Exercice 1 : requêtes simples

Pour traiter de la vente par correspondance on considère la modélisation du TP précédent (prenez un cas réel que vous connaissez) auquel on a ajouté des attributs de type `DATE` :

```
CLIENT(#client, nom, prenom, adresse, codePostal)
PRODUIT(#produit, designation, prixUnitaire)
FOURNISSEUR(#fournisseur, raisonSociale)
COMMANDE(#client, #produit, date, quantite)
FOURNIT(#fournisseur, #produit)
```

pour lesquels chaque numéro (noté `#`) identifie de façon unique un n-uplet. On rapelle que chaque numéro est sur 3 chiffres, que toute contrainte est identifiée à l'aide d'un nom et qu'un client commande des produits dans une quantité donnée, que chaque produit est fourni par un unique fournisseur et qu'un fournisseur peut fournir plusieurs produits. La table des commande renseigne les commandes non traitées.

Exprimez les requêtes suivantes en `SQL` en vous servant des schémas et instances créés au TP précédent. Lorsqu'il est indiqué de se servir du résultat d'une requête précédente, il ne s'agit ici que d'utiliser la **valeur** précédement trouvée/affichée et pas d'utiliser une sous-requête.

1. Donnez la liste des fournisseurs (numéro et raison sociale).
2. Donnez la liste des produits (numéro, désignation et prix unitaire), classée par ordre croissant de prix unitaire ? (Pour la suite gardez en mémoire le numéro de produit de la première ligne)
3. Quels sont les numeros de produits en commande ?
4. Quels sont les noms et prénoms des clients ?
5. Quel est le fournisseur (et le numéro de produit) du produit de prix unitaire le plus bas (en vous servant du résultat d'une requête précédente) ? (dans la suite on l'appelera F) Est-il possible d'avoir plusieurs de ces fournisseurs ? Est-il possible d'avoir plusieurs de ces produits ? Conclusion ?
6. Quel sont les numéros de produit (et le numéro de fournisseur) fournis par le fournisseur F ?
7. Choisissez deux numéros de produits. Quels en sont leurs fournisseurs (et le numéro de produit) ? Pouvez-vous formuler une autre requête pour cette question ?
8. Quel sont les numéros de produit (SANS `#fournisseur`) fournis par le fournisseur F ?
9. Quel sont les noms des clients qui habitent en Seine-Saint-Denis ?
10. Quel sont les numéros de fournisseurs dont la raison sociale contient `'SARL'` ?
11. Quel est le numéro du client qui a commandé le produit de prix unitaire le plus bas (en vous servant du résultat d'une requête précédente) ? Est-il possible d'obtenir plusieurs de ces numéros ? Est-il possible d'avoir plusieurs de ces produits ? Conclusion ?
12. Quels sont les numéros de clients (et les autres indications) qui ont passés une commande, classés par ordre décroissant de numéro de client ? De même classés par ordre décroissant de date de commande ? De même classés par ordre décroissant de quantité ? Exprimez les mêmes requêtes en n'indiquant que les numéros de clients mais pas les autres indications. Conclusion ?
13. Quel sont les numéros de clients qui n'ont pas de commande en cours ?
14. Quels sont les numéros de produits en commande dont le prix unitaire est au moins 10 euros ?

Exercice 2 optionnel : formatage (exercice pour répondre à la demande de certains)

Sur le schéma de base de données relationnelle de votre choix du TP précédent et après avoir cherché et décrit l'effet de la commande suivante, l'utiliser :
`COLUMN nom_de_colonne [FORMAT masque] HEADING titre_de_colonne TRUNCATE/WRAP`

Exemple : `COLUMN Nom FORMAT a4 HEADING 'Nom|Prenom' WRAP`
Exemple : `COLUMN Noss FORMAT 999.999.999 HEADING 'Numero|securite|sociale' TRUNCATE`

Expérimentez chacune des commandes SQL*PLUS suivantes :

`SQL> COLUMN`	affiche le formatage en cours
`SQL> COLUMN nom_de_colonne`	affiche le formatage en cours de la colonne
`SQL> COLUMN nom_de_colonne CLEAR`	efface le formatage en cours de la colonne
`SQL> COLUMN nom_de_colonne ON/OFF`	désactive temporairement/réactive
`SQL> CLEAR COLUMN`	efface tous les formatages en cours

Pour faire des rapports en SQL*PLUS, essayez chacune des commandes suivantes :

`set underline un_caractere`	définition des séparateurs dans une table
`set underline '-'`	
`set pagesize un_nombre`	nombre de ligne par page
`set linesize un_nombre`	nombre de caractères par ligne
`spool nom_de_fichier`	redirection de SQLPLUS
`spoll off`	fin de redirection
`set termout on/off`	desactive/active l'affichage à l'écran
`set echo on/off`	
`ttitle nom_d_entete_de_rapport`	entête
`btitle nom_de_pied_de_page_de_rapport`	pieds de page
`show all`	description des variables d'environnement

Exercice 3 : requêtes standards

Soit le schéma de base de données relationnelle du TP précédent :

```
EMP (nom, num, fonction, n_sup, embauche, salaire, comm, n_dept)
DEPT (n_dept, nom, lieu)
GRAD (grade, salmin, salmax)
```

Sachant que :
- Un employé de numéro `num` a un `nom` occupe l'emploi `fonction`, a un responsable identifié par le numéro `n_sup`, une date d'embauche `embauche`, un `salaire`, une commission `comm` et un département de rattachement `n_dept`,
- Un Département de numéro `n_dept`, de nom `nom` est situé à `lieu`,
- A un grade donné correspond un salaire minimal et un salaire maximal.

Vous effectuerez quand cela est possible les requêtes suivantes en algèbre relationnelle, mais contentez vous de ne traiter que les requêtes non imbriquées en SQL (c'est à dire dont la condition de `WHERE` n'utilise pas de clause `SELECT`).

1. Quelle est la liste des noms d'employés ayant une commission, classée par commission décroissante ?
2. Donner tous les noms d'employés ainsi que leur fonction et leur salaire, classés par fonction en ordre croissant, et pour chaque fonction classés par salaire décroissant.
3. Quel est le nom des personnes embauchées depuis janvier 2001 ?
4. Donner pour chaque employé son nom et son lieu de travail.

5. Donner pour chaque employé le nom de son supérieur hiérarchique.
6. Quels sont les employés ayant la même fonction que "CODD" ?
7. Quels sont les employés gagnant plus que tous les employés du département 30 ?
8. Quels sont les employés ne travaillant pas dans le même département que leur supérieur hiérarchique ?
9. Quels sont les employés travaillant dans un département qui a procédé à des embauches depuis le début de l'année 2008 ?
10. Donner le nom, la fonction et le salaire de l'employé (ou des employés) ayant le salaire le plus élevé.
11. Quel est le total des salaires pour chaque département ?
12. Donner le nombre d'ingénieurs ou de commerciaux des départements ayant au moins deux employés de ces catégories.
13. Quel est le département ayant le plus d'employés ?

Exercice 4 :

On considère la base de données relationnelle `COLLEGE` suivante pour laquelle on a :
– Les éléves et les professeurs sont identifiés par des numéros (`NumE, NumP`),
– Les numéros de classes sont définies parmis un ensemble d'éléments composés d'un chiffre et d'une lettre,
– `NoteMoy` désigne la moyenne trimestrielle d'une éléve dans une matière,
– Si vous avez un peu de temps à la fin du TP, vous ajouterez les contraintes de clefs étrangères pour `ENSEIGNE` et `RESULTATS`.

`ELEVES`(NumE, Nom, Prénom , Classe)
`COURS`(Matière, Classe, NbH)
`RESULTATS`(NumE, Matière, Trimestre, NoteMoy)
`PROFESSEURS`(NumP, NomP, Statut)
`ENSEIGNE`(NumP, Matière, Classe)

Exprimer chacune des requêtes suivantes en `SQL` éventuellement en vous aidant des requêtes en algèbre relationnelle :

1. Quels sont les noms et prénoms de tous les élèves ?
2. Quels sont les professeurs qui enseignent en classe de 5ème A (noms) ?
3. Qui est le professeur de mathématiques de 5ème A (noms et numéros) ?
4. Quelles sont les notes de mathématiques des élèves de 5ème A (numéro, nom, prénom et note) ?
5. Quels sont les professeurs qui ont le même nom qu'un des élèves (noms et numéros) ?
6. Quels sont les professeurs qui enseignent dans une des classes de 5ème (noms et numéros) ?
7. Quelles sont les classes dans lesquelles enseigne le professeur de mathématiques de la 5ème A ?
8. Quelles sont les matières dans lesquelles l'élève de numèro 150 n'a pas la moyenne en troisième trimestre ?
9. Quels sont les maîtres auxiliaires qui enseignent dans une des 3ème (noms et numéros) ?
10. Quel est l'emploi du temps de `Madame Topin`, numéro `115` (Matière , classe, nombre d'heures) ?

Annexe pour le TP : la clause `SELECT`

Voici une partie de la clause `SELECT` :

- `SELECT [ALL|DISTINCT] colonne | (fct(colonne) [AS nom]) [, liste_col_ou_fct(col)]`
  Liste de colonne ou fonctions sur une colonne (`AVG|MAX|MIN|SUM [ALL|DISTINCT]`) ou sur le nombre de ligne d'une table (`COUNT [ALL|DISTINCT]`).

- `FROM nom_de_table [alias] [,liste_de_nom_de_table]`
  Indique les tables à partir desquelles récupérer les données. Un alias permet un renommage local. On utilise l'alias à la place du nom de la table afin d'enlever des ambiguités, de simplifier l'écriture ou d'être plus explicite.

- `[WHERE condition]`
  Exemples d'opérateurs utilisables dans une condition :
  - opérateurs logiques (`AND, OR, NOT, IS NULL, IS NOT NULL`)
  - opérateurs de chaines (`[NOT] IN|BETWEEN|LIKE`)
  - opérateurs arithmétiques (`+, -, *, /, mod(,)`)
  - comparateurs arithmétiques (`<, >, =, <>, <=, >=, [NOT] IN`)

- `[ORDER BY colonne [ASC|DESC] [,list_col]]`
  Tri des résultats selon une (des) colonne(s). Toutes colonnes du select qui n'est pas sous une fonction doit etre spécifiée dans la liste des colonnes de `ORDER BY` si on souhaite effectuer un tri.

Clauses obtenues par opération emsembliste :
`(clause_select) UNION [ALL] (clause_select)`
`(clause_select) INTERSECT [ALL] (clause_select)`
`(clause_select) MINUS [ALL] (clause_select)`
Sans l'option `ALL` seuls les n-uplets distinct sont conservés (à tester : support de `ALL` sous Oracle 9i pour les deux dernières opérations). Rappel : il faut respecter les types (i.e. les domaines en algèbre relationnelle) colonne à colonne.

NB : pour ceux qui ont besoin d'aller un peu plus loin (il n'est normalement pas nécéssaire de le faire pour CE TP), allez à la dernière la page.

Annexe pour le prochain TP :

Voici une partie plus importante de la clause `SELECT` :

- `SELECT [ALL|DISTINCT] colonne | (fct(colonne) [AS nom]) [, liste_col_ou_fct(col)]`
  Liste de colonne ou fonctions sur une colonne (`AVG|MAX|MIN|SUM [ALL|DISTINCT]`) ou sur le nombre de ligne d'une table (`COUNT [ALL|DISTINCT]`). S'il y a des regroupements de colonnes `GROUP BY` alors les fonctions se font sur chaque regroupement (`AVG, MAX, MIN`) ou sur le nombre de ligne de chaque regroupement (`COUNT`).

- `FROM nom_de_table [alias] [,liste_de_nom_de_table]`
  Indique les tables à partir desquelles récupérer les données. Un alias permet un renommage local. On utilise l'alias à la place du nom de la table afin d'enlever des ambiguités, de simplifier l'écriture ou d'être plus explicite.

- `[WHERE condition]`

  Exemples d'opérateurs utilisables dans une condition :
  - opérateurs logiques (`AND, OR, NOT, IS NULL, IS NOT NULL`)
  - opérateurs de chaines (`[NOT] IN|BETWEEN|LIKE`)
  - opérateurs arithmétiques (`+, -, *, /, mod(,)`)
  - comparateurs arithmétiques (`<, >, =, <>, <=, >=, [NOT] IN`)

  Conditions avec sous-requêtes :
  `WHERE [NOT] EXISTS|IN (clause_select)`
  `WHERE colonne | (fct(colonne) [AS nom]) op [ALL|ANY] (clause_select)`
  avec
  `ALL` : la condition est vraie seulement si elle est vraie pour toutes les valeurs de la clause select. Donc la condition est vraie si la sous-requête est vide. Donc : `<ALL` = moins que le minimum, `>ALL` = plus que le maximum, `<>ALL` equivalent à `NOT IN`.
  `ANY` : la condition est vraie seulement si elle est vraie pour au moins l'une les valeurs de la clause select. Donc : `<ANY` = moins que le maximum, `>ANY` = plus que le minimum, `=ANY` equivalent à `IN`.

- `GROUP BY colonne [,list_col]`
  Regroupe des résultats selon une(des) colonne(s).

- `HAVING condition`
  est à `GROUP BY` ce que `WHERE` est à `FROM`.

- `[ORDER BY colonne [ASC|DESC] [,list_col]]`
  Tri des résultats selon une (des) colonne(s). Toutes colonnes du select qui n'est pas sous une fonction doit etre spécifiée dans la liste des colonnes de `ORDER BY` si on souhaite effectuer un tri.

Clauses obtenues par opération emsembliste :
`(clause_select) UNION [ALL] (clause_select)`
`(clause_select) INTERSECT [ALL] (clause_select)`
`(clause_select) MINUS [ALL] (clause_select)`
Sans l'option `ALL` seuls les n-uplets distinct sont conservés (à tester : support de `ALL` sous Oracle 9i pour les deux dernières opérations). Rappel : il faut respecter les types (i.e. les domaines en algèbre relationnelle) colonne à colonne.