■ DATABASE SCHEMA DOCUMENTATION

Architecture Overview

Database System: PostgreSQL
Normalization Level: 3NF (Third Normal Form)
Number of Databases: 2 (cricketperformance, footballperformance)
Total Tables: 6 (3 per database)

■ CRICKET_PERFORMANCE DATABASE

1. athletes (Master Table)

Columns:
athlete_id SERIAL PRIMARY KEY
name VARCHAR(100)
country VARCHAR(50)
role VARCHAR(30) – Batter, Bowler, All-rounder, Wicket-keeper
age INT
Purpose: Stores unique cricket player records
Records: 300 players
Normalization: ■ 1NF – Atomic values, unique rows

2. stats (Detail Table - 1:N relationship)

Columns:
stat_id SERIAL PRIMARY KEY
athlete_id INT REFERENCES athletes(athlete_id) ON DELETE CASCADE
format VARCHAR(10) – TEST, ODI, T20
matches INT, runs INT, wickets INT
batting_avg DECIMAL(5,2), bowling_avg DECIMAL(5,2), strike_rate DECIMAL(6,2)
Purpose: Multi-format performance statistics
Records: ~900 (3 formats × 300 players)
Normalization: ■ 2NF – Separate table for repeating groups (formats)

3. injuries (Detail Table - 1:N relationship)

Columns:
injury_id SERIAL PRIMARY KEY
athlete_id INT REFERENCES athletes(athlete_id) ON DELETE CASCADE
injury_type VARCHAR(100), injury_start DATE, injury_end DATE
recovery_status VARCHAR(30), impact_level DECIMAL(3,1)
Purpose: Injury history and recovery tracking
Records: ~100+ injury records
Normalization: ■ 3NF – No transitive dependencies

Trigger: trg_athlete_insert

Function: log_athlete_insert()
Action: Automatically logs to *athlete_audit_log* when a new athlete is inserted

Logs: athlete_id, action_type ('INSERT'), timestamp, changed_by ('system')
Purpose: Ensures insert operations are tracked for auditing.

■ FOOTBALL_PERFORMANCE DATABASE

4. footballers (Master Table)

Columns:
footballer_id SERIAL PRIMARY KEY
name VARCHAR(100) NOT NULL
country VARCHAR(50), position VARCHAR(30), age INT
Purpose: Stores unique football player records
Records: 260 players
Normalization: ■ 1NF – Atomic values, unique rows

5. footballer_stats (Detail Table - 1:N relationship)

Columns:
stat_id SERIAL PRIMARY KEY
footballer_id INT REFERENCES footballers(footballer_id) ON DELETE CASCADE
season VARCHAR(20), matches INT DEFAULT 0, goals INT DEFAULT 0, assists INT DEFAULT 0, clean_sheets INT DEFAULT 0, pass_accuracy DECIMAL(5,2), tackles INT DEFAULT 0, saves INT DEFAULT 0
Purpose: Season-wise performance statistics
Records: ~500+
Normalization: ■ 2NF – Separate table for repeating groups (seasons)

6. footballer_injuries (Detail Table - 1:N relationship)

Columns:
injury_id SERIAL PRIMARY KEY
footballer_id INT REFERENCES footballers(footballer_id) ON DELETE CASCADE
injury_type VARCHAR(100), injury_start DATE, injury_end DATE, recovery_status VARCHAR(50)
Purpose: Injury history and recovery tracking
Records: ~100+ injury records
Normalization: ■ 3NF – No transitive dependencies

Trigger: trg_footballer_insert

Function: log_footballer_insert()
Action: Automatically logs to *footballer_audit_log* when a new footballer is inserted
Logs: footballer_id, action_type ('INSERT'), timestamp, changed_by ('system')
Purpose: Ensures insert operations are tracked for auditing.

■ NORMALIZATION ANALYSIS

■ 1NF – Atomic values, unique rows
■ 2NF – No partial dependencies
■ 3NF – No transitive dependencies

■ OPTIMIZATION FEATURES

1. Indexing Strategy
- Primary key indexes auto-created
- Recommended: idx_athletes_name, idx_stats_athlete_id, idx_footballers_name
2. Foreign Key Constraints – Referential integrity & cascade delete
3. Data Type Optimization – SERIAL, DECIMAL, VARCHAR, DATE
4. Default Values – DEFAULT 0 to prevent NULLs

■ ENTITY RELATIONSHIPS

Cricket: athletes (1) — (N) stats, (N) injuries
Football: footballers (1) — (N) footballer_stats, (N) footballer_injuries
Relationship Type: One-to-Many (1:N)

SQL Analytical Queries

SQL ANALYTICAL QUERIES

1. Get all cricket players

```
SELECT * FROM athletes ORDER BY name;
```

2. Get all football players

```
SELECT * FROM footballers ORDER BY name;
```

3. Get players from a specific country

```
SELECT * FROM athletes WHERE country = 'India';
```

4. Get all batsmen

```
SELECT * FROM athletes WHERE role = 'batter';
```

5. Get all forwards in football

```
SELECT * FROM footballers WHERE position = 'Forward';
```

6. Get player with their total runs across all formats

```
SELECT a.name, a.country, SUM(s.runs) as total_runs
FROM athletes a
JOIN stats s ON a.athlete_id = s.athlete_id
GROUP BY a.athlete_id, a.name, a.country
ORDER BY total_runs DESC;
```

...

(Queries 7 to 25 included similarly)

ER Diagram (Cricket & Football Databases)

```
CRICKET_PERFORMANCE DATABASE
--------------------------------
athletes (1) -> (N) stats
athletes (1) -> (N) injuries

FOOTBALL_PERFORMANCE DATABASE
--------------------------------
footballers (1) -> (N) footballer_stats
footballers (1) -> (N) footballer_injuries

Each player can have multiple stats and injury records.
Foreign keys maintain referential integrity with cascade delete.
```

All 25 SQL Analytical Queries

```
SQL ANALYTICAL QUERIES (ALL 25)

1. Get all cricket players
SELECT * FROM athletes ORDER BY name;

2. Get all football players
SELECT * FROM footballers ORDER BY name;

3. Get players from a specific country
SELECT * FROM athletes WHERE country = 'India';

4. Get all batsmen
SELECT * FROM athletes WHERE role = 'batter';

5. Get all forwards in football
SELECT * FROM footballers WHERE position = 'Forward';

6. Get player with their total runs across all formats
SELECT a.name, a.country, SUM(s.runs) as total_runs
FROM athletes a
JOIN stats s ON a.athlete_id = s.athlete_id
GROUP BY a.athlete_id, a.name, a.country
ORDER BY total_runs DESC;

7. Get top 10 goal scorers
SELECT f.name, f.country, SUM(fs.goals) as total_goals
FROM footballers f
JOIN footballer_stats fs ON f.footballer_id = fs.footballer_id
GROUP BY f.footballer_id, f.name, f.country
ORDER BY total_goals DESC
LIMIT 10;

8. Get average batting average by country
SELECT a.country, AVG(s.batting_avg) as avg_batting
FROM athletes a
JOIN stats s ON a.athlete_id = s.athlete_id
WHERE s.batting_avg > 0
GROUP BY a.country
ORDER BY avg_batting DESC;

9. Get players with most matches
SELECT a.name, SUM(s.matches) as total_matches
FROM athletes a
JOIN stats s ON a.athlete_id = s.athlete_id
GROUP BY a.athlete_id, a.name
ORDER BY total_matches DESC
LIMIT 10;

10. Get footballer stats by season
```

```
SELECT f.name, fs.season, fs.goals, fs.assists, fs.matches
FROM footballers f
JOIN footballer_stats fs ON f.footballer_id = fs.footballer_id
WHERE fs.season = '2023-24'
ORDER BY fs.goals DESC;

11. Get all injured cricket players
SELECT a.name, i.injury_type, i.recovery_status, i.injury_start
FROM athletes a
JOIN injuries i ON a.athlete_id = i.athlete_id
WHERE i.recovery_status != 'Recovered'
ORDER BY i.injury_start DESC;

12. Get players with ACL injuries
SELECT a.name, i.injury_type, i.injury_start, i.injury_end
FROM athletes a
JOIN injuries i ON a.athlete_id = i.athlete_id
WHERE i.injury_type ILIKE '%acl%';

13. Count injuries by type
SELECT injury_type, COUNT(*) as injury_count
FROM injuries
GROUP BY injury_type
ORDER BY injury_count DESC;

14. Get players with highest injury impact
SELECT a.name, a.role, i.injury_type, i.impact_level
FROM athletes a
JOIN injuries i ON a.athlete_id = i.athlete_id
ORDER BY i.impact_level DESC
LIMIT 10;

15. Get recovered footballers
SELECT f.name, fi.injury_type, fi.injury_start, fi.injury_end
FROM footballers f
JOIN footballer_injuries fi ON f.footballer_id = fi.footballer_id
WHERE fi.recovery_status = 'Recovered'
ORDER BY fi.injury_end DESC;

16. Count players by role
SELECT role, COUNT(*) as player_count
FROM athletes
GROUP BY role
ORDER BY player_count DESC;

17. Count footballers by position
SELECT position, COUNT(*) as player_count
FROM footballers
GROUP BY position
ORDER BY player_count DESC;

18. Get average age by country
SELECT country, AVG(age) as avg_age, COUNT(*) as player_count
```

```
FROM athletes
GROUP BY country
ORDER BY avg_age DESC;

19. Get total goals and assists by country
SELECT f.country, SUM(fs.goals) as total_goals, SUM(fs.assists) as total_assists
FROM footballers f
JOIN footballer_stats fs ON f.footballer_id = fs.footballer_id
GROUP BY f.country
ORDER BY total_goals DESC;

20. Get format-wise statistics
SELECT format,
       SUM(matches) as total_matches,
       SUM(runs) as total_runs,
       SUM(wickets) as total_wickets
FROM stats
GROUP BY format;

21. Get players with stats and injuries (LEFT JOIN)
SELECT a.name, a.role,
       COUNT(DISTINCT s.stat_id) as stat_records,
       COUNT(DISTINCT i.injury_id) as injury_records
FROM athletes a
LEFT JOIN stats s ON a.athlete_id = s.athlete_id
LEFT JOIN injuries i ON a.athlete_id = i.athlete_id
GROUP BY a.athlete_id, a.name, a.role;

22. Get top performers with no injuries
SELECT a.name, a.role, SUM(s.runs) as total_runs
FROM athletes a
JOIN stats s ON a.athlete_id = s.athlete_id
LEFT JOIN injuries i ON a.athlete_id = i.athlete_id
WHERE i.injury_id IS NULL
GROUP BY a.athlete_id, a.name, a.role
ORDER BY total_runs DESC
LIMIT 10;

23. Get audit log for recent player additions
SELECT al.log_id, a.name, a.country, al.action_type, al.action_timestamp
FROM athlete_audit_log al
JOIN athletes a ON al.athlete_id = a.athlete_id
ORDER BY al.action_timestamp DESC
LIMIT 20;

24. Get players with multiple injury records
SELECT a.name, a.role, COUNT(i.injury_id) as injury_count
FROM athletes a
JOIN injuries i ON a.athlete_id = i.athlete_id
GROUP BY a.athlete_id, a.name, a.role
HAVING COUNT(i.injury_id) > 1
ORDER BY injury_count DESC;
```

```
25. Get comprehensive player profile
SELECT
    a.name,
    a.country,
    a.role,
    a.age,
    COUNT(DISTINCT s.stat_id) as formats_played,
    SUM(s.matches) as total_matches,
    SUM(s.runs) as total_runs,
    SUM(s.wickets) as total_wickets,
    COUNT(DISTINCT i.injury_id) as total_injuries
FROM athletes a
LEFT JOIN stats s ON a.athlete_id = s.athlete_id
LEFT JOIN injuries i ON a.athlete_id = i.athlete_id
GROUP BY a.athlete_id, a.name, a.country, a.role, a.age
ORDER BY total_runs DESC
LIMIT 20;
```