**Object Description**

Group 2

This document lists and describes all objects that we’ve created. In summary, we’ve created 6 sequences, 2 views, 4 procedures, 2 function, 1 package, 2 database triggers,1 scheduled job, 2 roles, and 2 alternate indexes.

**1. Sequences**

subdivition\_seq

room\_seq

choice\_seq

sales\_seq

staff\_seq

buyer\_seq

We create sequences mainly for \*id columns in the SUBDIVISION, ROOM, BUYER CHOICE, SALES, STAFF, BUYER table. The reason we do so is because sequences can help generate ID automatically and uniquely, which help save time for users to number those subdivisions, choices, sales, etc.

**2. Views**

sales\_view

staff\_view

The sales\_view is used to display the basic information of each sales record, including *sale\_id, sale\_date, buyer\_id, staff\_id, lot\_id*. Due to buyers’ privacy, other columns will not be available to be seen. The staff\_view contains staff information for managers, including *staff\_id, staff\_name, phone\_num*. Considering the privacy of staff, the license\_num won’t be seen by users.

**3. Procedures**

display\_house\_style

construction\_progress

sale\_contract

check\_cancel

Three procedures are created for displaying the figure 1, figure 4 and figure 5 shown in the case description. And one procedure is designed to show whether a buyer is eligible for canceling the contract or not.

For figure 1: Simply compile and execute display\_house\_style(subdivision\_id, style\_id). A list of details for the specified style in subdivision will be generated. It contains the size, price, room details, elevation details, and the size percentage of all rooms in this style size.

For figure 4: Simply compile and execute construction\_progress(progress\_id). A Construction Progress list can be output automatically with details about location, lot and tasks. The structure is the same as shown in the case.

For figure 5: Simply compile and execute sale\_contract(sale\_id). A sale receipt can be output with bunches of details about buyer and staff; subdivision, lot and house; bank and financial details. The structure is the same as shown in the case.

Simply compile and execute check\_cancel(sale\_id). A result will be output to show whether the buyer CAN or CANNOT cancel the contract. Followed by this, details of the sale record will be printed, including *sale\_id, buyer\_id, sake\_date, est\_date\_of\_completion, staff\_id, lot\_id*.

**4. Functions**

calc\_pct

if\_can\_cancel

The function calc\_pct(style\_id) is created for calculating the size percentage of all rooms in one specific style. As for one style, its size contains several components - rooms, elevation, entrance, etc. We want to calculate the percentage of all rooms, to let the users know the space utilization situation for each style.

The function if\_can\_cancel(sale\_id) is created for determining whether a buyer can cancel the contract or not. By inputting a *sale\_id*, the function will return a binary value – 0 or 1, 1 means can cancel and 0 means cannot.

**5. Package**

house\_details

A package named house\_details is created for extracting detailed information from tables related to houses. The package contains one function and one procedure. The function calc\_pct serves the same effect as the function we described above. The procedure display\_details can be used to return details pertaining to the style in the subdivision.

**6. Database Triggers**

print\_option\_change

staff\_deleting

The trigger print\_option\_change is used to display a difference of money cost between old and new options. Since buyers often change their minds and change stage for specific opinions, this trigger can be fired before updating the buyer\_choice table and then display how much money they will pay more or less than that before updating the options.

The trigger staff\_deleting is to control the problem of staff leaving. If one staff quits Eggshell and before the data in the staff table are deleted, the trigger will find out all related constructions, sales and choices and set the *staff\_id* in charge to null. This is a trigger to take care of staff leaving, by searching the null value, we could easily find those constructions, sales and choices without a staff in charge and then allocate them to other staff.

**7. Roles**

The two roles created for this project are:

Lot\_Buyer

Sales\_Staff

**Lot\_Buyer** is a role given to the buyer who has only select access on the following tables:

*Lot, Subdivision, Options and Room*. This is because the buyer would want to look at the different choices he/she has while selecting a lot with a specific style of house and its options.

**Sales\_Staff**, the second role, is related to the tasks that a Sales staff performs, in this case the role has select, insert, update and delete privileges over the following tables:

*Sales, Buyer\_choice and Options*.

This is because the sales staff works with the buyer to decide and finalize choices and updates the list of options available to the buyers as well. This involves adding new rows, updating the rows and also deleting in case some option is discontinued.

Since sales staff is directly involved in the sale of a lot, he also manages the Sales table, for example adding an entry in the table whenever there is a new sale.

**8. Scheduled Job**

View\_Construction\_Progress

In the Eggshell builders case, The corporation wants to keep track of latest construction progress like overall process and estimated time of completion to directly update the buyers via website.

In order to have the latest updates available periodically, we have scheduled an oracle called View\_Construction\_Progress. This job displays the latest updates within the last 15 days from the Construction table and displays it. Currently the frequency of this job is set to a minute, but in a real world scenario, this will be run every week, to view latest updates.

**9. Indexes**

index\_sale\_date

index\_lot\_sub\_style

index\_sale\_date is created since in the Eggshell’s business, buyers can cancel their contracts within one year if the house is not completed. Thus the sale\_date will be frequently queried, we regard indexing it a more efficient way. index\_sale\_date is created because in reality, a buyer would like to see what lots are available for one specific subdivision and style. So indexing on these two columns helps improve the efficiency.

**10. De-normalization Instance**

There is a frequent search which needs a join between lot and style. Since a lot is tailored to a specific style and buyers might be both interested in lot information as well as style information, they always request to check both of them. So a potential denormalization instance could be to merge the lot table and style table.