

School of Computing

Semester 2, 2023

Assignment 2

NORMALISATION AND SQL ASSIGNMENT

Gino Sunny, 47814403

COMP6750: Database Systems

TASK DESCRIPTION AND ANSWERS

1 a. The Entry data table is currently in 2NF (Second Normal Form).

Steps:

The steps below are to show that the table, is not in Unnormalised form (UNF), then to show it is in at least first normal form (1NF), then show it is in second normal form (2NF) then show it is not in third normal form (3NF).

- **Step 1**: show not in UNF In the table, there are no cell data values that hold multiple values. "Given names" could be assumed to have multiple names, but those are given names of 1 person and not multiple collections of given names separated by a comma. Therefore, since it is not in UNF, it is at least in 1NF.
- **Step 2:** show at least in 2NF: (In order to prove that the table is at least in 2NF, we need to identify a primary key and then show that there are no partial dependencies).
 - We need to identify if there is 1 column in the table whose values can uniquely identify each row in the table. In this case, the **entry attempt id** can uniquely identify each row in the table. I.e. if the value of the entry attempt id is known, then the row it is in can be identified.
 - If there wasn't 1 column, then we need to look for pairs of columns, then groups of 3, until a combination that works is found.
 - Since in this case, there is only 1 column in the Primary key, there is no partial dependency (all columns are dependent on the primary key only).
- **Step 3:** show not in 3NF: (In order to prove that the table is not in 3NF, we need to check if there are any columns that are transitively dependent on the PK through another combination of columns.
 - From the table, we could identify that there is a functional dependency between the entry attempt id and other columns.
 - {entry attempt id} -> {staff id, given names, family name, permit reservation id, permit from date, permit to date, permit for number

plate, staff card id, card issue date, parking area id, entry date time, exit date time}

- However, we could identify other functional dependencies.
 - {staff id} -> {given names, family name}
 - {permit reservation id} -> {permit from date, permit to date}
 - {staff card id} -> {card issue date}
 - {permit reservation id, permit for number plate, staff card id} -> {staff id}
- So, it could be technically concluded, that the following transitive dependencies exist:
 - {entry attempt id} -> {staff id} -> {given names, family name}
 - {entry attempt id} -> {permit reservation id} -> {permit from date, permit to date}
 - {entry attempt id} -> {staff card id} -> {card issue date}
 - {entry attempt id} -> {permit reservation id, permit for number plate, staff card id} -> {staff id}
- As transitive dependencies exist through {staff id}, {permit reservation id}, {permit reservation id, permit for number plate, staff card id} and {staff card id}, the table is not in 3NF.

1 b. The steps to convert the table from 2NF into BCNF are as follows:

Steps: Convert the table from 2NF to 3NF (remove transitive dependencies on non-PK columns) and then from 3NF to BCNF (Remove any functional dependencies where the selected columns are not candidate keys and also split the tables up where there is such a column grouping).

- **Step 1:** To convert the table from 2NF to 3NF, the transitive dependencies need to be moved to separate tables. The dependencies identified are as follows.
 - {staff id} -> {given names, family name}
 - {permit reservation id} -> {permit from date, permit to date}
 - {staff card id} -> {card issue date}
 - {permit reservation id, permit for number plate, staff card id} -> {staff id}

- Step 2: Now, the table is in 3NF and to convert the table to BCNF we need to identify the functional dependencies in each of these tables created as part of the normalization process. If dependencies exist, then those columns need to be split up into different tables.
 - Two functional dependencies exist in the table after 3NF normalization and they are as follows.
 - {entry attempt id} -> {permit reservation id, permit for number plate, staff card id}
 - {entry attempt id} -> {parking area id, entry date time, exit date time}
 - (Assumed that, the parking area id, entry date time, exit date time could have the same values in two different rows of the table and it will not be unique ever).
 - The column on the left side of the dependency i.e. the entry attempt id is a super key (not candidate key) and hence the table is now in BCNF.

The final list of tables after the normalization of the Entry data table from 2NF – BCNF are as follows:

- **Staff Card** (staff card id(**PK**), card issue date)
- **Permit Reservation** (permit reservation id(**PK**), permit from date, permit to date)
- **Staff** (staff id(**PK**), given names, family name)
- Entry Attempt (entry attempt id (PK), parking area id, entry date time, exit date time)
- **Staff Entry** (entry attempt id(**PK**, **FK**), staff id(**PK**, **FK**))
- Entry Data (entry attempt id (PK, FK), permit reservation id (PK, FK), staff card id (PK, FK), permit for number plate).

2 a). CREATE statements for the physical model are as follows:

-- Car Entity

```
CREATE TABLE `CAR` (
`numberPlate` VARCHAR(10) NOT NULL,
`carBrand` VARCHAR(45) NOT NULL,
`carModel` VARCHAR(45) NOT NULL,
PRIMARY KEY (`numberPlate`)
);
```

```
-- Swipe Car Entity
```

```
CREATE TABLE `SWIPE CARD` (
`cardId` INT(10) NOT NULL,
`nameOnCard` VARCHAR(100) NOT NULL,
`staffNo` VARCHAR(10),
`contactPhone` VARCHAR(20) NOT NULL,
PRIMARY KEY (`cardId`)
);
-- Spot Reservation Entity
CREATE TABLE `SPOT_RESERVATION` (
`reservationId` INT(10) NOT NULL,
`cardId` INT(10) NOT NULL,
`numberPlate` VARCHAR(10) NOT NULL,
`paymentAmount` DECIMAL(5,2) NOT NULL,
`whenCreated` DATETIME NOT NULL,
PRIMARY KEY (`reservationId`),
CONSTRAINT `spotreservation_ibfk_1` FOREIGN KEY (`numberPlate`) REFERENCES
`CAR` (`numberPlate`),
CONSTRAINT `spotreservation ibfk 2` FOREIGN KEY (`cardId`) REFERENCES
`SWIPE_CARD` (`cardId`)
);
-- Car Park Entity
CREATE TABLE `CAR PARK` (
`carparkId` INT(10) NOT NULL,
`mapReference` VARCHAR(10) NOT NULL,
'description' VARCHAR(200),
PRIMARY KEY ('carparkId')
);
-- Spot Reservation Parking Area Entity
CREATE TABLE `SPOT_RESERVATION_PARKING_AREA` (
`areaId` INT(10) NOT NULL,
`carParkId` INT(10) NOT NULL,
`areaName` VARCHAR(45) NOT NULL,
PRIMARY KEY (`areaId`),
CONSTRAINT `spotreservationparkingarea_ibfk_1` FOREIGN KEY (`carParkId`)
REFERENCES `CAR_PARK` (`carParkId`)
);
```

-- Spot Area Entry Attempt Entity

```
CREATE TABLE 'SPOT AREA ENTRY ATTEMPT' (
`attemptId` INT(10) NOT NULL,
`cardId` INT(10) NOT NULL,
`areaId` INT(10) NOT NULL,
`dateAndTimeOfEntry` DATETIME,
PRIMARY KEY (`attemptId`),
CONSTRAINT `spotareaentryattempt_ibfk_1` FOREIGN KEY (`cardId`) REFERENCES
`SWIPE CARD` (`cardId`),
CONSTRAINT `spotareaentryattempt_ibfk_2` FOREIGN KEY (`areaId`) REFERENCES
`SPOT_RESERVATION_PARKING_AREA` (`areaId`)
);
-- Numbered Parking Spot Entity
CREATE TABLE 'NUMBERED PARKING SPOT' (
`parkingSpotId` INT(10) NOT NULL,
`areaId` INT(10) NOT NULL,
`locationDescription` VARCHAR(100),
PRIMARY KEY (`parkingSpotId`),
CONSTRAINT `numberedparkingspot ibfk 1` FOREIGN KEY (`areaId`) REFERENCES
`SPOT RESERVATION PARKING AREA` (`areaId`)
);
-- Time Slot Entity
CREATE TABLE 'TIME SLOT' (
'year' YEAR(4) NOT NULL,
`semester` CHAR(2) NOT NULL,
PRIMARY KEY ('year', 'semester')
);
-- Allocation Entity
CREATE TABLE `ALLOCATION` (
`parkingSpotId` INT(10) NOT NULL,
`timeSlot_Year` YEAR(4) NOT NULL,
`timeSlot Semester` CHAR(2) NOT NULL,
`spotReservation reservationId` INT(10) NOT NULL,
PRIMARY KEY ( `parkingSpotId`, `timeSlot_Year`, `timeSlot_Semester`),
CONSTRAINT `allocation_ibfk_1` FOREIGN KEY (`spotReservation_reservationId`)
REFERENCES `SPOT_RESERVATION` (`reservationId`),
CONSTRAINT `allocation_ibfk_2` FOREIGN KEY (`parkingSpotId`) REFERENCES
`NUMBERED PARKING SPOT` (`parkingSpotId`),
```

```
CONSTRAINT `allocation_ibfk_3` FOREIGN KEY (`timeSlot_Year`, `timeSlot_Semester` )
REFERENCES `TIME_SLOT` (`year`, `semester` )
);
```

2 b). INSERT statements to add data the tables created above are as follows:

-- Car Entity

```
INSERT INTO CAR (numberPlate, carBrand, carModel)
VALUES

('DEF456', 'Ford', 'Fusion'),
('GHI789', 'Chevrolet', 'Malibu'),
('JKL321', 'Nissan', 'Altima'),
('XLR123', 'BMW', 'X'),
('ABC123', 'Toyota', 'Camry'),
('AOX123', 'Kia', 'Cruz'),
('XOI788', 'Honda', 'Wrv'),
('XYZ789', 'Honda', 'Civic');
```

-- Swipe Card Entity

```
INSERT INTO SWIPE_CARD (cardId, nameOnCard, staffNo, contactPhone) VALUES
```

```
(16683, 'Eva Williams', 'EMP004', '555-789-1234'), (22936, 'Bob Johnson', 'EMP003', '555-555-5555'), (36715, 'Bob Johnson', 'EMP003', '555-555-5555'), (37070, 'John Doe', 'EMP001', '555-123-4567'), (37267, 'Alice Smith', 'EMP002', '555-987-6543'), (42584, 'John Doe', 'EMP001', '555-123-4567'), (53007, 'Alice Smith', 'EMP002', '555-987-6543'), (63831, 'Mike Brown', 'EMP005', '555-321-5678'), (79862, 'Eva Williams', 'EMP004', '555-321-5678'), (80125, 'Mike Brown', 'EMP005', '555-321-5678'), (80126, 'Mike Brown', 'EMP005', '555-321-5678');
```

-- Spot Reservation Entity

INSERT INTO SPOT_RESERVATION (reservationId, cardId, numberPlate, paymentAmount, whenCreated)

```
VALUES
```

```
(171134, 79862, 'DEF456', 14.2, '2023-08-13 09:15'), (267892, 53007, 'XYZ789', 12.75, '2023-09-16 09:15'), (298415, 36715, 'DEF456', 22.3, '2023-08-16 09:15'), (313815, 37267, 'AOX123', 16.5, '2023-09-10 11:30'), (407153, 37070, 'ABC123', 15.5, '2023-08-21 08:30'), (456789, 36715, 'XLR123', 18.25, '2023-07-11 10:00'), (474056, 37070, 'ABC123', 15.3, '2023-09-15 12:30'), (563787, 80125, 'JKL321', 14.9, '2023-09-11 12:30'), (649697, 16683, 'GHI789', 14.5, '2023-08-14 09:15'), (669807, 16683, 'GHI789', 18.25, '2023-08-02 11:45'),
```

```
(803735, 16683, 'XOI788', 20, '2023-07-15 10:00'), (803794, 16683, 'GHI789', 20, '2023-07-15 10:00');
```

-- Car Park Entity

INSERT INTO CAR_PARK (carparkId, mapReference, description) VALUES

- (1, 'O1', 'Open area parking level 1'),
- (2, 'G1', 'Gated area parking level 1'),
- (3, 'O2', 'Open area parking level 2'),
- (4, 'G2', 'Gated area parking level 2'),
- (5, 'G3', 'Gated area parking level 3');

-- Spot Reservation Parking Area Entity

INSERT INTO SPOT_RESERVATION_PARKING_AREA (areaId, carParkId, areaName) VALUES

- (11, 1, 'Main Parking Area'),
- (22, 1, 'Visitor Parking'),
- (33, 2, 'Upper Level Parking'),
- (44, 2, 'Employee Parking'),
- (55, 3, 'Reserved Parking Area'),
- (66, 3, 'Reserved Parking L2'),
- (77, 4, 'Underground Level 1'),
- (88, 5, 'Underground Level 2');

-- Spot Area Entry Attempt Entity

 $INSERT\ INTO\ SPOT_AREA_ENTRY_ATTEMPT\ (attemptId,\ cardId,\ areaId,\ dateAndTimeOfEntry)$

VALUES

- (1, 42584, 11, '2023-09-21 08:30'),
- (2, 53007, 22, '2023-09-21 09:15'),
- (3, 63831, 33, '2023-09-21 10:00'),
- (4, 16683, 44, '2023-09-21 11:45'),
- (5, 80125, 55, '2023-09-21 12:30'),
- (6, 42584, 66, '2023-09-22 12:30'),
- (7, 42584, 11, '2023-09-20 12:30');

-- Numbered Parking Spot Entity

INSERT INTO NUMBERED_PARKING_SPOT (parkingSpotId, areaId, locationDescription) VALUES

- (70, 11, 'Lower Parking A1'),
- (153, 11, 'Near Entrance'),
- (554, 33, 'Parking 3 Level 2'),
- (555, 33, 'Parking 3 Level 1'),
- (588, 22, 'Upper Level Spot 1'),
- (665, 11, 'Parking 1 Level 2'),
- (666, 44, 'Parking 1 Level 1'),

```
(777, 55, 'Employee Parking 2C'),
(778, 66, 'Employee Parking 2D'),
(779, 77, 'Employee Parking 2E'),
(780, 88, 'Employee Parking 2F'),
(862, 33, 'Reserved Spot C3'),
(894, 44, 'Employee Parking 2B');
```

-- Time Slot Entity

```
INSERT INTO TIME_SLOT (year, semester)
VALUES
(2021, 'SU'),
(2022, 'SU'),
(2022, 'WI'),
(2023, 'SU'),
(2023, 'WI');
```

-- Allocation Entity

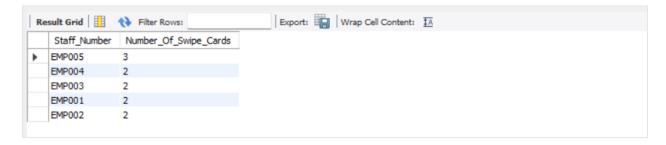
```
INSERT INTO ALLOCATION (parkingSpotId, timeSlot_Year, timeSlot_Semester, spotReservation_reservationId) VALUES
```

```
(70, 2022, 'WI', 267892),
(153, 2023, 'SU', 407153),
(862, 2023, 'SU', 563787),
(894, 2022, 'SU', 669807),
(588, 2023, 'WI', 803794);
```

<u>3a).</u>

SELECT staffNo, COUNT(cardId) AS NumberOfSwipeCards FROM SWIPE_CARD GROUP BY staffNo ORDER BY NumberOfSwipeCards DESC;

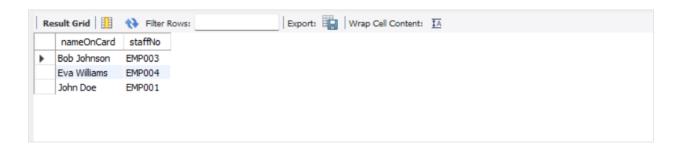
Result:



<u>3b).</u>

```
SELECT DISTINCT sc.nameOnCard, sc.staffNo FROM SWIPE_CARD AS sc
WHERE sc.cardId IN (
SELECT sr.cardId FROM SPOT_RESERVATION AS sr
GROUP BY sr.cardId
HAVING COUNT(sr.reservationId) >= 2
)
ORDER BY sc.nameOnCard asc;
```

Result:



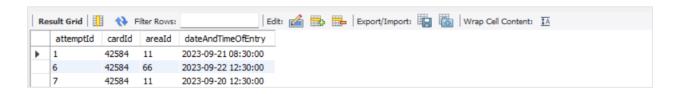
3c).

```
-- Test with card id = 42584

SELECT * FROM SPOT_AREA_ENTRY_ATTEMPT

WHERE cardId = 42584;
```

Result:



3d).

```
SELECT cp.carparkId, cp.description, cp.mapReference, COUNT(nps.parkingSpotId) as TotalNumberedParkingSpots
FROM CAR_PARK as cp
JOIN SPOT_RESERVATION_PARKING_AREA as srpa
```

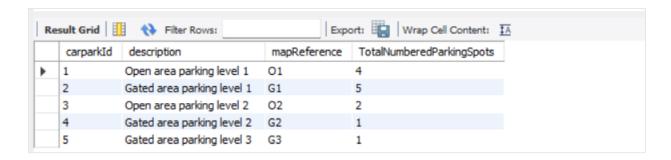
ON cp.carparkId = srpa.carParkId

JOIN NUMBERED_PARKING_SPOT as nps

ON nps.areaId = srpa.areaId

GROUP BY cp.carparkId, cp.description, cp.mapReference;

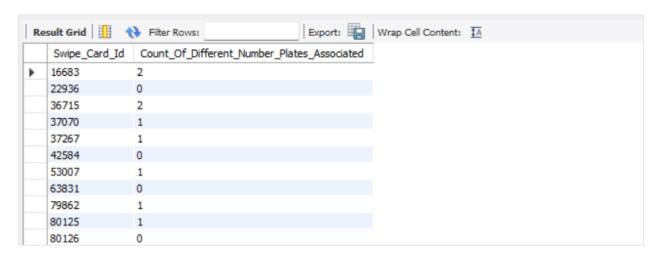
Result:



<u>3e).</u>

SELECT distinct sc.cardid as Swipe_Card_Id, count(distinct sr.numberplate) as Count_Of_Different_Number_Plates_Associated FROM SWIPE_CARD as sc left join SPOT_RESERVATION as sr on sc.cardid = sr.cardid left join CAR as cr on sr.numberplate = cr.numberplate group by sc.cardid;

Result:



3f).

```
select distinct spra.carparkid, count(nps.parkingspotId)
as NotAllocatedspots
from SPOT_RESERVATION_PARKING_AREA as spra
JOIN

NUMBERED_PARKING_SPOT as nps on
spra.areaid = nps.areaid
where

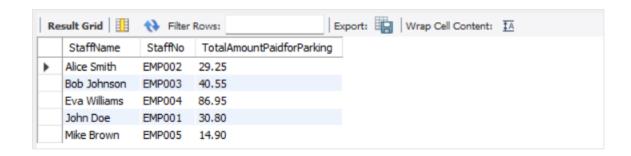
nps.parkingspotId not in (
select aa.parkingspotid from ALLOCATION as aa where
aa.timeslot_year = '2023' and aa.timeslot_semester = 'SU'
)
group by spra.carparkid;
```

Result:



3g).

Result:



<u>3h).</u>

SELECT cp.carparkId as CarPark, aa.timeSlot_Year as Year, SUM(sp.paymentAmount) as TotalAmountGenerated

FROM CAR_PARK cp

JOIN SPOT_RESERVATION_PARKING_AREA srp

ON cp.carparkId = srp.carParkId

JOIN NUMBERED PARKING SPOT np

ON srp.areaId = np.areaId

JOIN ALLOCATION aa

ON np.parkingSpotId = aa.parkingSpotId

JOIN SPOT RESERVATION sp

ON aa.spotReservation_reservationId = sp.reservationId

group by cp.carparkId, aa.timeSlot_Year;

Result:

