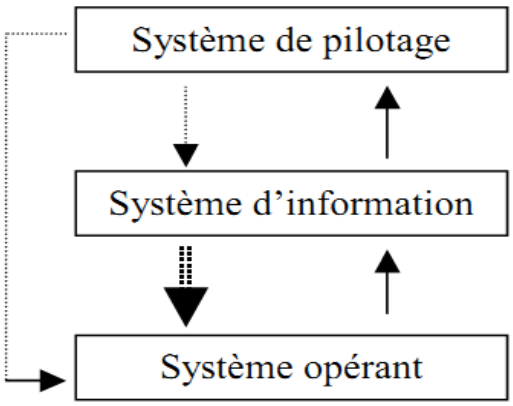
Année Universitaire : 2018 / 2019

Table des matières

1. Système d'information

L'entreprise est un système dynamique dans lequel transitent de très nombreux flux d'informations. Sans un dispositif de maîtrise de ces flux, l'entreprise peut très vite être dépassée et ne plus fonctionner avec une qualité de service satisfaisante. L'enjeu de toute entreprise qu'elle soit de négoce, industrielle ou de services consiste donc à mettre en place un système destiné à collecter, mémoriser, traiter et distribuer l'information (avec un temps de réponse suffisamment bref). Ce **système d'information** assurera le lien entre deux autres systèmes de l'entreprise : le **système opérant** et le **système de pilotage**.

- Le système de pilotage décide les actions à exécuter par le système opérant en fonction des objectifs et des politiques de l'entreprise,
- Le système opérant englobe toutes les fonctions liées à l'activité propre de l'entreprise : facturer les clients, régler les salariés, gérer les stocks…



information-représentation : information qui donne une représentation du réel ; ce réel pouvant être soit le système opérant (états discrets) soit le système d'information lui-même.

information-décision

information-intéraction : information nécessaire au système opérant pour qu'il fonctionne

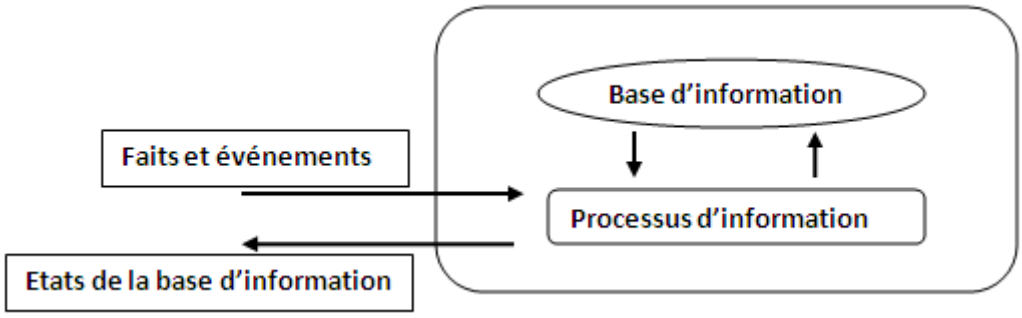
2. Conception des Systèmes d'information

La conception d'un système d'information (SI) de l'entreprise n'est pas évidente car il faut réfléchir à l'ensemble de l'organisation que l'on doit mettre en place. La phase de conception nécessite des méthodes permettant de mettre en place un modèle sur lequel on s'appuie. La modélisation consiste à créer une représentation virtuelle d'une réalité de telle façon à faire ressortir les points auxquels on s'intéresse. Ce type de méthode est appelé **analyse**. Il existe plusieurs méthodes d'analyse, la méthode la plus utilisée en France étant la méthode **MERISE** (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise). Le but de cette méthode est d'arriver à concevoir un système d'information. La méthode MERISE est basée sur la **séparation** des **données** et des **traitements** à effectuer en plusieurs modèles conceptuels, logiques et physiques (ces modèles sont abordés plus en détail dans les sections suivantes). La séparation des données et des traitements assure une longue durée de vie au modèle. La méthode MERISE date de 1978-1979, et fait suite à une consultation nationale lancée en 1977 par le ministère de l'industrie dans le but de choisir des sociétés de conseil en informatique afin de définir une méthode de conception de systèmes d'information. Les deux principales sociétés ayant mis au point cette méthode sont le CTI (Centre Technique d'Informatique) chargé de gérer le projet et le CETE (Centre d'Etudes Techniques de l'Equipement).

2.1. Architecture d'un système d'information

Le système d'information doit décrire (ou représenter) le plus fidèlement possible le fonctionnement du système opérant. Pour ce faire, il doit intégrer une base d'information dans laquelle seront mémorisés la description des objets, des règles et des contraintes du système opérant. Cette base subit des évolutions, le système

d'information doit être doté d'un mécanisme (appelé processeur d'information) destiné à piloter et à contrôler ces changements. Le schéma suivant synthétise l'architecture d'un système d'information.



Le processeur d'information produit des changements dans la base d'information à la réception d'un message. Un message contient des informations et exprime une commande décrivant l'action à entreprendre dans la base d'information. Le processeur d'information interprète la commande et effectue le changement en respectant les contraintes et les règles de gestion de l'entreprise. Si le message exprime une recherche sur le contenu de la base d'information, le processeur interprète la commande et émet un message rendant compte du contenu actuel de la base d'information. Dans tous les cas, l'environnement a besoin de connaître si la commande a été acceptée ou refusée. Le processeur émet, à cet effet, un message vers l'environnement.
Relativement à la conception d'un système d'information, l'architecture présentée ci-dessus induit une double conception :

- Celle de la base d'information ou base de données (aspect **statique**),
- Celle du processeur de traitement (aspect **dynamique**).

Pour aider le concepteur dans ces deux tâches, la méthode Merise propose un ensemble de formalismes et de règles destinées à modéliser de manière indépendante les données et les traitements du système d'information. Ces modèles ne sont qu'une base de réflexion pour le concepteur et un moyen de communication entre les divers acteurs du système d'information dans l'entreprise. Seule la validation de l'ensemble se fera en commun.

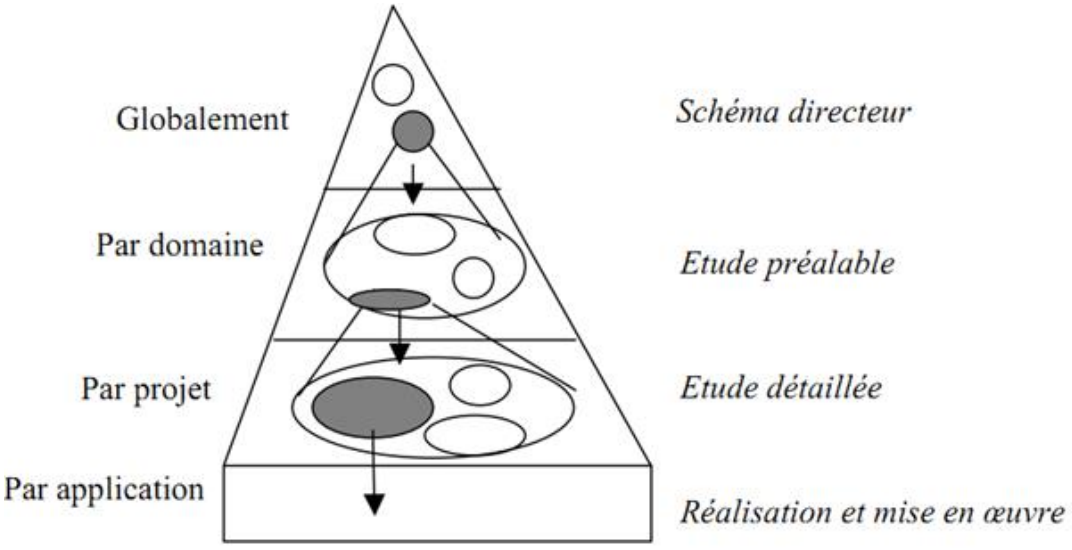
2.2. Système d'information et système informatique

Parmi les informations qui appartiennent au système d'information, certaines doivent ou peuvent faire l'objet d'un traitement **automatisé** grâce aux **outils informatiques**. Pour assurer la cohérence du système d'information, la méthode Merise propose une démarche d'informatisation comportant les étapes suivantes :

- **Le schéma directeur :** dont le rôle est de définir, de manière **globale**, la **politique d'organisation et d'automatisation du système d'information**. Pour ce faire, il est nécessaire de répertorier l'ensemble des applications informatiques existantes à modifier et à développer. Pour rendre contrôlable et modulable ce développement, il est nécessaire de **découper** le système d'information en sous-ensembles homogènes et relativement indépendant. Ces sous-ensembles sont appelés **domaines**. Par exemple, on peut trouver le domaine « comptabilité », le domaine « Personnel »... Les résultats attendus à la fin de cette étape sont une définition précise des domaines, une planification du développement de chaque domaine et un plan détaillé, année par année, des applications qui doivent être réalisées.

- **L'étude préalable par domaine :** qui doit aboutir à une présentation générale du futur système de gestion (modèles des données et des traitements) en indiquant les principales novations par rapport au système actuel, les moyens matériels à mettre en œuvre, les bilans coût – avantage. Cette étude est réalisée en quatre phases :
  - **Une phase de recueil** qui a pour objectif d'analyser l'existant afin de cerner les dysfonctionnements et les obsolescences les plus frappantes du système actuel.
  - **Une phase de conception** qui a pour objectif de formaliser et hiérarchiser les orientations nouvelles en fonction des critiques formulées sur le système actuel et d'autre part des politiques et des objectifs de la direction générale. Cela revient à modéliser le futur système avec une vue pertinente de l'ensemble.
  - **Une phase d'organisation** dont l'objectif est de définir le système futur au niveau organisationnel: qui fait quoi ?
  - **Une phase d'appréciation** dont le rôle est d'établir les coûts et les délais des solutions définies ainsi que d'organiser la mise en œuvre de la réalisation. A cet effet, un découpage en projets est effectué.
- **L'étude détaillée par projet** qui consiste d'une part à affiner les solutions conçues lors de l'étude préalable et d'autre part à rédiger, pour chaque procédure à mettre en œuvre, un dossier de spécifications détaillé décrivant les supports (maquettes d'états ou d'écran) ainsi que les algorithmes associés aux règles de gestion… A l'issue de cette étude, il est possible de définir le cahier des charges qui constitue la base de l'engagement que prend le concepteur vis à vis des utilisateurs. Le fonctionnement détaillé du futur système, du point de vue de l'utilisateur, y est entièrement spécifié.
- **La réalisation** dont l'objectif est l'obtention des programmes fonctionnant sur un jeu d'essais approuvés par les utilisateurs.
- **La mise en œuvre** qui se traduit par un changement de responsabilité : l'équipe de réalisation transfère la responsabilité du produit à l'utilisateur. Cette étape intègre en particulier la formation des utilisateurs. Après une période d'exploitation de quelques mois, la recette définitive de l'application est prononcée.
- **La maintenance** qui consiste à faire évoluer les applications en fonction des besoins des utilisateurs, de l'environnement et des progrès technologiques.

Le schéma suivant reprend les étapes décrites ci-dessus.



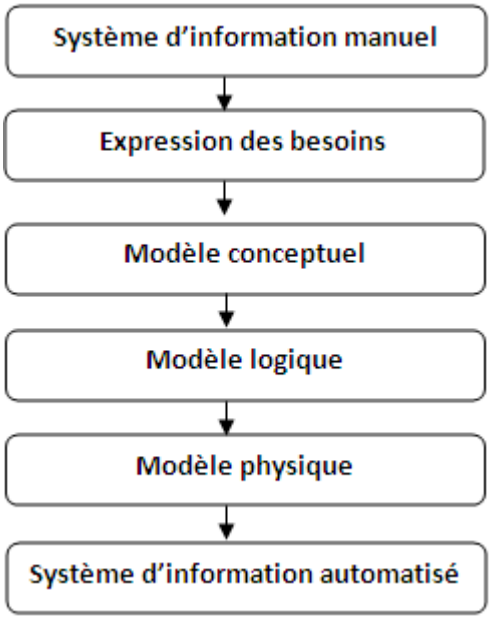
Cette démarche lourde et parfois complexe est adaptée à l'automatisation de « gros systèmes d'information ». Pour des informatisations plus modestes, elle est possible

d'**adapter** cette démarche afin de retenir uniquement les **concepts** et/ou les **étapes appropriées aux besoins**.

2.3. Cycle d'abstraction de conception des systèmes d'information

La conception du système d'information se fait par étapes, afin d'aboutir à un système d'information fonctionnel reflétant une réalité physique. Il s'agit donc de valider une à une chacune des étapes en prenant en compte les résultats de la phase précédente. D'autre part, les données étant séparées des traitements, il faut vérifier la concordance entre données et traitement afin de vérifier que toutes les données nécessaires aux traitements sont présentes et qu'il n'y a pas de données inutiles.

Cette succession d'étapes est appelée cycle d'abstraction pour la conception des systèmes d'information :



- L'expression des besoins aboutit au MCC (Modèle conceptuel de la communication) qui définit les flux d'informations à prendre en compte.
- L'étape suivante consiste à mettre au point le MCD (Modèle conceptuel des données) et le MCT (Modèle conceptuel des traitements) décrivant les règles et les contraintes à prendre en compte.
- Le modèle organisationnel consiste à définir le MLD (Modèle logique des données) qui représente un choix logiciel pour le système d'information et le MOT (Modèle organisationnel des traitements) décrivant les contraintes dues à l'environnement (organisationnel, spatial et temporel).
- Le modèle physique reflète un choix matériel pour le système d'information.

3. Système d'information de l'entreprise et TIC

Dans les entreprises, chaque ordinateur (poste de travail) est généralement dédié à un utilisateur, mais cela ne supprime pas la nécessité de partager des ressources :

- **Périphériques** : imprimantes, scanners, graveurs, …
- **Ensemble de données** : mise en commun des données, c.à.d. stockage accessible à tous, selon des règles à définir.
- **Espace disque** : la gestion des espaces disque est plus aisée si les données sont centralisées.
- **Politique de sécurité globale pour tous les utilisateurs** : si chacun fait ce qu'il veut dans son coin, il devient difficile de contrôler si tout le monde change bien son mot de passe, ferme correctement son poste, … Or, il ne s'agit pas d'un problème individuel, mais réellement d'un problème de l'entreprise, car toute information volée ou modifiée peut devenir une gêne au travail de l'entreprise.

3.1. Les contraintes à respecter

- **Rendre la communication possible entre :**
  - des machines distantes,
  - des systèmes d'exploitation différents,
  - des matériels différents.

- **Rendre transparent à l'utilisateur :**
  - les problèmes de connexion,
  - la distance entre les postes,
  - l'hétérogénéité des matériels et logiciels.

Pour cela, les postes doivent être reliés avec un média commun qui permet la communication entre les machines. Chaque machine dispose d'un point de connexion à ce média prenant en charge la communication entre les postes.

3.2. Composants d'un système informatique

Le système informatique est composé :

- Des **éléments matériels** (serveurs, postes de travail, tablette, smart phones, imprimantes, onduleur, carte réseau filaire ou sans fil …)
- Des **logiciels** (système d'exploitation, applications, logiciels, protocole de communication TCP/IP ou langage commun de communication …).
- De l'i**nfrastructure de communication** : Un support de communication, lien physique qui véhicule l'information échangée : le câblage (coaxial, paire torsadée, fibre optique), les équipements électroniques du réseau, tels que switch, routeur, point d'accès…

3.3. Principales parties matérielles

Les éléments matériels qui composent le système informatique sont appelé le **hardware**. Il est l'interface directe avec l'utilisateur. Cela concerne :

- Poste de travail, serveur, PDA, smartphone, tablette, imprimante, scanner, disque dur, onduleur, périphériques externes…
- Carte mère, processeur, carte d'extension…

3.4. Les principaux types de logiciels

Dans la partie software, nous avons :

- Les **logiciels** : ou **applications** permettent le traitement de texte, la gestion de bases de données, gestion de projets, gestion d'un cabinet médicale ou juridique, les tableurs…
- Les **programmes** : représentent les services de base fournit aux utilisateurs : interface graphique, gestionnaire de tâches…

- Le **système d'exploitation** : est une partie logicielle du système informatique. Plusieurs systèmes d'exploitation peuvent être utilisés sur le même ordinateur. Il est l'intermédiaire obligé entre l'utilisateur et le matériel. Il permet de gérer l'utilisation de la totalité des ressources : temps, mémoire, fichiers, communications, etc.

Il existe trois principaux types de logiciels, dont les logiciels libres, les logiciels payants et les logiciels gratuits. Chaque logiciel exige un système d'exploitation pour être utilisé.

Les logiciels payants ou logiciels propriétaires

Ces logiciels sont créés par des sociétés privées et sont vendus aux utilisateurs dans les magasins spécialisés ou sur Internet ou lors de l'achat d'un PC. Toutefois, il faut noter que l'achat d'un logiciel payant accorde seulement le droit d'usage à l'utilisateur et non le droit d'exploitation. Il n'est donc pas possible pour un utilisateur de revendre un logiciel propriétaire se trouvant par défaut dans son ordinateur lorsqu'il l'a acheté. Il n'a pas non plus le droit de transformer le logiciel pour ses besoins, et cela, quelle que soit son aptitude en informatique. Seul le cartel qui détient la licence de propriété a ce pouvoir. Le système d'exploitation « Windows » est par exemple un des logiciels payants.

Les logiciels gratuits

Les logiciels gratuits sont créés par des firmes spécialisées. Ils sont donnés gratuitement aux utilisateurs qui en ont besoin. Ces types de logiciels sont téléchargeables gratuitement sur internet et on peut aussi les trouver dans les magasins ou les kiosques de journaux sous forme de CD-Rom. Les utilisateurs ont le droit de les copier et de les faire circuler. Cependant, leur code source n'est pas transformable, ce qui fait que le logiciel ne peut être exploité que si son concepteur le souhaite. À noter que ce type de logiciel est généralement plus performant puisque les associations qui les produisent ont pour but de se faire connaître auprès des utilisateurs.

Les logiciels libres

Comme leur nom l'indique, les logiciels libres peuvent être utilisés et exploités par les utilisateurs. C'est-à-dire que leur code source est ouvert à tous et peut-être changé en toute légalité. Ces logiciels sont souvent gratuits et peuvent être téléchargés sur Internet. Il est donc possible d'en réaliser des copies.

3.5. Processus de développement d'un logiciel

La qualité du processus de réalisation est garante de la qualité du produit. Pour obtenir un logiciel de qualité, il faut en maîtriser le processus d'élaboration :

- La vie d'un logiciel est composée de différentes étapes,
- La succession de ces étapes forme le **cycle de vie** du logiciel,
- Il faut contrôler la succession de ces différentes étapes.

Les différentes étapes de réalisation d'un logiciel sont les suivantes :

- Etude de faisabilité ;
- Spécification ;
- Organisation du projet ;
- Conception ;
- Implémentation ;
- Tests ;
- Livraison ;
- Maintenance.

3.6. Infrastructure de communication

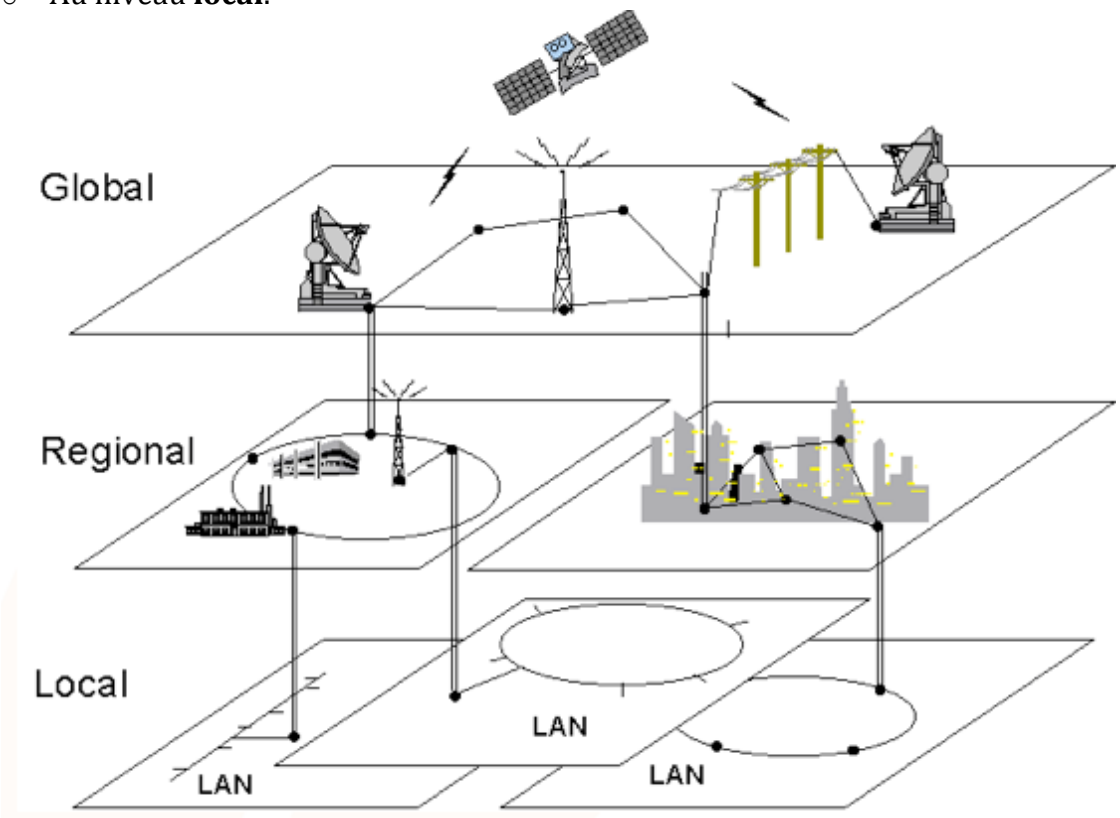
L'infrastructure de communication représente l'intégralité du réseau de communication créé par l'organisation.

Ce réseau permet l'interconnexion des utilisateurs entre eux afin de :

- **Faciliter les échanges ;**
- Permettre un **partage de l'information constant**, par tous et depuis n'importe quel matériel ;
- **Protéger les données** recueillies par l'organisation.

Ce réseau est ensuite découpé par zones géographiques selon l'étendue de sa portée :

- Au niveau **national**.
- Au niveau **régional**.
- Au niveau **local**.



3.7. Les TIC au profit des Systèmes d'Information

Les Technologies de l'Information et de la Communication (TIC) regroupent :

- L'informatique (ordinateurs, notebook, etc…),
- L'audiovisuel (téléviseurs),
- Le multimédia (sons, image, vidéos),
- L'Internet,
- Les télécommunications (téléphonie, smartphones).

Les TIC, Technologies de l'Information et de la Communication, rendent le Système d'Information plus performant aussi bien pour les informations opérationnelles que les informations décisionnelles.

- Informations opérationnelles :

Amélioration des opérations de collecte (scan, capteur, numérisation, …), de traitement (amélioration des applications, …) et de diffusion (partage en réseau),

- Informations décisionnelles :

Amélioration des opérations de stockage (performance, coût, fiabilité, …) et de traitement des informations opérationnelles (amélioration des applications).

Les TIC permettent de partager l'information, réduire les délais, traiter l'information en temps réel, etc.

3.8. Contribution du SI à la performance de l'entreprise

Le système informatique et plus globalement les TIC se développent de jours en jours, offrant toujours plus de possibilités pour les organisations de collaborer en ligne.
Le matériel utilisé conditionne les activités quotidiennes de chaque utilisateur. Ainsi, le système informatique contribue fortement au déroulement des activités des différents processus de l'organisation.

Exemple : Collaboration en ligne

Une organisation qui dispose d'un ordinateur doté d'une application d'analyse de données ne disposera pas du même processus que celle qui ne dispose pas de ce logiciel et qui par conséquent devra découper cette activité en deux : le traitement des données (par l'utilisateur 1) et l'analyse (par l'utilisateur 2).

On peut voir dans cet exemple que le matériel conditionne les processus de l'organisation.
Les TIC (Technologies de l'Information et de la Communication) sont une source de **gains de productivité** :

- Pour les **activités du processus** : traitements automatisé, stockage de données…
- Pour le **processus** lui-même : interface et coordination entre les activités (partage et échange d'information en réseau).

De plus, les TIC aident à générer des informations de contrôle par le biais d'indicateurs d'activité.
Les TIC ont permis de réduire le délai d'information (du client et peut-être de réalisation du processus) et le coût du processus (limitation de l'intervention humaine dans certaines activités ou processus).
Les TIC, en particulier les bases de données partagées en réseau, permettent de cumuler les avantages de :

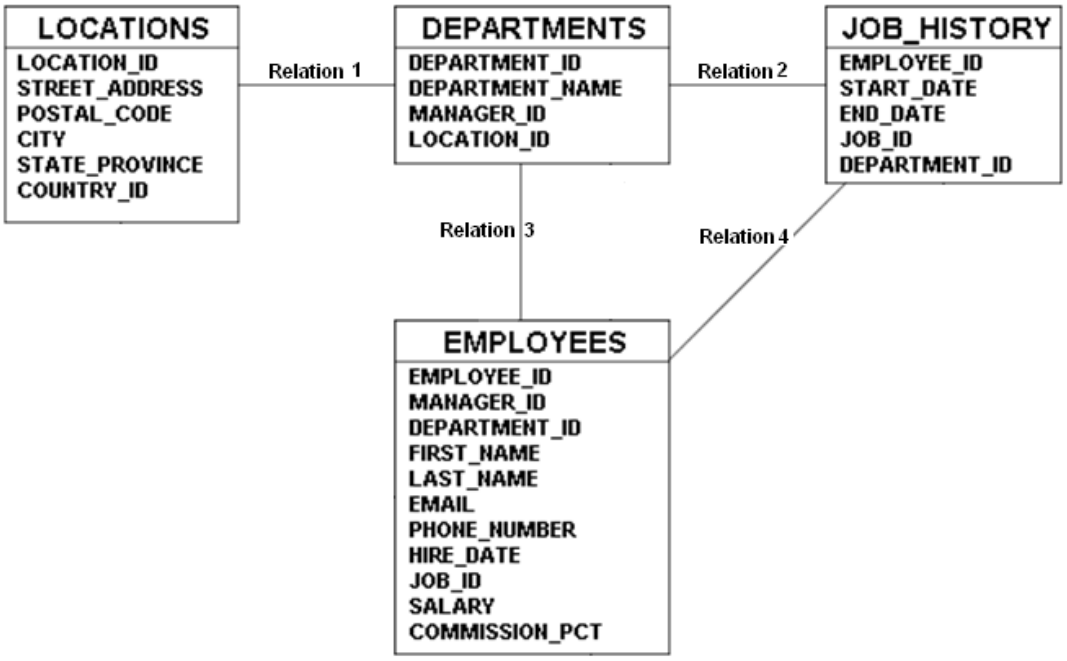
- La **centralisation** : la mutualisation des données, l'absence de redondance et la mise à jour unique créent des économies d'échelles.
- L'**accès décentralisé** : la mise à jour en temps réel, une disponibilité de l'information permanente, totale (sous réserve des droits d'accès) et en tout lieu (en cas d'architecture client-serveur utilisant les technologies d'internet).

Le partage de l'information peut dépasser les frontières de l'organisation pour s'étendre aux partenaires extérieurs (via les extranets).

4. Base de données

Une **base de données** est un lot d'informations stockées dans un dispositif informatique. Les technologies existantes permettent d'organiser et de structurer la base de données de manière à pouvoir facilement manipuler le contenu et stocker efficacement de très grandes quantités d'informations.
Une base de données est un **ensemble d'informations structurées en tables** dont l'implantation, la mise à jour et l'exploitation (l'ajout, la modification et la recherche de données) sont réalisées à l'aide d'un logiciel appelé **Système de gestion de bases de données relationnel (SGBDR)**. Une **table** est un ensemble de données organisées sous forme d'un **tableau** où les **colonnes** correspondent à des **catégories d'information** (une colonne peut stocker des noms, une autre des téléphones...) et les **lignes** à des **enregistrements**.

Une base de données est définie par son **schéma** (la structure) et son **contenu** (les valeurs). Une base de données relationnelle utilise **des tables avec des relations** pour le stockage des informations. Par exemple, vous pouvez stocker des informations concernant tous les employés d'une entreprise. Pour cela, vous créez plusieurs tables afin de stocker différentes informations sur les employés, telles qu'une table employés, une table départements et une table locations.



5. Systèmes de gestion de bases de données relationnelles : SGBDR

Un Système de Gestion de base de données Relationnelles SGBDR (MySQL, Access, SQL Server, Oracle, etc.) héberge généralement plusieurs bases de données. Actuellement, la plupart des SGBDR fonctionnent selon un mode client/serveur. Le serveur (la machine qui stocke les données) reçoit des requêtes SQL (langage de requêtes structurées) de plusieurs clients et ceci de manière concurrente. Le serveur analyse la requête, la traite et retourne le résultat au client.

6. Conception de bases de données

Quand nous commençons directement par la création des tables d'une base de données dans un SGBDR, nous sommes exposés à deux types de problèmes :

- Nous ne savons pas toujours dans quelle table placer certaines colonnes (par exemple, le nom de département, auquel l'employé est actuellement affecté, se met dans la table des employés ou dans la table des départements) ;
- Nous avons du mal à prévoir les tables de jonction intermédiaires (par exemple, la table des interprétations qui est indispensable entre la table des films et la table des acteurs).

Il est donc nécessaire de recourir à une étape préliminaire **d'analyse, de conception et de modélisation**. Dans le paragraphe suivant, nous présentons la méthode **Merise** (Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise) qui permet notamment de concevoir un système d'information d'une façon standardisée et méthodique.

La méthode Merise préconise trois niveaux d'abstraction :

- Le **niveau conceptuel** qui décrit **la statique et la dynamique** du système d'information en se préoccupant uniquement du point de vue du gestionnaire.
- Le **niveau organisationnel** décrit **la nature des ressources** qui sont utilisées pour supporter la description statique et dynamique du système d'information. Ces ressources peuvent être humaines et/ou matérielles et logicielles.
- Le **niveau opérationnel** dans lequel on choisit les **techniques d'implantation** du système d'information (données et traitements)

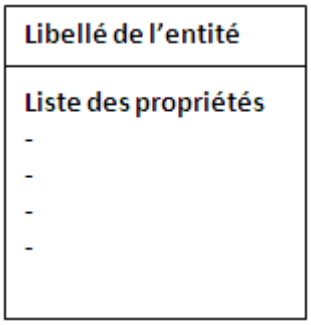
Niveau	*Statique (données)*	*Dynamique (traitements)*	
Conceptuel	**MCD**	**MCT**	Indépendant du système: *QUOI ?*
Organisationnel ou logique	**MLD** *(OU ?)*	**MOT** *(QUI ? QUAND ?)*	Choix du SGBD: *QUI ? QUAND ? OU ?*
Opérationnel ou physique	**MPD**	**MOPT**	Haute connaissance du SGBD: *COMMENT ?*

7. La modélisation entité relation

Le modèle conceptuel des données (**MCD**) est une représentation statique du système d'information de l'entreprise qui met en évidence sa sémantique. Il a pour but d'écrire de façon formelle les données qui seront utilisées par le système d'information. Il s'agit donc d'une représentation des données, facilement compréhensible. Cet aspect recouvre les entités qui décrivent le système ainsi que les liens existants entre ces entités. Le formalisme adopté par la méthode Merise pour réaliser cette description est basé sur les concepts « **entité-association** ». Dans ce qui suit, nous présentons quelques concepts de base.

7.1. Entité

Une entité est la représentation d'un élément **matériel** ou **immatériel** ayant un rôle dans le système que l'on désire décrire. On appelle « **classe d'entité »** un ensemble composé d'entités de même type, c'est-à-dire dont la définition est la même. Le classement des entités au sein d'une classe s'appelle classification (ou abstraction). Une entité est une instanciation de la classe. Chaque **entité** est composée de **propriétés**, c.à.d. données élémentaires permettant de la décrire. Prenons par exemple une Ford fiesta, une Renault Megane et une Peugeot 207. Il s'agit de trois entités faisant partie d'une classe d'entité que l'on pourrait appeler **voiture**. La Ford Fiesta est donc une instanciation de la classe voiture. Chaque entité peut posséder les propriétés : modèle, année de fabrication, puissance du moteur et couleur…



Les classes d'entités sont représentées par un rectangle. Ce rectangle est séparé en deux champs :

- Le champ du haut contient le **libellé de l'entité**. Ce libellé est généralement une abréviation pour une raison de simplification de l'écriture. Il s'agit par contre de vérifier qu'à chaque classe d'entité correspond à un et un seul libellé, et réciproquement.
- Le champ du bas contient la liste des **propriétés** de la classe d'entité.

7.2. Propriété

La propriété (ou attribut) est une information élémentaire, c.à.d. non déductible d'autres informations, qui présente un intérêt pour le domaine étudié. Par exemple, si l'on considère le domaine de gestion des commandes d'une société de vente, les données : « référence article », «désignation article », « prix unitaire HT », « taux de TVA » sont des propriétés pertinentes pour ce domaine. La donnée « prix unitaire TTC » n'est, d'après la définition, pas une propriété car ses valeurs peuvent être retrouvées à partir des propriétés «prix unitaire HT » et « taux de TVA ».

Chaque valeur prise par une propriété est appelée **occurrence**. Des occurrences de la rubrique «désignation article» sont par exemple : «T-shirt», « pantalon », « chaussures » …

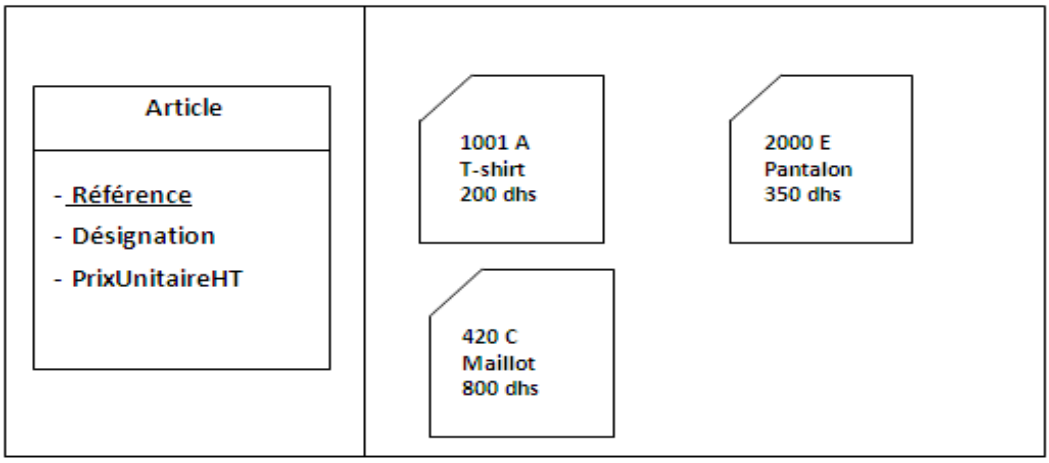
7.3. Identifiant

Un identifiant est une propriété ou un ensemble de propriétés permettant de désigner une et une seule entité. La définition originale est la suivante : L'identifiant est une propriété particulière d'un objet telle qu'il n'existe pas deux occurrences de cet objet pour lesquelles cette propriété pourrait prendre une même valeur. Le modèle conceptuel des données propose de souligner l'identifiant (ou les identifiants).

Exemple : La classe d'entité élève

7.4. Occurrence d'entité

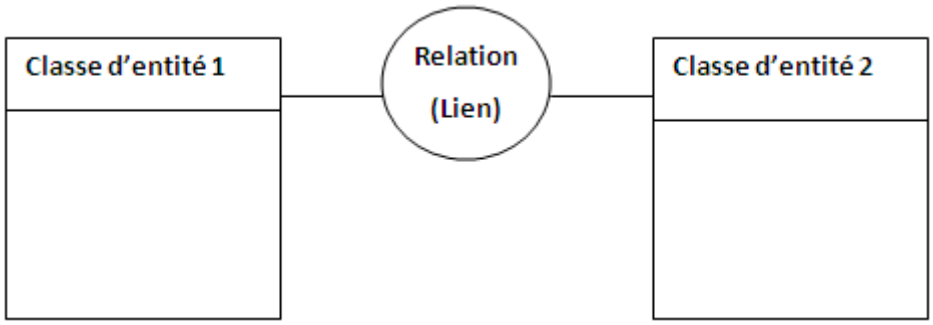
D'après la définition d'une entité, on sait que la valeur de l'identifiant détermine les valeurs des autres propriétés de l'entité. L'ensemble de ces valeurs est appelé occurrence d'entité ou individu. Le tableau suivant présente des exemples d'occurrences de l'entité ARTICLE.



7.5. Relation

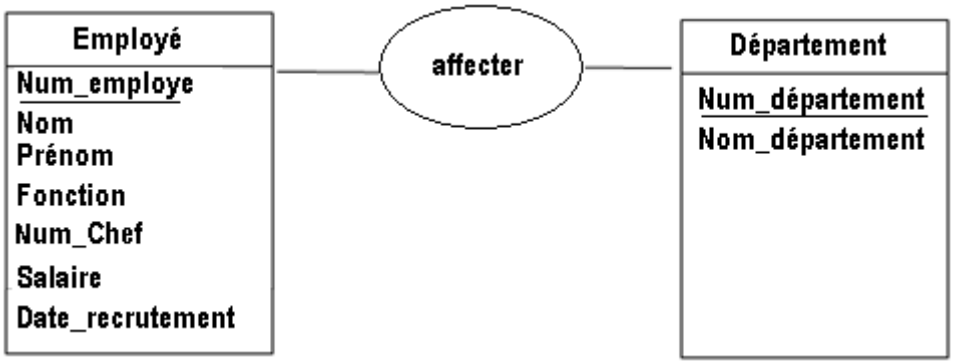
Une relation (appelée aussi **association**) est un lien sémantique entre plusieurs entités. Une classe de relation contient toutes les relations de même type (qui relient des entités appartenant à des mêmes classes d'entité).
Une classe de relation peut lier plus de deux classes d'entité. Voici les dénominations des classes de relation selon le nombre d'intervenants:

- Une classe de relation **binaire** relie deux classes d'entité.
- Une classe de relation **ternaire** relie trois classes d'entité.
- Une classe de relation **récursive** (ou **réflexive**) relie une classe d'entité avec elle même.

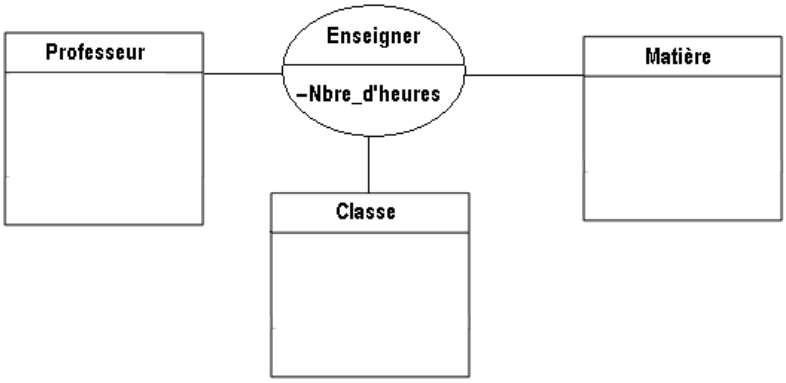


Exemples :

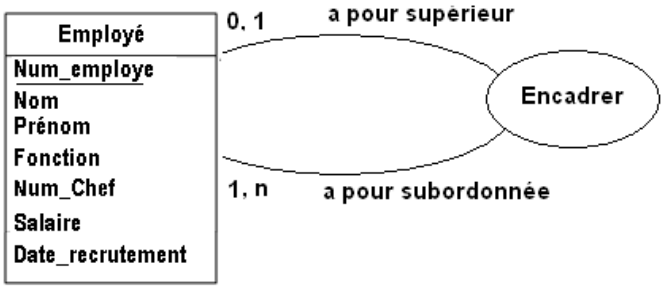
- ✓ **Relation binaire** : employé est affecté à un département.



- ✓ **Relation ternaire** : un professeur enseigne une matière pour une classe.



- ✓ **Relation réflexive** : chaque employé a un seul chef (supérieur hiérarchique direct) sauf le patron. Chaque chef a un ou plusieurs employés sous ses ordres.

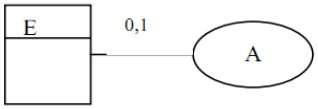
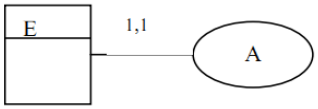
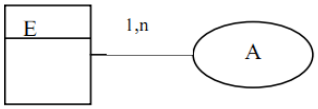
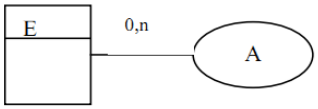


7.6. Cardinalité

Les cardinalités permettent de caractériser le lien qui existe entre une entité et la relation à laquelle elle est reliée. La cardinalité d'une relation est composé d'un couple comportant une borne minimale et une borne maximale, intervalle dans lequel la cardinalité d'une entité peut prendre sa valeur :

- La borne **minimale** (généralement 0 ou 1) décrit le nombre minimum de fois qu'une entité peut participer à une relation.
- La borne **maximale** (généralement 1 ou n) décrit le nombre maximum de fois qu'une entité peut participer à une relation.

Un couple de cardinalités placé entre une entité E et une association A représente le nombre minimal et maximal d'occurrences de l'association A qui peuvent être « attachées » à une occurrence de l'association E. Le tableau ci-après récapitule les valeurs que peut prendre ce couple.

	Pour chaque occurrence de E, le modèle admet : - soit l'absence de lien - soit la présence d'un seul lien
	Pour chaque occurrence de E, le modèle admet la présence d'un et un seul lien
	Pour chaque occurrence de E, le modèle admet la présence d'un seul ou de plusieurs liens
	Pour chaque occurrence de E, le modèle admet : - soit l'absence de lien - soit la présence de plusieurs liens

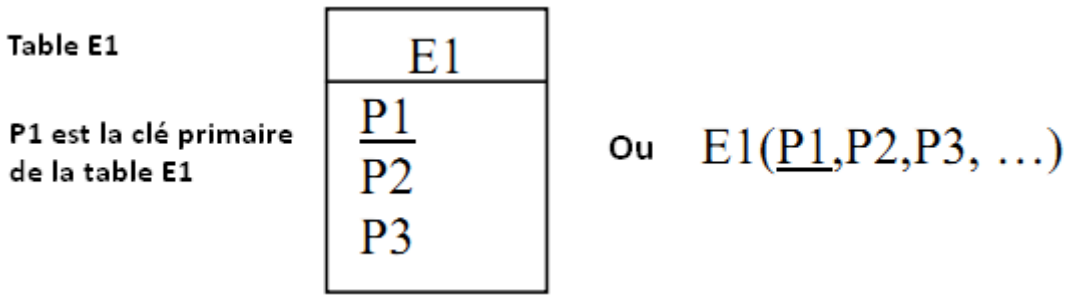
8. Le modèle relationnel

La description conceptuelle nous a permis de représenter le plus fidèlement possible les réalités de l'entreprise à informatiser. Mais cette représentation ne peut pas être directement manipulée et acceptée par un système informatique. Il est donc nécessaire de passer du niveau conceptuel à un autre niveau plus proche des capacités des systèmes informatiques qui est le **niveau logique**. Le modèle relationnel (modèle logique de données - MLD) repose sur des techniques d'organisation des données que des SGBD relationnels (tels qu'Access, SQL Server ou ORACLE) sont capables de gérer. Dans la section suivante, nous présentons les règles de passage du MCD au modèle logique des données (**MLD**).

8.1. Règles de passage du MCD au MLD

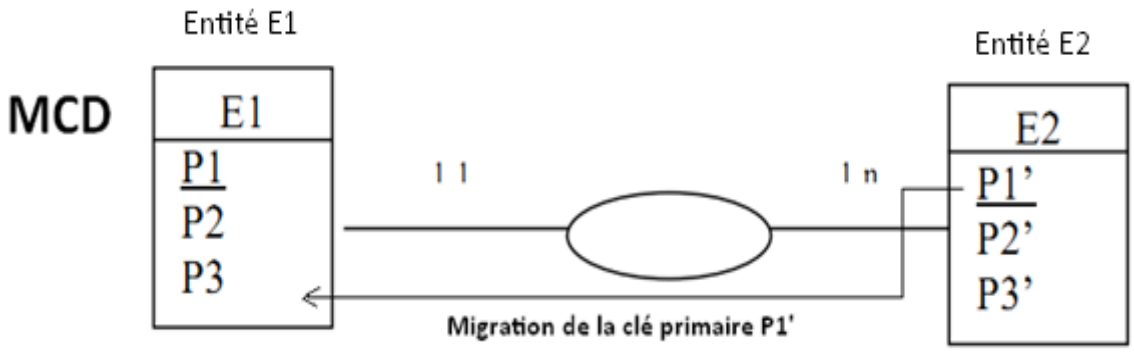
Ces règles sont de type algorithmique et peuvent donc être mises en œuvre par un logiciel tel que PowerAMC. La traduction des concepts de base du modèle conceptuel est régie par les trois règles suivantes :

- **Toute entité devient une table.** L'identifiant de l'entité devient clé primaire de la table.



Chaque classe d'**entité** du modèle conceptuel devient **une table dans le modèle logique**. Les **identifiants** de la classe d'entité sont appelé **clés de la table**, tandis que les attributs standards deviennent des attributs de la table, c'est-à-dire des colonnes.

- **Toute relation binaire de type (1-1) ou (1-n)** est caractérisée par l'existence d'une dépendance fonctionnelle entre l'identifiant de l'entité reliée par le segment portant la cardinalité 1,1 ou 0,1 et l'autre entité. Dans le schéma ci-dessous on a la dépendance fonctionnelle suivante : P1-> P1'.



Une telle relation (dépendance fonctionnelle) entraîne la migration de l'identifiant de l'entité but (E2 dans ce cas) vers l'entité source (E1 dans ce cas). La clé primaire migrée devient clé étrangère (propriété) dans l'entité source (voir figure ci-dessous). La clé étrangère P1' de la table E1 est une clé primaire de la table E2.

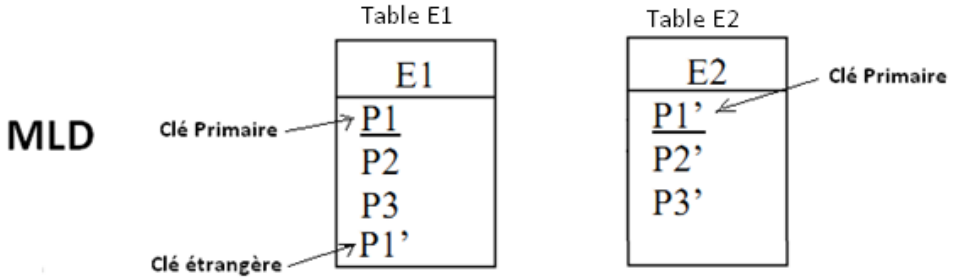
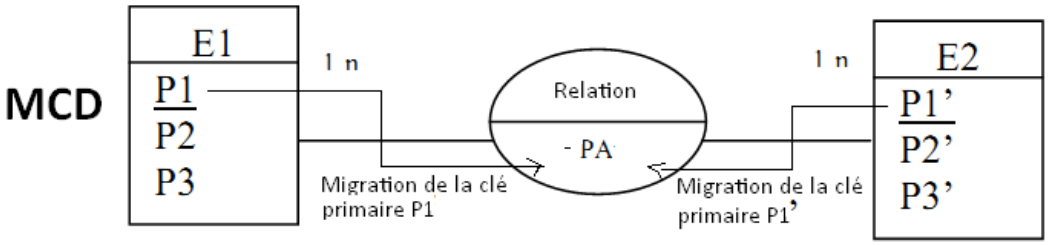


Table E1(P1,P2,P3,…P1')
Table E2(P1',P2',P3', …)

Toute relation de type (m-n) devient une Table qui hérite les identifiants des entités participants à la relation. Si l'association est porteuse, la relation sera complétée par la liste des propriétés portées.



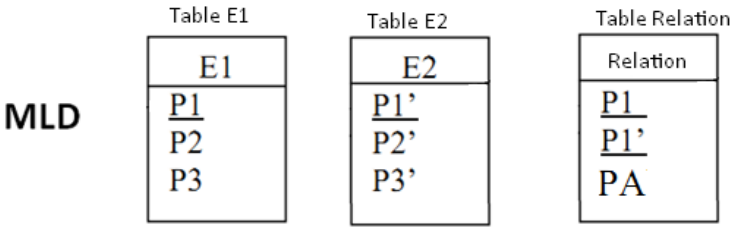
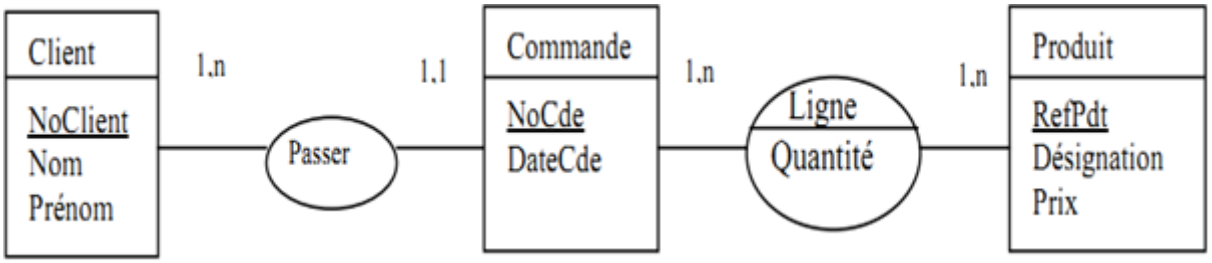


Table E1(P1,P2,P3, …)
Table E2(P1',P2',P3', …)
Table Relation (P1, P1',PA)

La clé de la table nommée «Relation» est composée du couple (P1, P1'). P1 et P1' sont deux clés primaires respectivement dans les tables E1 et E2.
L'**exemple** ci-dessous illustre l'application de ces trois règles sur le modèle conceptuel classique de **gestion des commandes** suivant :



Le Modèle logique textuel :

Client(NoClient, Nom, Prénom)
Produit(RefPdt, Désignation, Prix)
Commande (NoCde, DateCde, NoClient)
Ligne(NoCde, RefPdt, Qté)

La base de données créée dans le SGBDR est composée de quatre tables. Les tables client, produit et ligne contiennent chacune trois enregistrements. Dans la table commande sont stockés deux enregistrements.

Table *Client*

NoClient	Nom	Prénom
1	Lassus	Annick
2	Mundubeltz	Armelle
3	Chaulet	Bernadette

Table *Commande*

NoCde	DateCde	NoClient
100	14/04/2001	2
101	14/04/2001	1

Table *Produit*

RefPdt	Désignation	Prix (dh)
VE45	Vélo	1500
VE32	Kit 2 roues arrières	300
VE21	Kit éclairage	150

Table *Ligne*

NoCde	RefPdt	Qté
100	VE45	1
100	VE32	1
101	VE21	2

8.2. Base de données relationnelle

Une base de données relationnelle est un ensemble de relations ou de tables à deux dimensions. Les principes du modèle relationnel ont été décrits pour la première fois par Dr. E. F. Codd dans un article de juin 1970 intitulé « A Relational Model of Data for Large Shared Data Banks ». Les modèles courants utilisés à cette époque étaient le modèle hiérarchique : les structures de données à base de simples fichiers plats.

Dans les paragraphes suivants, nous nous intéressons à ACCESS comme système de gestion de bases de données relationnelles.

9. Système de Gestion de Bases de Données SGBD - Microsoft Access

9.1. Introduction à Access

Microsoft Access (2010, 2013, 2016 ou 2017 ''dernière version'') est un logiciel, qui fait partie de la suite Microsoft office, de déploiement et de conception d'application de base de données que vous pouvez utiliser pour effectuer le suivi d'informations importantes.

Vous pouvez conserver vos données sur ordinateur ou vous pouvez les publier sur le Web afin que d'autres puissent utiliser votre base de données avec un simple navigateur Web.

9.2. Excel Vs. Access

Excel est un tableur : il ne vous propose pas d'autre présentation que des tableaux, en lignes et en colonnes. Access vous propose davantage de possibilités et vous permet de personnaliser votre base de données pour en faire une **application sur mesure**. Avec Access, vous avez également la possibilité de **relier plusieurs tables de données** : vous pouvez ainsi mettre en relation le fichier client, les commandes, la facturation et la mise à jour des stocks pour effectuer une gestion commerciale complète. De plus, Access permet de gérer les données avec beaucoup plus de **sécurité** et de **facilité**.

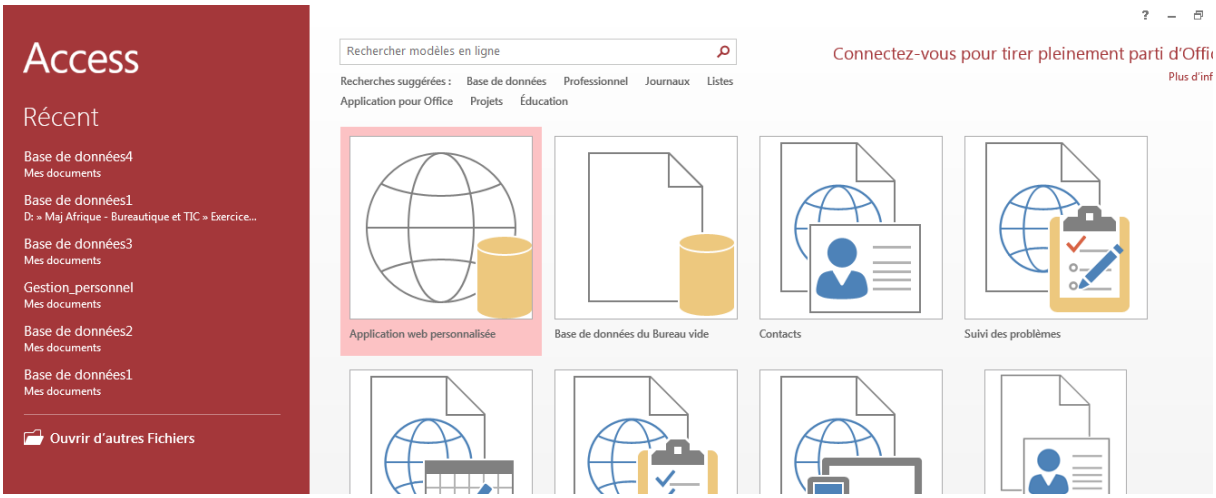
9.3. Bases de données relationnelles dans Access

Il est parfois nécessaire d'utiliser une base de données relationnelle pour effectuer le suivi de ce type d'informations : un entrepôt de stockage de données réparties en plusieurs collections de données plus petites (les tables) pour éliminer les redondances et reliées entre elles par les informations qu'elles ont en commun (les champs). Par exemple, une base de données relationnelle dédiée à la planification d'événements peut inclure une table contenant les informations sur les clients, une autre contenant les informations sur les fournisseurs et une autre avec les informations sur les événements. Cette dernière table peut contenir un champ la reliant à la table des clients et un autre la reliant à la table des fournisseurs. De cette façon, par exemple, lorsqu'un fournisseur change de numéro de téléphone, l'information est mise à jour dans la table des fournisseurs uniquement, et il n'est pas nécessaire de la modifier dans tous les événements impliquant le fournisseur.
Access est un outil qui permet de développer rapidement et facilement des applications de base de données relationnelle afin de gérer vos informations. Vous pouvez créer une base de données pour effectuer le suivi de n'importe quel type d'informations, par exemple, un inventaire, des contacts professionnels ou des procédures d'entreprise. En fait, Access est fourni avec des modèles que vous pouvez utiliser en l'état pour suivre toute une variété d'informations et qui rendent les tâches aisées même pour un débutant.

9.4. Lancer Access

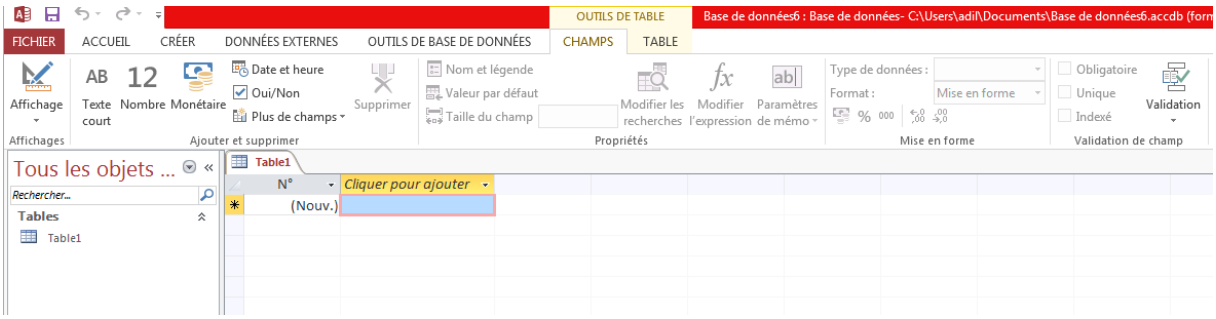
Lorsque vous démarrez Access 2010 (2013 ou 2016) le mode Microsoft Office Backstage apparaît : vous pouvez y obtenir des informations sur la base de données active, créer une nouvelle base de données, ouvrir une base de données existante et afficher du contenu proposé dans Office.com. Le mode Backstage contient également de nombreuses autres commandes que vous pouvez utiliser pour modifier, maintenir ou partager vos bases de données.

Les commandes dans le mode Backstage s'appliquent généralement aux bases de données entières et non aux objets contenus dans une base de données.
Remarque : Vous pouvez afficher le mode Backstage à tout moment en cliquant sur l'onglet Fichier.



9.5. Création d'une nouvelle base de données

Pour ouvrir Access, cliquer sur son icône : 
Cliquer sur **base de données du bureau vide,** spécifier un **nom** et un **emplacement** pour l'enregistrement, puis cliquer sur **créer.** La fenêtre principale de votre base de données apparait :



9.6. Les objets d'Access

On distingue différents types d'objets Access.

- **Table** : Les **tables** servent à **stocker** les informations. Ce sont des fichiers contenant un ensemble d'informations autour d'un même thème ou concept.

Exemple 1 : une table « employés » qui contient des renseignements sur chaque employé (matricule, nom, adresse, fonction, date d'embauche…).
Exemple 2 : une table « articles » qui contiendrait des informations techniques sur chaque produit (référence, désignation, prix, quantité en stock…).

- Les **requêtes** servent à **filtrer** les données en fonction de critères précis. Elles servent donc à extraire les données des tables, permettant de sélectionner une partie des données. Elles permettent également de réaliser des **actions** sur ces données, comme d'effectuer des calculs, des modifications, des suppressions… Les requêtes peuvent être enregistrées et ainsi réutilisées aussi souvent que nécessaire, et vous pourrez par la suite les intégrer aux formulaires et aux états.
- Les **formulaires** permettent la **saisie** et la **modification** d'informations dans les tables, mais de manière plus conviviale que dans les tables : plus besoin de se

contenter d'une présentation sous forme de lignes et de colonnes, vous pourrez ici intégrer des cases à cocher, des listes déroulantes, des titres, des cadres, des images… Les formulaires constituent un environnement facile à manipuler qui vous permettra même de laisser des novices utiliser votre base de données.

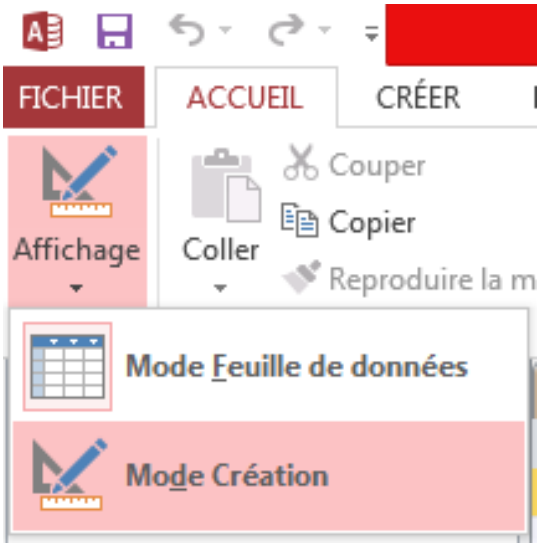
- Les **états** servent à **imprimer** les données, et permettent de présenter un même fichier de données de façons différentes : liste de clients sur trois colonnes, liste de clients par ordre alphabétique de nom et regroupés par région, liste de clients avec adresse complète…
- Les **macros** permettent d'**automatiser** certaines actions (tâches), en programmant des boutons de commande. Par exemple, vous pouvez dans un formulaire ajouter un bouton qui appelle un état.
- Les **modules** servent à **programmer** de manière beaucoup plus pointue que les macros, et ne sont pas destinés au grand public, mais aux utilisateurs avertis. Ils nécessitent la maîtrise du **langage de programmation VBA** (Visual Basic for Applications).
- Le **langage SQL** est le langage informatique universel qui permet de manipuler les objets et les enregistrements des bases de données.

Remarque :
Vous pouvez faire apparaître la fenêtre principale contenant les objets à tout moment en appuyant sur F11.

9.7. Les Tables

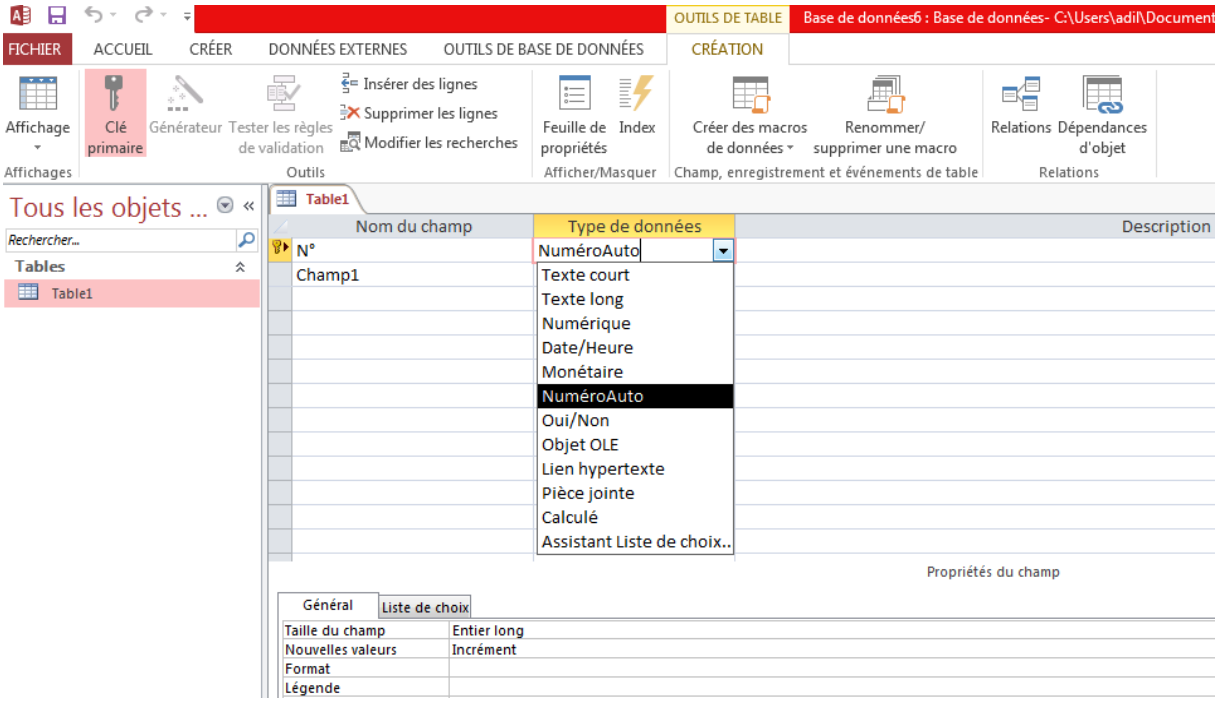
9.7.1 Création des tables

Vous devez commencer par construire la STRUCTURE de votre table, et non pas commencer par y stocker les données. Pour cela, il faut basculer en MODE CRÉATION en cliquant sur le bouton Affichage dans l'onglet Accueil. Saisir le NOM de la table et valider.



9.7.2 Création, modification et formatage des données

Après avoir sélectionner la table à modifier, basculer en MODE CRÉATION en cliquant sur le bouton Affichage dans l'onglet Accueil. Cette fenêtre apparaît :



La première colonne va contenir les **noms des champs**. Chaque champ représente un groupe de données dans la table. (Exemple : nom ou prénom).

La deuxième colonne définit le **type de données** que chaque champ va contenir. Par exemple numérique ou texte ou date… Cliquez dans la case, un menu déroulant vous est proposé : choisir le type désiré.

En fonction du type de données sélectionné, des **propriétés** supplémentaires, sur le formatage de données, vous sont proposées en bas de la fenêtre, dans deux onglets. Vous pouvez par exemple y spécifier la longueur des caractères, le format des données, des masques de saisie…

La troisième colonne **description** est réservée à l'utilisateur, vous pouvez y inscrire librement des commentaires pour chaque champ.

9.7.3 Clé primaire

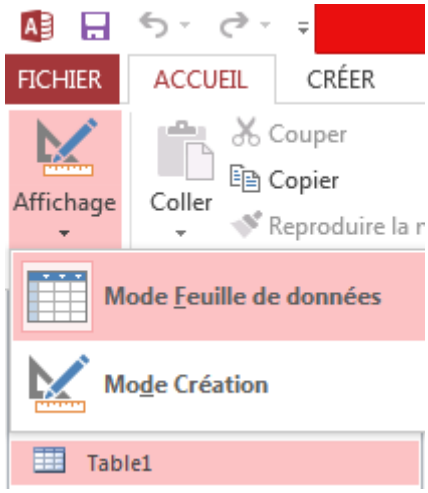
Access a automatiquement proposé un champ portant une clé primaire à votre première table. Ce champ est de type NumeroAuto, ce qui signifie qu'il contiendra un numéro créé automatiquement par Access, et incrémenté à chaque nouvel enregistrement. On affecte la clé primaire à un champ contenant pour chaque enregistrement une information unique. La clé primaire interdit la création de doublons dans le champ qui la contient.

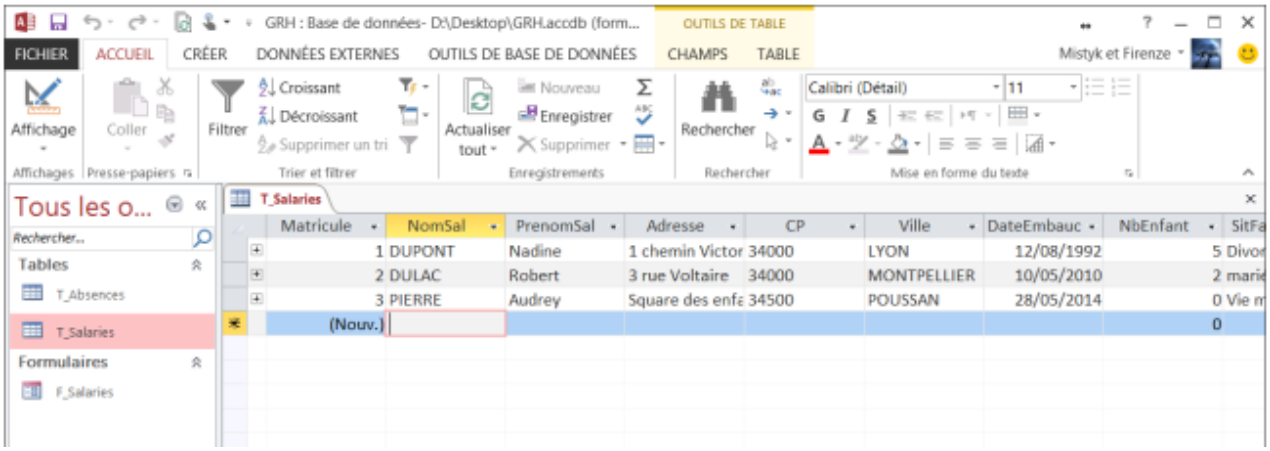
Pour les tables suivantes, il faudra manuellement créer un champ de type NumeroAuto et lui attribuer la clé primaire en cliquant sur l'icône Clé primaire.

Exemple : vous avez une liste de clients, plusieurs d'entre eux peuvent avoir le même nom, voire le même nom et le même prénom, pour être sûre de ne pas les confondre (et ne pas envoyer la facture au mauvais client), on peut ajouter un champ « NoClient » qui portera la clé primaire : deux clients ne pourront avoir le même numéro.

9.7.4 Ajout et stockage de données

L'étape suivante consiste à entrer les données dans la table. Pour retourner en mode de saisie des données, cliquer sur l'icône **Affichage** puis sur **mode feuille de données**.





Chaque colonne représente désormais un **champ**, et chaque ligne un **enregistrement**, c'est-à-dire l'ensemble des informations pour un client, ou un article, ou une commande…
Compléter chaque ligne soigneusement (attention aux erreurs et fautes d'orthographe qui vous gêneront énormément pour les filtres et les requêtes).

Remarque : La **sauvegarde des données** se fait automatiquement lorsque vous quittez une case pour passer à une autre, vous n'avez donc pas à craindre de perdre des informations.

9.7.5 Modification de la structure d'une table

- Pour **ajouter un champ**, il faut retourner en mode création. Ajouter alors le nom et le type de données du nouveau champ.
- Pour **supprimer un champ** (toujours en mode création), cliquer tout à fait à gauche de la ligne du champ à supprimer : la ligne est mise en surbrillance. Appuyer sur la touche Suppr du clavier.

- Pour **supprimer un enregistrement**, en **mode feuille de données**, cliquer tout à fait à gauche de la ligne à supprimer : elle est mise en surbrillance. Appuyer sur la touche **Suppr** du clavier.

9.7.6 Types de données et masques de saisie

Type de données

Chaque table dans Access se compose de champs. Les propriétés d'un champ décrivent les caractéristiques et le comportement des données qui y sont ajoutées. Le type de données d'un champ est la propriété la plus importante, car elle détermine la nature des données que vous pouvez y stocker.
La liste des types de données est présentée dans le tableau.

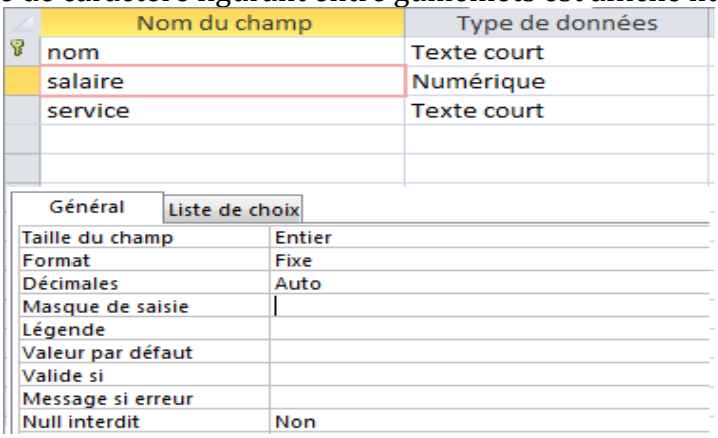
- Champ Texte
- Champ Mémo
- Champ Numérique
- Champ Monétaire
- Champ Date/Heure
- Champ NuméroAuto (Compteur)
- Champ Oui/Non
- Champ Liaison OLE
- Champ Lien hypertexte

Masques de saisie

Lors de la définition des champs, il est possible de définir des masques de saisie pour limiter les erreurs de saisie. Un masque de saisie définit le format de la donnée.
Exemple : Il est possible de préciser qu'un code postal est composé de 5 chiffres : 00000;

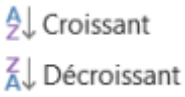
Les caractères utilisés pour composer un masque de saisie sont :

- 0 : l'utilisateur **doit** saisir un chiffre (entre 0 et 9).
- 9 : l'utilisateur **peut** saisir un chiffre (entre 0 et 9).
- # : L'utilisateur **peut** saisir un chiffre, un espace ou un signe plus ou moins ; si aucune valeur n'est saisie, Access insère un espace vide.
- L : L'utilisateur **doit** saisir une lettre.
- ? : L'utilisateur **peut** saisir une lettre.
- A : L'utilisateur **doit** saisir une lettre ou un chiffre.
- a : L'utilisateur **peut** saisir une lettre ou un chiffre.
- & : L'utilisateur **doit** saisir un caractère ou un espace.
- C : L'utilisateur **peut** saisir des caractères ou des espaces.
- > : Le caractère qui suit est converti en majuscule.
- < : Le caractère qui suit est converti en minuscule.
- \ : Le caractère qui suit est affiché littéralement.
- " : La chaîne de caractère figurant entre guillemets est affiché littéralement.



9.7.7 Tri des données

Dans la feuille de données, sélectionner le champ que vous voulez trier en cliquant n'importe où dans la colonne. Cliquer sur **Croissant** ou **Décroissant** dans l'onglet **Accueil**.

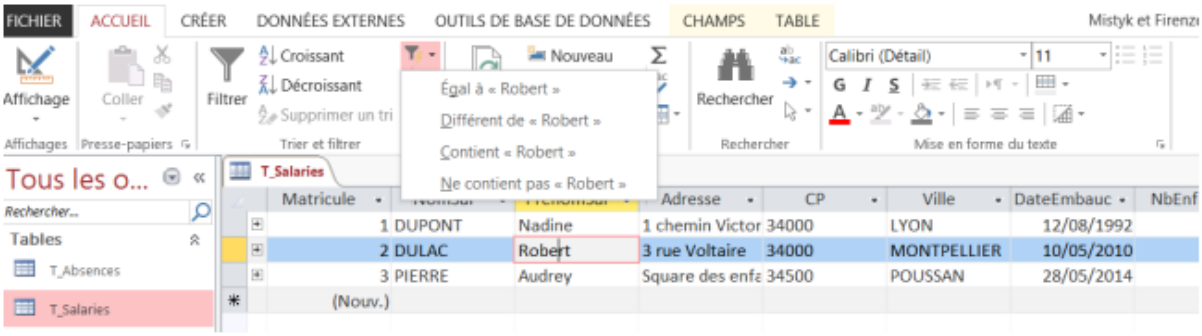


Les enregistrements sont maintenant triés par ordre alphabétique, numérique ou chronologique, selon le type de données.

9.7.8 Filtrage des données dans les tables

Les filtres dans les tables permettent de n'afficher qu'une partie des données de la table, ce qui devient nécessaire lorsque votre base contient de nombreux enregistrement.

Par exemple, vous pouvez ne vouloir afficher que les clients résidant à Paris, ou les commandes passées en février, etc.
Cliquer sur une valeur de votre choix dans la table, puis sur l'icône **SÉLECTION** et sur le **FILTRE** de votre choix.



Les filtres dans les tables sont éphémères, pour les sauvegarder, il faut créer des requêtes.

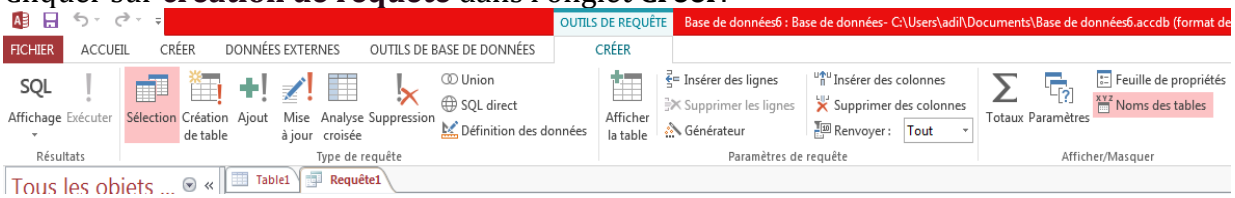
9.8. Requêtes Access

Il existe plusieurs **types de requêtes** :

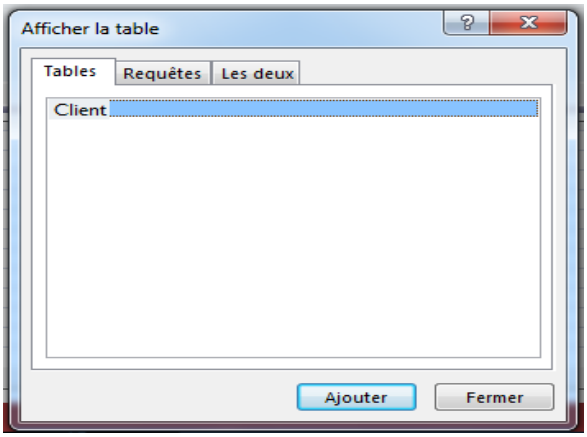
- Les **requêtes sélection** extraient des données en appliquant des **tris** et des **filtres**.
- Les **requêtes calculées** effectuent des **calculs** (TVA, total, pourcentage…).
- Les **requêtes paramétrées** fournissent une grande souplesse aux requêtes en les rendant **interactives**, dynamiques avec l'utilisateur de la base.
- Les **requêtes regroupement** permettent de regrouper les données et d'obtenir des indices **statistiques** (effectifs, moyenne, minimum…).
- Les **requêtes action** permettent de **modifier** les données : supprimer certaines informations, en ajouter d'autres, mettre à jour des données, en importer.
- Les **requêtes analyse croisée** sont l'équivalent du **tableau croisé dynamique** dans Excel.

9.8.1. Les requêtes de sélection

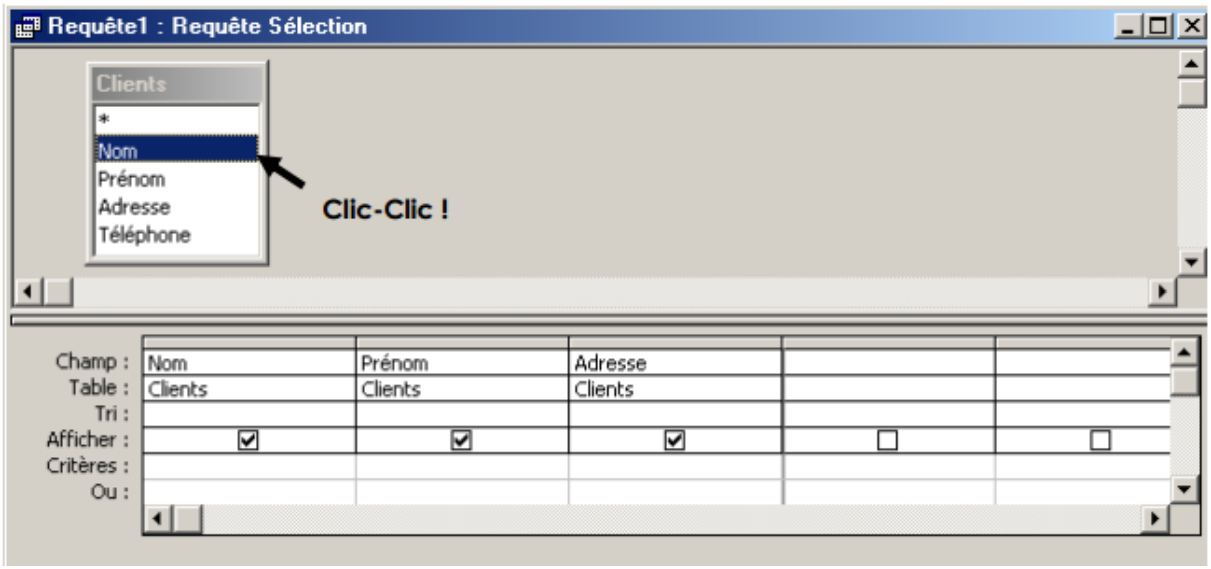
Cliquer sur **création de requête** dans l'onglet **Créer**.



1. Sélectionner la ou les tables nécessaire(s) à la requête en double-cliquant sur leur nom.

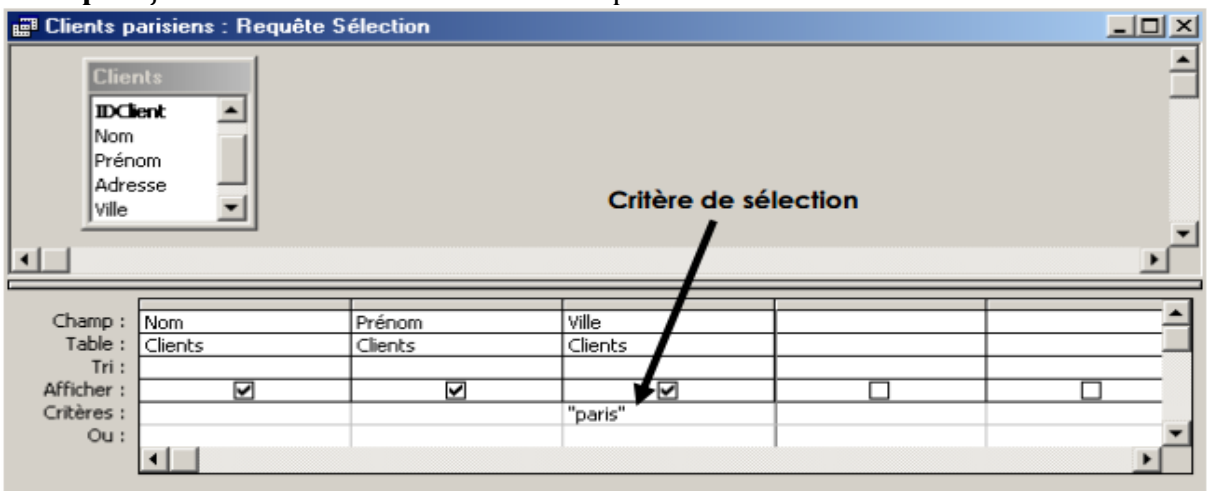


Sélectionner ensuite chaque champ nécessaire à la requête en double-cliquant sur son nom.



Définir vos critères de sélection dans la ligne **CRITÈRES** en respectant la syntaxe d'Access. Si besoin, définir des critères supplémentaires dans les lignes **OU.**

Exemple : je veux afficher la liste des clients parisiens :



Affichez le résultat de la requête en cliquant sur Exécuter.

Requête1 : Requête Sélection

Nom	Prénom	Adresse	Ville
Dulac	Louis	16, rue des Granges	Paris
Fournier	Marie	Impasse du Roy	Paris
Perdoux	Julien	58, rue du Théatre	Paris

Enr : 1 sur 3

Revenir en **mode Création** pour modifier la requête, ou fermer la requête sans oublier de l'enregistrer.

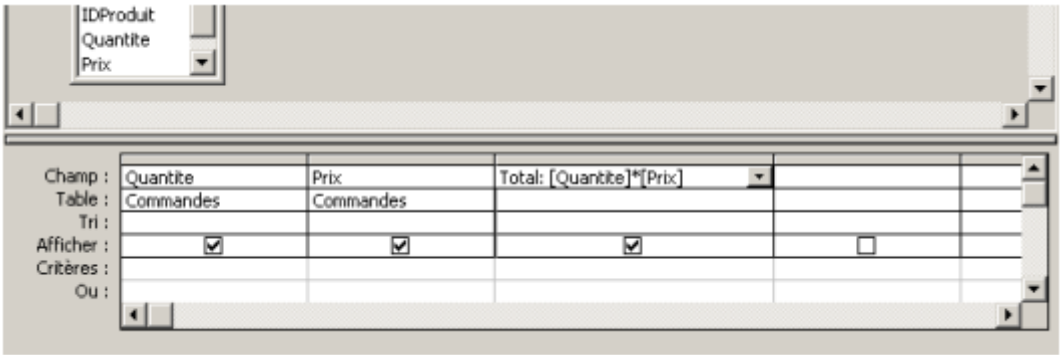
A. Quelques opérateurs de critères :

Logiques	Comparatifs [1]	Arithmétiques	Syntaxiques
Et / Ou / Dans	Supérieur à : >	Additionner : +	« texte »
Pas / !	Inférieur à : <	Soustraire : -	[champ]
Entre ... et ... / -	Sup. ou égal à : >=	Multiplier : *	[paramètre]
Null	Inf. ou égal à : <=	Diviser : /	#date#
* / ? / # [2]	Différent de : <>		nombre
& (concaténation)			oui/non *(case à cocher)*

B. Création d'un champ calculé

Pour créer un **nouveau champ** contenant le calcul, il faut utiliser une **nouvelle colonne**. Donner un **nom** au nouveau champ, suivi de **:** , puis écrire le **calcul** en indiquant les noms des champs concernés entre crochets.

Exemple 1 : PRIX HT: [QUANTITE]*[MONTANT]
Exemple 2 : TVA : [PRIX HT]*20,6/100



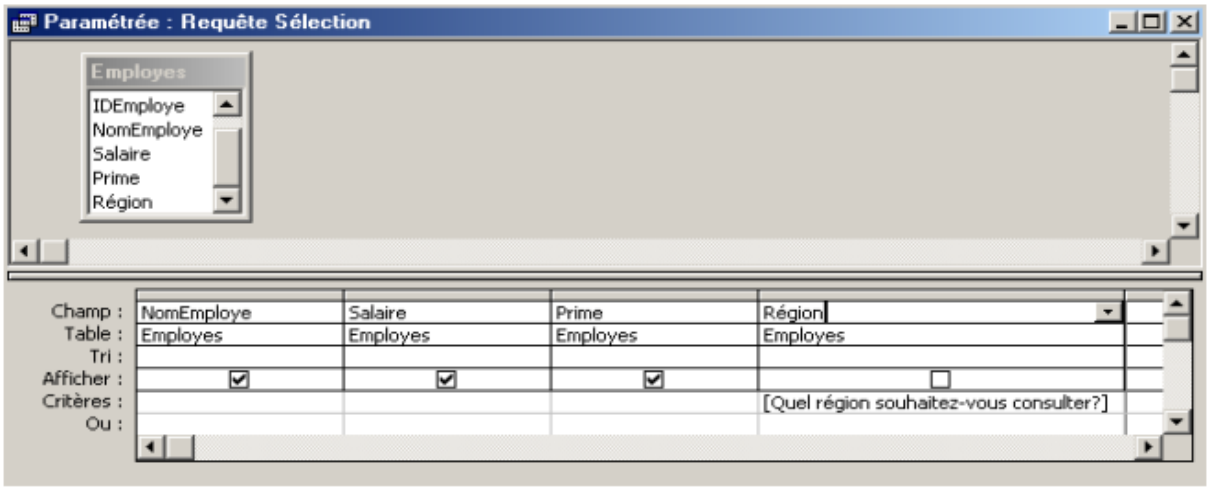
Remarque :

- Les opérateurs de comparaison fonctionnent aussi avec du texte : ils appliquent une comparaison par ordre alphabétique (exemple : « >d » vous donnera la liste des noms placés après le « d » dans l'alphabet).
- L'étoile permet de remplacer n'importe quelle suite de caractères, le ? remplace un seul caractère, le # remplace un chiffre (exemple : « du* » vous donnera la liste des noms commençant par « du », comme duchmol, dutronc, dufour…).

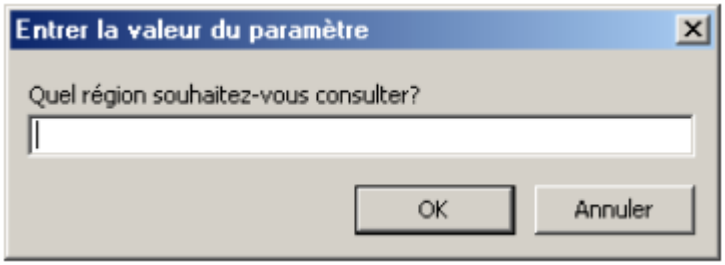
C. Les requêtes paramétrées

Elles permettent à l'utilisateur de définir lui-même le critère à chaque ouverture de la requête, rendant ainsi la requête **interactive** et **réutilisable** avec des critères changeant.
Dans la ligne critère du champ concerné, saisir **ENTRE CROCHETS** le texte à afficher dans la boîte de dialogue.

Par **exemple** : Vous gérez les ventes des agents commerciaux de votre entreprise par région. Vous souhaitez pouvoir consulter indépendamment les résultats de chaque secteur, sans pour autant devoir créer autant de requête qu'il y a de région.

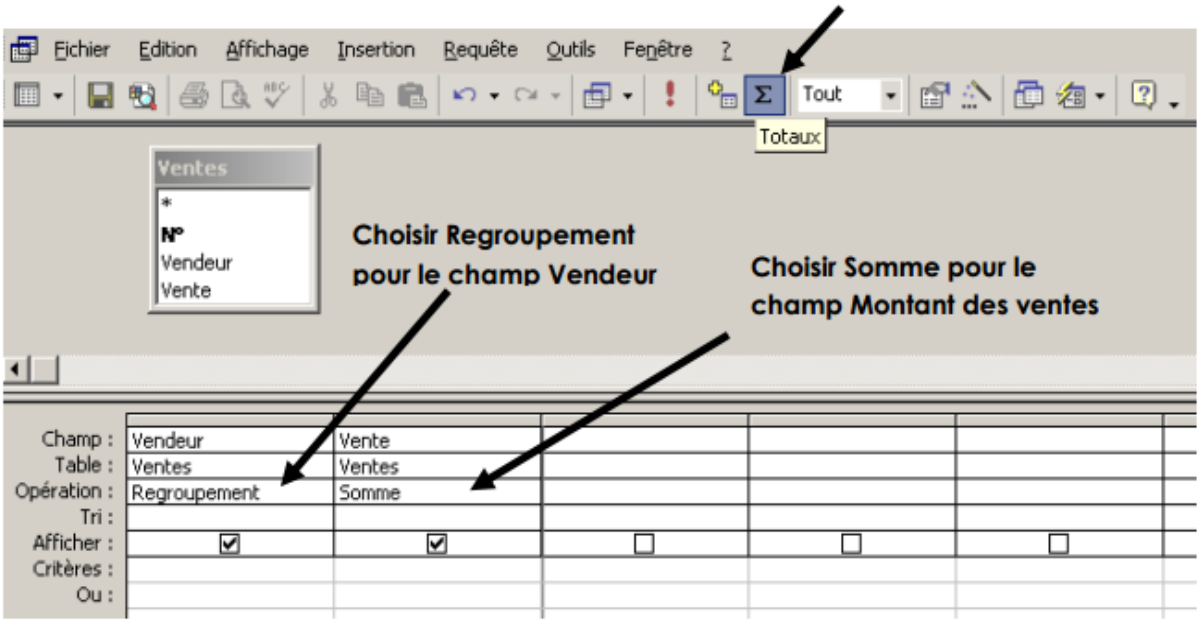


Chaque fois que vous lancerez la requête, vous n'aurez qu'à préciser le secteur choisi dans la boîte de dialogue qui apparaîtra.



D. Les regroupements et statistiques

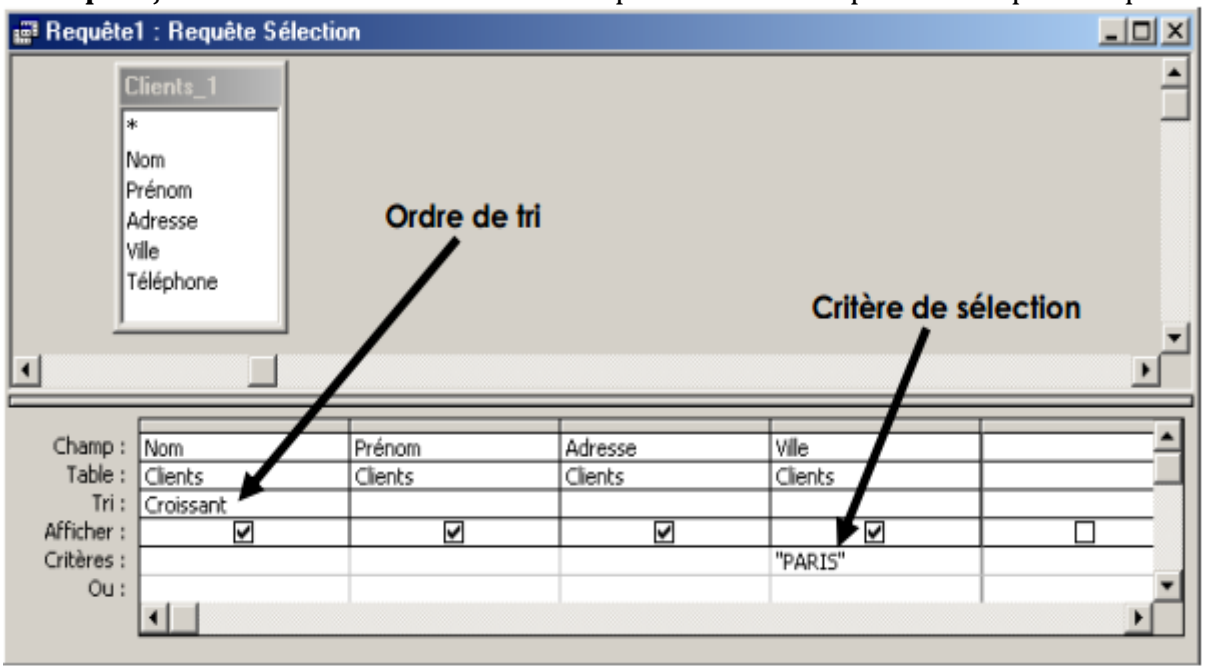
Vous voulez connaître le montant total des ventes de chaque vendeur. Il faut regrouper toutes les ventes par vendeur et effectuer la somme de ses ventes.
Vous avez besoin de la ligne **OPÉRATION**, qu'on obtient en cliquant sur l'icône **∑.**



E. Les tris et premières valeurs

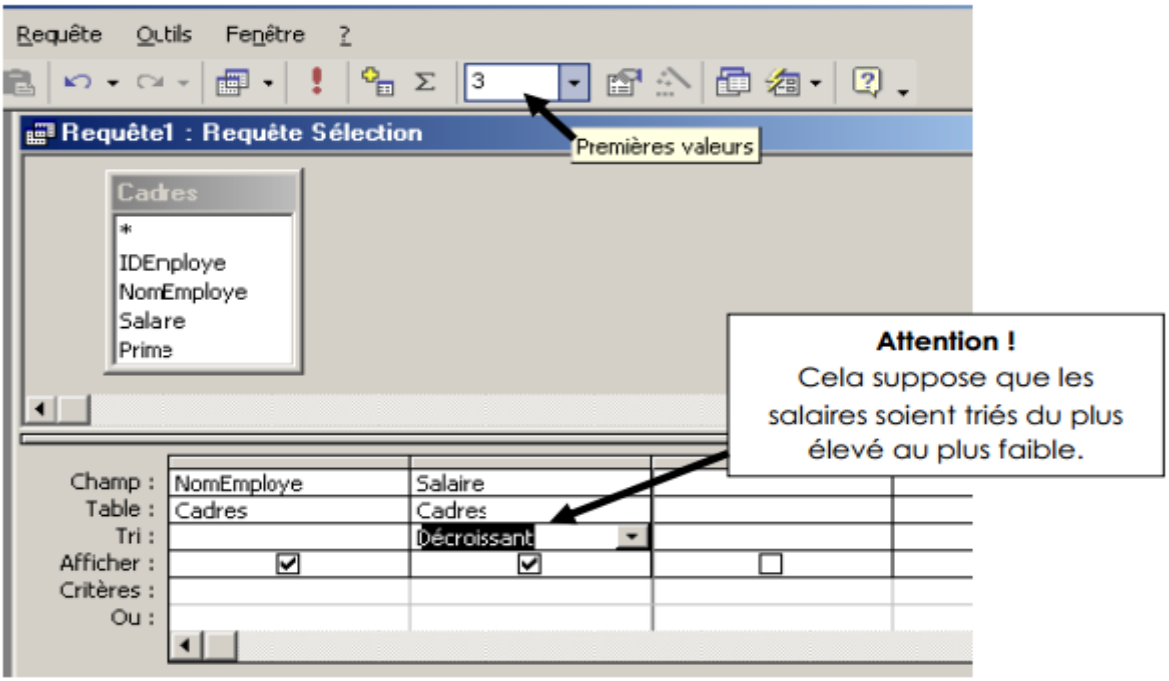
Définir **L'ORDRE DE TRI** (croissant ou décroissant) en cliquant dans la ligne **Tri** et en sélectionnant l'option désirée dans la liste qui apparaît.

Exemple : je veux afficher la liste des clients parisiens classés par ordre alphabétique :



Pour obtenir uniquement les premiers enregistrements du résultat d'une requête, utiliser la fonction **premières valeurs** : taper le nombre ou le pourcentage d'enregistrements désirés.

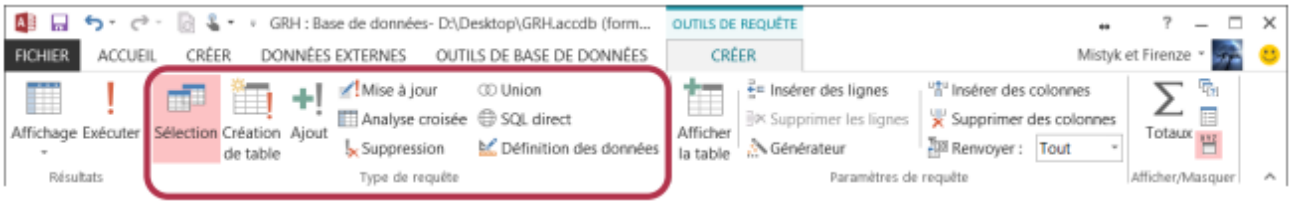
Par **exemple** : je veux afficher le nom des 3 employés les mieux payés :



9.8.2. Les requêtes action

Elles permettent d'effectuer automatiquement une **action** au lieu d'afficher une liste de résultats.

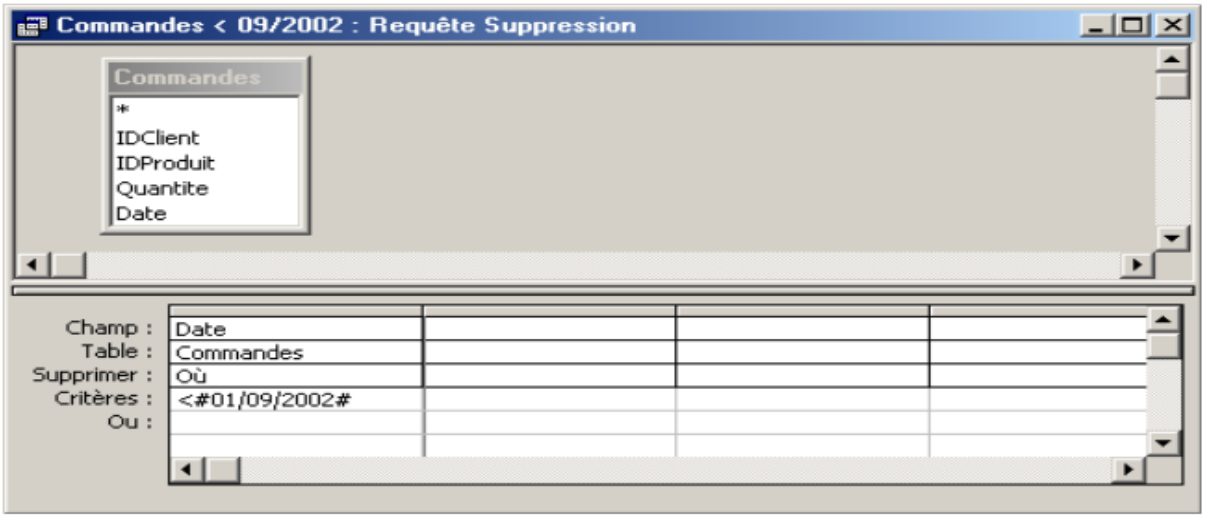
Ajoutez une nouvelle requête en **mode création**, puis précisez le **type de requête** que vous désirez effectuer, sinon vous aurez par défaut une requête sélection.



A. Requête suppression

Elle permet de **supprimer des données en fonction d'un critère**. Sélectionner le type **suppression**, faire basculer les champs contenant les données à supprimer, définir le critère permettant de savoir quelles données doivent être supprimées.

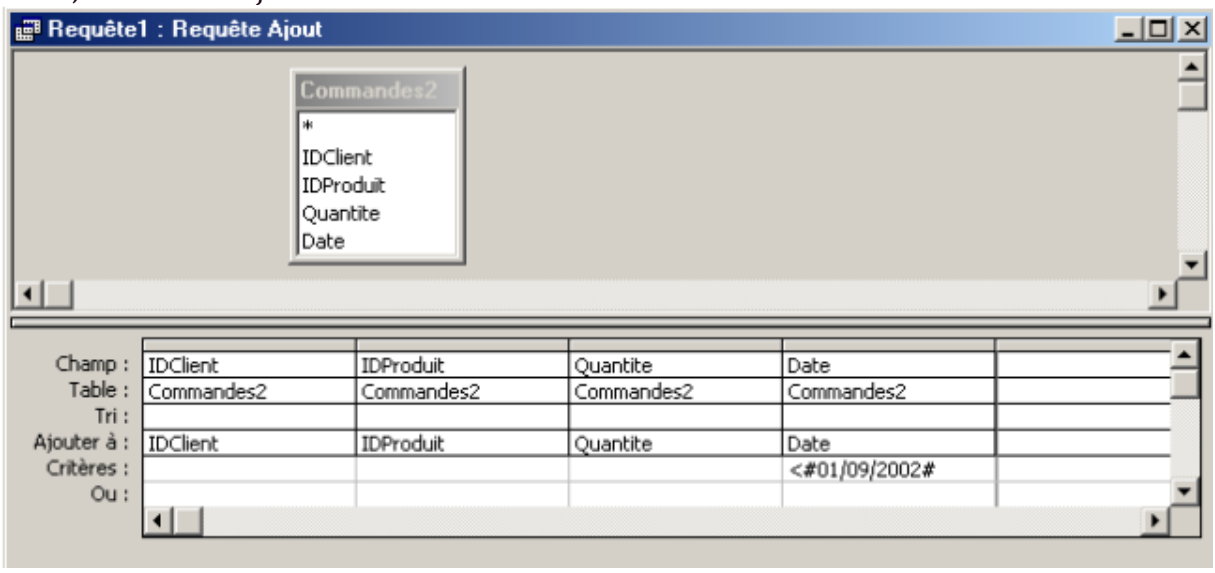
Exemple : supprimer toutes les commandes antérieures à septembre 2002.



B. Requête ajout

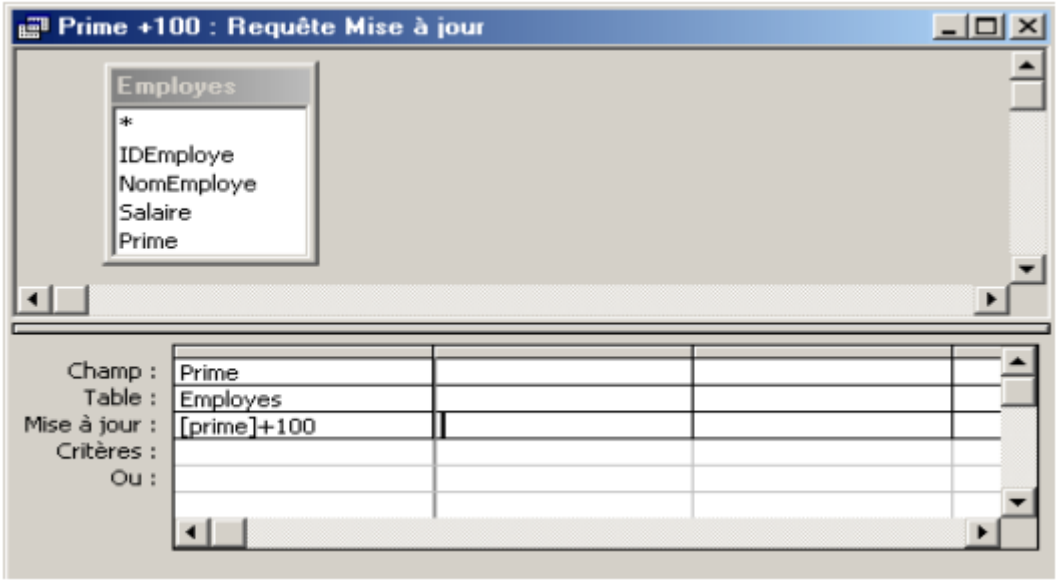
Elle permet d'ajouter à une table des enregistrements provenant d'une autre table (voire d'une autre base de données).
Créez une requête en sélectionnant les tables qui contiennent les enregistrements que vous souhaitez ajouter dans l'autre table.
Choisir le type **ajout**, spécifier le nom de la table de destination (celle qui va recevoir de nouveaux enregistrements).
Faire basculer les champs de la table d'origine (celle d'où proviennent les données) dans la requête, et éventuellement définir un critère si on ne veut pas ajouter tous les enregistrements de la table d'origine.

Exemple : on a une table « Commandes2 » contenant des commandes antérieurs à sept 2002, on veut les ajouter à la table « Commandes » :



C. Requête mise à jour

Elle permet de **modifier les données** d'un champ. Sélectionner le type **MISE À JOUR**, faire basculer le champ à modifier, dans la ligne **Mise à jour**, procéder au changement.
Exemple : on a une table « employés » contenant un champ « prime », on veut augmenter la prime de chaque employé de 100 € :



D. Requête création de table

Le résultat de la requête apparaîtra dans une nouvelle table.
Sélectionner le type **CRÉATION DE TABLE**. Dans la boîte de dialogue qui s'affiche, taper le nom de la table à créer ou à remplacer.

9.8.3. Requête analyse croisée

Ce type de requête nécessite **3 champs** :

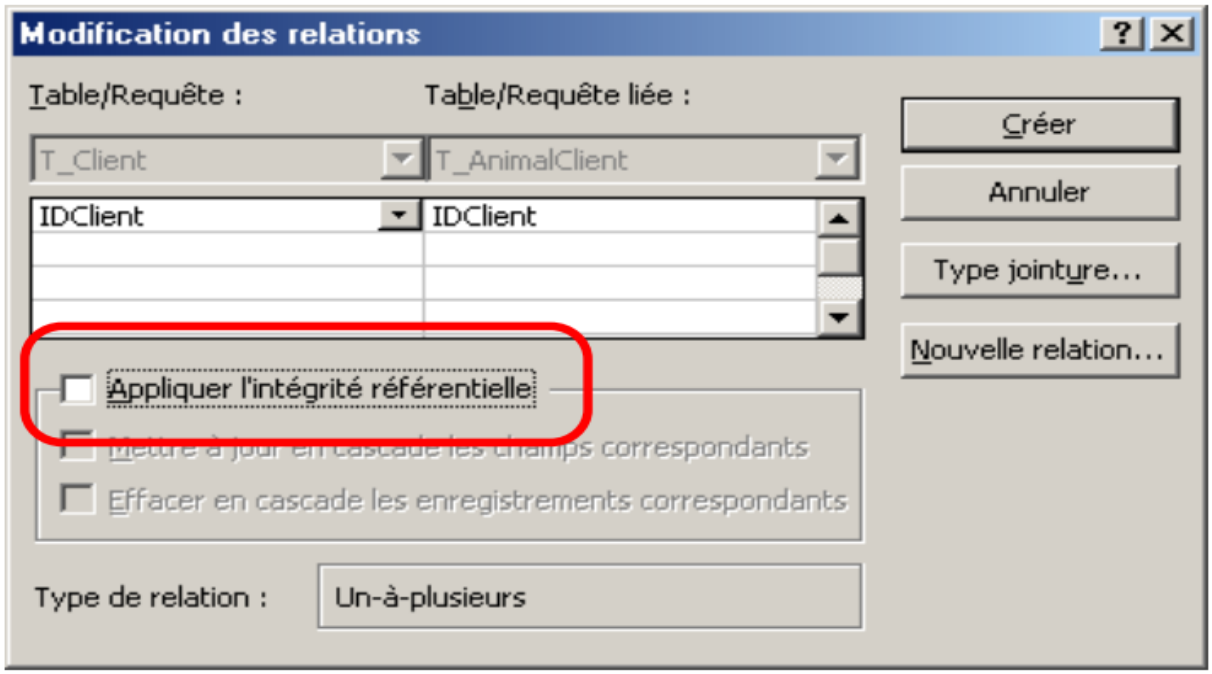
- Un **champ en en-tête de colonne**, auquel on applique un regroupement ;
- Au moins un **champ en en-tête de ligne**, auquel on applique un regroupement ;
- Un **champ valeur** : généralement de type numérique pour pouvoir y appliquer un indice statistique.

Champ :	Vendeur	Rayon	Vente		
Table :	Ventes	Ventes	Ventes		
Opération :	Regroupement	Regroupement	Somme		
Analyse :	En-tête de ligne	En-tête de colonne	Valeur		
Tri :					
Critères :					

9.9. Les relations

Dans l'onglet Outils de base de données, cliquer sur RELATIONS :

Ajouter toutes les tables à la fenêtre des relations (comme pour une requête). Faire glisser avec la souris les champs communs à deux tables les uns sur les autres. Exemple : « N°Client » de la table « Client » vers « N°Client » de la table « Commande » Access propose de créer un lien :



Cliquer sur **appliquer l'intégrité référentielle**. Cela permet d'établir une relation de type **un à plusieurs**.

Exemple 1 : Pour un client, il peut y avoir plusieurs véhicules ; mais pour un véhicule, il n'y a qu'un propriétaire. Dans ce type de relation, vous ne pourrez pas enregistrer un véhicule sans qu'il soit lié à un et un seul propriétaire.
Exemple 2 : Pour un pays, il peut y avoir plusieurs habitants. Mais pour un homme, il ne peut y avoir qu'un pays d'origine. Chaque personne sera liée à un et un seul pays d'origine.
Cliquer enfin sur **CRÉER**. Une ligne noire avec les symboles **1** et **∞** relie maintenant les tables.

Remarque : **Pour supprimer une relation :** cliquer sur le lien noir puis sur la touche **SUPPR** du clavier. Confirmer la suppression.

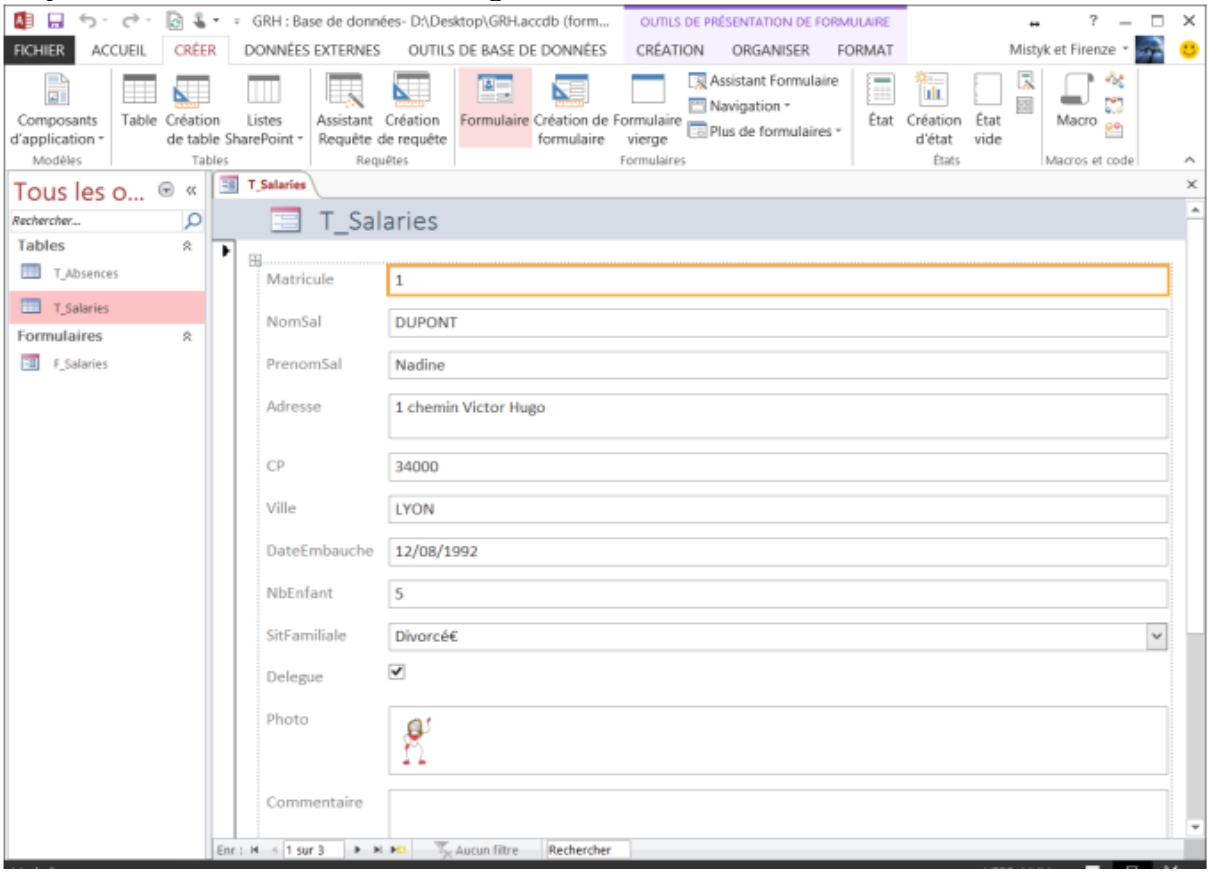
9.10. Manipulation de la structure Access

9.10.1. Les formulaires

Les **FORMULAIRES** permettent la **SAISIE** et la **MODIFICATION** d'informations dans les tables, mais de manière plus conviviale : plus besoin de se contenter d'une présentation sous forme de lignes et de colonnes, vous pourrez ici intégrer des cases à cocher, des listes déroulantes, des titres, des cadres, des images… Les formulaires constituent un environnement facile à manipuler qui vous permettra même de laisser des novices utiliser votre base de données.

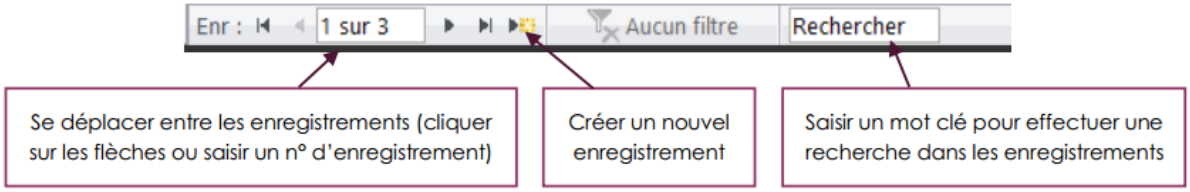
9.10.2. Création automatique

Dans le navigateur, sélectionner la **table** ou la **requête** qui servira de **source** à votre formulaire.
Cliquer sur **FORMULAIRE** dans l'onglet **Créer**. Vous obtenez un formulaire instantané :



9.10.3. Utilisation du formulaire

Basculer en **MODE FORMULAIRE**.
Pour naviguer entre les enregistrements, utiliser la zone au bas du formulaire :

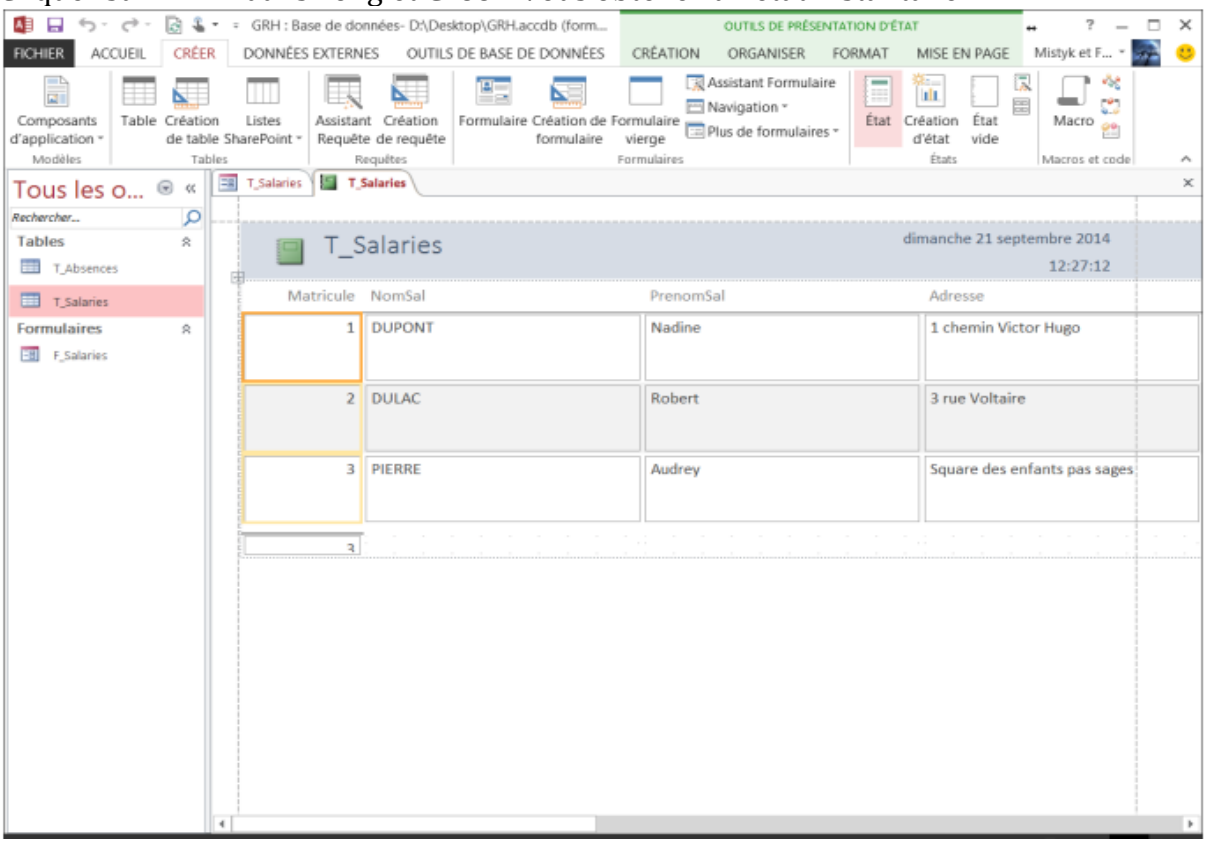


9.10.4. Les états

Les **ÉTATS** servent à **IMPRIMER** vos rapports de données, il est ainsi possible de présenter une même liste de données de façons différentes : liste de clients sur 3 colonnes, liste de clients par ordre alphabétique de nom et regroupés par région, liste de clients avec adresse complète…

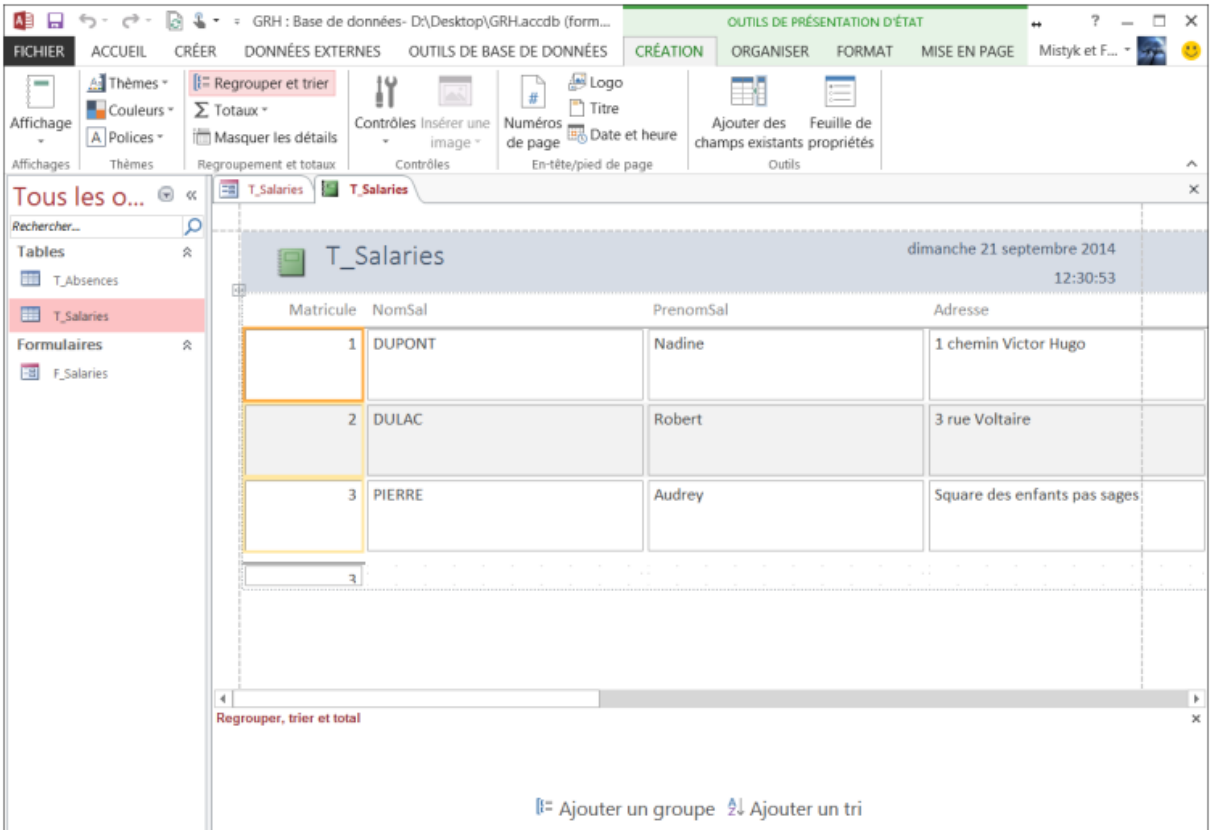
9.10.5. Création automatique

Dans le navigateur, sélectionner la **table** ou la **requête** qui servira de **source** à votre rapport.
Cliquer sur **ÉTAT** dans l'onglet **Créer**. Vous obtenez un état instantané :

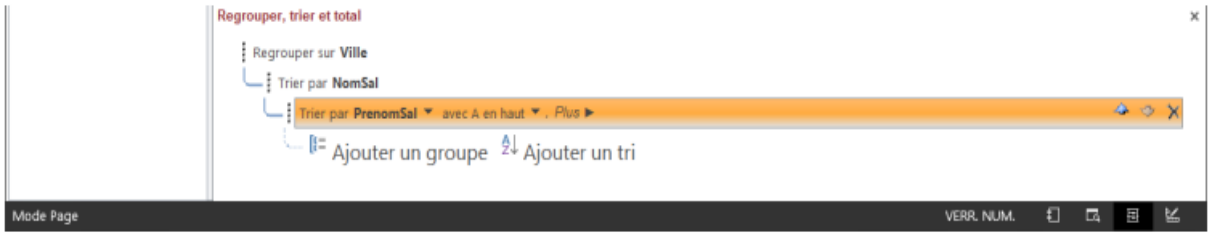


9.10.6. Etats avec groupe

Access vous permet de regrouper ou de trier les enregistrements dans votre rapport.
Par exemple, vous voulez la liste des clients classés par ville de résidence.
Cliquer sur **REGROUPER ET TRIER** dans l'onglet **Création**.



Une nouvelle zone apparait au bas de l'état, dans laquelle vous pouvez ajouter des **ordres de tri et de regroupement**.



9.10.7. Les macros

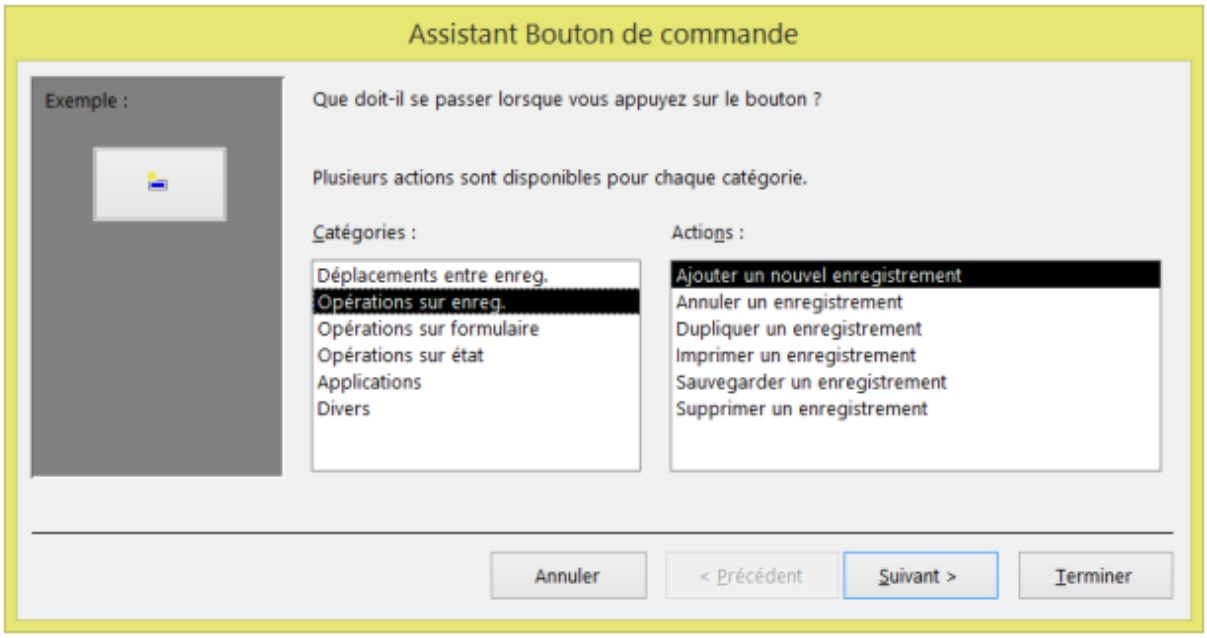
Les **MACROS** permettent d'**automatiser** certaines actions, en programmant des **BOUTONS DE COMMANDE** qui seront visibles dans les **formulaires** ou les **états**. Par exemple, vous pouvez ajouter un bouton de commande pour appeler un état, pour imprimer, fermer un formulaire, lancer une requête, un programme, quitter Access…

A. Bouton de commande avec macro automatique

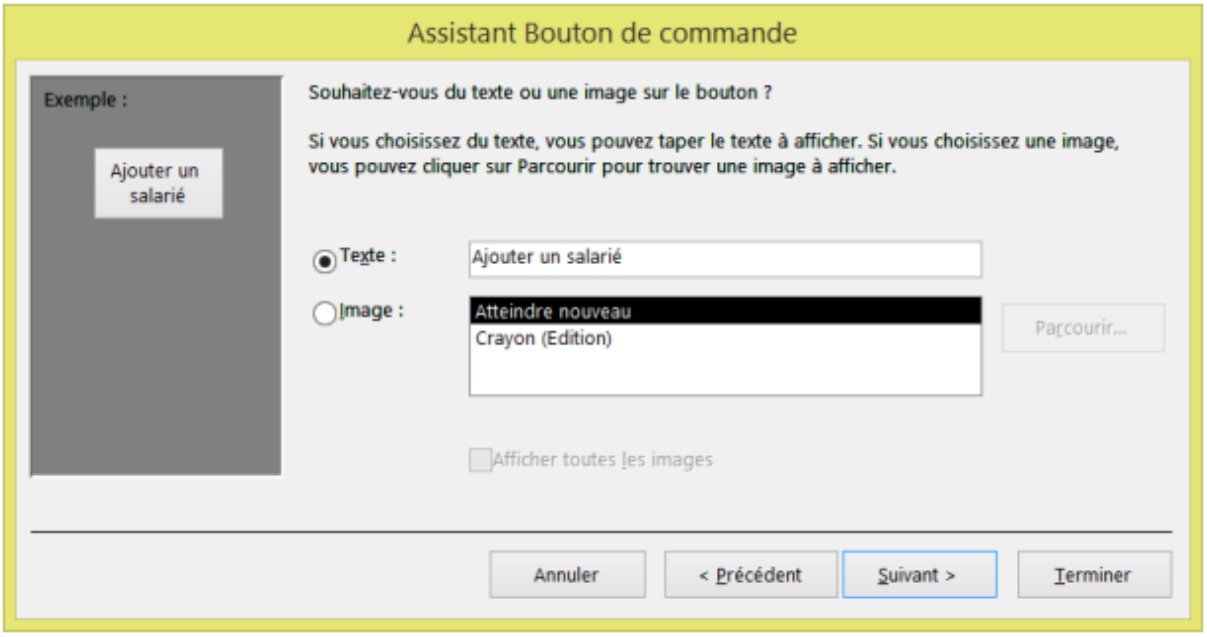
Ouvrir le formulaire ou l'état concerné en **MODE CRÉATION**.
Dans l'onglet **Création**, cliquer sur le **contrôle BOUTON**. Cliquer ensuite dans le formulaire ou l'état à la **position** où l'on souhaite placer le bouton de commande.

Une **boite de dialogue** apparait dans laquelle on choisit **l'action** que l'on souhaite provoquer en cliquant sur le bouton de commande.

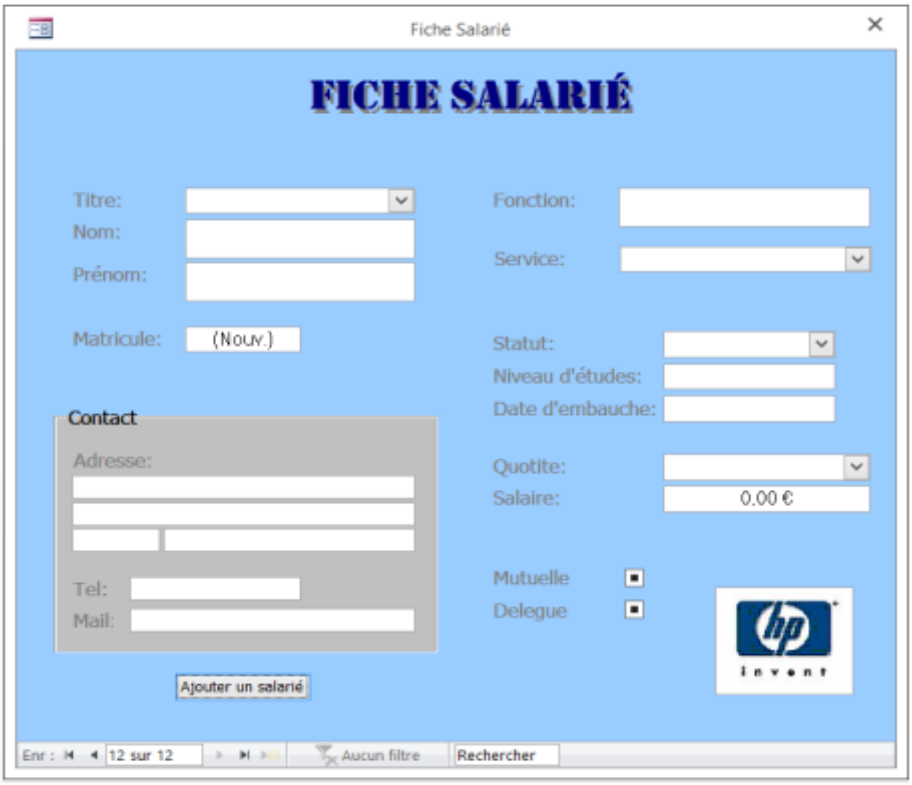
Par exemple, on peut paramétrer un bouton pour accéder à la saisie d'un nouveau salarié :



L'étape suivante permet de paramétrer l'aspect du bouton (texte ou image) :



Basculer en **MODE FORMULAIRE**, un bouton parfaitement opérationnel propose à présent la saisie d'un nouveau salarié :

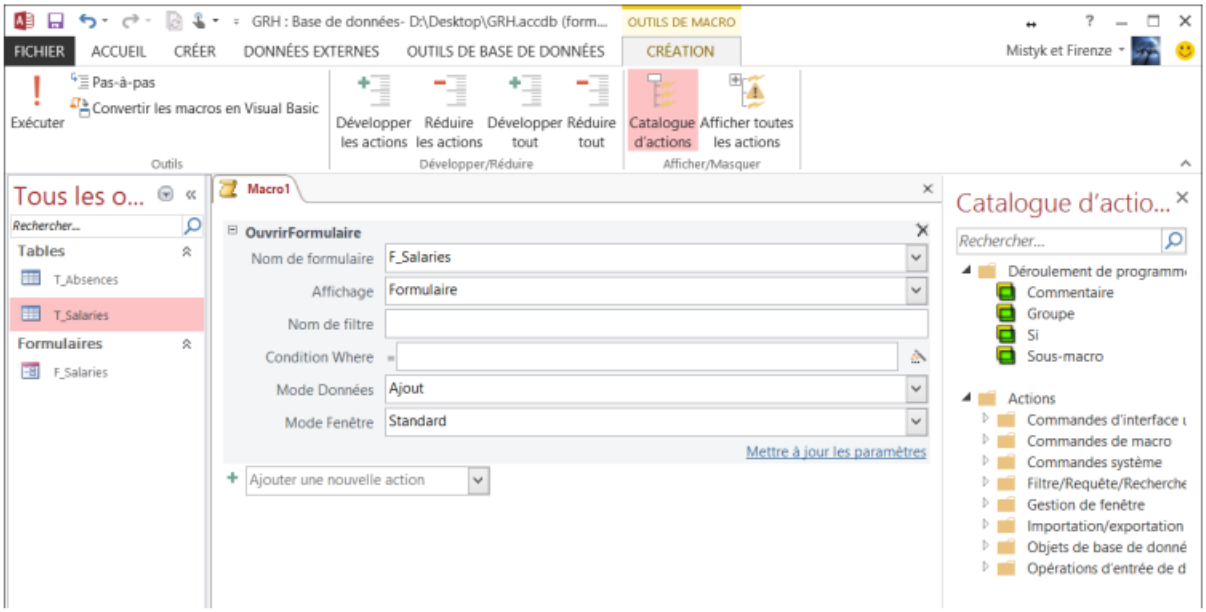


B. MACRO

La boite de dialogue proposée lors de la création d'un bouton de commande présente une liste limitée d'action, et ne permet pas au bouton d'effectuer plusieurs actions. Pour un choix d'action plus large et pour permettre à un bouton d'effectuer plusieurs actions successives, il faut créer manuellement la macro, puis l'affecter au bouton de commande.

Créer la macro

Dans l'onglet **Créer**, cliquer sur **MACRO**. Choisir l'action que votre macro doit effectuer dans la **LISTE DÉROULANTE** au milieu de l'écran.
Il faut ensuite paramétrer les **PROPRIÉTÉS** en fonction de l'action choisie.



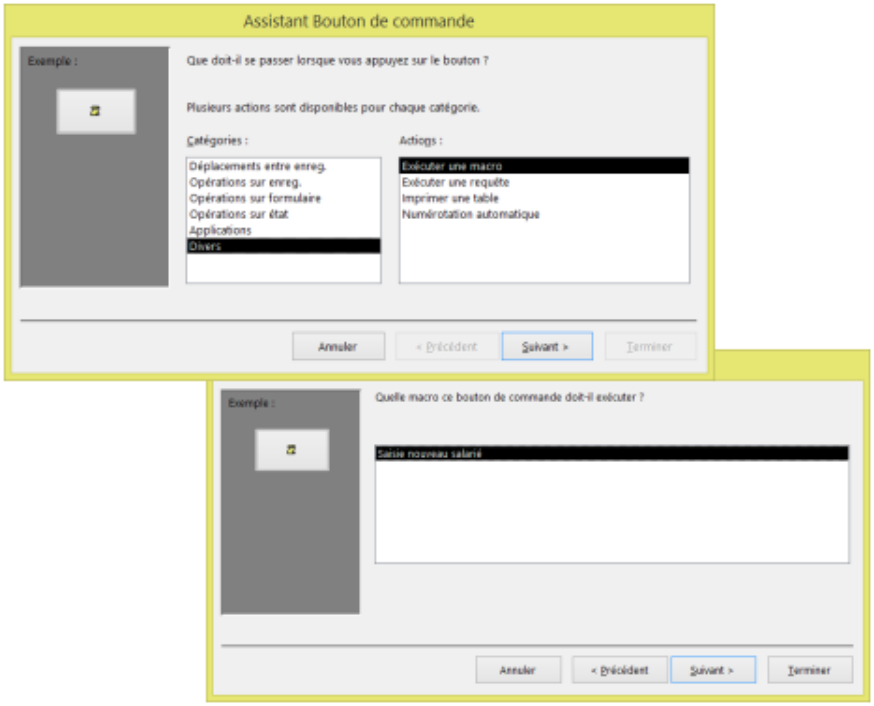
Il est possible de programmer **plusieurs actions consécutives** dans une seule macro. Access vous demande d'enregistrer la macro à la fermeture de la fenêtre.

Affecter la macro au bouton de commande

Pour utiliser la macro, il faut l'affecter à un **bouton de commande** dans un **formulaire** ou un **état**.

En **mode création**, ouvrir le formulaire ou état et insérer un **bouton** comme vu précédemment.

Dans la **boite de dialogue** de l'assistant, sélectionner la catégorie **DIVERS** et l'action **exécuter une macro**. L'étape suivante vous permettra de **sélectionner la macro** à activer.



9.10.8. Naviguer dans la base de données

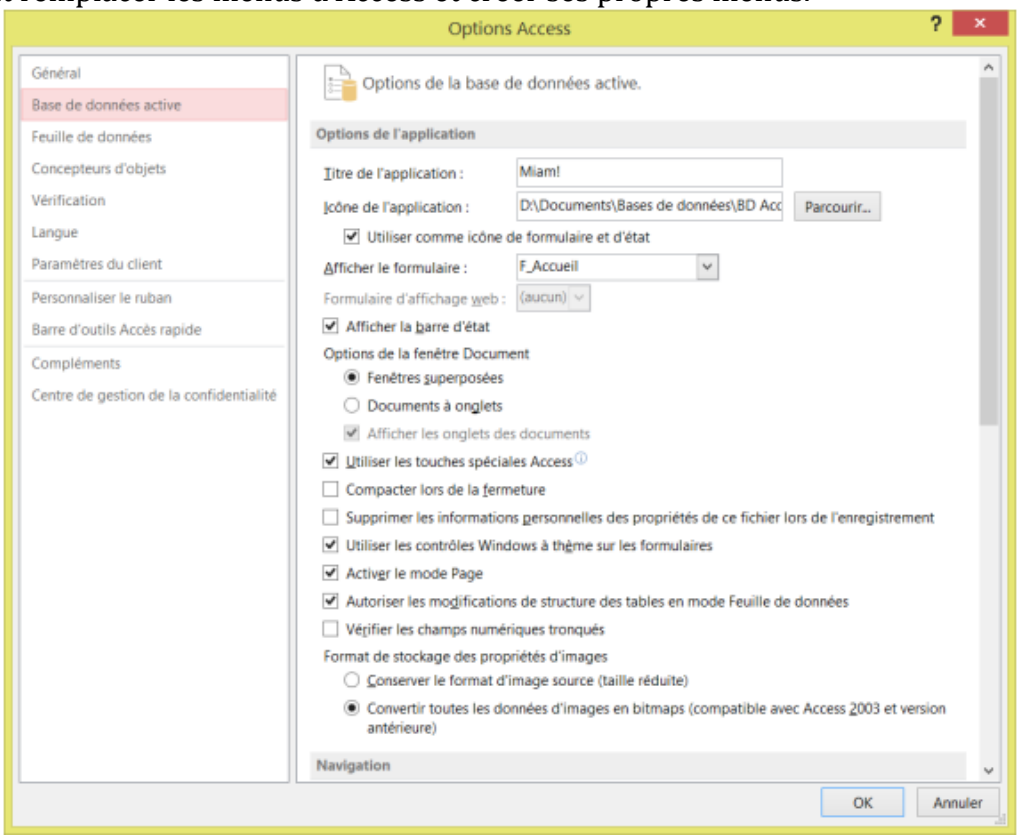
Un **MENU GÉNÉRAL** est un **formulaire** permettant de naviguer dans la base de données: ouvrir les formulaires, les états, les requêtes, quitter l'application…

Ce formulaire n'est basé sur aucune table ou requête mais contient principalement des **boutons de commandes**.

9.10.9. Options de la base de données

De nombreuses options sont proposées pour personnaliser votre base de données et en faire une véritable application sur mesure. Cliquer sur **fichier**, sur **options** puis sur **base de données active** :

Exemple :
Au démarrage de la base de données, on peut lancer l'ouverture d'un formulaire précis (par exemple le menu général). On peut également personnaliser le nom et l'icône de la base de données (remplacer l'icône et le nom Access).
On peut remplacer les menus d'Access et créer ses propres menus.



10. Algèbre relationnelle et forme normale

L'algèbre relationnelle se compose d'un ensemble d'opérateurs opérant sur des relations et produisant de nouvelles relations. Il est donc possible de construire de nouvelles informations à partir des relations de départ et d'une composition séquentielle d'opérateurs.

- ❖ On peut classifier les opérateurs relationnels en trois catégories :
  - Les opérateurs **unaires** : affectation, sélection et projection.
  - Les opérateurs **binaires** travaillant sur des relations de **même schéma** : union, intersection et différence.
  - Les opérateurs **binaires** travaillant sur des relations de **schémas différents** : jointure, produit cartésien, théta-jointure et division.
- ❖ La présentation des opérateurs est illustrée par un exemple de gestion d'une base d'invitations. Cette base de données décrit les personnes invitées et les plats qui ont été servis. Elle est composée de trois relations :
  - REPAS(**date,invité**) donne la liste des invités qui ont été reçus et à quelle date;
  - MENU(**date,plat**) donne le menu servi à chaque date
  - PREFERENCE(**personne,plat**) donne pour chaque personne ses plats préférés.

Remarque : les attributs "personne" et "invité" ont même domaine et les clés sont en gras.

10.1. Opérateurs unaires

- **Affectation :** R(A1, ..., An) <-- expression de sélection

l'affectation permet de sauvegarder le résultat d'une expression de recherche, ou bien de renommer une relation et ses attributs.

Exemple : **Quels sont les invités du repas du 010597 ?**

En algèbre

R1 <- SELECTION date=010597 (REPAS)
PROJECTION invité (R1)

En SQL

```
SELECT distinct invité FROM REPAS
WHERE date=010597
```

- **Sélection** : SELECTION condition-de-sélection (R)

La sélection prend en entrée une relation R définie sur un schéma SR et produit en sortie une nouvelle relation de même schéma SR ayant comme nuplets ceux de R satisfaisant à l'expression de sélection.

Exemple : **Quels sont les plats qui ont été servis à Alice ?**

En algèbre :

PROJECTION plat (SELECTION invité=Alice(REPAS) * MENU)

En SQL :

```
SELECT distinct plat
FROM REPAS R, MENU M
WHERE R.date=M.date AND invité='Alice‘
```

- **projection** : PROJECTION A1, ..., An (R)

la projection prend en entrée une relation R définie sur un schèma SR et produit en sortie une nouvelle relation de schéma A1, ..., An (schéma inclus dans SR) ayant comme nuplets ceux de R restreints au sous-schéma A1, ..., An.

Exemple : **Quels sont les invités qui lors d'un repas ont eu au moins un de leur plat préféré ?**

En algèbre :

PREFERENCES1 (personne1, plat1) <- PREFERENCES

PREFERENCES est renommée pour pouvoir faire une theta-jointure avec MENU (sinon un attribut est commun et la theta-jointure est alors impossible).

PROJECTION invité ((MENU * REPAS)*(invité=personne1 ET plat=plat1) PREFERENCES1)

En SQL :

```
SELECT distinct invité
FROM REPAS R, PREFERENCES P, MENU m
WHERE R.invité=P.personne AND
R.date=M.date AND P.plat=M.plat
```

10.2. Opérateurs binaires de même schéma

Les trois opérateurs ensemblistes opèrent sur des relations R et S de même schéma SRS.

- **Union** : R UNION S

Produit une nouvelle relation de schéma SRS ayant les nuplets de R et ceux de S (les doublons sont supprimés).

- **Intersection :** R INTERSECTION S

Produit une nouvelle relation de schéma SRS ayant les nuplets présents dans R et dans S.

- **différence** : R - S

Produit une nouvelle relation de schéma SRS ayant les nuplets de R qui ne sont pas dans S.

Exemple : **Quelles sont les personnes qui n'ont jamais été invités ?**

En algèbre :

PROJECTION personne (PREFERENCES) –
PROJECTION invité (REPAS)

En SQL :

```
SELECT personne FROM PREFERENCES
MINUS
SELECT invité FROM REPAS
```

10.3. Règles de passage de l'algèbre relationnelle vers SQL

Soient R(X) et S(Y) deux relations de schémas distincts, T(V,W), U(W,Z) deux relations ayant un ensemble d'attributs communs, Q1(X), Q2(X) deux relations de même schéma et P(W).

Projection : PROJECTION A1, ..., An (R) --> select distinct A1, ..., An
from R ;

Sélection : SELECTION condition (R) --> select *
from R
where condition ;

Jointure : T * U --> select V, W, Z
from T, U
where T.W = U.W

N.B. En réalité égalité sur tous les attributs communs.

10.4. Formes normales

Les formes normales définissent un ordre partiel sur les schémas de relation. On peut donc voir une forme normale comme une classe d'équivalence (on peut comparer deux schémas dans deux classes d'équivalence différentes mais pas dans la même). Il faut aussi noter que le seul élément qui est pris en compte par les formes normales est la non redondance d'informations d'un schéma. Selon les formes normales un "bon" schéma est un schéma sans redondance (ce qui ne veut pas forcément dire qu'il est efficace par exemple). Un schéma relationnel sans qualité particulière est appelé schéma en 1ère forme normale (on note 1FN) et si on rajoute certaines qualités on obtient les deuxième et troisième formes normales (on note 2FN et 3FN).

1ère forme normale

❖ **Définition**

Une relation est en première forme normale si tous ses attributs sont atomiques (inhérent au modèle relationnel).

❖ **Un attribut atomique n'est pas :**

- Multivalué (liste de valeurs) ;
- Composé (structuré en sous-attributs).

2ème forme normale

❖ **Définition**

Une relation est en deuxième forme normale ssi :

- Elle est en première forme normale ;
- Tout attribut non clé dépend de la totalité de toutes les clés.

❖ Exemple

C (nc, dateexp, qtéexp, nb) pas en 2FN car nc, dateexp clé et nc fi nb.

3ème forme normale

❖ Objectif : élimination des redondances dues aux dépendances fonctionnelles déduites par transitivité

❖ Définition :

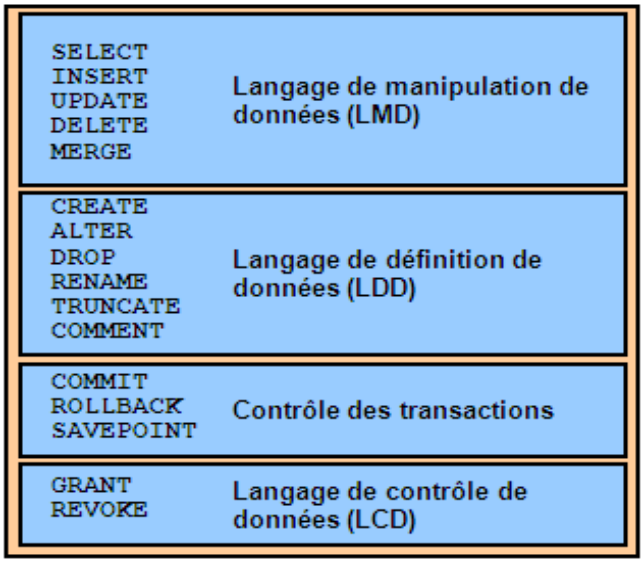
Une relation est en troisième forme normale ssi :

- Elle est en deuxième forme normale,
- Il n'existe aucune DF entre attributs non clé (si tout attribut non clé ne dépend pas d'un autre attribut non clé).

11. Langage SQL

Le langage SQL (Structured Query Language) est un langage de requête structuré qui permet de définir, extraire et manipuler les données des tables d'une base de données. SQL est un langage efficace et facile à apprendre et à utiliser. SQL est composé d'un :

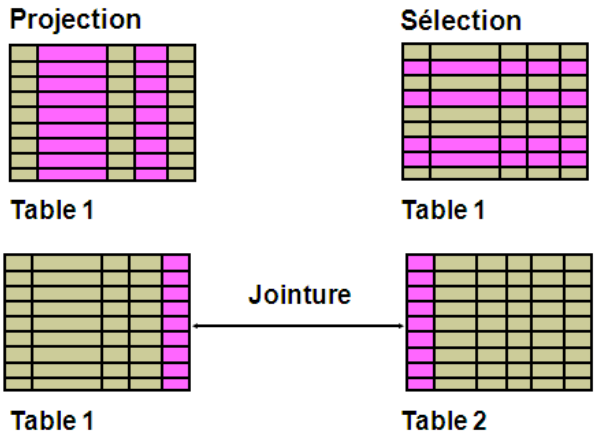
- Langage de définition de données (LDD) : permet de créer, modifier et supprimer des tables dans une base de données.
- Langage de manipulation de données (LMD) : permet de sélectionner (langage d'interrogation), insérer, modifier ou supprimer des données dans une table d'une base de données.
- Langage de contrôle de données (LCD) : permet de définir des permissions au niveau des utilisateurs d'une base de données.



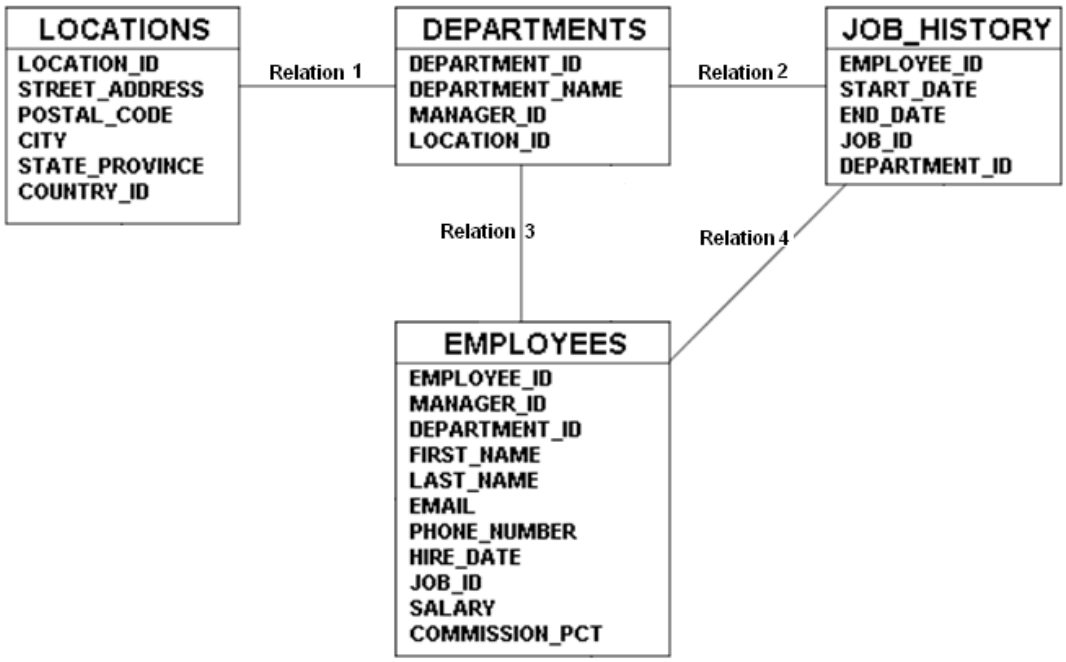
12. Langage d'interrogation de données – Instruction Select

Une instruction SELECT extrait des informations de la base de données. L'instruction SELECT offre les possibilités suivantes :

- **Projection** : choisissez les colonnes d'une table qui sont renvoyées par une interrogation. Choisissez le nombre de colonnes nécessaires (peu ou beaucoup).
- **Sélection** : choisissez les lignes d'une table qui sont renvoyées par une interrogation. Divers critères peuvent être utilisés pour limiter les lignes extraites.
- **Jointure** : combinez des données stockées dans différentes tables en indiquant le lien entre elles.



La structure de la base de données est donnée ci-dessous :



Instruction SELECT de base

SELECT *|{[DISTINCT] *column|expression,...*}
FROM *table;*

Dans la syntaxe :

SELECT	est une liste d'une ou plusieurs colonnes
*	sélectionne toutes les colonnes
DISTINCT	supprime les doublons
column|expression	sélectionne la colonne nommée ou l'expression
FROM *table*	désigne la table contenant les colonnes

Exemple: **Select last_name, salary**
from employees ;

Cette requête affiche le nom et le salaire de tous les employés de la table employees.
Vous pouvez afficher toutes les colonnes de données d'une table en faisant suivre le mot-clé SELECT d'un astérisque (*). Dans l'exemple ci-dessous, la table departments contient quatre colonnes : DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID et LOCATION_ID. La table contient huit lignes, une pour chaque département.

Select *
from departments ;

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

12.1. Expressions arithmétiques

Vous pouvez modifier la façon dont les données s'affichent, effectuer des calculs ou examiner des scénarios de simulation. Tous sont possibles à l'aide d'expressions arithmétiques. Une expression arithmétique peut contenir des noms de colonnes, des valeurs numériques constantes et des opérateurs arithmétiques (+ : ajouter, - : soustraire, * : multiplier et / : deviser). Les opérateurs * et / ont la priorité la plus élevée. On trouve ensuite, à un même niveau, les opérateurs + et -.

Exemple : **SELECT last_name, salary, salary + 300**
FROM employees;

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

...

20 rows selected.

L'exemple ci-dessus utilise l'opérateur d'addition pour calculer une augmentation de salaire de 300 dh pour tous les employés. L'exemple affiche également une colonne SALARY+300 dans la sortie. Notez que la colonne calculée résultante SALARY+300 n'est pas une nouvelle colonne de la table EMPLOYEES ; elle est destinée uniquement à l'affichage.

12.2. Priorités des opérateurs

L'exemple ci-dessous affiche le nom, le salaire et la rémunération annuelle des employés. Il calcule la rémunération annuelle en multipliant le salaire mensuel par 12 et en ajoutant un complément unique de 2000 dh. Notez que la multiplication est effectuée avant l'addition.

SELECT last_name, salary, 12*salary+2000
FROM employees;

Le deuxième exemple ci-dessous affiche le nom, le salaire et la rémunération annuelle des employés. Il calcule la rémunération annuelle de la façon suivante ; ajout d'un complément de 2000 dh au salaire mensuel, puis multiplication par 12. En raison des **parenthèses**, l'addition est effectuée avant la multiplication.

SELECT last_name, salary, 12*(salary+2000)
FROM employees;

12.3. Définir une valeur NULL

Si une valeur est **absente** d'une ligne pour une colonne particulière, cette valeur est dite *NULL*, ou contenant une valeur NULL. Une valeur NULL est différente d'un zéro ou d'un espace. Zéro est un chiffre et un espace est un caractère.

Exemple : **SELECT last_name, job_id, salary, commission_pct**
FROM employees;

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
...			
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2

12.4. Valeurs NULL dans les expressions arithmétiques

Si une valeur quelconque d'une expression arithmétique est NULL, le résultat est NULL.

Exemple : **SELECT last_name, salary+salary*commission_pct**
FROM employees;

LAST_NAME	JOB_ID	SALARY	SALARY+SALARY'COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
...			
Zlotkey	SA_MAN	10500	**12600**
Abel	SA_REP	11000	**14300**
Taylor	SA_REP	8600	**10320**

12.5. Fonction NVL

Pour convertir une valeur NULL en valeur réelle, utilisez la fonction NVL.

Syntaxe : **NVL (*expr1*, *expr2*)**

Dans la syntaxe :

- *expr1* est la valeur ou l'expression source pouvant contenir une valeur NULL.
- *expr2 est la valeur cible pour la conversion de la valeur NULL.*

NVL(commission_pct,0) permet de résoudre le problème de l'exemple précédent.
La requête de l'exemple précédent peut s'écrire :

SELECT last_name, salary+salary*NVL(commission_pct,0)
FROM employees;

LAST_NAME	JOB_ID	SALARY	SALARY+SALARY'COMMISSION_PCT
King	AD_PRES	24000	**24000**
Kochhar	AD_VP	17000	**17000**
...			
Zlotkey	SA_MAN	10500	**12600**
Abel	SA_REP	11000	**14300**
Taylor	SA_REP	8600	**10320**

12.6. Lignes en double

Par défaut, les interrogations SQL renvoient toutes les lignes, y compris les lignes en double.

Exemple 1: **SELECT department_id**
FROM employees;

DEPARTMENT_ID
90
90
90
...

20 rows selected.

La requête de l'exemple1 affiche tous les numéros de département de la table EMPLOYEES. Notez que les numéros de département sont répétés.

Exemple 2 :
```
SELECT DISTINCT department_id
FROM   employees;
```
Pour éliminer les lignes en double dans le résultat, incluez le mot-clé DISTINCT dans la clause SELECT, immédiatement après le mot-clé SELECT. Dans ce deuxième exemple, la table EMPLOYEES contient en fait 20 lignes, mais elle contient seulement sept numéros de département uniques.

12.7. Restreindre les lignes renvoyées à l'aide de la clause WHERE

Dans l'exemple suivant, supposons que vous souhaitiez afficher tous les employés du département 90. Les lignes dont la colonne DEPARTMENT_ID présente la valeur 90 sont les seules lignes renvoyées. Cette méthode de restriction est la base de la clause **WHERE** en langage SQL.

Syntaxe :
```
SELECT *|{[DISTINCT] column|expression,...}
FROM   table
[WHERE condition(s)];
```
Exemple :
```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

12.8. Conditions de comparaison

Le tableau suivant présente la liste des opérateurs de comparaison et leurs significations.

Opérateur	Signification
=	Egal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Non égal à

Opérateur	Signification
BETWEEN ...AND...	Entre deux valeurs (incluses)
IN(set)	Correspond à une valeur quelconque d'une liste
IS NULL	Est une valeur NULL

12.9. Jointure

Il arrive parfois que vous deviez utiliser des **données provenant de plusieurs tables**. Dans l'exemple ci-dessous, l'état affiche des données de deux tables distinctes :

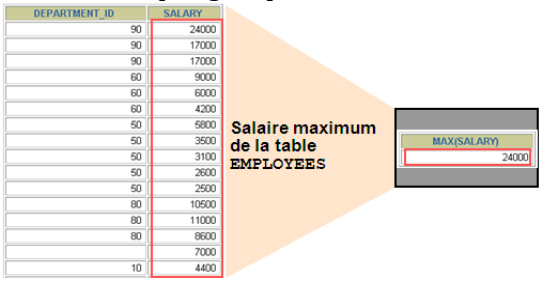
- Les ID d'employé sont présents dans la table EMPLOYEES.
- Les ID de département sont présents dans les tables EMPLOYEES et DEPARTMENTS.
- Les noms de département sont présents dans la table DEPARTMENTS.

Pour générer l'état, vous devez lier les tables EMPLOYEES et DEPARTMENTS, puis accéder aux données des deux.

Exemple :
```
SELECT employees.employee_id, department_id, departments.department_name
FROM   employees JOIN  departments
USING (department_id);
```

13. Utilisation des fonctions de groupe

Contrairement aux fonctions mono ligne, les fonctions de **groupe** opèrent sur des ensembles de lignes afin de renvoyer un seul résultat par groupe.



Exemple :

```
SELECT AVG(salary), MAX(salary), MIN(salary), SUM(salary)
FROM   employees
Where department_id=90;
```

Vous pouvez utiliser les fonctions **AVG**, **SUM**, **MIN**, **MAX, COUNT (Nombre de ligne) et STDDEV (Ecart type)** sur les colonnes permettant le stockage de données numériques. L'exemple ci-dessus affiche la moyenne, la valeur la plus élevée, la valeur la plus faible et la somme des salaires mensuels de tous les employés du département 90.

13.1. Fonction COUNT

COUNT(*) renvoie le nombre de lignes d'une table :

```
SELECT COUNT(*)
FROM   employees
WHERE  department_id = 50;
```

COUNT(expr) renvoie le nombre de lignes avec des valeurs non NULL pour expr (commission_pct dans ce cas) :

```
SELECT COUNT(commission_pct)
FROM   employees
WHERE  department_id = 80;
```

COUNT(DISTINCT expr) renvoie le nombre de valeurs non NULL distinctes de *expr*. Pour afficher le nombre de départements distincts de la table EMPLOYEES :

```
SELECT COUNT(DISTINCT department_id)
FROM   employees;
```

13.2. Fonctions de groupe et valeurs NULL

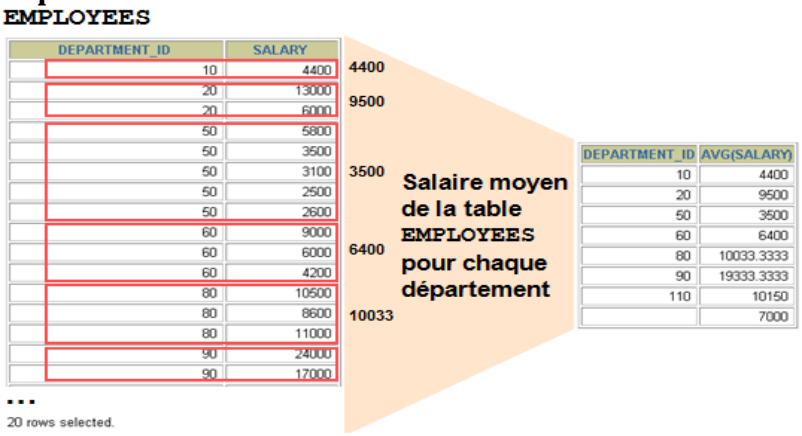
Les fonctions de groupe ignorent les valeurs NULL de la colonne :

```
SELECT AVG(commission_pct)
FROM   employees;
```

La fonction NVL force les fonctions de groupe à inclure les valeurs NULL :

```
SELECT AVG(NVL(commission_pct, 0))
FROM   employees;
```

13.3. Groupes de données - GROUP BY



Vous pouvez diviser les lignes d'une table en groupes plus petits à l'aide de la clause GROUP BY

Syntaxe :

```
SELECT   column, group_function
FROM     table
[WHERE   condition]
[GROUP BY group_by_expression]
[HAVING  group_condition]
[ORDER BY column];
```

Toutes les colonnes de la liste SELECT qui ne sont pas incluses dans des fonctions de groupe doivent figurer dans la clause GROUP BY.

```
SELECT   department_id, AVG(salary)
FROM     employees
GROUP BY department_id ;
```

L'exemple ci-dessus permet d'afficher le salaire moyen par departement_id.

13.4. GROUP BY sur plusieurs colonnes

Cette requête affiche la somme des salaires par département et par fonction.

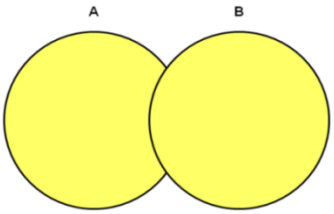
```
SELECT   department_id dept_id, job_id, SUM(salary)
FROM     employees
GROUP BY department_id, job_id ;
```

14. Utilisation des opérateurs ensemblistes

Les opérateurs ensemblistes combinent les résultats de plusieurs interrogations en un résultat unique. Les interrogations contenant des opérateurs ensemblistes sont appelées **interrogations composées**. Les tables utilisées dans ce paragraphe sont les suivantes :

- EMPLOYEES : contient les informations relatives à tous les employés actuels
- JOB_HISTORY : contient la date de début et la date de fin dans le poste précédent, ainsi que le numéro d'identification du poste et le département, lorsqu'un employé change de poste.

14.1. Opérateur UNION



L'opérateur UNION renvoie les résultats des deux interrogations après avoir éliminé les doublons.

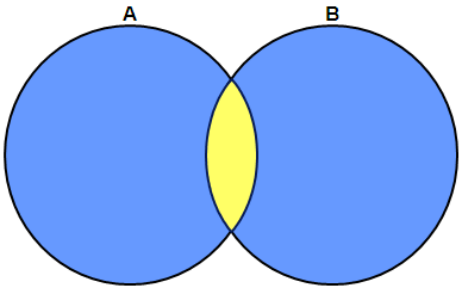
Exemple : Affichez les informations relatives au poste actuel et au poste précédent de tous les employés. Affichez chaque employé une seule fois.

```
SELECT employee_id, job_id
FROM   employees
UNION
SELECT employee_id, job_id
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID
100	AD_PRES
101	AC_ACCOUNT
200	AC_ACCOUNT
200	AD_ASST
205	AC_MGR
206	AC_ACCOUNT

L'opérateur UNION élimine tous les enregistrements en double. Si des enregistrements présents dans les tables EMPLOYEES et JOB_HISTORY sont identiques, ils ne sont affichés qu'une seule fois. Dans le résultat affiché ci-dessus, notez que l'enregistrement de l'employé dont la valeur EMPLOYEE_ID est 200 apparaît deux fois, car la valeur JOB_ID est différente dans chaque ligne.

14.2. Opérateur INTERSECT



L'opérateur INTERSECT renvoie les lignes qui sont communes aux deux interrogations.

Exemple : Affichez l'ID d'employé et l'ID de poste des employés dont le poste actuel est le même que celui qu'ils occupaient lors de leur embauche (c'est-à-dire ceux qui ont changé de poste, mais qui occupent aujourd'hui le même poste qu'à l'origine).

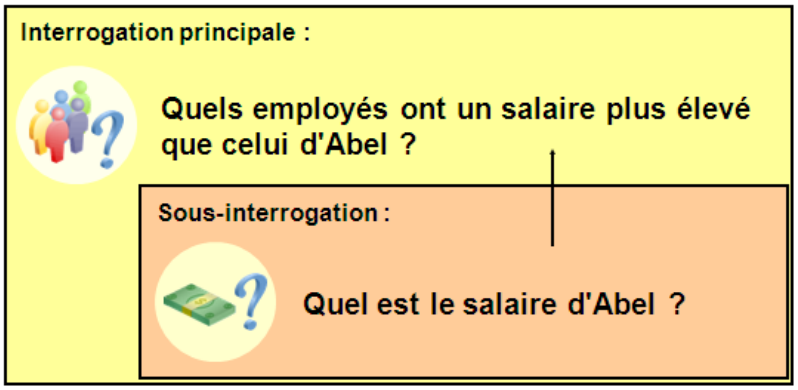
```
SELECT employee_id, job_id
FROM   employees
INTERSECT
SELECT employee_id, job_id
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST

Dans l'exemple ci-dessus, l'interrogation renvoie uniquement les enregistrements qui présentent les mêmes valeurs dans les colonnes sélectionnées des deux tables.

15. Utilisation des sous-interrogations

Supposons que vous souhaitiez écrire une interrogation permettant de déterminer les employés dont le salaire est plus élevé que celui d'Abel. Pour résoudre ce problème, vous avez besoin de deux interrogations : l'une pour déterminer combien gagne Abel, l'autre pour déterminer qui gagne plus que ce montant.



Vous pouvez résoudre ce problème en combinant les deux interrogations, c'est-à-dire en plaçant une interrogation à l'intérieur de l'autre.
L'interrogation interne (ou sous-interrogation) renvoie une valeur utilisée par l'interrogation externe (ou interrogation principale). L'utilisation d'une sous-interrogation revient à exécuter deux interrogations à la suite et à utiliser le résultat de la première interrogation comme valeur de recherche dans la deuxième.

Syntaxe :

```
SELECT    select_list
FROM  table
WHERE     expr operator
(SELECT   select_list
          FROM table);
```

Exemple:

```
SELECT last_name
FROM   employees
WHERE  salary >
                (SELECT salary
                 FROM   employees
                 WHERE  last_name = 'Abel');
```

11000

LAST_NAME
King
Kochhar
De Haan
Hartstein
Higgins

16. Langage de manipulation de données LMD

Le langage de manipulation de données (LMD) est une composante essentielle du langage SQL. Lorsque vous souhaitez ajouter, mettre à jour ou supprimer des données dans la base, vous exécutez une instruction LMD.

16.1. Ajouter une nouvelle ligne à une table - INSERT

Ajouter de nouvelles lignes à une table à l'aide de l'instruction INSERT.

Syntaxe :

```
INSERT INTO  table [(column [, column...])]
VALUES               (value [, value...]);
```

Exemple :

```
INSERT INTO departments(department_id,
 department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
```

Cette requête SQL permet d'ajouter un nouveau département appelé Public Relations. Ce département est identifié par 70 et géré par l'employée 100. Incluez les valeurs de type caractère et de type date entre apostrophes.

16.2. Mettre à jour une ligne existante - UPDATE

Syntaxe :

```
UPDATE          table
SET       column = value [, column = value, ...]
[WHERE          condition];
```

Exemple :

```
UPDATE employees
SET   department_id = 70
WHERE  employee_id = 113;
```

L'exemple ci-dessus transfère l'employé 113 vers le département 70.

16.3. Supprimer une ligne d'une table - DELETE

Syntaxe :

```
DELETE [FROM]     table
[WHERE       condition];
```

Exemple :

```
DELETE FROM departments
 WHERE  department_name = 'Finance';
```

L'exemple de la requête ci-dessus supprime le département Finance de la table DEPARTMENTS.

17. Langage de définition de données LDD

Le langage de définition de données (LDD) permet aux utilisateurs et administrateurs de base de données (DBA) de créer et gérer des objets de la base de données : table, vue, séquence, index, synonyme…

17.1. Objets de base de données

Une base de données Oracle peut contenir différentes structures de données. Chaque structure doit être décrite lors de la conception de la base de données, ce qui permet sa création au cours de la phase de développement de la base.

- **Table :** contient des données.
- **Vue :** sous-ensemble de données d'une ou plusieurs tables.
- **Séquence :** génère des valeurs numériques.
- **Index :** améliore les performances de certaines interrogations.
- **Synonyme :** affecte d'autres noms aux objets.

Remarque :

- Utilisez des noms descriptifs pour les tables et les autres objets de base de données.
- Les noms ne distinguent pas les majuscules des minuscules. Par exemple, le nom EMPLOYEES est considéré comme identique à eMPloyees ou eMpLOYEES.

17.2. Instruction CREATE TABLE

Vous pouvez créer des tables pour le stockage de données en exécutant l'instruction SQL CREATE TABLE. Cette instruction est l'une des instructions LDD, qui constituent un sous ensemble des instructions SQL utilisées pour créer, modifier ou supprimer des structures de base de données Oracle.

Pour créer une table, un utilisateur doit posséder le privilège CREATE TABLE et disposer d'une zone de stockage dans laquelle créer des objets. L'administrateur de base de données utilise les instructions du langage de contrôle de données (LCD) pour accorder des privilèges aux utilisateurs.

Dans la syntaxe : **CREATE TABLE [schema.]table (column datatype [DEFAULT expr][, ...]);**

schema	est identique au nom du propriétaire
table	est le nom de la table
DEFAULT *expr*	indique une valeur par défaut si une valeur est omise dans l'instruction INSERT
column	est le nom de la colonne
datatype	est le type de données et la longueur de la colonne

17.3. Option DEFAULT

- Indiquez une valeur par défaut pour une colonne lors d'une insertion.

  ... hire_date DATE DEFAULT SYSDATE, ...

- Le type de données par défaut doit correspondre au type de données de la colonne.

```
CREATE TABLE hire_dates
(id       NUMBER(8),
 hire_date DATE DEFAULT SYSDATE);
```

  Table created.

17.4. Créer des tables

Créez la table.

```
CREATE TABLE dept
(deptno     NUMBER(2),
 dname      VARCHAR2(14),
loc       VARCHAR2(13),
create_date DATE DEFAULT SYSDATE);
```

Table created.

Vérifiez la création de la table.

```
DESCRIBE dept ;
```

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)
CREATE_DATE		DATE

17.5. Types de données

Les types de données sont listés dans le tableau ci-dessous.

Type de données	Description
VARCHAR2(*size*)	Données de type caractère de longueur variable
CHAR(*size*)	Données de type caractère de longueur fixe

NUMBER(*p,s)*	Données numériques de longueur variable
DATE	Valeurs de date et d'heure
LONG	Données de type caractère de longueur variable (jusqu'à 2 Go)
CLOB	Données de type caractère (jusqu'à 4 Go)
RAW et LONG RAW	Données binaires raw
BLOB	Données binaires (jusqu'à 4 Go)
BFILE	Données binaires stockées dans un fichier externe (jusqu'à 4 Go)
ROWID	Système de numérotation en base 64, représentant l'adresse unique d'une ligne dans sa table

17.6. Contraintes d'intégrité des données

Le serveur Oracle utilise des contraintes pour empêcher l'insertion de données non valides dans les tables. Vous pouvez utiliser des contraintes pour effectuer les opérations suivantes :

- Appliquer des règles aux données d'une table chaque fois qu'une ligne est insérée, mise à jour ou supprimée dans cette table. La contrainte doit être satisfaite pour que l'opération réussisse.
- Empêcher la suppression d'une table s'il existe des dépendances par rapport à d'autres tables.
- Les types de contrainte suivants sont pris en charge :

  - **NOT NULL**
  - **UNIQUE**
  - **PRIMARY KEY**
  - **FOREIGN KEY**
  - **CHECK**

17.7. Règles relatives aux contraintes

- Vous pouvez nommer une contrainte ou le SGBDR génère un nom.
- Vous pouvez créer une contrainte à différents instants :
  - Lors de la création de la table ;
  - Après la création de la table.

- Définissez une contrainte au niveau colonne ou au niveau table.
- Vous pouvez afficher une contrainte via le dictionnaire de données.

17.8. Définir des contraintes

La syntaxe ci-dessous permettant de définir des contraintes lors de la création d'une table. Vous pouvez créer les contraintes au niveau colonne ou au niveau table. Les contraintes définies au niveau colonne sont incluses lors de la définition de la colonne. Les contraintes au niveau table sont définies à la fin de la définition de la table et doivent faire référence aux colonnes auxquelles s'applique la contrainte, entre parenthèses.

- **Syntaxe :**

```
CREATE TABLE [schema.]table
      (column datatype [DEFAULT expr]
      [column_constraint],
      ...
      [table_constraint][,...]);
```

- **Contrainte au niveau colonne :**

```
column [CONSTRAINT constraint_name] constraint_type,
```

- **Contrainte au niveau table :**

```
column,...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

Remarque :

- Les contraintes NOT NULL doivent être définies au niveau colonne.
- Les contraintes qui s'appliquent à plusieurs colonnes doivent être définies au niveau table.

Les deux exemples ci-dessous créent une contrainte de clé primaire sur la colonne EMPLOYEE_ID de la table EMPLOYEES.

- Le premier exemple utilise la syntaxe au niveau colonne pour définir la contrainte.

```
CREATE TABLE employees(
 employee_id  NUMBER(6)
  CONSTRAINT emp_emp_id_pk PRIMARY KEY,
 first_name   VARCHAR2(20),
 ...);
```

- Le deuxième exemple utilise la syntaxe au niveau table pour définir la contrainte.

```
CREATE TABLE employees(
 employee_id  NUMBER(6),
 first_name   VARCHAR2(20),
 ...
 job_id       VARCHAR2(10) NOT NULL,
 CONSTRAINT emp_emp_id_pk
  PRIMARY KEY (EMPLOYEE_ID));
```

17.9. Contrainte NOT NULL

La contrainte NOT NULL permet de garantir que la colonne ne contient pas de valeurs NULL. Les colonnes sans la contrainte NOT NULL peuvent contenir des valeurs NULL par défaut. Les contraintes NOT NULL doivent être définies au niveau colonne.

Garantit que les valeurs NULL ne sont pas autorisées pour la colonne :

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
100	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000	90
101	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000	90
102	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	90
103	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000	60
104	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000	60
178	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000	
200	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400	10

...

20 rows selected.

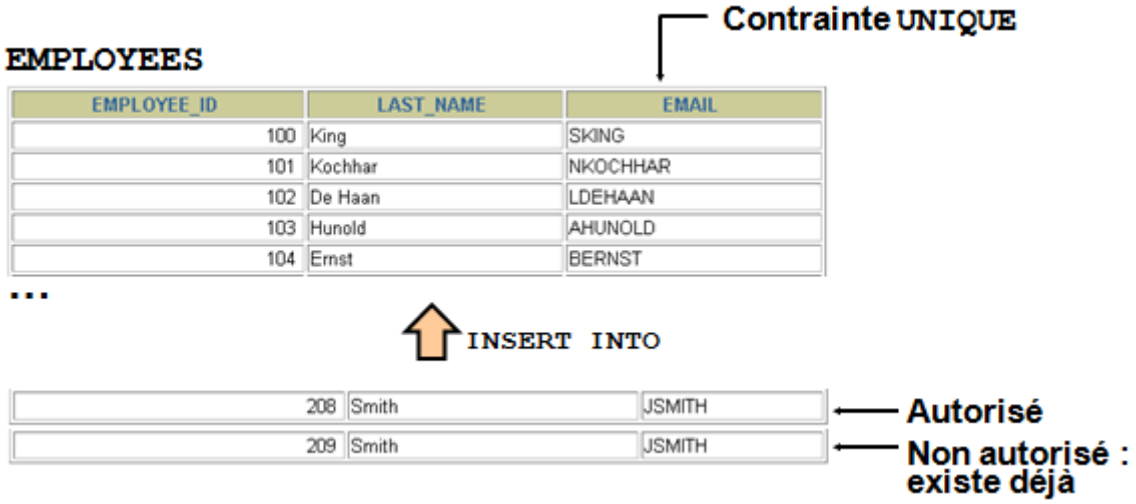
Contrainte NOT NULL (aucune ligne ne peut contenir de valeur NULL pour cette colonne)

Contrainte NOT NULL

Absence de contrainte NOT NULL (n'importe quelle ligne peut contenir une valeur NULL pour cette colonne)

17.10. Contrainte UNIQUE

Une contrainte d'intégrité de clé UNIQUE impose que chaque valeur d'une colonne ou d'un ensemble de colonnes (clé) soit unique, c'est-à-dire que deux lignes d'une table ne doivent pas comporter de valeurs en double dans une colonne ou un ensemble de colonnes. La colonne (ou l'ensemble de colonnes) incluse dans la définition de la contrainte de clé UNIQUE est appelée clé unique. Si la contrainte UNIQUE est constituée de plusieurs colonnes, ce groupe de colonnes est appelé clé unique composée.

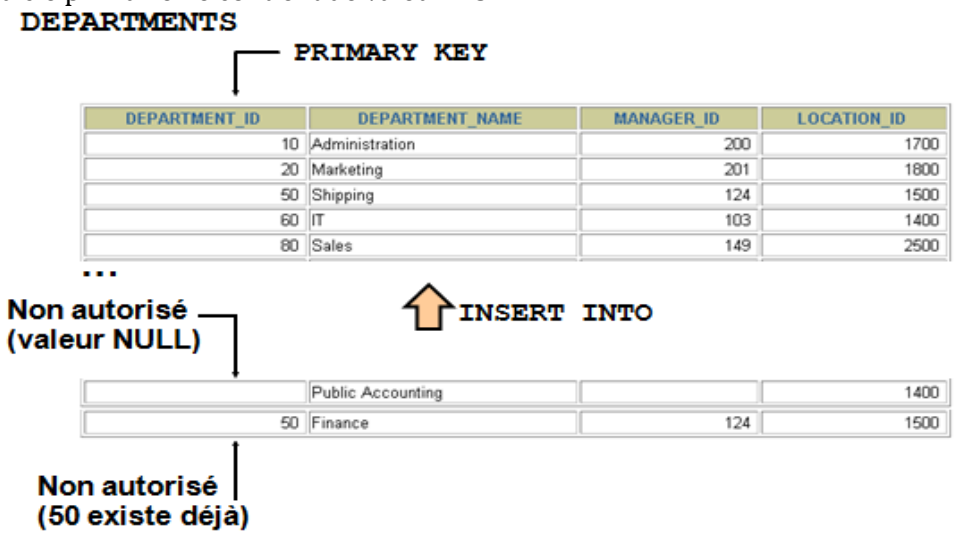


La synatxe :

```
CREATE TABLE employees(
  employee_id     NUMBER(6),
  last_name       VARCHAR2(25) NOT NULL,
  email           VARCHAR2(25),
  salary          NUMBER(8,2),
  commission_pct  NUMBER(2,2),
  hire_date       DATE NOT NULL,
...
  CONSTRAINT emp_email_uk UNIQUE(email));
```

17.11. Contrainte PRIMARY KEY

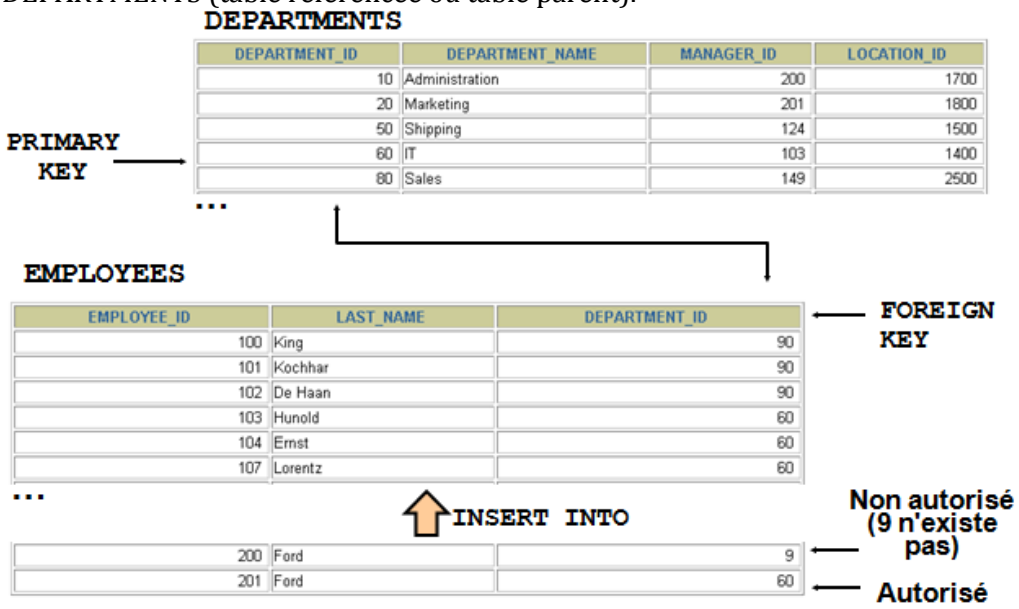
Une contrainte PRIMARY KEY crée une clé primaire pour la table. Une seule clé primaire peut être créée pour chaque table. La contrainte PRIMARY KEY est une colonne ou un ensemble de colonnes qui identifie de manière unique chaque ligne d'une table. Cette contrainte impose l'unicité de la colonne ou la combinaison de colonnes et garantit qu'aucune colonne faisant partie de la clé primaire ne contient de valeur NULL.



17.12. Contrainte FOREIGN KEY

La contrainte FOREIGN KEY (ou contrainte d'intégrité référentielle) désigne une colonne ou une combinaison de colonnes comme clé étrangère et établit une relation entre une clé primaire et une clé unique dans la même table ou dans une table différente.

Dans l'exemple ci-dessous, DEPARTMENT_ID a été défini comme clé étrangère dans la table EMPLOYEES (table dépendante ou table enfant) ; elle référence la colonne DEPARTMENT_ID de la table DEPARTMENTS (table référencée ou table parent).



La syntaxe:

```
CREATE TABLE employees(
   employee_id     NUMBER(6),
   last_name       VARCHAR2(25) NOT NULL,
   email           VARCHAR2(25),
   salary          NUMBER(8,2),
   commission_pct  NUMBER(2,2),
   hire_date       DATE NOT NULL,
...
   department_id   NUMBER(4),
   CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
    REFERENCES departments(department_id),
   CONSTRAINT emp_email_uk UNIQUE(email));
```

La clé étrangère est définie dans la table enfant, et la table contenant la colonne référencée est la table parent. La clé étrangère est définie à l'aide d'une combinaison des mots-clés suivants :

- Le mot-clé FOREIGN KEY est utilisé pour définir la colonne dans la table enfant, au niveau contrainte de table.
- Le mot-clé REFERENCES identifie la table et la colonne dans la table parent.
- Le mot-clé ON DELETE CASCADE indique que lorsque la ligne de la table parent est supprimée, les lignes dépendantes de la table enfant sont également supprimées.
- Le mot-clé ON DELETE SET NULL convertit les valeurs des clés étrangères en valeurs NULL lorsque la valeur parent est supprimée.

Le comportement par défaut est appelé *règle de restriction* et désactive la mise à jour ou la suppression des données référencées.

Sans les options ON DELETE CASCADE ou ON DELETE SET NULL, la ligne de la table parent ne peut pas être supprimée si elle est référencée dans la table enfant.

17.13. Contrainte CHECK

La contrainte CHECK définit une condition à laquelle chaque ligne doit satisfaire.

Exemple : **..., salary NUMBER(8,2)**
CONSTRAINT emp_salary_min
CHECK (salary > 0),...

17.14. Création de table employees

L'exemple de la diapositive ci-dessus illustre l'instruction permettant de créer la table EMPLOYEES dans le schéma HR :

```
CREATE TABLE employees
  ( employee_id    NUMBER(6)
     CONSTRAINT    emp_employee_id   PRIMARY KEY,
    first_name     VARCHAR2(20),
    last_name      VARCHAR2(25)
     CONSTRAINT    emp_last_name_nn  NOT NULL,
    email          VARCHAR2(25)
     CONSTRAINT    emp_email_nn      NOT NULL
     CONSTRAINT    emp_email_uk      UNIQUE,
    phone_number   VARCHAR2(20),
    hire_date      DATE
     CONSTRAINT    emp_hire_date_nn  NOT NULL,
    job_id         VARCHAR2(10)
     CONSTRAINT    emp_job_nn        NOT NULL,
    salary         NUMBER(8,2)
     CONSTRAINT    emp_salary_ck     CHECK (salary>0),
    commission_pct NUMBER(2,2),
    manager_id     NUMBER(6),
    department_id  NUMBER(4)
     CONSTRAINT    emp_dept_fk       REFERENCES
       departments (department_id));
```

18. Création de formulaires et sous formulaires

Dans cette partie relative aux formulaires et état, nous utilisons Access comme SGBDR.
Les formulaires dans Access sont comme les vitrines des magasins, il est facile de voir ou de prendre les éléments que vous convoitez. Comme les formulaires sont des objets par lesquels vous-même ou d'autres utilisateurs pouvaient ajouter, modifier ou afficher les données stockées dans votre base de données Access de bureau, la conception du formulaire revêt une importance particulière. Si votre base de données Access de bureau doit être utilisée par plusieurs utilisateurs, des formulaires bien conçus sont essentiels à l'efficacité et la précision de la saisie de données.
Il existe plusieurs moyens de créer un formulaire dans une base de données de bureau Access.

19.1. Créer un formulaire à partir d'une table ou requête existante dans Access

Dans le volet de navigation, cliquez sur la table ou la requête contenant les données pour le formulaire, puis, sous l'onglet **Créer**, cliquez sur **Formulaire**.
Access crée un formulaire et l'affiche en mode Page. Vous pouvez apporter des modifications de conception, comme ajuster la taille des zones de texte en fonction des données à afficher, si nécessaire.

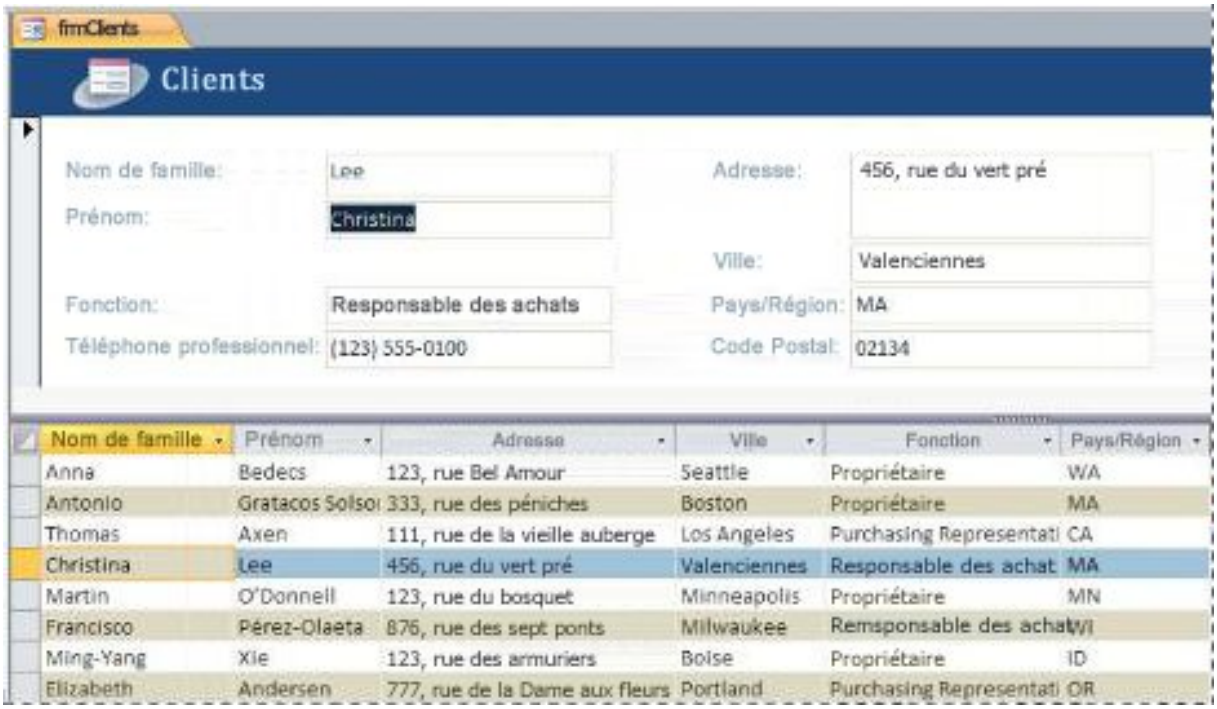
19.2. Créer un formulaire vierge dans Access

- Pour créer un formulaire sans contrôles ni éléments déjà mis en forme : sous l'onglet **Créer** tab, click **Formulaire vierge**. Access ouvre un formulaire vide en mode Page et affiche le volet **Liste de champs**.
- Dans le volet **Liste de champs**, cliquez sur le signe plus (**+**) en regard de la ou des tables contenant les champs que vous souhaitez faire figurer sur le formulaire.

- Pour ajouter un champ au formulaire, double-cliquez dessus ou faites-le glisser vers le formulaire. Pour ajouter plusieurs champs à la fois, maintenez la touche Ctrl enfoncée, cliquez sur plusieurs champs, puis faites-les glisser simultanément vers le formulaire.

19.3. Créer un formulaire double affichage dans Access

Le formulaire double affichage offre deux affichages simultanés des données : un affichage Formulaire et un affichage Feuille de données. L'utilisation de formulaires double affichage offre les avantages de deux types de formulaires dans un seul et même formulaire. Par exemple, vous pouvez utiliser la partie feuille de données du formulaire pour retrouver rapidement un enregistrement, et utiliser ensuite la partie formulaire pour consulter ou modifier l'enregistrement. Ces deux affichages sont rattachés à la même source de données et sont toujours synchronisés.



Pour créer un formulaire double affichage à l'aide de l'outil du même nom, dans le volet de navigation, cliquez sur la table ou la requête contenant les données, puis sous l'onglet **Création**, cliquez sur **Plus de formulaires**, puis cliquez sur **Formulaire double affichage**.

19.4. Créer un formulaire qui affiche plusieurs enregistrements dans Access

Un formulaire à plusieurs éléments, appelé également formulaire continu, peut s'avérer utile si vous voulez un formulaire qui affiche plusieurs enregistrements mais qui soit plus personnalisable qu'une feuille de données. Dans ce cas, utilisez l'outil Plusieurs éléments.

- Dans le volet de navigation, cliquez sur la table ou la requête qui contient les données que vous souhaitez faire figurer dans le formulaire.
- Sous l'onglet **Création**, cliquez sur **Plus de formulaires**> **Plusieurs éléments**.

19.5. Créer un formulaire contenant un sous-formulaire dans Access

Lorsque vous travaillez avec des données associées qui sont stockées dans des tables distinctes, vous devez souvent afficher les données à partir de plusieurs tables ou

requêtes sur le même formulaire. Les sous-formulaires sont un moyen pratique d'y parvenir.

19.6. Créer un formulaire de navigation dans Access

Un formulaire de navigation est tout simplement un formulaire contenant un contrôle de navigation. Les formulaires de navigation sont un vrai plus pour tous les types de bases de données, mais ils revêtent une importance particulière si vous envisagez de publier une base de données sur le Web, car le volet de navigation d'Access ne s'affiche pas dans un navigateur.

- Ouvrez la base de données à laquelle vous souhaitez ajouter un formulaire de navigation.
- Sous l'onglet **Création**, dans le groupe **Formulaires**, cliquez sur **Navigation**, puis sélectionnez le style de formulaire de navigation voulu.

19. Création des états et sous états

Les états offrent un moyen d'afficher, de mettre en forme et de synthétiser les informations de votre base de données Microsoft Access. Vous pouvez par exemple créer un simple état de numéros de téléphone de tous vos contacts ou un état récapitulatif du total des ventes dans différentes régions à des périodes différentes.
Un état est un objet de base de données qui s'avère utile quand vous voulez présenter les informations dans votre base de données dans l'un des cas suivants :

- Afficher ou distribuer une synthèse des données.
- Archiver des instantanés des données.
- Fournir des détails relatifs à des enregistrements individuels.
- Créer des étiquettes.

19.1. Créer un état dans Access

Vous pouvez créer des états pour votre base de données de bureau Access en respectant les étapes suivantes :
Étape 1 : choisir une source d'enregistrement
La source d'enregistrement d'un état peut être une table, une requête nommée ou une requête incorporée. Elle doit contenir toutes les lignes et les colonnes de données que vous souhaitez afficher dans l'état.
Étape 2 : choisir un outil d'état
Les outils d'état sont situés sous l'onglet **Créer**, dans le groupe **États**.
Étape 3 : créer l'état
Cliquez sur le bouton correspondant à l'outil que vous souhaitez utiliser. Si un Assistant s'affiche, suivez les étapes décrites dans celui-ci, puis cliquez sur **Terminer** dans la dernière page.
Access affiche l'état en mode Page.

19.2. Créer et utiliser des sous-états

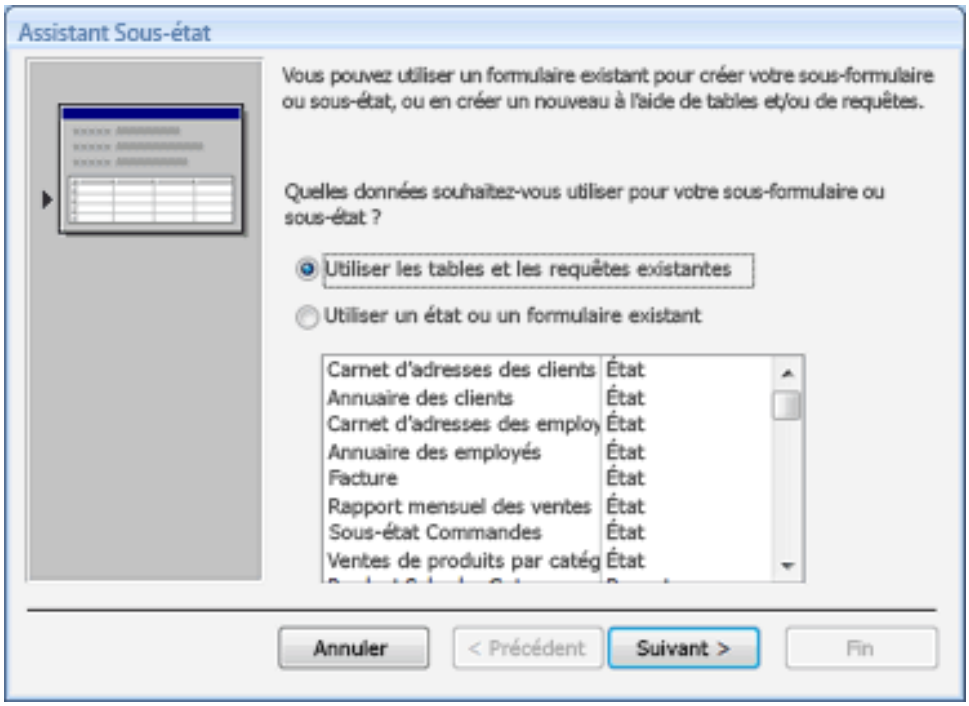
Lorsque vous travaillez avec des données relationnelles (où les données associées sont stockées dans des tables distinctes), vous devez souvent afficher les informations de plusieurs tables ou requêtes dans le rapport même. Par exemple, vous souhaitez afficher des données client, mais vous souhaitez également afficher des informations sur les commandes du client en même temps. Un sous-état est un outil utile pour ce faire, car il vous permet d'afficher les informations de commande de l'état avec les informations relatives aux clients, de manière logique et lisible.

Procédure de Création d'un sous-état

Si le sous-état doit être lié à l'état principal, vérifiez que les sources d'enregistrement sous-jacentes sont associées avant d'utiliser la procédure suivante.

Utilisez l'Assistant Sous-état pour créer un sous-état

1. Ouvrez l'état que vous souhaitez utiliser en tant que l'état principal dans mode Création.
2. Sous l'onglet **Création**, dans le groupe **contrôles**, vérifiez que **Utiliser les Assistants contrôle est sélectionné.**
3. Sous l'onglet **Création**, dans le groupe **contrôles**, cliquez sur **Sous-formulaire/sous-état.**
4. Dans l'état, cliquez où vous souhaitez placer le sous-état.
5. Dans la première page de l'Assistant sous-état, si vous voulez créer un nouveau sous-formulaire ou sous-état et baser sur une table ou requête, cliquez sur **utiliser les Tables et requêtes existantes**. S'il existe un rapport existant ou un formulaire que vous souhaitez utiliser comme sous-état, cliquez sur **utiliser un formulaire ou un rapport existant**, sélectionnez le rapport ou le formulaire dans la liste, puis sur **suivant**.



20. Informatique décisionnelle - Business Intelligence

L'informatique décisionnelle (BI) est l'ensemble des méthodes, moyens et outils informatiques utilisés pour piloter une entreprise et aider à la prise de décision.

Les environnements d'aide à la décision s'intéressent aux tendances, aux moyennes des principaux indicateurs de bonne santé de l'entreprise, et ce, à travers les mois ou les années.

Le monde décisionnel analyse des données agrégées pour meiux apprécier l'ensemble de l'activité de l'entreprise. L'aide à la prise de décision est la responsabilité de quelques personnes dans l'entreprise (décideurs). En effet, les décideurs s'intéressent à l'ensemble de l'activité et souhaitent avoir une vue globale de la société.

Dans le monde décisionnel, on ne supprime jamais des données, on archive. Les outils de BI permettent une exploitation rationnelle et coopérative des données (contrairement à

l'informatique de gestion ou l'on produit de l'information grâce aux bases de données relationnelles).

20.1. But de l'informatique décisionnelle

Le but de l'informatique décisionnelle est d'aider à la prise de décision et de permettre des analyses précises, complexes et de grandes envergures dans les entreprises.
Les systèmes décisionnels permettent de générer de la connaissance à partir des données produites par les systèmes opérationnels (bases de données relationnelles).
L'informatique décisionnelle permet de connaître les tendances des clients pour ainsi bien anticiper les réactions de ses derniers. Le BI permet donc de fidéliser les clients.

20.2. A qui s'adresse l'informatique décisionnelle ?

L'informatique décisionnelle ne s'adresse ni aux informaticiens ni aux statisticiens. En effet, elle s'adresse aux décideurs. Ces derniers utilisent les solutions de BI pour mieux comprendre le fonctionnement actuel de l'activité et d'anticiper des actions pour un pilotage éclairé de l'entreprise.

20.3. Analyse multi-démensionnelle

Depuis le début des années 1980 et avec une explosion au milieu des années 1990, les entreprises s'équipent de solutions de gestion et stockent un volume d'information qui s'agrandit un peu plus chaque jour. Rapidement apparait l'idée d'exploiter toutes ces données afin d'optimiser la décision. Naît ainsi l'informatique décisionnelle dont le but est de collecter, consolider, synthétiser l'information pour aider à la prise de décision. Partant de ce constat, le Docteur Edgar F. Cood publie en 1993, un papier intitulé **« Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate ». Les bases de données OLAP sont des bases de données multi-dimensionnelles** destinées à des analyses complexes sur des données. Les systèmes OLAP doivent :
1.Supporter les exigences complexes des décideurs en termes d'analyse,
2. Analyser les données à partir de différentes perspectives (dimensions métiers),
3. Supporter les analyses complexes impliquant des ensembles de données volumineux.

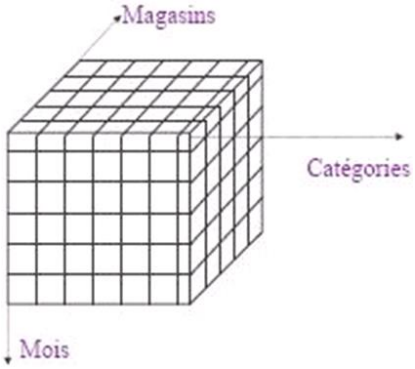
20.4. Méthodologie OLAP

Un cube OLAP est une représentation abstraite d'informations multidimensionnelles exclusivement numérique utilisé par l'approche OLAP (acronyme de On-line Analytical Processing). Cette structure est prévue à des fins d'analyses interactives par une ou plusieurs personnes du métier que ces données sont censées représenter.
Les cubes OLAP ont les caractéristiques suivantes :

1- Obtenir des informations déjà agrégées selon les besoins de l'utilisateur.
2-Simplicité et rapidité d'accès.
3- Capacité à manipuler les données agrégées selon différentes dimensions.
4- Un cube utilise les fonctions classiques d'agrégation : min, max, count, sum, avg, mais peut utiliser des fonctions d'agrégations spécifiques.

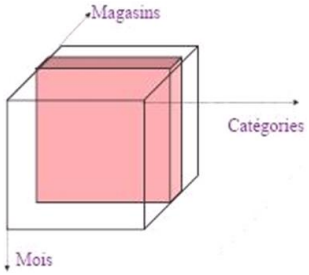
20.5. Requête OLAP

Voici un exemple de cube OLAP.
Dans notre exemple, nous allons nous intéresser aux ventes de tous les magasins "XXX".

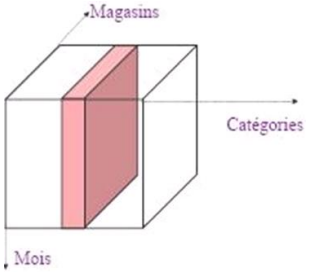


Nous allons nous intéresser aux différentes vues de ce cube.

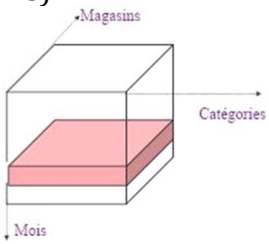
Vue n° 1 : On s'intéresse à toutes les ventes du magasin X1 (toutes catégories confondues durant toute l'année)



Vue n° 2 : On s'intéresse aux ventes de la catégorie "vêtements pour enfants" (tous les magasins durant toute l'année).



Vue n° 3 : On s'intéresse à toutes les ventes durant le mois de Février (toutes catégories confondues et dans tous les magasins).



Vue n° 4 : On s'intéresse aux ventes du magasin X1 dans la catégorie "vêtements pour enfants" durant le mois de Février.