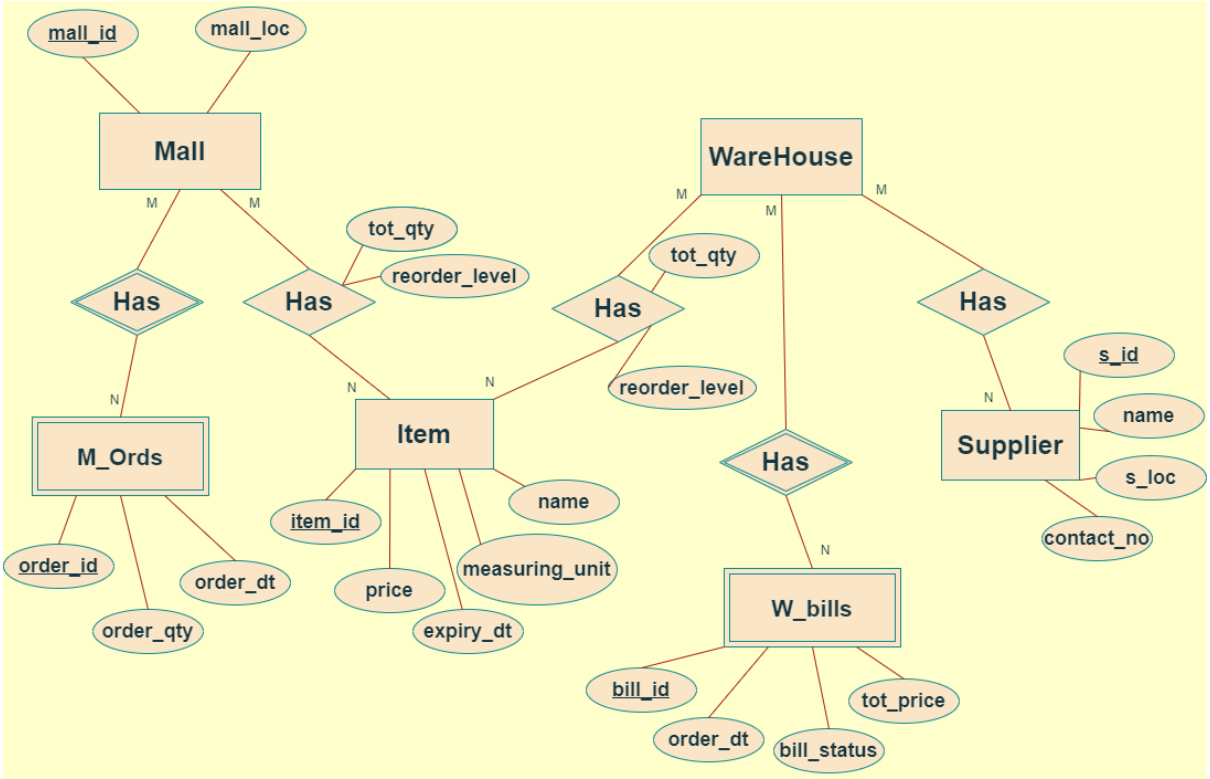
MALL BANEGA SMALL

We are creating a database for a mall for its inventory like for D-mart mall. In that we are considering different items in it. There are different malls and all are taking items from different ware house. In mall if items are ge□ng low from its reorder-level then it will order from warehouse and also in ware house if items are getting low from its reorder-level then it will order from suppliers. If Mall is ordering from warehouse and ware house is ordering from supplier the record about that will be stored. All Details about different supplier will store in new table. We also store details about different items like its id, name, price, expiry date in different table. We create following E R diagram :



The resulting tables are :

- warehouse (warehouse_id, warehouse_loc)
- mall (mall_id, mall_loc)
- item (item_id, item_name, expiry_dt,measuring_unit, price)
- supplier (s_id, name, s_loc, contact_no)
- warehouse-item (warehouse_id, item_id, tot_qty,reorder_level)
- mall_item (mall_id, item_id, tot_qty, reorder_level)
- warehouse_supp_details(warehouse_id, item_id, s_id)
- warehouse-bills (bill_id, order_dt, bill_status,tot_price, warehouse_id, item_id, s_id)
- mall-supp-details (mall_id, item_id, warehouse_id)
- mall-orders (order_id, order_qty, order_dt, mall_id,item_id, warehouse_id)

The following is the code for creating all the necessary tables:

```
create table warehouse(warehouse_id number(5) constraint pk_i_id primary key,warehouse_loc varchar2(20) constraint nn_i_loc not null);


create table mall(mall_id number(5) constraint pk_m_id primary key,mall_loc varchar2(20) constraint nn_m_loc not null);


create table item(item_id number(5) constraint pk_it_id primary key, it_name varchar2(20) not null,measuring_unit varchar2(20),price number(10),expiry_dt date,constraint ck_m_u check(upper(measuring_unit) in ('TONNES','KILOGRAMS','GRAMS','LITRES','NUMBER')));


create table supplier(s_id number(5) constraint pk_s_id primary key,s_name varchar2(20),s_loc varchar2(20) not null,contact_no number(10) constraint ck_c_no check(length(contact_no)=10));


create table warehouse_items(warehouse_id number(5) constraint fk_i_id_2 references warehouse(warehouse_id),item_id number(5) constraint fk_it_id_1 references item(item_id),available_qty number(10),reorder_level number(10),constraint pk_i_it primary key(warehouse_id,item_id));


create table mall_items(mall_id number(5) constraint fl_m_id_2 references mall(mall_id),item_id number(5) constraint fk_it_id_2 references item(item_id),available_qty number(10),reorder_level number(10),constraint pk_m_it primary key(mall_id,item_id));


create table warehouse_supplier_details(warehouse_id number(5) constraint fk_i_id_3 references warehouse(warehouse_id),item_id number(5) constraint fk_it_id_3 references item(item_id),s_id number(5) constraint fk_s_id_1 references supplier(s_id),constraint pk_i_s primary key(warehouse_id,item_id,s_id));


create table warehouse_bills(bill_id number(5) constraint pk_bill_id primary key,warehouse_id number(5),s_id number(5),item_id number(5),order_qty number(10),order_dt date,tot_price number(10),bill_status varchar2(20) constraint ck_b_s check(upper(bill_status) in ('PAID','UNPAID')),constraint fk_in_s_it foreign key(warehouse_id,s_id,item_id) references warehouse_supplier_details(warehouse_id,s_id,item_id));


create table mall_supplier_details(mall_id number(5) constraint fl_m_id_1 references mall(mall_id),warehouse_id number(5) constraint fk_i_id_1 references
```

```
warehouse(warehouse_id),item_id number(5) constraint fk_it_id_4 references
item(item_id),constraint pk_m_i_it primary key(mall_id,warehouse_id,item_id));

create table mall_orders(order_id number(5) constraint pk_o_id primary key,order_qty
number(10),order_dt date,mall_id number(5),item_id number(5),warehouse_id
number(5),constraint fk_m_i_id foreign key(mall_id,warehouse_id,item_id) references
mall_supplier_details(mall_id,warehouse_id,item_id));
```

The following is the code for inserting the values in some of the above tables :

```
INSERT INTO warehouse VALUES (1, 'Warehouse A');
INSERT INTO warehouse VALUES (2, 'Warehouse B');
INSERT INTO warehouse VALUES (3, 'Warehouse C');

INSERT INTO mall VALUES (1, 'Mall X');
INSERT INTO mall VALUES (2, 'Mall Y');
INSERT INTO mall VALUES (3, 'Mall Z');

INSERT INTO item VALUES (1, 'Rice', 'Kilograms', 10, TO_DATE('2024-12-31', 'YYYY-MM-DD'));
INSERT INTO item VALUES (2, 'Sugar', 'Kilograms', 20, TO_DATE('2025-06-30', 'YYYY-MM-DD'));
INSERT INTO item VALUES (3, 'Salt', 'Kilograms', 5, TO_DATE('2024-09-30', 'YYYY-MM-DD'));

INSERT INTO supplier VALUES (1, 'Supplier A', 'Location X', 1234567890);
INSERT INTO supplier VALUES (2, 'Supplier B', 'Location Y', 9876543210);
INSERT INTO supplier VALUES (3, 'Supplier C', 'Location Z', 4567890123);

INSERT INTO warehouse_items VALUES (1, 1, 100, 20);
INSERT INTO warehouse_items VALUES (1, 2, 200, 50);
INSERT INTO warehouse_items VALUES (2, 2, 150, 40);

INSERT INTO mall_items VALUES (1, 1, 50, 10);
INSERT INTO mall_items VALUES (2, 2, 100, 30);
```

```
INSERT INTO mall_items VALUES (3, 3, 80, 15);

INSERT INTO warehouse_supplier_details VALUES (1, 1, 1);
INSERT INTO warehouse_supplier_details VALUES (1, 2, 2);
INSERT INTO warehouse_supplier_details VALUES (2, 2, 3);

INSERT INTO mall_supplier_details VALUES (1, 1, 1);
INSERT INTO mall_supplier_details VALUES (2, 1, 2);
INSERT INTO mall_supplier_details VALUES (3, 2, 3);
```

The following is the code for trigger which will be triggered when the mall_items table is updated:

```
create or replace trigger m1
after update on mall_items for each row
declare
                a mall_items.available_qty%type;

  pragma autonomous_transaction;
        temp_ord_qty number;temp_r_lvl number;
begin
        select :new.available_qty,reorder_level into temp_ord_qty,temp_r_lvl from mall_items
where item_id=:new.item_id and mall_id=:new.mall_id;
        if temp_ord_qty<temp_r_lvl then
     generatemallorder(:new.mall_id,:new.item_id);
                        delete from mall_items where mall_id=m and item_id=i;
                        insert into mall_items values(m,i,a+reorder,reorder);
        end if;
        commit;
end;
```

The following is the code for trigger which will be triggered when a mall orders a warehouse :

```
create or replace trigger t4
after insert on mall_orders for each row
declare
   pragma autonomous_transaction;
        m mall_orders.order_qty%type;
        a warehouse_items.available_qty%type;
        r warehouse_items.reorder_level%type;
begin
        select available_qty,reorder_level into a,r from warehouse_items where
warehouse_id=:new.warehouse_id and item_id=:new.item_id;
        if a<:new.order_qty then dbms_output.put_line('Waiting for stock');
        else
        if a-:new.order_qty<r then generatewareorder(:new.warehouse_id,:new.item_id);
                end if;
                update warehouse_items set available_qty=a-:new.order_qty where
warehouse_id=:new.warehouse_id and item_id=:new.item_id;
        end if;
        commit;
end t4;
```

The following is the code for the trigger which will be triggered after insert on warehouse bills:

```
create or replace trigger t1
after insert on warehouse_bills for each row
declare
   pragma autonomous_transaction;
begin
        updatewarehousestock(:new.warehouse_id,:new.item_id);
```

```
        commit;
end t1;
```

The following is the code for the procedure generatemall order which basically inserts a value into the mall_orders table :

```
create or replace procedure generatemallorder(m mall.mall_id%type,i item.item_id%type)
is
reorder mall_items.reorder_level%type;
o mall_orders.order_id%type;
w warehouse.warehouse_id%type;
begin
select reorder_level into reorder from mall_items where mall_id=m and item_id=i;
select max(order_id) into o from mall_orders;
o:=nvl(o,0)+1;
select warehouse_id into w from mall_supplier_details where mall_id=m and item_id=i;
insert into mall_orders values(o,reorder,sysdate,m,i,w);
end generatemallorder;
```

The following is the package which consists the code for function and procedures:

```
create or replace package db_mall
as
title constant varchar2(50):='Mall Banega Small';
function calprice(it item.item_id%type,qty warehouse_bills.order_qty%type) return number;
procedure generatemallorder(m mall.mall_id%type,i item.item_id%type);
procedure generatewareorder(w warehouse.warehouse_id%type,i item.item_id%type);
procedure updatewarehousestock(w warehouse.warehouse_id%type,i item.item_id%type);
end db_mall;
```

```
create or replace package body db_mall
as
function calprice(it item.item_id%type,qty warehouse_bills.order_qty%type) return number
as
a number(5);
begin
select price into a from item where item_id=it;
return a*qty;
exception
when no_data_found then return 0;
end calprice;
procedure generatemallorder(m mall.mall_id%type,i item.item_id%type)
is
reorder mall_items.reorder_level%type;
o mall_orders.order_id%type;
w warehouse.warehouse_id%type;
begin
select reorder_level into reorder from mall_items where mall_id=m and item_id=i;
select max(order_id) into o from mall_orders;
o:=nvl(o,0)+1;
select warehouse_id into w from mall_supplier_details where mall_id=m and item_id=i;
insert into mall_orders values(o,reorder,sysdate,m,i,w);
end generatemallorder;
procedure generatewareorder(w warehouse.warehouse_id%type,i item.item_id%type)
is
reorder warehouse_items.reorder_level%type;
o warehouse_bills.bill_id%type;
```

```
s warehouse_supplier_details.s_id%type;
begin
select reorder_level into reorder from warehouse_items where warehouse_id=w and
item_id=i;
select max(bill_id) into o from warehouse_bills;
select max(s_id) into s from warehouse_supplier_details where warehouse_id=w and
item_id=i;
o:=nvl(o,0)+1;
insert into warehouse_bills values(o,w,i,s,reorder,sysdate,calprice(i,reorder),'PAID');
end generatewareorder;
procedure updatewarehousestock(w warehouse.warehouse_id%type,i item.item_id%type)
is
r warehouse_items%rowtype;
o warehouse_bills.order_qty%type;
b warehouse_bills.bill_status%type;
begin
select order_qty,bill_status into o,b from warehouse_bills where warehouse_id=w and
item_id=i;
if upper(b)='PAID' then
update warehouse_items set available_qty=nvl(available_qty,0)+o where warehouse_id=w
and item_id=i;
end if;
end updatewarehousestock;
end db_mall;
```

HOW IT WORKS :

- Whenever an item table is updated , it is checked that if the resulting available qty is less than reorder level for that item or not. If it is then it will order the warehouse.
- Now when the warehouse will receive the order, it would check whether the order will be accepted or not.
  If the order is accepted then it would update its item table.
  Now if the warehouse item qty is less than the reorder level then it would order the supplier.
  Once the bill is paid then it would again update its item table.