Project Phase 3

Team 9

I) Conversion of ER model into relational schema

Converting an Entity-Relationship (ER) model into a relational database schema involves systematic steps to ensure that all entities, attributes, and relationships in the ER model are accurately represented in relational terms. Here's a structured summary of the process:

Representing Entities

Strong entities are directly translated into relations. Each simple attribute of the entity becomes a column in the relation, while the entity's primary key is designated as the primary key of the relation. If the entity has composite attributes, their constituent parts are individually represented as columns within the relation. For multi-valued attributes, a new relation is created. This relation includes two attributes: one for storing the primary key of the original strong entity and the other for the multi-valued attribute. Together, these two attributes form the primary key of the new relation.

Weak entities require special handling since they lack a complete primary key. For each weak entity, a new relation is created. This relation includes all the weak entity's simple attributes and the partial key of the weak entity. To establish a connection with the owning strong entity, the primary key of the strong entity is added to this relation as a foreign key. The combination of the weak entity's partial key and the primary key of the strong entity forms the primary key of the weak entity's relation.

Representing Relationships

For 1:N or N:1 relationships, the primary key of the entity on the "1-side" is included as a foreign key in the relation representing the "N-side." This ensures that each entity instance on the "N-side" references a corresponding entity instance on the "1-side."

For **N:M relationships**, an additional relation is created. This new relation includes foreign keys referencing the primary keys of the participating entities. These foreign keys together form the primary key of the new relation. If the relationship has attributes of its own, they are added as columns to this relation.

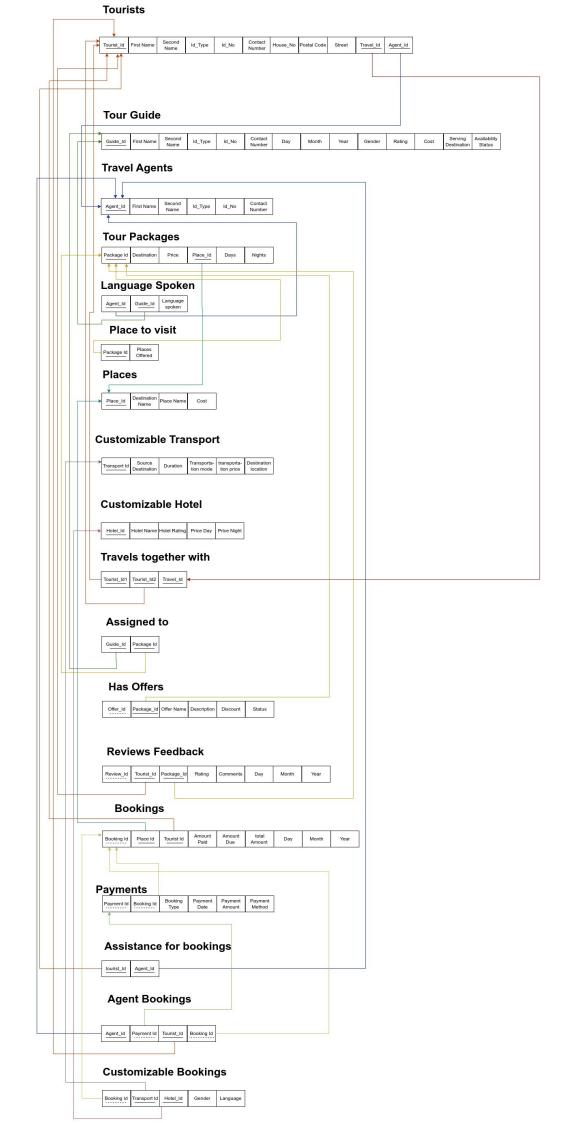


Figure 1: Initial Schema

Normalization

Changes made to schema while converting to 1NF:

we made sure that every attribute has a specific data type, and all attribute values are atomic (indivisible). There are no duplicate or repeating attributes in any table, and each table has a primary key to uniquely identify its records.

Changes made to schema while converting to 2NF:

Some tables, like Payments, Bookings, Offers, Reviews and Feedback, were split into two tables each (e.g., Payments1 and Payments2). This was necessary because certain attributes depended only on part of the original primary key, not the whole key. By creating separate tables for these attributes, we removed this partial dependency.

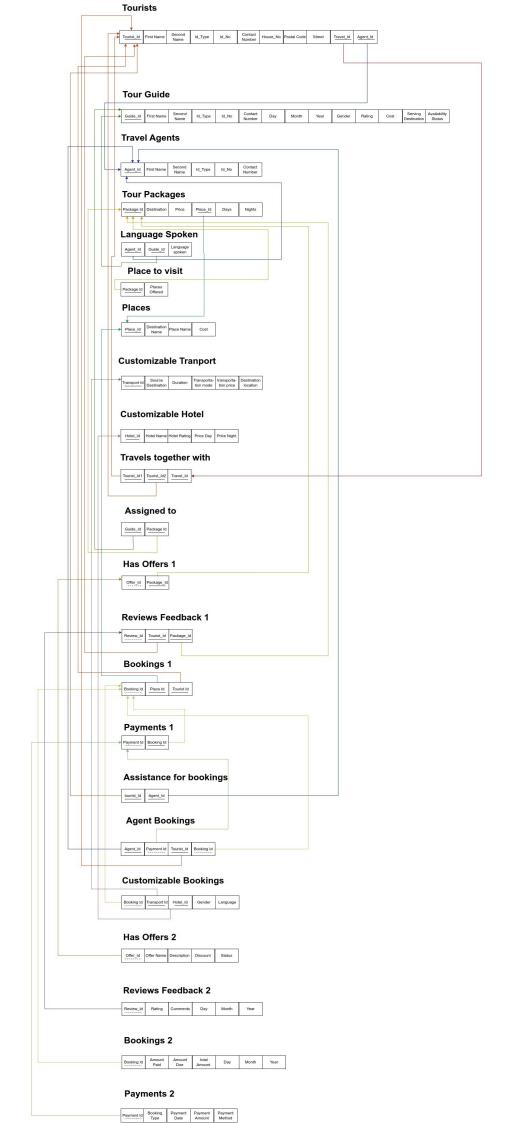


Figure 2: Schema After 2NF

Changes made to schema while converting to 3NF:

No changes were needed for 3NF because the tables were already designed to avoid transitive dependencies. All non-primary attributes depend directly on the primary key, so the database was already well-structured without redundancy