AddtoInventory

create or replace

PROCEDURE addToInventory (qtyy IN INT, typeInput IN VARCHAR, orderID IN INT)

IS

x INT;

xyy varchar(30);

BEGIN

dbms\_output.put\_line('start of add to inventory');

update inventory set qty=qty+qtyy where type=typeInput;

select o.qty into x from orders\_fake o where o.oid=orderId;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :inventory update '||x||' of type '||typeInput);

deliver (x, typeInput , orderId);

END;

ASSEMBLECOMPONENT

create or replace

PROCEDURE assembleComponents (qtyy IN INT, orderId IN INT, moduleId IN INT)

IS

x INT;

y INT;

z VarCHAR(5);

xyy varchar(30);

BEGIN

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line(xyy|| ' :start the assemblecomponents procedures');

update\_orders\_table();

select min(o.oid) into y from orders\_fake o where o.to\_be\_delivered>0;

select o.qty into x from orders\_fake o where o.oid=y;

select o.type into z from orders\_fake o where o.oid=y;

if (moduleId=1) then

Update components\_fake set qty=qty-qtyy where cid=1 OR cid=2;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy|| ' :number of components 1 and 2 decreased by '|| qtyy);

Update modules set qty=qty+qtyy where mid=1;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :'||qtyy||' module type 1 was made...total number: '|| qtyy);

checkModules (x, z);

End if;

if (moduleId=2) then

Update components\_fake set qty=qty-qtyy where cid=3 OR cid=4;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :number of components 3 and 4 decreased by '|| qtyy);

Update modules set qty=qty+qtyy where mid=2;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :'||qtyy||' module type 2 was made...total number: '|| qtyy);

checkModules (x, z);

End if;

if (moduleId=3) then

Update components\_fake set qty=qty-qtyy where cid=5 OR cid=6;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :number of components 5 and 6 decreased by '|| qtyy);

Update modules set qty=qty+qtyy where mid=3;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :'||qtyy||' module type 3 was made...total number: '|| qtyy);

checkModules (x, z);

End if;

if (moduleId=4) then

Update components\_fake set qty=qty-qtyy where cid=7 OR cid=8;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :number of components 7 and 8 decreased by '|| qtyy);

Update modules set qty=qty+qtyy where mid=4;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :'||qtyy||' module type 4 was made...total number: '|| qtyy);

checkModules (x, z);

End if;

if (moduleId=5) then

Update components\_fake set qty=qty-qtyy where cid=9 OR cid=10;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||'number of components 9 and 10 decreased by '|| qtyy);

Update modules set qty=qty+qtyy where mid=5;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :'||qtyy||' module type 5 was made...total number: '|| qtyy);

checkModules (x, z);

end if;

END;

ASSEMBLEMODULE

create or replace

PROCEDURE assembleModules (qtyy IN INT, typeInput IN VARCHAR, orderId IN INT)

IS

xyy varchar(30);

BEGIN

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line (xyy||' :start of assemblemodules');

if (typeInput='A') then

update modules set qty=qty-qtyy where mid=1 OR mid=3 OR mid=5;

end if;

if (typeInput='B') then

update modules set qty=qty-qtyy where mid=1 OR mid=4 OR mid=5;

end if;

if (typeInput='C')

then update modules set qty=qty-qtyy where mid=2 OR mid=3 OR mid=5;

end if;

if (typeInput='D')

then update modules set qty=qty-qtyy where mid=2 OR mid=4 OR mid=5;

end if;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy|| ' :'||qtyy || ' new mipads of type ' || typeInput || ' has been made' );

makeNewMipads (qtyy, typeInput, orderId);

END;

CHECKCOMPONENTS

create or replace

PROCEDURE assembleModules (qtyy IN INT, typeInput IN VARCHAR, orderId IN INT)

IS

xyy varchar(30);

BEGIN

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line (xyy||' :start of assemblemodules');

if (typeInput='A') then

update modules set qty=qty-qtyy where mid=1 OR mid=3 OR mid=5;

end if;

if (typeInput='B') then

update modules set qty=qty-qtyy where mid=1 OR mid=4 OR mid=5;

end if;

if (typeInput='C')

then update modules set qty=qty-qtyy where mid=2 OR mid=3 OR mid=5;

end if;

if (typeInput='D')

then update modules set qty=qty-qtyy where mid=2 OR mid=4 OR mid=5;

end if;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy|| ' :'||qtyy || ' new mipads of type ' || typeInput || ' has been made' );

makeNewMipads (qtyy, typeInput, orderId);

END;

CHECKMODULES

create or replace

PROCEDURE checkModules ( qtyy IN INT, mipadType IN VARCHAR )

IS

a INT;

b INT;

c INT;

d INT;

e INT;

x Varchar(5);

y INT;

xx INT;

yy INT;

z INT;

xyy varchar(30);

BEGIN

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line(xyy||' :start of checkmodules procedures');

select m.qty into a from modules m where m.mid=1;

select m.qty into b from modules m where m.mid=2;

select m.qty into c from modules m where m.mid=3;

select m.qty into d from modules m where m.mid=4;

select m.qty into e from modules m where m.mid=5;

select min(o.oid) into y from orders\_fake o where o.to\_be\_delivered>0;

select o.qty into z from orders\_fake o where o.oid=y;

select o.type into x from orders\_fake o where o.oid=y;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

if (mipadType='A') then

if (a>=z AND c>=z AND e>=z) then assembleModules (z, 'A', y);

else DBMS\_OUTPUT.put\_line(xyy|| ' :we need more modules');

end if;

elsif (mipadType='B') then

if (a>=z AND d>=z AND e>=z) then assembleModules (z, 'B', y);

else DBMS\_OUTPUT.put\_line(xyy|| ' :we need more modules');

end if;

elsif (mipadType='C') then

if (b>=z AND c>=z AND e>=z) then assembleModules (z, 'C', y);

else DBMS\_OUTPUT.put\_line(xyy||' :we need more modules');

end if;

elsif (mipadType='D') then

if (b>=z AND d>=z AND e>=z) then assembleModules (z, 'D', y);

else DBMS\_OUTPUT.put\_line(xyy ||' :we need more modules');

end if;

end if;

END;

DELIVER

create or replace

PROCEDURE deliver (qtyy IN INT, typeInput IN VARCHAR, orderId IN INT)

IS

x int;

xyy varchar(30);

BEGIN

DBMS\_OUTPUT.put\_line('start of deliver function');

select i.qty into x from inventory i where i.type=typeInput;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :we have enough mipads in our inventory');

update inventory set qty=qty-qtyy where type=typeInput;

INSERT into orderdelivery values (orderid, typeInput, qtyy, 0);

update orders\_fake set to\_be\_delivered=0 where oid=orderid;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :delivering '|| qtyy ||' type ' || typeInput || ' mipads to group C');

DBMS\_OUTPUT.put\_line('setting status of delivered mipads to sold');

setSaleStatus (qtyy , typeInput,orderId);

END;

MAKENEWMIPADS

create or replace

PROCEDURE makeNewMipads (qtyy IN INT, typeInput IN VARCHAR, orderId IN INT)

IS

a INT;

b INT;

x INT;

xyy varchar(30);

BEGIN

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

select max(m.serial\_no) into a from mipad m where m.serial\_no>0 ;

DBMS\_OUTPUT.put\_line(xyy||' :last mipad has serial-no: ' ||a);

for b IN 1..qtyy LOOP

x:=a+b;

INSERT INTO Mipad values (x, typeInput, 'n', 'n');

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :A mipad with serial number of ' ||x|| ' has been made');

testMipad (x);

END LOOP;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy|| ' :All the tests were done successfully');

addToInventory(qtyy, typeInput, orderId);

END;

ORDERFROMGROUPA

create or replace

PROCEDURE orderFromGroupA ( qtyy IN INT, compId IN INT, suppId IN INT)

IS

xyy varchar(30);

BEGIN

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line(xyy||' :start of order from A procedures');

if (compId=1 OR compId=3 OR compId=5 OR compId=7 OR compId=9) then

INSERT INTO dcvan.order\_hub VALUES(compId,'groupB', qtyy);

else

INSERT INTO dcvan.order\_hub VALUES(compId,'groupB', qtyy);

end if;

END;

PROCINVENTORY

create or replace

PROCEDURE procInventory (qty IN INT, typeInput IN VARCHAR, orderId IN INT)

AS

x int;

y int;

xyy varchar(30);

BEGIN

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :start of ProcInventory function');

select i.qty into x from inventory i where i.type=typeInput;

if (qty<=x) then

deliver( qty , typeInput , orderId );

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line (xyy||' :deliver started');

elsif (qty>x) then

y:=qty-x;

dbms\_output.put\_line(xyy ||' :searchMipadTable started, starting to search mipad table');

searchMipadTable(y,typeInput,orderId);

end if;

END;

SEARCHCOMPONENTSTABLE

create or replace

PROCEDURE searchComponentsTable ( qtyy IN INT, typeInput IN VARCHAR, orderId IN INT, moduleId IN INT)

IS

a INT;

b INT;

c INT;

d INT;

e INT;

f INT;

g INT;

h INT;

I INT;

j INT;

x INT;

y INT;

z INT;

xyz INT;

xyy varchar(30);

BEGIN

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line(xyy||' :start of search component table');

select c.qty into a from components\_fake c where c.cid=1;

select c.qty into b from components\_fake c where c.cid=2;

select c.qty into c from components\_fake c where c.cid=3;

select c.qty into d from components\_fake c where c.cid=4;

select c.qty into e from components\_fake c where c.cid=5;

select c.qty into f from components\_fake c where c.cid=6;

select c.qty into g from components\_fake c where c.cid=7;

select c.qty into h from components\_fake c where c.cid=8;

select c.qty into i from components\_fake c where c.cid=9;

select c.qty into j from components\_fake c where c.cid=10;

select min(o.qty) into xyz from orders\_fake o where o.to\_be\_delivered>0;

if (moduleId=1) then

if(a>=qtyy AND b>=qtyy)then

assembleComponents (qtyy, orderId, 1);

End if;

x:=qtyy-a;

if (x>0) then

orderFromGroupA ( xyz, 1, 1);

end if;

x:=qtyy-b;

if (x>0) then

orderFromGroupA (xyz,2,2);

end if;

end if;

if (moduleId=2) then

if(c>=qtyy AND d>=qtyy)then

assembleComponents (qtyy, orderId, 2) ;

End if;

x:=qtyy-c;

if (x>0) then

orderFromGroupA (xyz,3,1);

end if;

x:=qtyy-d;

if (x>0) then

orderFromGroupA (xyz,4,2);

end if;

end if;

if (moduleId=3) then

if(e>=qtyy AND f>=qtyy)then

assembleComponents (qtyy, orderId, 3);

End if;

x:=qtyy-e;

if (x>0) then

orderFromGroupA (xyz,5,1);

end if;

x:=qtyy-f;

if (x>0) then

orderFromGroupA (xyz,6,2);

end if;

end if;

if (moduleId=4) then

if(g>=qtyy AND h>=qtyy)then

assembleComponents (qtyy, orderId, 4) ;

end if;

x:=qtyy-g;

if (x>0) then

orderFromGroupA (xyz,7,1);

end if;

x:=qtyy-h;

if (x>0)then

orderFromGroupA (xyz,8,2);

end if;

end if;

if (moduleId=5) then

if(i>=qtyy AND j>=qtyy) then

assembleComponents (qtyy, orderId, 5);

x:=qtyy-i;

end if;

if (x>0) then

orderFromGroupA (xyz,9,1);

end if;

x:=qtyy-j;

if (x>0) then

orderFromGroupA (xyz,10,2);

end if;

end if;

END;

SEARCHMIPADTABLE

create or replace

PROCEDURE searchMipadTable (qtyWeNeed IN INT, typeInput IN VARCHAR, orderId IN INT)

IS

x int;

b int;

c int;

xyy varchar(30);

BEGIN

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :start of searchMipadTable function');

select count(m.serial\_no) into x from mipad m where m.type=typeInput AND m.test\_result='s' AND m.sale\_status='fs';

if (x>=qtyWeNeed) then

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line(xyy||' :start to add to inventory');

addToInventory(qtyWeNeed, typeInput, orderId);

elsif (x<qtyWeNeed) then

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line(xyy||' :start to search module table');

b:=qtyWeNeed - x;

searchModulesTable (b, typeInput, orderId);

end if;

END;

SEARCHMODULETABLE

create or replace

PROCEDURE searchModulesTable (qty IN INT, typeInput IN VARCHAR, orderId IN INT)

IS

a int;

b int;

c int;

d int;

e int;

x int;

y int;

xyy varchar(30);

BEGIN

select m.qty into a from modules m where m.mid=1;

select m.qty into b from modules m where m.mid=2;

select m.qty into c from modules m where m.mid=3;

select m.qty into d from modules m where m.mid=4;

select m.qty into e from modules m where m.mid=5;

if (typeInput='A') then

begin

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line(xyy||' :start the search module table');

if ( a>=qty AND c>=qty AND e>=qty ) then

assembleModules (qty, 'A', orderId);

else

begin

y:=qty-a;

if (y>0) then searchComponentsTable (y, 'A', orderId, 1);

end if;

y:=qty-c;

if (y>0) then searchComponentsTable (y, 'A', orderId, 3);

end if;

y:=qty-e;

if (y>0) then searchComponentsTable (y, 'A', orderId, 5);

end if;

end;

end if;

end;

elsif (typeInput='B') then

begin

if ( a>=qty AND d>=qty AND e>=qty ) then

assembleModules (qty, 'B', orderId);

else

begin

y:=qty-a;

if (y>0) then searchComponentsTable (y, 'B', orderId, 1);

end if;

y:=qty-d;

if (y>0) then searchComponentsTable (y, 'B', orderId, 4);

end if;

y:=qty-e;

if (y>0) then searchComponentsTable (y, 'B', orderId, 5);

end if;

end;

end if;

end;

elsif (typeInput='C') then

begin

if ( b>=qty AND c>=qty AND e>=qty ) then

assembleModules (qty, 'C', orderId);

else

begin

y:=qty-b;

if (y>0) then searchComponentsTable (y, 'C', orderId, 2);

end if;

y:=qty-c;

if (y>0) then searchComponentsTable (y, 'C', orderId, 3);

end if;

y:=qty-e;

if (y>0) then searchComponentsTable (y, 'C', orderId, 5);

end if;

end;

end if;

end;

elsif (typeInput='D') then

begin

if ( b>=qty AND d>=qty AND e>=qty ) then

assembleModules (qty, 'D', orderId);

else

begin

y:=qty-b;

if (y>0) then searchComponentsTable (y, 'D', orderId, 2);

end if;

y:=qty-d;

if (y>0) then searchComponentsTable (y, 'D', orderId, 4);

end if;

y:=qty-e;

if (y>0) then searchComponentsTable (y, 'D', orderId, 5);

end if;

end;

end if;

end;

end if;

END;

SETSALESTATUS

create or replace

PROCEDURE setSaleStatus (qty IN INT , typeInput IN VARCHAR, orderId IN INT)

IS

i int;

a int;

xyy varchar(30);

BEGIN

DBMS\_OUTPUT.put\_line('start of setSaleStatus function');

FOR i IN 1..qty LOOP

select min(m.serial\_no) into a from mipad m where m.type=typeInput AND m.sale\_status='fs' and m.test\_result='s';

UPDATE Mipad set sale\_status='s' where serial\_no=a;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :mipad number ' || a || ' of type ' || typeInput || ' was sold');

DBMS\_OUTPUT.put\_line('write\_into\_groupc(a, typeinput, orderID);');

END LOOP;

update inventory set qty=0 where type= typeinput;

END;

SLEEP

create or replace

PROCEDURE sleep (time\_in IN INT)

IS v\_now DATE;

BEGIN

SELECT SYSDATE INTO v\_now FROM DUAL;

LOOP

EXIT

WHEN v\_now+(time\_in\*(1/86400))=SYSDATE;

END LOOP;

RETURN;

END;

TESTMIPAD

create or replace

PROCEDURE testMipad (mipadNum IN INT)

IS

xyy varchar(30);

BEGIN

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :testing mipad with serial number'|| mipadNum);

sleep(1);

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

DBMS\_OUTPUT.put\_line(xyy||' :test took 1 seconds');

Update Mipad set test\_result='s' where serial\_no=mipadNum;

Update Mipad set sale\_status='fs' where serial\_no=mipadNum;

END;

UPDATE\_ORDER\_TABLE

create or replace

PROCEDURE UPDATE\_ORDERS\_TABLE

IS

aa integer;

b integer;

cursor ab is

select o.oid from orders\_fake o where o.to\_be\_delivered>0;

cursor cd is

select oo.oid from orderdelivery oo;

ba ab%rowtype;

dc cd%rowtype;

BEGIN

open ab;

loop

fetch ab into ba;

exit when ab%NOTFOUND;

open cd;

loop

fetch cd into dc;

exit when cd%NOTFOUND;

if (ba.oid=dc.oid) then update orders\_fake set to\_be\_delivered=0 where oid=ba.oid;

end if;

end loop;

close cd;

end loop;

close ab;

END;

WRITE\_INTO\_GROUPC

create or replace

PROCEDURE WRITE\_INTO\_GROUPC(pid IN Integer, type in Varchar, orderId IN INT)

IS

b char;

d char;

xyy varchar(30);

BEGIN

insert into lezha.inventory values (pid,type,orderId);

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line (xyy||' :inserted into inventory table of group C');

update lezha.A\_order set Is\_received='y' where ao\_id=orderid;

select to\_char(systimestamp,'HH24:MI:SSFF3') into xyy from dual;

dbms\_output.put\_line(xyy||' :updated table A\_order of group C for order number '|| orderid);

END;

TRIGGERS

DELIVERCOMPONENTS

create or replace

TRIGGER deliveredComponents

After update on components

FOR EACH ROW

declare

x int;

y int;

BEGIN

DBMS\_OUTPUT.put\_line('starting trigger');

select c.qty into x from components\_fake c where c.cid=:new.cid;

update components\_fake c set c.qty=:new.qty where c.cid=:new.cid;

DBMS\_OUTPUT.put\_line('starting if/then');

if (:new.qty<x) then

DBMS\_OUTPUT.put\_line('no need to do anything!');

end if;

if (:new.qty>x)then

y:=:new.qty-x;

DBMS\_OUTPUT.put\_line(y||' type '|| :new.cid|| ' components delivered to us');

checkComponents (:new.cid);

end if;

ENd;

ORDERS

create or replace

TRIGGER Orders

After INSERT on Orders

FOR EACH ROW

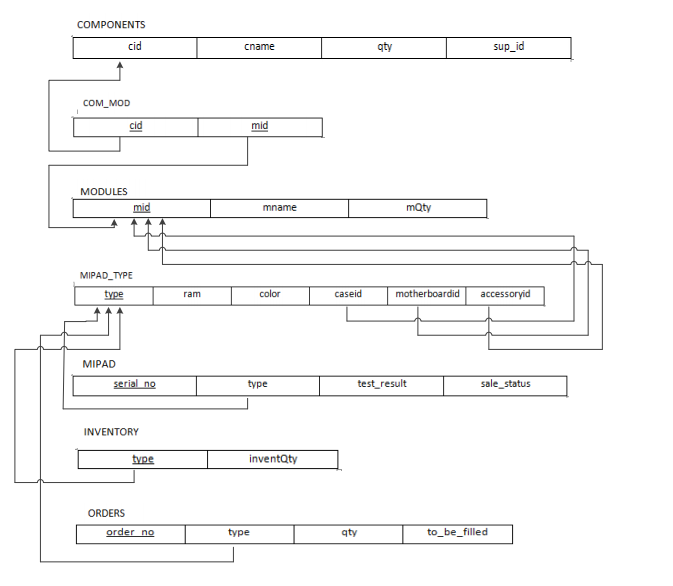
Begin

DBMS\_OUTPUT.put\_line('Group C has ordered: ' || :New.qty || ' type ' || :New.type || ' mipads');

insert into orders\_fake values (:new.oid, :new.type, :new.qty, :new.qty);

procInventory (:new.qty, :new.type, :new.oid);

END;



CREATE TABLE Components(

cid INT NOT NULL,

cname VARCHAR(25) NOT NULL,

qty INT NOT NULL,

sup\_id Char(5) NOT NULL,

PRIMARY KEY(cid),

CONSTRAINT fk\_xx

FOREIGN KEY(sup\_id)

REFERENCES Supplier(sup\_id ) ON DELETE SET NULL

);

The table Components records the components we purchase from Team A’s suppliers and use in creating modules. The table records a unique identifier for each component (cid)--the value of which is the same across Team A and Team B--the name of each component (cname), the amount of each component we have in stock (qty), and the id of the supplier that we order each component from (supp). When our orders to Team A are fulfilled, the qty field will be updated to reflect the number of components added. The table references Team A’s Supplier table to keep the supplier names consistent. Components also has a relationship with Com\_mod, specifying which components are used for which modules.

CREATE TABLE Com\_mod(

cid INT NOT NULL,

mid INT NOT NULL,

CONSTRAINT com\_mod\_pk PRIMARY KEY(cid, mid),

CONSTRAINT fk\_com

FOREIGN KEY(cid)

REFERENCES Components(cid) ON DELETE SET NULL,

CONSTRAINT fk\_mod

FOREIGN KEY(mid)

REFERENCES Modules(mid) ON DELETE SET NULL

);

The table Com\_mod connects the components to the modules they are used to create. The entities “components” and “modules” have a many-to-many relationship, so this table bridges that.

CREATE TABLE Modules(

mid INT NOT NULL,

mname VARCHAR(25) NOT NULL,

mQty INT NOT NULL,

CONSTRAINT mod\_pk PRIMARY KEY(mid, qty)

);

The table Modules records information about each module type that is manufactured. It includes a unique id for each type of module (mid), the name of the module (mname), and the number of each type of module we currently have in stock (qty). It is related to the table Com\_mod, which records what components go into making each module, and to the table Types, which records what modules are used to create which types of miPad.

CREATE TABLE Types(

type CHAR(5) NOT NULL,

ram CHAR(5) NOT NULL,

color CHAR(5) NOT NULL,

caseid INT NOT NULL,

motherboardid INT NOT NULL,

accessoryid INT NOT NULL,

CONSTRAINT types\_pk PRIMARY KEY(type),

CONSTRAINT fk\_mod1

FOREIGN KEY(caseid)

REFERENCES Modules(mid),

CONSTRAINT fk\_mod2

FOREIGN KEY(motherboardid)

REFERENCES Modules(mid),

CONSTRAINT fk\_mod3

FOREIGN KEY(accessoryid)

REFERENCES Modules(mid),

);

The table Types records information about the types of miPads we assemble. Each type has a unique name (type), a given amount of RAM (ram), a color (color), and is made up of three modules: a case (caseid), an integrated motherboard (motherboardid), and a module holding several other parts and accessories (accessoryid). The modules fields reference the Modules table. The Types table is also related to both the MiPad table, which records information about the individual miPads produced, and to the inventory table, which tracks the number of miPads of each type we have.

CREATE TABLE MiPad(

serial\_no INT NOT NULL,

type VARCHAR(5) NOT NULL,

test\_result VARCHAR(2) NOT NULL,

sale\_status VARCHAR(2) NOT NULL,

CONSTRAINT mipad\_pk PRIMARY KEY(pid),

CONSTRAINT fk\_type

FOREIGN KEY(type)

REFERENCES Types(type),

);

The table MiPad records information for each individual miPad we make. Each miPad has a unique identifier (pid), is of a given type (type), and is sold or unsold (sale\_status). The table also records whether or not a miPad has been tested yet and, if it has been tested, if it passed or failed the test (test\_result). MiPad references the Types table to indicate what types of miPad can be and are present.

CREATE TABLE Inventory(

type VARCHAR(5) NOT NULL,

qty INT,

CONSTRAINT Inventory\_pk PRIMARY KEY(type),

CONSTRAINT fk\_type2

FOREIGN KEY(type)

REFERENCES Types(type),

);

The table Inventory records numbers of currently-in-stock, ready-to-be-sold miPads. The table includes the type of miPad (type) and the number of that type of miPad that are available for immediate sale (inventQty). Like the MiPad table, Inventory references the Types table for referential integrity on the types of miPads available.

CREATE TABLE Orders(

order\_no INT NOT NULL,

type VARCHAR(5) NOT NULL,

qty INT NOT NULL,

to\_be\_filled INT NOT NULL,

CONSTRAINT order\_pk PRIMARY KEY(oid),

CONSTRAINT fk\_inv

FOREIGN KEY(type)

REFERENCES Inventory(type) ON DELETE SET NULL

);

The table Orders records the orders that we receive from Team C. Each order has a unique, identifying number (oid), is for a type of miPad (type), gives the number of miPads wanted (qty), and records how many miPads still need to be provided to Team C to complete the order (to\_be\_filled). This is the table Team C will insert into the order miPads from us.