

# E-commerce Journey

Andy && Theodore

How not to yield to dynamics...



# RoadMap (keypoints)

- Approach
- Business logic
- User Case
- Take away

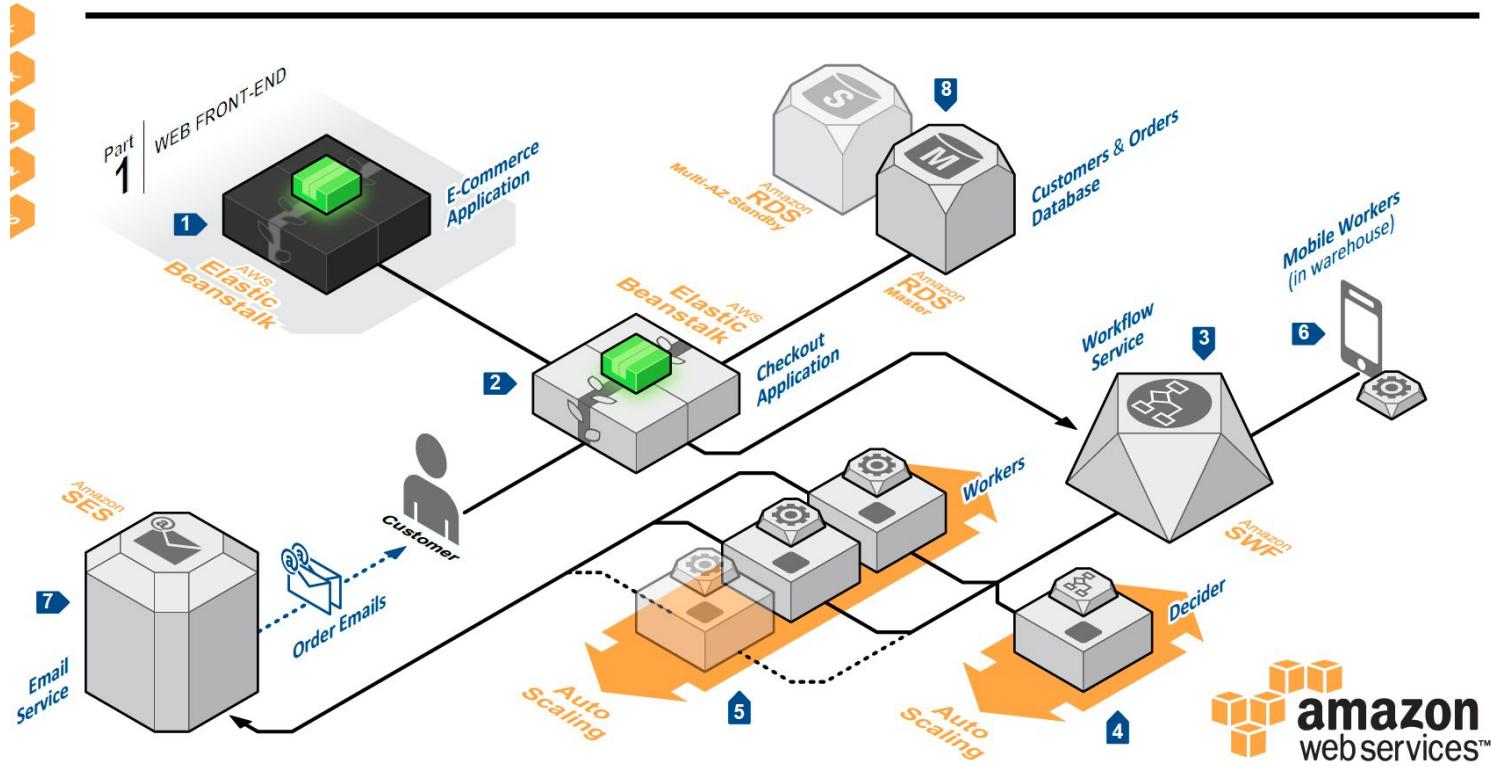
# Insights: E-commerce in 90s vs today

## First Websites

The screenshot shows the first version of Google's homepage. At the top, it says "First Websites". Below that is the iconic "Google! BETA" logo. A search bar contains the placeholder "Search the web using Google!". Below the search bar are two buttons: "Google Search" and "I'm feeling lucky". Underneath the search bar is a teal-colored footer bar with links: "Special Searches", "Stanford Search", "Linux Search", "Help!", "About Google", "Company Info", "Google Legos", "Get Google! updates monthly", "your e-mail", "Subscribe", and "Archive". At the bottom, it says "Copyright ©1998 Google Inc." and "of Famous Companies".

The screenshot shows the Amazon website for the search term "pc-". The top navigation bar includes categories like Electronics, Books, Best Sellers, Gift Ideas, New Releases, Whole Foods, Today's Deals, AmazonBasics, Coupons, and Gift Cards. The main content area shows a search result for "1-24 of over 70,000 results for Electronics : 'pc-'". On the left, there are filters for Delivery Day (Get it Today or Get it by Tomorrow), Eligible for Free Shipping (Free Shipping by Amazon), Department (Electronics, Desktops, Towers, Monitors, Computer Cases, External Hard Drives, etc.), and Avg. Customer Review (5 stars & up). In the center, there is an "Editorial recommendations" section featuring the "EDITOR'S CHOICE" Dell OptiPlex Intel Core 2 Duo. To the right, there are three product cards: "CONTENDER" Dell Inspiron Gaming PC, "BEST VALUE" Dell Optiplex Intel Core i5, and another Dell OptiPlex model. Each card includes a small image of the computer, its name, a brief description, and its price.

# What hasn't changed.

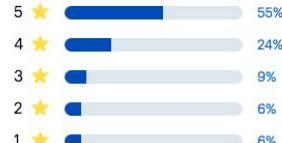


# The Workflow step 1

HP - Pavilion x360 2-in-1 11.6" Touch-Screen Laptop - Intel Pentium - 4GB Memory  
- 500GB Hard Drive - HP Finish In Natural Silver

Model: 11M-AD113D SKU: 6237353

★★★★★ 4.2 (797 Customers) | 3.4 (19 Experts) | 63 Answered Questions



[Read Reviews](#)



# The Workflow step 2

OrderManager Database Cache

File Edit View Insert Format Data Tools Add-ons Help Saving...

undo redo 100% \$ % .0 .00 123 Arial 10 A

fx 3

	A	B	C	D	E
1	Orders_ID int not null	Product_SKU varchar(12) not null	UnitPrice decimal(13,2) not null	UnitAmount int not null	OrderRecord_Status boolean DEFAULT false
2	1	PC-123456-1B	2120.23	4	FALSE
3	2	PC-123456-1C	2120.23	2	FALSE
4	1	PC-123456-0C	2110.22	3	FALSE
5	3	PC-123456-0B	2090.23	3	FALSE
6	4	PC-123456-AA	2230	3	FALSE
7	5	PC-123456-AA	2230	12	FALSE

# The Workflow step 3

Properties Data ER Diagram Derby - Order\_Manager 2

PRODUCT | Enter a SQL expression to filter results (use Ctrl+Space)

Grid Value X

NAME	DESCRIPTION
MacBook 11 inch 2017	"MacBook Pro 2017 has a new eighth-generation quad-core Intel processor with Turbo
MacBook Pro 15 inch 2019	"MacBook Pro 2019 has a new eighth-generation quad-core Intel processor with Turbo
MacBook Pro 15 inch 2018	"MacBook Pro 2018 has a new eighth-generation quad-core Intel processor with Turbo
MacBook Pro 15 inch 2020	"MacBook Pro 2020 has a new eighth-generation quad-core Intel processor with Turbo
MacBook Pro 13 inch 2020	"MacBook Pro 2020 has a new eighth-generation quad-core Intel processor with Turbo
MacBook Pro 15 inch 2016	"MacBook Pro 2016 has a new eighth-generation quad-core Intel processor with Turbo
MacBook Pro 15 inch 2017	"MacBook Pro 2017 has a new eighth-generation quad-core Intel processor with Turbo
DELL INSPIRON 15 inch 2014	"Featuring an Intel Core i7 processor, this notebook ensures consistently smooth pe

# Configured-DataTypes

Name	Precision	Min scale	Max scale	Description
123 BIGINT	19	0	0	BIGINT
010 LONG VARCHAR FOR BIT DATA	32,700	0	0	LONG VARCHAR FOR BIT DATA
011 VARCHAR () FOR BIT DATA	32,672	0	0	VARCHAR () FOR BIT DATA
012 CHAR () FOR BIT DATA	254	0	0	CHAR () FOR BIT DATA
T LONG VARCHAR	32,700	0	0	LONG VARCHAR
RBC CHAR	254	0	0	CHAR
123 CHAR AL	31	0	31	DECIMAL
123 INTEGER	10	0	0	INTEGER
123 SMALLINT	5	0	0	SMALLINT
123 FLOAT	52	0	0	FLOAT
123 REAL	23	0	0	REAL
123 DOUBLE	52	0	0	DOUBLE
RBC VARCHAR	32,672	0	0	VARCHAR
✓ BOOLEAN	1	0	0	BOOLEAN
⌚ DATE	10	0	0	DATE
⌚ TIME	8	0	0	TIME
⌚ TIMESTAMP	29	0	9	TIMESTAMP
?] OBJECT	0	0	0	OBJECT
?] BLOB	2,147,483,647	0	0	BLOB
?] CLOB	2,147,483,647	0	0	CLOB
✗ XML	0	0	0	XML

# Implemented Function For Order Manager DataBase

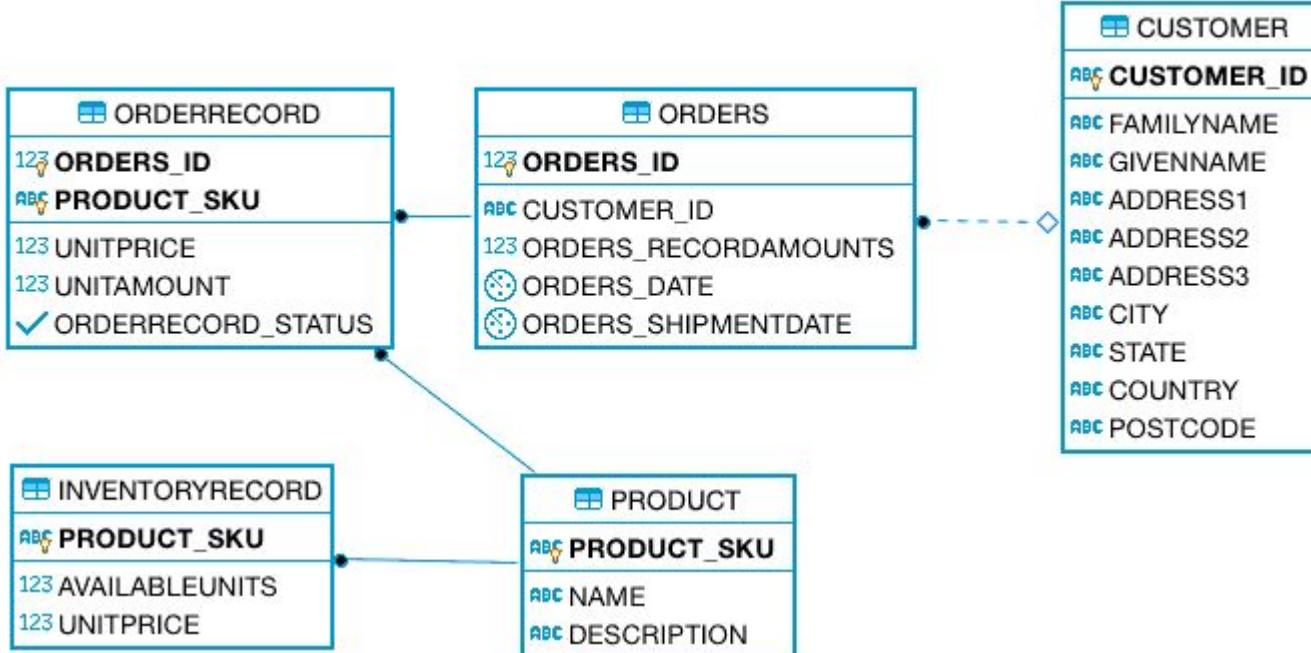
1. Auto Order Date
2. Auto Ship Date
3. Support Replenish of Inventory
4. Support BackOrder
5. Multiple Triggers to achieve maximum automation

```
// triggers created by this program
String dbTriggers[] = {
    "InsertUnitPriceInOrderRecord", "InsufficientInventory", "ChangeOrderRecordStatus",
    "AutoDeductInventory", "AutoShipDate", "SendEmailToCustomer", "ReturnItemsToInventory",
    "NoRecordForOrder", "BackOrder"
};
```

# Key Design Choice

1. Primary Key in Product: SKU only ( not combined with UnitPrice or no Primary Key)
2. Orders has a field “**Record Amounts**” (Order 1 : 2 Records)
3. OrderRecord has a field **Record\_Status**
  - False: not enough stock
  - True: enough stock, but **not Shipped** unless every Record Under the same Order\_ID has status true
4. If Available Units = 0 for SKU in Inventory Do not delete this SKU - Replenish will check OrderRecord with status = false

# DDL - ER diagram



```
// Create the Product table
String createTable_Product =
    "create table Product (" +
    + "  Name varchar(32) not null," +
    + "  Description varchar(255) not null unique," +
    + "  Product_SKU varchar(12) not null," +
    + "  primary key (Product_SKU)," +
    + "  check(isSKU(Product_SKU)))" +
    + ");"

// Inventory Record table - stores unit price for current item
String createTable_InventoryRecord =
    "create table InventoryRecord (" +
    + "  Product_SKU varchar(12) not null," +
    + "  AvailableUnits int not null," +
    + "  UnitPrice decimal(13,2) not null," +
    + "  primary key (Product_SKU), " // one Product_SKU can only have one price
    // + "  check(isRounded(UnitPrice)),"
    + "  check(AvailableUnits >= 0),"
    + "  check(UnitPrice >= 0.00), " // free item
    + "  foreign key (Product_SKU) references Product(Product_SKU) on delete cascade"
    + ");"
```



Unique / Decimal(13,2) / Free item / SKU with 0 stock

Adding **On Update Restrict** / **On Update Cascade** for  
mySQL only

```
String createTable_Customer =  
    "create table Customer ("  
    + "  FamilyName varchar(16) not null,"  
    + "  GivenName varchar(16) not null,"  
    + "  Customer_ID varchar(320) not null,"  
    + "  Address1 varchar(255) not null,"  
    + "  Address2 varchar(255) default null," // could be null  
    + "  Address3 varchar(255) default null," // could be null  
    + "  City varchar(255) not null,"  
    + "  State varchar(2) not null constraint State_Constraint"  
    + "      check (State in ('CA','AZ','TX','KL','UP','WB','AH','GS','BJ')),"  
    + "  Country varchar(3) not null constraint Country_Constraint"  
    + "      check (Country in ('USA','IND','CHN')),"  
    + "  PostCode varchar(11) not null,"  
    + "  primary key (Customer_ID),"  
    + "  check (isCustomerIDEmailAddress(Customer_ID)),"  
    + "  check (isZipCode(PostCode))"  
    + ")";
```

Enum / 3 Address subfields with 1 not null and 2 default null

```
// GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1)
String createTable_Orders =
    "create table Orders("
    + " Orders_ID int not null GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1) "
    + " Customer_ID varchar(320) not null,"
    + " Orders_RecordAmounts int not null,"
    + " Orders_Date TIMESTAMP not null DEFAULT CURRENT_TIMESTAMP, " // check format and dat
    + " Orders_ShipmentDate TIMESTAMP default null," // could be null when not shipped
    + " primary key (Orders_ID), " // Orders_ID↑completely determines relation
    + " check(Orders_RecordAmounts > 0),"
    + " foreign key (Customer_ID) references Customer(Customer_ID) on delete cascade"
    + ")";
stmt.executeUpdate(createTable_Orders);
String createTable_OrderRecord =
    "create table OrderRecord("
    + " Orders_ID int not null,"
    + " Product_SKU varchar(12) not null,"
    + " UnitPrice decimal(13,2) not null,"
    + " UnitAmount int not null, " //
    + " OrderRecord_Status boolean DEFAULT false," // (meaning incomplete/pending) or
    + " primary key (Orders_ID,Product_SKU),"
    + " foreign key (Orders_ID) references Orders(Orders_ID) on delete cascade,"
    + " foreign key (Product_SKU) references Product(Product_SKU) on delete cascade,"
    + " check(UnitAmount > 0)"
    + ")";
stmt.executeUpdate(createTable_OrderRecord);
```

AUTO\_INCREMENT / DEFAULT TimeStamp / DEFAULT value NULL for ShipDate

# Cascading Triggers

Input Data for OrderRecord

1	PC-123456-1B	4
1	PC-123456-0C	3
2	PC-123456-1C	2
3	PC-123456-0B	99
4	PC-123456-1B	1
5	PC-123456-1B	1
5	PC-123456-1C	1
5	PC-123456-0B	1
6	PC-123456-00	1

```
( 1, PC-123456-1B, $ 2120.23, 4, true)
( 1, PC-123456-0C, $ 2110.22, 3, true)
( 2, PC-123456-1C, $ 2120.23, 2, true)
( 3, PC-123456-0B, $ 2090.23, 99, false)
( 4, PC-123456-1B, $ 2120.23, 1, true)
( 5, PC-123456-1B, $ 2120.23, 1, true)
( 5, PC-123456-1C, $ 2120.23, 1, true)
( 5, PC-123456-0B, $ 2090.23, 1, true)
( 6, PC-123456-00, $ 2110.23, 1, true)
```

Input Data for Orders

1	mjames@husky.neu.edu	2
2	kud@hotmail.com	1
3	prav@gmail.com	1
4	cchap@foxmail.com	1
5	andy@gmail.com	3
6	ted@hotmail.com	2

Orders\_Id: 1,Email: mjames@husky.neu.edu, Record Amount: 2, OrderDate:  
Orders\_Id: 2,Email: kud@hotmail.com, Record Amount: 1, OrderDate:  
Orders\_Id: 3,Email: prav@gmail.com, Record Amount: 1, OrderDate:  
Orders\_Id: 4,Email: cchap@foxmail.com, Record Amount: 1, OrderDate:  
Orders\_Id: 5,Email: andy@gmail.com, Record Amount: 3, OrderDate:  
Orders\_Id: 6,Email: ted@hotmail.com, Record Amount: 2, OrderDate:

Tables Orders has 6 rows

2019-06-25 11:18:03.872,	ShipDate:	2019-06-25 11:18:24.189
2019-06-25 11:18:03.896,	ShipDate:	2019-06-25 11:18:24.195
2019-06-25 11:18:03.898,	ShipDate:	null
2019-06-25 11:18:03.9,	ShipDate:	2019-06-25 11:18:24.235
2019-06-25 11:18:03.901,	ShipDate:	2019-06-25 11:18:24.253
2019-06-25 11:18:03.903,	ShipDate:	null

# Cascading Triggers on OrderRecord

## 1. Auto Select Price for SKU from Inventory: Two Ways / Using Pure Trigger or Trigger Calling Stored Procedures

```
// create trigger for auto insertion of unitPrice in InventoryRecord table
String createTrigger_InsertUnitPriceInOrderRecord =
    "create trigger InsertUnitPriceInOrderRecord"
    + " after insert on OrderRecord"
    + " REFERENCING NEW ROW AS NewTuple"
    + " for each row MODE DB2SQL"
    + " UPDATE OrderRecord"
    + "     SET UnitPrice = "
    + "         (SELECT UnitPrice from InventoryRecord "
    + "             where InventoryRecord.Product_SKU = NewTuple.Product_SKU)"
    + "         where Product_SKU = newTuple.Product_SKU";
```

# Cascading Triggers on OrderRecord

2. Check If Inventory Stock is not Sufficient, when = if check

```
String createTrigger_InsufficientInventory =
    "create trigger InsufficientInventory"
+ " NO CASCADE BEFORE insert on OrderRecord"
+ " REFERENCING NEW ROW AS NewTuple"
+ " for each row MODE DB2SQL"
+ " WHEN (NewTuple.UnitAmount >"
+ "        (SELECT AvailableUnits from InventoryRecord "
+ "         where InventoryRecord.Product_SKU = NewTuple.Product_SKU))"
+ " CALL InsufficientItems(NewTuple.Product_SKU, NewTuple.UnitAmount)";
```

# Cascading Triggers on OrderRecord

3. Check If Inventory Stock is Sufficient, the else branch after if because there is branching conditions check in Derby Trigger

```
String createTrigger_ChangeOrderRecordStatus =  
    "create trigger ChangeOrderRecordStatus"  
+ " AFTER insert on OrderRecord"  
+ " REFERENCING NEW ROW AS NewTuple"  
+ " for each row MODE DB2SQL"  
+ " WHEN (NewTuple.UnitAmount <="  
+ "     (SELECT AvailableUnits from InventoryRecord "  
+ "         where InventoryRecord.Product_SKU = NewTuple.Product_SKU))"  
+ " UPDATE OrderRecord set OrderRecord_Status = true"  
+ "     where (OrderRecord.Product_SKU = NewTuple.Product_SKU"  
+ "     and OrderRecord.Orders_ID = NewTuple.Orders_ID);
```

Change status = True if enough stock, Change Trigger After Insert

# Cascading Triggers on OrderRecord

4. Trigger on change of the status, if true, then deduct the items from Inventory

```
String createTrigger_AutoDeductInventory =  
    "create trigger AutoDeductInventory"  
    + " AFTER Update OF OrderRecord_Status"  
    + " on OrderRecord"  
    + " REFERENCING NEW ROW AS NewTuple"  
    + " for each row MODE DB2SQL"  
    + " Update InventoryRecord"  
    + "      set AvailableUnits = AvailableUnits - NewTuple.UnitAmount"  
    + "      where (InventoryRecord.Product_SKU = NewTuple.Product_SKU)";  
stmt.executeUpdate(createTrigger_AutoDeductInventory);  
System.out.println("Created trigger for AutoDeductInventory");
```

# Cascading Triggers on OrderRecord

5. Yet another Trigger on change of the status, to check if the shipDate should be Set

```
String createTrigger_AutoShipDate =
    "create trigger AutoShipDate"
    + " AFTER Update OF OrderRecord_Status"
    + " on OrderRecord"
    + " REFERENCING NEW ROW AS NewTuple"
    + " for each row MODE DB2SQL"
    + " CALL AutoShipCheck(NewTuple.Orders_ID)";
stmt.executeUpdate(createTrigger_AutoShipDate);
System.out.println("Created trigger for AutoShipDate");
```

# Cascading Triggers on OrderRecord

```
public static void AutoShipCheck(int Orders_ID) {
    try {
        Connection conn = DriverManager.getConnection("jdbc:default:connection");

        PreparedStatement pstmt = conn.prepareStatement("SELECT COUNT(*) FROM OrderRecord where (OrderRecord.Orders_ID = ? AND OrderRecord.OrderRecord");
        pstmt.setInt(1, Orders_ID);
        pstmt.execute();
        ResultSet rs1 = pstmt.getResultSet();
        int True_Records = -1; // guard value
        if (rs1.next()) {
            True_Records = rs1.getInt(1);
        }
        System.out.printf("True_Status_Records for Orders_ID %d is %d\n",Orders_ID,True_Records);

        pstmt = conn.prepareStatement("Update Orders SET Orders_ShipmentDate = CURRENT_TIMESTAMP where (Orders.Orders_ID = ? AND ((Select Orders_Reco
        pstmt.setInt(1, Orders_ID);
        pstmt.setInt(2, Orders_ID);
        pstmt.setInt(3, True_Records);
        pstmt.execute();
        // System.out.printf("Record_Amount is %d\n",Record_Amount);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
public static BigDecimal getUnitPriceFromInventoryRecord(String Product SKU) {
```

# Functional Triggers

If Status = True, And No ShipDate yet (All records ship together), meaning there are other Records with False status  
Delete record will return Items back to Inventory

```
g createTrigger_ReturnItemsToInventory =
  "create trigger ReturnItemsToInventory"
+ " AFTER DELETE "
+ " on OrderRecord"
+ " REFERENCING OLD ROW AS OldTuple"
+ " for each row MODE DB2SQL"
+ " WHEN ( (OldTuple.OrderRecord_Status = true) AND "
+ " ((Select Orders_ShipmentDate From Orders where Orders_ID = OldTuple.Orders_ID) is null))
+ " Update InventoryRecord"
+ "      set AvailableUnits = AvailableUnits + OldTuple.UnitAmount"
+ "      where (InventoryRecord.Product_SKU = OldTuple.Product_SKU);"
```

# Functional Triggers

If this Record is the last Record for Order\_ID ( regardless of status), then that Order\_ID tuple is deleted

```
String createTrigger_NoRecordForOrder =
    "create trigger NoRecordForOrder"
+ " AFTER DELETE "
+ " on OrderRecord"
+ " REFERENCING OLD ROW AS OldTuple"
+ " for each row MODE DB2SQL"
+ " WHEN ((Select COUNT(*) FROM OrderRecord Where OrderRecord.Orders_ID = OldTuple.Orders_ID) = 0)"
+ " Delete From Orders"
+ "      where (Orders.Orders_ID = OldTuple.Orders_ID)";
stmt.executeUpdate(createTrigger_NoRecordForOrder);
System.out.println("Created trigger for NoRecordForOrder");
```

# Functional Triggers

If Inventory is replenished, the check OrderRecord with status = false and that SKU, try to set the status = true  
And set status = true will invoke cascading triggers (decrease stock, etc)

```
// and auto-triggers
String createTrigger_BackOrder =
    "create trigger BackOrder"
    + " AFTER UPDATE OF AvailableUnits "
    + " on InventoryRecord"
    + " REFERENCING NEW ROW AS NewTuple"
    + " for each row MODE DB2SQL"
// + " WHEN ((Select AvailableUnits FROM InventoryRecord Where InventoryRecord.Product_SKU = NewTuple.Product_SKU) < NewTuple.AvailableUnits)"
    + " UPDATE OrderRecord SET "
    + " OrderRecord.OrderRecord_Status = true "
    + " WHERE (OrderRecord.OrderRecord_Status = false AND "
    + "        (Select UnitAmount FROM OrderRecord Where "
    + "        (OrderRecord.Product_SKU = NewTuple.Product_SKU)) < NewTuple.AvailableUnits)";
stmt.executeUpdate(createTrigger_BackOrder);
```

# UseCases

## {DQL}

# Transaction Roll-back

\*

API for insert/delete/update record and rollback

After insertion, Table InventoryRecord has 7 rows  
InventoryRecords:

PC-123456-00, cnt: 70, \$2110.23  
PC-123456-0B, cnt: 40, \$2090.23  
PC-123456-0C, cnt: 30, \$2110.22  
PC-123456-1B, cnt: 10, \$2120.23  
PC-123456-1C, cnt: 20, \$2120.23  
PC-123456-1N, cnt: 12, \$2130.12  
PC-123456-AA, cnt: 60, \$2320.23

executing following query: delete from InventoryRecord where ProductId = 1  
InventoryRecords:

PC-123456-00, cnt: 70, \$2110.23  
PC-123456-0B, cnt: 40, \$2090.23  
PC-123456-0C, cnt: 30, \$2110.22  
PC-123456-1B, cnt: 10, \$2120.23  
PC-123456-1C, cnt: 20, \$2120.23  
PC-123456-AA, cnt: 60, \$2320.23

After execution, Table InventoryRecord has 6 rows  
InventoryRecords:

PC-123456-00, cnt: 70, \$2110.23  
PC-123456-0B, cnt: 40, \$2090.23  
PC-123456-0C, cnt: 30, \$2110.22  
PC-123456-1B, cnt: 10, \$2120.23  
PC-123456-1C, cnt: 20, \$2120.23  
PC-123456-1N, cnt: 12, \$2130.12  
PC-123456-AA, cnt: 60, \$2320.23

After Roll Back, Table InventoryRecord has 7 rows

# **Know Thou Users**

**Lets run a demo ~**

# DQL Conclusion:

\* API for Customer Orders system

\* API for Data Analyst

The screenshot shows a Java API documentation interface. At the top, there is a navigation bar with links: OVERVIEW, PACKAGE, CLASS, USE, TREE (which is highlighted in orange), DEPRECATED, INDEX, and HELP. Below the navigation bar, there are links for PREV, NEXT, FRAMES, and NO FRAMES. The main content area has a dark blue header with the title "Hierarchy For Package API". Underneath, there is a section titled "Package Hierarchies:" followed by a link to "All Packages". Below this, there is a section titled "Class Hierarchy" with a list of classes: "java.lang.Object", "API.DBAdminAPI", and "API.PrintUtil". At the bottom of the page is another navigation bar with the same set of links as the top one.

OVERVIEW PACKAGE CLASS USE **TREE** DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES

## Hierarchy For Package API

**Package Hierarchies:**

All Packages

## Class Hierarchy

- java.lang.Object
  - API.**DBAdminAPI**
  - API.**PrintUtil**

OVERVIEW PACKAGE CLASS USE **TREE** DEPRECATED INDEX HELP

# **Future Goal**

- \* Automation (data loader sequentially)
- \* Encapsulations (apis -> RESTful apis)
- \* Scalability ( single user at time -> Concurrent users)

# Future Goal

- \* Set the UnitPrice of OrderRecord to be Sale Price
  - 1.Does not violate 3rd Normal Form
  - 2.Meet Boyce–Codd Normal Form
  - 3.Calculate Profit

Compare the efficiency by triggers and stored procedures

# Reference

- [1]. slides2 online seller's joke : <https://www.kyozou.com/8-jokes-online-seller-will-love/>
- [2] slides 5 AWS <https://aws.amazon.com/cn/blogs/aws/three-new-aws-reference-architectures-for-e-commerce/>
- [3] slides 6 Product sku & description  
<https://www.bestbuy.com/site/hp-pavilion-x360-2-in-1-11-6-touch-screen-laptop-intel-pentium-4gb-memory-500gb-hard-drive-hp-finish-in-natural-silver/6237353.p?skuld=6237353>