DataBase Design

4.1: Introduction

Database design is used to manage large bodies of information. The management of data involves both the definition of structure of storage and provision for the manipulation of information. In addition, the database system must provide the safety of the information solved despite system crashes or due to the attempts at unauthorized access. We have to fulfil certain conditions such as:

* Control Redundancy.
* Easy to use.
* Data Independence.
* Accuracy and Integrity.
* Recovery and Security.
* Performance.

4.2: Purpose and Scope

The purpose of database design is to create a structured and efficient system for organizing and storing data. It involves defining the logical and physical structure of a database, determining the relationships between data elements, and establishing rules to ensure data integrity and security. The ultimate goal is to provide a solid foundation for data management, retrieval, and manipulation.

4.3: DataBase Identification

Database identification refers to the process of determining the need for a database in a particular context or scenario. It involves recognizing the data requirements, the purpose of data storage, and the potential benefits of using a database management system (DBMS) to organize and manage that data.

4.4: Schema Information

Schema information refers to the details and specifications of a database schema. A database schema defines the structure, organization, and relationships of the data within a database. It includes information about tables, columns, data types, constraints, relationships, and other elements that define the logical and physical structure of the database.

4.5: Table Definition

Tbl\_admin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ad\_id | ad\_name | ad\_email | ad\_password | ad\_date |

Tbl\_doctor

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| doc\_id | doc\_  name | doc\_  img | doc\_  about | doc\_  phone | doc  \_email | doc\_exp | doc\_  educ | doc\_  password | doc\_  status |

Tbl\_patient

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| pat\_id | pat\_name | pat\_phone | pat\_email | pat\_password | pat\_date |

Tbl\_appointment

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ap\_id | Doc\_id | Pat\_  id | age | phoneno | gender | Ap\_  date | Ap\_  time | problem | Amount | Ap\_  status | App\_  date |

Tbl\_feedback

|  |  |  |  |
| --- | --- | --- | --- |
| feed\_id | pat\_id | feed | feed\_date |

Tbl\_payment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pay\_id | ap\_id | pay\_method | amount | trans\_id | pay\_status | pay\_date |

Tbl\_schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| sch\_id | doc\_id | days | time | sch\_status | sch\_date |

Tbl\_prescription

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| pr\_id | doc\_id | pat\_id | advice | symptoms | test | pr\_status | pr\_date |

Tbl\_prescription details

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| pd\_id | pr\_id | medicine | timings | bf\_af | days\_weeks | pd\_status | pd\_date |

4.6: Physical Design

Physical design refers to the process of determining the physical implementation details of a database schema in a specific database management system (DBMS). It involves making decisions about how the data will be stored, organized, and accessed on the underlying storage devices. The physical design is responsible for optimizing performance, storage efficiency, and scalability of the database.

4.7: Data Dictionary

* Admin table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| ad\_id | int | 11 | Primary key | Primary key of the table |
| ad\_name | varchar | 50 | Not null | To store admin’s id |
| ad\_email | varchar | 50 | Not null | To store admin’s email |
| ad\_password | varchar | 250 | Not null | To store admin’s password |
| ad\_date | date | 6 | Not null | To store date |

* Doctor table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| doc\_id | int | 11 | primary key | primary key of the table |
| doc\_name | varchar | 50 | not null | to store doctor’s name |
| doc\_img | varchar | 350 | not null | to store doctor’s image |
| doc\_about | varchar | 250 | not null | to store doctor’s about |
| doc\_phone | bigint | 12 | not null | to store doctor’s phone number |
| doc\_email | varchar | 50 | not null | To store doctor’s email |
| doc\_exp | varchar | 200 | not null | To store doctor’s experience |
| doc\_educ | varchar | 200 | not null | To store doctor’s education |
| doc\_password | varchar | 250 | not null | To store doctor’s password |
| doc\_status | varchar | 100 | not null | To store doctor’s status |

* patient table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| p\_id | int | 11 | Primary key | Primary key of the table |
| p\_name | varchar | 50 | Not null | To store patient’s name |
| p\_phone | bigint | 12 | Not null | To store patient’s phone |
| p\_email | varchar | 50 | Not null | To store patient’s email |
| p\_password | varchar | 250 | Not null | To store patient’s password |
| p\_date | date | 6 | Not null | To store date |

* Appointment table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| ap\_id | int | 11 | primary key | primary key of the table |
| doc\_id | int | 50 | Foreign key | Foreign key of the table |
| p\_id | int | 350 | Foreign key | Foreign key of the table |
| age | varchar | 250 | not null | to store age of the patient |
| Phone number | bigint | 12 | not null | to store patient’s phone number |
| gender | varchar | 50 | not null | To store patient’s gender |
| ap\_date | date | 200 | not null | To store appointment date |
| ap\_time | time | 200 | not null | To store appointment time |
| problem | varchar | 250 | not null | To store patient’s problem |
| amount | Varc har | 100 | not null | To store appointment amount |
| ap\_status | varchar | 100 | Not null | To store appointment status |
| app\_date | Date | 6 | Not null | To store date |

* Feedback table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| feed\_id | int | 11 | Primary key | Primary key of the table |
| p\_id | Int | 11 | Foreign key | Foreign key of the table |
| feed | varchar | 50 | Not null | To store feedback |
| feed\_date | date | 6 | Not null | To store feedback date |

* Feedback table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| feed\_id | int | 11 | Primary key | Primary key of the table |
| p\_id | Int | 11 | Foreign key | Foreign key of the table |
| feed | varchar | 50 | Not null | To store feedback |
| feed\_date | date | 6 | Not null | To store feedback date |

* payment table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| pay\_id | int | 11 | Primary key | Primary key of the table |
| ap\_id | int | 11 | Foreign key | Foreign key of the table |
| pay\_method | varchar | 12 | Not null | To store payment method |
| amount | varchar | 50 | Not null | To store payment amount |
| trans\_id | varchar | 20 | Not null | To store transaction id |
| pay\_status | varchar | 100 | Not null | To store the status |
| pay\_date | date | 6 | Not null | To store payment date |

* Schedule table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| sch\_id | int | 11 | Primary key | Primary key of the table |
| doc\_id | varchar | 50 | Not null | Foreign key of the table |
| days | varchar | 50 | Not null | To store the schedule days |
| time | varchar | 100 | Not null | To store time |
| Sch\_status | varchar | 10 | Not null | To store status |
| Sch\_date | date | 6 | Not null | To store schedule date |

* Prescription table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| pr\_id | int | 11 | Primary key | Primary key of the table |
| doc\_id | int | 11 | Foreign key | Foreign key of the table |
| p\_id | int | 11 | Foreign key | Foreign key of the table |
| advice | varchar | 50 | Not null | To store prescription advice |
| symptoms | varchar | 20 | Not null | To store prescription symptoms |
| test | varchar | 100 | Not null | To store the prescription test details |
| pr\_status | varchar | 6 | Not null | To store the status |
| pr\_date | date | 6 | Not null | To store prescription date |

* Prescription details table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| pd\_id | int | 11 | Primary key | Primary key of the table |
| pr\_id | int | 11 | Foreign key | Foreign key of the table |
| medicine | varchar | 50 | Not null | to store prescription |
| timings | varchar | 50 | Not null | To store timings |
| bf\_af | varchar | 20 | Not null | To store prescription before and after meals |
| days\_weeks | varchar | 100 | Not null | To store the days |
| pd\_status | varchar | 6 | Not null | To store the status |
| pd\_date | date | 6 | Not null | To store prescription date |

An ER (Entity-Relationship) diagram is a graphical representation of the entities (objects or concepts) within a system or domain, their attributes (properties or characteristics), and the relationships between entities. It is a visual tool used in database design to illustrate the logical structure of a database.

|  |  |  |
| --- | --- | --- |
| name | notation | discription |
| entity |  | The entity is represented by a box within the ERD. Entities are abstract concepts, each representing one or more instances of the concept in question. An entity might be considered a container that holds all of the instances of a particular thing in a system. |
| Relationship |  | Relationships are represented by  Diamonds. A relationship is a named collection or association between entities or used to relate  to two or more entities with some common attributes or meaningful interaction between the objects. |
| link |  | Lines link attributes to entity sets or entity sets to relationship sets |
| attribute |  | Attributes are represented by  Oval. An attribute is a single data item related to a database object. The database scheme  associates with each database entity. |
| Derived attribute | Circle | Dashed ellipse denotes derived attributes. |
| Key Attribute or Single  Valued Attribute |  | As entity type usually has an attribute whose values are distinct for each individual entry in the entity set. It is represented by an underline word in ellipse. |
| Multivalued Attribute |  | Attributes that have different numbers of values for a particular attribute. It is represented by a double ellipse. |
| Cardinality Ratio | 1:1  1:M  M:1  M:M | It specifies the maximum  number of relationships  instances that an entity can participate in. There are four cardinality ratios. |
|  |  |  |

4.9: Data Base Administration

4.9.1: System Information

* **Server:** localhost via TCP/IP
* **Server type**: MySQL
* **Server version:** 5.6.12-log - MySQL Community Server (GPL)
* **Protocol version:** 10
* **User:** root@localhost
* **Server charset:** UTF-8 Unicode (utf8)

4.9.2: DBMS Configuration

* **XAMPP Version:** 2.4
* **MySQL version:** 5.6.12

4.9.3: Support Software Required

The system installs MYSQL server while installing WAMP software. All the backup content stores in MYSQL data folder.

4.9.4: Storage Requirement

The storage engine represents the heart of a MySQL Server.

* Recovering the database from system failure
* Management of files and database pages used to store data
* Manage data buffers and system IO to the physical data pages
  + Manage locking and concurrency issues

4.9.5: Backup and Recovery

Database recovery is the process of restoring the databases to a correct state following a failure. The failure may be the result of a system crash due to hardware of software errors, a media failure, such as a head crash, or a software error in the application, such as a logical error in the program that is accessing the database. It is the responsibility of DBMS to ensure that the database is reliable and remains in a consistent state in the presence of failure. In general, backup and recovery refer to the various strategies and procedures involved in protecting the database against data loss and reconstructing the data such as that no data is lost after failure.