Shannon Imbo - 1303744

Assignment 1: Building and Design of an environment

Table of Contents

Brief Business Scenario and Business Rules

Business Scenario

Hotel Stay Longer consists of multiple different buildings found in different cities. They own 14 different branches throughout New Zealand. Each hotel building is the same design of having 9 floors with 150 rooms in total. Each room have different types ranging from Suits, Singles, Doubles, and Twins. The hotel takes in about 1000 reservations per day with 60% of that being repeat customers. The hotel now needs a reliable database to store all their information in regarding their customer details, reservations, invoices, rooms, and branches.

Business Rules

1. Reservations can be cancelled.
2. Any change to existing reservations will create a new reservation.
3. Customer addresses are updated regularly.
4. Customer and reservation data are not deleted by the hotel but archived instead.
5. Invoices are delivered at the end of a customer's stay.
6. Customer names are needed to invoice a departing customer.
7. Customers are charged depending on how many days they stay.
8. Balance can only be calculated or updated when the customer is checking out.
9. Invoices from customers will be deleted after paying their balance.

Assumptions

1. There will be minimal changes towards Branches and Rooms therefore it will be assumed that there will be no need to update these tables as well.

Project Introduction:

From here, the first part to be discussed are the tables of the Room Booking Database. Explanations about the tables will be displayed to show and justify each decision made on them. It will start with a quick summary of the ER diagram to show relationships between each table and their attributes.

Shannon Imbo - 1303744

Table Definitions

ER Diagram of Complete System Proposal

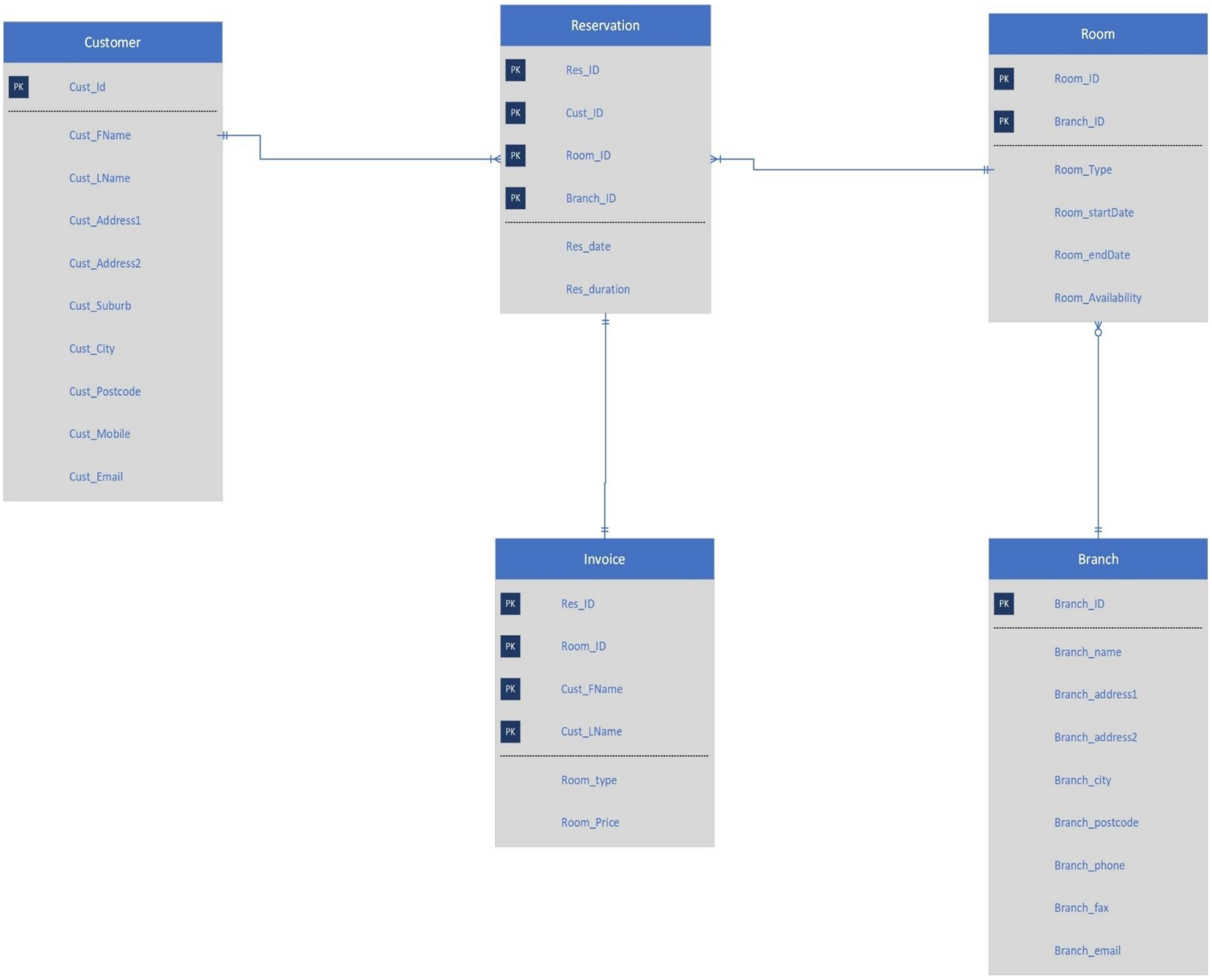


Table Structures/Choice and Explanations for Table Parameters and Values

Customer Table

```
CREATE TABLE Customer (

Cust_id             Number(5),

Cust_FName    Varchar2(20),

Cust_LName    Varchar2(20),

Cust_address1 Varchar2(20),

Cusr_address2 Varchar2(20),

Cust_suburb   Varchar2(20),

Cust_city     Varchar2(20),

Cust_Postcode       Varchar2(20),

Cust_Mobile   Number(12),

Cust_Email    varchar2(20),

CONSTRAINT Customer_PK PRIMARY KEY(Cust_id)

)TABLESPACE gsj6766_TS02_Customer PCTFREE 30 PCTUSED 60

STORAGE(INITIAL 3400K NEXT 3200K PCTINCREASE 0 MAXEXTENTS 5);
```

Customer Table	**Field Type**	**Primary Key/Foreign Key**
Cust_Id	Number (5)	PK
Cust_FName	varchar2(20)	PK
Cust_LName	varchar2(20)	PK
Cust_Address1	varchar2(20)	N/A
Cust_Address2	varchar2(20)	N/A
Cust_Suburb	varchar2(20)	N/A
Cust_City	varchar2(20)	N/A
Cust_Postcode	varchar2(20)	N/A
Cust_Mobile	Number(12)	N/A
Cust_Email	varchar2(20)	N/A

Explanation for Customer table

The customer table has three primary keys. Cust_Id is used to uniquely identify each customer even if their other details are identical. Cust_FName and Cust_LName are also identified as primary keys since they will be used by customers to identify themselves during check-out to receive their invoice. The rest of the details are for identifying the customers' contact details and their address.

Storage explanations

cust_id = uses 5 different numbers to be queried and keep it unique.

Cust_fname + cust_lname = uses varchar2 to list the customers full name.

Cust_address1 until cust_postcode = uses varchar2 to list the customer's full address.

Cust_mobile = will use only numbers to list down customer's mobile number.

Cust_email = emails usually consist of a mix between integers and characters hence varchar2 will be best.

Reservation Table

```
CREATE TABLE gsj6766_Reservation (

Res_id       Number(5),

Cust_id             Number(5),

Room_id      Number(5),

Branch_id    Number(5),

Res_date     Date,

Res_duration Number(3),

CONSTRAINT Reservation_PK PRIMARY KEY(Res_id, Cust_id, Room_id, Branch_id)

)

TABLESPACE gsj6766_TS02_Res_In PCTFREE 30 PCTUSED 60

STORAGE(INITIAL 270K NEXT 110K PCTINCREASE 0 MAXEXTENTS 2);
```

Reservation Table	**Field Type**	**Primary Key/Foreign Key**
Res_ID	Number (5)	PK
Cust_ID	Number(5)	PK/ FK
Room_ID	Number(5)	PK/ FK
Branch_ID	Number(5)	PK/ FK
Res_date	Date	N/A
Res_duration	Number(3)	N/A

<u>Explanation for Reservation table</u>

The reservation table is used to link the customer, the room, and their invoice. It uses primary keys from the other tables to bridge them together and it is dependent on the other tables as well. It contains its own primary key to uniquely identify each reservation. It has 4 different primary keys to be able to pinpoint each reservation precisely and will not cause issues upon getting the wrong customer, room, or branch. It also uses Res_date to determine the date when the reservation was made and is simply tracked with Res_duration to count the amount of days the reservation has existed for.

<u>Storage explanations</u>

Res_id, cust_id, room_id, branch_id = these parameters use 5 numbers to uniquely identify them.

Res_date = will use the date type to record the exact date the reservation was made from the system's time.

Res_duration = use numbers to record the duration of the reservation in days (e.g. 15 days).

Room Table

```
CREATE TABLE gsj6766_Room (

Room_id              Number(5),

Branch_od            Number(5),

Room_type            varchar2(20),

Room_startDate       Number(6),

Room_endDate         Number(6),

Room_availability    Varchar2(10),

CONSTRAINT Room_PK PRIMARY KEY (room_id)

)

TABLESPACE gsj6766_TS03_Branch_Room

STORAGE(INITIAL 380K NEXT360K);
```

Room Table	**Field Type**	**Primary Key/Foreign Key**
Room_ID	Number (5)	PK
Branch_ID	Number(5)	FK
Room_Type	Varchar2(20)	N/A
Room_startDate	Number(6)	N/A
Room_endDate	Number(6)	N/A
Room_Availability	Varchar2(10)	N/A

<u>Explanation for Room table</u>

The room table contains its own primary key to uniquely identify each room differently. It borrows branch_Id from the branch table to determine which branch the room belongs to. It also contains a room type to differentiate between single, double, twin, and suit. Room_startDate and Room_endDate are used to track when it begins to be occupied and the end date is when the customers vacate the room. Room_availability determines if the room is occupied or not.

<u>Storage Explanations</u>

Room_id, branch_id, = id will use 5 numbers to be identified uniquely.

Room_type = uses varchar2 to identify whether it’s single, double, twin, or a suite.

Room_startDate, room_endDate = uses number(6) to identify the day, month, and year format (11.12.18).

Room_availability = uses varchar2 to indicate whether it is “available” or “unavailable”.

Branch Table

```
CREATE TABLE gsj6766_Branch (

Branch_id           Number(5),

Branch_name         Varchar2(10),

Branch_name         Varchar2(20),

Branch_address1     Varchar2(20),

Branch_city         Varchar2(20),

Branch_postcode     Varchar2(20),

Branch_phone        Number(12),

Branch_fax          Number(12),

Branch_email        Varchar2(30),

CONSTRAINT Branch_PK PRIMARY KEY (branch_id)

)

TABLESPACE gsj6766_TS03_Branch_Room

STORAGE(INITIAL 380 NEXT 360);
```

Branch Table	**Field Type**	**Primary Key/Foreign Key**
Branch_ID	Number (5)	PK
Branch_name	Varchar2(10)	N/A
Branch_address1	varchar2(20)	N/A
Branch_address2	varchar2(20)	N/A
Branch_city	varchar2(20)	N/A
Branch_postcode	varchar2(20)	N/A
Branch_phone	Number(12)	N/A
Branch_fax	Number(12)	N/A
Branch_Email	Varchar2(30)	N/A

<u>Explanation for Branch table</u>

There are currently 14 different branches and the branch_id helps to determine which branch the room, reservation, or customer will belong to. The branch_name will help identify the correct branch through the building's name. The rest of the details are used to determine the location of the branch and its contact details to communicate with the branch.

<u>Storage Explanations</u>

Branch_id = uses Number(5) to identify the id uniquely.

Branch_name, branch_address1, branch_address2, branch_city, branch_postcode, branch_email = uses varchar2 as these parameters will have a mix of both numbers and characters and has been given a limit of 20 - 30 characters per value.

Branch_phone, branch_fax = given 12 different numbers to identify the branch's contact details.

Invoice Table

```
CREATE TABLE gsj6766_Invoice (

Res_id        Number(5),

Room_id       Number(5),

Cust_Fname    Varchar2(20),

Cust_Lname    Varchar2(20),

Room_type     Varchar2(20),

Room_price    Number(9),

CONSTRAINT Invoice_PK PRIMARY KEY (res_id, room_id, cust_Fname,cust_lname)

)

TABLESPACE gsj6766_TS02_Res_In PCTFREE 30 PCTUSED 60

STORAGE(INITIAL 270K NEXT 110K MAXEXTENTS 2);
```

<u>Calculation and reasoning for initial size for the tables reservation and invoice:</u>

These tables will not be updated as much as the customer table, however it will still require enough space for updating details. Due to this, PCTFREE will have to be set to 30 again to allow enough space for quarterly updates. PCTUSED can be set to 60 as it will need the space in block as the reservations will get archived but the invoices can be deleted after payment. PCTINCREASE will be set to 0 to allow the database to grow consistently. MAXEXTENTS will be set to 2 to allow space for potential updates in the future. There are a total of 1000 reservations on average per day. Taking this number and assuming that this will be quite constant, rows per block can be used to divide this number to receive the total table size. The total table size will then be multiplied by 8 as each block consists of 8 kilobytes. To receive the INITIAL size of the table, 33 must be multiplied by 8 which yields 264KB. To allow some extra space, the customer table's INITIAL size will be set to 270KB, NEXT will be set to 110KB as this is 40% of the INITIAL size since 60% of the customer's data are repeat business.

Invoice Table	**Field Type**	**Primary Key/Foreign Key**
Res_ID	Number (5)	PK/FK
Room_ID	Number(5)	PK/FK
Cust_FName	varchar2(20)	PK/FK
Cust_LName	varchar2(20)	PK/FK
Room_type	varchar2(20)	FK
Room_Price	Number(9)	FK

Explanation for Invoice table

Lastly, the invoice table requires 4 different primary keys that are derived off the reservation, room, and customer table. It uses these primary keys to identify which invoice is being queried by finding the res_id, room_id, and the customer's first name and last name. It also states the type of room the customer booked and the price for the entire stay's duration. The room_price is calculated through the type or room and the length the customer stayed for.

Storage Explanations

Res_id, room_id = will use 5 different numbers to search for the id uniquely.

Cust_fname, cust_lname = given 20 different varchar2 characters to give customers enough space for their first and last names.

Room_type = uses varchar2 to identify the type of room the customer hired such as single, double, twin, or suite.

Room_price = price calculated in New Zealand dollars and uses 9 figure numbers to identify the total price of rented by customer.

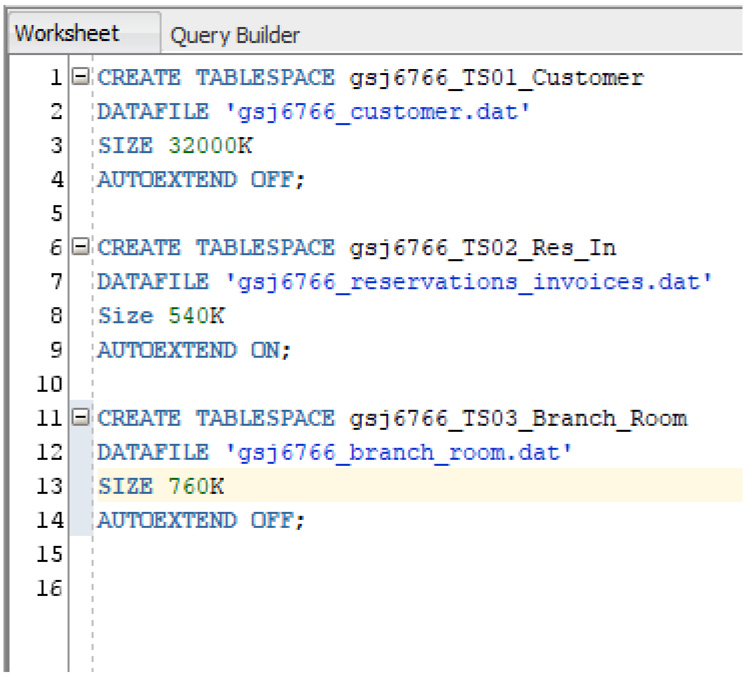
Shannon Imbo - 1303744

Tablespace

Tablespace Creation Script

<u>Explanation for Tablespaces choices</u>

The tablespaces will be split into 3 different ones. The customer table shall have its own tablespace as it will require the most space and constant updating. Putting the customer table on its own tablespace allows it to be independent from any potential errors in the future. Reservation and Invoices are placed together in the same tablespace as invoices cannot be generated without reservations and they tend to go hand in hand. Branch and room table are combined since branches contain rooms.



```sql
CREATE TABLESPACE gsj6766_TS01_Customer
DATAFILE 'gsj6766_customer.dat'
SIZE 32000K
AUTOEXTEND OFF;

CREATE TABLESPACE gsj6766_TS02_Res_In
DATAFILE 'gsj6766_reservations_invoices.dat'
Size 540K
AUTOEXTEND ON;

CREATE TABLESPACE gsj6766_TS03_Branch_Room
DATAFILE 'gsj6766_branch_room.dat'
SIZE 760K
AUTOEXTEND OFF;
```

Shannon Imbo - 1303744

Calculation of Tablespace Size

Header Size

HSIZE = DB_BLOCK – KCBH – UB4 – KTBBH – (INITRANS – 1) * KTBIT – KDBH

DB_BLOCK_SIZE = 8192

(i) KCBH	20
(ii) UB4	4
(iii) KTBBH	48
(iv) KTBIT	24
(v) KDBH	14
(vi) UB1	1
(vii) SB2	2

8192 – 20 – 4 – 48 – (1 – 1) * 24 – 14 = 8106 (HSIZE)

Calculation and reasoning for initial size for the customer table:

The customer table will be the most updated and will use the most memory out of all the other tables. It already has 12,000 customers recorded at the start of the quarterly processing cycle. Hence, it will require a high PCTFREE of 30 for future updates. PCTUSED will also be quite high due to the existing customers and must be set to 60. PCTINCREASE will be set to 0 for consistency. MAXEXTENTS will be set to 5 to support future updates and inserts. The total table size will be calculated by dividing 12,000 customer data by 30 row blocks from the customer table. It will yield 400KB worth of data which can then be multiplied by 8. The total table size will be 3,200KB or 3.2MB. INITIAL size will be set to 3.4M and NEXT size will be 3.2M.

<u>Calculation and reasoning for initial size for the tables reservation and invoice:</u>

These tables will not be updated as much as the customer table, however it will still require enough space for updating details. Due to this, PCTFREE will have to be set to 30 again to allow enough space for quarterly updates. PCTUSED can be set to 60 as it will need the space in block as the reservations will get archived but the invoices can be deleted after payment. PCTINCREASE will be set to 0 to allow the database to grow consistently. MAXEXTENTS will be set to 2 to allow space for potential updates in the future. There are a total of 1000 reservations on average per day. Taking this number and assuming that this will be quite constant, rows per block can be used to divide this number to receive the total table size. The total table size will then be multiplied by 8 as each block consists of 8 kilobytes. To receive the INITIAL size of the table, 33 must be multiplied by 8 which yields 264KB. To allow some extra space, the customer table's INITIAL size will be set to 270KB, NEXT will be set to 110KB as this is 40% of the INITIAL size since 60% of the customer's data are repeat business.

<u>Calculation and reasoning for initial size for the tables branch and room:</u>

These tables do not require a lot of updating hence it will need quite little space and processing power. PCTFREE, PCTUSED, PCTINCREASE, and MAXEXTENTS will be set to default as there will be little change overall for these tables. The calculation method used was to multiply the rooms by the branches to get an even amount of space for all branches as all the layouts are the same for each building. In total, there are 2100 rooms per branch. Dividing the rooms by 47 rows per block will yield 45. 45 will then be multiplied by 8KB and the result will be 360KB. Thus, the INITIAL will be set to 380KB and NEXT to 360KB.

Shannon Imbo - 1303744

Calculations for customer table

Availspace for customer

availspace = CEIL(hsize*(1 – PCTFREE/100)) – KDBT

8106*(1-30/100)-14 = 5660

Rowspace and total table size for customer

Calculating the column size

total column size = column size + byte length

177 + 10 = 187

Rowsize = row header + sum of all column sizes

3 + 177 = 180

Rowspace = max(row header+UB4+SB2, rowsize)+SB2

Max(3+4+2,180)+2

Rowspace = 182

RB = FLOOR(availspace/rowspace)

RB = 5660/187 = 30

Total table size = 12000 customers / 30 rows per block = 400

Calculations for Reservation and Invoice Table

Calculations for reservation and invoice table

Availspace for reservation and invoice

availspace = CEIL(hsize*(1 – PCTFREE/100)) – KDBT

8106*(1-30/100)-14 = 5660

Rowspace and total table size for reservation and invoice

Calculating the column size

total column size = column size + byte length

79 + 6 = 85

Rowsize = row header + sum of all column sizes

3 + 79 = 82

Rowspace = max(row header+UB4+SB2, rowsize)+SB2

Max(3+4+2,82)+2

Rowspace = 84

RB = FLOOR(availspace/rowspace)

RB = 5660/84 = 67

Total table size = 1000 reservations / 67 rows per block = 14.9

Shannon Imbo - 1303744

Calculations for Branch and Room Table

Calcuations for branch and room table

Availspace for branch and room

availspace = CEIL(hsize*(1 – PCTFREE/100)) – KDBT

8106*(1-10/100)-14 = 7281

Rowspace and total table size branch and room

Calculating the column size

total column size = column size + byte length

149 + 9 = 158

Rowsize = row header + sum of all column sizes

3 + 149 = 152

Rowspace = max(row header+UB4+SB2, rowsize)+SB2

Max(3+4+2,152)+2

Rowspace = 154

RB = FLOOR(availspace/rowspace)

RB = 7281/154 = 47

Total table size = (150 rooms * 14 branches) / 47 rows per block = 44.68

Creation of Users, Roles, and Profiles

Security Policy and Matrix

User Type	System /Object Privileges	Justification
Database Administrator (DBA)	All privileges	The Database administrator is responsible for handling the entire database and needs all access to be able to maintain and update database.
Developer	Create Table, alter table, drop table, create any table, alter any table, drop any table, delete any table, insert ay table, select any table, update any table, create index, alter index, drop index, create any index, alter any index, drop any index, create view, drop view, create session, alter session, create procedure, alter procedure, drop procedure, create any procedure, alter any procedure, execute any procedure, create trigger, alter trigger, drop trigger, create function, alter function, drop function, create sequence, create any sequence, alter any sequence, drop any sequence, select any sequence.	Developers will be able to do most actions except those associated with users, roles, profiles, tablespaces as this is the database administrator's role. Developers will be required to ask permission to create test accounts or anything involved with privileges that they do not currently have.
User	Create session only	Users such as hotel receptionists or hotel managers can create sessions to log on and access the application within boundaries set by DBA.

Creating Users

Database Administrator:

```
CREATE USER databaseadmin

IDENTIFIED BY defaultadmin

DEFAULT TABLESPACE gsj6766_TS02_Res_In

TEMPORARY TABLESPACE temp

QUOTA 4M ON gsj6766_reservations_invoices.dat;
```

Developer:

```
CREATE USER developer

IDENTIFIED BY defaultdev

DEFAULT TABLESPACE gsj6766_TS02_Res_In

TEMPORARY TABLESPACE temp

QUOTA 4M ON gsj6766_reservations_invoices.dat;
```

Receptionist User:

```
CREATE USER receptionist

IDENTIFIED BY defaultreceptionist

DEFAULT TABLESPACE gsj6766_TS02_Res_In

TEMPORARY TABLESPACE temp

QUOTA 2M ON gsj6766_reservations_invoices.dat;
```

Hotel Manager User:

```
CREATE USER manager

IDENTIFIED BY gsj6766_TS02_Res_In

DEFAULT TABLESPACE hotel_ts01.dat

TEMPORARY TABLESPACE temp

QUOTA 2M ON gsj6766_reservations_invoices.dat;
```

Explanations and Assumptions for Users

There have been 2 extra users added which are the manager and the receptionist users to allow different users to access the database within a controlled environment. The Developer and the database admin are both quite similar in terms of user creations.

Creating Roles

Database Administrator Role:

```
GRANT CREATE SESSION to databaseadmin;
GRANT ANY PRIVILEGE TO databaseadmin;
```

Explanation

The DBA will have all system and object privileges as they will need full control over the database. They will be solely responsible for maintaining and updating the entire database hence having all the authority.

Developer Role:

```
GRANT CREATE ANY Table TO developer;
GRANT ALTER ANY Table TO developer;
GRANT DROP ANY Table TO developer;
GRANT INSERT ANY Table TO developer;
GRANT DELETE ANY Table TO developer;
GRANT UPDATE ANY Table TO developer;
GRANT SELECT ANY Table TO developer;
GRANT CREATE ANY Index TO developer;
GRANT ALTER ANY Index TO developer;
GRANT DROP ANY Index TO developer;
GRANT CREATE ANY View TO developer;
GRANT ALTER ANY View TO developer;
GRANT DROP ANY View TO developer;
GRANT CREATE ANY Session TO developer;
GRANT ALTER ANY Session TO developer;
GRANT CREATE ANY Procedure TO developer;
GRANT ALTER ANY Procedure TO developer;
GRANT DROP ANY Procedure TO developer;
GRANT CREATE ANY Trigger TO developer;
GRANT ALTER ANY Trigger TO developer;
GRANT DROP ANY Trigger TO developer;
```

Explanation

Developers will provide more of a supporting role when it comes to the database. They will be able to obtain most system privileges except those that are related to manipulating tablespace, users, roles, and profiles. This way, the DBA will have full control and developers only having partial. The developers will need permission from the DBA to access privileges that they do not have.

Receptionist User Role:

```
GRANT CREATE SESSION to receptionist;
```

Hotel Manager User Role:

```
GRANT CREATE SESSION to manager;
```

Explanation

Both the receptionist and the hotel managers will be given the same privileges. They will mainly be able to create sessions and interact with the database in a controlled environment under the DBA's authority.

Creating Profiles

```
CREATE PROFILE developer

SESSIONS_PER_USER   2

CPU_PER_SESSION     UNLIMITED

CPU_PER_CALL        UNLIMITED

LOGICAL_READS_PER_SESSION unlimited

LOGICAL_READS_PER_CALL UNLIMITED

CONNECT_TIME  UNLIMITED

FAILED_LOGIN_ATTEMPTS 3

PASSWORD_LOCK_TIME 30;


CREATE PROFILE user

SESSIONS_PER_USER   2

CPU_PER_SESSION     3000

CPU_PER_CALL        2000

LOGICAL_READS_PER_SESSION unlimited

LOGICAL_READS_PER_CALL 100

CONNECT_TIME  600

FAILED_LOGIN_ATTEMPTS 3

PASSWORD_LOCK_TIME 30;
```

Explanation

There will only be 2 types of profiles. The developer profile will mostly have unlimited access to resources from the database. Those that have the developer profile will be highly responsible for any action taken within the database. User profiles have very limited access and will not be given a lot of resources as they will mainly create sessions to view database information and records.

Assigning the profiles to the correct users

```
ALTER USER databaseadmin
       PROFILE developer;
ALTER USER developer
       PROFILE developer;
ALTER USER receptionist
       PROFILE user;
ALTER USER manager
       PROFILE user;
```

Explanation

This just assigns the profiles to the correct users.

```
CREATE OR REPLACE PROCEDURE insert_customer

(vCust_id IN cust.Cust_id%TYPE)

IS
BEGIN

        SELECT Cust_id, Cust_FName, Cust_LName, Cust_Address1, Cust_Address2, Cust_Suburb, Cust_City,
Cust_Postcode, Cust_mobile, Cust_Email

        INTO vCust_id, vCust_FName, vCust_LName, vCust_Address1, vCust_Address2, vCust_Suburb,
vCust_City, vCust_Postcode, vCust_Mobile, vCust_Email

        FROM cust

END insert_customer;

/
```

SQL Script for Inserting into Reservation Table

```
CREATE OR REPLACE PROCEDURE insert_reservation

(vRest_ID IN res.Res_Id%TYPE,

vCust_ID IN res.Cust_Id%TYPE,

vRoom_ID IN res.Room_Id%TYPE,

vBranch_ID IN res.Branch_ID%TYPE,

vRes_date IN res.Res_date%TYPE,

vRes_duration IN res.Res_duration)

IS

BEGIN

       SELECT Rest_ID, Cust_ID, Room_ID, Branch_ID , Res_date , Res_duration

       INTO vRest_ID, vCust_ID, vRoom_ID, vBranch_ID, vRes_date, vRes_duration

       WHERE res

END insert_reservation;

/
```

Shannon Imbo - 1303744

```
CREATE OR REPLACE PROCEDURE insert_room
(vRoom_ID IN room.Room_ID%TYPE,
vBranch_ID IN room.Branch_ID%TYPE,
vRoom_Type IN room.Room_Type%TYPE,
vRoom_startDate IN room.Room_startDate%TYPE,
vRoom_availability IN room.Room_availability%TYPE,
vRoom_endDate IN room.Room_endDate%TYPE)
IS
BEGIN
        SELECT Room_ID, Branch_ID, Room_Type, Room_startDate, Room_availability, Room_endDate
        INTO   vRoom_ID, vBranch_ID, vRoom_Type, vRoom_startDate, vRoom_availability, vRoom_endDate
        WHERE room_table;
```