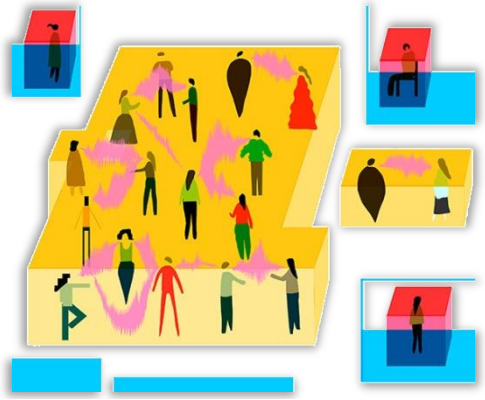
Normalisation / Dénormalisation SGBDR	BDD avancé #1 - ENSG – 3A



Réaliser une solution qui permet de visualiser la proximité de personnes physiques avec des entreprises proches en laissant à l'utilisateur le soin de préciser un rayon et une activité précise. Nous nous appuierons sur des fichiers en opendata de type Sirene, Bano et crawling/parsing.

Objectif

Comprendre l'intérêt de la normalisation/dénormalisation sur une base de données relationnelle avec des volumes de données important. Nous utiliserons au choix le SGBDR postgreSQL ou MySQL.

Téléchargement du jeu de données

Récupérer le jeu de données remis par l'enseignant (fichiers csv, json,…).

Modélisation

Vous allez analyser les fichiers constituants le zip et concevoir le diagramme EA (sans réaliser d'optimisation). Puis, créez le modèle relationnel depuis l'EA et enfin, générer le code SQL DDL permettant de créer les tables cibles de la base de données. Vous pouvez utiliser www.dbdesigner.net pour créer votre modèle relationnel et générer directement le code SQL.

Intégration

Une fois que les tables sont créées, réalisez les transformations sur les fichiers de données pour les intégrer dans la base de données. Vous pouvez réaliser ces opérations dans le langage de votre choix (python, php, etc.). L'objectif est d'automatiser les chargements à l'aide d'un script réalisé par vos soins. Vous gérerez les rejets le cas échéant en proposant des solutions adaptées lors du chargement.

Requêtage

Vous allez rédiger les requêtes SQL décrites dans la section « problématique » à partir des tables alimentées pour afficher le résultat sous forme de listes. Exécuter ces requêtes avec le mode EXPLAIN afin de noter les temps de traitement. On pourra utiliser une procédure stockée en PL/SQL pour calculer la distance entre 2 coordonnées GPS. Celle-ci est donnée en annexe. Bien évidemment, d'autres solutions sont possibles…

Index

Vous pouvez créer des clefs et indexes permettant d'améliorer les performances de la requête. Pour ce faire, utilisez la commande EXPLAIN et noter les améliorations le cas échéant.

Enrichissement de données

Vous pouvez créer une nouvelle table à partir des tables sources afin d'éviter les jointures et accélérer les requêtes. Proposez une telle table que vous alimenterez via un script php ou python en automatique. L'objectif étant de construire une table unique avec des données enrichies. Vous devrez gérer les cas de gestion afin de réconcilier les informations pertinentes pour effectuer vos requêtes (ajouter à la table sirene : les urls, les téléphones et les emails du fichier de crawling fourni ainsi que les coordonnées GPS de BANO). Vous effectuerez à nouveau les traitements précédents et noterez les temps de traitements avec et sans clef/index afin d'établir un comparatif avec l'étape précédente et ainsi tirer vos conclusions sur cette solution.

Dénormalisation

Dans notre application, la requête précédente est très importante, il faut donc garantir un temps de réponse minimum. Proposez une dénormalisation du modèle qui permette de simplifier le modèle avec des dimensions adaptées à vos requêtes. Intégrez les données par rapport à votre dénormalisation via des scripts php ou python et modifiez les requêtes de manière à profiter de cette modification. Notez à nouveau les temps de traitements et conclure quant à ce procédé. Vous paraît-il judicieux ? si oui, pourquoi ? sinon, proposer une solution mieux adaptée.

Modification

Etudiez l'aspect modification (insert, update, delete) d'une donnée identifiée dans la base.

Rendu

Il s'agit de rendre un PDF/Word succinct qui comprend :

1. Diagramme EA et détailler le code SQL DDL (format txt)
2. Préciser l'outil choisi et le commenter le code d'intégration (format txt)
3. Faire un tableau de synthèse de benchmark sur les requêtes étudiées :
    a. Sans optimisation
    b. Avec une clef
    c. Avec clef + index
    d. Optimisation avec les formes normales (1,2,3 et 4)
    e. Optimisation avec la dénormalisation (schéma en étoile)
4. Détailler le code SQL DDL (format txt) de la dénormalisation + dump (format.sql) des dimensions et des faits
5. Etude des modifications et adaptations
6. Conclure

Problématique

Le but est de déterminer, à partir de la position de personnes physiques dont on possède les coordonnées GPS, la distance les séparant à des entreprises autour d'elle dans un rayon prédéfini et d'afficher par ordre croissant le résultat avec la possibilité de choisir l'activité de l'entreprise (code APE). On se basera sur des fichiers sirene, BANO en open data sur un périmètre géographique simple (Département 93) et un fichier de crawling et parsing en json.

Le travail consiste d'une part à écrire la ou les requêtes permettant d'obtenir la liste des entreprises dans un rayon défini sur une activité choisie trier par ordre croissant en fonction de la distance.

D'autre part, de créer une table supplémentaire qui permettra d'optimiser ce type de traitement ; celle-ci permettra de croiser les informations pour accélérer la requête précédente.

Vous pourrez utiliser des requêtes, des procédures stockées et/ou programme python/php avec mysql/postgres pour effectuer ce type de travail.

ANNEXE 1 (Rappel des types de commande SQL) :

DDL : Data Definition Language statements are used to define the database structure or schema. Some examples:

- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

DML : Data Manipulation Language statements are used for managing data within schema objects. Some examples:

- SELECT - retrieve data from the a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - deletes all records from a table, the space for the records remain
- MERGE - UPSERT operation (insert or update)

- CALL - call a PL/SQL or Java subprogram
- EXPLAIN PLAN - explain access path to data
- LOCK TABLE - control concurrency

DCL : Data Control Language statements. Some examples:

- GRANT - gives user's access privileges to database
- REVOKE - withdraw access privileges given with the GRANT command

TCL : Transaction Control statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

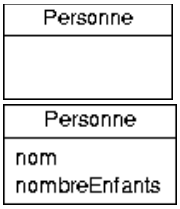
- COMMIT - save work done
- SAVEPOINT - identify a point in a transaction to which you can later roll back
- ROLLBACK - restore database to original since the last COMMIT
- SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use

ANNEXE 2 (Rappel des concepts de base du modèle EA)

Le modèle *Entité-Association* (notation : EA) est aussi fréquemment nommé *Entité-Relation* et parfois *Entité-Relation-Attribut*. Le modèle EA propose des concepts (principalement les entités, les associations et les attributs) permettant de décrire un ensemble de données relatives à un domaine défini afin de les intégrer ensuite dans une BDD.

Une **entité** est un objet, une chose concrète ou abstraite qui peut être reconnue distinctement. Exemples d'entité : Jean Dupont, Pierre Bertrand,…

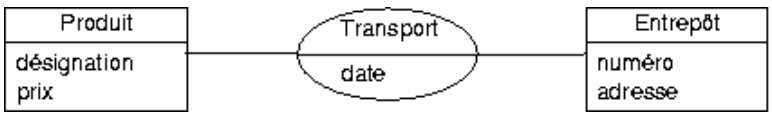
Un **type-entité** est un ensemble d'entités qui possèdent les mêmes caractéristiques.



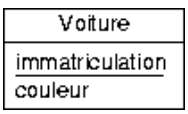
Un **attribut** (ou une **propriété**) est une caractéristique associée à un type-entité. Exemples d'attribut : l'âge d'une personne, le code d'un fournisseur,…

Une **association** (ou une **relation**) est un lien entre plusieurs entités. Exemples d'association : le transport de la Clio 3333 XR 06 vers le dépôt de Nice.

Un **type-association** (ou un **type-relation**) est un ensemble de relations qui possèdent les mêmes caractéristiques. Exemples de type-association : le transport d'un produit vers un entrepôt



Un **identifiant** d'un type-entité ou d'un type-association est constitué par un ou plusieurs de ses attributs qui doivent avoir une valeur unique pour chaque entité ou association de ce type. Exemples d'identifiant : le numéro d'immatriculation.



La **cardinalité** d'un type-association est le nombre de fois minimal et maximal qu'une entité peut intervenir dans une association de ce type. Exemple de cardinalité : un client peut commander entre 1 et n produits.

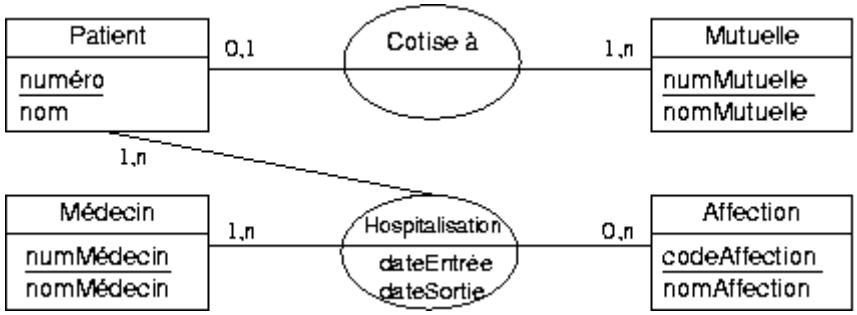
La cardinalité minimale peut-être :

- 0 signifie qu'une entité peut exister tout en étant impliquée dans aucune association.
- 1 signifie qu'une entité ne peut exister que si elle est impliquée dans au moins une association.
- n signifie qu'une entité ne peut exister que si elle est impliquée dans plusieurs associations.

La cardinalité maximale peut-être :

- 0 signifie qu'une entité ne peut pas être impliquée dans une association.
- 1 signifie qu'une entité peut être impliquée dans au maximum une association.
- n signifie qu'une entité peut être impliquée dans plusieurs associations.

Voici un exemple de modélisation EA, très simplifiée :



Voici les relations déduites de ce schéma :

- `Patient(numéro, nom, numMutuelle)`
- `Mutuelle(numMutuelle, nomMutuelle)`
- `Médecin(numMédecin, nomMédecin)`
- `Affection(codeAffection, nomAffection)`
- `Hospitalisation(numéro, codeAffection, numMédecin, dateEntrée, dateSortie)`

On utilise la notation suivante : `nomRelation(clé, autresAttributs)`

ANNEXE 3 (Rappel des formes normales)

Une **dépendance fonctionnelle**, notée DF, indique que la valeur d'un ou plusieurs attributs est associée à *au plus une valeur* d'un ou plusieurs autres attributs. Par exemple, voici une relation :

`Anniversaire(numéro : Entier, amie : Chaîne, ville : Chaîne, cadeau : Chaîne)`

Une relation est en **première forme normale** 1NF si et seulement si tous ses attributs ont des **valeurs simples** (non multiples, non composées). Si besoin est, on décompose les attributs ou la relation pour respecter la 1NF. Ex : Nom + prénom

Soient C une clé candidate de R et A un attribut de R. R est en **deuxième forme normale** (2NF) si et seulement si elle est en 1NF et pour tout A tel que A n'appartient pas à C, on a C →R A élémentaire (cf. définition d'élémentaire ci-dessus).

- Une relation peut être en 2NF par rapport à une de ses clés candidates et ne pas l'être par rapport à une autre.

- Pour rechercher une 2NF, il est au préalable nécessaire de déterminer toutes les DF et de choisir une clé candidate. Il est recommandé de trouver toutes les clés candidates afin de ne pas en laisser passer une plus intéressante qu'une autre.
- Si besoin est, on décompose les attributs ou la relation pour respecter la 2NF.
- Une relation avec une clé candidate choisie réduite à un seul attribut est, par définition, forcément en 2NF.

Soient C une clé candidate de R et A et B deux ensembles non vides disjoints d'attributs de R. R est en **troisième forme normale** (3NF) si et seulement si elle est en 2NF et pour tous A et B tels que A et Bdisjoints de C, on n'a pas A →R B.

- Une relation peut être en 3NF par rapport à une de ses clés candidates et ne pas l'être par rapport à une autre.
- Si besoin est, on décompose les attributs ou la relation pour respecter la 3NF.
- Une relation en 2NF avec au plus un attribut qui n'appartient pas à la clé candidate choisie est, par définition, forcément en 3NF.

Une relation est en **forme normale de Boyce-Codd** (BCNF) si et seulement si ses clés candidates sont les uniques sources de DF.

- Si une relation est en BCNF, elle l'est par définition pour toutes ses clés candidates.
- Si besoin est, on décompose les attributs ou la relation pour respecter la BCNF.

ANNEXE 4 (Rappel des formes dénormalisées OLAP) :

Un schéma en étoile consiste en une table de faits centrale et plusieurs tables dimensions. Les mesures d'intérêt pour l'OLAP sont stockées dans la table des faits (e.g., ventes, stock). Pour chaque dimension du modèle multidimensionnel il existe une table (e.g., région, produit, temps). Cette table stocke les informations relatives aux dimensions.

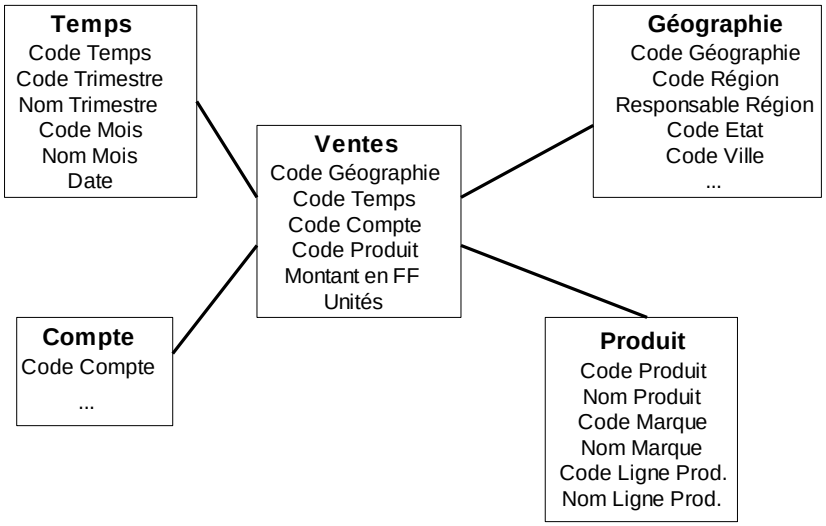


Figure 3 : Schéma en étoile

La figure 3 est un exemple de schéma en étoile. La table ventes au centre de l'étoile est la table de faits avec les clés étrangères code géographie, code temps, code compte, et code produit vers les tables dimensions correspondantes.
Les dimensions sont souvent organisées de façon hiérarchique. Chaque table dimension consiste en plusieurs attributs décrivant un niveau de dimension. Par exemple, la dimension produit est organisée dans une hiérarchie comportant les niveaux produit, catégorie de produit, et marque.

La structure hiérarchique d'une dimension est particulièrement importante pour les opérations drill-down et roll-up. Chaque hiérarchie est représentée par un ou plusieurs attributs dans la table dimension. Les tables dimensions ont une structure dénormalisée. A cause des problèmes relatifs aux structures de données dénormalisées, comme la redondance, il peut être utile de créer un schéma en flocon. Ceci est fait en normalisant les tables dimensions du schéma en étoile.

Les opérations typiques exécutées par les clients OLAP sont :

. Roll-up (relever le niveau d'agrégation)
. Drill-down (diminuer le niveau d'agrégation)
. Slice et Dice (sélection et projection)
. Pivot (ré-orienter la vue multidimensionnelle)

ANNEXE 5 (Procédure stockée PL/SQL de calcul de distance pour MySQL) :

```
DELIMITER $$
--
-- Functions
--
CREATE DEFINER=`root`@`localhost` FUNCTION `get_distance_metres`(lat1
DOUBLE, lng1 DOUBLE, lat2 DOUBLE, lng2 DOUBLE) RETURNS double
BEGIN
   DECLARE rlo1 DOUBLE;
   DECLARE rla1 DOUBLE;
   DECLARE rlo2 DOUBLE;
   DECLARE rla2 DOUBLE;
   DECLARE dlo DOUBLE;
   DECLARE dla DOUBLE;
   DECLARE a DOUBLE;

   SET rlo1 = RADIANS(lng1);
   SET rla1 = RADIANS(lat1);
   SET rlo2 = RADIANS(lng2);
   SET rla2 = RADIANS(lat2);
   SET dlo = (rlo2 - rlo1) / 2;
   SET dla = (rla2 - rla1) / 2;
   SET a = SIN(dla) * SIN(dla) + COS(rla1) * COS(rla2) * SIN(dlo) * SIN(dlo);
   RETURN (6378137 * 2 * ATAN2(SQRT(a), SQRT(1 - a)));
END$$

DELIMITER ;
```