R3.07 – SQL dans un langage de programmation

TD2 - Modélisation / LDD

Afin d'aider les techniciens en informatique de l'IUT, il vous est demandé de créer une base de données permettant de gérer les machines virtuelles (VM) créées et leurs versions.

Remarque : Dans un souci de simplification, nous ne gérerons pas les salles dans lesquelles les VM sont déployées, ni les déploiements.

Une machine virtuelle a une plateforme de virtualisation dédiée (Microsoft Hyper-v, Virtualbox, Vmware, Microsoft VirtualPC, etc.) peu importe la version de celle-ci, un système d'exploitation, un nom, une description, un demandeur, une taille du disque dur virtuel (en Go), un type de stockage (taille fixe ou taille dynamiquement allouée) et une taille de mémoire RAM nécessaire (en Mo). Exemple :

N° VM	Nom VM	Logiciel de virtualisation	OS	Description	Demandeur	Taille DD (en Go)	Type stockage	Mémoire RAM (en Mo)
1	Visual Studio	VMWare	Windows 10 64 bits	Développement .NET module R3.04	V. Couturier	100	Dynamiquement alloué	8192
2	TPs systèmes	VirtualBox	Ubuntu 64 bits	TPs systèmes INFO1	B. Diard	10	Taille fixe	2048
Etc.								

Pour des questions de performance (et surtout d'un point de vue pédagogique !), il a été décidé de limiter la mémoire RAM des VM à 16384 Mo au maximum.

Un système d'exploitation a un type. Exemples :

Type	OS
Microsoft Windows	Windows 7 32 bits
Microsoft Windows	Windows 7 64 bits
Microsoft Windows	Windows 10 32 bits
Microsoft Windows	Windows 10 64 bits
Microsoft Windows	Windows 11 64 bits
…	
Linux	Ubuntu 32 bits
Linux	Ubuntu 64 bits
Linux	
…	
BSD	…

Pour chaque demandeur, on désire sauvegarder son nom, prénom, adresse, mail, tel. fixe auquel le contacter. Pour des soucis de simplification, seules les adresses françaises sont enregistrées. L'adresse n'est pas forcément saisie.

Il n'est pas possible de changer la version d'OS. Par exemple, une « mise à jour » majeure de Windows 7 vers Windows 8 correspond à la création d'une nouvelle VM, même si l'on part d'une existante pour la réaliser.

La machine virtuelle a une ou plusieurs versions dépendant des logiciels qui y sont installés. Un logiciel est un programme dans sa version. Par exemple, PostgreSQL 14 et PostgreSQL 13 seront considérés comme 2 logiciels différents.
Ainsi, une première version de la VM nommée « Visual Studio » a été créée par exemple le 01/09/2020 contenant Microsoft Visual Studio 2019 et Microsoft SQL Server 2016 (Cf. tableau suivant). Une mise à

jour – version n°2 de la VM – a été réalisée le 20/09/2021. Cette version a consisté à passer Microsoft SQL Server en version 2019 et à ajouter PostgreSQL 12.
Le numéro de version de la machine virtuelle dépend également du numéro de la VM.

N° VM	Nom VM	N° version VM	Date version	Logiciels installés
1	Visual Studio	1	01/09/2020	- Microsoft Visual Studio 2019 - Microsoft SQL Server 2016
		2	20/09/2021	- Microsoft Visual Studio 2019 - Microsoft SQL Server 2019 - PostgreSQL 12
		3	20/09/2021	- Microsoft Visual Studio 2019 - Microsoft SQL Server 2019 - PostgreSQL 13
Etc.				
10	SGBD Oracle	1	15/08/2018	- Oracle Database 12c R2
		2	01/02/2022	- Oracle Database 19c

Q1 :

a. A partir de l'exemple fourni dans le document « Rappel Elaboration MCD », réaliser le MCD.

b. Après discussion avec les demandeurs, les besoins suivants ont été oubliés :

- La gestion des départements a été oubliée. Comment modéliseriez-vous le tableau suivant ?

N° VM	Nom VM	Logiciel de virtualisation	OS	Description	Demandeur	Département	UFR	Taille DD (en Go)	Type stockage	Mémoire RAM (en Mo)
1	Visual Studio	VMWare	Windows 10 64 bits	Développement .NET module R3.04	V. Couturier	INFO	IUT A	100	Dynamiquement alloué	8192
2	TPs systèmes	VirtualBox	Ubuntu 64 bits	TPs systèmes INFO1	B. Diard	INFO	IUT A	10	Taille fixe	2048
3	TPs sécurité	VirtualBox	Ubuntu 64 bits	Sécurité Linux	B. Dupont	R&T	IUT A	20	Taille fixe	512
Etc.										

- On nous apprend que les demandeurs peuvent changer de département. Cela affecte-t-il le modèle ?

- La gestion des demandeurs est plus complexe qu'il n'y paraît. En effet, il y a 2 types de demandeurs : vacataire et permanent. L'adresse et le téléphone fixe ne sont saisis que pour les vacataires. On souhaite également enregistrer le nom de leur entreprise. Pour les permanents, on souhaite enregistrer le numéro de bureau. Proposer une modification au modèle.

- Un logiciel n'est compatible que sur certains OS. On souhaite gérer cette compatibilité afin de ne pas saisir d'erreur dans les logiciels installés. Exemple :

Nom logiciel	Noms OS
PowerAMC 15	Windows 7, Windows 10, Windows 11
PowerAMC 16	Windows 7, Windows 10, Windows 11
Microsoft Office 2013	Windows 7, Windows 10, Windows 11
Microsoft Office 2011	macOS
Microsoft Office 365	Windows 7, Windows 10, Windows 11, macOS

c. Réaliser le MLD relationnel.

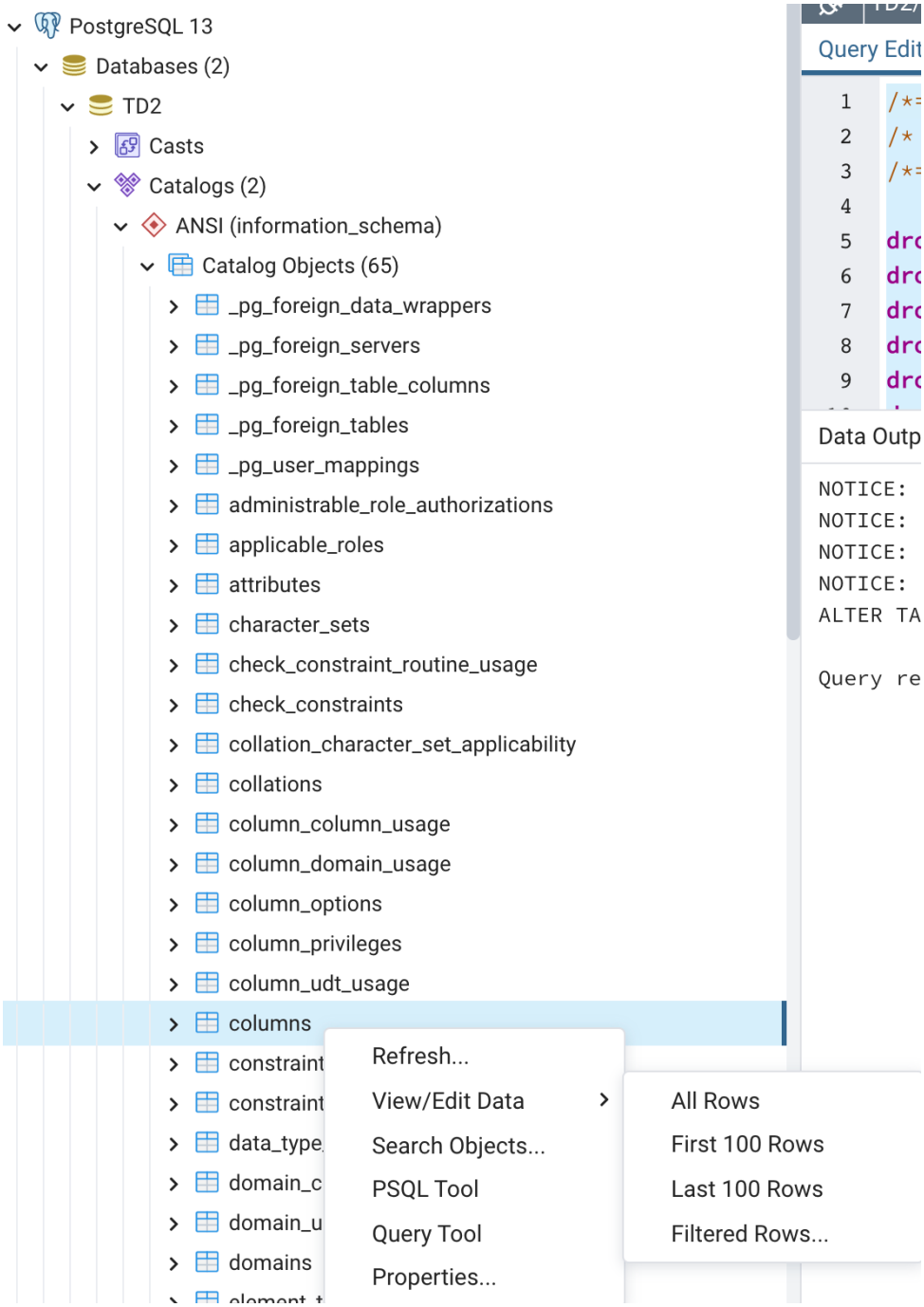
d. Réaliser le MPD.
   Quelques règles simples d'optimisation à appliquer pour créer le MPD :
   - Conservation ou non des tables de référence, i.e. tables en général ayant peu de données et dont les celles-ci évolueront peu ou pas ? Si le contenu de celles-ci n'évolue pas et n'évoluera jamais dans le futur, il n'est pas nécessaire de les conserver. Les tables de référence qui n'évolueront pas seront remplacées par des contraintes CHECK afin de ne pas pouvoir saisir n'importe quoi et dans l'application cliente par des listes déroulantes dont les éléments sont fixes.
   - Limitation des clés primaires composées à 2 champs au maximum (sinon fichier d'index imposant en mémoire). Comment faire alors pour assurer l'unicité des champs composant la clé primaire si ceux-ci ne le sont plus : en utilisant des contraintes de clé unique.

Pour répondre aux questions suivantes, utilisez les annexes et/ou la doc PostgreSQL (http://docs.postgresqlfr.org/)**.**

Q2 : A partir des fichiers « script Virtualisation sans contrainte.sql » et de la **correction du MPD**, ajouter l'ensemble des contraintes. Les clés primaires, contraintes de validation et clés uniques seront créées dans les `CREATE TABLE` ; les clés étrangères à la fin du script (Cf. Annexe 1).
Indications :
- *Un script commence toujours par des instructions* `DROP TABLE` *permettant de supprimer les tables si celles-ci ont déjà été créées.*
- ***Créer une nouvelle base de données***

Q3 : Schéma d'information
Le schéma d'informations (`information_schema`) (Cf. chapitre 34 de la documentation) consiste en un ensemble de vues systèmes contenant des informations sur les différents objets (tables, contraintes, vues, etc.) définis dans une base de données. Les vues du schéma d'information ne contiennent pas d'information sur les fonctionnalités spécifiques à PostgreSQL ; pour cela, vous devez utiliser le catalogue système propre (`pg_catalog`) ou d'autres vues spécifiques à PostgreSQL (cf. Q4).
Il est toujours nécessaire de bien connaître les vues et/ou tables systèmes quand on utilise un SGBD.

Visualiser les données stockées dans les vues du schéma d'informations (`information_schema`) de la base que vous venez de créer.
Se positionner sur votre base puis *Catalogues > ANSI (Information_schema)* et sélectionner le bon objet puis bouton droit de la souris *View/Edit data > All rows*.



OU écrire directement la requête SQL dans l'éditeur.

Vues système à essayer : `tables`, `columns`, `table_constraints`, `constraint_column_usage`, `constraint_table_usage`, `check_constraints`, `key_column_usage`, `referential_constraints`.
Plus d'informations sur chacune d'elles ici : https://doc.postgresql.fr/13/information-schema.html

Ex. de requêtes SQL :

```
select * from information_schema.tables where table_schema = 'public';
-- Il est nécessaire de préfixer par le nom du schéma, sinon la recherche se fait dans le schéma actuel (i.e. public)
select * from information_schema.constraint_column_usage;
```

A quoi sert la vue système `table_constraints` ? Expliquer son contenu.
Exécuter avec et sans la clause `where` :

```
select * from information_schema.tables;
select * from information_schema.tables where table_schema = 'public';
```

Expliquer la différence entre les vues système `constraint_column_usage` et `constraint_table_usage` :

A quoi sert la vue système `check_constraints` ? Expliquer son contenu.

A quoi sert la vue système `key_column_usage` ? Expliquer son contenu.

Q4 : Catalogue système propriétaire

En plus du schéma `public` et de ceux créés par les utilisateurs, chaque base de données contient un schéma `pg_catalog`. Celui-ci contient les tables systèmes et tous les types de données, fonctions et opérateurs intégrés.

Visualiser les données du catalogue système.
Se positionner sur votre base puis *Catalogues* > *PostgreSQL Catalog (pg_catalog)* > *Views OU Tables* et sélectionner le bon objet puis bouton droit de la souris *View/Edit data* > *All rows* OU écrire la requête SQL : vue `pg_tables`, table `pg_constraint`
Ex. : `select * from pg_tables;`
Puis `select * from pg_tables where schemaname='public';`
Expliquer le contenu de la table `pg_constraint`.

ANNEXE

1. Création d'une table

La création consiste à définir (en fonction de l'analyse) le nom des colonnes, leur type, une valeur par défaut à la création de la ligne (default), les règles de gestion s'appliquant à la colonne (`CONSTRAINT`). Si une règle de gestion concerne plusieurs colonnes de la ligne, on définit une contrainte de table.

Syntaxe

```
CREATE TABLE nom (nomcolonne type [DEFAULT expr]
[[CONSTRAINT nom contrainte-de-colonne...],...
[,CONSTRAINT nom contrainte-de-table...]);
```

a. Contraintes de colonne

NULL/NOT NULL

PRIMARY KEY

UNIQUE

REFERENCES table [(colonne)] [ON DELETE CASCADE]

CHECK (condition)

Remarque : Attention, il n'est pas possible dans une contrainte CHECK de faire référence à la fonction SYSDATE (Oracle) ou bien CURRENT_DATE (PostgreSQL) pour comparer la valeur d'une colonne de type date avec le résultat de l'exécution de cette fonction. Il est par contre tout à fait possible, si on le souhaite, de comparer à l'aide d'une contrainte CHECK, les valeurs de 2 colonnes de type date. Si l'on souhaite comparer la valeur d'une colonne de type date avec la date courante, il est nécessaire de passer par la mise en place d'un déclencheur de base de données. Une autre possibilité consiste à créer une colonne de type date avec une valeur par défaut égale à la date du

jour plus une contrainte CHECK qui compare les valeurs présentes dans les deux colonnes.

b. Contraintes de table (portant sur plusieurs colonnes)

PRIMARY KEY (colonne1,colonne2,…)
Désigne la concaténation des attributs cités comme clé primaire de la table. Cette contrainte ne peut apparaître qu'une seule fois dans l'instruction.

UNIQUE (colonne1,colonne2,…)
Désigne la concaténation des attributs cités comme clé secondaire de la table. Dans ce cas de contrainte UNIQUE portant sur des colonnes concaténées, au moins une des colonnes participant à cette clé secondaire doit permettre de distinguer la ligne. Cette contrainte peut apparaître plusieurs fois dans l'instruction.

FOREIGN KEY (colonne,…) REFERENCES table [(colonne)] [ON DELETE {CASCADE | SET NULL}]
Contrainte d'intégrité référentielle pour l'ensemble des attributs dans la table détail en cours de définition. Les valeurs prises par ces attributs doivent exister dans l'attribut colonne qui possède une contrainte PRIMARY KEY ou UNIQUE dans la table maître table.

CHECK (condition)
Cette contrainte permet d'exprimer une condition qui doit exister entre plusieurs attributs de la ligne.

Remarque : Les contraintes de tables portent sur plusieurs colonnes de la table sur laquelle elles sont définies. Il n'est pas possible de définir une contrainte d'intégrité utilisant des colonnes provenant de deux ou plusieurs tables. Ce type de contrainte sera mis en oeuvre par l'intermédiaire de déclencheurs de base de données (triggers).

c. Complément sur les contraintes

ON DELETE CASCADE
Demande la suppression des lignes dépendantes dans la table en cours de définition, si la ligne contenant la clé primaire correspondante dans la table maître est supprimée. Si cette option n'est pas indiquée, la suppression sera impossible dans la table maître s'il existe des lignes référençant cette valeur de clé primaire.

ON DELETE SET NULL
Demande la mise à NULL des colonnes constituant la clé étrangère qui font référence à la ligne supprimée. Si cette option n'est pas indiquée, la suppression sera impossible dans la table maître s'il existe des lignes référençant cette valeur de clé primaire.

[NOT] DEFERRABLE
Repousse ou non (NOT) la vérification de la contrainte au moment de la validation de la transaction.

d. Dénomination des contraintes

Les contraintes peuvent être nommées afin d'être plus facilement manipulées ultérieurement (activation, suppression). Lors de l'affectation explicite d'un nom à une contrainte, on utilise en général la dénomination suivante :

`Table_Colonne_TypeDeContrainte` OU `TypeDeContrainte_Table_Colonne`

`Table` Nom de la table sur laquelle est définie la contrainte.

`Colonne` Nom de la (ou des) colonne(s) sur laquelle est définie la contrainte.

`TypeDeContrainte` :

`PK`	Clé primaire
`UQ`	Unique
`NN`	Not Null
`CK`	Check
`RF`	References
`FK`	Clé étrangère