To resolve this issue and allow users to select any source and destination while still showing relevant bus routes, you can create a more flexible database structure. You can use the following tables:

Stops Table:

Stop\_ID (Unique, Not Null)

Stop\_Name

Latitude

Longitude

Routes Table:

Route\_ID (Unique, Not Null)

Route\_Name

Description (Optional)

Route\_Stops Table (to store the sequence of stops for each route):

Route\_Stop\_ID (Unique, Not Null)

Route\_ID (Foreign Key to Routes Table)

Stop\_ID (Foreign Key to Stops Table)

Sequence\_Order (to determine the order of stops on a route)

Buses Table:

Bus\_ID (Unique, Not Null)

Bus\_Number

Bus\_Type

Depo (Depot)

Bus\_Schedule Table:

Schedule\_ID (Unique, Not Null)

Bus\_ID (Foreign Key to Buses Table)

Route\_ID (Foreign Key to Routes Table)

Departure\_Time

Arrival\_Time

Now, when a user selects a source and destination, you can follow these steps to find relevant bus routes:

Find the Stop\_IDs for the selected source and destination stops from the Stops Table.

Query the Route\_Stops Table to find all routes that include both the source and destination stops. This query should check if the Route\_ID contains both the source and destination Stop\_IDs, allowing for flexibility in selecting different source-destination pairs.

Retrieve the Route\_IDs from step 2 and query the Bus\_Schedule Table to find buses that operate on those routes.

This database structure allows you to handle various source-destination combinations and find relevant bus routes that include those stops. Users will be able to see buses operating on the same route, even if their specific source-destination pair isn't explicitly stored in the database.

To design a database for a bus reservation system that allows users to book buses from any of the intermediate halts between the source and destination, you can create a database schema with the following tables:

1. \*\*Bus Table\*\*:

- `bus\_id` (Primary Key)

- `service\_no` (Unique)

- `source\_id` (Foreign Key referencing `BusStops` table)

- `destination\_id` (Foreign Key referencing `BusStops` table)

- `departure\_time`

- `arrival\_time`

- Other relevant bus information

2. \*\*BusStops Table\*\*:

- `stop\_id` (Primary Key)

- `stop\_name`

3. \*\*BusStopsSequence Table\*\*:

- `sequence\_id` (Primary Key)

- `bus\_id` (Foreign Key referencing `Bus` table)

- `stop\_id` (Foreign Key referencing `BusStops` table)

- `sequence\_number` (to define the order of stops for a particular bus)

With this schema, you can represent the bus routes and their stops. Here's how it works:

- The `BusStops` table stores all the bus stops (e.g., Shimla, Solan, Dharampur, Chandigarh).

- The `BusStopsSequence` table links bus routes (identified by `bus\_id`) to the bus stops (identified by `stop\_id`) in the order they occur for each bus. This table allows you to define the sequence of stops for each bus route.

- When a user wants to book a bus, you can query the database to find buses that have the source and destination stops matching their input.

For example, if a user wants to book a bus from Solan to Chandigarh, you can query the database to find all buses that have Solan as a stop and Chandigarh as a stop, regardless of whether Solan is the source or destination. You can use SQL JOIN operations to achieve this.

Here's a simplified SQL query to find buses from Solan to Chandigarh:

```sql

SELECT DISTINCT B.\*

FROM Bus AS B

JOIN BusStopsSequence AS S1 ON B.bus\_id = S1.bus\_id

JOIN BusStopsSequence AS S2 ON B.bus\_id = S2.bus\_id

WHERE S1.stop\_id = 'Solan' AND S2.stop\_id = 'Chandigarh'

```

This query retrieves all buses that stop at Solan and Chandigarh, regardless of their order in the route.

This database design allows for flexibility in booking buses from any intermediate halt between the source and destination without being limited to only booking from the source and destination.