HOTEL DATABASE SYSTEMS & DESIGN

By

Michael Kolawole-Idowu

Relational Schema

7 key tables of the database system created from the relational schema:

CUSTOMER (cusID, cusFName, cusLName, last4Digits, cardHolderName, address)
BOOKINGS (bookingID, cusID, checkInDate, checkOutDate, empId)
EMPLOYEE (empId, empFName, empLName, phoneNo)
GUEST (guestsID, Fname, Lname, DOB)
ROOM (RoomNo, RoomType)
ROOM TYPE (Room Type, Room Rate)
BOOKING DETAILS (BookingDetailsNO, bookingID, guestID, RoomNo)

Create Table Commands

Create tables code can be found in the attached submission file "CreateTables.xml".

The previously completed normalisation process of the initial schema led to a streamlined table creation process. Through normalisation the tables created allowed for a system which works under condition without duplications or inconsistencies.

When first creating each table, I had to be cautious initially to ensure that the order of creation didn't provide a conflict. Each table was created in order to allow tables with conflicting foreign keys to be created after the independent table had been created first, for example Room Type was created first and then the room table was created after as the room Table depends on room types.

Order of creation:

1) Room Type
2) Room
3) Employee
4) Customer
5) Guest
6) Bookings
7) Booking Details

Examples of created tables:

2 individual tables (Customer and Employee) which were required to be created before the BookingDetails table which used information from both.

CUS_ID	CUS_FNAME	CUS_LNAME	LAST4DIGITS	CARDHOLDERNAME	ADDRESS
1	FAITH	VINCENTS	1291	MISS F VINCENTS	22 MELROCK ROAD, SW16 2LP
2	JAMES	KAY	0091	MR M KAY	13 MISS BOW ROAD, SE1 10P
3	LEANNE	VINCENT	1918	MISS L VINCENT	49 ROSSDALE CLOSE, N17 5YG
4	KAREEM	USMAN	2115	MR K BOSSMAN	201 DISNEY WAY, CV1 2TF
5	SARAH	BANKS	2098	MRS S BANKS	13A BIRMS WALK, E1 6GF
6	RYAN	CLARKE	1314	MR R CLARKE	35 WELLER STREET, NW2 4BV
7	JACK	LENO	8954	MR J LENO	302 BETHNAL WAY, N1 4GH

Figure. Customer Table

EMP_NO	EMP_FNAME	EMP_LNAME	PHONENO
1	JEFF	RICHARDS	+44 73449993
2	MILES	MARTINS	+44 72727772
3	CHRIS	PETERS	+44 78193743
4	TONY	HALLOWAY	+44 7211177

Figure. Employee table

BOOKING_ID	CUS_ID	CHECKIN	CHECKOUT	EMP_NO
1	1	10-JAN-19	12-JAN-19	4
2	1	01-MAR-19	14-MAR-19	1
3	2	28-MAY-19	03-JUN-19	2
4	3	01-OCT-19	14-OCT-19	3
5	4	10-JAN-19	12-JAN-19	4
6	5	01-MAR-19	14-MAR-19	1
7	6	28-MAY-19	03-JUN-19	2
8	7	21-OCT-19	28-OCT-19	3
9	6	08-OCT-19	14-OCT-19	3

Figure. BookingDetails table

Insertion of Test Data

Insert data SQL code can be found in the attached submission file "Insert.xml"

Sample data was created with the goal of allowing a variety of situations to be tested via queries. I focused my data to reflect some real-life scenarios, for example I had Guests with multiple bookings, Customers who have made a booking but not stayed, a varied age range etc.

The Primary keys for the Booking, Guest, Customer, Booking Details and employee Tables are all constantly growing, as a result I added a sequence generator for these columns. This allows my tables to be incremented automatically, and lessens administrative workload.

Examples of Insert Data:

1) Customers 1 and 6 have created multiple bookings and so have different Booking ID's but the same Customer I.

BOOKING_ID	CUS_ID	CHECKIN	CHECKOUT	EMP_NO
1	1	10-JAN-19	12-JAN-19	4
2	1	01-MAR-19	14-MAR-19	1
3	2	28-MAY-19	03-JUN-19	2
4	3	01-OCT-19	14-OCT-19	3
5	4	10-JAN-19	12-JAN-19	4
6	5	01-MAR-19	14-MAR-19	1
7	6	28-MAY-19	03-JUN-19	2
8	7	21-OCT-19	28-OCT-19	3
9	6	08-OCT-19	14-OCT-19	3

Figure. Bookings table showing variation on bookings

2) Variations of Adult and Child Guests such as Bria Clarke who was born in 1989 and Eve Mathews who was born in 2006

GUEST_ID	GUEST_FNAME	GUEST_LNAME	DOB
1	FAITH	VINCENTS	03-FEB-01
2	BRIA	CLARKE	17-MAY-89
3	SARAH	BANKS	11-AUG-99
4	JOSH	REYNOLDS	31-DEC-72
5	AARON	HARDY	26-JUN-67
6	MARIE	STEVENS	09-JAN-94
7	EVE	MATTHEWS	13-FEB-06
8	RYAN	CLARKE	12-JUN-07
9	ALEX	SAKA	01-MAY-61
10	PAUL	BECKHAM	07-DEC-83
11	DAVID	OWEN	26-AUG-01
12	KEVIN	KEANE	07-FEB-95
13	CHARLOTTE	BORTHWICK	13-FEB-06
14	LEONIE	HAMMOND	17-MAY-89
15	LUCY	DUMMER	11-SEP-89
16	SUE	CARTWRIGHT	21-JUL-72

Figure. Bookings tables showing variation on bookings

Customer Kareem Usman has created a booking (so has a unique Customer Id) but did not stay at the hotel (does not have an entry in the Guest table)

CUS_ID	CUS_FNAME	CUS_LNAME	LAST4DIGITS	CARDHOLDERNAME	ADDRESS
1	FAITH	VINCENTS	1291	MISS F VINCENTS	22 MELROCK ROAD, SW16 2LP
2	JAMES	KAY	0091	MR M KAY	13 MISS BOW ROAD, SE1 10P
3	LEANNE	VINCENT	1918	MISS L VINCENT	49 ROSSDALE CLOSE, N17 5YG
4	KAREEM	USMAN	2115	MR K BOSSMAN	201 DISNEY WAY, CV1 2TF
5	SARAH	BANKS	2098	MRS S BANKS	13A BIRMS WALK, E1 6GF
6	RYAN	CLARKE	1314	MR R CLARKE	35 WELLER STREET, NW2 4BV
7	JACK	LENO	8954	MR J LENO	302 BETHNAL WAY, N1 4GH

GUEST_ID	GUEST_FNAME	GUEST_LNAME	DOB
1	FAITH	VINCENTS	03-FEB-01
2	BRIA	CLARKE	17-MAY-89
3	SARAH	BANKS	11-AUG-99
4	JOSH	REYNOLDS	31-DEC-72
5	AARON	HARDY	26-JUN-67
6	MARIE	STEVENS	09-JAN-94
7	EVE	MATTHEWS	13-FEB-06
8	RYAN	CLARKE	12-JUN-07
9	ALEX	SAKA	01-MAY-61
10	PAUL	BECKHAM	07-DEC-83
11	DAVID	OWEN	26-AUG-01
12	KEVIN	KEANE	07-FEB-95
13	CHARLOTTE	BORTHWICK	13-FEB-06
14	LEONIE	HAMMOND	17-MAY-89
15	LUCY	DUMMER	11-SEP-89
16	SUE	CARTWRIGHT	21-JUL-72

Figure. Highlighting the absence of Kareem Usman from the Guest Table

Relational Database Views

Relational Database Views SQL code can be found in the attached submission file "Views.doc"

The creation of Relational Database Views assists end users to easily retrieve frequently used data and to access important stored data sets without requiring complex queries. It also helps in creating further complex queries, as complex formulas require multiple combinations and storage. My Views again were selected to solve or display real-life scenarios a Hotel may encounter.

1) **Booking Length View -**
   A view was created to show the length of stay for each booking and room stayed at the hotel. This view was useful to create multiple queries and views, as well as to create equations such as, average stay time, most popular rooms etc.

BOOKING_ID	ROOM_NO	DATEDIFF
1	305	2
2	199	13
3	219	6
4	222	13
5	337	2
6	111	13
7	267	6
8	375	7
9	261	6

Figure. "Days Stayed" Relational Database View

2) **Extended Booking Details View -**
   I created a View for all extended information attached to Guests and their booking. This view would be very useful for hotel staff employees, as it quickly and easily shows which guests stayed at the hotel, how often and what days they stayed.

GUEST_ID	GUEST_FNAME	GUEST_LNAME	BOOKING_ID	CHECKIN	CHECKOUT
1	FAITH	VINCENTS	1	10-Jan-19	12-Jan-19
1	FAITH	VINCENTS	2	01-Mar-19	14-Mar-19
1	FAITH	VINCENTS	4	01-Oct-19	14-Oct-19
3	SARAH	BANKS	4	01-Oct-19	14-Oct-19
4	JOSH	REYNOLDS	1	10-Jan-19	12-Jan-19
4	JOSH	REYNOLDS	9	08-Oct-19	14-Oct-19
5	AARON	HARDY	9	08-Oct-19	14-Oct-19
6	MARIE	STEVENS	8	21-Oct-19	28-Oct-19
7	EVE	MATTHEWS	8	21-Oct-19	28-Oct-19
8	RYAN	CLARKE	8	21-Oct-19	28-Oct-19
9	ALEX	SAKA	7	28-May-19	03-Jun-19
10	PAUL	BECKHAM	1	10-Jan-19	12-Jan-19
11	DAVID	OWEN	3	28-May-19	03-Jun-19
12	KEVIN	KEANE	3	28-May-19	03-Jun-19
13	CHARLOTTE	BORTHWICK	5	10-Jan-19	12-Jan-19
14	LEONIE	HAMMOND	2	01-Mar-19	14-Mar-19
14	LEONIE	HAMMOND	9	08-Oct-19	14-Oct-19
15	LUCY	DUMMER	4	01-Oct-19	14-Oct-19
17	DAN	BROWN	9	08-Oct-19	14-Oct-19
18	ALEX	MICHAELS	6	01-Mar-19	14-Mar-19
19	JACK	LENO	7	28-May-19	03-Jun-19

Figure. Created View “Extended Booking Details”

3) Pricing View

To calculate the pricing of each booking, this view creates and combines many other views together as the price is made up of many components. Firstly, the base price (retrieved from the room type) was added to by the number of adult and/or children staying in the room multiplied by the number of days spent in the room:

Therefore, the pricing View required the creation of several separate database views to build off of.

Two views were then created to highlight the adult and child guests staying at the hotel.

BOOKING_ID	GUEST_FNAME	GUEST_LNAME	ROOM_NO	CHECKIN	CHECKOUT	DOB
5	CHARLOTTE	BORTHWICK	337	10-JAN-19	12-JAN-19	13-FEB-06
7	JACK	LENO	267	28-MAY-19	03-JUN-19	13-JUN-11
8	EVE	MATTHEWS	375	21-OCT-19	28-OCT-19	13-FEB-06
8	RYAN	CLARKE	375	21-OCT-19	28-OCT-19	12-JUN-07

Figure. “Child Guests” Relational Database View

BOOKING_ID	GUEST_FNAME	GUEST_LNAME	ROOM_NO	CHECKIN	CHECKOUT	DOB
1	PAUL	BECKHAM	305	10-JAN-19	12-JAN-19	07-DEC-83
1	FAITH	VINCENTS	305	10-JAN-19	12-JAN-19	03-FEB-01
1	JOSH	REYNOLDS	305	10-JAN-19	12-JAN-19	31-DEC-72
2	FAITH	VINCENTS	199	01-MAR-19	14-MAR-19	03-FEB-01
2	LEONIE	HAMMOND	199	01-MAR-19	14-MAR-19	17-MAY-89
3	KEVIN	KEANE	219	28-MAY-19	03-JUN-19	07-FEB-95
3	DAVID	OWEN	219	28-MAY-19	03-JUN-19	26-AUG-01
4	SARAH	BANKS	222	01-OCT-19	14-OCT-19	11-AUG-99
4	FAITH	VINCENTS	222	01-OCT-19	14-OCT-19	03-FEB-01
4	LUCY	DUMMER	222	01-OCT-19	14-OCT-19	11-SEP-89
6	ALEX	MICHAELS	111	01-MAR-19	14-MAR-19	10-MAR-92
7	ALEX	SAKA	267	28-MAY-19	03-JUN-19	01-MAY-61
8	MARIE	STEVENS	375	21-OCT-19	28-OCT-19	09-JAN-94
9	AARON	HARDY	261	08-OCT-19	14-OCT-19	26-JUN-67
9	LEONIE	HAMMOND	261	08-OCT-19	14-OCT-19	17-MAY-89
9	JOSH	REYNOLDS	261	08-OCT-19	14-OCT-19	31-DEC-72

Figure. “Adult Guests’’ Relational Database View

Two count views were created to track the number of adults or children attached to a booking

BOOKING_ID	ADULT_BOOKING_COUNT
1	3
2	2
3	2
4	3
6	1
7	1
8	1
9	4

Figure. “Pricing’’ Relational Database View

BOOKING_ID	CHILD_BOOKING_COUNT
5	1
7	1
8	2

Figure. “Child Count” Relational Database View

A combination count was then created to show both adults and children per booking. This View was then used to develop the formula for the final Pricing view creation.

BOOKING_ID	ROOM_NO	DATEDIFF	ADULT_BOOKING_ID	ADULT_BOOKING_COUNT	CHILD_BOOKING_ID	CHILD_BOOKING_COUNT
1	305	2	1	3	–	–
2	199	13	2	2	–	–
3	219	6	3	2	–	–
4	222	13	4	3	–	–
5	337	2	–	–	5	1
6	111	13	6	1	–	–
7	267	6	7	1	7	1
8	375	7	8	1	8	2
9	261	6	9	4	–	–

Figure. “Combo Count’ Relational Database View

BOOKING_ID	PRICE
1	240
2	1820
3	1080
4	2080
5	300
6	1560
7	1020
8	700
9	1080

Figure. “Pricing’’ Relational Database View

Database Queries

SQL code to run Queries can be found in the attached submission file "Queries.doc"

To understand reliability, complex queries must be used to test the consistency of a system.

1. Display all guests and the room they stayed in, ordered by check in date

```
SELECT GUEST.GUEST_FNAME, GUEST.GUEST_LNAME, ROOM_NO, BOOKINGS.CHECKIN,
BOOKINGS.CHECKOUT
FROM BOOKINGDETAIL
INNER JOIN GUEST ON GUEST.GUEST_ID = BOOKINGDETAIL.GUEST_ID
INNER JOIN BOOKINGS ON BOOKINGS.BOOKING_ID = BOOKINGDETAIL.BOOKING_ID
order by BOOKINGS.CHECKIN;
```

2. List all the Guests older than 18

```
SELECT GUEST_FNAME, GUEST.GUEST_LNAME, DOB
FROM GUEST
where DOB < '13-DEC-2001';
```

3. Increase room rates by 10%

```
UPDATE ROOMTYPE
SET ROOM_RATE = ROOM_RATE * 1.1;
```

4. List the Guests by youngest to oldest

```
SELECT GUEST_FNAME, GUEST_LNAME, DOB
FROM GUEST
ORDER BY DOB DESC;
```

5. Display how many guests per booking

```
SELECT BOOKING_ID, COUNT(BOOKING_ID) as NumberofGuest
FROM BOOKINGDETAIL
INNER JOIN GUEST ON GUEST.GUEST_ID = BOOKINGDETAIL.GUEST_ID
Group by BOOKING_ID
order by BOOKING_ID;
```

6. List all hotel rooms that have not had a Booking

```
SELECT DISTINCT ROOM.ROOM_NO
FROM ROOM
FULL JOIN BOOKINGDETAIL ON ROOM.ROOM_NO = BOOKINGDETAIL.ROOM_NO
WHERE NVL(BOOKINGDETAIL.BOOKING_ID, 0) = 0;
```

7. Display all the Guests checked in by Miles

```
SELECT BOOKINGDETAIL.GUEST_ID, GUEST.GUEST_FNAME, GUEST.GUEST_LNAME,
BOOKINGS.BOOKING_ID, BOOKINGS.CHECKIN, BOOKINGS.CHECKOUT
FROM BOOKINGDETAIL
INNER JOIN BOOKINGS ON BOOKINGS.BOOKING_ID = BOOKINGDETAIL.BOOKING_ID
```

```
INNER JOIN GUEST ON GUEST.GUEST_ID = BOOKINGDETAIL.GUEST_ID
WHERE EMP_NO IN (SELECT EMP_NO FROM BOOKINGS WHERE EMP_NO = 2);
```

8. Which guests stayed longer than the average stay time (Display Guest ID & days stayed)

```
SELECT VIEW_DAYS_STAYED.BOOKING_ID, BOOKINGDETAIL.GUEST_ID,
VIEW_DAYS_STAYED.DATEDIFF
FROM VIEW_DAYS_STAYED
INNER JOIN BOOKINGDETAIL ON VIEW_DAYS_STAYED.BOOKING_ID =
BOOKINGDETAIL.BOOKING_ID
where DATEDIFF > (select AVG(DATEDIFF) from VIEW_DAYS_STAYED)
ORDER BY VIEW_DAYS_STAYED.BOOKING_ID;
```

9. Which Guests spent the most money (Display Guest ID & Price)

```
Select BOOKINGDETAIL.BOOKING_ID, BOOKINGDETAIL.GUEST_ID, PRICING.PRICE
from PRICING
INNER JOIN BOOKINGDETAIL ON PRICING.BOOKING_ID = BOOKINGDETAIL.BOOKING_ID
where PRICE = (select max(PRICE) from PRICING);
```

10. List all guests who stayed between 1-JAN-2019 dates and 31-MAR-2019. Display Guest name, Room stayed and check in/out date.

```
SELECT GUEST.GUEST_FNAME, GUEST.GUEST_LNAME, ROOM_NO, BOOKINGS.CHECKIN,
BOOKINGS.CHECKOUT
FROM BOOKINGDETAIL
INNER JOIN GUEST ON GUEST.GUEST_ID = BOOKINGDETAIL.GUEST_ID
INNER JOIN BOOKINGS ON BOOKINGS.BOOKING_ID = BOOKINGDETAIL.BOOKING_ID
WHERE CHECKIN >= '01-JAN-19'AND CHECKOUT<= '31-MAR-19'
ORDER BY CHECKIN;
```

11. List which Months had bookings and the amount of bookings per month -

```
SELECT DISTINCT to_char(BOOKINGS.CHECKIN, 'MM') AS MONTHS_BOOKED,
COUNT(BOOKINGDETAIL.BOOKING_ID) AS AMOUNT_OF_BOOKINGS
FROM BOOKINGDETAIL
INNER JOIN GUEST ON GUEST.GUEST_ID = BOOKINGDETAIL.GUEST_ID
INNER JOIN BOOKINGS ON BOOKINGS.BOOKING_ID = BOOKINGDETAIL.BOOKING_ID
GROUP BY to_char(BOOKINGS.CHECKIN, 'MM')
ORDER BY to_char(BOOKINGS.CHECKIN, 'MM');
```

12. List all guests that stayed in a Deluxe suite -

```
SELECT BOOKINGDETAIL.GUEST_ID, GUEST.GUEST_FNAME, GUEST.GUEST_LNAME,
BOOKING_ID, ROOM.ROOM_NO, ROOM.ROOM_TYPE
FROM BOOKINGDETAIL
INNER JOIN GUEST ON GUEST.GUEST_ID = BOOKINGDETAIL.GUEST_ID
INNER JOIN ROOM ON ROOM.ROOM_NO = BOOKINGDETAIL.ROOM_NO
WHERE ROOM_TYPE='DELUXE';
```