420-W23-SF **Bases de données relationnelles**	**Exercices**[1] **Les vues et les tables temporaires** **Travail de préférence individuel.** **Durée :** 2 heures **Environnement :** MySQL Serveur 8.0 **Fichier SQL :** marketing.sql

1. Création de vue

Les vues sont des objets de la base de données, constitués d'un nom et d'une requête de sélection.

Une fois qu'une vue est définie, on peut l'utiliser comme on le ferait avec une table; table qui serait constituée des données sélectionnées par la requête définissant la vue.

a) Création d'une première vue :

```
create view v_simplifierClient as
select c.nom_client, adresse_client, v.ville, c.code_postal_client
from client as c inner join ville as v on c.id_ville = v.id_ville
order by c.nom_client;
select * from v_simplifierClient;
```

b) Création d'une première vue :

```
create view v_nbClientVille as
select v.ville, count(c.id_client)
from client as c inner join ville as v on c.id_ville= v.id_ville
group by v.ville;
select * from v_nbclientville;
```

2. Clause Limit

a) Nous allons utiliser les vues avec la notion de **limit** et **offset.**

[1] Ces exercices ont été préparés par Jean-Pierre Duchesneau

La clause LIMIT est à utiliser dans une requête SQL pour spécifier le nombre maximum de résultats que l'on souhaite obtenir. Cette clause est souvent associée à un OFFSET, c'est-à-dire effectuer un décalage sur le jeu de résultat. Ces 2 clauses permettent par exemple d'effectuer des systèmes de pagination (exemple : récupérer les 10 articles de la page 4)[2]

b) La syntaxe commune aux principaux systèmes de gestion de bases de données est la suivante :

```
11 •  select * from v_simplifierClient limit 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

nom_client	adresse_client	ville	code_postal_client
Académie de l'Entrepreneurship	1001, rue Sherbrooke Est, bureau 350	Montréal	H2L1L3
Académie de l'Entrepreneurship	4660, montée Saint-Hubert, local 104	Longueuil	J3Y1V1
Académie des arts et design	7305, Marie-Victorin	Brossard	J4W1A6
Académie du Savoir	455, rue du Marais	Québec	G1M3A2
Agrinova	640, rue Côté Ouest	Alma	G8B7S8

Cette requête permet de récupérer seulement les 5 premiers résultats d'une table. Bien entendu, si la table contient moins de 5 résultats, alors la requête retournera toutes les lignes.

La bonne pratique lorsque l'on utilise LIMIT consiste à utiliser également la clause ORDER BY pour s'assurer que quoi qu'il en soit-ce sont toujours les bonnes données qui sont présentées. En effet, si le système de tri est non spécifié, alors il est en principe inconnu et les résultats peuvent être imprévisible, mais nous l'avons inséré dans notre vue, donc, il n'est pas nécessaire de l'ajouter.

c) Avec deux arguments, le premier argument spécifie le décalage de la première ligne à retourner et le second spécifie le nombre maximal de lignes à renvoyer. Le décalage de la ligne initiale est 0 (pas 1) :

```
11 •  select * from v_simplifierClient limit 1, 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

nom_client	adresse_client	ville	code_postal_client
Académie de l'Entrepreneurship	4660, montée Saint-Hubert, local 104	Longueuil	J3Y1V1
Académie des arts et design	7305, Marie-Victorin	Brossard	J4W1A6
Académie du Savoir	455, rue du Marais	Québec	G1M3A2

Cette requête vous a renvoyé les enregistrements 2,3 et 4.

[2] Source : https://sql.sh/cours/limit

d) Tapez cette requête pour avoir les enregistrements 7 jusqu'à 11:

```
11 •  select * from v_simplifierClient limit 6, 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

nom_client	adresse_client	ville
Air Richelieu (secteur anglophone)	5800, route de l¿Aéroport	Longueuil
Biopterre - Centre de développement des biopr...	1642, rue de la Ferme	Sainte-Anne-de-la-Pocatière
Campus collégial de Lotbinière	1080, avenue Bergeron	Saint-Agapit
Campus d'Amos	341, rue Principale Nord	Amos
Campus de Montréal / Montreal Campus	1001, rue Sherbrooke Est, bureau 200	Montréal

e) Il peut être intéressant d'utilisé Limit 1 pour afficher la dernière valeur insérée dans une table.

Exemple :

```
12 •  select * from ville order by id_ville desc limit 1;
```

Result Grid | Filter Rows: | Edit: | Export/Impor

id_ville	mcode	ville	regadm	mpopul	date_maj
1131	99140	Ivujivik	Nord-du-Québec (10)	415	2019-02-09 17:12:12

f) Déclaration d'une variable

D'abord nous allons créer une table temporaire tmp_ville et prendre les 10 premières lignes de la table ville.

```
Create temporary table tmp_ville (
id_ville int primary key,
mcode char(8),
ville char(45),
date_maj timestamp default current_timestamp);
insert into tmp_ville
select id_ville, mcode, ville, date_maj from ville limit 10;
```

Ensuite, vous pourriez utiliser cette méthode pour déclarer et utiliser une variable :

```
set @v_id_ville := (select id_ville from tmp_ville order by id_ville desc limit 1);
set @v_mcode := (select sort_order from tc_pays where Commo_name like 'Italy');
insert into tmp_ville(id_ville,mcode,ville) value (@v_id_ville+1, @v_mcode, 'Rome');
insert into tmp_ville(id_ville,mcode,ville) value (@v_id_ville+5, @v_mcode, 'Milano');
select * from tmp_ville order by id_ville desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_ville	mcode	ville	date_maj
15	82	Milano	2020-09-01 10:23:52
11	82	Rome	2020-09-01 10:15:25
10	3015	Petite-Vallée	2019-02-09 17:12:12
9	3010	Cloridorme	2019-02-09 17:12:12

Performance

Certains développeurs pensent à tort que l'utilisation de LIMIT permet de réduire le temps d'exécution d'une requête. Or, le temps d'exécution est sensiblement le même car la requête va permettre de récupérer toutes les lignes (donc temps d'exécution identique) PUIS seulement les résultats définis par LIMIT et OFFSET seront retournés. Au mieux, utiliser LIMIT permet de réduire le temps d'affichage car il y a moins de lignes à afficher.

3. Les tables temporaires

3.1 Créer une table à partir de la structure d'une autre

Il est possible de créer la copie exacte d'une table, en utilisant la commande suivante :

```
Create temporary table tmp_contrat like contrat;
describe tmp_contrat;
describe contrat;
select * from tmp_contrat;
```

Si l'on crée une table temporaire avec cette commande, tous les attributs de la table d'origine seront conservés, sauf les clés étrangères, puisqu'on ne peut avoir de clé étrangère dans une table temporaire. À noter que cette commande ne permet pas de copier les données!

Maintenant, ajoutons des données dans la table tmp_contrat :

```
insert into tmp_contrat
select * from contrat
where id_representant=1;
select * from tmp_contrat;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_contrat	date_contrat	date_maj	id_representant	id_contact
27	2018-07-30	2019-02-11 14:42:20	1	84
30	2018-10-16	2019-02-11 14:42:20	1	10

Il est possible d'ajouter des enregistrements dans une table temporaire :

```
insert into tmp_contrat( date_contrat, id_representant, id_contact)
values ('2019-03-08',1,1999),
       ('2019-03-8',2,2),
       ('2020-08-23',2,13);
select * from tmp_contrat order by id_contrat desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_contrat	date_contrat	date_maj	id_representant	id_contact
375	2020-08-23	2020-09-01 10:56:29	2	13
374	2019-03-08	2020-09-01 10:56:29	2	2
373	2019-03-08	2020-09-01 10:56:29	1	1999
372	2018-01-26	2019-02-12 10:58:12	1	115
368	2019-01-24	2019-02-12 10:58:12	1	54

Il faut remarquer que la table n'a pas d'intégrité référentielle, car aucun client n'a le numéro 1999.

Vous pouvez aussi modifier, supprimer des enregistrements et même supprimer la table tmp_contrat.

```
update tmp_contrat set date_contrat='2020-07-25' where id_contrat=375;
Drop table tmp_contrat;
```

3.2 Créer une table à partir de données sélectionnées

Essayons la deuxième méthode de création de tables temporaire. Nous allons créer la nouvelle table temporaire depuis la table **contrat** :

```
15 •  Create temporary table tmp_contrat
16    select * from contrat;
17 •  describe tmp_contrat;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id_contrat	int(11)	NO		0	NULL
date_contrat	date	YES		NULL	NULL
date_maj	timestamp	NO		CURRENT_TIMESTAMP	NULL
id_representant	int(11)	NO		NULL	NULL
id_contact	int(11)	NO		NULL	NULL

```
19 •  describe contrat;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id_contrat	int(11)	NO	PRI	NULL	auto_increment
date_contrat	date	YES		NULL	
date_maj	timestamp	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED
id_representant	int(11)	NO	MUL	NULL	
id_contact	int(11)	NO	MUL	NULL	

Observez les différences entre la table contrat et tmp_contrat.

Les types de colonnes et les contraintes NOT NULL sont les mêmes, la valeur par défaut CURRENT_ TIMESTAMP est encore présente, mais les index ont disparu ainsi que l'auto-incrémentation.

Forcer le type des colonnes

On peut laisser MySQL déduire le type des colonnes du SELECT, mais il est également possible de préciser le type que l'on désire, en faisant attention à la compatibilité entre les types que l'on précise et les colonnes sélectionnées. On peut également préciser les index désirés, les clés et l'éventuelle colonne à auto-incrémenter.

La syntaxe est alors similaire à un CREATE [TEMPORARY] TABLE classique, si ce n'est qu'on rajoute une requête de sélection à la suite.

Attention à l'instruction SQL_MODE voire la note de bas de page[3]

```
5  Drop table if exists tmp_contrat;
6  set SQL_MODE='ALLOW_INVALID_DATES';
7  CREATE TEMPORARY TABLE tmp_contrat(
8  id_contrat int unsigned primary key auto_increment,
9  date_contrat date not null default '0000-00-00',
10 date_maj datetime not null default current_timestamp,
11 id_representant int not null,
12 id_contact int not null);
13 describe tmp_contrat;
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id_contrat	int(10) unsigned	NO	PRI	NULL	auto_increment
date_contrat	date	NO		0000-00-00	NULL
date_maj	datetime	NO		CURRENT_TIMESTAMP	NULL
id_representant	int(11)	NO		NULL	NULL
id_contact	int(11)	NO		NULL	NULL

[3] Le mode NO_ZERO_DATE détermine si le serveur autorise ou non '0000-00-00' comme date valide. Son effet dépend également de l'activation du mode SQL strict.
Donc ici nous modifions la variable SQL_MODE de façon temporaire (session) pour permettre l'utilisation de la valeur par défaut 0000-00-00

```
-- insérons des données qui respectent les contraites d'intégrités
insert into tmp_contrat (date_contrat, id_representant, id_contact)
values ('2018-09-12',7,9);
-- sans respecter l'intégrité des données, l'id de representant et le
-- contact n'existent pas!
insert into tmp_contrat (date_contrat, id_representant, id_contact)
values ('2019-09-12',999,999);
select * from tmp_contrat;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_contrat	date_contrat	date_maj	id_representant	id_contact
1	2018-09-12	2020-09-01 11:44:00	7	9
2	2019-09-12	2020-09-01 11:44:34	999	999

Il est donc possible d'insérer des données, et ce, même si ces données ne respectent pas l'intégrité référentielle.

Maintenant, utilisons une fonction date et deux sous-requêtes pour insérer un contrat pour le représentant Jean-Pierre Duchesneau avec pour contact du client Thomas Greene.

D'abord on doit chercher l'id du représentant et l'id du contact!

```
select id_representant from representant
 where nom_rep like 'Duchesneau' and prenom_rep like 'jean-Pierre%';
 select id_contact from contact where nom_contact like 'Greene'
 and prenom_contact like 'Thomas';
```

Mettez le tout dans une même requête :

```
insert into tmp_contrat (date_contrat, id_representant, id_contact)
values (adddate(curdate(),-23),(select id_representant from representant
 where nom_rep like 'Duchesneau' and prenom_rep like 'jean-Pierre%'),
 (select id_contact from contact where nom_contact like 'Greene'
 and prenom_contact like 'Thomas'));
```

Testez à nouveau la même requête deux ou trois fois. Vérifier les résultats :

```
select * from tmp_contrat;
```

En conclusion, si l'on crée une table temporaire avec cette commande, tous les attributs de la table d'origine seront conservés, sauf les clés étrangères, puisqu'on ne peut avoir de clé étrangère dans une table temporaire.

Vous pouvez bien sûr, jumeler les données d'une vue avec une table temporaire et table permanente :

```
create view v_resumeClient as
select c.id_client,c.nom_client as Nom,adresse_client As Adresse,
v.ville As Ville,c.code_postal_client As CP
from client as c inner join ville as v
on c.id_ville=v.id_ville
order by c.nom_client;


select cli.Nom, cli.Adresse, cli.Ville, cli.CP,
concat(nom_contact,',',prenom_contact) As Contact, date_Contrat
from v_resumeClient as cli inner join contact as c
on cli.id_client=c.id_client
inner join tmp_contrat as ct
on c.id_contact=ct.id_contact
order by cli.nom and date_contrat desc;
```

Résultats :

Nom	Adresse	Ville	CP	Contact	date_Contrat
Collège Salette inc.	418, rue Sherbrooke Est	Montréal	H2L1J6	Valencia,Colby	2018-09-12
Cargair ltée (secteur anglophone)	6100, chemin de l'Aéroport	Longueuil	J3Y8Y9	Greene,Thomas	2020-08-09
Cargair ltée (secteur anglophone)	6100, chemin de l'Aéroport	Longueuil	J3Y8Y9	Greene,Thomas	2020-08-09