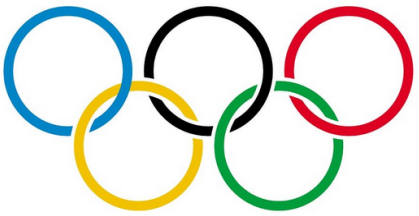


S204

Exploitation d’une Base de Données

VOUILLOUX Nicolas

Partie 1 - Mise en place de la base de données

I - Attribution des droits sûr GestionJO et AnalyseJO :

Analyse JO:

```
grant select on ATHLETE to AnalyseJO;
grant select on COMPOSITION_EQUIPE to AnalyseJO;
grant select on DISCIPLINE to AnalyseJO;
grant select on EQUIPE to AnalyseJO;
grant select on EVENEMENT to AnalyseJO;
grant select on HOTE to AnalyseJO;
grant select on NOC to AnalyseJO;
grant select on PARTICIPATION_EQUIPE to AnalyseJO;
grant select on PARTICIPATION_INDIVIDUELLE to AnalyseJO;
grant select on SPORT to AnalyseJO;
revoke all on log from AnalyseJO;
Grant execute on biographie to AnalyseJO;
grant execute on resultats to AnalyseJO;
```

Gestion JO :

```
grant select,update,insert,delete on ATHLETE to GestionJO;
grant select,update,insert,delete on COMPOSITION_EQUIPE to GestionJO;
grant select,update,insert,delete on DISCIPLINE to GestionJO;
grant select,update,insert,delete on EQUIPE to GestionJO;
grant select,update,insert,delete on EVENEMENT to GestionJO;
grant select,update,insert,delete on HOTE to GestionJO;
grant select,update,insert,delete on NOC to GestionJO;
grant select,update,insert,delete on PARTICIPATION_EQUIPE to GestionJO;
grant select,update,insert,delete on PARTICIPATION_INDIVIDUELLE to GestionJO;
grant select,update,insert,delete on SPORT to GestionJO;
grant select on log to GestionJO;
grant execute on biographie to GestionJO;
grant execute on resultats to GestionJO;
grant select on log to GestionJO;
grant execute on ajouter_resultat_individuelle to GestionJO;
grant execute on ajouter_resultat_equipe to GesionJO;
```

II - Création des Vues :

MEDAILLES_ATLHETES :

```
CREATE OR REPLACE VIEW MEDAILLES_ATHLETES AS
WITH
Medailles_Indiv AS (
        SELECT idathlete,
        SUM(CASE WHEN medaille = 'Gold' THEN 1 ELSE 0 END) AS NB_MEDAILLES_OR_INDIV,
        SUM(CASE WHEN medaille = 'Silver' THEN 1 ELSE 0 END) AS NB_MEDAILLES_ARGENT_INDIV,
        SUM(CASE WHEN medaille = 'Bronze' THEN 1 ELSE 0 END) AS NB_MEDAILLES_BRONZE_INDIV
        FROM participation_individuelle
        GROUP BY idathlete
),
Medailles_Equipe AS (
        SELECT ce.idathlete,
        SUM(CASE WHEN pe.medaille = 'Gold' THEN 1 ELSE 0 END) AS NB_MEDAILLES_OR_GROUP,
        SUM(CASE WHEN pe.medaille = 'Silver' THEN 1 ELSE 0 END) AS
NB_MEDAILLES_ARGENT_GROUP,
        SUM(CASE WHEN pe.medaille = 'Bronze' THEN 1 ELSE 0 END) AS
NB_MEDAILLES_BRONZE_GROUP
        FROM participation_equipe pe
        INNER JOIN composition_equipe ce ON ce.idequipe = pe.idequipe
        GROUP BY ce.idathlete
)
SELECT a.idathlete, a.nomathlete, a.prenomathlete,
        NVL(mi.NB_MEDAILLES_OR_INDIV, 0) + NVL(me.NB_MEDAILLES_OR_GROUP, 0) AS NB_OR,
        NVL(mi.NB_MEDAILLES_ARGENT_INDIV, 0) + NVL(me.NB_MEDAILLES_ARGENT_GROUP, 0) AS
NB_ARGENT,
        NVL(mi.NB_MEDAILLES_BRONZE_INDIV, 0) + NVL(me.NB_MEDAILLES_BRONZE_GROUP, 0) AS
NB_BRONZE,
        (NVL(mi.NB_MEDAILLES_OR_INDIV, 0) + NVL(me.NB_MEDAILLES_OR_GROUP, 0) +
        NVL(mi.NB_MEDAILLES_ARGENT_INDIV, 0) + NVL(me.NB_MEDAILLES_ARGENT_GROUP, 0) +
        NVL(mi.NB_MEDAILLES_BRONZE_INDIV, 0) + NVL(me.NB_MEDAILLES_BRONZE_GROUP, 0)) AS
NB_TOTAL
FROM ATHLETE a
LEFT JOIN Medailles_Indiv mi ON a.idathlete = mi.idathlete
LEFT JOIN Medailles_Equipe me ON a.idathlete = me.idathlete
ORDER BY NB_OR DESC, NB_ARGENT DESC, NB_BRONZE DESC, NB_TOTAL DESC, a.nomathlete,
a.prenomathlete, a.idathlete;
```

MEDAILLES_NOC :

```
CREATE OR REPLACE VIEW MEDAILLES_NOC AS
WITH
Medailles_Indiv AS (
```

```
        SELECT noc,
        SUM(CASE WHEN medaille = 'Gold' THEN 1 ELSE 0 END) AS NB_MEDAILLES_OR_INDIV,
        SUM(CASE WHEN medaille = 'Silver' THEN 1 ELSE 0 END) AS NB_MEDAILLES_ARGENT_INDIV,
        SUM(CASE WHEN medaille = 'Bronze' THEN 1 ELSE 0 END) AS NB_MEDAILLES_BRONZE_INDIV
        FROM participation_individuelle
        GROUP BY noc
),
Medailles_Equipe AS (
        SELECT e.noc,
        SUM(CASE WHEN pe.medaille = 'Gold' THEN 1 ELSE 0 END) AS NB_MEDAILLES_OR_GROUP,
        SUM(CASE WHEN pe.medaille = 'Silver' THEN 1 ELSE 0 END) AS
NB_MEDAILLES_ARGENT_GROUP,
        SUM(CASE WHEN pe.medaille = 'Bronze' THEN 1 ELSE 0 END) AS
NB_MEDAILLES_BRONZE_GROUP
        FROM participation_equipe pe
        INNER JOIN equipe e ON e.idequipe = pe.idequipe
        GROUP BY e.noc
)
SELECT n.codenoc,
        NVL(mi.NB_MEDAILLES_OR_INDIV, 0) + NVL(me.NB_MEDAILLES_OR_GROUP, 0) AS NB_OR,
        NVL(mi.NB_MEDAILLES_ARGENT_INDIV, 0) + NVL(me.NB_MEDAILLES_ARGENT_GROUP, 0) AS
NB_ARGENT,
        NVL(mi.NB_MEDAILLES_BRONZE_INDIV, 0) + NVL(me.NB_MEDAILLES_BRONZE_GROUP, 0) AS
NB_BRONZE,
        (NVL(mi.NB_MEDAILLES_OR_INDIV, 0) + NVL(me.NB_MEDAILLES_OR_GROUP, 0) +
        NVL(mi.NB_MEDAILLES_ARGENT_INDIV, 0) + NVL(me.NB_MEDAILLES_ARGENT_GROUP, 0) +
        NVL(mi.NB_MEDAILLES_BRONZE_INDIV, 0) + NVL(me.NB_MEDAILLES_BRONZE_GROUP, 0)) AS
NB_TOTAL
FROM noc n
LEFT JOIN Medailles_Indiv mi ON n.codenoc = mi.noc
LEFT JOIN Medailles_Equipe me ON n.codenoc = me.noc
ORDER BY NB_OR DESC, NB_ARGENT DESC, NB_BRONZE DESC, NB_TOTAL DESC, n.codenoc;
```

Je me suis servi de la fonction nvl pour éviter les erreurs de compilation car elle remplace les médailles ‘null’ par 0.

III - Création des Fonctions :

biographie(id_athlete NUMBER)

```
create or replace FUNCTION get_athlete_info(id_athlete NUMBER)
RETURN VARCHAR2 IS
        athlete_info VARCHAR2(32767);
BEGIN
        SELECT JSON_OBJECT('nom' IS a.nomAthlete,
                'prénom' IS a.prenomAthlete,
                'surnom' IS a.surnom,
                'genre' IS SUBSTR(a.genre,0,1),
                'dateNaissance' IS TO_CHAR(a.DateNaissance, 'yyyy-MM-dd'),
                'dateDécès' IS TO_CHAR(a.DateDeces, 'yyyy-MM-dd'),
                'taille' IS (CAST(a.taille AS VARCHAR(3))||' cm'),
                'poids' IS (CAST(a.poids AS VARCHAR(3))||' kg'),
                'médaillesOr' IS CAST(ma.nb_or AS VARCHAR(2)),
                'médaillesArgent' IS CAST(ma.nb_argent AS VARCHAR(2)),
```

```
                'médaillesBronze' IS CAST(ma.nb_bronze AS VARCHAR(2)),
                'médaillesTotal' IS CAST(ma.nb_total AS VARCHAR(2)))
        INTO athlete_info
        FROM ATHLETE a
        LEFT JOIN MEDAILLES_ATHLETES ma ON a.idathlete = ma.idathlete
        WHERE a.idathlete = id_athlete;
        RETURN athlete_info;
EXCEPTION
        WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20011, 'Athlète inconnu');
END;
```

```
create or replace FUNCTION biographie(id_athlete NUMBER)
RETURN VARCHAR2 IS
        athlete_info VARCHAR2(2000);
BEGIN
        BEGIN
        athlete_info := get_athlete_info(id_athlete);
        RETURN athlete_info;
        EXCEPTION
        WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20011, 'Athlète inconnu');
        END;
END;
```

De multiples modifications ont été apportées aux fonctions ci-dessus ce qui a mené à avoir une fonction get_athlete_infos, inutile en sachant que tout peut être mis dans biographie.

L'objet JSON permet de transformer un String en JSON ce qui permet de faire un code plus concis.

Resultats(id _evenement NUMBER)

```
create or replace FUNCTION get_resultats_info(id_evenement NUMBER)
RETURN VARCHAR2  IS
        resultat_info VARCHAR2(2000);
        cursor solocurseur IS
        SELECT *
        FROM participation_individuelle
        NATURAL JOIN athlete
        WHERE idevent = id_evenement
        ORDER BY TO_NUMBER(REPLACE(resultat, '=', '')), nomathlete, prenomathlete;


BEGIN
        resultat_info := '{"résultats":'||'[';
        for lignec in solocurseur LOOP
        resultat_info := resultat_info || '{"position": "' || lignec.resultat || '", "athlète(s)": "' || lignec.prenomathlete|| '
'|| lignec.nomathlete|| '", "noc": "' || lignec.noc || '", "médaille": ' || CASE WHEN lignec.medaille IS NOT NULL
THEN '"' || lignec.medaille|| '"' ELSE 'null' END ||'},';
        END LOOP;
        resultat_info := resultat_info ||']'||'}';
        RETURN resultat_info;
EXCEPTION
```

```
        WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20012, 'Événement inconnu');
END;

create or replace FUNCTION resultats(idevenement NUMBER)
RETURN VARCHAR2 IS
        resultats_info VARCHAR2(32767);
BEGIN
        begin
        resultats_info := get_resultats_info(idevenement);
        RETURN resultats_info;
        EXCEPTION
        WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20012, 'Événement inconnu');
        end;
END;
```

Il en est de même avec la fonction résultats ou une fonction seule était possible.

IV - Création des procedures :

Ajouter_Resultat_Individuel

```
create or replace procedure ajouter_resultat_individuel(id_evenement evenement.idevenement%type,id_athlete
athlete.idathlete%type, code_noc noc.codenoc%type, resultat participation_individuelle.resultat%type) as
BEGIN
        dbms_output.put_line('TODO');
END ajouter_resultat_individuel;
```

Ajouter_Resultat_Equipe

```
create or replace procedure ajouter_resultat_equipe(id_evenement evenement.idevenement%type,id equipe
equipe.equipe%type, resultat participation_equipe.resultat%type) as
BEGIN
        dbms_output.put_line('TODO');
END ajouter_resultat_equipe;
```

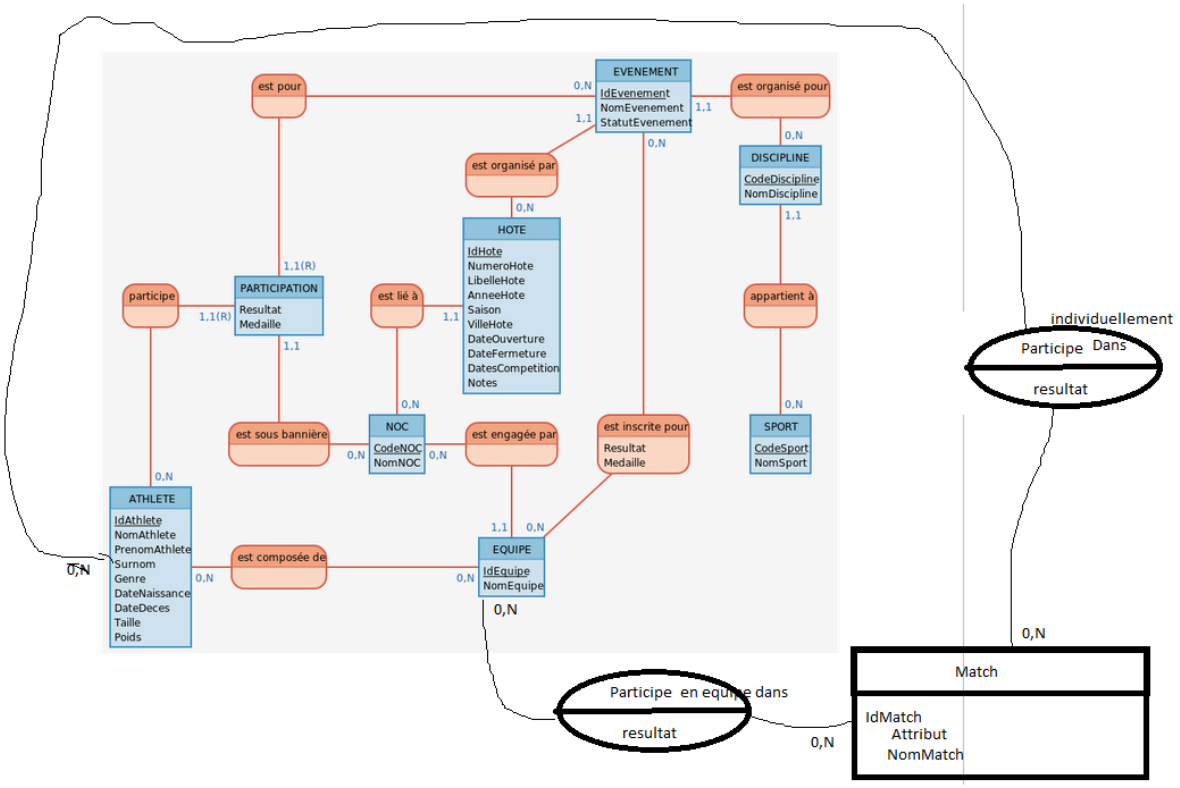
V - Création de la Table Log :

```
CREATE TABLE LOG (
        IDLOG NUMBER(10) PRIMARY KEY,
        IDAUTEUR VARCHAR(200) NOT NULL,
        ACTION VARCHAR(6) NOT NULL CHECK (action IN('INSERT', 'UPDATE','DELETE')),
        DATEACTION DATE NOT NULL,
        LIGNEAVANT VARCHAR(1023),
        LIGNEAPRES VARCHAR(1023)
);
```

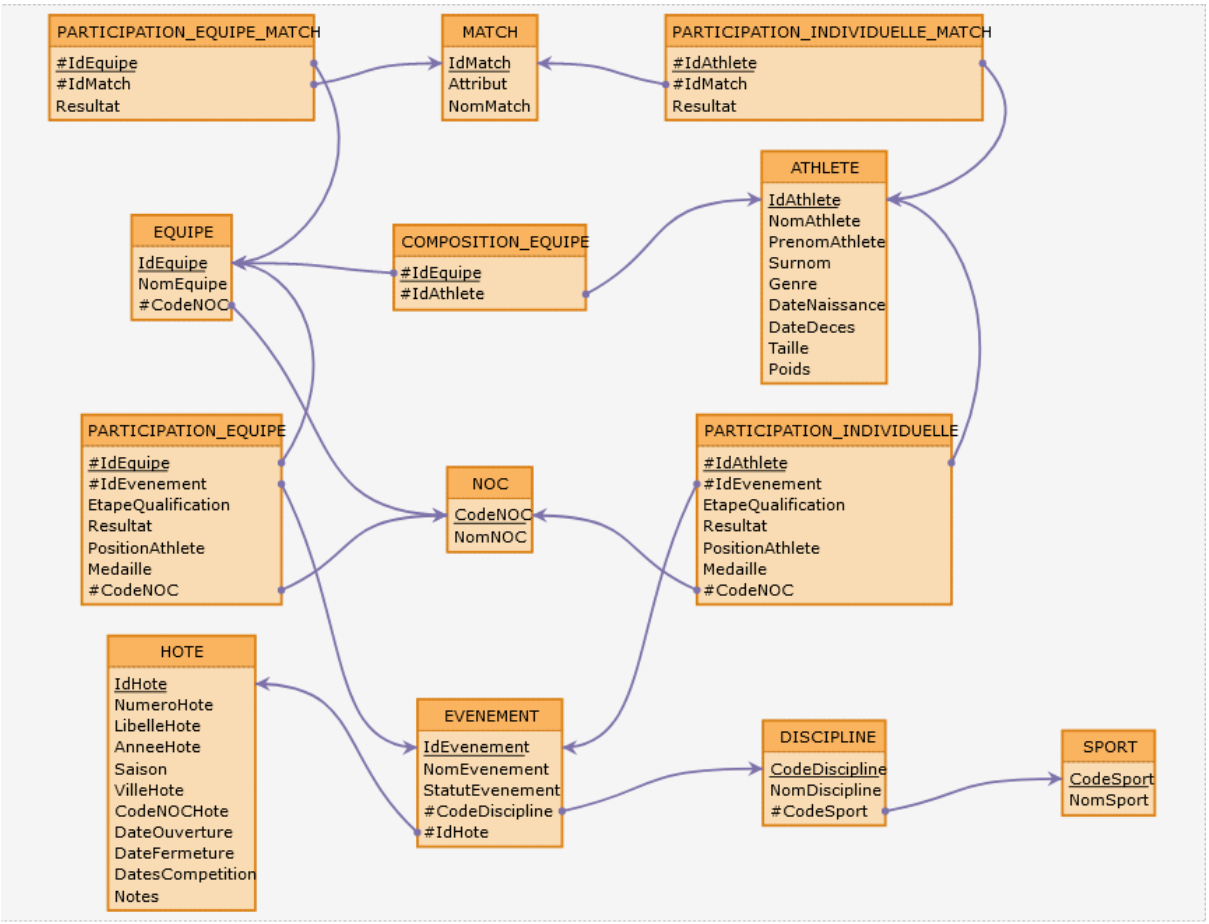
VI - Extension de la base :

Faute de temps suite à l'organisation des projets et des partiels (tous concentrés sur deux semaines d'un semestre complet, le projet S204 n'est pas le seul merci de votre compréhension.

MCD Base Extended



Sr base extended



Script d'insertion de données

Pour le script d'insertion de données il aura juste a ce basée sur les données suivantes des JO de 1992:

4x7.5 kilometers relay, Men : https://www.olympedia.org/results/1552
Date - 16 February 1992 – 9:30

Super G, Women : https://www.olympedia.org/results/1396
Date 18 February 1992 – 12:15

1,000 Meters, Men : https://www.olympedia.org/results/481
Date 18–20 Ferbruary 1992