CREATE STATEMENT GENERATOR

Pranit Das

Schema of Structured Data

The schema defines two entities (Entity1 and Entity2) and a relationship between them. Each entity consists of attributes with data types and constraints.

Entity 1:

Attributes are defined dynamically by the user.

Constraints include Key or None.

Entity 2:

Attributes are defined dynamically by the user.

Constraints include Key or None.

Bridge Table:

Attributes include primary and foreign key constraints derived from the entities to establish a bridge between Entity1 and Entity2.

Relation Among Schemas

The relationships among schemas are based on the following principles:

- **Primary and Foreign Key Constraints:**
  - Attributes in `Entity1` and `Entity2` are linked through a relationship table, ensuring referential integrity.
  - The relationship table includes attributes from both entities that act as the linking bridge.
- **Entity-Relationship Design:**
  - Relationships represent connections between `Entity1` and `Entity2`. This ensures a normalized design with well-defined associations.

Cardinalities

The cardinalities between Entity1 and Entity2 are not explicitly mentioned in the code but can be inferred based on the constraints:

One-to-One (1:1):

- If both entities have a unique primary key that corresponds one-to-one.

One-to-Many (1:N):

- If Entity1 has a primary key and Entity2 includes a foreign key referencing it.

Many-to-Many (M:N):

- Implemented via the bridge table, where primary keys from both entities are included as foreign keys.

Normal Forms

The schema is designed to adhere to database normalization principles:

First Normal Form (1NF):

Each entity has atomic values in each attribute.

Implementation

The implementation is done using Python. Key features of the implementation include:

Dynamic Input:

Users can define attributes, data types, and constraints for both entities.

Constraint Validation:

Primary keys and foreign keys are captured for relational integrity.

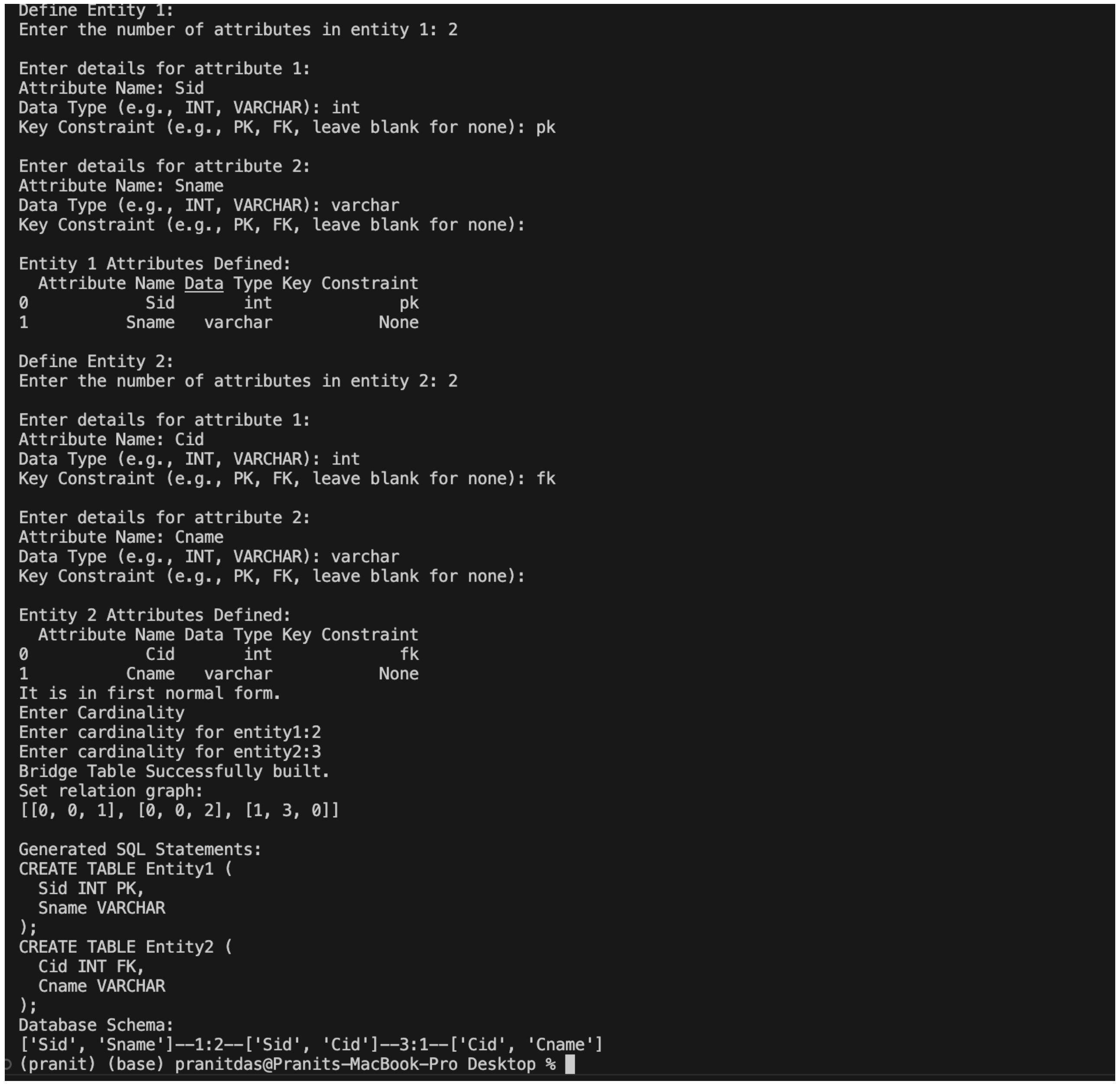
SQL Code Generation:

The system generates SQL CREATE TABLE statements for Entity1, Entity2, and the bridge table.

Establish Relationship Graph

The system establishes directed graph between entities

Output



```
Define Entity 1:
Enter the number of attributes in entity 1: 2

Enter details for attribute 1:
Attribute Name: Sid
Data Type (e.g., INT, VARCHAR): int
Key Constraint (e.g., PK, FK, leave blank for none): pk

Enter details for attribute 2:
Attribute Name: Sname
Data Type (e.g., INT, VARCHAR): varchar
Key Constraint (e.g., PK, FK, leave blank for none):

Entity 1 Attributes Defined:
  Attribute Name Data Type Key Constraint
0            Sid       int             pk
1          Sname   varchar           None

Define Entity 2:
Enter the number of attributes in entity 2: 2

Enter details for attribute 1:
Attribute Name: Cid
Data Type (e.g., INT, VARCHAR): int
Key Constraint (e.g., PK, FK, leave blank for none): fk

Enter details for attribute 2:
Attribute Name: Cname
Data Type (e.g., INT, VARCHAR): varchar
Key Constraint (e.g., PK, FK, leave blank for none):

Entity 2 Attributes Defined:
  Attribute Name Data Type Key Constraint
0            Cid       int             fk
1          Cname   varchar           None
It is in first normal form.
Enter Cardinality
Enter cardinality for entity1:2
Enter cardinality for entity2:3
Bridge Table Successfully built.
Set relation graph:
[[0, 0, 1], [0, 0, 2], [1, 3, 0]]

Generated SQL Statements:
CREATE TABLE Entity1 (
  Sid INT PK,
  Sname VARCHAR
);
CREATE TABLE Entity2 (
  Cid INT FK,
  Cname VARCHAR
);
Database Schema:
['Sid', 'Sname']--1:2--['Sid', 'Cid']--3:1--['Cid', 'Cname']
(pranit) (base) pranitdas@Pranits-MacBook-Pro Desktop %
```