# Object Description

## **Sequence:**

The Surrogate keys are widely applied for our database. It is a unique, automatically generated identifier for record in a table. It sacrifices the business meaning within the advantage of simplicity, consistency, and performance improvement due to more index by database. We use apply sequences to generate the surrogate keys for some tables as follows:

| **Sequence** | **Table** |
| --- | --- |
| bank\_id\_seq | Bank |
| construction\_project\_id\_seq | Construction\_Project |
| schoice\_id\_seq | Chosen\_Style |
| customer\_id\_seq | Customer |
| dchoice\_id\_seq | Decorator\_Choice |
| employee\_id\_seq | Employee |
| license\_no\_seq | Sales\_Representative |
| crew\_no\_seq | Construction\_Manager |
| escrow\_id\_seq | Escrow |
| lot\_id\_seq | Lot |
| option\_id\_seq | Option\_List |
| invoice\_id\_seq | Sale |
| district\_id\_seq | School\_District |
| subdivision\_id\_seq | Subdivision |

## **Views:**

ACCESSIBLE\_PROJECTS:

* This view provides relevant information on the status of their purchased lot. It provides headers like the Project ID, chosen style and elevation, sale date, construction start date, estimated completion date, completion percentage and project stage.
* As per requirements, the customer will only see project stage updates 1, 4 and 7. The current progress stage is always rounded down. Example, the project stage is currently at 3, the customer will see the Project Stage as 1.

PROJECT\_OPTIONS\_VIEW:

* This is a view that limits the decorator options the clients can view and choose from according to the current construction project stage.
* Options that have Stage\_no(x) indicate that this option cannot be selected at a Project\_Stage higher than itself.

CustomerSaleRecord:

This view restricts customers to access their own sale record only, which includes the following columns: invoice\_id, lot\_premium, date\_sold, financing\_method.

## **Procedures:**

Filter Options By Category:

This procedure takes option category as a parameter and filters the available options by category.

Create\_Sale\_Record:

This procedure takes in the mandatory information (customer id, sales rep license, bank id, lot id, financing method, agent id, deposit amt, reverse style, elevation chosen, house style and construction manager), creates and link all of the tables to the Sales table.

Cancel\_Sale\_Record:

This procedure accepts invoice\_id as the parameter and implements sale record deletion. It also outputs the amount of escrow refund according to the formula listed in the business rule document.

## **Functions:**

GET\_NEXT\_ELEVATION\_ID:

This function accepts p\_house\_style as a parameter and returns the current elevation\_id for the new elevation entry. It ensures that the elevation\_id increments properly for each house style.

GET\_NEXT\_ROOM\_ID:

This function accepts p\_house\_sylte as a parameter and returns the current room\_id for the new room entry, ensuring that the room\_id increments properly for each house style.

## **Package:**

Sale\_Operations:

This package mainly performs operations for the Sale table and consists of the following function and procedures:

| **Object** | **Name** | **Description** |
| --- | --- | --- |
| Function | calc\_refund | It calculates the refund based on the escrow deposit and project completion percentage. |
| Procedure | generate\_next\_id | It generates the next ID for any table's column. |
| Procedure | create\_sale\_record | It creates new records for a sale, including entries for escrow, construction\_project, and chosen\_style, while also generating the necessary IDs for these records |

## **Database Triggers**

SALE\_DEL\_CASCADE:

This trigger is executed before a sale record is deleted. It ensures that related records in Construction\_Project, Escrow, and Chosen\_Style linked to the deleted sale are also removed to prevent orphan records.

CP\_DEL\_CASCADE:

This trigger runs before a Construction\_Project record is deleted. It cascades the deletion to associated records in Task and Decorator\_Choice that rely on the deleted project.

HOUSE\_STYLE\_DEL\_CASCADE:

This trigger executes after a House\_Style record is deleted, removing corresponding Elevation and Room records linked to the deleted house style, maintaining consistency across related tables.

UPDATE\_STYLE\_SIZE:

The style\_size attribute in the House\_Style table is derived by summing the associated room size from the Room table. After a new entry is inserted or updated in the Room table, this trigger will sum all the room sizes with the house style name of the new entry and update the style\_size field in the House\_Style table correspondingly.

UPDATE\_LOT\_PREMIUM\_BY\_ELEVATION\_COST:

After a new elevation is inserted or updated in the Elevation table, this trigger will add the new elevation cost to the lot\_premium of the corresponding sale record in the Sale table.

UPDATE\_PROJECT\_PCT\_COMPLETE:

After a new task is inserted or updated in the Task table, this trigger will update the project\_pct\_complete of its parent construction project by the following rules:

* project\_pct\_complete = number of task completed / total number of tasks under a project \* 100
* Example:

Assume there are three tasks under one project (i.e. matched with the same project\_id). If one of the task’s task\_pct\_complete has achieved 100 while that of the others are less than 100, the project\_pct\_complete will equal to 33.33.

## **Scheduled Job**

YEARLY\_INACTIVE\_CUSTOMER\_CLEANUP:

It is designed to remove inactive customers from the database annually. It uses the stored procedure sp\_delete\_inactive\_customers, which deletes customers who have never made a sale. The job is scheduled to run once a year and enabled upon creation, ensuring that inactive customer records are cleaned up automatically on an annual basis, keeping the database current.

## **Roles:**

ROLE:CustomerRole: restricted access to certain table records and stage

GRANT to the following:

VIEW: CustomerSaleRecord: restrict customer to only visit his/her own purchase record from the sale table

TABLE: Option\_List, Decorator\_Choice, Construction\_project, Lot

ROLE:Employee: restricted access to most table records and stage

GRANT TO FOLLOWING TABLE:

School\_District,Subdivision,School,Elevation,Lot,Chosen\_Style,House\_Style,Room,Sale,Bank,

Option\_List, Escrow

## **Instance of De-normalization:**

## 

Option List Denormalization.

| Option ID | Stage | Description | Style |
| --- | --- | --- | --- |
| 1027 | 4 | Wire for Ceiling fan | Renaissance |
| 1027 | 7 | Wire for Ceiling fan | Renaissance |

1. This is a truncated version of the original table. As noted, the description is repeated multiple times. In the initial database design, a description table was split off and a composite key of (option\_id, style\_name) was created.
2. When we were doing triggers and views, we noticed that we were always joining the tables above together. The Option ID and Stage did not provide much usable information.
3. As most queries involving this table required a join, we decided to denormalize this table and recombine them into one.
4. Moreover, in this case, this particular table is a read-heavy table as any customer that wants to select options will have to query this table.

## **Alternate Index:**

We created an alternate index of Option Category for the Options table. As most customers will inadvertently query options list based on the categories, this will speed up query process. This is done simply by running

**CREATE INDEX idx\_option\_category ON option\_list(OPTION\_CATEGORY);**

Afterwards, we verify that the index has been created appropriately by running the following 2 select queries and comparing how they are retrieved.

| -- With index, this should show INDEX RANGE SCAN  EXPLAIN PLAN FOR  SELECT \* FROM option\_list WHERE option\_category = 'electrical';  SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);  -- Without index (force full table scan), this should show Table Access Full  EXPLAIN PLAN FOR  SELECT /\*+ FULL(option\_list) \*/ \* FROM option\_list WHERE option\_category = 'electrical';  SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);’ |
| --- |