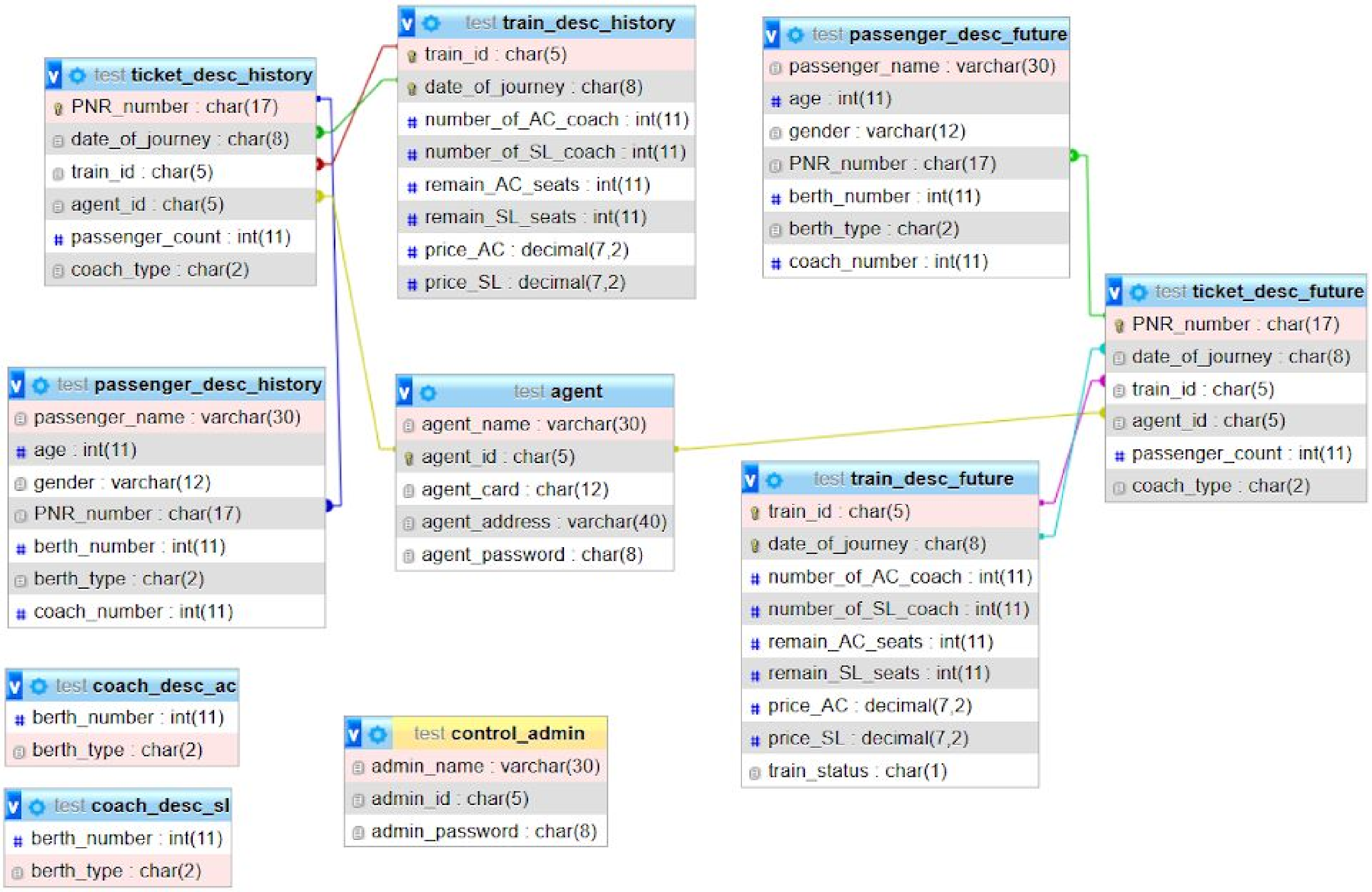
<u>CS301 DBMS Project Report</u>

Abhaysinh Sunil Patil : 2018CSB1063
Kanniganti Nagasrikar : 2018CSB1098
Sarvesh Vhaval : 2018CSB1121

<u>Part 1:</u>

1. <u>Schema</u>
    - All the variables and the attributes of all the tables are shown clearly in the below image.
    - The small key-like icon beside an attribute shows that it is the primary key of the table.
    - And all the connected lines show the foreign key constraint for each table.



2. To update the remaining seats for a particular table, a trigger is created in the passenger list table. After every insert into the passenger list, the remaining seats in that train are decremented by 1.
3. There are two versions for certain tables, namely future and history. The future table consists of the info for each bookable entry, whereas the history table contains the past archives.
4. There is a function made, which will manage the transfer of entries from future table to history table at the end of each day.
5. The admin is responsible for inserting a new train and date, which is to be released in the bookable train list.
6. The admin gives the train_id and the date_of_journey.
7. This would manipulate only the *train_desc_future* table.
8. The booking agent would be making bookings for one or more passengers.
9. He would give the number of passengers and the details of each passenger.
10. First a PNR number would be generated, and a ticket would be created, which is then inserted into the *ticket_desc_future*.
11. And for each passenger, an entry is made into the *passenger_desc_future* table.

Part 2:

1. Schema:
    - We are going to have a table for each station.
    - We will use a method similar to Suffix Tries. We are going to store all the possible destinations a train can reach from each station, without changing the current train.
    - Each table is going to have the following attributes:
        - i. Train ID
        - ii. Departure time (From this station)
        - iii. Destination
        - iv. Arrival time (To the destination station)
2. When the admin adds a train, he/she can add all the stations where the train will stop.
3. The server would take this input and add the destinations to all the relevant station tables. If the train stops at n stations, we will need to update n-1 tables - with the ith table being updated n-i times. Hence, the complexity will be O(N^2).
4. When a user requests for a new route:
    - we check all the destinations possible from this station by iterating through its table.
    - If the destination exists, we output the trains.
    - Else, we go to each of the stations reachable from the origin and look for the destination in their respective tables. If the destination is not found in any of them, we return that no trains are available.