**DATABASE DESIGN**

**DATA INTEGRITY**

* The term *data integrity* refers to the accuracy and consistency of data.
* When creating databases, attention needs to be given to data integrity and how to maintain it. A good database will enforce data integrity whenever possible.
* For example, a user could accidentally try to enter a phone number into a date field. If the system enforces data integrity, it will prevent the user from making these mistakes.

**Risks to Data Integrity: -**

Some Examples where data integrity is at risk: -

* A user tries to enter a date outside an acceptable range.
* A user tries to enter a phone number in the wrong format.
* A bug in an application attempts to delete the wrong record.
* While transferring data between two databases, the developer accidentally tries to insert the data into the wrong table.

**Types of Data Integrity: -**

* Entity Integrity
* Referential Integrity
* Domain Integrity
* User-defined Integrity

**Entity Integrity**

* *Entity integrity* defines each row to be unique within its table. No two rows can be the same.
* To achieve this, a primary key can be defined. The primary key field contains a unique identifier – no two rows can contain the same unique identifier.

**Referential Integrity**

* Referential integrity is concerned with relationships.
* When two or more tables have a relationship, we have to ensure that the foreign key value matches the primary key value at all times. We don’t want to have a situation where a foreign key value has no matching primary key value in the primary table. This would result in a create inconsistency in database.

So referential integrity will prevent users from:

* Adding records to a related table if there is no associated record in the primary table.
* Changing values in a primary table that result in orphaned records in a related table.
* Deleting records from a primary table if there are matching related records.

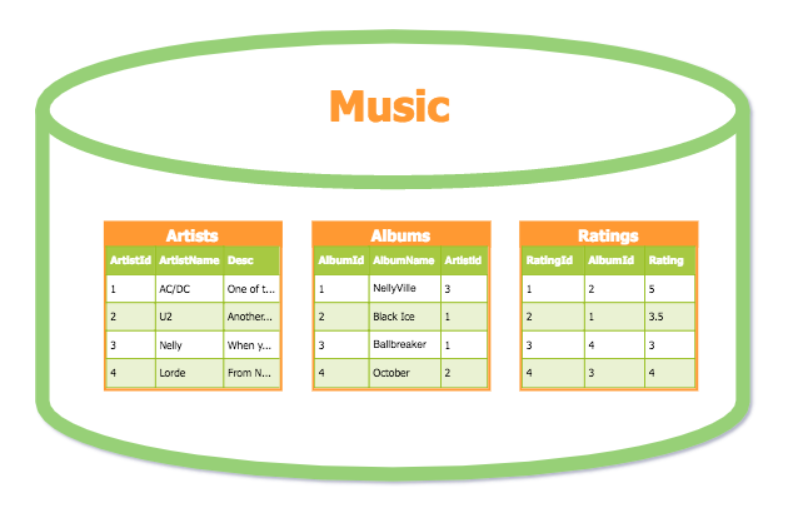
**Domain Integrity**

* *Domain integrity* concerns the validity of entries for a given column.
* Selecting the appropriate data type for a column is the first step in maintaining domain integrity.
* Other steps could include, setting up appropriate constraints and rules to define the data format and/or restricting the range of possible values.

**User-Defined Integrity**

* *User-defined integrity* allows the user to apply business rules to the database that aren’t covered by any of the other three data integrity types.

**RELATIONSHIPS**

* Relationships are an integral part of relational databases.
* Relationships enable us to join data from different tables, so that we can see data that is related – regardless of which table it was entered into.
* If you look at the above diagram, you can see that the *Artists* table has an *ArtistId* field. But you can see that the *Albums* table also has an *ArtistId* field.
* There’s a good reason for this. The *ArtistId* field in the Albums table stores the same value as the *ArtistId* field in the Artists table. Doing this enables us to see which artist each album belongs to.
* So we can see that the album with an *AlbumId* of 1 belongs to the artist with an *ArtistId* of 3.
* One of the main benefits of doing this is that we only need to enter data once. We can enter an artist once, then have many albums referencing that artist.
* If we didn’t use a relationship, we would need to store the full artist details against every album related to that artist. So if an artist produced 10 albums, that artist’s name and other details would need to be duplicated 10 times in the Artists table. Which is not at all a good practice.

**Types of Relationships: -**

There are three types of relationships in database design:

* **One-to-One:** A row in table A can have only one matching row in table B, and vice versa.
* **One-to-Many (or Many-to-One):** A row in table A can have many matching rows in table B, but a row in table B can have only one matching row in table A.
* **Many-to-Many:** A row in table A can have many matching rows in table B, and vice versa.

**What are the Benefits of Relationships?**

Relationships are the basis of any relational database management system (RDBMS). Relationships are a very powerful tool to use in database design.

Some key benefits of relationships in database design: -

* Reduces storage requirements
* Helps maintain data integrity (Referential Data Integrity)
* Helps increase usability (the degree to which something is able to be used) for end users
* Easier data maintenance
* Helps with security
* Helps with scalability or expansion of the database

**Reduces Storage Requirement: -**

* Storing an ID is typically more efficient than storing the full text – especially if the text is long.

**Helps Maintain Referential Data Integrity: -**

* By entering the data once, then referencing that one record, there is less room for error.
* In above example, if there is no relationship between ALBUM and ARTIST then we need to give complete artist details for each album. This will increase chances of error, which in return will give wrong output when we query database.
* By using relationship, one can avoid those types of problems.

**Helps increase usability for end users: -**

* By creating a relationship, you can now provide users with a widget for selecting the desired option (a drop down/combo box for example).
* So, instead of having to type the full city name, they simply select the city from the drop down list.
* You can do this easily, by populating the drop down list with the contents of the table.

**Easier data maintenance: -**

Below considering the example, where there are two tables one is customer table and another one is city table and relationship between them is *one Customer can belong to only one City and a City can have multiple Customer (One to Many)*.

* Updates to data only need to be done in one place. For example, if a city’s name changes (yes, it does happen), you can update it once – in the City table.
* You won’t need to update thousands, or even millions, of records that hold city information, because they only store the CityId. And because the CityId stays the same, you won’t even need to look at the Customer table or any other table.

**Helps With Security: -**

* Sensitive data can be stored in a table that has certain privileges applied. The system will decide to which extend a user can access the data in the database, depending upon which user has logged in into the system.

**Helps with Scalability or Expansion of the Database: -**

* Having certain data in a separate table allows you to add records that aren’t necessarily needed now, but may be needed in the future. For example, you could add new cities to the City table even if no other record references it yet.