HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA VIỄN THÔNG 1

_____ o0o _____

BÀI TIỂU LUẬN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Đề tài: Xây dựng hệ cơ sở dữ liệu phân tích kết quả giải bóng đá Ngoại hạng Anh.

Số thứ tự nhóm: 09

Nguyễn Văn Tiến **MSSV: D23DCKD068**
Đỗ Ngọc Quý **MSSV: D23DCKD058**
Đỗ Hữu Mạnh **MSSV: D23DCKD042**

Giảng viên hướng dẫn : **PGS.TS. Lê Hải Châu**
: **TS. Vũ Thị Thúy Hà**

HÀ NỘI, 11/2025

Mục lục

Danh sách hình vẽ

Danh sách bảng

Phần I

MỞ ĐẦU

Giải bóng đá Ngoại hạng Anh (Premier League) là một trong những giải đấu bóng đá hàng đầu thế giới, thu hút hàng triệu người hâm mộ và tạo ra một lượng dữ liệu khổng lồ về các trận đấu, cầu thủ, đội bóng và kết quả thi đấu. Việc phân tích dữ liệu này không chỉ giúp các câu lạc bộ, huấn luyện viên và chuyên gia chiến lược đưa ra quyết định mà còn hỗ trợ người hâm mộ, nhà báo và các nhà phân tích thể thao hiểu rõ hơn về xu hướng, hiệu suất và dự báo kết quả. Trong bối cảnh công nghệ số phát triển mạnh mẽ, việc xây dựng một hệ cơ sở dữ liệu (CSDL) chuyên biệt để lưu trữ và phân tích kết quả giải bóng đá Ngoại hạng Anh trở nên cần thiết.

Đề tài "Xây dựng hệ cơ sở dữ liệu phân tích kết quả giải bóng đá Ngoại hạng Anh" tập trung vào việc thiết kế và triển khai một hệ thống CSDL hiệu quả, sử dụng các công nghệ và phương pháp hiện đại để xử lý dữ liệu lớn (big data) liên quan đến giải đấu. Mục tiêu chính là tạo ra một nền tảng dữ liệu có khả năng lưu trữ thông tin về các trận đấu, cầu thủ, đội bóng, thống kê bàn thắng, thẻ phạt, và các chỉ số hiệu suất khác, đồng thời hỗ trợ các truy vấn phức tạp để phân tích, báo cáo và dự đoán.

Tầm quan trọng của đề tài nằm ở việc áp dụng CSDL để chuyển đổi dữ liệu thô thành thông tin có giá trị. Ví dụ, qua hệ thống này, có thể phân tích xu hướng thi đấu của một đội bóng qua các mùa giải, đánh giá hiệu suất cầu thủ dựa trên dữ liệu lịch sử, hoặc dự báo kết quả dựa trên các mô hình thống kê. Đề tài sẽ dựa trên các nguyên tắc cơ bản của quản lý CSDL, từ mô hình hóa dữ liệu đến tối ưu hóa truy vấn, nhằm đảm bảo hệ thống hoạt động hiệu quả, đáng tin cậy và dễ mở rộng.

Phần II

NỘI DUNG CHÍNH

CHƯƠNG 1. Cơ sở lý thuyết

1.1 Mô hình dữ liệu (Relational, Dimensional)

Việc lựa chọn mô hình dữ liệu là quyết định kiến trúc cơ bản nhất, ảnh hưởng trực tiếp đến hiệu suất truy vấn và khả năng phân tích của hệ thống. Trong lĩnh vực CSDL, hai mô hình chi phối hai mục đích sử dụng khác nhau là: Mô hình quan hệ (thường dùng cho OLTP) và mô hình chiều (dùng cho OLAP).

1.1.1 Mô hình dữ liệu quan hệ (Relational Model - OLTP)

a) Khái niệm

Mô hình dữ liệu quan hệ (Relational Model) là mô hình lưu trữ dữ liệu dưới dạng các bảng (tables), trong đó các bảng có mối quan hệ với nhau thông qua các khóa chính (primary keys) và khóa ngoại (foreign keys). Kiến trúc này là nền tảng của các hệ thống xử lý giao dịch trực tuyến (OLTP - Online Transaction Processing). Mục tiêu thiết kế cốt lõi của OLTP là tối ưu hóa cho các thao tác ghi dữ liệu (như INSERT, UPDATE, DELETE) và đảm bảo tính toàn vẹn, nhất quán của dữ liệu. Điều này đạt được thông qua quá trình chuẩn hóa (Normalization), nhằm loại bỏ sự trùng lặp dữ liệu. Ví dụ, trong một CSDL chuẩn hóa, tên của một đội bóng sẽ chỉ được lưu trữ một lần duy nhất trong bảng Teams và được tham chiếu bởi các bảng khác thông qua một Team_ID.

b) Đánh giá độ phù hợp

Mặc dù hiệu quả cho việc ghi dữ liệu (ví dụ: cập nhật tỷ số trận đấu theo thời gian thực), mô hình quan hệ chuẩn hóa cao lại bộc lộ nhiều nhược điểm nghiêm trọng khi dùng cho mục đích phân tích:

- **Hạn chế với hệ cơ sở dữ liệu phân tích:** Do dữ liệu bị chia nhỏ ra quá nhiều bảng để đạt chuẩn hóa, việc thực hiện các truy vấn phân tích tổng hợp (ví dụ: "Tính tổng số bàn thắng kỳ vọng (xG) của tiền đạo A trong tất cả các trận sân khách mùa giải 2023-2024") sẽ đòi hỏi kỹ thuật JOIN (kết nối) qua rất nhiều bảng trung gian. Điều này làm câu lệnh SQL trở nên phức tạp và giảm hiệu suất truy vấn khi lượng dữ liệu lớn.

Kết luận: Mô hình này không tối ưu cho tầng Data Warehouse phục vụ phân tích.

1.1.2 Mô hình Dữ liệu Chiều (Dimensional Model - OLAP)

a) Khái niệm

Mô hình dữ liệu chiều (Dimensional Modeling) là kỹ thuật thiết kế cơ sở dữ liệu được tối ưu hóa cho việc truy vấn và phân tích dữ liệu trong các hệ thống Kho dữ

liệu (Data Warehouse). Thay vì tập trung vào sự toàn vẹn mối quan hệ, mô hình này tổ chức dữ liệu thành hai loại bảng chính: Bảng sự kiện (Fact tables) chứa các số liệu đo lường và Bảng chiều (Dimension tables) chứa thông tin ngữ cảnh mô tả. Kiến trúc này là nền tảng của các hệ thống xử lý phân tích trực tuyến (OLAP - Online Analytical Processing). Mục tiêu thiết kế cốt lõi của OLAP là tối ưu hóa tốc độ cho các thao tác đọc dữ liệu (như SELECT, tổng hợp, báo cáo) nhằm trả lời nhanh chóng các câu hỏi nghiệp vụ phức tạp. Điều này thường đạt được thông qua kỹ thuật phi chuẩn hóa (Denormalization) — chấp nhận sự dư thừa dữ liệu có kiểm soát để giảm thiểu số lượng phép nối bảng (JOIN) phức tạp, từ đó tăng hiệu suất truy vấn.

b) Đánh giá độ phù hợp

Mô hình chiều chấp nhận sự dư thừa dữ liệu có kiểm soát để đổi lấy tốc độ truy xuất và sự đơn giản.

- **Ưu điểm với hệ cơ sở dữ liệu phân tích:** Các chỉ số như Goals, Assists, xG (trong bảng Fact) có thể dễ dàng được cắt lớp (slice and dice) theo mùa giải, đội bóng, hoặc cầu thủ (từ các bảng Dimension) để tạo ra các báo cáo nhanh chóng.

Kết luận: lựa chọn mô hình chiều (Dimensional Model) cho hệ cơ sở dữ liệu phân tích vì sự phù hợp vượt trội so với mô hình quan hệ (Relational Model).

1.2 Các lược đồ trong mô hình dữ liệu chiều (Dimensional Modeling Schemas)

1.2.1 Lược đồ hình sao (Star Schema)

Đây là đơn vị cơ bản nhất của mô hình chiều.

- **Cấu trúc:** Một bảng Fact duy nhất nằm ở trung tâm, kết nối trực tiếp với các bảng Dimension vệ tinh.
- **Đặc điểm:** Dữ liệu chiều được phi chuẩn hóa (denormalized) để tối ưu tốc độ truy vấn.
- **Hạn chế:** Chỉ phù hợp để phân tích một quy trình nghiệp vụ đơn lẻ.

1.2.2 Lược đồ bông tuyết (Snowflake Schema)

Snowflake Schema là biến thể của Star Schema, trong đó các bảng Dimension được chuẩn hoá và tách thành nhiều bảng con. Mô hình này giúp tiết kiệm không gian lưu trữ và giảm trùng lặp dữ liệu, nhưng làm giảm hiệu suất truy vấn vì cần thực hiện nhiều phép JOIN hơn.

1.2.3 Lược đồ Galaxy (Galaxy Schema)

Trong các bài toán thực tế phức tạp, doanh nghiệp thường cần phân tích nhiều quy trình nghiệp vụ khác nhau cùng một lúc. Khi đó, một bảng Fact đơn lẻ là không đủ, dẫn đến sự hình thành của lược đồ Galaxy (hay còn gọi là Fact Constellation Schema - Lược đồ chòm sao sự kiện).

- **Cấu trúc:** Bao gồm nhiều bảng Fact riêng biệt cùng tồn tại trong một mô hình.
- **Đặc điểm:** Các bảng Fact này chia sẻ các bảng Dimension dùng chung (gọi là Conformed Dimensions).

1.2.4 Lựa chọn lược đồ phù hợp

Đối với hệ thống phân tích dữ liệu bóng đá Ngoại hạng Anh, nhu cầu phân tích không chỉ dừng lại ở một khía cạnh đơn lẻ mà bao gồm nhiều quy trình nghiệp vụ song song như:

- Thống kê hiệu suất của Đội bóng (Team Stats).
- Thống kê hiệu suất chi tiết của Cầu thủ (Player Stats).
- Theo dõi cục diện Bảng xếp hạng (League Standings).

Các quy trình này tạo ra các luồng dữ liệu đo lường khác nhau do đó cần lưu trữ trong các bảng Fact riêng biệt. Tuy nhiên, chúng lại có mối liên hệ chặt chẽ thông qua các đối tượng dùng chung như Mùa giải, Đội bóng, Sân vận động. Chính vì vậy, nhóm quyết định lựa chọn Lược đồ Galaxy (Galaxy Schema) làm kiến trúc nền tảng.

CHƯƠNG 2. Phân tích và thiết kế hệ thống

2.1 Phân tích nguồn dữ liệu

Hệ thống ETL Football sử dụng hai nguồn dữ liệu chính cho giải Ngoại hạng Anh:

- Trang FBref (thông qua thư viện Python `soccerdata`)
- Trang Flashscore (thông qua kỹ thuật web scraping bằng Selenium)

Hai nguồn này không trùng lặp mà bổ sung cho nhau: FBref cung cấp lớp dữ liệu thống kê chi tiết về cầu thủ và đội bóng, Flashscore cung cấp thông tin bảng xếp hạng, điểm số và phong độ thực tế.

2.1.1 Tổng quan về đặc điểm dữ liệu

Từ góc độ kỹ thuật, cả hai nguồn dữ liệu đều có một số đặc điểm chung:

- Dữ liệu thay đổi theo thời gian (sau mỗi vòng đấu, mỗi mùa giải).
- Dữ liệu được tổ chức theo mùa giải (season), trận đấu (match), đội bóng (team) và cầu thủ (player).
- Mỗi nguồn có cách đặt tên, cấu trúc cột và kiểu dữ liệu khác nhau, dẫn tới nhu cầu chuẩn hoá trước khi đưa vào hệ quản trị CSDL.

2.1.2 Nguồn dữ liệu FBref (qua `soccerdata.FBref`)

a) Nội dung dữ liệu

FBref là một trang thống kê bóng đá chuyên sâu, cung cấp nhiều lớp dữ liệu chi tiết cho các giải đấu lớn. Trong hệ thống, dữ liệu được truy cập gián tiếp thông qua thư viện `soccerdata.FBref`, với bốn nhóm chính:

- **Thống kê cầu thủ theo mùa** – `read_player_season_stats`: Ví dụ: tổng bàn thắng, kiến tạo, số phút thi đấu, số lần ra sân, xG/xA trong cả mùa.
- **Thống kê cầu thủ theo trận** – `read_player_match_stats`: Thông tin chi tiết cho từng cầu thủ ở từng trận: bàn thắng, kiến tạo, số cú sút, chuyền, tắc bóng, thẻ phạt...
- **Thống kê đội theo trận** – `read_team_match_stats`: Kết quả từng trận ở mức đội bóng: bàn thắng, bàn thua, số cú sút, tỷ lệ kiểm soát bóng, sơ đồ chiến thuật,...
- **Thống kê đội theo mùa** – `read_team_season_stats`: Tổng hợp cho cả mùa: tổng bàn thắng, tổng bàn thua, tổng số trận, hiệu số, các chỉ số nâng cao.

Dữ liệu được trả về dưới dạng DataFrame của Pandas, một cấu trúc rất thuận tiện cho việc lọc, chuyển đổi và ghi ra file CSV trong bước Extract.

b) Ưu điểm

- **Có cấu trúc rõ ràng:** `soccerdata` đã chuẩn hoá lại các bảng thống kê, do đó không phải xử lý HTML thô hay tự tìm selector phức tạp.
- **Độ chi tiết cao:** Phục vụ rất tốt cho việc xây dựng các bảng fact: `fact_player_match` (mức độ cầu thủ–trận), `fact_team_match` (mức độ đội–trận).
- **Tương thích tốt với Python/ETL:** Dữ liệu DataFrame dễ dàng được chuyển sang CSV, merge nhiều mùa, join với các bảng khác.

c) Hạn chế và thách thức

- **Cột MultiIndex:** Một số bảng có header nhiều tầng (ví dụ: nhóm "Standard", "Shooting", "Passing"...). Điều này yêu cầu một bước "làm phẳng" (flatten) tên cột trước khi xử lý tiếp.
- **Thay đổi schema theo thời gian:** FBref có thể bổ sung/bớt cột, hoặc đổi tên một số thống kê. Hệ thống cần được thiết kế đủ linh hoạt để không "vỡ" khi có thay đổi nhỏ.
- **Trùng dữ liệu khi crawl nhiều lần:** Nếu mỗi lần chạy lại lấy toàn bộ lịch sử mùa giải, sẽ dẫn đến trùng bản ghi. Vì vậy, ở tầng Extract phải có cơ chế 'merge', dựa trên các trường khoá để loại trùng và chỉ giữ bản ghi mới hoặc cập nhật.

d) Vai trò trong hệ thống

Nguồn FBref được xem là xương sống dữ liệu thống kê của hệ thống:

- Cung cấp dữ liệu cho các bảng dimension: `dim_player`, `dim_team`, `dim_match`.
- Cung cấp dữ liệu chính cho các bảng fact: `fact_player_match`, `fact_team_match`.

Nhờ FBref, hệ thống có thể phân tích chi tiết hiệu suất của từng cầu thủ, từng đội bóng dưới nhiều góc độ khác nhau.

2.1.3 Nguồn dữ liệu Flashscore (qua Selenium)

a) Nội dung dữ liệu

Flashscore là trang chuyên về tỷ số trực tiếp và bảng xếp hạng giải đấu. Khác với FBref, Flashscore không cung cấp API chính thức, nên hệ thống phải sử dụng Selenium để:

- Mở trang web standings của giải Ngoại hạng Anh.
- Truy cập bảng xếp hạng theo 3 chế độ: Overall (tổng cộng), Home (sân nhà),

Away (sân khách).

- Từ mỗi bảng xếp hạng, hệ thống trích xuất các trường: Hạng (Rank), Tên đội (Team), Số trận (MP), Thắng (W), hoà (D), thua (L), Chuỗi GF:GA (bàn thắng : bàn thua), Hiệu số (GD), Điểm (Pts), Phong độ gần đây (Recent_Form – chuỗi W/D/L).

Các dữ liệu này được dùng để xây dựng bảng fact `fact_team_point`.

b) Ưu điểm

- **Phản ánh "kết quả cuối cùng" của đội bóng:** Khác với thống kê chi tiết, bảng xếp hạng cho biết đội đang đứng thứ mấy, được bao nhiêu điểm, phong độ gần đây ra sao.
- **Bổ sung góc nhìn standings mà FBref không nhấn mạnh:** Đặc biệt là phân tách theo sân nhà/sân khách và chuỗi phong độ (Recent_Form), hữu ích cho các phân tích thành tích.

c) Hạn chế và thách thức

- **Phụ thuộc vào giao diện website:** Nếu Flashscore thay đổi cấu trúc HTML, class CSS, hoặc layout bảng, các selector trong Selenium sẽ không còn đúng $\rightarrow$ script cần được cập nhật.
- **Định dạng GF:GA dạng chuỗi:** Cần bước xử lý tách chuỗi GF:GA thành hai cột số riêng (GF, GA) trong giai đoạn Transform.
- **Tên đội không đồng nhất với FBref:** Có thể tồn tại các dạng rút gọn như "Man Utd", "Spurs"... khác với cách ghi trên FBref. Điều này đòi hỏi một bước chuẩn hoá tên đội và map sang `team_id` chung.

d) Vai trò trong hệ thống

Nguồn Flashscore đóng vai trò bổ trợ cho FBref:

- Cung cấp thêm bảng fact `fact_team_point`: Điểm số, thứ hạng, phong độ theo từng mùa.
- Phân biệt giữa overall, home, away.
- Cho phép kết hợp standings với thống kê chi tiết: So sánh đội có xG cao nhưng điểm thấp, hay ngược lại; Phân tích sự khác biệt giữa phong độ sân nhà và sân khách.

2.1.4 So sánh và kết hợp hai nguồn dữ liệu

Việc sử dụng đồng thời FBref và Flashscore là một lựa chọn có chủ đích:

- FBref mạnh về "how they played" – các chỉ số chuyên môn bên trong trận đấu.

- Flashscore mạnh về "what they got" – điểm số, thứ hạng, phong độ tổng thể.

Khi kết hợp: Các bảng fact từ FBref (`fact_team_match`, `fact_player_match`) có thể được đối chiếu với `fact_team_point` từ Flashscore thông qua `team_id` (đã chuẩn hoá từ tên đội) và `season_id` hoặc `season`. Điều này mở ra nhiều loại phân tích:

- Đội thi đấu "đẹp mắt" (nhiều cơ hội, xG cao) nhưng hiệu quả điểm số chưa tương xứng.
- Đội có phong độ sân nhà vượt trội so với sân khách.
- Đánh giá tính ổn định dựa trên `Recent_Form` và hiệu số bàn thắng thua.

2.1.5 Rủi ro và chiến lược xử lý

Trong quá trình khai thác hai nguồn, hệ thống phải đối mặt với một số rủi ro:

- Rủi ro lệch/mất dữ liệu giữa hai nguồn: Có thể có trận đấu, vòng đấu hoặc mùa giải chỉ xuất hiện ở một trong hai nguồn.
- Rủi ro không map được tên đội/cầu thủ: Do khác cách viết, rút gọn, hoặc ký tự đặc biệt.
- Rủi ro thay đổi cấu trúc (schema/layout) từ phía nhà cung cấp dữ liệu.

Chiến lược xử lý:

- Thiết kế một tầng Transform riêng để: Chuẩn hoá tên đội, tên cầu thủ trước khi gán ID; Ghi log các trường hợp không map được để xử lý thủ công sau.
- Giữ staging data (các file trong `data_raw/` và `data_processed/`) để: Dễ dàng kiểm tra lại dữ liệu khi có sự cố; Có thể chạy lại từng bước ETL mà không cần crawl lại dữ liệu lịch sử.

Nhờ phân tích rõ đặc điểm từng nguồn dữ liệu và thiết kế chiến lược xử lý phù hợp, hệ thống có nền tảng vững chắc để triển khai pipeline ETL và mô hình cơ sở dữ liệu ở các phần tiếp theo.

2.2 Thiết kế pipeline ETL (kiến trúc luồng xử lý)

Trong hệ thống ETL Football, pipeline ETL được thiết kế theo hướng phân tầng rõ ràng, tách biệt giữa:

- Nguồn dữ liệu bên ngoài (External Sources)
- Tầng lưu trữ trung gian file-based (Staging)
- Tầng xử lý/transformation (ETL Scripts)
- Tầng kho dữ liệu quan hệ (Data Warehouse – PostgreSQL)

- Tầng phục vụ báo cáo/phân tích (Serving/BI)

Mục tiêu của thiết kế này là đảm bảo:

- Dễ bảo trì, dễ mở rộng: có thể thay đổi từng tầng mà không phá toàn bộ hệ thống.
- Có nhật ký rõ ràng theo từng bước: có file trung gian và log ở từng lớp để kiểm tra khi có lỗi.
- Hỗ trợ chạy lại (re-run) và incremental: hạn chế tải lại toàn bộ lịch sử, tiết kiệm thời gian và tài nguyên.

2.2.1 Các lớp trong kiến trúc pipeline

1. **Lớp nguồn dữ liệu (Source Layer):** Hệ thống lấy dữ liệu từ hai nguồn chính:
   - **FBref** (thông qua thư viện `soccerdata.FBref`): cung cấp dữ liệu thống kê chi tiết cầu thủ, đội bóng, trận đấu.
   - **Flashscore** (thông qua Selenium): cung cấp dữ liệu bảng xếp hạng, điểm số và phong độ gần đây.
2. **Lớp staging file-based (Staging Layer):** Lớp này sử dụng hai thư mục:
   - `data_raw/`: lưu dữ liệu thô sau khi extract từ FBref và Flashscore.
   - `data_processed/`: lưu dữ liệu đã xử lý/chuẩn hoá, tương ứng với từng bảng dim/fact.
3. **Lớp ETL Scripts (Logic xử lý):** Lớp này được cài đặt bằng các file Python trong thư mục `scr/`: `Extract.py`, `Transform.py`, `Load.py`. Mỗi script đảm nhiệm một nhiệm vụ rõ ràng.
4. **Lớp Data Warehouse – PostgreSQL:** Dữ liệu được đưa vào PostgreSQL với thiết kế galaxy schema:
   - Các bảng dimension: `dim_player`, `dim_team`, `dim_match`, `dim_stadium`, `dim_season`.
   - Các bảng fact: `fact_team_match`, `fact_player_match`, `fact_team_point`
5. **Lớp Orchestration – Apache Airflow:** Thay vì chạy ETL bằng tay, hệ thống sử dụng Apache Airflow để gom ba bước Extract → Transform → Load thành một DAG, chạy theo lịch định kỳ.

2.2.2 Nguyên tắc thiết kế pipeline ETL trong hệ thống

Khi xây dựng pipeline, hệ thống áp dụng một số nguyên tắc:

- **Batch + Incremental:** Pipeline được chạy theo lô và sử dụng file `.last_extract_date`

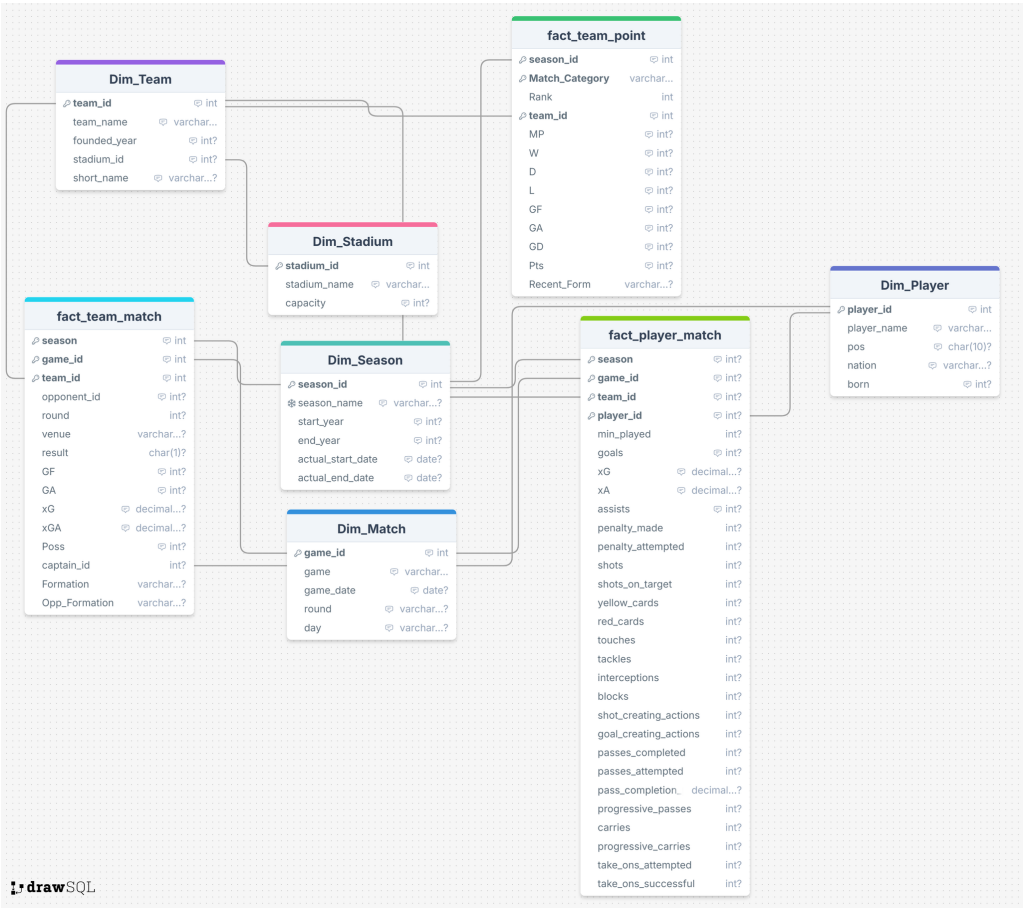
để chỉ xử lý phần dữ liệu mới.

- **Idempotent (chạy lại không phá dữ liệu):** Bước Load sử dụng `INSERT ... ON CONFLICT DO UPDATE/DO NOTHING` để không tạo record trùng.
- **Tách bạch trách nhiệm (Separation of Concerns):** Mỗi script (Extract, Transform, Load) có một nhiệm vụ duy nhất.
- **Dễ quan sát & debug:** Mỗi tầng đều có artifact (file CSV, log). Airflow lưu log từng task, dễ truy vết lỗi.

Nhờ vậy, pipeline ETL không chỉ chạy được "một lần cho xong" mà còn chịu tải được việc vận hành dài hạn, dễ bảo trì và nâng cấp.

2.3 Thiết kế cơ sở dữ liệu

Hệ thống được thiết kế theo mô hình Galaxy Schema. Kiến trúc này cho phép tích hợp dữ liệu từ ba quy trình nghiệp vụ khác nhau (Thống kê đội bóng, Thống kê cầu thủ, Bảng xếp hạng) vào cùng một Kho dữ liệu thống nhất.



Hình 2.1: Biểu đồ ma trận tấn công – phòng ngự.

2.3.1 Thiết kế các bảng chiều (Dimensions)

Các bảng chiều lưu trữ thông tin mô tả, ít thay đổi theo thời gian và được chuẩn hóa.

a) Bảng Dim_Season

Bảng này lưu trữ danh mục các mùa giải bóng đá Ngoại hạng Anh.

Bảng 2.1: Thiết kế bảng Dim_Season

Tên cột	Kiểu dữ liệu	Mô tả
season_id	INT	(PK) Mã định danh mùa giải
season_name	VARCHAR(15)	Tên mùa giải (Unique), VD: '2023-2024'
start_year	INT	Năm bắt đầu
end_year	INT	Năm kết thúc
actual_start_date	DATE	Ngày bắt đầu thực tế
actual_end_date	DATE	Ngày kết thúc thực tế

b) Bảng Dim_Stadium

Bảng chứa thông tin địa lý và cơ sở vật chất của các sân vận động.

Bảng 2.2: Thiết kế bảng Dim_Stadium

Tên cột	Kiểu dữ liệu	Mô tả
stadium_id	INT	(PK) Mã định danh sân vận động
stadium_name	VARCHAR(255)	Tên sân vận động
capacity	INT	Sức chứa

c) Bảng Dim_Team

Bảng chứa thông tin hồ sơ cố định của các câu lạc bộ.

Bảng 2.3: Thiết kế bảng Dim_Team

Tên cột	Kiểu dữ liệu	Mô tả
team_id	INT	(PK) Mã định danh đội bóng
team_name	VARCHAR(255)	Tên đầy đủ của đội bóng
founded_year	INT	Năm thành lập
stadium_id	INT	(FK) Mã sân nhà (liên kết với Dim_Stadium)
short_name	VARCHAR(20)	Tên viết tắt của đội

d) Bảng Dim_Player

Bảng lưu trữ hồ sơ cá nhân của cầu thủ.

Bảng 2.4: Thiết kế bảng Dim_Player

Tên cột	Kiểu dữ liệu	Mô tả
player_id	INT	(PK) Mã định danh cầu thủ
player_name	VARCHAR(255)	Tên cầu thủ
pos	CHAR(10)	Vị trí thi đấu
nation	VARCHAR(20)	Quốc tịch
born	INT	Năm sinh

e) Bảng Dim_Match

Bảng lưu trữ lịch thi đấu và metadata về trận đấu.

Bảng 2.5: Thiết kế bảng Dim_Match

Tên cột	Kiểu dữ liệu	Mô tả
game_id	INT	(PK) Mã định danh trận đấu
game	VARCHAR(255)	Tên trận đấu (VD: Man City vs Arsenal)
game_date	DATE	Ngày thi đấu
round	VARCHAR(50)	Vòng đấu
day	VARCHAR(10)	Thứ trong tuần

2.3.2 Thiết kế chi tiết các bảng sự kiện (Fact Tables)

a) Bảng Fact_Team_Match

Ghi lại các chỉ số hiệu suất của từng đội bóng trong mỗi trận đấu.

Bảng 2.6: Thiết kế bảng Fact_Team_Match

Tên cột	Kiểu dữ liệu	Mô tả
season	INT	(PK, FK) Mã mùa giải
game_id	INT	(PK, FK) Mã trận đấu
team_id	INT	(PK, FK) Mã đội bóng
opponent_id	INT	(FK) Mã đội đối thủ
round	INT	Vòng đấu (dạng số)
venue	VARCHAR(10)	Sân nhà/Sân khách (Home/Away)
result	CHAR(1)	Kết quả (W/D/L)
GF	INT	Bàn thắng ghi được (Goals For)
GA	INT	Bàn thua (Goals Against)
xG	NUMERIC(4,2)	Bàn thắng kỳ vọng
xGA	NUMERIC(4,2)	Bàn thua kỳ vọng
Poss	INT	Tỷ lệ kiểm soát bóng
captain_id	INT	(FK) Đội trưởng (liên kết Dim_Player)
Formation	VARCHAR(20)	Đội hình ra sân
Opp_Formation	VARCHAR(20)	Đội hình đối thủ

b) Bảng Fact_Player_Match

Chứa dữ liệu chi tiết nhất, ghi nhận đóng góp của từng cầu thủ trong từng trận đấu.

Bảng 2.7: Thiết kế bảng Fact_Player_Match

Tên cột	Kiểu dữ liệu	Mô tả chi tiết
season	INT	(FK) Mã mùa giải
game_id	INT	(FK) Mã trận đấu
team_id	INT	(FK) Mã đội bóng cầu thủ khoác áo
player_id	INT	(FK) Mã định danh cầu thủ
min_played	INT	Số phút thi đấu thực tế trong trận
goals	INT	Số bàn thắng ghi được
xG	NUMERIC(4,2)	Bàn thắng kỳ vọng (Expected Goals)
xA	NUMERIC(4,2)	Kiến tạo kỳ vọng (Expected Assists)
assists	INT	Số pha kiến tạo thành bàn
shots	INT	Tổng số cú sút đã tung ra
yellow_cards	INT	Số thẻ vàng phải nhận
red_cards	INT	Số thẻ đỏ phải nhận
...	...	...

c) Bảng Fact_Team_Point

Chứa thành tích của các đội theo sân nhà, sân khách trong các mùa.

Bảng 2.8: Thiết kế bảng Fact_Team_Point

Tên cột	Kiểu dữ liệu	Mô tả
season_id	INT	(PK, FK) Mã mùa giải
team_id	INT	(PK, FK) Mã đội bóng
Match_Category	VARCHAR(100)	(PK) Loại bảng xếp hạng (Overall/Home/Away)
Rank	INT	Thứ hạng
MP	INT	Số trận đã đấu
W / D / L	INT	Thắng / Hòa / Thua
GF / GA / GD	INT	Bàn thắng / Bàn thua / Hiệu số
Pts	INT	Điểm số
Recent_Form	VARCHAR(50)	Phong độ gần đây (VD: W-W-D-L-W)

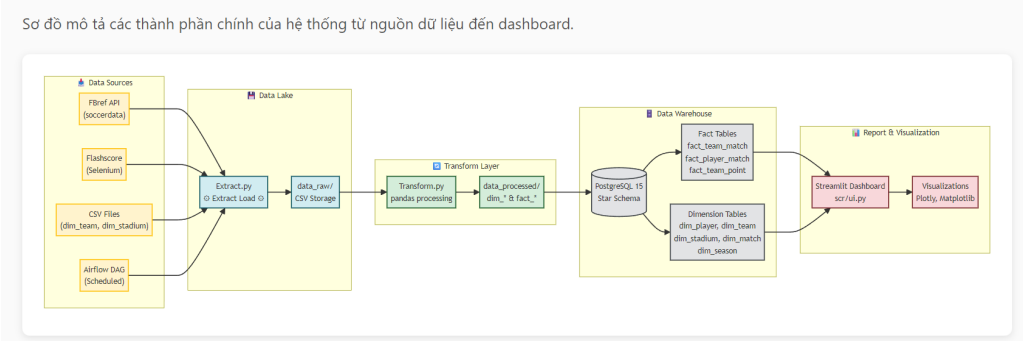
2.3.3 Giải thích logic thiết kế và Mối quan hệ

Việc lựa chọn mô hình Galaxy với 3 bảng Fact thay vì gộp chung vào một bảng duy nhất dựa trên các nguyên lý logic sau:

- **Khác biệt về Độ mịn (Granularity Mismatch):** Không thể lưu trữ thông tin Cầu thủ và thông tin Bảng xếp hạng trong cùng một bảng.
- **Sử dụng lại các Chiều dùng chung (Conformed Dimensions):** Dữ liệu liên kết chặt chẽ nhờ chia sẻ các bảng Dimension `Dim_Team` và `Dim_Season`.
- **Hiệu suất truy vấn (Query Performance):** Việc phân tách này giúp tối ưu hóa tài nguyên hệ thống và tăng tốc độ phản hồi báo cáo.

2.4 Thiết kế hệ thống

Thiết kế sơ đồ kiến trúc chi tiết hệ thống.



Hình 2.2: kiến trúc hệ thống.

CHƯƠNG 3. Triển khai hệ thống và đánh giá kết quả

3.1 Thiết lập môi trường

Để hệ thống ETL Football hoạt động ổn định và dễ bảo trì, trước hết cần thiết lập một môi trường làm việc thống nhất, bao gồm: phiên bản Python, các thư viện phụ thuộc, hệ quản trị cơ sở dữ liệu PostgreSQL, cùng với một số biến môi trường và file cấu hình.

3.1.1 Yêu cầu hệ thống và dịch vụ nền

Hệ thống ETL Football được xây dựng trên ngôn ngữ Python, sử dụng các thư viện như `pandas`, `soccerdata`, `psycopg2`, `Selenium`,... Do đó, máy cài đặt cần đáp ứng một số yêu cầu tối thiểu:

- **Hệ điều hành:** Windows 10/11, macOS hoặc một bản phân phối Linux.
- **Python:** Phiên bản khuyến nghị là Python 3.10 trở lên.
- **Trình duyệt Chrome/Chromium:** Được sử dụng cùng với Selenium.
- **Hệ quản trị cơ sở dữ liệu:** PostgreSQL 15.
- **Tự động hoá và điều phối pipeline:** Docker + Docker Compose để dựng các container cho PostgreSQL và Apache Airflow.

3.1.2 Tạo môi trường Python và cài đặt thư viện

1. Tạo môi trường ảo:

   ```
   python -m venv .venv
   ```

2. Kích hoạt môi trường ảo (trên Windows):

   ```
   .venv\Scripts\activate
   ```

3. Cài đặt các thư viện phụ thuộc:

   ```
   pip install -r requirements.txt
   ```

3.1.3 Cấu hình biến môi trường cho dự án

Hệ thống sử dụng một số biến môi trường quan trọng.

Bảng 3.1: Các biến môi trường

Biến môi trường	Vai trò	Ví dụ giá trị
ETL_FOOTBALL_BASE_DIR	Trỏ tới thư mục gốc dự án	D:\ETL_Football
POSTGRES_HOST	Địa chỉ host của PostgreSQL	localhost
POSTGRES_PORT	Cổng PostgreSQL	5432
POSTGRES_DB	Tên database	ETL_FOOTBALL
POSTGRES_USER	Tên người dùng	airflow
POSTGRES_PASSWORD	Mật khẩu tương ứng	airflow

3.1.4 Thiết lập file cấu hình kết nối PostgreSQL

Hệ thống sử dụng file `scr/database.ini` để lưu thông tin kết nối.

```
[postgresql]
host=localhost
port=5432
database=ETL_FOOTBALL
user=airflow
password=airflow
```

Listing 3.1: Nội dung file database.ini

3.1.5 Thiết lập dịch vụ PostgreSQL và Airflow/Docker

Để tự động hóa pipeline, hệ thống sử dụng Docker để chạy PostgreSQL và Apache Airflow. Các thư mục dự án được mount vào container Airflow:

- `./dags` → `/opt/airflow/dags`
- `./scr` → `/opt/airflow/scr`
- `./data_raw` → `/opt/airflow/data_raw`
- `./data_processed` → `/opt/airflow/data_processed`

Các lệnh khởi động cơ bản:

```
# Bật PostgreSQL
docker compose up -d postgres

# Khởi tạo Airflow (chạy một lần)
docker compose up airflow-init

# Bật Airflow
docker compose up -d airflow-webserver airflow-scheduler
```

Sau đó, truy cập `http://localhost:8080` để quản lý DAG.

3.2 Thu thập, xử lý và chuẩn hóa dữ liệu

Quy trình được thực hiện qua hai script Python: `scr/Extract.py` và `scr/Transform.py`.

3.2.1 Thu thập dữ liệu (Extract)

a) Thu thập dữ liệu thống kê từ FBref

Sử dụng thư viện `soccerdata.Fbref` để lấy các bộ dữ liệu. `Extract.py` đọc file `data_raw/.last_extract_date.txt` để xác định dữ liệu mới cần tải, tránh tải lại toàn bộ lịch sử. Dữ liệu mới được gộp với dữ liệu cũ và loại bỏ trùng lặp.

b) Thu thập bảng xếp hạng từ Flashscore

Sử dụng Selenium headless để truy cập trang bảng xếp hạng, trích xuất dữ liệu ở ba chế độ: chung, sân nhà, sân khách. Dữ liệu được lưu vào `data_raw/team_point.csv`.

3.2.2 Xử lý và chuẩn hóa dữ liệu (Transform)

Bước này được thực hiện trong `scr/Transform.py`, kết quả lưu tại `data_processed/`. Mục tiêu là làm sạch, chuẩn hoá và xây dựng các bảng dimension và fact.

a) Các hàm trợ giúp dùng chung

- `flatten_dataframe_columns()`: "Làm phẳng" các DataFrame có MultiIndex.
- `save_table(df, name)`: Ghi DataFrame ra file CSV và in log.

b) Xây dựng các bảng Dimension

- **dim_player:** Kết hợp dữ liệu, loại bỏ trùng, chuẩn hoá trường và gán `player_id`.
- **dim_team:** Chuẩn hoá tên đội, tạo `team_id`, `short_name`.
- **dim_stadium:** Tách riêng thông tin sân, gán `stadium_id`.
- **dim_match:** Loại bỏ trùng, sinh `game_id`, chuẩn hoá ngày, vòng đấu.
- **dim_season:** Chuẩn hoá thông tin mùa giải, gán `season_id`.

c) Xây dựng các bảng Fact

- **fact_team_match:** Ánh xạ tên đội, đối thủ, trận đấu sang các ID tương ứng; giữ lại các chỉ số quan trọng.
- **fact_player_match:** Ánh xạ tên cầu thủ, đội, trận sang ID; giữ lại các chỉ số cá nhân.
- **fact_team_point:** Xây dựng từ dữ liệu Flashscore, chuẩn hoá mùa, tách chuỗi GF:GA, ánh xạ tên đội sang `team_id`.

3.3 Ingestion dữ liệu với Apache Airflow

Apache Airflow được sử dụng như lớp điều phối trung tâm.

3.3.1 Cấu trúc DAG `football_etl_pipeline`

DAG được định nghĩa trong `dags/football_etl_dag.py` và được cấu hình chạy định kỳ (ví dụ, 02:00 sáng Thứ Tư hằng tuần). DAG gồm 3 task chính với quan hệ phụ thuộc: `extract_data` → `transform_data` → `load_data`.

3.3.2 Hoạt động chi tiết của các task

1. **Task `extract_data`:** Gọi logic trong `scr/Extract.py`. Thu thập dữ liệu từ FBref và Flashscore, xử lý incremental, và lưu vào thư mục `data_raw/`.
2. **Task `transform_data`:** Gọi các hàm trong `scr/Transform.py`. Đọc dữ liệu từ `data_raw/`, xây dựng các bảng dimension và fact, rồi lưu kết quả vào `data_processed/`.
3. **Task `load_data`:** Thực thi logic trong `scr/Load.py`. Kết nối PostgreSQL, tạo bảng (nếu chưa có), và nạp dữ liệu từ các file CSV trong `data_processed/` bằng câu lệnh `INSERT ... ON CONFLICT`.

3.4 Lưu trữ dữ liệu có cấu trúc trong PostgreSQL

PostgreSQL là đích đến cuối của pipeline và là nguồn dữ liệu cho các tác vụ phân tích.

3.4.1 Vai trò của PostgreSQL trong hệ thống

- Lưu trữ dữ liệu theo mô hình galaxy schema.
- Đảm bảo toàn vẹn dữ liệu (khoá chính, khoá ngoại).
- Tối ưu cho các truy vấn phân tích.
- Nền tảng để kết nối với các công cụ BI.

3.4.2 Mô hình dữ liệu galaxy schema trong PostgreSQL

Dữ liệu được tổ chức thành các bảng dimension (`dim_player`, `dim_team`, ...) và các bảng fact (`fact_team_match`, `fact_player_match`, `fact_team_point`) như đã thiết kế ở phần trước.

3.4.3 Quy trình nạp dữ liệu vào PostgreSQL

Quy trình được thực hiện bởi `scr/Load.py`:

1. **Kết nối CSDL:** Đọc cấu hình từ `database.ini` và kết nối bằng `psycopg2`.
2. **Tạo và nạp bảng dimension:** Sử dụng `CREATE TABLE IF NOT EXISTS`

và `INSERT ... ON CONFLICT DO UPDATE` để chèn và cập nhật dữ liệu.

3. **Tạo và nạp bảng fact:** Sử dụng `CREATE TABLE IF NOT EXISTS` và `INSERT ... ON CONFLICT DO NOTHING` để chèn dữ liệu mới mà không tạo bản ghi trùng.

3.5 Truy vấn và phân tích dữ liệu

Trong hệ thống phân tích dữ liệu Premier League, các truy vấn SQL giữ vai trò trung tâm, đảm nhiệm việc trích xuất, tổng hợp và xử lý dữ liệu từ mô hình kho dữ liệu dạng sao (galaxy schema). Các bảng chiều như `dim_team`, `dim_player`, `dim_season` và các bảng sự kiện như `fact_team_match`, `fact_team_point`, `fact_player_match` được kết hợp thông qua các câu lệnh JOIN và được phân tích bằng các hàm tổng hợp (SUM, COUNT, AVG, ROUND).

3.5.1 Truy vấn thông tin mùa giải và bảng xếp hạng

```sql
SELECT
    ftp.Rank,
    dt.team_name ,
    ftp.mp ,
    ftp.w ,
    ftp.d ,
    ftp.l ,
    ftp.gf ,
    ftp.ga ,
    ftp.gd ,
    ftp.pts
FROM fact_team_point ftp
JOIN dim_team dt ON ftp.team_id = dt.team_id
JOIN dim_season ds ON ftp.season_id = ds.season_id
WHERE ds.season_name = %s
  AND ftp.Match_Category = 'overall'
ORDER BY ftp.Rank;
```

Listing 3.2: Truy vấn lấy bảng xếp hạng

3.5.2 Truy vấn thông tin cầu thủ

```sql
SELECT
    dp.player_name,
    dt.team_name,
    SUM(fpm.goals) as total_goals
FROM fact_player_match fpm
JOIN dim_player dp ON fpm.player_id = dp.player_id
JOIN dim_team dt ON fpm.team_id = dt.team_id
JOIN dim_season ds ON fpm.season = ds.season_id
WHERE ds.season_name = %s
GROUP BY dp.player_name, dt.team_name
HAVING SUM(fpm.goals) > 0
ORDER BY total_goals DESC
LIMIT %s;
```

Listing 3.3: Truy vấn cầu thủ ghi bàn nhiều nhất

```sql
SELECT
    dp.player_name,
    dt.team_name,
    SUM(fpm.assists) as total_assists
FROM fact_player_match fpm
JOIN dim_player dp ON fpm.player_id = dp.player_id
JOIN dim_team dt ON fpm.team_id = dt.team_id
JOIN dim_season ds ON fpm.season = ds.season_id
WHERE ds.season_name = %s
GROUP BY dp.player_name, dt.team_name
HAVING SUM(fpm.assists) > 0
ORDER BY total_assists DESC
LIMIT %s;
```

Listing 3.4: Truy vấn cầu thủ kiến tạo nhiều nhất

3.5.3 Thống kê tổng quan mùa giải

```sql
SELECT
    COALESCE(COUNT(DISTINCT ftm.game_id), 0) as total_matches,
    COALESCE(SUM(ftm.GF), 0) as total_goals
FROM fact_team_match ftm
JOIN dim_season ds ON ftm.season = ds.season_id
WHERE ds.season_name = %s;
```

Listing 3.5: Truy vấn thống kê tổng quan mùa giải

3.5.4 Phân tích hiệu suất đội bóng

```sql
SELECT *
FROM (
    SELECT
        dt.team_name,
        SUM(CASE WHEN LOWER(ftp.match_category) = 'home' THEN ftp.pts ELSE 0 END) as home_pts,
        SUM(CASE WHEN LOWER(ftp.match_category) = 'away' THEN ftp.pts ELSE 0 END) as away_pts,
        SUM(CASE WHEN LOWER(ftp.match_category) = 'home' THEN ftp.w ELSE 0 END) as home_wins,
        SUM(CASE WHEN LOWER(ftp.match_category) = 'away' THEN ftp.w ELSE 0 END) as away_wins
    FROM fact_team_point ftp
    JOIN dim_team dt ON ftp.team_id = dt.team_id
    JOIN dim_season ds ON ftp.season_id = ds.season_id
    WHERE ds.season_name = %s AND LOWER(ftp.match_category) IN ('home', 'away')
    GROUP BY dt.team_name
) AS performance_summary
ORDER BY (performance_summary.home_pts + performance_summary.away_pts) DESC;
```

Listing 3.6: Truy vấn hiệu suất sân nhà - sân khách

```sql
SELECT
    dt.team_name,
    ftp.ga as goals_conceded,
    ftp.mp as matches_played,
    ROUND(CAST(ftp.ga AS DECIMAL) / NULLIF(ftp.mp, 0), 2) as avg_goals_conceded
FROM fact_team_point ftp
JOIN dim_team dt ON ftp.team_id = dt.team_id
JOIN dim_season ds ON ftp.season_id = ds.season_id
WHERE ds.season_name = %s AND LOWER(ftp.match_category) = 'overall'
ORDER BY avg_goals_conceded ASC;
```

Listing 3.7: Truy vấn hiệu suất phòng ngự

```sql
SELECT
    dt.team_name,
    ftp.gf as goals_scored,
    ftp.mp as matches_played,
    ROUND(CAST(ftp.gf AS DECIMAL) / NULLIF(ftp.mp, 0), 2) as
   avg_goals_scored
FROM fact_team_point ftp
JOIN dim_team dt ON ftp.team_id = dt.team_id
JOIN dim_season ds ON ftp.season_id = ds.season_id
WHERE ds.season_name = %s AND LOWER(ftp.match_category) = '
   overall'
ORDER BY avg_goals_scored DESC;
```

Listing 3.8: Truy vấn hiệu suất tấn công

```sql
SELECT
    dm.game_date,
    o_dt.team_name as opponent_name,
    ftm.venue,
    ftm.result,
    ftm.gf as goals_for,
    ftm.ga as goals_against
FROM fact_team_match ftm
JOIN dim_team dt ON ftm.team_id = dt.team_id
JOIN dim_team o_dt ON ftm.opponent_id = o_dt.team_id
JOIN dim_season ds ON ftm.season = ds.season_id
JOIN dim_match dm ON ftm.game_id = dm.game_id
WHERE ds.season_name = %s AND dt.team_name = %s
ORDER BY dm.game_date DESC
LIMIT %s;
```

Listing 3.9: Truy vấn 5 trận gần nhất

3.5.5 So sánh theo mùa giải

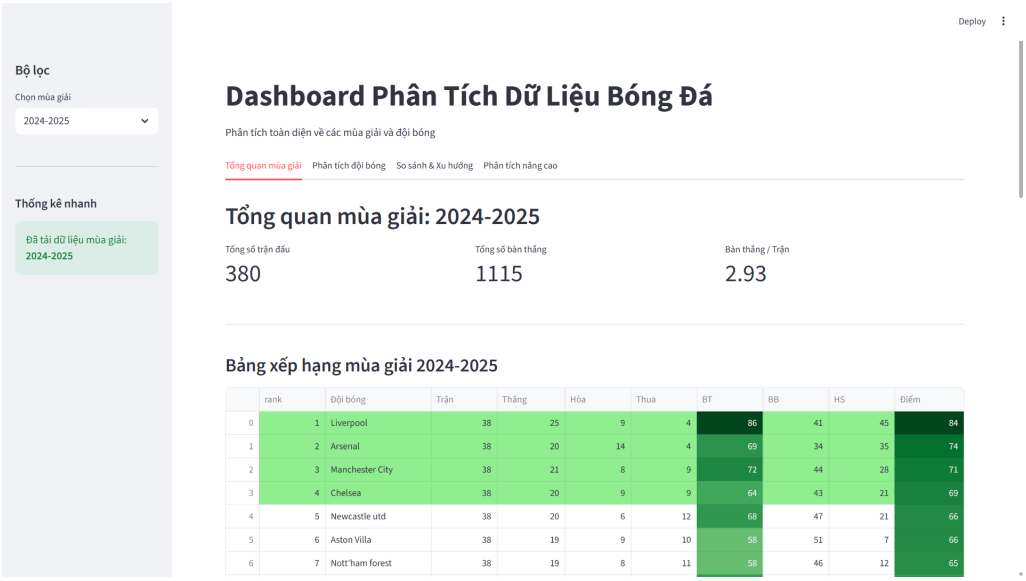
Việc so sánh giữa các mùa giải được thực hiện bằng cách tổng hợp dữ liệu theo từng mùa, bao gồm tổng số trận, tổng bàn thắng và tỷ lệ bàn thắng/trận. Đây là cơ sở cho các biểu đồ xu hướng trong phần trực quan hóa.

3.6 Trực quan hóa dữ liệu với Streamlit

Trực quan hóa là phần quan trọng trong hệ thống, giúp chuyển đổi kết quả phân tích dữ liệu thành các biểu đồ, bảng và chỉ số trực quan, hỗ trợ người dùng dễ dàng tiếp cận và đánh giá thông tin. Ứng dụng sử dụng thư viện Streamlit kết hợp với Plotly Express và Plotly Graph Objects để tạo ra các thành phần tương tác.

3.6.1 Bảng xếp hạng và các chỉ số tổng quan

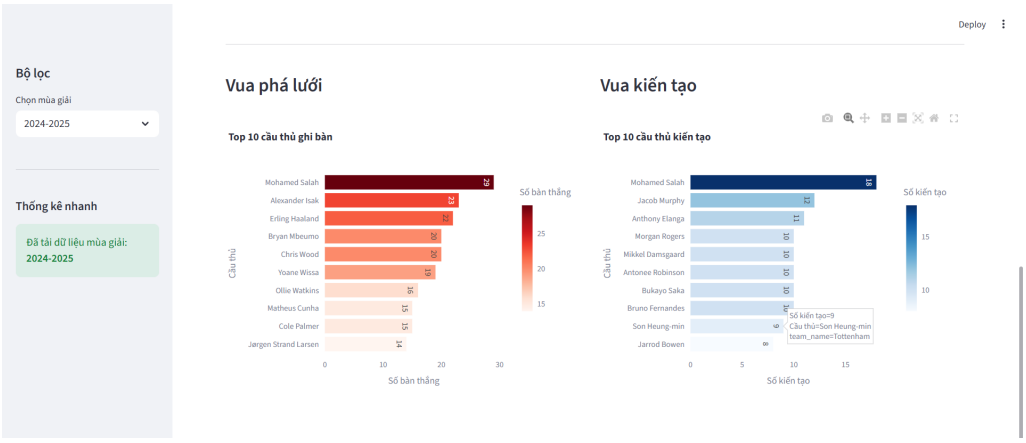
Bảng xếp hạng giải đấu được hiển thị bằng `st.dataframe`, cho phép làm nổi bật các đội dẫn đầu và các đội nhóm cuối bằng hệ thống màu sắc. Các chỉ số tổng quan (KPI) như tổng trận, tổng bàn thắng, bàn thắng trung bình/trận được thể hiện bằng `st.metric`, giúp tạo cái nhìn nhanh về quy mô và chất lượng mùa giải.



Hình 3.1: Bảng xếp hạng và các chỉ số tổng quan.

3.6.2 Biểu đồ phân tích cầu thủ

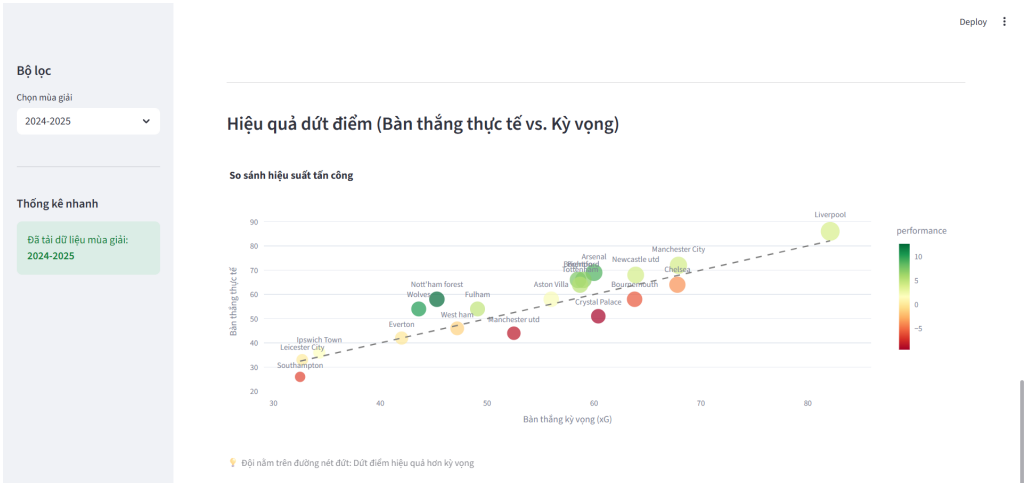
Hai biểu đồ quan trọng là Top cầu thủ ghi bàn và Top cầu thủ kiến tạo. Các biểu đồ dạng thanh ngang được dùng để tối ưu hóa không gian và nhấn mạnh sự so sánh giữa các giá trị.



Hình 3.2: Biểu đồ phân tích cầu thủ.

3.6.3 Biểu đồ xG so với bàn thắng thực tế

Biểu đồ scatter thể hiện mối tương quan giữa bàn thắng kỳ vọng (xG) và bàn thắng thực tế. Đường chéo y = x được thêm vào nhằm đánh giá hiệu suất dứt điểm.



Hình 3.3: Biểu đồ xG so với bàn thắng thực tế.

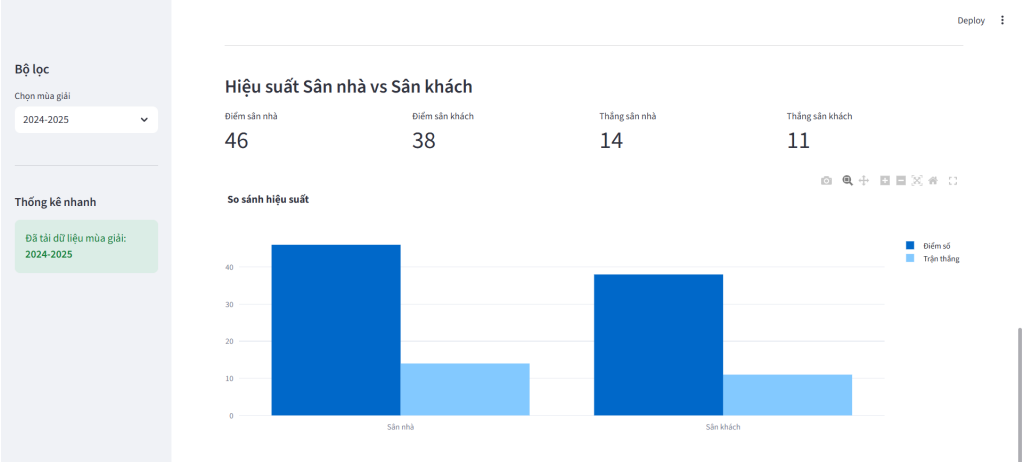
3.6.4 Phân tích phong độ gần nhất

Phong độ 5 trận gần nhất được biểu diễn bằng biểu đồ thanh theo thời gian, với màu sắc tương ứng với kết quả (thắng, hòa, thua).

Hình 3.4: Phân tích phong độ gần nhất.

3.6.5 Hiệu suất sân nhà – sân khách

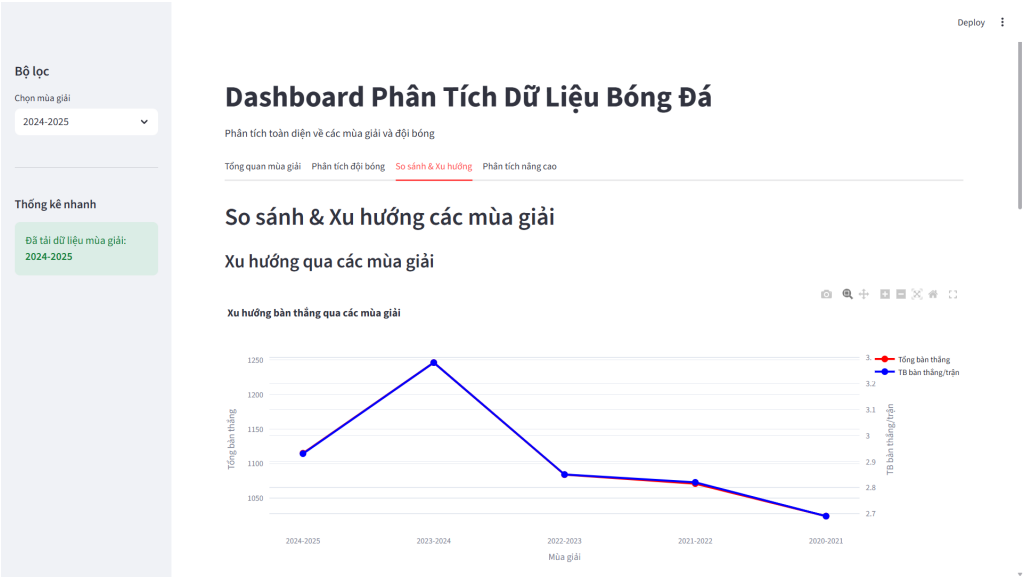
Dữ liệu sân nhà và sân khách được trực quan hóa theo hai dạng: biểu đồ thanh so sánh điểm số và số trận thắng; biểu đồ scatter đối chiếu tổng điểm sân nhà và sân khách. Các biểu đồ này hỗ trợ đánh giá mức độ phụ thuộc sân nhà hoặc khả năng thi đấu xa nhà của các đội.



Hình 3.5: Phân tích hiệu suất sân nhà – sân khách.

3.6.6 Xu hướng bàn thắng qua nhiều mùa giải

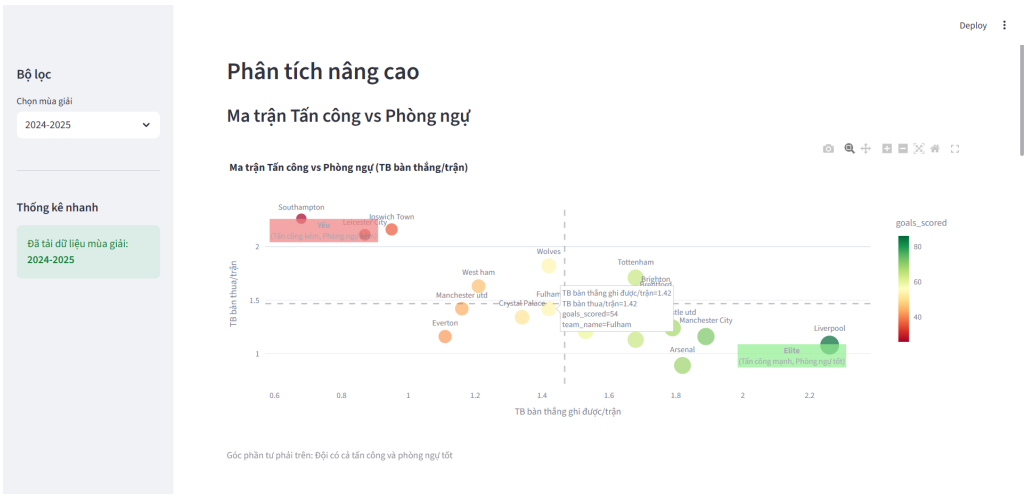
Biểu đồ đường thể hiện tổng bàn thắng và số bàn thắng trung bình theo mùa. Biểu đồ hai trục giúp so sánh hai biến động cùng lúc, hỗ trợ phân tích xu hướng tấn công theo thời gian.



Hình 3.6: Xu hướng bàn thắng qua nhiều mùa giải.

3.6.7 Ma trận tấn công – phòng ngự

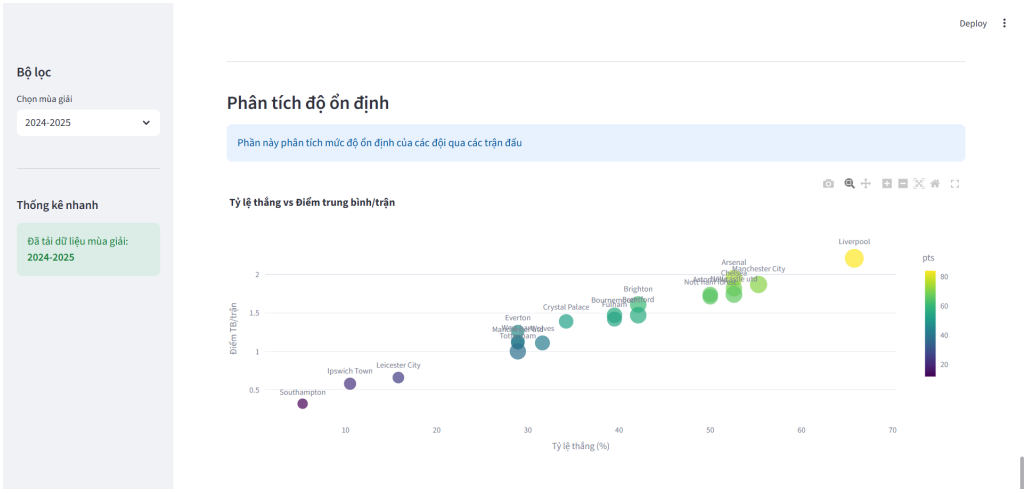
Biểu đồ scatter phân chia thành bốn vùng dựa trên ngưỡng trung bình: Công mạnh – Thủ tốt, Công mạnh – Thủ yếu, Công yếu – Thủ tốt, Công yếu – Thủ yếu. Việc phân vùng này giúp đánh giá toàn diện đặc điểm thi đấu của từng đội bóng trong mùa giải.



Hình 3.7: Biểu đồ ma trận tấn công – phòng ngự.

3.6.8 Biểu đồ đánh giá độ ổn định

Độ ổn định được thể hiện thông qua tỷ lệ thắng và điểm trung bình mỗi trận, được biểu diễn bằng biểu đồ scatter giúp xác định đội bóng có chiến lược lâu dài ổn định hoặc các đội thi đấu thất thường theo từng trận.



Hình 3.8: Biểu đồ đánh giá độ ổn định.

Phần III

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

CHƯƠNG 1. Kết luận chung

Qua quá trình nghiên cứu và thực hiện đề tài "Xây dựng hệ cơ sở dữ liệu phân tích kết quả giải bóng đá Ngoại hạng Anh", Nhóm 9 đã hoàn thành các mục tiêu đề ra ban đầu:

- **Xây dựng thành công quy trình ETL tự động:** Hệ thống đã có khả năng tự động trích xuất dữ liệu từ các nguồn API phức tạp (FBref, Understat), làm sạch và chuẩn hóa dữ liệu đa cấp (multi-level headers) để nạp vào kho dữ liệu.
- **Thiết kế Kho dữ liệu tối ưu:** Mô hình lược đồ ngân hà (Galaxy Schema) với 7 bảng (5 bảng chiều, 2 bảng sự kiện) đã chứng minh được hiệu quả trong việc tổ chức dữ liệu logic, giảm thiểu dư thừa và hỗ trợ truy vấn phân tích nhanh chóng.
- **Trực quan hóa dữ liệu hiệu quả:** Dashboard xây dựng bằng Streamlit đã cung cấp cái nhìn tổng quan và chi tiết về giải đấu, giúp người dùng dễ dàng tra cứu thông tin về bảng xếp hạng, hiệu suất cầu thủ và đội bóng.

CHƯƠNG 2. Hạn chế của đề tài

Bên cạnh những kết quả đạt được, dự án vẫn còn một số hạn chế nhất định:

- **Phụ thuộc vào nguồn dữ liệu bên thứ ba:** Tốc độ trích xuất dữ liệu (Extract) còn chậm (mất khoảng 45-60 phút cho lần tải đầu tiên) do giới hạn rate-limit từ phía API của FBref và Understat.
- **Độ trễ dữ liệu:** Hệ thống hiện tại hoạt động theo cơ chế xử lý theo lô (Batch Processing) định kỳ, chưa hỗ trợ xử lý dữ liệu thời gian thực (Real-time streaming) ngay khi trận đấu đang diễn ra.
- **Phạm vi dữ liệu:** Hiện tại dự án mới chỉ tập trung vào giải Ngoại hạng Anh, chưa bao phủ các giải đấu lớn khác như La Liga, Serie A hay Bundesliga.

CHƯƠNG 3. Hướng phát triển

Để nâng cao chất lượng và tính ứng dụng của hệ thống, nhóm đề xuất các hướng phát triển trong tương lai:

- **Mở rộng phạm vi dữ liệu:** Tích hợp thêm dữ liệu từ các giải đấu khác và mở rộng lịch sử dữ liệu về các mùa giải trước năm 2021 để phục vụ phân tích xu hướng dài hạn.
- **Ứng dụng Học máy (Machine Learning):** Sử dụng dữ liệu lịch sử trong Kho dữ liệu để huấn luyện các mô hình dự đoán kết quả trận đấu, dự đoán phong độ cầu thủ hoặc gợi ý đội hình tối ưu.
- **Tối ưu hóa hiệu năng truy vấn:** Cài đặt thêm các chỉ mục (Index) nâng cao cho các cột thường xuyên truy vấn và xem xét sử dụng Materialized Views trong PostgreSQL để lưu trữ trước các kết quả tính toán phức tạp.
- **Nâng cấp giao diện:** Cải thiện Dashboard với các biểu đồ tương tác nâng cao (sử dụng Plotly/Altair), thêm tính năng so sánh trực tiếp giữa hai cầu thủ bất kỳ và cho phép xuất báo cáo dưới dạng PDF/Excel.

TÀI LIỆU THAM KHẢO

[1] W. McKinney, *Python for Data Analysis*, 2nd. O'Reilly Media, 2017.

[2] Sports Reference LLC. "Fbref: Football statistics and history." Nguồn dữ liệu thống kê cầu thủ và đội bóng, Accessed: Jan. 1, 2024. [Online]. Available: `https://fbref.com/`

[3] B. Harenslak and J. de Ruiter, *Data Pipelines with Apache Airflow*. Manning Publications, 2021, Tham khảo cho phần thiết kế DAG và Orchestration.

[4] J. VanderPlas, *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media, Inc., 2016, Tài liệu tham khảo cho Pandas và xử lý dữ liệu.