IFT2935 – PROJET FINAL
LA COUPE DU MONDE DES NATIONS DU FOOTBALL
RAPPORT

JAD BIKARBASS AZMI
Matricule : 20252136
jad.bikarbass.azmi@umontreal.ca

DIROOSHATH THAMBOO
Matricule : 20243538
dirooshath.thamboo@umontreal.ca

HICHAM BENNOUR
Matricule : 20265790
hicham.bennour@umontreal.ca

DE-WEBERTHO DIEUDONNÉ
Matricule : 20262379
de-webertho.dieudonne@umontreal.ca

27 Avril, 2025

INTRODUCTION–

La fédération Internationale de football souhaite créer une base de données des coupes du monde des nations depuis la création de cette compétition.

Dans chaque coupe du monde, un nombre d'équipes se présentent, représentant chacune un pays. Chaque équipe est constituée d'un certain nombre de joueurs et d'un staff technique (l'entraîneur et ses collaborateurs). Chaque joueur appartenant à une équipe est identifié par un numéro. Un joueur peut recevoir des sanctions durant les matchs de la compétition (cartons jaunes ou rouges). Chaque équipe joue un nombre de matchs. Un match est caractérisé par sa date, son lieu (un stade), son rang dans la compétition, les équipent participantes ainsi que le score final. Un match est géré par un arbitre principal et trois arbitres assistants. On souhaite obtenir des statistiques diverses sur la participation des joueurs, des équipes et des arbitres dans une ou plusieurs coupes du monde.

On pourra inclure dans chaque entité tous les éléments réalistes possibles à condition de les justifier.

VIDEO–

Voici le lien vers une petite vidéo qui montre notre programme en action. La vidéo montre toutes les fonctionnalités demandées. (lien)

1 MODÉLISATION–

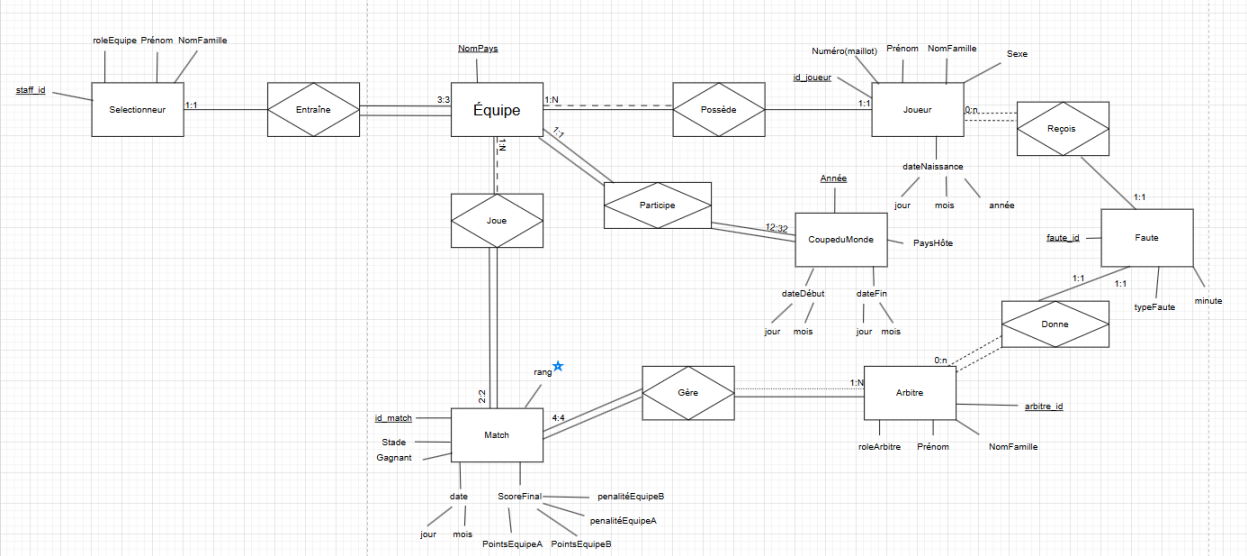

Figure 1: Modèle E/A

La Figure 1 présente le diagramme Entité/Association (E/A) que nous avons élaboré afin de modéliser la base de données de la Coupe du Monde décrite dans le sujet. Le modèle couvre l'ensemble des informations requises pour suivre les compétitions en tenant compte des contraintes organisationnelles, sportives et statistiques imposées par la FIFA.

1.1 Entités

- `CoupeDuMonde`[1]
  Permet de distinguer chaque édition du tournoi et d'enregistrer la période ainsi que le pays organisateur.
- `Équipe`
  Représente la sélection nationale engagée dans une édition donnée.
- `Joueur`
  Identifie chaque footballeur indépendamment de sa sélection à une édition donnée.
- `Selectionneur`
  Englobe le sélectionneur.
- `Arbitre`
  Représente l'arbitre qui gère un match.
- `Faute`
  Encode les sanctions infligées aux joueurs.
- `Match`
  Représente un match disputé par 2 équipes. L'attribut `rang` indique le rang dans la compétition pour ce match.

[1] Nous limitons le périmètre aux Coupes du Monde de 1930 à 2022

1.2 Associations

`Possède` Lie chaque `Équipe` aux `Joueurs` qui la composent.

`Participe` Contraint exactement 32 équipes par édition de `CoupeDuMonde`. La borne supérieure est configurable si le format change.

`Entraîne` Chaque `Sélectionneur` appartient à une unique `Équipe`. La contrainte 1:1 reflète l'exclusivité.

`Joue` Relie une `Équipe` à un `Match`. La cardinalité $(2, 2)$ côté `Match` matérialise le duel entre deux sélections.

`Gère` Spécifie que quatre `Arbitres` (un principal, trois assistants) officient chaque `Match`.

`Reçois` Attribue des `Fautes` aux `Joueurs` lors d'un match.

`SeProduit` Permet de rattacher une `Faute` au `Match` où elle a été infligée.

`Donne` Permet de rattacher une `Faute` à l'`Arbitre` qui l'a donnée.

1.3 Contraintes métiers principales

C1 Chaque match est géré par **1 arbitre principal** et **3 assistants**. Cette contrainte est modélisée par la cardinalité $(4, 4)$ sur l'association `Gère` et pourra être complétée par un déclencheur `BEFORE INSERT` côté SQL.

C2 Un joueur ne peut recevoir plus de deux avertissements (cartons jaunes) *ou* un carton rouge le même jour. Une contrainte d'invariant (`CHECK`) sera ajoutée dans la table `Faute`.

C3 Le gagnant d'un match doit être un participant de celle-ci, et il doit avoir plus de but que l'équipe adversaire, incluant tirs au but si il y a lieu, sauf en cas de match nul (Avec `FUNCTION`).

1.4 Domaines de valeurs

- *rang* ∈{phase de poules, 1/8, 1/4, 1/2, finale consolation, finale}
- *roleArbitre* ∈{principal, assistant}
- *typeFaute* ∈{cartonJaune, cartonRouge}
- *roleEquipe* ∈{selectionneur}
- *Sexe* ∈{homme, femme}

1.5 Justification des choix de modélisation

1. Séparer `Joueur` et `Équipe` permet de suivre la carrière d'un joueur sur plusieurs éditions.
2. L'entité `Faute` centralise les sanctions pour faciliter les statistiques disciplinaires. La granularité à la minute et à la période (première ou seconde mi-temps, prolongations) est nécessaire pour certaines analyses (*e.g.* fréquence des fautes par tranche de temps).
3. Le triplet d'entraîneurs est imposé par les règlements FIFA (et l'énoncé).
4. L'association `Participe` rend explicite la liste des équipes qualifiées et contraint leur nombre.

2 TRANSFORMATION–

Cette section applique les règles classiques de passage E/A $\rightarrow$ relationnel:[2]

- **R1 : Entités fortes**. Chaque entité devient une relation portant le même nom; son identifiant conceptuel devient clé primaire.
- **R2 : Attributs composés**. Les dates sont éclatées en trois attributs atomiques $(\textit{jour}, \textit{mois}, \textit{annee})$ afin de respecter la 1FN.
- **R3 : Associations** $N:M$. Les associations `Joue`, `Possède`, `Gère`, `Entraîne` et `Donne` génèrent une table associative dont la clé primaire est la concaténation des clés étrangères participantes.
- **R4 : Associations** $1:N$. Les cardinalités $(1, N)$ introduisent des clés étrangères non nulles (`#anneeCoupe` dans `Équipe`, etc.).
- **R5 : Contraintes additionnelles**. Les règles C1–C4 sont implémentées par déclencheurs et contraintes dans le SGBD.

2.1 Schéma relationnel obtenu

CoupeDuMonde(annee, paysHote, jourD, moisD, jourF, moisF)

Joueur(id_joueur, numero, #nomPays, #anneeCoupe, prenom, nomFamille, jourN, moisN, anneeN, sexe)

Équipe(id_equipe, nomPays, #anneeCoupe, #id_Selectionneur)

Match(id_match, jourM, moisM, #nomPaysA, #nomPaysB, #anneeCoupe, rang, stade, gagnant, #arbitrePrincipal_id)

ScoreFinal(#match_id, pointsEquipeA, pointsEquipeB, penaliteEquipeA, penaliteEquipeB)

StaffTechnique(id_staff, roleEquipe, #nomPays, #anneeCoupe, prenomStaff, nomStaff, jourN, moisN, anneeN, #id_equipe)

Arbitre(arbitre_id, roleArbitre, prenom, nomFamille)

Joue(#nomEquipeA, #nomEquipeB, #id_match)

Gère(#id_match, #arbitrePrincipal_Id, #arbitreAssit1_Id, #arbitreAssit2_Id, #arbitreAssit3_Id)

Faute(#id_faute, #joueur_Id, #match_Id, typefaute, faute$_minute$)

Donne(#id_arbitre, #id_faute)

Possède(#nomPays, #id_joueur)

[2]Nous suivons les prescriptions du cours IFT2935.

3 NORMALISATION–

Cette partie précise les dépendances fonctionnelles identifiées pour chaque relation, justifie leur validité et montre la décomposition éventuelle requise pour atteindre la **BCNF**. Nous terminons par le schéma relationnel final.

3.1 Méthodologie

1. Recenser les DF *essentielles* à partir des règles métier et des identifiants conceptuels.
2. Déterminer les clés candidates par calcul de fermeture.
3. Vérifier la violation éventuelle des formes normales; si une DF non triviale $X \to Y$ a un déterminant X qui n'est pas super-clé, on décompose.

3.2 Dépendances fonctionnelles par relation

Relation	**DF essentielles**	**Clés candidates**
CoupeDuMondeInfos	annee → jourD, moisD, jourF, moisF	annee
CoupeDuMondeHôte	paysHote → annee	paysHote
Equipe	id_equipe → id_Capitaine, id_Selectionneur, nomPays, anneeCoupe	id_equipe; nomPays, anneeCoupe
Joueur	id_joueur → numero, prenom, nomFamille	id_joueur ; numero, nomPays, anneeCoupe
StaffTechnique	id_staff → roleEquipe, prenomStaff, nomStaff, jourN, moisN, anneeN	id_staff
PersonneArbitre	arbitre_id → prenom, nomFamille	arbitre_id
ArbitreCoupe	arbitre_id, anneeCoupe → roleArbitre	arbitre_id, anneeCoupe
Match	id_match → jourM, moisM, nomPaysA, nomPaysB, anneeCoupe, rang, stade, gagnant, arbitrePrincipal_id	id_match
Gere	id_match → arbitrePrinpal_id, arbitreAssistant1_id, arbitreAssistant2_id, arbitreAssistant3_id	id_match, arbitre_id
ScoreFinal	match_id → ptsA, ptsB, penalA, penalB	match_id
Faute	faute_id → id_joueur, typeFaute, minute_faute	faute_id

Table 1: Dépendances fonctionnelles et clés candidates par relation

3.3 Décomposition vers la BCNF

Seules deux relations violent la BCNF:

- **CoupeDuMonde**: la DF inverse paysHote → annee impose la décomposition:

  CoupeDuMondeInfos(annee, jourD, moisD, jourF, moisF)
  CoupeDuMondeHôte(paysHote, annee)

- **Joueur**: la DF numero, nomPays, anneeCoupe → id_joueur crée une clé candidate secondaire, mais la table reste en BCNF une fois cette clé déclarée.

3.4 Schéma relationnel final

CoupeDuMondeInfos(annee, paysHote, jourD, moisD, jourF, moisF)

CoupeDuMondeHote(paysHote, #annee)

Equipe(id_equipe, nomPays, #anneeCoupe, #id_Capitaine, #id_Selectionneur)

Joueur(id_joueur, numero, #nomPays, #anneeCoupe,
prenom, nomFamille, jourN, moisN, anneeN)

StaffTechnique(id_staff, roleEquipe, #nomPays, #anneeCoupe,
prenomStaff, nomStaff, jourN, moisN, anneeN)

PersonneArbitre(arbitre_id, prenom, nomFamille)

ArbitreCoupe(#arbitre_id, #anneeCoupe, roleArbitre)

Match(id_match, jourM, moisM, #nomPaysA, #nomPaysB, #anneeCoupe,
rang, stade, gagnant, #arbitrePrincipal_id)

Gere(#id_match, #arbitrePrinpal_id, #arbitreAssistant1_id, #arbitreAssistant2_id, #arbitreAssistant3_id)

ScoreFinal(#match_id, ptsA, ptsB, penalA, penalB)

Faute(faute_id, #id_joueur, #id_match, typeFaute, minute)

4 IMPLÉMENTATION–

Toute l'implémentation est déjà dans le dépôt GitHub indiqué ci-dessous ; par ailleurs, il suffit de suivre le guide du `README.md` pour démarrer le projet.

- **Schéma normalisé** : le fichier `world-cup-bd/docker/db/backup.sql` crée l'intégralité des tables du schéma BCNF défini à la section 3, avec leurs clés primaires et étrangères.
- **Jeux de données** : les fichiers CSV réels collectés (`matches.csv`, `players.csv`, `referees.csv`, ...) se trouvent dans `world-cup-bd/data/`. Chacun contient (très) largement plus de dix tuples. (source)
- **Scripts de chargement** : les scripts Python d'insertion automatisée sont dans `world-cup-bd/docker/scripts/`. Ils lisent chaque CSV et remplissent les tables correspondantes.
- **Guide d'exécution** : le `README.md` du dépôt décrit pas-à-pas comment tout démarrer.
  Aucun réglage supplémentaire n'est requis : suivre le guide suffit pour reconstruire la base avec toutes les données.

Référence complète du dépôt : `https://github.com/ImDeeWee/Projet_World_Cup`

5 QUESTION/RÉPONSE–

5.1 Q1 – Quels sont les 10 pays ayant remporté le plus de finales, en distinguant les victoires masculines et féminines ?

Algèbre relationnelle :

$$
\begin{aligned}
S \;\leftarrow\; & \sigma_{\text{rang}='Finale'}(\textit{Matchs}) \\
& \bowtie_{gagnant\_id=id\_equipe} \textit{Equipe} \\
& \bowtie_{id\_equipe=equipe\_id} \textit{Possede} \\
& \bowtie_{joueur\_id=id\_joueur} \textit{Joueur} \\
R \;\leftarrow\; & \gamma_{\text{nomPays;count}(*)\rightarrow\text{nb\_finales, count}_{\text{M}}\rightarrow\text{finales\_M, count}_{\text{F}}\rightarrow\text{finales\_F}}(\, \pi_{id\_match,\ \text{nomPays, sexe}}\, S)
\end{aligned}
$$

SQL (`./world-cup-bd/docker/db/MostWorldCupWins.sql`) :

```
WITH sexe_finales AS (
    SELECT DISTINCT
        m.id_match,
        e.nomPays,
        j.sexe
    FROM
        matchs m
    INNER JOIN equipe e ON m.gagnant_id = e.id_equipe
    INNER JOIN possede p ON p.equipe_id = e.id_equipe
    INNER JOIN joueur j ON p.joueur_id = j.id_joueur
    WHERE m.rang = 'Finale'
)

SELECT
    nomPays,
    COUNT(*) AS nombre_finales_gagnees,
    COUNT(CASE WHEN sexe = 'M' THEN 1 END) AS finales_masculines,
    COUNT(CASE WHEN sexe = 'F' THEN 1 END) AS finales_feminines
FROM
    sexe_finales
GROUP BY
    nomPays
ORDER BY
    nombre_finales_gagnees DESC
LIMIT 10;
```

Optimisation :

- **Filtre précoce** : le prédicat `rang = 'Finale'` réduit la table *Matchs* à quelques lignes avant toute jointure.
- **Jointures sur clés** : toutes les jointures utilisent des clés étrangères ; l'optimiseur peut choisir des HASH JOIN ou NESTED LOOP indexés très rapides.
- **Agrégation légère** : après filtrage, l'agrégat ne traite que deux lignes (M/F) par finale ; le tri + `LIMIT 10` se fait entièrement en mémoire.

5.2 Q2 – Quels sont les 20 joueurs *masculins* ayant reçu le plus de cartons, en précisant combien sont jaunes et combien sont rouges ?

Algèbre relationnelle :

$S \leftarrow \sigma_{\text{sexe}='M'}(\textit{Joueur}) \bowtie_{id\_joueur=joueur\_id} \textit{Faute}$

$R \leftarrow \gamma_{id\_joueur,\ \text{prenom, nomFamille;}\ \text{count}(*)\rightarrow\text{total\_cartons,}\ \text{count}_{\text{typeFaute}='jaune'}(*)\rightarrow\text{cartons\_jaunes,}\ \text{count}_{\text{typeFaute}='rouge'}(*)\rightarrow\text{cartons\_rouges}}(S)$

SQL (`./world-cup-bd/docker/db/BlackAirForceEnergy.sql`) :

```
SELECT
    j.prenom,
    j.nomfamille,
    COUNT(*) AS total_cartons,
    COUNT(CASE WHEN f.typefaute = 'jaune' THEN 1 END) AS cartons_jaunes,
    COUNT(CASE WHEN f.typefaute = 'rouge' THEN 1 END) AS cartons_rouges
FROM
    faute f
INNER JOIN
    joueur j ON f.joueur_id = j.id_joueur
WHERE
    j.sexe = 'M'
GROUP BY
    j.id_joueur, j.prenom, j.nomfamille
ORDER BY
    total_cartons DESC, cartons_rouges DESC
LIMIT 20;
```

Optimisation :

- **Filtre précoce** : la sélection `sexe = 'M'` est appliquée sur *Joueur* avant la jointure, divisant immédiatement le volume de données.
- **Index ciblés** : index fonctionnel ou composé *Joueur(sexe, id_joueur)* accélère le filtre, tandis qu'un index *Faute(joueur_id, typeFaute)* optimise la jointure et les comptes conditionnels.
- **Jointure clé–clé étrangère** : la condition `j.id_joueur = f.joueur_id` permet à l'optimiseur de choisir un HASH JOIN ou un NESTED LOOP indexé très efficace.
- **Agrégation sur clé primaire** : le `GROUP BY id_joueur` utilise une clé, si bien qu'un HASH AGGREGATE tient en mémoire.
- **Tri final + LIMIT 20** : après agrégation, le résultat contient une ligne par joueur ; le tri se fait donc en RAM avant l'application du `LIMIT`.

5.3 Q3 – Pour chaque personne ayant d'abord joué en équipe nationale puis exercé comme sélectionneur, indiquer (1) les années où elle a joué ; (2) le pays qu'elle représentait comme joueur ; (3) les années où elle a été sélectionneur ; (4) le pays qu'elle a entraîné.

Algèbre relationnelle :

$$\text{JS} \leftarrow \sigma_{\text{lower}(J.prenom)=\text{lower}(S.prenomStaff) \,\wedge\, \text{lower}(J.nomFamille)=\text{lower}(S.nomStaff)} \big(Joueur \bowtie Selectionneur \big)$$

$$\text{JPlayed} \leftarrow \text{JS} \bowtie_{id\_joueur=joueur\_id} Possede \bowtie_{equipe\_id=id\_equipe} Equipe \quad \text{(années et pays comme joueur)}$$

$$\text{JCoach} \leftarrow \text{JS} \bowtie_{id\_staff=S.id\_staff} Selectionneur \bowtie_{id\_equipe=E.id\_equipe} Equipe \quad \text{(années et pays comme sélectionneur)}$$

$$R \leftarrow \gamma_{\text{prenomStaff, nomStaff, JPlayed.nomPays, JCoach.nomPays;}}$$
$$\gamma_{\text{list\_agg distinct}(JPlayed.anneeCoupe)\rightarrow\text{années\_joueur, list\_agg distinct}(JCoach.anneeCoupe)\rightarrow\text{années\_coach}}$$
$$(\text{JPlayed} \bowtie \text{JCoach})$$

SQL (`./world-cup-bd/docker/db/PlayerTurnedIntoCoach.sql`) :

```
WITH joueurs_selectionneurs AS (
    SELECT
        s.id_staff,
        s.prenomstaff,
        s.nomstaff,
        j.id_joueur
    FROM
        selectionneur s
    INNER JOIN
        joueur j
      ON LOWER(j.prenom) = LOWER(s.prenomstaff)
     AND LOWER(j.nomfamille) = LOWER(s.nomstaff)
)

SELECT
    js.prenomstaff,
    js.nomstaff,
    -- 1) Les années où il a joué
    STRING_AGG(DISTINCT (e_joueur.anneecoupe)::text, ', '
               ORDER BY (e_joueur.anneecoupe)::text)
      AS annees_joueur,
    -- 2) Le pays où il a joué
    e_joueur.nompays         AS nompays_joueur,
    -- 3) Les années où il a été sélectionneur
    STRING_AGG(DISTINCT (e_sel.anneecoupe)::text, ', '
               ORDER BY (e_sel.anneecoupe)::text)
      AS annees_selectionneur,
    -- 4) Le pays qu'il entraîne
    e_sel.nompays            AS nompays_selectionneur
FROM
    joueurs_selectionneurs js
    -- toutes les éditions où il a joué
    INNER JOIN possede p
      ON p.joueur_id = js.id_joueur
    INNER JOIN equipe e_joueur
      ON e_joueur.id_equipe = p.equipe_id
    -- sa/ ses sélections
    INNER JOIN selectionneur s
      ON s.id_staff = js.id_staff
    INNER JOIN equipe e_sel
      ON e_sel.id_equipe = s.id_equipe
GROUP BY
    js.prenomstaff,
```

```
    js.nomstaff,
    e_joueur.nompays,
    e_sel.nompays
ORDER BY
    js.nomstaff,
    js.prenomstaff;
```

Optimisation :

- **CTE pré-filtrée** : la CTE `joueurs_selectionneurs` isole d'abord la jointure coûteuse *nom* + *prénom*, puis ré-utilise le résultat, évitant de répéter la comparaison.
- **Index fonctionnels** : index sur `lower(prenom)` et `lower(nomFamille)` (dans *Joueur* et *Selectionneur*) rendent la correspondance insensible à la casse indexable.
- **Jointures clé–clé étrangère** : toutes les autres jointures s'appuient sur des clés primaires/étrangères (*Possede*, *Equipe*), ce qui autorise HASH ou NESTED-LOOP JOINS rapides.
- **Projection push-down** : seuls `id_joueur` et `id_staff` transitent après la CTE, réduisant la largeur des tuples avant les jointures suivantes.
- **Agrégation compacte** : l'agrégat final (`STRING_AGG DISTINCT`) s'effectue sur un ensemble restreint (une ligne par doublon joueur-sélectionneur), donc le tri interne et le group-by restent en mémoire.

5.4 Q4 – Quelles sont les 25 personnes qui ont le plus grand nombre de coupes remportées (totales joueur + sélectionneur) et, pour chacune, combien de finales elles ont disputées au total ?

Algèbre relationnelle :

$$\text{PWin} \leftarrow \gamma_{\text{personne;count}(*)\rightarrow\text{wins\_player}}\Big(\sigma_{\text{rang}='Finale'}(\textit{Matchs}) \bowtie_{gagnant\_id=equipe\_id} \textit{Possede} \bowtie_{joueur\_id=id\_joueur} \textit{Joueur}\Big)$$

$$\text{CWin} \leftarrow \gamma_{\text{personne;count}(*)\rightarrow\text{wins\_coach}}\Big(\sigma_{\text{rang}='Finale'}(\textit{Matchs}) \bowtie_{gagnant\_id=id\_equipe} \textit{Selectionneur}\Big)$$

$$\text{PFin} \leftarrow \gamma_{\text{personne;count distinct}(id\_match)\rightarrow\text{finals\_player}}\Big(\begin{array}{l}\sigma_{\text{rang}='Finale'}(\textit{Matchs}) \\ \bowtie_{id\_equipeA=equipe\_id \vee id\_equipeB=equipe\_id} \textit{Possede} \\ \bowtie_{joueur\_id=id\_joueur} \textit{Joueur}\end{array}\Big)$$

$$\text{CFin} \leftarrow \gamma_{\text{personne;count distinct}(id\_match)\rightarrow\text{finals\_coach}}\Big(\begin{array}{l}\sigma_{\text{rang}='Finale'}(\textit{Matchs}) \\ \bowtie_{id\_equipeA=id\_equipe \vee id\_equipeB=id\_equipe} \textit{Selectionneur}\end{array}\Big)$$

$$\text{Comb} \leftarrow \gamma_{\text{personne;}} \gamma_{\begin{array}{l}\text{sum}(wins\_player \cup 0 \cup wins\_coach)\rightarrow\text{total\_wins,} \\ \text{sum}(finals\_player \cup 0 \cup finals\_coach)\rightarrow\text{total\_finals}\end{array}}\Big(\text{PWin} \cup \text{CWin} \cup \text{PFin} \cup \text{CFin}\Big)$$

SQL (`./world-cup-bd/docker/db/TheWorldCupGoats.sql`) :

```
WITH
-- Victoires en tant que joueur
wins_player AS (
  SELECT
    j.prenom || ' ' || j.nomfamille AS personne,
    COUNT(*) AS wins_player
  FROM possede p
  JOIN joueur j ON j.id_joueur = p.joueur_id
  JOIN matchs m
    ON m.rang = 'Finale'
   AND m.gagnant_id = p.equipe_id
  GROUP BY personne
),
-- Victoires en tant que coach
wins_coach AS (
  SELECT
    s.prenomstaff || ' ' || s.nomstaff AS personne,
    COUNT(*) AS wins_coach
  FROM selectionneur s
  JOIN equipe e ON e.id_equipe = s.id_equipe
  JOIN matchs m
    ON m.rang = 'Finale'
   AND m.gagnant_id = e.id_equipe
  GROUP BY personne
),
-- Finales jouées comme joueur
finals_player AS (
  SELECT
    j.prenom || ' ' || j.nomfamille AS personne,
    COUNT(DISTINCT m.id_match) AS finals_player
  FROM possede p
  JOIN joueur j ON j.id_joueur = p.joueur_id
  JOIN matchs m
    ON m.rang = 'Finale'
   AND (m.id_equipea = p.equipe_id OR m.id_equipeb = p.equipe_id)
  GROUP BY personne
),
```

```
-- Finales jouées comme coach
finals_coach AS (
  SELECT
    s.prenomstaff || ' ' || s.nomstaff AS personne,
    COUNT(DISTINCT m.id_match) AS finals_coach
  FROM selectionneur s
  JOIN equipe e ON e.id_equipe = s.id_equipe
  JOIN matchs m
    ON m.rang = 'Finale'
   AND (m.id_equipea = e.id_equipe OR m.id_equipeb = e.id_equipe)
  GROUP BY personne
),
-- On fusionne les stats
combined AS (
  SELECT
    COALESCE(wp.personne, wc.personne) AS personne,
    COALESCE(wp.wins_player, 0)  + COALESCE(wc.wins_coach, 0)  AS total_wins,
    COALESCE(fp.finals_player, 0) + COALESCE(fc.finals_coach, 0) AS total_finals
  FROM wins_player wp
  FULL  JOIN wins_coach  wc USING (personne)
  FULL  JOIN finals_player fp USING (personne)
  FULL  JOIN finals_coach  fc USING (personne)
)
SELECT
  personne,
  total_wins      AS coupes_gagnees,
  total_finals    AS finales_participees
FROM combined
ORDER BY total_wins DESC, total_finals DESC LIMIT 25;
```

Optimisation :

- **Filtre sélectif** : le prédicat `rang = 'Finale'` réduit *Matchs* à une poignée de lignes, avant toute jointure ou agrégation.
- **CTE spécialisées** : quatre agrégations distinctes (`wins_player`, `wins_coach`, `finals_player`, `finals_coach`) traitent chacune un petit ensemble, ce qui permet à l'optimiseur de les matérialiser directement en mémoire.
- **Index composites** :
  - *Matchs(rang, gagnant_id)* pour les victoires,
  - *Matchs(rang, id_equipeA)*, *Matchs(rang, id_equipeB)* pour les participations, accélèrent la recherche des finales pertinentes.
- **Jointures clé–clé étrangère** : les liens `Possede(joueur_id)`, `Selectionneur(id_equipe)` autorisent des HASH JOINS ou NESTED LOOPS indexés efficaces.
- **Fusion par FULL JOIN** : chaque CTE produit peu de lignes (au plus une par personne); la fusion se fait donc à coût négligeable, suivie d'un agrégat global pour calculer `total_wins` et `total_finals`.
- **Tri final +** `LIMIT 25` : l'ensemble combiné reste petit (une ligne par personne); le tri s'effectue en RAM avant d'extraire les 25 premiers résultats.

6 DÉVELOPPEMENT–

Nous avons développé une petite application web pour répondre à l'ensemble des quatre questions de la section 5, comme demandé.

- **Langages utilisés :** Node.js (backend) et React (frontend).
- **Interface :** la page web contient quatre boutons, chacun correspondant à une question.
- **Fonctionnement :** lorsqu'on clique sur un bouton, l'application interroge la base de données et affiche directement les résultats à l'écran.
- **Déploiement :** toutes les instructions pour démarrer la page web, installer les dépendances et lancer le serveur sont disponibles dans le `README.md` du dépôt GitHub.

L'application est conçue pour être légère et facilement exécutable localement, en lien direct avec la base PostgreSQL créée dans les sections précédentes.

6.1 Aperçu

nompays	nombre_finales_gagnees	finales_masculines	finales_feminines
Brazil	5	5	0
Italy	4	4	0
United States	4	0	4
Argentina	3	3	0
Germany	3	1	2
West Germany	3	3	0
France	2	2	0
England	1	1	0
Spain	1	1	0
Japan	1	0	1

Figure 2: Résultat affiché de la question 1

Un guide utilisateur complet se trouve dans le `README` du dépôt.