Outdoor Center Application Express Documentation

Ashley Ng

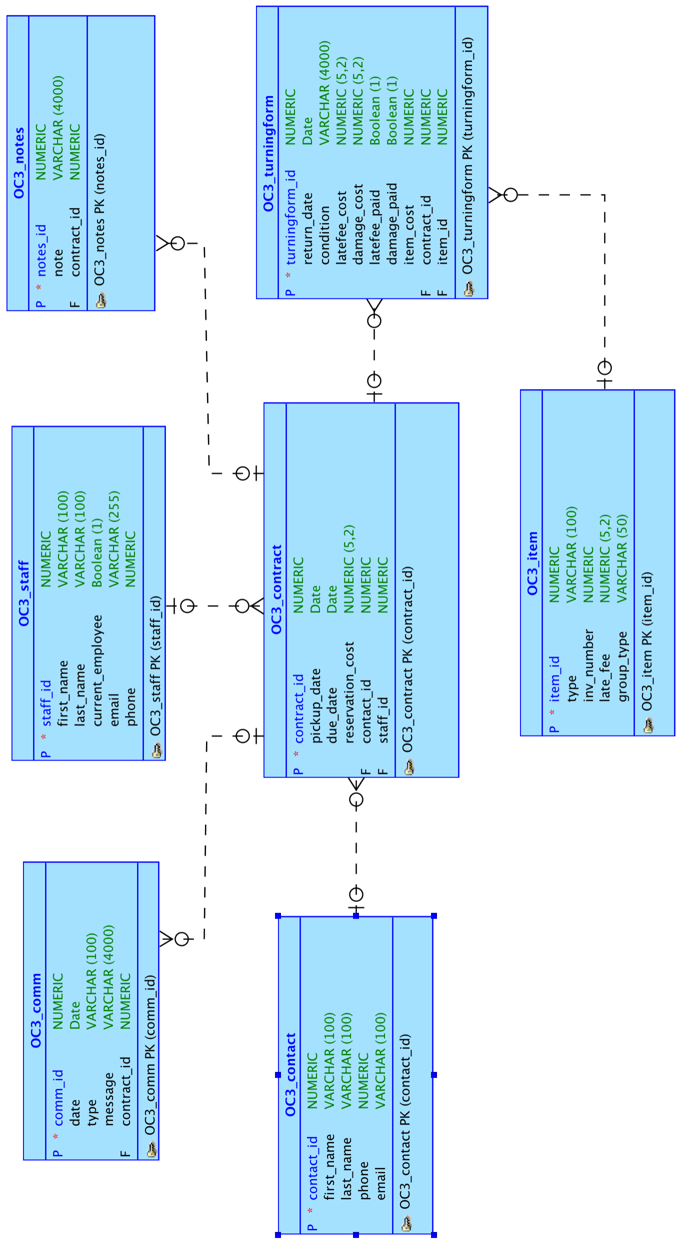
August 4, 2014

Abstract

This documentation explains the data model used,explains the SQL queries used, and triggers used for the Outdoor Center's online Database application on Oracle's Application Express (APEX).

Contents

1 Data Model


OC3_notes
P * notes_id NUMERIC
note VARCHAR (4000)
F contract_id NUMERIC
OC3_notes PK (notes_id)
OC3_turningform
P * turningform_id NUMERIC
return_date Date
condition VARCHAR (4000)
latefee_cost NUMERIC (5,2)
damage_cost NUMERIC (5,2)
latefee_paid Boolean (1)
damage_paid Boolean (1)
item_cost NUMERIC
F contract_id NUMERIC
F item_id NUMERIC
OC3_turningform PK (turningform_id)
OC3_staff
P * staff_id NUMERIC
first_name VARCHAR (100)
last_name VARCHAR (100)
current_employee Boolean (1)
email VARCHAR (255)
phone NUMERIC
OC3_staff PK (staff_id)
OC3_contract
P * contract_id NUMERIC
pickup_date Date
due_date Date
reservation_cost NUMERIC (5,2)
F contact_id NUMERIC
F staff_id NUMERIC
OC3_contract PK (contract_id)
OC3_item
P * item_id NUMERIC
type VARCHAR (100)
inv_number NUMERIC
late_fee NUMERIC (5,2)
group_type VARCHAR (50)
OC3_item PK (item_id)
OC3_comm
P * comm_id NUMERIC
date Date
type VARCHAR (100)
message VARCHAR (4000)
F contract_id NUMERIC
OC3_comm PK (comm_id)
OC3_contact
P * contact_id NUMERIC
first_name VARCHAR (100)
last_name VARCHAR (100)
phone NUMERIC
email VARCHAR (100)
OC3_contact PK (contact_id)

The data model consists of seven different tables; communication, staff, notes, contact, contract, turning form, and item as seen from image ().

1.1 OC3_comm Table

The OC3_comm table is for any form of communication between the Outdoor Center and the customer. It can be anything from, general emails and calls, to late calls (you're items are now considered late), to fee emails (you have either late fees and/or damage fees). The primary key is the comm_id, the date is when the communication occurred, the type is one of the things listed above (i.e late call, damage email, general call, general email), message is just what the communication was about, and the contract_id is just the foreign key to OC3_contract.

1.2 OC3_staff Table

The staff table will hold information on all staff that will work in the Outdoor Center. The reason for this table is so the application can dynamically list staff to attached to a contract instead of having to hard code lists into the application. The lists will just name staff that currently work there, which is denoted by the field current_employee.

1.3 OC3_notes Table

The notes table hold anything that needs to be know about the contract. This can vary from early pick-up times to not returning all their items in on time. This is so staff from other shifts know what is going on for the particular contract. This table has only 3 attributes, 2 of which are some sort of key. There is the primary key, notes_id as the primary key, and contract_id as the foreign key. Then there is the note attribute which will hold the notes.

1.4 OC3_contact Table

The contact table is information of all the customer that rent gear from the Outdoor Center; whether it's only once or every week. We have phone number and email attributes as a way to contact them incase something is late or something is damaged from their reservation. Right now the unique identifier (primary key) is contact_id, which is just a number. For implementation, this would be the student/staff's UTEID.

1.5 OC3_item Table

The item table is where information on all the equipment lives. This table will only change in the event that new equipment is bought or sold. The type attribute is the kind of item it is; 45 degree sleeping bag, backpack, 1-Person Tent, or LED lantern. The inv_number is a number given to the item. Every item is uniquely identified by the combination of the type and inv_number attributes. The late fee attribute is just what the item would cost of it was one day late. This makes calculating the late fee with the trigger a lot easier. Finally, the group type is just categorizing the items. For example, the sleeping group type might have 45 degree sleeping bags, 20 degree sleeping bags, air mattresses, and sleeping pads.

1.6 OC3_turningform Table

The turning form table is made for every item that is reserved. Initially, the only fields that will be filled out at the time of reservation is item_cost and the two foreign keys; contract_id and item_id. The rest of the attributes will be filled in either at the time the item is returned or once the item has been ok'd to be put back on the shelf.

1.7 OC3_contract Table

The contract table is the center piece to this data model, and ties everything together. The table has two date attributes; pickup date and due date. The reservation cost is just a number that will automatically be calculated with a trigger. Then each contract has a contract (or customer) and staff member who filled out the contract.

2 Home Page

The Home page is meant to be something staff will only see, and contains important information that they need on a daily basis. This page contains four different tables; late contracts table, contracts out this week table, contracts due today table, and the contracts with fees table.

2.1 Late contracts table

The "late contracts" table's SQL is as follows

Listing 1: Late Contracts query

```
1 SELECT contract_id, pickup_date, due_date, reservation_cost,
2 OC3_contact_id, OC3_staff_id,
3 cc.last_name || ’, ’ || cc.first_name as "Name", cc.phone, cc.email
4 FROM OC3_contract c
5 JOIN OC3_turningform t
6 ON c.cotract_id = t.OC3_contract_id
7 JOIN OC3_contact cc
8 ON cc.contact_id = c.OC3_contact_id
9 WHERE (t.return_date is NULL AND c.due_date < SYSDATE);
```

The query joins three tables together to get all the information. The way we get only the late contracts is from the where clause on line 9. The return date should be null because it will only have a value when the item is returned and the date is inputed by staff, and the SYSDATE would be larger than the due date.

2.2 Contracts out this week table

The next table is the “Contracts out this week” table. Sometimes we have very busy weekend where we almost clear out our shelves. On these kind of weekends we like to pull the gear ahead of time. So that way when the customer shows up to pick up their stuff, all they have to do is sign they pick it up, and they’re on their way. The query for the table is as follows

Listing 2: Contracts out this week query

```
1 SELECT contract_id, pickup_date, due_date, reservation_cost,
2 OC3_contact_od, OC3_staff_id,
3 cc.last_name || ’, ’ || cc.first_name as "Name"
4 FROM OC3_contract cr
5 JOIN OC3_contact cc
6 ON cr.OC3_contact_id = cc.contact_id
7 WHERE pickup_date BETWEEN SYSDATE AND next_day(SYSDATE, ’FRIDAY’);
```

This is a pretty simple query, from line 7 we just get all the contract that are between the current day and the upcoming Friday.

2.3 Contracts due today Table

The next table on the home page is the contract that are due today. From line 10 of the query, it just returns contracts that have due date equal to the system date.

Listing 3: Contracts due today query

```
 1  SELECT contract_id, pickup_date, due_date, reservation_cost,
 2  OC3_contact_id, OC3_staff_id,
 3  cc.last_name || ', ' || cc.first_name as "Name"
 4  FROM OC3_contract cr
 5  JOIN OC3_contact cc
 6  ON cr.OC3_contact_id = cc.contact_id
 7  WHERE due_date = to_date(SYSDATE, 'DD-MON-YY');
```

2.4 Contracts with Fees Table

The final table on the home page is the Contracts with fees query. This query has quite a long where clause on lines 10 and 11. The query checks to see if the cost attributes are more than zero, and if they paid attributes are 'N' for no. It's joined together with a where clause because some contracts don't have a damage fee and others don't have a late fee.

Listing 4: Contracts with fees query

```
 1  SELECT contract_id, pickup_date, due_date, reservation_cost,
 2  OC3_contact_id, OC3_staff_id,
 3  '$' || (NVL(t.damage_cost, 0) + NVL(t.latefee_cost, 0)) as "Fee",
 4  cc.last_name || ', ' || cc.first_name as "Name", cc.phone, cc.email
 5  FROM OC3_contact c
 6  JOIN OC3_turningform t
 7  ON c.contract_id = t.OC3_contract_id
 8  JOIN OC3_contact cc
 9  ON cc.contact_id = c.OC3_contact_id
10  WHERE ((t.damage_paid = 'N' AND t.damge_cost > 0) OR
11  (t.latefee_cost > 0 AND t.latefee_paid = 'N');
```

3 Contacts Page

The contracts page is just an interactive report where there is a search bar present at the top of the table. This search bar only searches through the columns that are present. The interactive report is sorted by pickup_date in descending order. If you click on the magnifying glass image in the far left column, it takes you to what is called the “Contracts Details Page”. This page just shows all the information pertaining to only a single contract.

4 Contacts Detail Page

On the contracts detail page, there are four distinct sections; the contract detail, turning form, communication, and notes. Each of these tables has a where clause that looks like the following

Listing 5: where clause

```
WHERE contract_id = :P15_id;
```

The :P15_id is a hidden item on the page, and the contract_id gets stored in this hidden item when you click on the magnifying glass from the previous page (the interactive report).

5 Item Availability Pages

This is the only public page for the application. This page allows customer to select a date they would like to rent gear, and see what equipment is available to rent. What was done on this page was similar to a setValue type of action, but since APEX doesn’t support setValue on date pickers, I had to figure out a way around this. Before, I delve into that, let look at the query

Listing 6: Item availability Table

```
SELECT * from OC3_item where item_id not in
(SELECT item_id
FROM OC3_item i
JOIN OC3_turningform t
ON i.item_id = t.OC3_item_id
JOIN OC3_contract c
ON c.contract_id = t.OC3_contract_id
WHERE (to_date(:P16_DATE, 'DD-MON-YY') > c.pickup_date AND
to_date(:P16_DATE, 'DD-MON-YY') < c.due_date));
```

6 Triggers

There are currently two triggers for this application, which are both compound triggers. One calculates the reservation cost depending on what is entered in the turning form, and the other calculates the late fee when the turning form return_date field is updated.

6.1 Reservation Cost Trigger

The first one is the trigger that updates the reservation cost.

Listing 7: reservation cost trigger

```
CREATE OR REPLACE TRIGGER update_reservation_cost
FOR INSERT OR UPDATE OF item_cost
ON OC3_turningform
COMPOUND TRIGGER
        temp_id NUMBER;
BEFORE EACH ROW IS
BEGIN
        temp_id := :new.OC3_contract_id;
END BEFORE EACH ROW;
AFTER STATEMENT IS
        res_cost        NUMBER(5,2);
BEGIN
        SELECT SUM(item_cost)
        INTO res_cost
        FROM OC3_turningform
        WHERE OC3_contract_id = temp_id
        GROUP BY OC3_contract_id;
```

```
18
19          UPDATE OC3_contract
20          SET reservation_cost = res_cost
21          WHERE contract_id = temp_id;
22  END AFTER STATEMENT;
23  END;
```

This trigger fires whenever an insert or update of the attribute item_cost on the OC3_turningform table. What the trigger does is it first stores the contract_id associated with the turning form on line 8. After the statement is complete, the trigger does a the sum aggregate function on the item_cost, and stores it in a variable, for all the turning forms associated with the contract_id we stored earlier. Once that is done, the trigger does an update on the contract and puts in the sum we just calculated.

6.2 Late fee Trigger

The next trigger is the late fee calculation trigger. This trigger fires when you update the return_date attribute on the OC3_turningform table. The trigger first stores two values; the return date (that was just updated), and the turning form id. It then does a select statement on the OC3_item to get the late fee and stores that in a variable also. In the after statement, the trigger does a select statement and stores the due date variable for the corresponding turning form id we stored earlier. It then does calculations on line 29. Here, it subtracts how many days it was late and multiplies it by the late fee cost. After that, it updates the latefee_cost attribute on the OC3_turningform table for the appropriate turning from id

Listing 8: late fee calculation trigger

```
1   CREATE OF REPLACE TRIGGER late_fee_calculation
2   FOR UPDATE OF return_date
3   ON OC3_turningform
4   COMPOUND TRIGGER
5           late_total      NUMBER(5,2);
6           due_date_var    DATE;
7           return_date_var DATE;
```

```
        temp_id NUMBER;
        late_fee_var    NUMBER(5,2);
BEFORE EACH ROW IS
BEGIN
        return_date_var := :new.return_date;
        temp_id := :new.turningform_id;

        SELECT late_fee
        INTO late_fee_var
        FROM OC3_item
        WHERE item_id = :new.OC3_item_id;
END BEFORE EACH ROW;
AFTER STATEMENT IS
BEGIN
        SELECT ct.due_date
        INTO due_date_var
        FROM OC3_contract ct
        JOIN OC3_turningform t
        ON ct.contract_id = t.OC3_contract_id
        WHERE turningform_id = temp_id;

        late_total := (return_date_var - due_date_var) * late_fee_var;

        UPDATE OC3_turningform
        SET latefee_cost = late_total
        WHERE turningform_id = temp_id;
END AFTER STATEMENT;
END;
```

7 Plug-ins

Mini-Calendar v.2 [plug-in]

8 Bugs and Beta features

The APEX version that was used is APEXEA 5, where the EA stands for early access. So this application is not a full product, and the time of completion of the Outdoor Center Database, the full product was predicted to be released in late August 2014.

One of the bugs present is that you are unable to attach a URL link in a column. For example, I was unable to link the customers emails on the front page. I would of like for it to be able to open a new message on the computers email services, but for right now, you have to copy and paste.

Currently, there are two “Item Availability” tabs, one with “with Date and Group” appended at the end. The item availability tab with date and group, again allows you to select a date to see what items are available, but it also lets you select the kind of equipment you’re looking for. That way the customer is not scrolling through the hundreds of individual equipment that are currently available. The bug with this tab, is that the transition from an existing search to another isn’t perfect. If you already have a date selected and a group selected, and decided you want a different day; it resets the whole search instead of just updating the date field.

The Outdoor Center does these bi-monthly inventory check, just to make sure everything is present. If something is missing or in repair, they mark it off the reservation pages (which are currently excel documents). I would of like to have this ”availability” check on the OC3_item table. Where they just check a box if something is still on the shelf or uncheck it if it’s in repair or missing. It seems like there is a small error with APEX where I try to create a tabular form and change the availability column to a checkbox. There is a transaction rollback error, but this only occurs after I’ve changed the column to checkbox. Otherwise, it completely fine before that. Once the official version of APEX 5 is released, and hopefully this bug is fixed, I’ll update the applications to have this features; which is very important for the use of this application,

Another issue that has come up on interactive reports is that any sort of filtering can only be done on the interactive report. You cannot put in group by, order by, or where clause in the query. The interactive report just ignores it. This would become an issue on the contracts page where all of the contracts will be displayed. I would of like to only display the contracts that have pick up dates and return dates that are in the range SYSDATE - 7 and SYSDATE + 14.