Mehmet Egehan Pala

31089

CS 306 Project Phase 2 Report

Choosing the Tables and Create Statements:

I have picked Ticket and Entitled_baggage.

```
CREATE TABLE ticket(
        ticket_id char(64),
        price FLOAT,
        class CHAR(64),
        primary key (ticket_id)
);
CREATE TABLE Entitled_baggage (
        ticket_id CHAR(64) NOT NULL,
        baggage_id CHAR(64),
        weight INT,
        insurance CHAR(5),
        primary key (baggage_id),
        foreign key (ticket_id) references ticket(ticket_id) ON DELETE CASCADE
);
```

<u>ticket table functional dependency:</u>

F = {ticket_id → price, class}

- This table is in BCNF form because it has a primary key that all other attributes are functionally dependent on.
-

<u>Entitled_baggage functional dependency:</u>

F = {baggage_id → weight, insurance}

- This table is not in BCNF because baggage_id does not cover ticket_id, therefore, there is no key or super key (baggage_id is not a key or super key) in this table which violates the BCNF rules. So, we have to decompose.

Tables after decomposition:

```
CREATE TABLE BaggageInfo (
    baggage_id CHAR(64),
    weight INT,
    insurance CHAR(5),
    primary key (baggage_id)
);
CREATE TABLE BaggageRef
    ticket_id CHAR(64) NOT NULL,
    baggage_id CHAR(64),
    foreign key (baggage_id) references BaggageInfo(baggage_id) ON DELETE CASCADE,
    foreign key (ticket_id) references ticket(ticket_id) ON DELETE CASCADE
);
```

BaggageRef

ticket_id	baggage_id
T123	BAG001
T456	BAG002
T789	BAG003
T012	BAG004
T345	BAG005
T678	BAG006
T901	BAG007
T234	BAG008
T567	BAG009
T890	BAG010

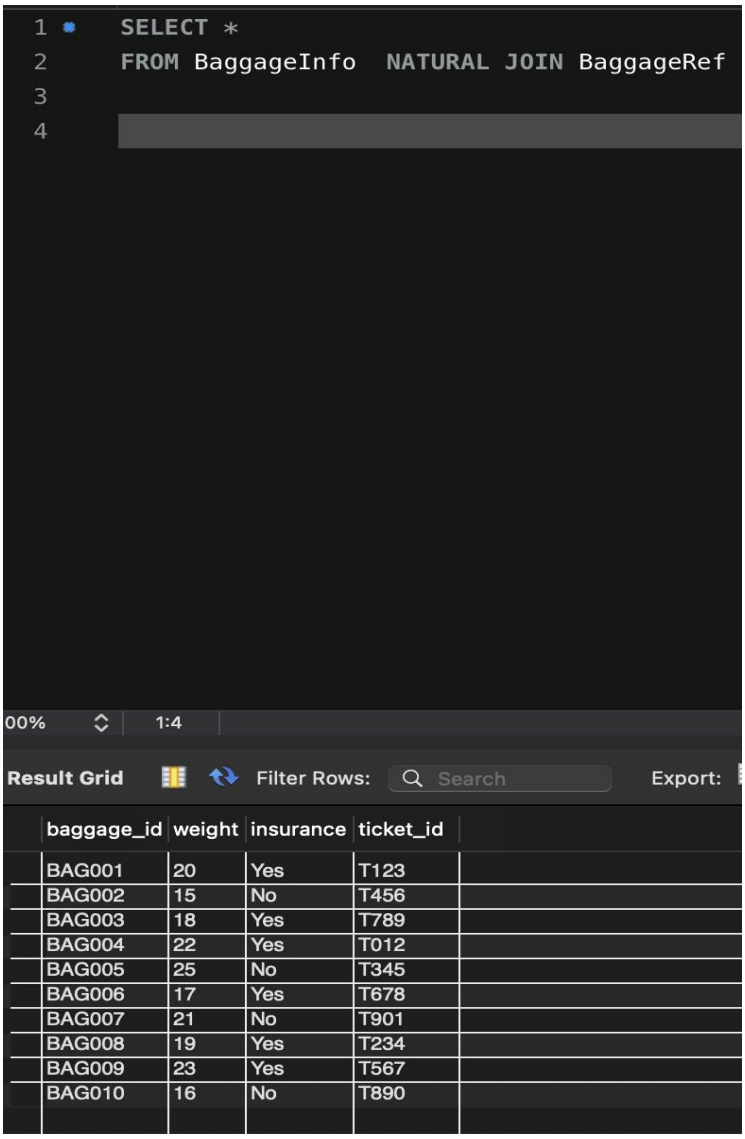
BaggageInfo

baggage_id	weight	insurance
BAG001	20	Yes
BAG002	15	No
BAG003	18	Yes
BAG004	22	Yes
BAG005	25	No
BAG006	17	Yes
BAG007	21	No
BAG008	19	Yes
BAG009	23	Yes
BAG010	16	No

ticket

ticket_id	price	class
T901	270	Business
T890	280	Business
T789	200	Premium Economy
T678	220	Premium Economy
T567	210	Premium Economy
T456	150	Economy
T345	180	Economy
T234	160	Economy
T123	250	Business
T012	300	Business

- The query states: “Project all the columns, which includes baggage_id, weight, insurance, ticket_id, after the natural join implementation.”
- π baggage_id, weight, insurance, ticket_id
  (BaggageInfo $\bowtie$ BaggageInfo .baggage_id = BaggageRef.baggage_id BaggageRef)



baggage_id	weight	insurance	ticket_id
BAG001	20	Yes	T123
BAG002	15	No	T456
BAG003	18	Yes	T789
BAG004	22	Yes	T012
BAG005	25	No	T345
BAG006	17	Yes	T678
BAG007	21	No	T901
BAG008	19	Yes	T234
BAG009	23	Yes	T567
BAG010	16	No	T890

Write down a query which will require "group by" operation in both English and relational algebra:

- Query: This query retrieves the ticket id's values associated with baggage items where the average weight exceeds the overall average weight of baggage items with having insurance.

```sql
SELECT b1.ticket_id
FROM BaggageRef b1 NATURAL JOIN BaggageInfo b2
WHERE EXISTS (
    SELECT b2.weight
    WHERE b1.baggage_id = b2.baggage_id AND b2.insurance = 'Yes'
    GROUP BY b2.weight
    HAVING b2.weight > (
        SELECT AVG(b3.weight)
        FROM BaggageInfo b3
        WHERE b3.insurance = 'Yes'
    )
);
```

100% 6:11

Result Grid Filter Rows: Search Export:

ticket_id
T123
T012
T567

Adding Check Constraint:

- I am adding a check constraint to check whether the baggage weights more 32 kilograms. Since it is not allowed to take baggage which weighs more than it will not allow user to pass.