BASES DE DONNEES

Langage SQL
Correction

Application BOTANIQUE

⇨ SCHEMA RELATIONNEL DE L'APPLICATION

PLANTE (NOMSCIENT, NOMCOMMUN, COULEURFLEUR, DATEFLORMIN, DATEFLORMAX, ESPECE, DESCRIPTIF)

ZONEPEUP (CADASTRE, LIEUDIT, COMMUNE, DEPARTEMENT, SOL, INCLINAISON, TYPEARBRE, ALTMIN, ALTMAX)

AGENT (MATRICULE, NOM, PRENOM, RUE, CODEPOST, VILLE, MEL, TEL)

PRESCRIPTEUR (NUMADMIN, ORGANISME, SERVICE, RUE, CODEPOST, VILLE, MEL, TEL)

RELEVE (***LAPERS, LEREC, LAPLANTE, LAZONE***, QUANTITE, ETATM, TAILLEM)

RECENSEMENT (NUMERO, DATEDEM, DATERECENS, ***CODEPRESC***, ***CODERESP***)

La clé primaire est soulignée ; les clés étrangères sont indiquées en italique.

⇨ DEFINITION DES RELATIONS

RE3.01) **Créer les relations PLANTE, RECENSEMENT et AGENT.**

--SCRIPT DE CREATION DES TABLES DE L'APPLICATION BOTANIQUE

```
CREATE TABLE plante
  (
  nomscient       VARCHAR(30),
  nomcommun       VARCHAR(35)  NOT NULL,
  couleurfleur    VARCHAR(18),
  dateflormin     DATE,
  dateflormax     DATE,
  espece          VARCHAR(30)  NOT NULL,
  descriptif      LONG,
  CONSTRAINT pkplante PRIMARY KEY (nomscient),
  CONSTRAINT ckdateflor CHECK (dateflormax >= dateflormin)
  ) ;
```

```
CREATE TABLE recensement
  (
  numero        CHAR(11)      NOT NULL,
  datedem       DATE          NOT NULL,
  daterecens    DATE          NOT NULL,
  codepresc     CHAR(5)       NOT NULL,
  coderesp      CHAR(5)       NOT NULL,
  CONSTRAINT pkrecensement PRIMARY KEY (numero),
  CONSTRAINT fkrecens_presc FOREIGN KEY (codepresc)     REFERENCES  prescripteur  (numadmin),
  CONSTRAINT fkrecens_agent FOREIGN KEY (coderesp)      REFERENCES  agent  (matricule)
  ) ;

CREATE TABLE agent
  (
  matricule     CHAR(5),
  nom           VARCHAR(25)  NOT NULL,
  prenom        VARCHAR(25),
  rue           VARCHAR(50)  NOT NULL,
  codepost      CHAR(5)      NOT NULL,
  ville         VARCHAR(50)  NOT NULL,
  mel           VARCHAR(28),
  tel           CHAR(10),
  CONSTRAINT pkagent PRIMARY KEY (matricule)
  ) ;
```

RE3.02) **L'attribut MEL dans la relation AGENT passe maintenant à 30 caractères maximum.**

```
ALTER TABLE agent
MODIFY (mel VARCHAR(30)) ;
```

RE3.03) **L'attribut TEL dans la relation AGENT passe maintenant à 8 caractères au lieu de 10.**

Impossible, risque de perte d'information, opération non réalisable par le SGBD.

RE3.04) **Ajouter l'attribut NB_RELEVES à la relation RECENSEMENT. Cet attribut va permettre de disposer du nombre de relevés effectués pour un recensement. Quels sont vos commentaires ?**

```
ALTER TABLE recensement
ADD (nb_releves NUMERIC(3)) ;
```

RE3.05) **Proposer la requête de mise à jour de cet attribut NB_RELEVES.**

```
UPDATE recensement
SET nb_releves =  (
                  SELECT COUNT(*)
                  FROM releve
                  WHERE lerec = numero ) ;
COMMIT ;
```

RE3.06) **Supprimer de la relation RECENSEMENT l'attribut NB_RELEVES?**

```
ALTER TABLE recensement
DROP COLUMN nb_releves ;
```

⇨ INDEX

RE3.07) **Définir quelques index. Expliquer les raisons de vos choix.**

```
--3 types d'index

--Index sur les clés primaires :
CREATE UNIQUE INDEX iu_agent ON agent (matricule) ;

--Index sur les attributs faisant souvent objet de jointure. Typiquement les clés étrangères :
CREATE INDEX i_relevezone ON releve (lazone) ;

--Index sur les attributs non clés pouvant être impliqué par l'applicatif dans une clause where :
CREATE INDEX i_plantenom ON plante (nomcommun) ;
```

RE3.08) **Quel est le principe de fonctionnement d'un index ? Comment sont-ils utilisés dans une application ?**

L'index = une structure arborescente assurant une recherche beaucoup plus courte.
L'utilisation d'un index est transparente et le choix de son utilisation s'il existe est laissé au moteur de recherche.

RE3.09) **Un index mal conçu peut-il aller jusqu'à dégrader les performances d'une base de données ?**

Oui, car la structure arborescente doit être équilibrée et cela demande des ajustements lors des mises à jours.
On ne va donc pas placer d'index sur des attributs souvent mis à jour.

⇨ SCRIPTS SQL

RE3.10) **Accédez au répertoire BD sur le réseau LPD2i pour récupérer les scripts de création des relations de l'application Natura ainsi que le jeu de données. Exécuter ces requêtes. Cela permettra de disposer du même jeu de données pour toutes les requêtes d'interrogation.**

```
START "unite:\chemin\bdnatura_struct.sql"
START "unite:\chemin\bdnatura_data.sql"
```

⇨ MISES A JOUR

RE3.11) **Lors de la venue d'un membre de l'ONF au siège de Natura à Saint-Affrique (12), un recensement a été défini. Il est prévu le 14 octobre 2008. Tout de suite, deux agents y ont été affectés. Comme ce recensement devrait être effectué près de Najac (12), Natura a sollicité Marcel Fraysse qui habite tout proche à Villefranche de Rouergue, 28 rue de l'optimisation, 12200. Son portable est le 06.21.12.12.12. Sylvie Marcenac est très heureuse de faire partie de la mission aussi. Elle habite au 44 rue de la transaction à Féneyrols 82400. Pour la joindre, il vaut mieux l'appeler le soir au 06.82.40.00.82. Tous deux ont une adresse mail, respectivement marcel@villef.cty et sissi_m@lpd2i.edu.**

```
-- Attention, les scripts de récupération des données auront dû être préalablement exécutés
-- Marcel Fraysse est l'agent responsable de la mission
-- L'ONF est déjà connue dans notre base, on va récupérer son code

INSERT INTO agent VALUES
    ('SE504', 'Fraysse', 'Marcel', '28 rue de l''optimisation', '12200', 'Villefranche de Rouergue',
            'marcel@villef.cty', '0621121212');

INSERT INTO agent VALUES
    ('AC907', 'Marcenac', 'Sylvie ', '44 rue de la transaction', '82400', 'Féneyrols',
            'sissi_m@lpd2i.edu', '0682400082');

INSERT INTO recensement VALUES
    ('R1208200008',TO_DATE('07/12/2008', 'MM/DD/YYYY'),TO_DATE('07/19/2008', 'MM/DD/YYYY'),
            (SELECT numadmin FROM prescripteur WHERE organisme='ONF'), 'SE504',0);
```

RE3.12) **Le recensement R1208200124 s'effectuera 3 jours plus tard.**

```
UPDATE recensement
SET daterecens = daterecens + 3
WHERE numero = 'R1208200124' ;
```

RE3.13) **Mademoiselle Véronique Lanvin a cessé toute activité pour Natura. Elle avait pour matricule X1245. Il vous est demandé de l'enlever de la liste des agents. Vous proposerez une requête SQL et son commentaire.**

```
DELETE FROM agent
WHERE matricule = '0B190' ;
```

Cette suppression serait impossible s'il existait des relevés pour cet agent ou des recensements pour lesquels il est responsable.

RE3.14) **J'observe le résultat de mes mises à jour. Très content, je l'annonce fièrement à mon voisin. Il ne me croit pas. Je lui donne tout d'abord les droits voulus puis il consulte mais pour lui, ça ne marche pas : il ne voit pas mes modifications. Expliquer ce qui peut se passer.**

Pas COMMIT de mon côté.
Mon collègue ne voit pas les modifications en instance. Les transactions respectent le principe d'isolement.
N'ayant pas validé la transaction, mes modifications en cours ne sont pas visibles par les autres utilisateurs et programmes potentiels.

RE3.15) **Effectuez un ROLLBACK et expliquez quel est son impact depuis RE3.01).**

```
ROLLBACK ;
```
L'impact est uniquement sur les mises à jour.

⇨ INTERROGATION

RE3.16) **Afficher la liste des plantes.**

```
SELECT * FROM plante;
-- Préférer une liste d'attributs plutôt que * (mais ici, tous ces attributs sont peut-être à visualiser

SELECT nomscient, nomcommun, couleurfleur, dateflormin, dateflormax, espece, descriptif
FROM plante ;
```

RE3.17) **Quelles sont les espèces que l'on a observées dans les recensements de juillet cette année ? (Suivre la norme SQL1 pour les jointures)**

```
SELECT DISTINCT espece
FROM plante, releve, recensement
WHERE laplante=nomscient
AND lerec=numero
AND daterecens BETWEEN TO_DATE('07/01/2010', 'MM/DD/YYYY') AND TO_DATE('07/31/2010', 'MM/DD/YYYY') ;
```

RE3.18) **Quelles sont les espèces que l'on a observées dans les recensements de juillet cette année ? (même requête en suivant la norme SQL2).**

```
SELECT DISTINCT espece
FROM plante
   JOIN releve          ON nomscient = laplante
   JOIN recensement   ON numero = lerec
WHERE daterecens BETWEEN TO_DATE('07/01/2010', 'MM/DD/YYYY') AND TO_DATE('07/31/2010', 'MM/DD/YYYY') ;
```

RE3.19) **Afficher les plantes groupées par espèce.**

```
SELECT espece, nomscient, nomcommun
FROM plante
ORDER BY espece ;
```

RE3.20) **Afficher les différentes espèces ainsi que le nombre de plantes existant dans notre base pour ces espèces.**

```
SELECT espece, COUNT(*) AS nb_plantes
FROM plante
GROUP BY espece ;
```

RE3.21) **Afficher les informations sur les agents ayant déjà été responsables de recensements en les regroupant par ville.**

```
SELECT matricule, nom, prenom
FROM agent, recensement
WHERE matricule=coderesp
ORDER BY ville;
```

RE3.22) **Classer les plantes selon la durée de floraison. Vous afficherez en premier les plantes à floraison longue.**

```
SELECT nomscient, nomcommun, dateflormax-dateflormin AS dureeflor
FROM plante
ORDER BY dureeflor DESC;
```

RE3.23) **Visualiser le nom et le prénom des agents qui ont été responsables de plusieurs recensements au deuxième semestre 2010.**

```
SELECT coderesp, nom, prenom, COUNT (*) AS nb_recens
FROM recensement
JOIN agent
ON matricule=coderesp
WHERE daterecens BETWEEN TO_DATE('07/01/2010', 'MM/DD/YYYY') AND TO_DATE('12/31/2010', 'MM/DD/YYYY')
GROUP BY coderesp, nom, prenom
HAVING COUNT(*)>1;
```

RE3.24) **Quelle est l'orchidée présentant la fin de floraison la plus tardive de l'année.**

```
SELECT nomscient, nomcommun, dateflormax
FROM plante
WHERE espece='orchidée'
AND dateflormax = ( SELECT MAX(dateflormax) FROM plante
                    WHERE espece='orchidée');
```

RE3.25) **Quelles sont les plantes que l'on ne trouve que sous les sapins.**

```
SELECT DISTINCT nomscient
FROM plante P, releve, zonepeup
WHERE laplante=P.nomscient
AND lazone=cadastre
AND typearbre='sapin'
AND NOT EXISTS (SELECT * FROM releve, zonepeup
                WHERE laplante=P.nomscient
                AND lazone=cadastre
                AND typearbre<>'sapin');
```

RE3.26) **Quels sont les agents qui ont un prescripteur situé dans leur ville ?**

```
SELECT A.matricule, A.nom, A.prenom, A.ville
FROM agent A, prescripteur P
WHERE A.ville=P.ville;
```

RE3.27) **Quel mécanisme permettrait de veiller à la cohérence de l'attribut nb_relevés.**

Les déclencheurs ou trigger sont exécutés quand une opération de mise à jour de la base de données est nécessaire.

RE3.28) **Quels sont les plantes qui fleurissent plus de 60 jours ?**

```
SELECT nomscient, nomcommun, dateflormax-dateflormin AS dureeflor
FROM plante
WHERE dateflormax-dateflormin >=60
ORDER BY dureeflor DESC;
```

RE3.29) **Quels sont les recensements qui sont faits sur plusieurs départements ?**

```
SELECT DISTINCT numero , count(DISTINCT departement) nb_dep
FROM recensement, releve, zonepeup
WHERE numero=lerec
AND lazone=cadastre
GROUP BY numero
HAVING COUNT (DISTINCT departement)>1 ;
```

RE3.30) **Y a-t-il des fleurs dont le nom commence par la lettre G ?**

```
SELECT *
FROM plante
WHERE lower(nomscient) LIKE 'g%';
-- lower utile ici mais ralentit la requête.
```

RE3.31) **A quelle date a-t-on effectué des recensements sur la commune d'Aubin ? (Sous-Requête demandée)**

```
SELECT DISTINCT numero , daterecens
FROM recensement
WHERE numero IN    (     SELECT lerec
                         FROM  releve, zonepeup
                         WHERE  lazone=cadastre
                         AND commune='Aubin'    )
ORDER BY daterecens DESC ;
```

RE3.32) **Fournir la liste des agents qui n'ont jamais été responsables de recensements. Vous montrerez dans cette requête comment utiliser l'opérateur unaire NOT EXISTS. Donner ensuite une formulation à l'aide de l'opérateur NOT IN.**

```
SELECT matricule, nom, prenom
FROM agent
WHERE NOT EXISTS (   SELECT *  FROM recensement
                         WHERE  coderesp = matricule) ;

SELECT matricule, nom, prenom
FROM agent
WHERE matricule NOT IN (
            SELECT coderesp FROM recensement);
```

RE3.33) **Afficher la liste des communes dans lesquelles des plantes de grande dimension ont été rencontrées. En botanique, on parle de grandes plantes à partir de 90 centimètres de taille moyenne. Vous penserez à mentionner l'altitude.**

```
SELECT DISTINCT commune, altmin, altmax
FROM plante, releve, zonepeup
WHERE laplante=nomscient
AND lazone=cadastre
AND taillem>=90;
```

RE3.34) **Quelles sont les couleurs de fleurs que l'on trouve en novembre 2010?**

```
SELECT DISTINCT couleurfleur
FROM plante
WHERE       dateflormin<TO_DATE('11/01/2010', 'MM/DD/YYYY') AND dateflormax>=TO_DATE('11/01/2010', 'MM/DD/YYYY')
          OR   dateflormin>=TO_DATE('11/01/2010', 'MM/DD/YYYY') AND dateflormax<=TO_DATE('11/31/2010', 'MM/DD/YYYY');
```

RE3.35) **Quelle est la couleur de plante la plus courante ?**

```
SELECT DISTINCT couleurfleur, COUNT(*) AS nb_plantes
FROM plante
GROUP BY couleurfleur
HAVING COUNT(*)= (     SELECT MAX (COUNT(*))
                         FROM plante
                         GROUP BY couleurfleur) ;
```

RE3.36) **Tester la requête précédente successivement avec sqlplus et isqlplus. Comment expliquer la différence pour les dates ?**

La différence vient des variables d'environnement (les formats de date, les jeux de caractères) du système : sqlplus est installé sur chaque poste de travail (des ordinateurs en configuration « française ») alors que isqlplus est dans le SGBD installé dans l'environnement « américain » de la machine nommée «seroracle ».

RE3.37) **Afficher la liste de tous les agents ainsi que, s'ils ont été responsables de recensements, le numéro de ces recensements et leurs dates.**

```
SELECT matricule, nom, prenom, numero, daterecens
FROM agent
LEFT OUTER JOIN recensement
ON matricule=coderesp
ORDER BY nom ;
```

HORS CATEGORIE
Quelles sont les espèces que l'on a rencontrées dans tous les recensements ?

```
SELECT DISTINCT espece
FROM plante P1
WHERE NOT EXISTS (SELECT * FROM recensement
                  WHERE NOT EXISTS (SELECT *
                                    FROM releve, plante P
                                    WHERE numero=lerec
                                    AND laplante=P.nomscient
                                    AND espece=P1.espece));
```

⇨ CONTROLE DES DONNEES, DEFINITION DE VUES, ACCES CONCURRENTS

RE3.38) **Afin de publier quelques informations à destination du grand public sur le site internet de Natura, préparer une vue dans laquelle on mentionnera le nom commun, le nom scientifique, la durée de floraison et toutes les autres caractéristiques à l'exception de l'espèce.**

```
CREATE VIEW v_plantes
  ( nomcommun, nomscient, dureeflor, couleurfleur, dateflormin, dateflormax, descriptif)
  AS SELECT
  nomcommun, nomscient, dateflormax- dateflormin, couleurfleur, dateflormin, dateflormax,
        descriptif
  FROM plante ;
```

RE3.39) **Permettre à l'utilisateur BOTANISTE02, ainsi qu'à votre voisin, la consultation des relations RECENSEMENT, PRESCRIPTEUR et AGENT. Il doit aussi être possible de mettre à jour des n-uplets de la relation RECENSEMENT, toutefois on ne pourra modifier ni le numéro de ce recensement et sa date, ni le prescripteur.**

```
GRANT SELECT, UPDATE (daterecens, coderesp)
 ON recensement
TO botaniste02 ;

GRANT SELECT ON prescripteur  TO botaniste02 ;

GRANT SELECT ON agent TO botaniste02 ;
```

RE3.40) **Donner à l'utilisateur BOTANISTE02 l'accès aux zones de peuplement du Tarn et aux diverses plantes que l'on a pu y trouver.**

```
CREATE VIEW v_tarnplantes AS
SELECT cadastre, lieudit, commune, departement, sol, inclinaison, typearbre, altmin, altmax, nomscient,
        nomcommun, couleurfleur, dateflormin, dateflormax, espece, descriptif
  FROM zonepeup
  JOIN releve ON cadastre=lazone
  JOIN plante ON laplante=nomscient
  WHERE departement = 'Tarn' ;

GRANT SELECT ON v_tarnplantes TO botaniste02 ;
```

RE3.41) **Attribuer à votre voisin divers droits sur des objets de votre schéma, tester et écrivez le scénario de vos tests. Vous modifierez notamment un n-uplet puis vous demanderez à votre voisin de consulter cette modification et de modifier ce même n-uplet. Que se passe-t-il ? Pourquoi ?**

```
GRANT SELECT …
GRANT UPDATE …

UPDATE dans mon schema
Mon voisin à qui j'ai confié des droits de modification cherche à modifier le/les n-uplet(s) que je modifie et un
blocage se produit.
```

⇨ DICTIONNAIRE DES DONNEES

RE3.42) **Obtenir le nom de toutes vos relations.**

```
SELECT table_name
FROM user_tables
ORDER BY table_name ;
```

RE3.43) **Quel est le nom de toutes les relations auxquelles vous avez accès et le nom de leur propriétaire (en excluant les relations 'système') ?**

```
SELECT owner, table_name FROM all_tables
WHERE owner NOT LIKE '%SYS%'
ORDER BY owner;
```

RE3.44) **Trouver le nom des index, leur type ainsi que le nom des relations et attributs sur lesquels ils portent.**

```
SELECT *
FROM dictionary
WHERE table_name LIKE '%IND%';

SELECT index_name, index_type, table_owner, table_name, table_type, uniqueness
FROM user_indexes ;

SELECT U1.index_name, U1.table_name, U1.column_name, U2. uniqueness, U1.descend
FROM  user_ind_columns U1, user_indexes U2
WHERE U1. index_name = U2.index_name
ORDER BY U1.index_name;
```

RE3.45) **Quels sont les droits qui vous ont été affectés et par qui ?**

```
GRANT INSERT ON releve  TO PUBLIC;

SELECT grantor, owner, grantee, table_name, SUBSTR(privilege,1,12), grantable
FROM  ALL_TAB_PRIVS_RECD
-- Affinage, on enlève les "parasites"
WHERE grantor NOT LIKE '%SYS%'
AND grantor NOT IN ( 'ORDPLUGINS', 'WK_TEST', 'XDB', 'FLOWS_020100');
```

RE3.46) **Quelles contraintes d'intégrité ont-elles été définies ? Sur quels attributs ? Quelles sont les contraintes de domaine ?**

```
SELECT owner,  constraint_name, constraint_type,
            table_name, search_condition, r_owner, r_constraint_name
FROM user_constraints ;
```