Compte rendu des Travaux Pratiques : Génération et insertion de dates

TP Exercice 2: Génération et insertion de dates dans une table date_dim

Objectif du TP : Le but de ce TP était de générer des dates pour une année donnée à partir du 1er janvier 2024 jusqu'au 31 décembre 2024, et de les insérer dans une table appelée date_dim. Les étapes suivantes ont été réalisées pour atteindre cet objectif :

1- Création de la table date_dim :

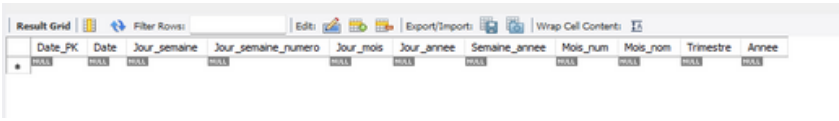
Une table nommée date_dim a été créée avec les attributs nécessaires pour stocker les informations sur les dates générées, y compris le jour de la semaine, le numéro du jour de la semaine, le jour du mois, le jour de l'année, la semaine de l'année, le numéro du mois, le nom du mois, le trimestre et l'année. Une contrainte de clé primaire a été ajoutée sur l'attribut Date_PK, et un index a été créé sur la colonne Date pour optimiser les performances des requêtes.

code:

```
mysql> create database TP;
Query OK, 1 row affected (0.03 sec)

mysql> use TP
Database changed
mysql> CREATE TABLE date_dim (
    ->     Date_PK INT NOT NULL AUTO_INCREMENT,
    ->     Date DATE NOT NULL,
    ->     Jour_semaine VARCHAR(20) NOT NULL,
    ->     Jour_semaine_numero INT NOT NULL,
    ->     Jour_mois INT NOT NULL,
    ->     Jour_annee INT NOT NULL,
    ->     Semaine_annee INT NOT NULL,
    ->     Mois_num INT NOT NULL,
    ->     Mois_nom VARCHAR(20) NOT NULL,
    ->     Trimestre INT NOT NULL,
    ->     Annee INT NOT NULL,
    ->     PRIMARY KEY (Date_PK),
    ->     INDEX idx_Date (Date)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

resultat:

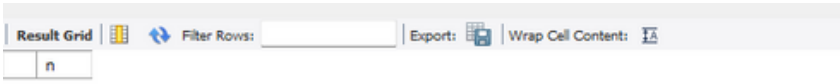


2- Génération des nombres :

Une table nommée nombres avec un seul attribut n a été créée pour stocker une séquence de nombres.

code:

resultat:



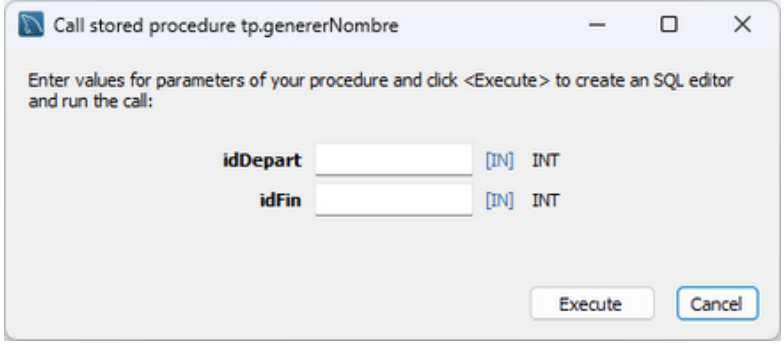
3- Création de la procédure genererNombre :

Une procédure stockée genererNombre a été définie pour insérer des nombres de 0 à 365 dans la table nombres.

code:

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE genererNombre(IN idDepart INT, IN idFin INT)
    -> BEGIN
    ->     DECLARE i INT;
    ->     SET i = idDepart;
    ->     WHILE i <= idFin DO
    ->         INSERT INTO nombres (n) VALUES (i);
    ->         SET i = i + 1;
    ->     END WHILE;
    -> END//
Query OK, 0 rows affected (0.01 sec)
```

resultat:



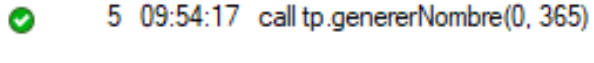
4- Appel de la procédure genererNombre :

La procédure genererNombre a été appelée avec les paramètres 0 et 365 pour générer les nombres de 0 à 365.

code:

```
mysql> CALL genererNombre(0, 365);
Query OK, 1 row affected (0.40 sec)
```

resultat:



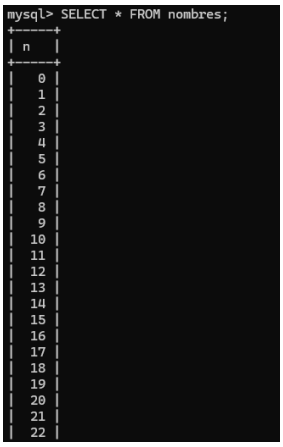
5- Vérification des enregistrements dans la table nombres :

Une requête SELECT a été utilisée pour vérifier que les enregistrements ont été correctement insérés dans la table nombres.

code:

```
mysql> SELECT * FROM nombres;
```

resultat:



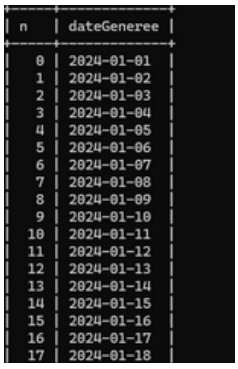
6- Exécution de la requête SELECT pour générer les dates :

Une requête SELECT a été exécutée pour générer les dates en ajoutant un certain nombre de jours à la date de départ '2024-01-01'. Seules les dates inférieures ou égales au 31 décembre 2024 ont été sélectionnées.

code:

```
mysql> SELECT n, DATE_ADD('2024-01-01', INTERVAL n DAY) AS dateGeneree FROM nombres WHERE DATE_ADD('2024-01-01', INTERVA
L n DAY) <= '2024-12-31';
```

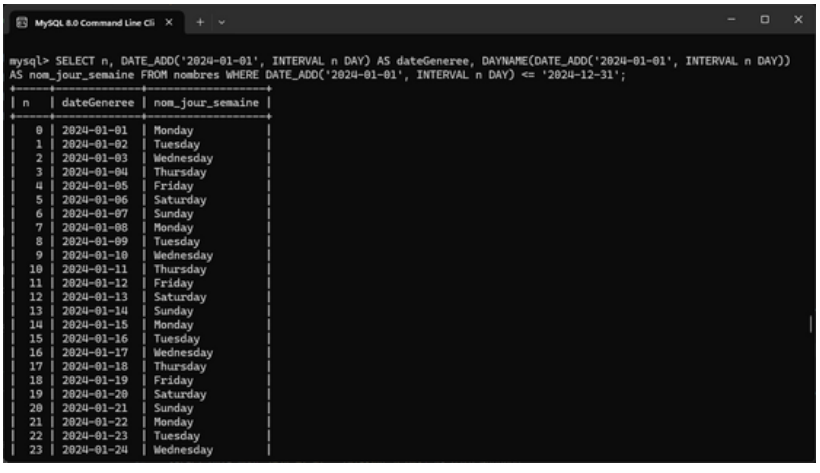
resultat:



7- Utilisation de la requête imbriquée pour obtenir le nom du jour de la semaine :

Une requête SELECT imbriquée a été utilisée pour obtenir le nom du jour de la semaine à partir des dates générées.

code et resultat:



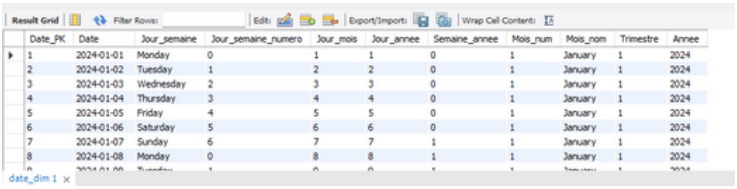
8- Ajout des autres colonnes et insertion dans la table date_dim :

Les autres colonnes de la table date_dim ont été remplies en utilisant les fonctions MySQL appropriées pour extraire les informations nécessaires à partir des dates générées.

code:

```
mysql> INSERT INTO date_dim (Date, Jour_semaine, Jour_semaine_numero, Jour_mois, Jour_annee, Semaine_annee, Mois_num, Mo
is_nom, Trimestre, Annee)
    -> SELECT DATE_ADD('2024-01-01', INTERVAL n DAY), DAYNAME(DATE_ADD('2024-01-01', INTERVAL n DAY)), WEEKDAY(DATE_ADD(
'2024-01-01', INTERVAL n DAY)), DAY(DATE_ADD('2024-01-01', INTERVAL n DAY)), DAYOFYEAR(DATE_ADD('2024-01-01', INTERVAL n
 DAY)), WEEK(DATE_ADD('2024-01-01', INTERVAL n DAY)), MONTH(DATE_ADD('2024-01-01', INTERVAL n DAY)), MONTHNAME(DATE_ADD(
'2024-01-01', INTERVAL n DAY)), QUARTER(DATE_ADD('2024-01-01', INTERVAL n DAY)), YEAR(DATE_ADD('2024-01-01', INTERVAL n
DAY))
    -> FROM nombres WHERE DATE_ADD('2024-01-01', INTERVAL n DAY) <= '2024-12-31';
Query OK, 366 rows affected (0.01 sec)
Records: 366  Duplicates: 0  Warnings: 0
```

resultat:



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Date_PK	Date	Jour_semaine	Jour_semaine_numero	Jour_mois	Jour_annee	Semaine_annee	Mois_num	Mois_nom	Trimestre	Annee
1	2024-01-01	Monday	0	1	1	0	1	January	1	2024
2	2024-01-02	Tuesday	1	2	2	0	1	January	1	2024
3	2024-01-03	Wednesday	2	3	3	0	1	January	1	2024
4	2024-01-04	Thursday	3	4	4	0	1	January	1	2024
5	2024-01-05	Friday	4	5	5	0	1	January	1	2024
6	2024-01-06	Saturday	5	6	6	0	1	January	1	2024
7	2024-01-07	Sunday	6	7	7	1	1	January	1	2024
8	2024-01-08	Monday	0	8	8	1	1	January	1	2024

date_dim 1

9&11- Requête SELECT pour afficher le trimestre et indiquer si une date correspond à un jour de fin de semaine :

Une requête SELECT a été utilisée pour afficher le trimestre correspondant à chaque date et indiquer si cette date est un jour de fin de semaine (samedi ou dimanche).

code et resultat:

```
MySQL 8.0 Command Line Cli

mysql> SELECT DATE_ADD('2024-01-01', INTERVAL n DAY) AS Date_Generee,
    ->        QUARTER(DATE_ADD('2024-01-01', INTERVAL n DAY)) AS Trimestre,
    ->        CASE WHEN DAYNAME(DATE_ADD('2024-01-01', INTERVAL n DAY)) IN ('Saturday', 'Sunday') THEN 'Oui' ELSE 'Non'
END AS isWeekEnd
    -> FROM nombres WHERE DATE_ADD('2024-01-01', INTERVAL n DAY) <= '2024-12-31';
+--------------+-----------+-----------+
| Date_Generee | Trimestre | isWeekEnd |
+--------------+-----------+-----------+
| 2024-01-01   |         1 | Non       |
| 2024-01-02   |         1 | Non       |
| 2024-01-03   |         1 | Non       |
| 2024-01-04   |         1 | Non       |
| 2024-01-05   |         1 | Non       |
| 2024-01-06   |         1 | Oui       |
| 2024-01-07   |         1 | Oui       |
| 2024-01-08   |         1 | Non       |
| 2024-01-09   |         1 | Non       |
| 2024-01-10   |         1 | Non       |
| 2024-01-11   |         1 | Non       |
| 2024-01-12   |         1 | Non       |
| 2024-01-13   |         1 | Oui       |
| 2024-01-14   |         1 | Oui       |
| 2024-01-15   |         1 | Non       |
| 2024-01-16   |         1 | Non       |
| 2024-01-17   |         1 | Non       |
| 2024-01-18   |         1 | Non       |
| 2024-01-19   |         1 | Non       |
| 2024-01-20   |         1 | Oui       |
| 2024-01-21   |         1 | Oui       |
```

10- Combinaison de requête INSERT et SELECT pour insérer le résultat d'une SELECT dans la table date_dim :

Une combinaison de requête INSERT et SELECT a été utilisée pour insérer le résultat de la requête SELECT dans la table date_dim, remplissant ainsi la table avec les informations sur les dates générées.

code :

```
mysql> INSERT INTO date_dim (Date, Jour_semaine, Jour_semaine_numero, Jour_mois, Jour_annee, Semaine_annee, Mois_num, Mo
is_nom, Trimestre, Annee)
    -> SELECT
    ->     Date_Generee,
    ->     DAYNAME(Date_Generee),
    ->     WEEKDAY(Date_Generee),
    ->     DAY(Date_Generee),
    ->     DAYOFYEAR(Date_Generee),
    ->     WEEK(Date_Generee),
    ->     MONTH(Date_Generee),
    ->     MONTHNAME(Date_Generee),
    ->     QUARTER(Date_Generee),
    ->     YEAR(Date_Generee)
    -> FROM (
    ->     SELECT DATE_ADD('2024-01-01', INTERVAL n DAY) AS Date_Generee, n FROM nombres WHERE DATE_ADD('2024-01-01', IN
TERVAL n DAY) <= '2024-12-31'
    -> ) AS generated_dates;
Query OK, 366 rows affected (0.01 sec)
Records: 366  Duplicates: 0  Warnings: 0
```

En conclusion, ce TP a permis de générer et d'insérer efficacement des dates dans une table MySQL en utilisant des requêtes SQL et des procédures stockées. Il a également permis de comprendre l'utilisation des fonctions MySQL pour manipuler et extraire des informations à partir de dates.

Compte rendu des Travaux Pratiques : Modélisation dimensionnelle des performances sportives

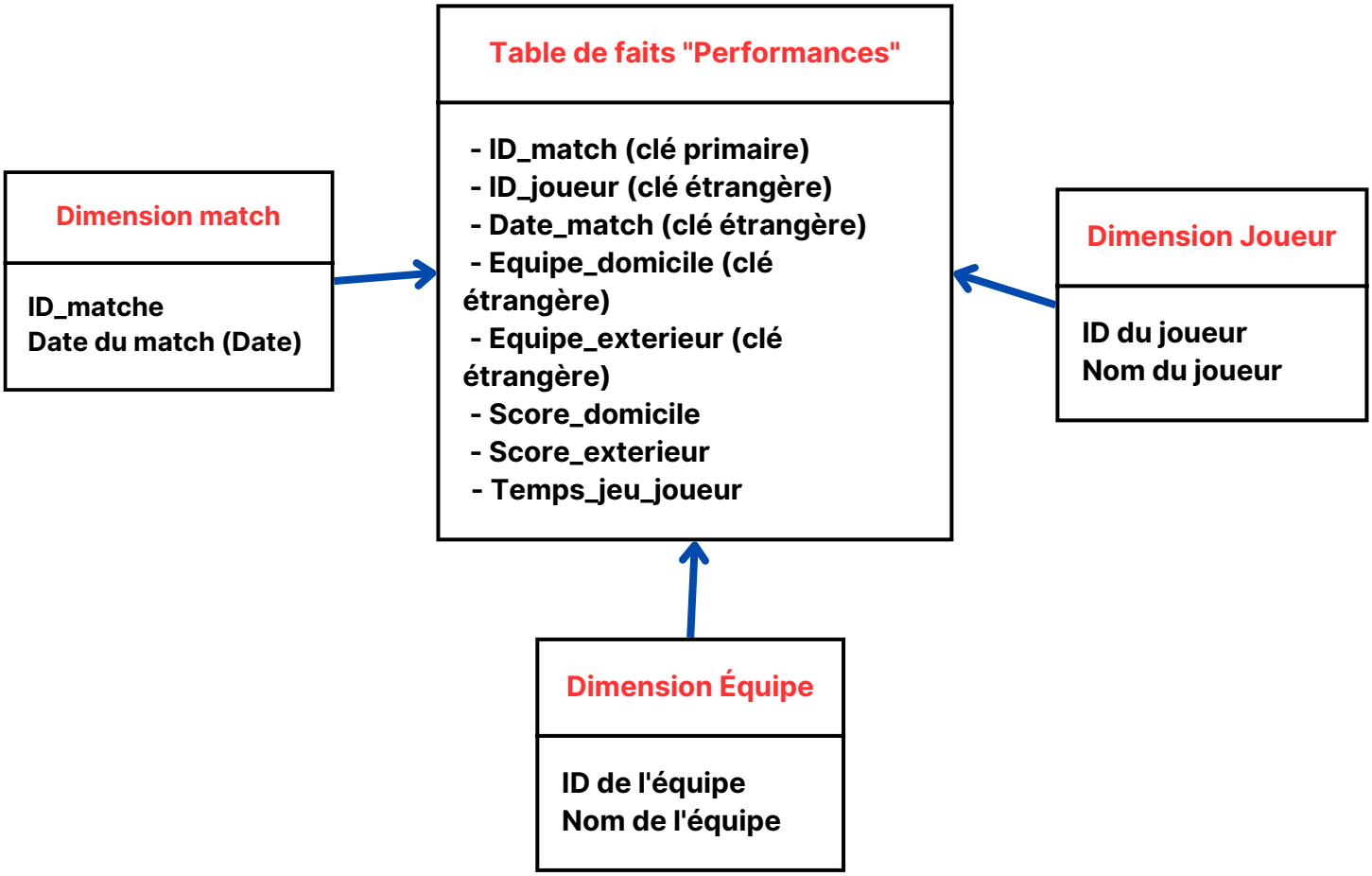
TP Exercice 3: **Modélisation dimensionnelle des performances sportives**

Objectif du TP : **L'objectif de cet exercice était de concevoir une modélisation dimensionnelle pour un système de suivi des performances sportives, puis de créer les tables de faits et de dimensions nécessaires pour stocker les données.**

1. Identification des faits et des dimensions :

Faits : **“Performances”**
Dimensions : **Équipe - Joueur - match**

2. Modélisation dimensionnelle :



3. Création de la table de faits "Performances" :

Resultat :

```
mysql> CREATE TABLE Performances (
    ->     ID_Performance INT PRIMARY KEY,
    ->     ID_Match INT,
    ->     ID_Equipe_Domicile INT,
    ->     ID_Equipe_Extérieur INT,
    ->     ID_Joueur INT,
    ->     Score_Equipe_Domicile INT,
    ->     Score_Equipe_Extérieur INT,
    ->     Temps_Jeu_Joueur INT,
    ->     FOREIGN KEY (ID_Match) REFERENCES Dimension_Match(ID_Match),
    ->     FOREIGN KEY (ID_Equipe_Domicile) REFERENCES Dimension_Equipe(ID_Equipe),
    ->     FOREIGN KEY (ID_Equipe_Extérieur) REFERENCES Dimension_Equipe(ID_Equipe),
    ->     FOREIGN KEY (ID_Joueur) REFERENCES Dimension_Joueur(ID_Joueur)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

Code :

ID_Performance	ID_Match	ID_Equipe_Domicile	ID_Equipe_Exté rieur	ID_Joueur	Score_Equipe_Domicile	Score_Equipe_Exté rieur	Temps_Jeu_Joueur
* NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL