**ASSIGNMENT COVER SHEET**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Course: BSc Computing (SE)** | | | | **Year: 2** | | **CSY2038** | |
| **Group Project** | | **Title:** Design, create and test methods for building an object relational database with useful data abstraction and automatizing useful processes in Pl/SQL | | | | | |
| Date due out: | Date due in: 13th Nov, 2022 | | Extension date: | | | | Extension agreed by: |
| **Team: 19**  **Group Members:**   1. Miraj Thapa 2. Rupak Upreti 3. Sakshyam Aryal 4. Suyog Kadariya | | | | | | **Tutor: Ankit Thapa** | |
| Student comment, specific request for feedback etc. | | | | | Marker’s General View of the work | | |

Video Demo Link:

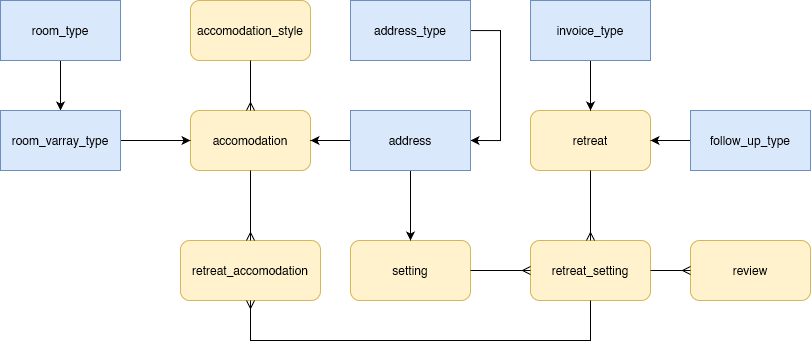
**Contents**

**Abstract**

In any system where data management is required, the implementation of a database is absolutely essential. In order to establish an object relational database with usable data abstraction and automate helpful procedures in PL/SQL, we designed and created methods for doing so. We, the team members, talked on how to organize files securely and appropriately as a collaborative project. First, we generated a skeleton table in accordance with the schema we had defined. All the tables, or UDTs, are included in our skeleton table. For this project, we have also created functions, triggers, and cursors as well as developed queries and procedures. By assigning responsibilities to each member independently, the four of us worked on distinct parts of the project. As part of our collaboration, we discussed each question and came up with answers to each issue. With the completion of this project, we hope to design, develop, and test techniques for creating object relational databases.

1. **Database Schema Design**

Schema design is the initial step in the database design process. The database schema refers to the overall layout of the database. It stands for the logical overview of the entire database architecture. The database schema that we designed is provided below.



*Fig: Database Schema Design*

1. **Skeleton Table Design**

To comprehend the database structure at its most fundamental level, a skeleton table is required. Based on the schema design, the skeleton table that follows shows all the specifics of the database tables, including their characteristics, keys, datatypes, constraints and default values.

| **Tables** | **attribute** | | **Key** | **Datatype** | **Constraints \**  **Defaults** |
| --- | --- | --- | --- | --- | --- |
| **Retreats** | retreat\_id | | *pk* | NUMBER(6) |  |
|  | retreat\_name | |  | VARCHAR2(30) | NOT NULL |
|  | invoice | |  | invoice\_type |  |
|  | follow\_up | |  | follow\_up\_type |  |
| **Settings** | setting\_id | | *pk* | NUMBER(6) |  |
|  | setting\_name | |  | VARCHAR2(30) | NOT NULL |
|  | address | |  | address\_type |  |
| **retreat\_setting** | | retreat\_setting\_id | *pk* | NUMBER(6) |  |
|  | setting\_id | | *FK* | NUMBER(6) | NOT NULL |
|  | retreat\_id | | *FK* | NUMBER(6) | NOT NULL |
| **accomodation\_style** | accomodation\_style\_id | | *pk* | NUMBER(6) |  |
|  | accomodation\_style\_name | | *pk fk* | NUMBER(6) | NOT NULL |
| **accomodations** | accommodation\_id | | *pk* | NUMBER(6) |  |
|  | accommodation\_name | |  | VARCHAR2(50) | NOT NULL |
|  | room | |  | room\_varray\_type |  |
|  | no\_of\_room | |  | VARCHAR2(12) |  |
|  | address | |  | address\_type |  |
|  | description | |  | VARCHAR2(100) |  |
|  | accommodation\_style\_id | | *FK* | NUMBER(6) | NOT NULL |
| **retreat\_accommodation** | retreat\_setting\_id | | *pk*  *FK* | NUMBER(6) | NOT NULL |
|  | accommodation\_id | | *pk fk* | NUMBER(6) | NOT NULL |

UDT’s used are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **address\_type** | street |  | VARCHAR2(50) |  |
|  | city |  | VARCHAR2(50) |  |
|  | country |  | VARCHAR2(50) |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **follow\_up\_type** | start\_time |  | VARCHAR2(12) |  |
|  | duration |  | VARCHAR2(12) |  |
|  | date |  | DATE |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **invoice\_type** | invoice\_holder |  | NUMBER(6) |  |
|  | amount |  | VARCHAR2(10,2) |  |
|  | release\_date |  | DATE |  |
|  | due\_date |  | DATE |  |

Varray\_type used are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **room\_type** | room\_id |  | NUMBER(6) |  |
|  | capacity |  | VARCHAR2(10,2) |  |
|  | description |  | VARCHAR2(100) |  |

1. **Database tables**

To make clear of all the tables, separate tables have been made for good understandings. Attributes, keys, datatypes, constraints and default values are also included.

**Table no 1)** retreat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| retreats | retreat\_id | *pk* | NUMBER(6) | NOT NULL |
|  | retreat\_name |  | VARCHAR2(30) | NOT NULL |
|  | invoice |  | invoice\_type |  |
|  | follow\_up |  | follow\_up\_type |  |

**Table no 2)** setting

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| settings | setting\_id | *pk* | NUMBER(6) | NOT NULL |
|  | setting\_name |  | VARCHAR2(30) | NOT NULL |
|  | address |  | address\_type |  |

**Table no 3)** retreat\_setting

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| retreat\_setting | retreat\_setting\_id | *pk* | NUMBER(6) | NOT NULL |
|  | setting\_id | *fk* | NUMBER(6) | NOT NULL |
|  | retreat\_id | *f*k | NUMBER(6) | NOT NULL |

**Table no 4)** accommodation\_style

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| accommodation\_style | accommodation\_style\_id | *pk* | NUMBER(6) | NOT NULL |
|  | accommodation\_style\_name | *pk, fk* | NUMBER(6) | NOT NULL |

**Table no 5)** accommodation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| accommodations | accommodation\_id | *pk* | NUMBER(6) | NOT NULL |
|  | accommodation\_name |  | VARCHAR2(50) |  |
|  | room |  | room\_varray\_type |  |
|  | no\_of\_rooms |  | VARCHAR2(12) |  |
|  | address |  | address\_type |  |
|  | description |  | VARCHAR2(100) |  |
|  | accomodation\_style\_id | *fk* | NUMBER(6) | NOT NULL |

**Table no 6)** retreat\_accommodation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| retreat\_accommodation | retreat\_setting \_id | *pk, fk* | NUMBER(6) | NOT NULL |
|  | accommodation\_id | *pk, fk* | NUMBER(6) | NOT NULL |

1. **Functions, Sequences, Procedures, Triggers, Cursors and Queries**

Table below consists of all the functions, procedures, triggers and queries created. Table is divided into Name and description of the names.

|  |  |
| --- | --- |
| **NAME** | **DESCRIPTION** |
| Functions |  |
| *name\_of\_the\_function* | *e.g., function to determine the (its function)* |
|  |  |
|  |  |
|  |  |
| Sequence |  |
| *name\_of\_the\_sequence* | *starts with 1000000 and increments by 1* |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **NAME** | **DESCRIPTION** |
| Procedures |  |
| *name\_of\_the\_procedure* | *what does that procedure do* |
|  |  |
|  |  |
|  |  |
| Triggers |  |
| *name\_of\_the\_trigger* | *what does that trigger do* |
|  |  |
|  |  |
| Cursors |  |
| *name\_of\_the\_cursor* | *what does that cursor do* |
|  |  |
|  |  |
| Queries | |
| *e.g., highest booking cost using functions* | |
|  | |
|  | |
|  | |
|  | |
|  | |

1. **Proposed Automation Strategy**
2. **Evidence of Additional Research**

In addition to the assignment briefs and project files we received, we had to conduct extensive study on database construction techniques before beginning the job. We conducted team research on many topics. We read a few research papers and forum posts from Oracle developers before having a team discussion. The "**Oracle Database Online Documentation 12c Release 1(12.1)**" and a few other articles listed in the reference section of this document were the most helpful ones we could find.

1. **Description along with screenshots**

Below contains all the function, triggers, procedures functionalities that describes all the functionalities along with the name.

* 1. **Procedures**

Similar to database functions, procedures comprise SQL statements that are only used to carry out certain tasks. Procedures are mostly used for data retrieval, data retrieval, access control, and multiple parameter queries. Below table consists procedure name and description.

|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **DESCRIPTION** |
| 1 | *procedure\_one* | *its description, same as above with a little more brief* |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

* + 1. **Procedure functionalities**

1. procedure\_one\_name

Add screenshot of the procedure.

1. procedure\_one\_name

Add screenshot of the procedure.

1. procedure\_three\_name

Add screenshot of the procedure.

1. procedure\_four\_name

Add screenshot of the procedure.

* 1. **Functions**

A function is a collection of SQL statements that carry out a particular task. To make operations simpler and quicker, we have created three distinct functionalities. The table below lists the functions and their descriptions.

|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **DESCRIPTION** |
| 1 | *function\_one\_name* | *its description, same as above with a little more brief* |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

* + 1. **Function functionalities**

1. function\_one\_name

Add screenshot of the function.

1. function\_one\_name

Add screenshot of the function.

1. function\_one\_name

Add screenshot of the function.

1. function\_one\_name

Add screenshot of the function.

**7.3** **Triggers**

A trigger in a database is a piece of procedural code that is automatically performed out in response to specific events on a specific table or view in a database. We created 3 different triggers for specific events that is described in the table below.

|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **DESCRIPTION** |
| 1 | *trigger\_one\_name* | *its description, same as above with a little more brief* |
| 2 |  |  |
| 3 |  |  |

* + 1. **Trigger functionalities**

1. trigger\_one\_name

Add screenshot of the function.

1. trigger\_one\_name

Add screenshot of the function.

1. trigger\_one\_name

Add screenshot of the function.

1. **Test Plan**

The testing phase of a project is one of the most fundamental stages. In the testing phase, we put our attention on investigation and discovery. All of the functions, procedures, sequences, cursors and triggers we developed were tested to make sure they all produced the desired outcome. We created a list of all the test procedures to be followed, serially numbered the IDs, and then ran the test. The table below displays the findings of our tests.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **TEST DESCRIPTION** | **EXPECTED RESULT** | **ACTUAL RESULT** | **ACTION** |
| 1 | *e.g.,*  SELECT booking\_id,  CEIL(cost), FLOOR(cost),  ROUND(cost),  TRUNC(cost), cost  FROM bookings  WHERE cost = (  SELECT MAX(cost)  FROM bookings  ); | 2880000 | 2880000 (PASS) |  |
| 2 | *e.g.,*  **Test proc\_higher\_rate**  EXEC proc\_higher\_rate(5); | Designers with a rating  higher than 5  1. Designer ID: 10000001  Designers Rating: 10  2. Designer ID: 10000003  Designers Rating: 6 | Designers with a rating  higher than 5  1. Designer ID: 10000001  Designers Rating: 7  2. Designer ID: 10000001  Designers Rating: 7  3. Designer ID: 10000001  Designers Rating: 10  4. Designer ID: 10000003  Designers Rating: 6 | Need to account for cartesian product by limiting results to singular designer and then using a function such as average or max to give a rating value. |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

1. **References**
2. Drew, A. and Paapanen, E. (2003) *Using Triggers*, *9 using triggers*. Oracle Corporation. [Available at: https://docs.oracle.com/cd/B13789\_01/appdev.101/b10795/adfns\_tr.htm](Available%20at:%20https:/docs.oracle.com/cd/B13789_01/appdev.101/b10795/adfns_tr.htm) (Accessed: October 15, 2022).
3. Sharan, A. (2018) *Cursors in PL/SQL*, *GeeksforGeeks*. GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/cursors-in-pl-sql/> (Accessed: October 15, 2022).
4. Amazon Web Services, Inc. (1989) *Oracle Procedures and Functions and MySQL Stored Procedures*, *Amazon*. Det Danske Missionsselskab. Available at: <https://docs.aws.amazon.com/dms/latest/oracle-to-aurora-mysql-migration-playbook/chap-oracle-aurora-mysql.sql.stored.html> (Accessed: October 23, 2022).
5. Oracle Tutorial (2020) *Oracle create sequence explained by practical examples*, *Oracle Tutorial*. Available at: <https://www.oracletutorial.com/oracle-sequence/oracle-create-sequence/> (Accessed: November 03, 2022).
6. Pedamkar, P. (2021) *Function in oracle: How do functions in Oracle work with examples*, *EDUCBA*. Available at: [https://www.educba.com/function-in-oracle/](https://www.educba.com/function-in-oracle/%20) (Accessed: November 03, 2022).
7. **Appendix**