Rapport

Françon Clement et Malih Amine

2025-10-19

Sommaire

1. Prompt

Prompt sous format R.I.C.A.R.D.O, envoyé à Chat GPT 5

Tu travailles dans le domaine de **jeu vidéo compétitif (esport)**. Ton **entreprise** a comme activité de **collecter, stocker et analyser des données liées aux joueurs, aux matchs, aux champions et aux statistiques de League of Legends**. C'est une **plateforme d'analyse de données esport** comme **OP.GG, League of Graphs ou U.GG**.
La base doit contenir : les informations sur les joueurs (pseudonyme, tag Riot, niveau, rang), les équipes, les matchs (date, durée, mode de jeu), les champions choisis et bannis, les objets et runes utilisés, ainsi que les statistiques de performance (kills, deaths, assists, cs, vision, dégâts, etc.).
Inspire-toi des sites suivants :
→ https://www.op.gg,
→ https://www.leagueofgraphs.com,
→ https://u.gg.

Ton **entreprise** veut appliquer MERISE pour concevoir un système d'information. Tu es chargé de la partie analyse, c'est-à-dire de collecter les besoins auprès de l'entreprise. Elle a fait appel à un étudiant en ingénierie informatique pour réaliser ce projet, tu dois lui fournir les informations nécessaires pour qu'il applique ensuite lui-même les étapes suivantes de conception et développement de la base de données.

D'abord, établis les règles de gestions des données de ton **entreprise**, sous la forme d'une liste à puce. Elle doit correspondre aux informations que fournit quelqu'un qui connaît le fonctionnement de l'entreprise, mais pas comment se construit un système d'informations.

Ensuite, à partir de ces règles, fournis un dictionnaire de données brutes avec les colonnes suivantes, regroupées dans un tableau : signification de la donnée, type, taille en nombre de caractères ou de chiffres. Il doit y avoir entre 25 et 35 données. Il sert à fournir des informations supplémentaires sur chaque données (taille et type) mais sans a priori sur comment les données vont être modélisées ensuite.

Fournis donc les règles de gestion et le dictionnaire de données.

2. Version prise en compte

2.1. Dictionnaire de données

	Signification de la donnée	Type	
1	Pseudonyme actuel (gameName)	Texte	30
2	Tag Riot (tagLine, ex. « EUW »)	Texte	10
3	Région/Shard du compte (EUW, NA…)	Enum	5
4	Niveau de compte (summoner level)	Entier	3
5	Rang – Tier (Iron… Challenger) à la date du match	Enum	12
6	Rang – Division (IV–I) à la date du match	Enum	2
7	League Points (LP) à la date du match	Entier	3
8	Rôle principal déclaré (TOP/JGL/MID/ADC/SUP)	Enum	3
9	Équipe ID (si joueur en équipe)	Texte	36
10	Nom d'équipe	Texte	50
11	ID du match (ex. EUW1_1234567890)	Texte	30
12	Date-heure du match (UTC)	Date-Heure	—
13	Mode/Queue (Normal, Ranked Solo, Flex, Tournoi…)	Enum	20
14	Version de patch (ex. 14.18)	Texte	10
15	Durée du match (secondes)	Entier	5
16	Côté (BLUE/RED) pour l'équipe/joueur	Enum	4
17	Champion ID (numérique référentiel)	Entier	4
18	Nom du champion	Texte	30
19	Liste des bans (IDs champions par côté)	JSON	200
20	Kills	Entier	2
21	Deaths	Entier	2
22	Assists	Entier	2
23	CS total (minions + monstres)	Entier	4
24	Or total gagné	Entier	6
25	Dégâts infligés aux champions	Entier	7
26	Score de vision	Entier	3
27	Objets finaux (IDs objets slots 1–6)	JSON	120
28	Runes principales (chemin + runes)	JSON	120
29	Runes secondaires (chemin + runes)	JSON	120
30	Sorts d'invocateur (deux IDs)	JSON	20
31	Résultat du joueur/équipe (win/lose)	Booléen	1

	Signification de la donnée	**Type**	
32	Lane/Position jouée (TOP/JGL/MID/BOT/SUP)	Enum	3

2.2. Règles de gestion

- Un joueur est identifié de manière unique par son **nom d'utilisateur**
- Un joueur peut appartenir à 0 ou 1 équipe à un instant donné (file solo/duo/flex : équipe = indéfinie), et l'historique des appartenances doit être traçable (dates d'entrée/sortie).
- Pour chaque joueur, on stocke les rôles joués (top, jungle, mid, ADC, support) et le rôle principal déclaré ou inféré.
- Les rangs classés (tier/division/LP) sont capturés à la date du match afin de pouvoir reconstituer le contexte compétitif.
- Un match est identifié de manière unique (ID de match Riot) et possède une date/heure, un mode/queue, un patch et une durée.
- Un match oppose deux équipes ; on stocke le vainqueur
- Pour chaque match, on enregistre les champions bannis par côté et les champions choisis par joueur.
- Pour chaque joueur dans un match, on saisit les statistiques: K/D/A, CS, dégâts infligés, or gagné, score de vision, objectifs participés, etc.
- Les objets et runes utilisés par un joueur dans un match sont conservés (objets finaux et/ou timeline d'achats si disponible).
- Les sorts d'invocateur utilisés par joueur sont saisis.
- Les régions (EUW, EUNE, NA, KR, etc.), serveurs et langues sont normalisés via référentiels.
- Les champions, objets, runes et patchs proviennent de référentiels (Data Dragon ou équivalent) versionnés par patch.
- Les données sensibles de compte (email, IP, etc.) ne sont pas collectées ; seules les données publiques/ingame sont conservées.
- Les doublons sont évités via clés naturelles (PUUID, Match ID) et contrôles d'intégrité.
- Toute statistique agrégée (moyennes, pourcentages, tendances) est recalculable à partir des données brutes.

3. Normalisation de la Base de Données

Dans cette section, nous démontrons rigoureusement que la base de données League of Legends est en troisième forme normale (3FN).

3.1. Preuve : Première Forme Normale (1FN)

Définition : Première Forme Normale

Une relation est en 1FN si et seulement si tous ses attributs sont atomiques, c'est-à-dire qu'aucun attribut ne contient de valeurs multiples, de structures composées ou de listes.

Preuve : 1FN de la base de données

Examinons l'atomicité des attributs dans chaque table :

Tables principales :

- **Joueur** : Les attributs `pseudo` (VARCHAR), `niveau_compte` (INT), `rang_palier` (VARCHAR), `division_palier` (CHAR), `points_ligue` (INT) sont tous des types scalaires simples.
- **Champion** : `id_champion` (VARCHAR) et `nom_affiche` (VARCHAR) sont atomiques.
- **Match** : `date_match` (DATETIME) et `duree_secondes` (INT) sont des valeurs simples.
- **PageStatistiqueJoueur** : Tous les attributs (`kills`, `deaths`, `assists`, `cs_total`, etc.) sont des entiers ou décimaux simples représentant des valeurs uniques.
- **Rune** : Bien que plusieurs attributs décrivent les arbres de runes (`arbre_principal1`, `arbre_principal2`, etc.), chacun est une chaîne atomique distincte.

Aucune table ne contient d'attributs multi-valués tels que des tableaux, des ensembles ou des structures JSON. Chaque attribut représente une valeur unique et indivisible dans le contexte métier.

Corollaire : Atomicité garantie

La base de données respecte strictement la 1FN car tous les attributs de toutes les relations sont atomiques.

3.2. Preuve : Deuxième Forme Normale (2FN)

Définition : Deuxième Forme Normale

Une relation est en 2FN si :
1. Elle est en 1FN
2. Tout attribut non-clé dépend fonctionnellement de la totalité de la clé primaire (absence de dépendances partielles)

Preuve : 2FN de la base de données

La base étant en 1FN, analysons les dépendances partielles.

Cas 1 : Tables avec clé primaire simple

Les tables `Joueur`, `Champion`, `Match_`, `ModeJeu`, `Item`, `Rune`, `ItemJoueur`, `RuneJoueur` possèdent une clé primaire composée d'un seul attribut.

Pour ces tables, il estimpossible qu'un attribut non-clé dépende partiellement de la clé, car la clé n'a qu'une seule composante. Aucune dépendance partielle n'est possible.

Cas 2 : Tables avec clé primaire composite

Analysons rigoureusement chaque table avec clé composite :

Clé primaire : (id_match, id_joueur)

Attributs non-clés : lane, resultat_joueur, niveau_fin, kills, deaths, assists, cs_total, degats_infliges, score_vision, cs_par_minute, degats_par_minute, vision_par_minute, side

Analyse des dépendances :

- Les statistiques de jeu (kills, deaths, assists, degats_infliges, etc.) caractérisent la performance d'un joueur spécifique dans un match spécifique.
- On ne peut pas déterminer `kills` uniquement avec `id_match` (car 10 joueurs participent au match, chacun avec ses propres kills).
- On ne peut pas déterminer `kills` uniquement avec `id_joueur` (car le joueur participe à plusieurs matchs avec des performances différentes).
- Formellement : (id_match) $\nrightarrow$ kills et (id_joueur) $\nrightarrow$ kills
- Mais : (id_match, id_joueur) $\rightarrow$ kills

Cette logique s'applique à tous les attributs de statistiques.

Conclusion : Aucune dépendance partielle. La table est en 2FN.

Corollaire : 2FN globale

La base de données est en 2FN car aucune dépendance fonctionnelle partielle n'existe dans aucune de ses relations.

3.3. Preuve : Troisième Forme Normale (3FN)

Définition : Troisième Forme Normale

Une relation est en 3FN si :
1. Elle est en 2FN
2. Aucun attribut non-clé ne dépend transitivement de la clé primaire

Preuve : 3FN de la base de données

La base étant en 2FN, vérifions l'absence de dépendances transitives.

Clé : (id_match, id_joueur)

Analyse des dépendances potentiellement transitives :

Examinons si certaines statistiques pourraient être déduites d'autres :

1. **cs_par_minute vs cs_total** :
   - `cs_par_minute` est calculé à partir de `cs_total` et de la durée du match
   - Mais la durée du match n'est pas stockée dans cette table (elle est dans `Match_`)
   - Donc : cs_total $\nrightarrow$ cs_par_minute dans cette table
2. **degats_par_minute vs degats_infliges** :
   - Même raisonnement : nécessite la durée du match
3. **KDA (kills/deaths/assists)** :
   - Ces trois valeurs sont indépendantes et ne se déterminent pas mutuellement
   - Connaître les kills ne permet pas de déduire les deaths ou assists
4. **lane vs side** :
   - La lane (TOP, MID, etc.) ne détermine pas le côté (BLEU/ROUGE)
   - Ces attributs sont indépendants

Conclusion : Toutes les statistiques dépendent directement et uniquement de la clé primaire (id_match, id_joueur). Aucune chaîne de dépendances transitives n'existe.

Preuve : StatistiquesChamp en 3FN

Clé : (id_joueur, id_champion, id_mode)

Analyse :

Les attributs `note`, `taux_victoire`, `parties_jouees`, `kda_moyen`, `taux_ban` sont tous des mesures statistiques indépendantes :

- Le `taux_victoire` ne détermine pas le `kda_moyen` (un joueur peut avoir un bon winrate avec un KDA moyen)
- Le `kda_moyen` ne détermine pas le `taux_ban` (un champion peut être ban pour d'autres raisons)
- La `note` est une évaluation globale mais ne dérive pas mécaniquement des autres attributs

Formellement, il n'existe pas de dépendance :

$$\text{taux\_victoire} \not\mapsto \text{kda\_moyen}$$

$$\text{kda\_moyen} \not\mapsto \text{taux\_ban}$$

Conclusion : Pas de transitivité. La table est en 3FN.

Preuve : Joueur en 3FN

Clé : id_joueur

Analyse délicate :

Les attributs `rang_palier`, `division_palier`, et `points_ligue` sont tous liés au système de classement. Vérifions s'il y a transitivité :

1. **rang_palier → division_palier ?**
   - Non, car plusieurs divisions existent dans un même rang (Division I, II, III, IV)
   - Le rang seul ne détermine pas la division
2. **rang_palier → points_ligue ?**
   - Non, car les points varient de 0 à 100 indépendamment du rang
   - Deux joueurs au même rang peuvent avoir des points différents
3. **division_palier → points_ligue ?**
   - Non, pour les mêmes raisons

Ces trois attributs représentent l'état actuel du classement du joueur. Ils sont tous directement déterminés par l'identité du joueur (id_joueur) à un instant donné, sans relation de dépendance entre eux.

$$\text{id\_joueur} \to \text{rang\_palier}$$

$$\text{id\_joueur} \to \text{division\_palier}$$

$$\text{id\_joueur} \to \text{points\_ligue}$$

Mais :

$$\text{rang\_palier} \not\mapsto \text{division\_palier} \not\mapsto \text{points\_ligue}$$

Conclusion : Pas de transitivité. La table est en 3FN.

Conclusion :

La base de données League of Legends est en troisième forme normale (3FN).

Démonstration :

1. La base est en 1FN (tous les attributs sont atomiques) ✓
2. La base est en 2FN (aucune dépendance partielle) ✓
3. La base est en 3FN car pour toutes les tables :
   - Les attributs non-clés dépendent directement de la clé primaire
   - Aucune dépendance fonctionnelle transitive n'existe
   - Les attributs sont soit atomiques, soit des mesures indépendantes
4. Vérification exhaustive :
   - Tables à clé simple : 3FN par construction
   - PageStatistiqueJoueur : statistiques indépendantes ✓
   - Joueur : attributs de classement indépendants ✓
   - Tables de jonction : pas d'attributs transitifs ✓

Par conséquent, la base de données satisfait toutes les conditions de la 3FN.

■

4. Scénarios d'utilisations

Dans cette partie, on s'occupera que des requêtes pertinentes.C'est à dire qui répondent à un besoin selon un contexte donné.

4.1. Requête Perspective Joueur

Mise en contexte
Un joueur passionné de League of Legends, classé Platine I, souhaite progresser vers les rangs Diamant ou Maître. Il a un pool restreint de champions mécaniquement exigeants : Yasuo, Yone et Irelia. Il désire identifier sur lesquels concentrer ses efforts, en se basant sur des données réelles de performance (winrate, KDA, itemisation et runes).

Exemple : Objectif de la requête

La **Requête 1** agrège les statistiques issues de la table `StatistiquesChamp`, en filtrant sur le mode classé (`mode1`) et les trois champions cibles. Elle calcule :

- Le **winrate moyen** et le **KDA moyen** sur l'ensemble du ladder
- Le **plus haut rang** atteint par chaque champion
- Le **build d'items** le plus fréquent et les **runes dominantes**

4.1.1. Exemple de sortie

Champion	Winrate moyen	KDA moyen	Plus haut rang	Build recommandé	Runes dominantes
Yasuo	51.2%	3.4	Maître I	Kraken → IE → Death's Dance	Conqueror / Resolve
Yone	52.5%	3.6	Diamant II	Shieldbow → IE → BT	Lethal Tempo / Domination
Irelia	49.8%	3.1	Diamant III	BORK → Wit's End → GA	Conqueror / Resolve

Tableau 1. – Comparaison des statistiques par champion

Remarque : Interprétation des résultats

Analyse comparative :

1. **Yone** présente le meilleur winrate (52.5%) et le meilleur KDA (3.6) → **Champion prioritaire**
2. **Yasuo** atteint les rangs les plus élevés (Maître I) → Potentiel élevé, mais nécessite plus d'investissement
3. **Irelia** affiche des statistiques plus faibles (49.8% WR, 3.1 KDA) → Champion à éviter ou nécessitant beaucoup plus de pratique

4.2. Requête 2 : Perspective « Data Analyst »

Définition : Contexte

Un data analyst e-sport au sein d'une équipe professionnelle souhaite repérer les joueurs individuels les plus performants sur certains champions, afin de proposer des essais ou d'analyser les tendances de la méta compétitive.

Exemple : Objectif de la requête

La **Requête 2** combine des fonctions et des sous-requêtes corrélées pour :

- Identifier les joueurs **au-dessus de la moyenne globale** sur leur champion
- Calculer un **score de performance** intégrant winrate, KDA et volume de parties :

$$\text{score} = (\text{taux\_victoire} \times 0.6) + (\text{KDA} \times 10) + \left(\frac{\text{parties}}{20}\right)$$

- Filtrer les joueurs actifs en **haut elo** (Émeraude, Diamant, Maître)

4.2.1. Exemple de sortie

Joueur	Rang	Champion	Winrate	KDA	Score performance
ProPlayer42	Maître I	Yasuo	58.5%	4.2	112.3
DiamondMain	Diamant II	Yone	61.2%	3.8	116.8
EmeraldSmurf	Émeraude I	Irelia	55.0%	3.5	103.5

Tableau 2. – Joueurs les plus performants identifiés

Remarque : Exploitation pratique

Applications concrètes :

1. **Identifier les joueurs constants** → Haut score + grand volume de parties = fiabilité
2. **Détecter les spécialistes** → « One-tricks » à fort potentiel sur un champion spécifique
3. **Comparer à la moyenne globale** → Un joueur à 61.2% WR vs 50% global = surperformance significative

Décisions possibles :

- Proposer un essai à un joueur performant dans l'équipe
- Étudier les profils des joueurs à KDA élevé pour comprendre leurs builds
- Suivre l'évolution des performances dans le temps

4.3. Requête 3 : Perspective « Game Designer »

Définition : Contexte

L'équipe d'équilibrage chez Riot Games ou un studio d'analyse communautaire veut identifier les champions **surperformants** à corriger dans les patchs à venir.

Remarque : Utilisateur concerné

Profil :

- Game designer
- Équipe de développement en charge de l'équilibrage

Exemple : Objectif de la requête

La **Requête 3** utilise une approche statistique pour repérer les anomalies :

1. Calcul de l'écart par rapport à la moyenne globale) sur :
   - Winrate, KDA, taux de ban, pickrate
2. On construit un score mettant en avant les anomalies :

$$0.45 \times \text{z\_wr} + 0.25 \times \text{z\_kda} + 0.20 \times \text{z\_ban} + 0.10 \times \text{p\_pickrate}$$

4.3.1. Exemple de sortie

Champion	Winrate	z_wr	z_ban	Anomaly score	Goredrinker ?	Flag OP
K'Sante	54.2%	2.3	2.8	2.45	✅	TRUE
Briar	53.8%	2.1	1.9	2.12	✅	TRUE
Zeri	52.5%	1.8	2.5	2.01	❌	TRUE

Tableau 3. – Champions détectés comme statistiquement anormaux

Remarque : Interprétation des résultats

Analyse :

- Ces champions ont des performances **statistiquement anormales** (> 2 écarts-types)
- Ils ont à la fois : winrate élevé + haut taux de ban + popularité haute
- Le flag `use_goredrinker` indique une synergie d'objet problématique

Exemple K'Sante :

- z_wr $= 2.3 \Rightarrow$ Winrate 2.3 écarts-types au-dessus de la moyenne
- z_ban $= 2.8 \Rightarrow$ Taux de ban élevé
- Anomaly score $= 2.45 \Rightarrow$ Alerte critique