Chapitre 5

Bases de données et langage SQL

Introduction

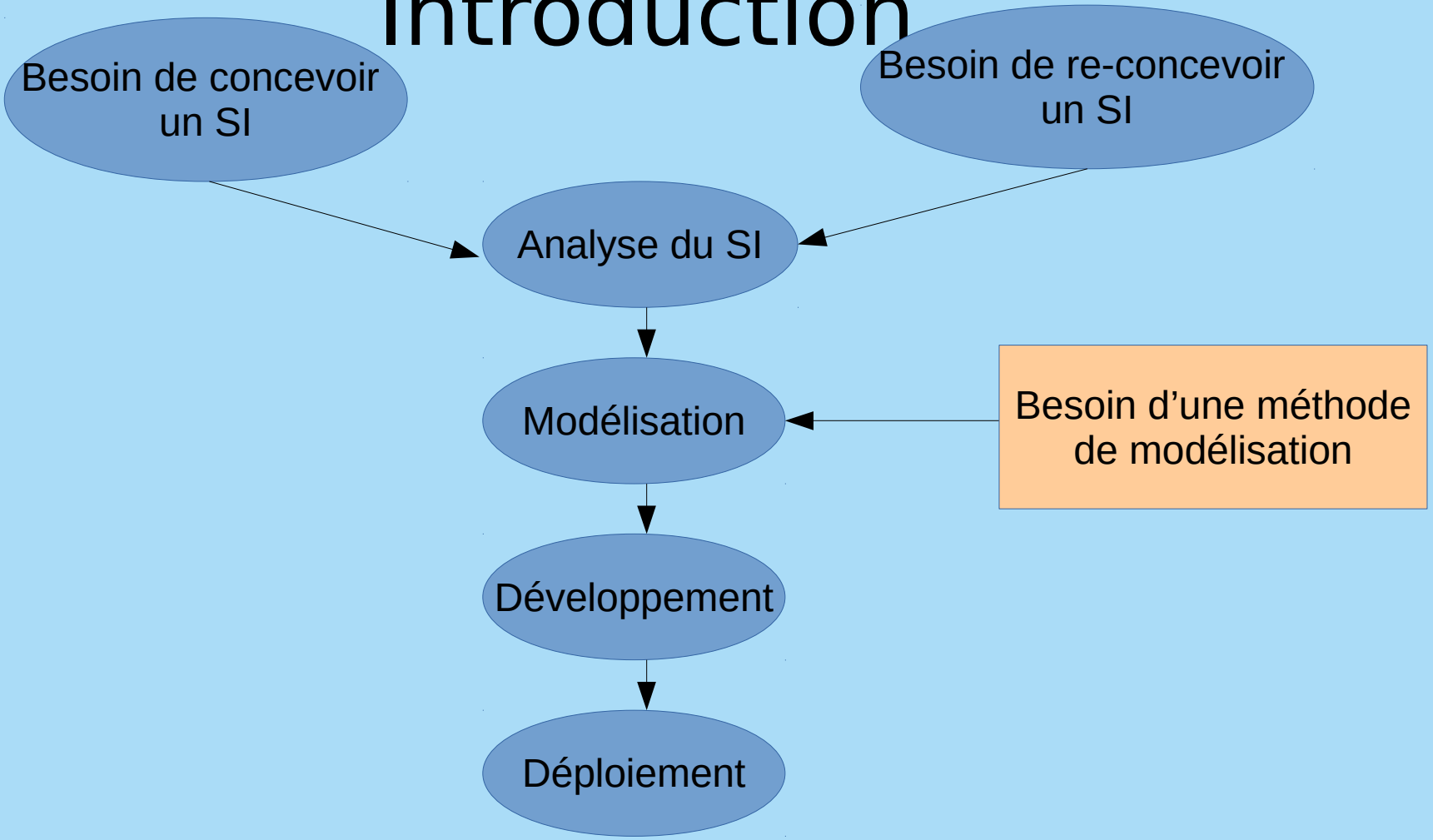
- Quasiment toutes les informations numérisées sont stockées sous forme de **base de données relationnelles***.

- Ces bases de données sont gérées par des logiciels **SGDB** ou Systèmes de Gestion de Bases de Données.

- Un langage commun à toutes les bases de données permet d'interagir avec ces logiciels : **le langage SQL**.

*La traduction de base de données : database.

Introduction



Plan

1) Les bases de données, le schéma relationnel
2) Normalisation des bases de données
3) SGBD
4) Langage SQL – la requête SELECT
5) Langage SQL – Autres requêtes
6) Aller plus loin

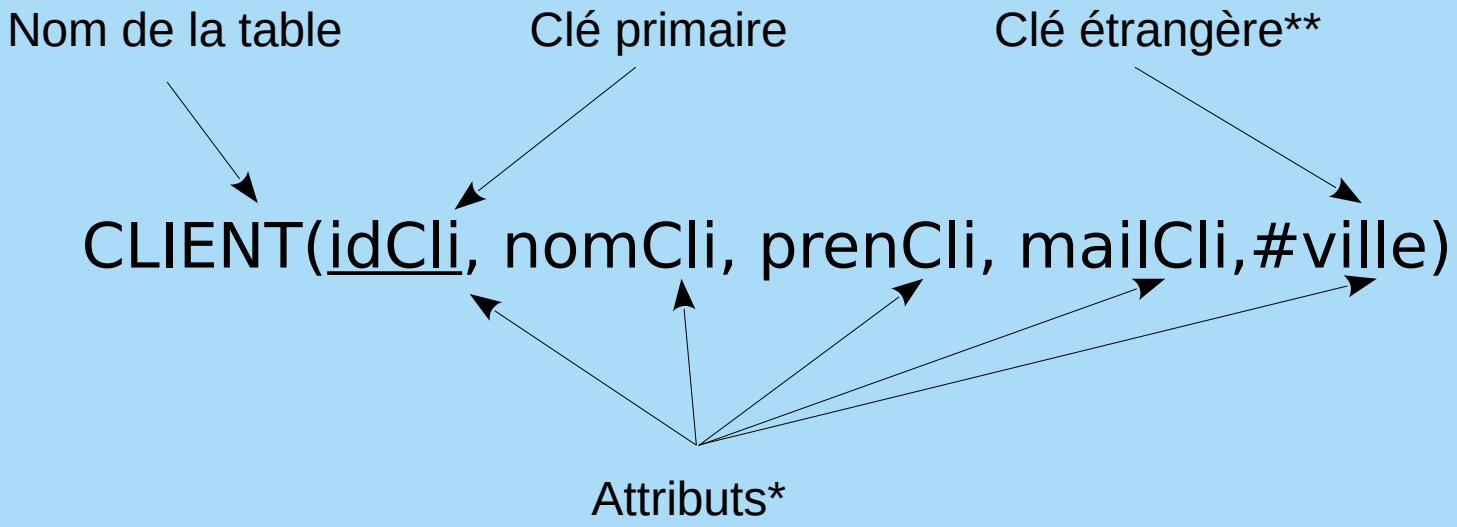
Plan

1) **Les bases de données, le schéma relationnel**
2) Normalisation des bases de données
3) SGBD
4) Langage SQL – la requête SELECT
5) Langage SQL – Autres requêtes
6) Aller plus loin

Schéma relationnel

- Un schéma relationnel est la description d’une base de données
- Dans un schéma relationnel, les informations sont organisées en **tables**
- Les tables sont aussi appelées des relations puisqu’elles mettent en relation des informations

Schéma relationnel et BD



* On parle aussi de propriété ou de champ.

** On parle aussi de clé secondaire

Schéma relationnel et BD

Une table dans un schéma relationnel symbolise un tableau avec l'information

CLIENT(idCli, nomCli, prenCli, mailCli,#ville)

idCli	nomCli	prenCli	mailCli	ville
124765	Knol	Denis	d.knol@gmail.com	123
126537	Parilly	Léa	lpar@hotmail.com	154
127645	Hutre	Sara	deleme@gmail .com	167
254366	Sanchez	Ajna	as876@yahoo.es	654

Schéma relationnel et BD

On travaillera avec la base suivante BD1 :

VILLE(idVille, nomVille)

idVille	nomVille
123	Paris
124	Lyon
167	Saint-Paul

CLIENT(idCli, nomCli, prenCli, mailCli,#ville)

idCli	nomCli	prenCli	mailCli	ville
124765	Knol	Denis	d.knol@gmail.com	123
126537	Parilly	Léa	lpar@hotmail.com	123
127645	Hutre	Sara	deleme@gmail .com	167
254366	Sanchez	Ajna	as876@yahoo.es	123

COMMANDE(numC, dateC, montant,#idCli)

numC	dateC	montant	idCli
201901	12/06/2019	1245,77	124765
201902	13/06/2019	3385,43	126537
201903	-	765,70	254366
201904	14/06/2019	56,87	124765

Schéma relationnel

- Pour chaque ligne d'une table, on parlera indifféremment d'occurrence, de tuple, de n-uplet, de réalisation ou de ligne.

- Dans la table précédente, on dira qu'il a 4 occurrences de la table Client (ie il y a 4 clients enregistrés).

Schéma relationnel

- Une clé primaire permet d’identifier une occurrence de manière unique.
- **Pour chaque occurrence, la clé primaire a une valeur unique.**
- Les clés primaires permettent de distinguer les différentes occurrences.

Schéma relationnel

- CLIENT(nomCli, prenCli, mailCli,#ville) : où est le problème ?

- CLIENT(nomCli, prenCli, mailCli,#ville) : où est le problème ?

  => on crée un identifiant indépendant idCli : CLIENT(idCli, nomCli, prenCli, mailCli,#ville)

Schéma relationnel

- Des attributs qui sont utilisés comme clés primaires dans les systèmes réels ?

- Plusieurs attributs peuvent constituer la clé primaire.

Schéma relationnel

- L’enregistrement des informations se fait par le jeu des clés. On peut retrouver de l’information au travers des clés.

- Pour chaque clé secondaire, elle fait référence à une clé primaire ... mais parfois l’identification est difficile

Schéma relationnel

A partir de la base de données utilisées :

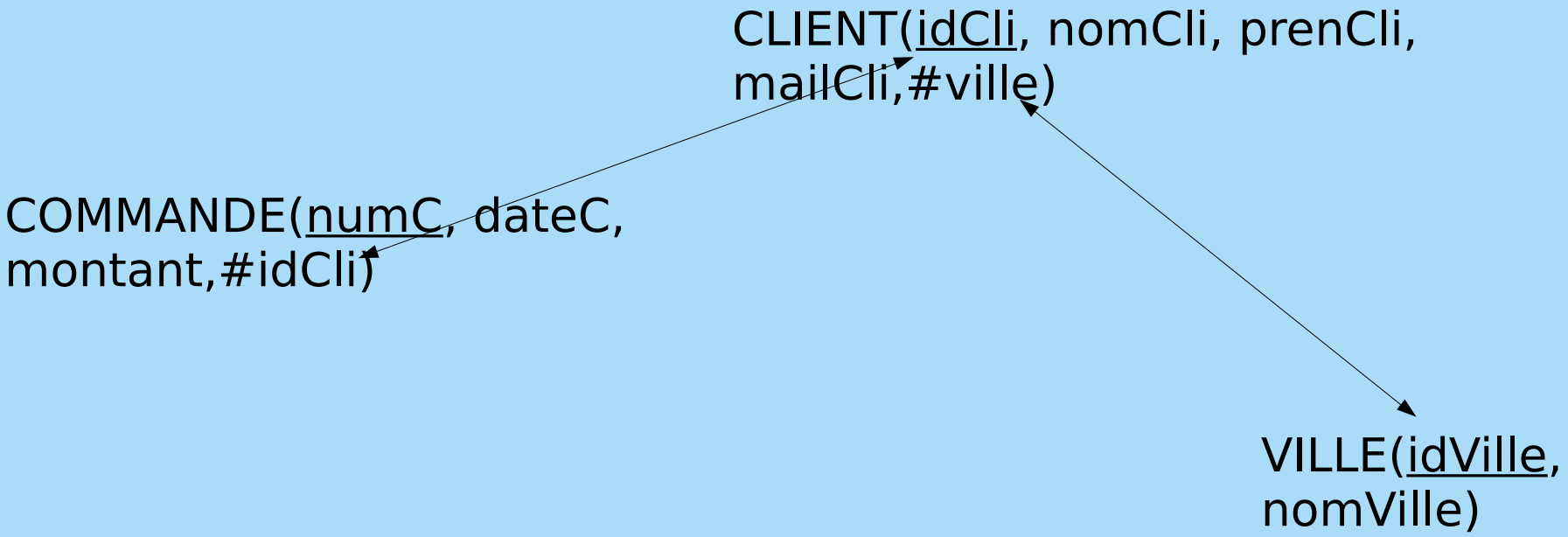


Schéma relationnel

En comprenant le mécanisme clé primaire / clé secondaire, on peut retrouver l'information dans les bases de données.

- Quels sont les noms des clients qui ont fait des commandes en juin 2019 ?
- Quelles sont les villes des clients qui ont commandé en 2019 ?
- Dans quel ville est le client qui a fait la plus grosse commande ?

Schéma relationnel

- Le principe du relationnel est de séparer l’information en différentes tables pour la stocker en utilisant le moins d’espace possible en mémoire

- A partir du schéma relationnel, on reconstitue ensuite les informations en fonction des besoins.

Schéma relationnel

- Dans les bases de données, pour chaque champ, on définit son type.

- Les types de données disponibles : **entier**, **réel**, **chaîne de caractères**, **booléen**, **date** (que contiennent les dates ?).

- Quid des types sur la base de données BD1 ?

Schéma relationnel

- Dans une table, on met d’abord les clés primaires, les attributs puis les clés secondaires.

- Les clés secondaires qui sont aussi des clés primaires sont mises au début.

- On peut aussi proposer un forme graphique du schéma relationnel.

Schéma relationnel

- D’autres configurations de table sont possibles
- BD2 :

HEBERGEMENT(idHeb, adresse)

HOTEL(#idHeb, nbEtoiles)

CHAMBRE_HOTE(#idHeb)

LOUE(#numCh, #idHeb, #idCli, #date)

CLIENT(idCli, nomCli)

DATE(date)

CONTACT(#idHeb, #idCli, idPers)

PERSONNEL(idPers, nomP)

TRAVAILLE_A(#idPers, #idHeb)

CHAMBRE(numCh, #idHeb)

Schéma relationnel

- Commentaire de la table TRAVAILLE_A(#idPers,#idHeb)
- Quelle est la clé de la table ?
- Quelle information permet d’enregistrer la table ?
- Peut-on enregistrer qu’un personnel travaille dans différents lieux ?

Schéma relationnel

- Les tables HOTEL(#idHeb, nbEtoiles) et CHAMBRE_HOTE(#idHeb)

- Concept de spécification : des sous-catégories d'une catégorie principale.

Schéma relationnel

- Commentaire de la table CONTACT(#idHeb, #idCli, idPers)
- Pour une valeur d'idHeb et d'idCli, combien de valeurs de idPers ?
- Quelle information permet d'enregistrer la table CONTACT ?

Schéma relationnel

- Commentaire de la table CHAMBRE(numCh, #idHeb)

- Quelle est la clé de la table ?

- On parle ici d’identification relative.

Schéma relationnel

- Commentaire de la table LOUE(#numCh, #idHeb, #idCli, #date)

- Simuler des occurrences de la table.

- Quelle information enregistre la table ?

Plan

1) Les bases de données, le schéma relationnel
2) **Normalisation des bases de données**
3) SGBD
4) Langage SQL – la requête SELECT
5) Langage SQL – Autres requêtes
6) Aller plus loin

Normalisation des BD

- Pourquoi normaliser les bases de données ?
- Pour un ensemble d'information donnée, on veut minimiser l'espace de stockage requis.
- Une base de données optimale sera obtenue en concevant un schéma relationnel optimal

Normalisation des BD

- Un schéma relationnel optimal est défini par le respect de trois règles.
- Ces règles sont définies en utilisant la notion de dépendance fonctionnelle.
- La dépendance fonctionnelle définit une relation entre les attributs.

Normalisation des BD

- Soient a1 et a2 deux attributs, on note le fait que a1 est en dépendance relationnelle sur a2 :

$$a1 \rightarrow a2$$

- Cela signifie que pour 1 valeur de a1, on a une et une seule valeur de a2.

Normalisation des BD

Quelles dépendances fonctionnelles sont valables dans ce qui suit ?

nomCli → idCli

idCli → nomCli

idCli → ville

idCli → nomVille

numC → montant

Normalisation des BD

- Par nature, une clé primaire est en dépendance fonctionnelle sur les autres attributs.

- Ecrire les dépendances fonctionnelles de BD1.

Normalisation des BD

- Forme normale 1 (FN1) : tous les attributs sont **atomiques**.

- Le cas de BD1 ?

- Un exemple d’attribut qui ne serait pas atomique ?

Normalisation des BD

- Forme normale 2 (FN2) : FN1 + toutes les dépendances fonctionnelles du modèle sont **élémentaires**.

- Soient 3 attributs, A, B et C, on dit que A, B → C est une dépendance élémentaire si on a ni A → C, ni B → C

- Dit autrement, il faut connaître A et B pour déterminer C : A ne suffit pas et B ne suffit pas.

Normalisation des BD

Des exemples de tables induisant des dépendances fonctionnelles non élémentaires :

OBTIENT(#idEtudiant, #idDevoir, note, nom)

COMMANDE(#idProd, #idClt, mailClit, libelleProd)

Normalisation des BD

- Forme normale 3 (FN3) : FN2 + toutes les dépendances fonctionnelles du modèle sont **directes**.

- Soient 3 attributs, A, B et C, on dit que A → C est une dépendance directe si on n'a pas A → B et B → C

Normalisation des BD

Un modèle avec des dépendances fonctionnelles non directes :

CLIENT(#idCli, nomCli, adresse)

COMMANDE(idCmd, dateCmd, #idClt)

LIVRAISON(idLivraison, date, #idCmd, #idClt)

PRODUIT(idProd, libelle, idTypeProd, libelleType)

Normalisation des BD

- Dans le cas d’un schéma non normalisé en 3ème FN : on stockera trop d’information et on aura des risques d’erreurs.

- A noter qu’existent d’autres formes normales.

Plan

1) Les bases de données, le schéma relationnel
2) Normalisation des bases de données
3) **SGBD**
4) Langage SQL – la requête SELECT
5) Langage SQL – Autres requêtes
6) Aller plus loin

SGBD

- SGBD : Système de Gestion de Base de Données.
- SGBD : Catégorie de logiciels, comme les tableurs, les traitements de texte ...
- SGBD : MySQL, Oracle, Microsoft Access, Microsoft SQL Server, PostGreSQL, DB2...

SGBD

- Oracle 

- Entreprise créée en 77.
- Oracle Database : parmi les 2-3 SGBD les plus employés
- Rachat de PeopleSoft en 2004 (PGI et CRM), rachat de SunMicrosystems en 2010.

SGBD

- MySQL 

- SGBD créé en 1995 en Sudède par l’entreprise MySQL AB sous licence GPL.
- MySQL : parmi les 2-3 SGBD les plus employés
- Rachat par SunMicrosystems en 2008.

SGBD

- Microsoft SQL Server 

- Première version en 1989
- Le 3ème SGBD le plus employé (évaluation 2014)
- Edité par … Microsoft.

SGBD

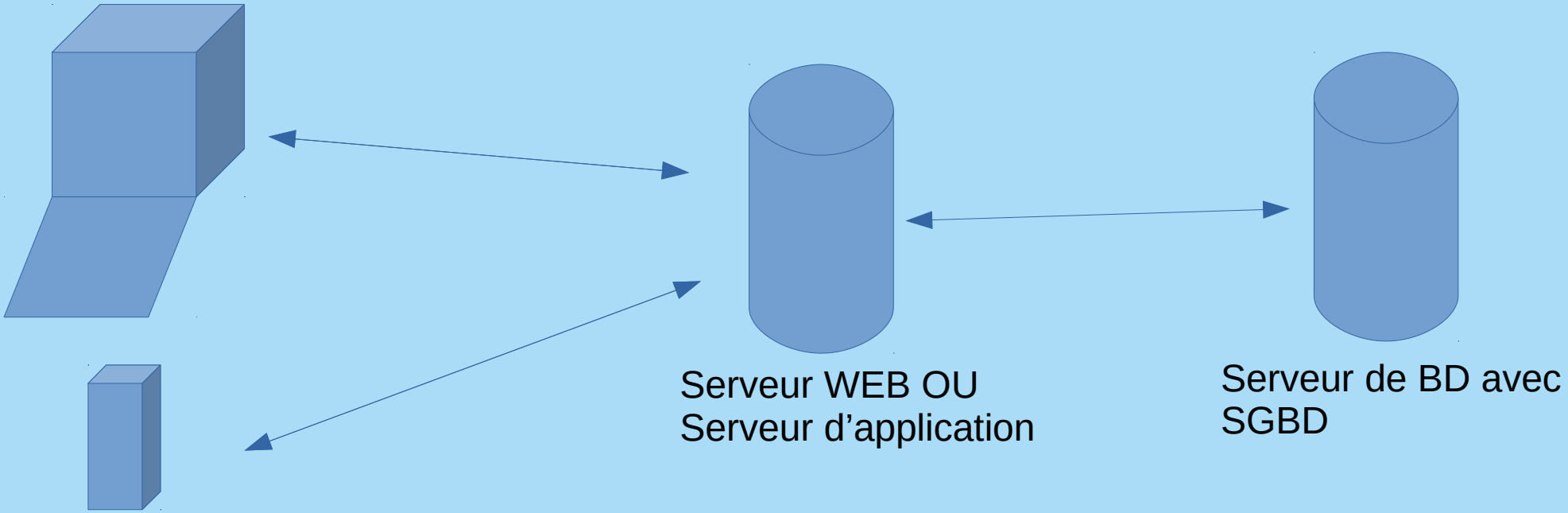
- Access
- Appartient à la suite Microsoft Office
- Logiciel qui offre une interface graphique, tant pour la visualisation du schéma relationnel, que pour la saisie des requêtes.

SGBD

- Libre Office Base
- Le pendant de Access dans la suite bureautique Libre Office
- On retrouve des conceptions similaires dont l’interface graphique.

SGBD

Où sont les SGBD ? Notion d’architecture :



SGBD

- Interagir avec un SGBD : le mode graphique (phpMyAdmin, interface graphique d'Access)

- Interagir avec un SGBD : langage SQL.

SGBD

- Le SGBD permet d’imposer des contraintes sur les données.

- Notion d’intégrité réferentielle => garantie de cohérence de la base.

SGBD

- Le SGBD permet de gérer des droits des utilisateurs.
- Droit de consultation, droit de suppression, droit d’insertion etc.
- Droit sur telle ou telle table, sur telle ou telle base de données...

Plan

1) Les bases de données, le schéma relationnel
2) Normalisation des bases de données
3) SGBD
4) **Langage SQL – la requête SELECT**
5) Langage SQL – Autres requêtes
6) Aller plus loin

Généralités sur le langage SQL

- SQL signifie **S**tructured **Q**uery **L**angage.
- "Langage de requête standard"
- C’est un langage supporté par les différentes SGBD.

Généralités sur le langage SQL

- Une requête est une ligne de code qui permet de faire une opération sur la base de données.
- Opérations de consultation.
- Opérations de modifications de la base de données.

Généralités sur le langage SQL

- Les opérations de consultation : SELECT

- Les opérations de modifications des occurrences de la base de données : INSERT, DELETE, UPDATE

Généralités sur le langage SQL

On considère que l'on dispose de la base de données suivante BD2 :

CLIENT(idCli,nomCli,pays,#csp)

CSP(idCSP,libelleCSP)

COMMANDE(numC,dateC,montant,#idCli)

Généralités sur le langage SQL

On a les occurrences suivantes :

CLIENT

idCli	nomCli	pays	csp
124765	Knol	France	3
126537	Parilly	Belgique	1
127645	Hutre	Suède	1
254366	Sanchez	France	3

CSP

idCSP	libelleCSP
1	Fonctionnaire
2	Salarié
3	Libéral
4	Sans emploi

COMMANDE

numC	dateC	montant	idCli
201920	12/05/2019	120,67	124765
201921	12/05/2019	2 398,66	124765
201922	01/06/2019	567,00	254366
201923	02/06/2019	3 400	254366

Requête SELECT

Les requêtes SELECT permettent de consulter des bases de données.

FROM Client

idCli	nomCli	pays	csp
124765	Knol	France	3
126537	Parilly	Belgique	1
127645	Hutre	Suède	1
254366	Sanchez	France	3

FROM Client

WHERE pays="France"

idCli	nomCli	pays	csp
124765	Knol	France	3
254366	Sanchez	France	3

SELECT nomCli,csp

FROM Client

WHERE pays="France"

nomCli	csp
Knol	3
Sanchez	3

Requête SELECT

Dans la clause SELECT, on indique les colonnes que l'on voir apparaître dans le résultat

SELECT nomCli,
pays
FROM Client

nomCli	pays
Knol	France
Parilly	Belgique
Hutre	Suède
Sanchez	France

SELECT csp, idCli,
nomCli
FROM Client

csp	idCli	nomCli
3	124765	Knol
1	126537	Parilly
1	127645	Hutre
3	254366	Sanchez

SELECT idCli
FROM Client

idCli
124765
126537
127645
254366

Requête SELECT

Pour afficher tous les champs de la table, on peut les énumérer ou mettre "*" (lire "all")

```
SELECT idCli, nomCli, pays, csp
FROM Client
```

<=>

```
SELECT *
FROM Client
```

idCli	nomCli	pays	csp
124765	Knol	France	3
126537	Parilly	Belgique	1
127645	Hutre	Suède	1
254366	Sanchez	France	3

Requête SELECT - WHERE

- La clause WHERE permet de selectionner des lignes.
- Derrière le WHERE on spécifie des conditions. Ne seront conservées que les lignes qui respectent les conditions spécifiées.
- Les conditions sont des expressions à valeur booléenne.

Requête SELECT - WHERE

- Les conditions s'expriment à partir d'opérateurs de comparaison : = (égal), > (supérieur strict), < (inférieur strict), <= (inférieur),>= (supérieur), <> (différent)

- Les opérateurs de comparaison peuvent s'appliquer à tous les types d'attribut.

Requête SELECT - WHERE

IdCli<200000

idCli=123764

idCli<>123764

nomCli="Parilly"

dateC>"2019/06/01"*

dateC>="2019/06/01"

nomCli<"B"

dateC>"B"

*Bien noter que les dates se présentent au format anglo-saxon

Requête SELECT - WHERE

- Il est possible d’enchaîner les conditions avec AND et OR.
- La condition C1 AND C2 est vraie si C1 est vraie et si C2 est vraie
- La condition C1 OR C2 est vraie si C1 est vraie ou si C2 est vraie, ou si les deux sont vraies.

Requête SELECT - WHERE

idCli<200 AND idCli>=300000

idCli<>123764 AND nomCli="Parilly"

dateC>"2019/10/12" AND dateC<"2019/10/20"

dateC>"2019/10/12" OR dateC<"2019/10/20"

Requête SELECT - WHERE

Priorité du and sur or => le parenthèsage peut être nécessaire

Attention, des expressions ne sont pas possibles : 7<a<8

Attention aux conditions qui ne sauraient être satisfaites : (7<a or b>5) and a<6

Requête SELECT - WHERE

- La plupart des opérateurs de comparaison s’utilisent de manière attendue
- A noter qu’existent des opérateurs spécifiques à des types de données.
- On évoque ici LIKE et BETWEEN

Requête SELECT - WHERE

- LIKE concerne les chaînes de caractères.
- LIKE compare une chaîne de caractères à un "schéma"
- * désigne n’importe quel caractère, % désigne n’importe quelle chaîne de caractères.

Requête SELECT - WHERE

WHERE mailCli LIKE " %@gmail.Com"

WHERE nomCli LIKE "D %"

WHERE nomCli LIKE "***"

WHERE pays LIKE "%g %"

Requête SELECT - WHERE

- BETWEEN concerne les dates.

- BETWEEN date1 AND date2

Requête SELECT - WHERE

```
WHERE dateC BETWEEN "2019/03/01" AND "2019/05/01"
```

```
WHERE dateC BETWEEN "2019/01/01" AND CURRENT_DATE()
```

Requête SELECT - WHERE

- A noter qu’un champ peut n’être pas rempli

- Dans ce cas, sa valeur est à NULL :

WHERE csp=NULL

- Certains champs ne peuvent pas être à NULL : les clés primaires par exemple.

Fonctions disponibles

- SQL prévoit un certain nombre de fonctions.
- Elles peuvent être utilisées pour calculer de nouveaux éléments à partir des éléments des tables
- Ex : l'age à partir de la date de naissance, un délai de livraison à partir d'un date de commande et d'une date de livraison...

Fonctions disponibles

- ROUND(valeur numérique, nombre) → arrondi
- DAY(date) → le jour de la date Ex : DAY("2013/05/17") renvoie 17.
- MONTH(date) → le mois de la date.
- YEAR(date) → l ‘année de la date.

Fonctions disponibles

- CURRENT_DATE**()** renvoie la date du jour.

- DATEDIFF(date1, date2) → la différence, en nombre de jours, entre les deux dates.

Fonctions disponibles

```
SELECT *
FROM Commande
WHERE MONTH(dateC)=5 AND YEAR(dateC)=2018

SELECT ROUND(pib/pop,2) AS "Pib moyen"
FROM Pays
```

Fonctions disponibles

```
SELECT idCmd, DATEDIFF(dateCde, dateLiv) AS delai
FROM Commande
```

```
SELECT DATEDIFF(dateNaiss, CURRENT_DATE()) AS age
FROM Personne
WHERE age>5
```

Renommage

- AS sert au renommage.
- On peut renommer un attribut ou un calcul.
- On peut utiliser l’attribut renommé dans les conditions et dans les autres clauses.

Requête SELECT – ORDER BY

- On peut ajouter une autre clause qui permet de classer les résultats : ORDER BY.
- Le ORDER BY intervient comme dernière clause.
- Il est également possible de trier sur différents critères
- ASC / DESC

Requête SELECT – ORDER BY

```
SELECT nomCli,csp
FROM Client
WHERE pays="France"
ORDER BY nomCli ;
```

```
SELECT nomCli,csp
FROM Client
ORDER BY nomCli DESC;
```

```
SELECT nomCli,csp
FROM Client
ORDER BY nomCli DESC, csp ASC;
```

SELECT DISTINCT

- Supposons que l’on veuille la liste des pays dans lequels on a des clients :
- SELECT pays FROM Client => cette requête va afficher chaque pays autant de fois qu’il y a un client dans ce pays
- Pour afficher sans répétition :

  SELECT DISTINCT pays FROM Client

Opérateurs d’agrégation

- COUNT(*)
- AVG(att)
- SUM(att)

Jointure de tables

- On peut vouloir de l’information issue de différentes tables.

- Par exemple, on veut récupérer les noms et les CSP des clients qui ont passé des commandes en 2018.

=> on a besoin d’utiliser les tables CLIENT et COMMANDE

=> on fait une jointure entre les tables.

Jointure de tables

- Une première syntaxe de jointure

```
SELECT nom, csp
FROM Client JOIN Commande
ON Client.idCli=Commande.idCli
WHERE YEAR(dateC)=2018
```

Jointure de tables

Client JOIN Commande ON Client.idCli=Commande.idCli

- Est une nouvelle table stockée temporairement en mémoire
- Peut-être jointe à une autre table
- A tous les champs de la table Client et de la table Commande

Jointure de tables

- Une jointure plus complexe :

```
SELECT nom, csp, libelleCSP
FROM Client JOIN Commande
ON Client.idCli=Commande.idCli
JOIN CSP ON Client.csp=CSP.idCSP
WHERE YEAR(dateC)=2018
```

Jointure de tables

- Une autre syntaxe de jointure (qu'on considérera équivalente) :

```
SELECT nom, csp, libelleCSP
FROM Client, Commande
WHERE Client.idCli=Commande.idCli
AND YEAR(dateC)=2018
```

Jointure de tables

- Une version simplifiée en renommant les tables :

```
SELECT nom, csp, libelleCSP
FROM Client Cl, Commande Co
WHERE Cl.idCli=Co.idCli
AND YEAR(dateC)=2018
```

Jointure de tables

- Avec l'autre type de jointure :

```
SELECT nom, csp
FROM Client Cl JOIN Commande Co
ON Cl.idCli=Co.idCli
WHERE YEAR(dateC)=2018
```

Requête SELECT – GROUP BY

- On considère une simple table :

  PERSONNE(idPers, nomP, preP, salaire, ville, age)

- On peut faire une requête simple

  SELECT AVG(salaire)

  FROM Personne ;

- On obtient une seule valeur

Requête SELECT – GROUP BY

- On peut faire une requête plus complexe

  SELECT ville, AVG(salaire)

  FROM Personne

  GROUP BY ville;

- On obtient **plusieurs valeurs** : un salaire moyen **par ville**.

Requête SELECT – GROUP BY

- On peut provoquer d’autres regroupements :

```
SELECT age, AVG(salaire)
FROM Personne
GROUP BY age;
```

- On obtient le salaire moyen par age.

Requête SELECT – GROUP BY

- On peut continuer les variations :

```
SELECT ville, AVG(age)
FROM Personne
GROUP BY ville;
```

- On obtient l’age moyen par ville.

Requête SELECT – GROUP BY

- Et encore :

```
SELECT AVG(age)
FROM Personne
WHERE age>=50
GROUP BY ville;
```

- L’age moyen, pour chaque ville, des personnes de plus de 50 ans.

Requête SELECT – GROUP BY

AVG : correspond à AVERAGE (moyenne). Il s'agit d'un opérateur d'agrégation. On a d'autres opérateurs :

- SUM pour somme
- MAX et MIN pour le MAX et le MIN respectivement
- COUNT pour compter un nombre d'occurrences d'une table.

Requête SELECT – GROUP BY

- L’utilisation est similaire :

```
SELECT ville, COUNT(*)
FROM Personne
GROUP BY ville;
```

- Le nombre des personnes de la base par ville

Requête SELECT – GROUP BY

- L’utilisation est similaire :

```
SELECT ville, SUM(salaire)
FROM Personne
GROUP BY ville;
```

- Le salaire total des gens de la base, pour chaque ville.

Requête SELECT – GROUP BY

- Un critère à retenir : est ce qu'on veut une valeur ou plusieurs valeurs dans le résultat final ?

- Mettre le critère de regroupement parmi les attributs affichés pour améliorer la lisibilité.

- Bien retenir l'ordre des clauses.

Requête SELECT – GROUP BY

On peut faire un regroupement sur plusieurs critères.

```
SELECT ville, SUM(salaire)
FROM Personne
GROUP BY ville,age;
```

Requête SELECT avec imbrication

- Le cas des requêtes imbriquées.
- A partir de BD1, trouver les clients qui n’ont pas passé de commande.
- A partir de BD1, trouver les clients qui n’ont pas passé de commande en 2019.
- La liste des clients ayant passé une commande de montant maximal.

Plan

1) Les bases de données, le schéma relationnel
2) Normalisation des bases de données
3) SGBD
4) Langage SQL – la requête SELECT
5) **Langage SQL – Autres requêtes**
6) Aller plus loin

Modifier la base de données

- SELECT permet de consulter des données insérées dans les tables.

- D’autres requêtes permettent de modifier la base de données.

- DELETE permet de supprimer des occurrences, UPDATE permet de modifier des occurrences, INSERT INTO permet d’insérer des occurrences.

DELETE

DELETE permet de supprimer des occurrences, des lignes des tables.

```
DELETE FROM Client
WHERE ville=123
```

```
DELETE FROM Commande
WHERE YEAR(dateC)<2010
```

UPDATE

UPDATE permet de mettre à jour des occurrences.

UPDATE Commande
SET montant=montant*0.9
WHERE idCli=127675

UPDATE Commande
SET dateC="2019/09/01"
WHERE numC=167

INSERT

INSERT permet d’ajouter des occurrences

```
INSERT INTO Client (idCli, nomCli, mailCli)
VALUES (176523, "Robert", "Robert69@hotmail.Com)

INSERT INTO Commande (numC, dateC, montant, idCli)
VALUES (777,"2019/09/08",876.35,176523)
```

Que se passerait-il si on exécutait la seconde requête avant la première ?

A quelle notion cela renvoie t’il ?

Plan

1) Les bases de données, le schéma relationnel
2) Normalisation des bases de données
3) SGBD
4) Langage SQL – la requête SELECT
5) Langage SQL – Autres requêtes
6) **Aller plus loin**

Aller plus loin

- On parle de bases de données **relationnelles**. Il existe des bases de données différentes et qui répondent à d’autres impératifs.

- Les BD relationnelles sont une forme de données structurées. Par opposition à ?