Documentazione basi di dati:

A cura di:

1.Lucas Giavelli
2.Leonardo Scardoni
3.Alessio Tamburrano

Indice

Sezione 1: Specifiche del progetto

1.1 Introduzione

Nelle seguenti pagine si riporta la documentazione relativa alla realizzazione di un progetto volto a creare un portale web dedicato alla visualizzazione dei risultati e delle statistiche dell'NBA. L'idea alla base di questo progetto è soddisfare l'esigenza degli appassionati di sport di avere una fonte affidabile e facilmente accessibile per ottenere dati aggiornati sulle partite e sulle prestazioni dei giocatori. Il focus principale è orientato alla creazione di una piattaforma intuitiva, efficiente nella presentazione dei dati e compatibile sia con computer che con dispositivi mobili.

Sezione 2: Progettazione del diagramma ER

2.1 Dall’analisi allo schema ER

La progettazione concettuale è una fase critica nello sviluppo di una base di dati, poiché consente di trasformare i requisiti raccolti dalla fase iniziale in un modello comprensibile e orientato al significato. Uno strumento comune utilizzato per la progettazione concettuale è il Diagramma ER, che sta per "Entity-Relationship" (Entità-Relazione).

Il Diagramma ER rappresenta graficamente le entità, le proprietà e le relazioni tra di esse. Le entità sono oggetti o concetti significativi nel contesto del sistema che si sta progettando, le proprietà sono attributi associati alle entità, e le relazioni indicano come le entità sono connesse tra loro.

In un Diagramma ER, le entità sono solitamente rappresentate mediante rettangoli, le proprietà mediante ovali e le relazioni mediante linee che collegano le entità coinvolte. Le linee possono anche includere simboli, come frecce o rombi, per indicare la natura della relazione (ad esempio, uno a uno, uno a molti).

Questo tipo di modello consente ai progettisti di concentrarsi sulla comprensione del contesto e delle relazioni tra i dati senza dover affrontare dettagli tecnici implementativi. Una volta completato, il Diagramma ER serve da base per la progettazione logica del database, che definisce come questi concetti si tradurranno in strutture di dati effettive, tabelle e vincoli nel sistema di gestione di database scelto.

2.2 Dettagli sulle entità

Arena

- Descrizione: Rappresenta le arene dove si svolgono le partite NBA.
- Attributi: name (chiave primaria), city, state, country.

Game

- Descrizione: Dettagli sulle singole partite giocate.
- Attributi: id (chiave primaria), season, date, time, stage, totPeriods, status, nameArena (chiave esterna da arena), idVisitor (chiave esterna da team), idHome (chiave esterna da team).

Team

- Descrizione: Squadre NBA partecipanti al campionato.
- Attributi: id (chiave primaria), name, nickname, code, city, logo, allstar, conference, division.

Player

- Descrizione: I giocatori che partecipano ai campionati NBA.
- Attributi: id (chiave primaria), firstname, lastname, dateOfBirth, country, startYear, pro, height, weight, college, affiliation, jersey, isActive, pos.

PlayerTeam

- Descrizione: Associazione tra giocatori e squadre per specifiche stagioni.
- Attributi: idPlayer (chiave esterna da player), idTeam (chiave esterna da team), season. Chiave primaria composta da (idPlayer, idTeam, season).

StatsGame

- Descrizione: Statistiche di gioco per ogni partita e squadra.
- Attributi: idTeam (chiave esterna da team), idGame (chiave esterna da game), e vari attributi di statistica. Chiave primaria composta da (idTeam, idGame).

StatsPlayer

- Descrizione: Statistiche individuali dei giocatori per ogni partita.
- Attributi: idPlayer (chiave esterna da player), idTeam (chiave esterna da team), idGame (chiave esterna da game), e vari attributi di statistica. Chiave primaria composta da (idPlayer, idTeam, idGame).

StatsTeam

- Descrizione: Statistiche complessive di squadra per ogni stagione.
- Attributi: idTeam (chiave esterna da team), season, e vari attributi di statistica. Chiave primaria composta da (idTeam, season).

User

- Descrizione: Utenti del sistema che possono essere fan, analisti, ecc.
- Attributi: id (chiave primaria), email, password, name, role.

FavTeam

- Descrizione: Squadre preferite dagli utenti.
- Attributi: idUser (chiave esterna da user), idTeam (chiave esterna da team). Chiave primaria composta da (idUser, idTeam).

FavPlayer

- Descrizione: Giocatori preferiti dagli utenti.
- Attributi: idUser (chiave esterna da user), idPlayer (chiave esterna da player). Chiave primaria composta da (idUser, idPlayer).

Blog

- Descrizione: Blog o articoli scritti dagli utenti.
- Attributi: id (chiave primaria), idUser (chiave esterna da user), title, subtitle, dateBlog, img.

Paragraph

- Descrizione: Paragrafi specifici all'interno di un blog.
- Attributi: idBlog (chiave esterna da blog), index, title, content. Chiave primaria composta da (idBlog, index).

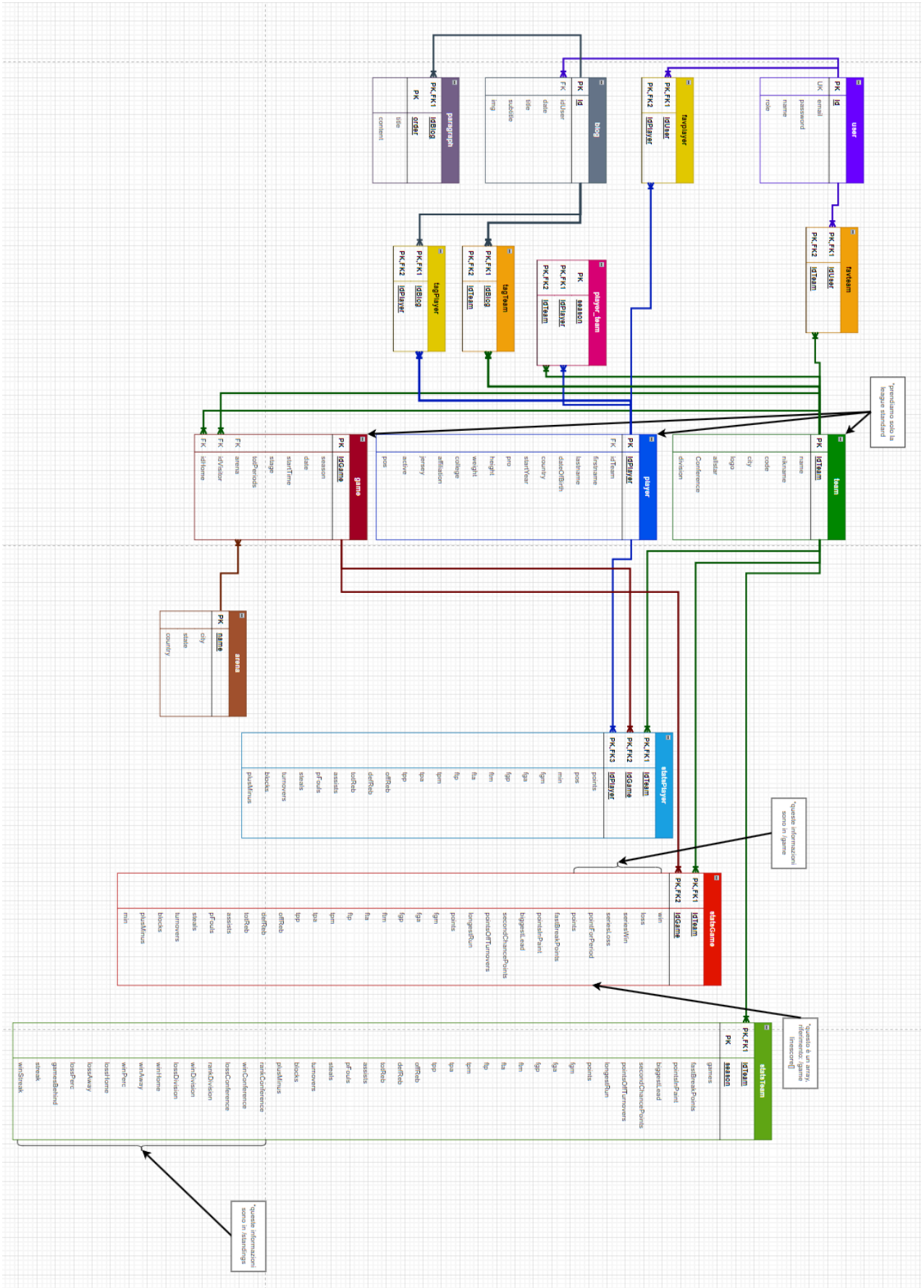
TagTeam

- Descrizione: Etichettatura dei blog con squadre specifiche.
- Attributi: idBlog (chiave esterna da blog), idTeam (chiave esterna da team). Chiave primaria composta da (idBlog, idTeam).

TagPlayer

- Descrizione: Etichettatura dei blog con giocatori specifici.
- Attributi: idBlog (chiave esterna da blog), idPlayer (chiave esterna da player). Chiave primaria composta da (idBlog, idPlayer).

2.3 Diagramma ER esteso



Sezione 3: Dal diagramma ER a Modello Relazionale

3.1 Passaggio dal diagramma ER al Modello Relazionale

Comprensione del Diagramma ER:

Prima di iniziare la trasformazione, è essenziale comprendere a fondo il diagramma ER. Questo include l'identificazione delle entità, delle relazioni, degli attributi, delle chiavi primarie e delle vincoli di integrità.

Trasformazione delle Entità:

Ogni entità nel diagramma ER diventa una tabella nel Modello Relazionale. Gli attributi dell'entità diventano le colonne della tabella. La chiave primaria dell'entità viene mantenuta come chiave primaria nella tabella corrispondente.

- Esempio: L'entità "Player" con attributi come id, nome, data di nascita si trasforma in una tabella "Player" con colonne corrispondenti.

Trasformazione delle Relazioni:

Le relazioni tra entità nel diagramma ER vengono trasformate in base al loro tipo (uno-a-uno, uno-a-molti, molti-a-molti):

- Relazioni Uno-a-Uno: Generalmente mantenute con chiavi esterne.
- Relazioni Uno-a-Molti: La chiave primaria dell'entità su "lato uno" diventa una chiave esterna nell'entità su "lato molti".
- Relazioni Molti-a-Molti: Trasformate in una nuova tabella che include le chiavi primarie delle entità collegate come chiavi esterne e chiave primaria composta.
- Esempio: Una relazione molti-a-molti tra "Player" e "Team" può essere trasformata in una tabella "PlayerTeam".

Trasformazione degli Attributi:

Gli attributi delle relazioni nel diagramma ER diventano parte delle tabelle nel modello relazionale. Questi possono essere inclusi come colonne nelle tabelle esistenti o nelle nuove tabelle create per le relazioni molti-a-molti.

Gestione delle Chiavi Esterne:

Importante per mantenere l'integrità referenziale. Le chiavi esterne sono utilizzate per collegare tabelle basate su relazioni definite nel diagramma ER.

Normalizzazione:

Dopo aver trasformato il diagramma ER in un modello relazionale, si procede con la normalizzazione. Questo processo ottimizza il modello riducendo la ridondanza dei dati e migliorando l'integrità. La normalizzazione è realizzata attraverso varie forme normali (1NF, 2NF, 3NF, ecc.).

Revisione e Validazione:

Infine, il modello relazionale ottenuto viene rivisto e validato contro i requisiti del sistema per assicurarsi che tutte le informazioni necessarie siano state correttamente rappresentate.

Sezione 4: Codice SQL

4.1 Struttura codice SQL per la realizzazione delle tabelle

```sql
-- phpMyAdmin SQL Dump
-- version 5.2.1
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Creato il: Dic 14, 2023 alle 10:45
-- Versione del server: 10.4.28-MariaDB
-- Versione PHP: 8.2.4
DROP DATABASE IF EXISTS `nbastat`;

CREATE DATABASE IF NOT EXISTS `nbastat` DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_general_ci;

use nbastat;

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";
```

phpMyAdmin SQL Dump

- Versione e Informazioni: Dettagli sulla versione di phpMyAdmin, server, e PHP.
- Host e Data: Specifica l'host (127.0.0.1, ovvero localhost) e la data di creazione del dump.

Creazione e Configurazione Iniziale del Database

- DROP DATABASE IF EXISTS nbastat: Rimuove il database nbastat se già esiste, per evitare conflitti.
- CREATE DATABASE IF NOT EXISTS nbastat: Crea un nuovo database chiamato nbastat, se non esiste già.
- USE nbastat: Seleziona nbastat come database su cui operare.
- Impostazioni Generali: Configura vari aspetti come il modo SQL, il fuso orario, e le impostazioni del set di caratteri.

Struttura delle Tabelle

Ogni blocco CREATE TABLE crea una nuova tabella nel database nbastat.

Tabella arena

```
CREATE TABLE `arena` (
  `name` varchar(255) NOT NULL,
  `city` varchar(255) DEFAULT NULL,
  `state` varchar(10) DEFAULT NULL,
  `country` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

- Scopo: Memorizza informazioni sulle arene, come nome, città, stato e paese.
- Struttura: Campi per nome, città, stato e paese.

Tabella game

```
CREATE TABLE `game` (
  `id` int(11) NOT NULL,
  `season` smallint(5) UNSIGNED NOT NULL,
  `date` date DEFAULT NULL,
  `time` time DEFAULT NULL,
  `stage` tinyint(4) DEFAULT NULL,
  `totPeriods` tinyint(4) DEFAULT NULL,
  `status` smallint(5) DEFAULT NULL,
  `nameArena` varchar(255) DEFAULT NULL,
  `idVisitor` int(11) DEFAULT NULL,
  `idHome` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

- Scopo: Contiene dettagli sui giochi, come stagione, data, ora, e squadre partecipanti.
- Struttura: Campi per ID, stagione, data, ora, e altri dettagli del gioco.

Tabella team

```
CREATE TABLE `team` (
  `id` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `nickname` varchar(255) DEFAULT NULL,
  `code` varchar(3) DEFAULT NULL,
  `city` varchar(255) DEFAULT NULL,
  `logo` varchar(255) DEFAULT NULL,
  `allstar` tinyint(1) DEFAULT NULL,
  `conference` varchar(255) DEFAULT NULL,
  `division` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

- Scopo: Raccoglie informazioni sulle squadre, come nome, nickname, codice, città e logo.
- Struttura: Campi per ID, nome, nickname, codice e altri dettagli della squadra.

Tabella player

```
CREATE TABLE `player` (
  `id` int(11) NOT NULL,
  `firstname` varchar(255) DEFAULT NULL,
  `lastname` varchar(255) DEFAULT NULL,
  `dateOfBirth` date DEFAULT NULL,
  `country` varchar(255) DEFAULT NULL,
  `startYear` smallint(6) DEFAULT NULL,
  `pro` tinyint(4) DEFAULT NULL,
  `height` float DEFAULT NULL,
  `weight` float DEFAULT NULL,
  `college` varchar(255) DEFAULT NULL,
  `affiliation` varchar(255) DEFAULT NULL,
  `jersey` tinyint(4) DEFAULT NULL,
  `isActive` tinyint(1) DEFAULT NULL,
  `pos` varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

- Scopo: Detiene i dettagli sui giocatori, come nome, data di nascita, altezza, peso e college.
- Struttura: Campi per ID, nome, cognome, data di nascita, paese e altri dettagli personali.

Tabella playerteam

```
CREATE TABLE `playerteam` (
  `idPlayer` int(11) NOT NULL,
  `idTeam` int(11) NOT NULL,
  `season` smallint(5) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

- Scopo: Associa giocatori a squadre per specifiche stagioni.
- Struttura: Campi per l'ID del giocatore, l'ID della squadra e la stagione.

Altre Tabelle

- statsgame: Statistiche specifiche delle partite.

```
CREATE TABLE `statsgame` (

  `idTeam` int(11) NOT NULL,

  `idGame` int(11) NOT NULL,

  `win` smallint(5) UNSIGNED DEFAULT NULL,

  `lose` smallint(5) UNSIGNED DEFAULT NULL,

  `seriesWin` tinyint(4) DEFAULT NULL,

  `seriesLose` tinyint(4) DEFAULT NULL,

  `pointsPeriod` varchar(255) DEFAULT NULL,

  `fastBreakPoint` smallint(5) UNSIGNED DEFAULT NULL,

  `pointsInPaint` smallint(5) UNSIGNED DEFAULT NULL,

  `biggestLead` tinyint(4) DEFAULT NULL,

  `secondChancePoints` smallint(5) UNSIGNED DEFAULT NULL,

  `pointsOffTurnovers` smallint(5) UNSIGNED DEFAULT NULL,
```

```
  `longestRun` tinyint(4) DEFAULT NULL,

  `points` smallint(5) UNSIGNED DEFAULT NULL,

  `fgm` smallint(5) UNSIGNED DEFAULT NULL,

  `fga` smallint(5) UNSIGNED DEFAULT NULL,

  `fgp` float DEFAULT NULL,

  `ftm` smallint(5) UNSIGNED DEFAULT NULL,

  `fta` smallint(5) UNSIGNED DEFAULT NULL,

  `ftp` float DEFAULT NULL,

  `tpm` smallint(5) UNSIGNED DEFAULT NULL,

  `tpa` smallint(5) UNSIGNED DEFAULT NULL,

  `tpp` float DEFAULT NULL,

  `offReb` smallint(5) UNSIGNED DEFAULT NULL,

  `defReb` smallint(5) UNSIGNED DEFAULT NULL,

  `totReb` smallint(5) UNSIGNED DEFAULT NULL,

  `assists` smallint(5) UNSIGNED DEFAULT NULL,

  `pFouls` smallint(5) UNSIGNED DEFAULT NULL,

  `steals` smallint(5) UNSIGNED DEFAULT NULL,

  `turnovers` smallint(5) UNSIGNED DEFAULT NULL,

  `blocks` smallint(5) UNSIGNED DEFAULT NULL,

  `plusMinus` smallint(6) DEFAULT NULL,

  `min` varchar(6) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

- statsplayer: Statistiche dettagliate dei giocatori.

```
CREATE TABLE `statsplayer` (
  `idPlayer` int(11) NOT NULL,
  `idTeam` int(11) NOT NULL,
  `idGame` int(11) NOT NULL,
  `points` tinyint(4) DEFAULT NULL,
  `pos` varchar(2) DEFAULT NULL,
  `min` float DEFAULT NULL,
  `fgm` smallint(5) UNSIGNED DEFAULT NULL,
  `fga` smallint(5) UNSIGNED DEFAULT NULL,
  `fgp` float DEFAULT NULL,
  `ftm` smallint(5) UNSIGNED DEFAULT NULL,
  `fta` smallint(5) UNSIGNED DEFAULT NULL,
  `ftp` float DEFAULT NULL,
  `tpm` smallint(5) UNSIGNED DEFAULT NULL,
  `tpa` smallint(5) UNSIGNED DEFAULT NULL,
  `tpp` float DEFAULT NULL,
  `offReb` smallint(5) UNSIGNED DEFAULT NULL,
  `defReb` smallint(5) UNSIGNED DEFAULT NULL,
  `totReb` smallint(5) UNSIGNED DEFAULT NULL,
  `assists` smallint(5) UNSIGNED DEFAULT NULL,
```

```
  `pFouls` smallint(5) UNSIGNED DEFAULT NULL,

  `steals` smallint(5) UNSIGNED DEFAULT NULL,

  `turnovers` smallint(5) UNSIGNED DEFAULT NULL,

  `blocks` smallint(5) UNSIGNED DEFAULT NULL,

  `plusMinus` smallint(6) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

- statsteam: Statistiche complessive delle squadre.

```
CREATE TABLE `statsteam` (

  `idTeam` int(11) NOT NULL,

  `season` smallint(5) UNSIGNED NOT NULL,

  `games` smallint(5) UNSIGNED DEFAULT NULL,

  `fastBreakPoints` smallint(5) UNSIGNED DEFAULT NULL,

  `pointsInPaint` smallint(5) UNSIGNED DEFAULT NULL,

  `biggestLead` smallint(5) UNSIGNED DEFAULT NULL,

  `secondChancePoints` smallint(5) UNSIGNED DEFAULT NULL,

  `pointsOffTurnovers` smallint(5) UNSIGNED DEFAULT NULL,

  `longestRun` smallint(5) UNSIGNED DEFAULT NULL,

  `points` smallint(5) UNSIGNED DEFAULT NULL,

  `fgm` smallint(5) UNSIGNED DEFAULT NULL,

  `fga` smallint(5) UNSIGNED DEFAULT NULL,
```

```
`fgp` float DEFAULT NULL,

`ftm` smallint(5) UNSIGNED DEFAULT NULL,

`fta` smallint(5) UNSIGNED DEFAULT NULL,

`ftp` float DEFAULT NULL,

`tpm` smallint(5) UNSIGNED DEFAULT NULL,

`tpa` smallint(5) UNSIGNED DEFAULT NULL,

`tpp` float DEFAULT NULL,

`offReb` smallint(5) UNSIGNED DEFAULT NULL,

`defReb` smallint(5) UNSIGNED DEFAULT NULL,

`totReb` smallint(5) UNSIGNED DEFAULT NULL,

`assists` smallint(5) UNSIGNED DEFAULT NULL,

`pFouls` smallint(5) UNSIGNED DEFAULT NULL,

`steals` smallint(5) UNSIGNED DEFAULT NULL,

`turnovers` smallint(5) UNSIGNED DEFAULT NULL,

`blocks` smallint(5) UNSIGNED DEFAULT NULL,

`plusMinus` smallint(6) DEFAULT NULL,

`rankConference` smallint(5) DEFAULT NULL,

`winConference` smallint(5) DEFAULT NULL,

`lossConference` smallint(5) DEFAULT NULL,

`rankDivision` smallint(5) DEFAULT NULL,

`winDivision` smallint(5) DEFAULT NULL,

`lossDivision` smallint(5) DEFAULT NULL,

`winHome` smallint(5) DEFAULT NULL,
```

```
  `winAway` smallint(5) DEFAULT NULL,

  `winPerc` float DEFAULT NULL,

  `lossHome` smallint(5) DEFAULT NULL,

  `lossAway` smallint(5) DEFAULT NULL,

  `lossPerc` float DEFAULT NULL,

  `gamesBehind` smallint(5) DEFAULT NULL,

  `streak` smallint(5) DEFAULT NULL,

  `winStreak` tinyint(1) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

- user: Dettagli degli utenti, come email e password.

```
CREATE TABLE `user` (

  `id` int(11) NOT NULL AUTO_INCREMENT,

  `email` varchar(255) NOT NULL,

  `password` varchar(255) NOT NULL,

  `name` varchar(255) NOT NULL,

  `role` smallint(5) NOT NULL,

  PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

- favteam e favplayer: Preferenze dei fan per squadre e giocatori.

```
CREATE TABLE `favteam` (

  `idUser` int(11) NOT NULL,

  `idTeam` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;

CREATE TABLE `favplayer` (

  `idUser` int(11) NOT NULL,

  `idPlayer` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

- blog, paragraph, tagTeam, tagPlayer: Gestisce un sistema di blog, con post, paragrafi e tag.

```
CREATE TABLE `blog` (

  `id` int(11) NOT NULL AUTO_INCREMENT,

  `idUser` int(11) NOT NULL,

  `title` varchar(255) NOT NULL,

  `subtitle` varchar(255) DEFAULT NULL,

  `dateBlog` date NOT NULL,

  `img` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`id`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;

CREATE TABLE `paragraph` (

  `idBlog` int(11) NOT NULL,

  `index` int(11) NOT NULL,

  `title` varchar(255) NOT NULL,

  `content` text NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;

CREATE TABLE `tagTeam` (

  `idBlog` int(11) NOT NULL,

  `idTeam` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;

CREATE TABLE `tagPlayer` (

  `idBlog` int(11) NOT NULL,

  `idPlayer` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

Indici e Vincoli

Per ogni tabella, vengono creati indici primari e chiavi esterne per ottimizzare le prestazioni e mantenere l'integrità dei dati.

Indici tabella arena

```
ALTER TABLE `arena`

  ADD PRIMARY KEY (`name`);
```

Indici e limiti tabella game

```
--

-- Indici per le tabelle `game`

--

ALTER TABLE `game`

  ADD PRIMARY KEY (`id`),

  ADD KEY `fk_arena` (`nameArena`),

  ADD KEY `fk_home_team` (`idHome`),

  ADD KEY `fk_visitor_team` (`idVisitor`);

--

-- Limiti per la tabella `game`

--

ALTER TABLE `game`

  ADD CONSTRAINT `fk_arena` FOREIGN KEY (`nameArena`) REFERENCES
`arena` (`name`),

  ADD CONSTRAINT `fk_home_team` FOREIGN KEY (`idHome`) REFERENCES
`team` (`id`),
```

```
  ADD CONSTRAINT `fk_visitor_team` FOREIGN KEY (`idVisitor`) REFERENCES
`team` (`id`);
```

Indici tabella team

```
--

-- Indici per le tabelle `team`

--

ALTER TABLE `team`

  ADD PRIMARY KEY (`id`);
```

Indici tabella player

```
--

-- Indici per le tabelle `player`

--

ALTER TABLE `player`

  ADD PRIMARY KEY (`id`);
```

Indici e limiti per la tabella playerteam

```
--

-- Indici per le tabelle `playerteam`

--

ALTER TABLE `playerteam`

  ADD PRIMARY KEY (`idPlayer`,`idTeam`,`season`),
```

```
  ADD KEY `fk_playerteam_player` (`idPlayer`),

  ADD KEY `fk_playerteam_team` (`idTeam`);

--

-- Limiti per la tabella `playerteam`

--

ALTER TABLE `playerteam`

  ADD CONSTRAINT `fk_playerteam_player` FOREIGN KEY (`idPlayer`)
REFERENCES `player` (`id`),

  ADD CONSTRAINT `fk_playerteam_team` FOREIGN KEY (`idTeam`) REFERENCES
`team` (`id`);
```

Indici e limiti per la tabella statsteam

```
--

-- Indici per le tabelle `statsteam`

--

ALTER TABLE `statsteam`

  ADD PRIMARY KEY (`idTeam`,`season`),

  ADD KEY `fk_statsTeam_team` (`idTeam`);

--

-- Limiti per la tabella `statsteam`

--

ALTER TABLE `statsteam`

  ADD CONSTRAINT `fk_statsTeam_team` FOREIGN KEY (`idTeam`) REFERENCES
`team` (`id`);
```

Indici e limiti per la tabella statsgame

```
--

-- Indici per le tabelle `statsgame`

--

ALTER TABLE `statsgame`

  ADD PRIMARY KEY (`idTeam`,`idGame`),

  ADD KEY `fk_statGame_game` (`idGame`),

  ADD KEY `fk_statGame_team` (`idTeam`);

--

-- Limiti per la tabella `statsgame`

--

ALTER TABLE `statsgame`

  ADD CONSTRAINT `fk_statGame_game` FOREIGN KEY (`idGame`) REFERENCES
`game` (`id`),

  ADD CONSTRAINT `fk_statGame_team` FOREIGN KEY (`idTeam`) REFERENCES
`team` (`id`);
```

Indici e limiti per la tabella statsplayer

```
--

-- Indici per le tabelle `statsplayer`

--

ALTER TABLE `statsplayer`
```

```
  ADD PRIMARY KEY (`idPlayer`,`idTeam`,`idGame`),

  ADD KEY `fk_statsPlayer_team` (`idTeam`),

  ADD KEY `fk_statsPlayer_game` (`idGame`),

  ADD KEY `fk_statsPlayer_player` (`idPlayer`);

--

-- Limiti per la tabella `statsplayer`

--

ALTER TABLE `statsplayer`

  ADD CONSTRAINT `fk_statsPlayer_game` FOREIGN KEY (`idGame`)
REFERENCES `game` (`id`),

  ADD CONSTRAINT `fk_statsPlayer_player` FOREIGN KEY (`idPlayer`)
REFERENCES `player` (`id`),

  ADD CONSTRAINT `fk_statsPlayer_team` FOREIGN KEY (`idTeam`)
REFERENCES `team` (`id`);
```

Indici e limiti per la tabella favteam

```
--

-- Indici per le tabelle 'favteam'

--

ALTER TABLE `favteam`

  ADD PRIMARY KEY (`idUser`, `idTeam`),

  ADD KEY `fk_favteam_user` (`idUser`),

  ADD KEY `fk_favteam_team` (`idTeam`);

--
```

```
-- Limiti per la tabella 'favteam'

--

ALTER TABLE `favteam`

  ADD CONSTRAINT `fk_favteam_team` FOREIGN KEY (`idTeam`) REFERENCES
`team`(`id`),

  ADD CONSTRAINT `fk_favteam_user` FOREIGN KEY (`idUser`) REFERENCES
`user`(`id`);
```

Indici e limiti per la tabella favplayer

```
--

-- Indici per le tabelle 'favplayer'

--

ALTER TABLE `favplayer`

  ADD PRIMARY KEY (`idUser`, `idPlayer`),

  ADD KEY `fk_favplayer_user` (`idUser`),

  ADD KEY `fk_favplayer_player` (`idPlayer`);

--

-- Limiti per la tabella 'favplayer'

--

ALTER TABLE `favplayer`

  ADD CONSTRAINT `fk_favplayer_player` FOREIGN KEY (`idPlayer`)
REFERENCES `player`(`id`),
```

```
  ADD CONSTRAINT `fk_favplayer_user` FOREIGN KEY (`idUser`) REFERENCES
`user`(`id`);
```

Indici e limiti per la tabella blog

```
--

-- Indici per le tabelle 'blog'

--

ALTER TABLE `blog`

  ADD KEY `fk_blog_user` (`idUser`);

--

-- Limiti per la tabella 'blog'

--

ALTER TABLE `blog`

  ADD CONSTRAINT `fk_blog_user` FOREIGN KEY (`idUser`) REFERENCES
`user` (`id`);
```

Indici e limiti per la tabella paragraph

```
--

-- Indici per le tabelle `paragraph`

--

ALTER TABLE `paragraph`

  ADD PRIMARY KEY (`idBlog`,`index`),

  ADD KEY `fk_paragraph_blog` (`idBlog`);
```

```
--

-- Limiti per la tabella `paragraph`

--

ALTER TABLE `paragraph`

  ADD CONSTRAINT `fk_paragraph_blog` FOREIGN KEY (`idBlog`) REFERENCES
`blog` (`id`);
```

Indici e limiti per la tabella tagteam

```
--

-- Indici per la tabella 'tagTeam'

--

ALTER TABLE `tagTeam`

  ADD PRIMARY KEY (`idBlog`, `idTeam`),

  ADD KEY `fk_tagTeam_blog` (`idBlog`),

  ADD KEY `fk_tagTeam_team` (`idTeam`);

--

-- Limiti per la tabella 'tagteam'

--

ALTER TABLE `tagteam`

  ADD CONSTRAINT `fk_tagteam_blog` FOREIGN KEY (`idBlog`) REFERENCES
`blog`(`id`),

  ADD CONSTRAINT `fk_tagteam_team` FOREIGN KEY (`idTeam`) REFERENCES
`team`(`id`);
```

Indici e limiti per la tabella tagplayer

```
--

-- Indici per le tabelle 'tagPlayer'

--

ALTER TABLE `tagPlayer`

  ADD PRIMARY KEY (`idBlog`, `idPlayer`),

  ADD KEY `fk_tagPlayer_blog` (`idBlog`),

  ADD KEY `fk_tagPlayer_player` (`idPlayer`);

--

-- Limiti per la tabella 'tagplayer'

--

ALTER TABLE `tagplayer`

  ADD CONSTRAINT `fk_tagplayer_blog` FOREIGN KEY (`idBlog`) REFERENCES
`blog`(`id`),

  ADD CONSTRAINT `fk_tagplayer_player` FOREIGN KEY (`idPlayer`)
REFERENCES `player`(`id`);
```

Creazione Utente e Assegnazione dei Permessi

```
CREATE USER IF NOT EXISTS 'backend-lastgame'@'localhost' IDENTIFIED BY
'5lE;R230LO_%*v{(3+ECbd:';
GRANT SELECT, INSERT, UPDATE, DELETE ON nbastat.* TO
'backend-lastgame'@'localhost';
FLUSH PRIVILEGES;
```

- CREATE USER: Crea un nuovo utente nel database.
- GRANT: Assegna permessi specifici all'utente, come SELECT, INSERT, UPDATE e DELETE.

4.2 Interrogazioni, Inserimenti e Cancellazioni degne di nota

Da un punto di vista del popolamento del database, il codice della classe ApiCaller gioca un ruolo cruciale nel processo di acquisizione e inserimento dei dati nell'ambito di un'applicazione che si occupa di dati relativi alla NBA. Ecco un riassunto del suo ruolo e funzionamento in questo contesto:

Ruolo della Classe ApiCaller nel Popolamento del Database

- **Chiamate API per Dati NBA:**

  La classe ApiCaller è responsabile per fare chiamate API all'API NBA fornita da api-sports.io. Questo permette di accedere a una vasta gamma di dati relativi alla NBA, come informazioni su partite, squadre, giocatori, statistiche, ecc.

- **Acquisizione Dati Dinamica:**

  Utilizzando diversi endpoint e parametri di query, ApiCaller può richiedere specifici set di dati necessari per popolare il database dell'applicazione. Questo include dati per diversi giochi, stagioni, statistiche dei giocatori, risultati delle partite, e così via.

- **Elaborazione e Formattazione dei Dati:**

  Dopo aver ricevuto i dati dall'API, questi possono essere elaborati, formattati e trasformati in modo che corrispondano allo schema del database dell'applicazione.

- **Inserimento dei Dati nel Database:**

  Una volta che i dati sono stati elaborati, possono essere inseriti nel database. Questo può essere fatto attraverso meccanismi di

persistenza dati integrati nell'applicazione, come ORM (Object-Relational Mapping) o direttamente attraverso SQL.

- **Aggiornamento Regolare del Database:**

  Poiché le informazioni relative alla NBA possono cambiare frequentemente (ad esempio, risultati delle partite, statistiche dei giocatori), ApiCaller può essere utilizzato per aggiornare regolarmente il database con i dati più recenti.

- **Automazione e Scheduling:**

  Il processo di chiamata API e popolamento del database può essere automatizzato e programmato per avvenire a intervalli regolari, assicurando così che il database sia sempre aggiornato.