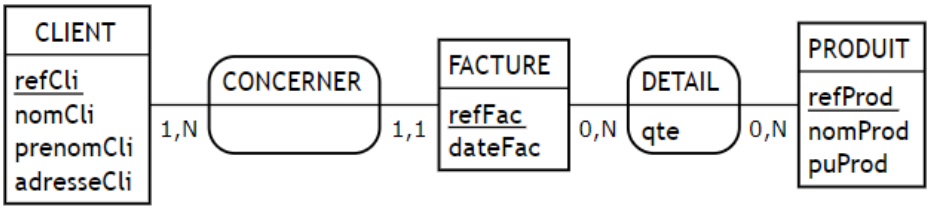
Bases de données relationnelles, initiation au langage SQL

- Le modèle relationnel étudié précédemment est un modèle mathématique permettant de raisonner sur des données en tables.
- Ce modèle est mis en oeuvre par un logiciel particulier : le Système de Gestion de Base de Données (SGBD).
- La grande majorité des SGBD relationnels utilisent le langage SQL (Structured Query Langage).
- Le langage SQL permet d'envoyer des ordres au SGBD de deux natures :
  - Les mises à jour qui permettent la création de relations, d'ajout d'entité, leur modification ou leur suppression.
  - Les requêtes qui permettent de récupérer des données correpondant à un critère.
- Dans cette feuille, nous allons , à partir d'un MCD et un MLD définis(voir feuille sur le modèle relationnel), découvrir un environnement qui utilise SQL et créer/supprimer des tables de données pour pouvoir ensuite ajouter/supprimer des données.

Exemple :

L'entreprise QuiVendTout a créé une base de données lui permettant d'éditer les factures de ses clients.

- Voici Le MCD et le MLD de cette base de données :



CLIENT (refCli, nomCli, prenomCli, adresseCli)
FACTURE (refFac, dateFac, *refCli*)
DETAIL (*refFac*, *refProd*, qte)
PRODUIT (refProd, nomProd, puProd)

- Et voici à quoi peuvent ressembler les tables créées et peuplées par le système à l'aide du langage SQL :

1

refCli	nomCli	prenomCli	adresseCli
1	BANAZIAI	Jules	Grenoble
2	DONGEPLI	Armelle	Limoges
3	BOBAINO	Julie	Nimes
4	BOUTIDICHT	Maxime	Grenoble

2

refProd	nomProd	puProd
1	Sucre	1.20
2	Cereales	1.30
3	Biscottes	1.15
4	Poudre petit dejeuner	2.30
5	Cafe	2.50
6	The	3.10

3

refFac	refProd	qte
1	2	8
1	3	2
2	6	7
3	1	2
3	3	2
3	5	9

4

refFac	dateFac	refCli
1	2019-06-14	2
2	2019-12-26	2
3	2019-03-14	1

Exercice 1 :

1. Nommer les quatre tables créées.

   - 1 :
   - 2 :
   - 3 :
   - 4 :

1. Que peut-on dire de la propriété `refCli` ?

Réponse :

1. Pourquoi n'y en a-t-il que quatre ?

Réponse :

1. Que contient la table DETAIL ?

Réponse :

1. Création de tables

Le langage SQL permet de créer des tables en spécifiant leur nom, leurs attributs, les types de ces derniers et les contraintes associées à la table. Voici les ordres saisis qui permetttent la création des tables `CLIENT` et `FACTURE` :

```
CREATE TABLE CLIENT (
  refCli INT,
  nomCli VARCHAR(15),
  prenomCli VARCHAR(15),
  adresseCli VARCHAR(20),
  PRIMARY KEY (refCli)
);
```

```
CREATE TABLE FACTURE (
  refFac INT,
  dateFac DATE,
  refCli INT,
  PRIMARY KEY (refFac)
);
```

```
ALTER TABLE FACTURE ADD FOREIGN KEY (refCli) REFERENCES CLIENT (refCli);
```

Syntaxe

- L'interaction avec le SGBD se fait par l'envoi d'une suite d'ordres SQL.
- Un ordre peut s'étendre sur plusieurs lignes.
- Les blancs et l'indentation ne sont pas significatifs mais améliorent la lisibilité.
- Un ordre se termine par `;` .Il y en a ici donc trois.
- Le langage SQL est insensible à la casse : on peut écrire `CREATE` , `Create` , `create` , `cREATE` , ...
- Une convention est néanmoins de saisir les principaux mots du langage en majuscules et les attributs en minisucules.
- Un attribut ne peut pas contenir d'espace; on écrira ainsi pour un attribut `ref_cli` ou `refcli` mais PAS `ref Cli` .
- SQL est assez verbeux, les ordres ressemblent au langage naturel anglais.

Types de données

A la creation d'une table, on spécifie le type de données de chaque attribut. Voici les principaux types de données SQL.

Types numériques

Type	exact/approché	description
SMALLINT	exact	entier 16 bits signé
INTEGER	exact	entier 32 bits signé
INT	exact	alias pour INTEGER
BIGINT	exact	entier 64 bits signé
DECIMAL(a,b)	exact	decimal signé de a chiffres dont b après la virgule
REAL	approché	flottant 32 bits
DOUBLE PRECISION	approché	flottant 64 bits

Remarque :

- On distingue les types numériques exacts et les types numériques approchés.
- Le type `DECIMAL` est très utile pour représenter des sommes d'argent.
- Exemple : `DECIMAL(5,2)` permet de représenter les nombres signés à 5 chiffres dont 2 après la virgule, c'est à dire entre $-999,99$ et $999,99$

Types texte

Type	description
CHAR(n)	Chaîne contenant exactement n caractères .Les caractères manquants sont complétés par des espaces
VARCHAR(n)	Chaîne contenant au plus n caractères
TEXT	Chaîne de taille quelconque

Remarques :

- Il existe de nombreux autres types de données texte, mais ils sont inégalement supportés par les différents systèmes.
- Utiliser le type `CHAR` est utile lorsque l'on stocke des chaînes de taille fixe.
- Si l'on utilise le type `CHAR(10)` pour stocker le mot `hello` , alors le système stockera `hello_____` .

Type booléen

- Le type `BOOLEAN` est inégalement supporté par les systèmes.
- En pratique, on utilise plutôt `CHAR(1)` en indiquant utilisant deux caractères possibles, `T` et `F` par exemple.

Type des dates, durées et instants

Type	description
DATE	date au format 'AAAA-MM-JJ'
TIME	heure au format 'hh:mm:ss'
TIMESTAMP	instant (date et heure) au format 'AAAA-MM-JJ hh:mm:ss'

Remarques :

- La gestion des dates et durées est plus complexe qu'il n'y paraît et source de nombreux bugs.
- Les valeurs de ces types s'écrivent comme des chaînes de caractères.
- Une fonctionnalité intéressante:
  - En utilisant le type DATE, on peut facilement ajouter des jours et le système effectue lui même les calculs avec les changements éventuels de mois ou d'années.
  - Ainsi l'opération `2012-12-21` + 12 donnera la valeur `2013-01-01` .

Valeur NULL

- Elle correspond partiellement au type `None` de python et désigne une absence de valeur.
- Son utilisation est parfois délicate et source de bugs.

Clés

- Les mots clés `PRIMARY KEY` permettent d'indiquer qu'un attribut est une clé primaire. Il y a plusieurs façons de l'utiliser. Dans l'ordre cité en exemple, on indique après la création des attributs celui qui est la clé primaire : `PRIMARY KEY (refCli)` .
- Les mots clés `FOREIGN KEY` et `REFERENCES` permettent d'indiquer qu'un attribut est une clé étrangère. Là encore, il y a plusieurs façons de faire. Pour plus de souplesse, on pourra écrire les ordres de création de clés étrangères après les ordres de création de tables. C'est ce qui est fait à la dernière ligne : `ALTER TABLE FACTURE ADD FOREIGN KEY (refCli) REFERENCES CLIENT (refCli);`

Exercice 2 :

1. Ecrire l'ordre `SQL` qui permet de créer la table `PRODUIT` .
2. Ecrire l'ordre `SQL` qui permet de créer la table `DETAIL` .
3. Ajouter les ordres qui permettent de spécifier les clés étrangères.

Réponses :

1.

1.

1.

Exercice 3 : Application

Commencer l'activité contenue dans ce document : Activite_SQL/SQL_TP1.ipynb (Activite_SQL/SQL_TP1.ipynb)

2. Insertion dans une table

Voici l'ordre qui a permis de peupler la table `CLIENT` :

```
INSERT INTO CLIENT VALUES
    (1,'BANAZIAI','Jules','Grenoble'),
    (2,'DONGEPLI','Armelle','Limoges'),
    (3,'BOBAINO','Julie','Nimes'),
    (4,'BOUTIDICHT','Maxime','Grenoble');
```

Remarques :

- On spécifie le nom de la table, suivi d'une suite de quadruplets chacun entre parenthèses. Chaque quadruplet représente une nouvelle ligne de la table.
- Les valeurs des attributs sont supposées dans le même ordre que lors du `CREATE TABLE` .
- Les contraintes d'intégrité sont vérifiées au moment de l'insertion. Si une valeur incorrecte est saisie, alors la ligne n'est pas ajoutée et un message d'erreur est affiché.

Exercice 4 :

Ecrire l'ordre qui a permis d'insérer les valeurs de la table `DETAIL` précédente.

Réponse

Exercice 5

Poursuivre l'activité contenue dans ce document : Activite_SQL/SQL_TP1.ipynb (Activite_SQL/SQL_TP1.ipynb)

3. Suppression

- Une fois qu'une table est créée, il n'est pas possible d'en créer une autre avec le même nom. Par exemple, si l'on souhaite modifier certaines contraintes des attributs, il faut d'abord supprimer la table portant le même nom.
- L'instruction qui permet de supprimer une table est `DROP TABLE` suivie du nom de la table à supprimer.
  - Exemple : `DROP TABLE CLIENT;` supprime la table `CLIENT`
- ATTENTION ! :
  - On ne peut pas supprimer des tables dans n'importe quel ordre
  - En effet, il n'est pas possible de supprimer une table qui sert de référence pour un clé étrangère d'une autre table
    - Exemple : on ne peut pas supprimer la table `CLIENT` sans avoir supprimé le table `FACTURE` à cause de l'attribut `ref_Cli`

Exercice 6 :

On veut supprimer toute la base précédente, écrire les ordres qui permettent la suppression des quatre tables contenues dans celle-ci.

Réponse :

Exercice 7 :

Tester cette suppression en terminant l'activité contenue dans ce document : Activite_SQL/SQL_TP1.ipynb (Activite_SQL/SQL_TP1.ipynb)

3. S'entraîner avec SQL

- Dans une prochaine feuille, nous effectuerons des requêtes en langage SQL pour interroger des bases et sélectionner des données.
- Il existe de nombreux SGBD(Système de Gestion de Base de Données).Si le langage SQL est standardisé, il est difficile de le maîtriser car ses réactions peuvent différer d'un système à l'autre.
- Voici néanmoins quelques sites qui permettent de s'entraîner à ce langage :
  - https://www.w3schools.com/sql/sql_intro.asp (https://www.w3schools.com/sql/sql_intro.asp)
  - https://sqlzoo.net/ (https://sqlzoo.net/)
  - http://webtic.free.fr/sql/exint/q1.htm (http://webtic.free.fr/sql/exint/q1.htm)