MEMBERS:

Rui Cheng

Bowen Lui

Rahul Arora

Prateek Trivedi

Aditya Behal

Navin Mohan

|  |  |
| --- | --- |
| http://www.dpcio.ir/Upload/Modules/Contents/asset0/IBM%20Optim%20framework.jpg  EDM PROJECT REPORT  FOR FXH INC. | Abstract  Report on a database system that handles the learning content for different industry verticals. These are provided by a multitude of third party providers that would help our client to create a personalized learning content. The DBMS handles the creation of these individualized learning portions, storing the third party vendors content, storing the payment information of the client amongst other critical parts.  Navin Mohan  ENTERPRISE DATABASE MANAGEMENT – MIS 531 |

Table of Contents

[CHAPTER 1: INTRODUCTION 2](#_Toc437811663)

[CHAPTER 2: REQUIREMENT ANALYSIS 3](#_Toc437811664)

[CHAPTER 3: ER DIAGRAM 6](#_Toc437811665)

[DATA DICTIONARY (CONCEPTUAL / FOR ER MODELLING) PART 1 of 2 7](#_Toc437811666)

[DATA DICTIONARY (CONCEPTUAL / FOR ER MODELLING) PART 2 of 2 8](#_Toc437811668)

[CHAPTER 4: RELATIONAL SCHEMA 9](#_Toc437811669)

[DATA DICTIONARY (RELATIONAL / FOR ER MODELLING) PART 1 of 2 12](#_Toc437811670)

[DATA DICTIONARY (RELATIONAL / FOR ER MODELLING) PART 2 of 2 13](#_Toc437811671)

[CHAPTER 5: DATA POPULATION AND QUERIES 14](#_Toc437811672)

[CHAPTER 6: TRIGGERS AND PROCEDURES 24](#_Toc437811673)

[CHAPTER 7: INTERFACE AND REPORTS 32](#_Toc437811674)

[ADMIN LOGIN 32](#_Toc437811675)

[Customer Login: 38](#_Toc437811676)

[CHAPTER 8: CONCLUSION AND IMPLEMENTATION PLAN 41](#_Toc437811677)

[Implementation Plan 41](#_Toc437811678)

[Installation Plan 41](#_Toc437811679)

[System Cost 42](#_Toc437811680)

[Conclusion 42](#_Toc437811681)

[Lessons Learned 43](#_Toc437811682)

[APPENDIX 1: TABLE CREATION 44](#_Toc437811683)

[APPENDIX 2: INSERTION OF DATA 51](#_Toc437811684)

# CHAPTER 1: INTRODUCTION

An electronic document management initiative/project is usually launched to satisfy a wide range of requirements. In order to ensure the goals for the EDM project are achieved, the requirements should be written down in a formal document. That helps the team wrap its head around the requirements/needs of the clients.

A requirement analysis process happens between the client and the consultant. This process is undertaken to define “what” is required in order to improve the business processes and achieve the organization’s vision for growth. The process of documenting the requirement is crucial as it enables the project team to establish a common understanding of the unique business process and content management needs for their organization. This also helps the team to align itself with the thought process of the client.

For our project, our client Mr. Ryan Gyure, CEO and Founder of FXH Arizona has a requirement for a database system that handles the learning content that he proposes to sell to his customers. Since this is a new project all together even for the client the team is working consistently with the client to understand his requirements. FXH Arizona is in the business of web design, web design and development, hosting, website management and backup services for businesses. The learning contents for different industries and functions are provided by a multitude of third party vendors that he would compile and create a personalized learning content for each of his customers. Our project handles the creation of these individualized study portions, storing of the third party vendors content, store the payment information of the client amongst other critical parts. The user guide for the end user that is put together by Ryan would not be covered by the team.

By providing the client with a database management solution, the values we provide are:

1. Controlling redundancy in a stored data. No multiple entries leading to a single client.
2. The integrity of the database is protected. What this means is that data in the database are always/most of the time accurate.
3. Standardization of the data, which implies that a single data can only be entered in a particular way/format.
4. Provides a simple, elegant and low cost solution to solve a companywide problem rather than a single user problem.

# CHAPTER 2: REQUIREMENT ANALYSIS

Each of Ryan’s client has a client name, and email address associated with the client, a unique ID number and the scale. The scale for a client refers to the size of the company that the client represents – small, medium or large. The client buys the knowledge from FXH Arizona. When a client buys the knowledge, the payment details are recorded. Thus include Payment ID, Mode of payment, Start date, End date, a fee to the vendor that provided the contents for that particular knowledge, and the fee to the provider. *This payment details are validated by the accounting staffs of the vendors. The accounting staffs may have multiple certifications that they have undergone*.

Each knowledge has a Knowledge ID associated with it. Other than the ID, the knowledge also contains the Knowledge name and the amount that the knowledge costs. This knowledge has knowledge details such as Start date, Duration of validity as well as the cost. The duration of the knowledge is the period of contract is defined by Ryan when he creates a knowledge for his client.

Each knowledge contains multiple knowledge nugget. Each knowledge nugget is provided by different providers/vendors. Therefore, each knowledge can be a conglomerate of different knowledge nuggets provided by different knowledge providers. This is captured in the form of contribution where the weightage of provider content is calculated/found and stored. **CID** is the identifier for the contribution.

The knowledge nugget contains the name of the knowledge nuggets, the maturity level of the company that the knowledge nugget would belong to (in a company point of view), the company size (or scale of the company) and a unique ID. Each of these knowledge nuggets could be any of the following: video, power point, worksheet amongst others deemed necessary. **Path** is used to record the URL address of the nuggets.

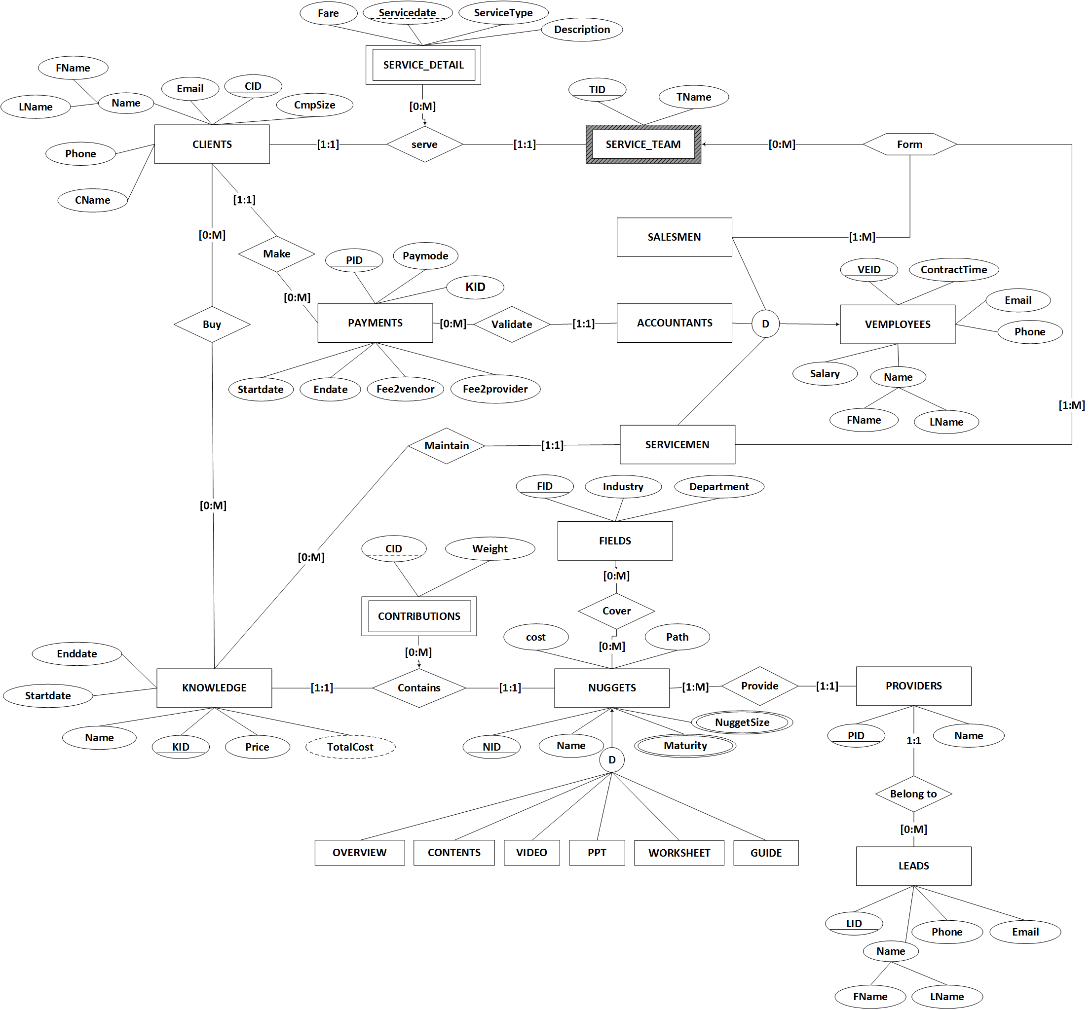
Each provider provides a knowledge nugget and they have a name and ID number associated with each of these providers. For each of the provider we also collect the engineer details who is the person that was responsible for signing the contract with Ryan. Although the vendor buys the knowledge from providers, it is the exact engineers or salesperson of the providers who actually conduct the deals.  We capture details such as EmployeeID, Name, Email address and Phone number on the engineers of these providers.

The knowledge maintenance, accounting validation (as mentioned in the first paragraph) as well as customer service are handled by the vendor employees. Vendor employees can be classified into Customer service employees, Accounting staffs as well as Technical Service employees. For all the employees we record the VendorID, contract time, Email, Phone, Salary and Name.

The customer service employee are a part of the customer service team. Each of the customer service team have a TeamID, level of service as well as the fare being recorded. When they serve the clients, they record the same details, along with the date as well as the service fee charged.

Client Notes: The attribute CMM Level, Department and Business scale would be used as criteria for filtering the results. A client may end up purchasing more than one knowledge and each of the knowledge can be bought separately.

# CHAPTER 3: ER DIAGRAM



## DATA DICTIONARY (CONCEPTUAL / FOR ER MODELLING) PART 1 of 2

## 

## DATA DICTIONARY (CONCEPTUAL / FOR ER MODELLING) PART 2 of 2



# CHAPTER 4: RELATIONAL SCHEMA

Clients (CID, FName, LName, Cname, CmpSize, Phone, Email)

VEmployees (VEID, ContractTime, FName, LName, Salary, Phone, Email)

Salesmen (VEID)

FOREIGN KEY(VEID) REFERENCES VEmployees (VEID)

Servicemen(VEID)

FOREIGN KEY(VEID) REFERENCES VEmployees (VEID)

Accountants (VEID)

FOREIGN KEY(VEID) REFERENCES VEmployees (VEID)

Service\_Team (TID, Name)

Providers (PID, Name)

Leads (LID, PID, FName, LName, Phone, Email)

FOREIGN KEY(PID) REFERENCES Providers (PID)

Nuggets (NID, Name, Path, PID)

FOREIGN KEY(PID) REFERENCES Providers (PID)

Fields (FID, Industry, Department)

Nugget\_Maturity (NID, Maturity)

FOREIGN KEY(NID) REFERENCES Nuggets(NID)

Nugget\_Size (NID, NuggetSize)

FOREIGN KEY(NID) REFERENCES Nuggets(NID)

Overview (NID)

FOREIGN KEY(NID) REFERENCES Nuggets (NID)

PPT(NID)

FOREIGN KEY(NID) REFERENCES Nuggets (NID)

Video(NID)

FOREIGN KEY(NID) REFERENCES Nuggets (NID)

Worksheet(NID)

FOREIGN KEY(NID) REFERENCES Nuggets (NID)

Guide(NID)

FOREIGN KEY(NID) REFERENCES Nuggets (NID)

Contents(NID)

FOREIGN KEY(NID) REFERENCES Nuggets (NID)

Knowledge (KID, VEID, Startdate, Enddate, Price, TotalCost, Name)

FOREIGN KEY(VEID) REFERENCES Servicemen(VEID)

Payments (PID, Paymode, Startdate, Enddate, Fee2provider, Fee2vendor, KID, VEID, CID)

FOREIGN KEY(KID) REFERENCES Knowledge (KID),

FOREIGN KEY(VEID) REFERENCES Accountants (VEID),

FOREIGN KEY(CID) REFERENCES Clients (CID)

Contributions (CID, KID, NID, Weight)

FOREIGN KEY(KID) REFERENCES Knowledge (KID),

FOREIGN KEY(NID) REFERENCES Nuggets (NID),

PRIMARY KEY(CID,KID,NID)

Service\_Detail (CID, TID, Servicedate, Serivcetype, Fare, Description)

FOREIGN KEY(CID) REFERENCES Clients (CID),

FOREIGN KEY(TID) REFERENCES Service\_Team (TID)

Form\_Salesmen(VEID, TID,

FOREIGN KEY(VEID) REFERENCES Salesmen (VEID),

FOREIGN KEY(TID) REFERENCES Service\_Team (TID)

Form\_Servicemen(VEID, TID)

FOREIGN KEY(VEID) REFERENCES Servicemen (VEID),

FOREIGN KEY(TID) REFERENCES Service\_Team (TID)

Buy (CID, KID)

FOREIGN KEY(CID) REFERENCES Clients (CID),

FOREIGN KEY(KID) REFERENCES Knowledge (KID),

Cover (FID, NID)

FOREIGN KEY(FID) REFERENCES Fields (FID),

FOREIGN KEY(NID) REFERENCES Nuggets (NID)

## DATA DICTIONARY (RELATIONAL / FOR ER MODELLING) PART 1 of 2



## DATA DICTIONARY (RELATIONAL / FOR ER MODELLING) PART 2 of 2



# CHAPTER 5: DATA POPULATION AND QUERIES

**Query 1:** We are fetching the details of all existing knowledge based on their type that is recognized by putting maturity level, department type, and the size of the organization in the filters. A join has been implemented to retrieve the knowledge names since there are multiple tables where the conditions are being applied and to which these are being checked for.

Code:

//

SELECT kid, startdate, price

FROM knowledge

WHERE kid IN

(SELECT c.kid

FROM contributions c

WHERE c.nid IN

(SELECT k.nid

FROM nuggets k

WHERE k.nid IN

(SELECT m.nid

FROM nugget\_maturity m,

cover c,

fields f,

nugget\_size s

WHERE m.nid = c.nid

AND c.fid = f.fid

AND s.nid = m.nid

AND m.maturity = ' '

AND f.department = ' '

AND S.NUGGETSIZE = ' '

)

)

)

**Query 2**: We are retrieving all the nuggets that a particular knowledge contains. We have used multiple level of subqueries to fetch the recordset. The table NUGGETS contain these nugget records which we are getting by writing a where clause that would take nugget ids from the intermediate table i.e CONTRIBUTIONS. The table CONTRIBUTIONS is using KID as a condition which is the ID for the Knowledge name that we want the nugget details for.

Code:

//

SELECT n.name

FROM nuggets n

WHERE n.nid IN

(SELECT c.nid

FROM contributions c

WHERE c.kid IN

(SELECT k.kid FROM knowledge k WHERE k.name = ''

)

);

**Query 3**: This query is fetching all nuggets that are not part of the selected knowledge but they belong to the category for which we want to create a knowledge. The category includes Maturity level, the department and the company size. We are just extracting those records that are not part of the selected knowledge so we are using a MINUS clause here. The minus class will remove all nuggets that the selected knowledge contains and displays rest of the record satisfying the search criteria.

Code:

//

SELECT k.name

FROM nuggets k

WHERE k.nid IN

(SELECT m.nid

FROM nugget\_maturity m,

cover c,

fields f,

nugget\_size s

WHERE m.nid = c.nid

AND c.fid = f.fid

AND s.nid = m.nid

AND m.maturity = ' '

AND f.department = ' '

AND S.NUGGETSIZE = ' '

)

MINUS

(SELECT n.name

FROM nuggets n

WHERE n.nid IN

(SELECT c.nid

FROM contributions c

WHERE c.kid IN

(SELECT k.kid FROM knowledge k WHERE k.name = ' '

)

)

);

**Query 4:** Based on the same logic as above, we are fetching the list of knowledge nuggets based on Department, CMM Level, and Business Scale.

Code:

//

SELECT k.nid

FROM nuggets k

WHERE k.nid IN

(SELECT m.nid

FROM nugget\_maturity m,

cover c,

fields f,

nugget\_size s

WHERE m.nid = c.nid

AND c.fid = f.fid

AND s.nid = m.nid

AND m.maturity = ' '

AND f.department = ' '

AND S.NUGGETSIZE = ' '

);

**Query 5:** All the customers who have bought the knowledge are being fetched using this query. We have a join of three tables to fetch only those customers who have bought the knowledge.

Code:

//

SELECT c.fname

|| ' '

|| c.lname "Customer Name",

c.phone,

c.email,

k.name "Knowledge Name"

FROM clients c,

buy b,

knowledge k

WHERE c.cid = b.cid

AND b.kid = k.kid

**Query 6:** This query would help to display people who have bought a certain knowledge belonging to a particular department, industry size and maturity level. A join has been performed for multiple tables which contain the desirable recordset.

Code:

//

SELECT distinct cl.fname

|| ' '

|| cl.lname "Customer Name",

cl.phone,

cl.email,

k.name "Knowledge Name"

FROM nugget\_maturity m,

cover c,

fields f,

nugget\_size s,

nuggets n,

contributions cc,

knowledge k,

buy b,

clients cl

WHERE m.nid = c.nid

AND c.fid = f.fid

AND s.nid = m.nid

AND k.kid = cc.kid

AND cc.nid = n.nid

AND n.nid = m.nid

AND b.kid = k.kid

AND cl.cid = b.cid

AND m.maturity = ''

AND f.department = ''

AND S.NUGGETSIZE = '';

**Query 7:** This query helps in fetching a report that contains list of people who have bought the knowledge. It provides the knowledge name of a particular knowledge that a customer has bought. It also displays the name of the provider who provides this knowledge.

Code:

//

SELECT x."CNAME",

x."MODE",

x."DATE",

x."PFEE",

x."VFEE",

y."KNAME",

y."PNAME"

FROM

(SELECT c1.fname

|| ' '

|| c1.lname "CNAME",

p1.paymode "MODE",

p1.startdate "DATE",

p1.fee2provider "PFEE",

p1.fee2vendor "VFEE",

p1.KID "KID"

FROM payments p1,

clients c1

WHERE p1.cid = c1.cid

AND to\_char(p1.startdate, 'dd/mm/yyyy') > ''

AND to\_char(p1.startdate, 'dd/mm/yyyy') < ''

) x,

(SELECT distinct k.name "KNAME",

k.kid "ID",

p.name "PNAME"

FROM nugget\_maturity m,

cover c,

fields f,

nugget\_size s,

contributions co,

knowledge k,

nuggets n,

providers p

WHERE m.nid = c.nid

AND n.pid = p.pid

AND c.fid = f.fid

AND s.nid = m.nid

AND n.nid = m.nid

AND co.nid = n.nid

AND k.kid = co.kid

) y

WHERE x."KID" = y."ID";

**Query 8:** A list of customers who have bought the knowledge. It extracts records for the person whose CID is entered in the query. We are using this query in the customer’s page where the customer who is logged in to the system can view the details of the knowledge he has bought.

Code:

//

SELECT k.kid

FROM clients c,

buy b,

knowledge k

WHERE c.cid = b.cid

AND b.kid = k.kid

AND c.cid = ''

;

**Query 9:** Deletes records for a particular customer.

DELETE

FROM buy b

WHERE b.kid IN

(SELECT k.kid

FROM clients c,

buy b,

knowledge k

WHERE c.cid = b.cid

AND b.kid = k.kid

)

AND b.cid = ''

;

**Query 10:** Display customer details

Code:

//

SELECT fname || ' ' || lname,

cname,

cmpsize,

phone,

email

FROM clients

WHERE cid =

;

# CHAPTER 6: TRIGGERS AND PROCEDURES

**Trigger 1: Recalculate the weight of nugget in a specific knowledge after assigning a new nugget into a knowledge.**

Each knowledge consists of several nuggets. The total cost and price of knowledge is decided by the nuggets and the weight of that nugget in the knowledge is decided by its own cost and the total cost of the knowledge.

This trigger will update the total cost and weigh of each nugget in the knowledge that you assigned a new nugget in. And it will refuse the insert request, if a transaction is already made about this knowledge.

Code:

//

Create or replace trigger recal\_weight\_insert

before insert

on contributions

for each row

declare

pre\_rec number(10,0);

temp\_totalcost knowledge.totalcost%type;

temp\_cost nuggets.cost%type;

cursor\_cost nuggets.cost%type;

cursor c1 is select kid, nid

from contributions

where kid = :new.kid

for update of weight;

begin

SELECT 'CON' || to\_char(fields\_seq.nextval) INTO :new.CID FROM dual;

select count(\*) into pre\_rec from buy where kid = :new.kid;

if (pre\_rec > 0)

then

raise\_application\_error('-20001','An transction already made, please create a new knowledge');

rollback;

end if;

select totalcost into temp\_totalcost from knowledge where kid = :new.kid;

select cost into temp\_cost from nuggets where nid = :new.nid;

temp\_totalcost:= temp\_totalcost + temp\_cost;

for rec in c1 loop

select cost

into cursor\_cost

from nuggets

where nid = rec.nid;

update contributions set weight = cursor\_cost/temp\_totalcost\*100 where current of C1;

end loop;

:new.weight := temp\_cost/temp\_totalcost \*100;

update knowledge set totalcost = temp\_totalcost where kid = :new.kid;

update knowledge set price = temp\_totalcost\*1.2 where kid = :new.kid;

exception

when no\_data\_found then

:new.weight := 100;

update knowledge set totalcost = temp\_totalcost where kid = :new.kid;

update knowledge set price = temp\_totalcost\*1.2 where kid = :new.kid;

end;

//

**Trigger 2: Divide the payment into different part and assign to different providers.**

When one payments, our client want to divide the money into several part and physically record how much a provider earned in that transaction. In this trigger we assume that the payment is equals the price of that knowledge, but in real it can be lower or higher.

This trigger will first divide the payment into 2 parts. One is money to our client company and another is money to provider companies. Then the trigger will find which nuggets belong to this knowledge and who are the providers and record these data.

Code:

//

create or replace trigger makepay

before insert

on payments

for each row

declare

Cursor C1 is select nid from contributions where kid = :new.kid;

begin

SELECT 'P' || to\_char(payment\_seq.nextval) INTO :new.PID FROM dual;

INSERT INTO BUY VALUES(:new.cid,:new.kid);

select price - totalcost into :new.fee2vendor from knowledge where kid = :new.kid;

select totalcost into :new.fee2provider from knowledge where kid = :new.kid;

FOR res in C1 LOOP

insert into provider\_revenue values(:new.PID,res.nid,:new.startdate);

END LOOP;

end;

//

**Trigger 3: To generate primary key for tables**

These triggers are only simply generating primary key for new records.

1. The following trigger is used to generate the primary key for adding new clients.

create or replace trigger add\_client

before insert

on clients

for each row

declare

begin

SELECT 'C' || to\_char(client\_seq.nextval) INTO :new.CID FROM dual;

end;

2. The following trigger is used to generate the Field ID within the Field table.

create or replace trigger add\_fields

before insert

on fields

for each row

declare

begin

SELECT 'F' || to\_char(fields\_seq.nextval) INTO :new.FID FROM dual;

end;

3. The following trigger is used to create the KID (Knowledge ID) for the Knowledge table

create or replace trigger add\_knowledge

before insert

on knowledge

for each row

declare

begin

SELECT 'K' || to\_char(knowledge\_seq.nextval) INTO :new.KID FROM dual;

end;

4. The following trigger is used to create the NID (Nugget ID) for the Knowledge table

create or replace trigger add\_nuggets

before insert

on nuggets

for each row

declare

begin

SELECT 'N' || to\_char(nugget\_seq.nextval) INTO :new.NID FROM dual;

end;

5. The following trigger is used to create the PID (Provider ID) for the Provider table

create or replace trigger add\_provider

before insert

on providers

for each row

declare

begin

SELECT 'P' || to\_char(provider\_seq.nextval) INTO :new.PID FROM dual;

end;

**Procedure 1:**

**It is useful for our client to see which provider is valuable to them. So it is important to generate report to know how much sales revenue each provider earned by month.**

This procedure will take 3 parameters: 1. ProviderID, which uses to uniquely determine a provider.2. Report month. 3. Report year.

This procedure will generate report for this provider from last report date to input date.

Code:

//

create or replace procedure month\_report\_procedure(this\_PID month\_report.pid%type,

this\_rmonth month\_report.revmonth%type,

this\_ryear month\_report.revyear%type) AS

last\_modified\_y month\_report.revyear%type;

last\_modified\_m month\_report.revmonth%type;

this\_revenue month\_report.revenue%type;

last\_revenue month\_report.revenue%type;

BEGIN

--Get the last report time--

select revyear,revmonth,revenue

into last\_modified\_y,last\_modified\_m,last\_revenue

from(select \* from month\_report

where pid = this\_PID

order by revyear,revmonth desc)

where rownum = 1;

-- If no previous report for that route

EXCEPTION WHEN NO\_DATA\_FOUND THEN

last\_modified\_m := 1;

last\_modified\_y := this\_ryear;

last\_revenue := 0;

while(last\_modified\_y <= this\_ryear) LOOP

if (last\_modified\_y<this\_ryear)

THEN WHILE(last\_modified\_m < 12) LOOP

select coalesce(SUM(fee2provider\*weight),0)

into this\_revenue

from PAYMENTS P

join CONTRIBUTIONS C on P.KID = C.KID

join NUGGETS N on C.NID = N.NID

where N.PID = this\_PID

And to\_number(extract(month from P.startdate)) = last\_modified\_m + 1

AND to\_number(extract(year from P.startdate)) = last\_modified\_y;

insert into month\_report(PID,revmonth,revyear,revenue) values(this\_PID,last\_modified\_m + 1,last\_modified\_y,this\_revenue);

last\_modified\_m := last\_modified\_m + 1;

END LOOP;

END IF;

if (last\_modified\_y = this\_ryear)

THEN WHILE(last\_modified\_m < this\_rmonth) LOOP

select coalesce(SUM(fee2provider\*weight),0)

into this\_revenue

from PAYMENTS P

join CONTRIBUTIONS C on P.KID = C.KID

join NUGGETS N on C.NID = N.NID

where N.PID = this\_PID

And to\_number(extract(month from P.startdate)) = last\_modified\_m + 1

AND to\_number(extract(year from P.startdate)) = last\_modified\_y;

insert into month\_report(PID,revmonth,revyear,revenue) values(this\_PID,last\_modified\_m + 1,last\_modified\_y,this\_revenue);

last\_modified\_m := last\_modified\_m + 1;

END LOOP;

END IF;

last\_modified\_y := last\_modified\_y + 1;

last\_modified\_m := 0;

END LOOP;

END;

//

# CHAPTER 7: INTERFACE AND REPORTS

**Website Address: 54.213.37.16**

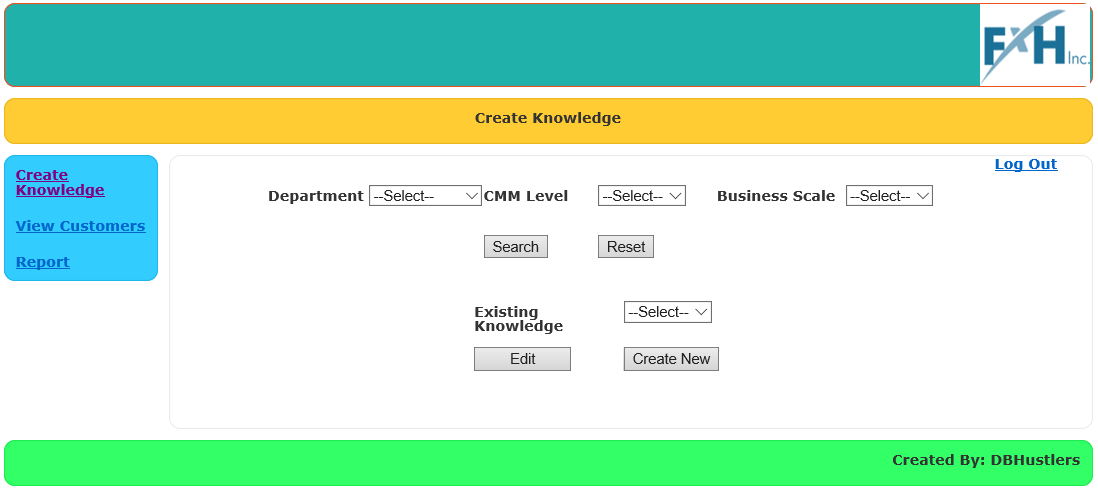
## ADMIN LOGIN

**Username: admin**

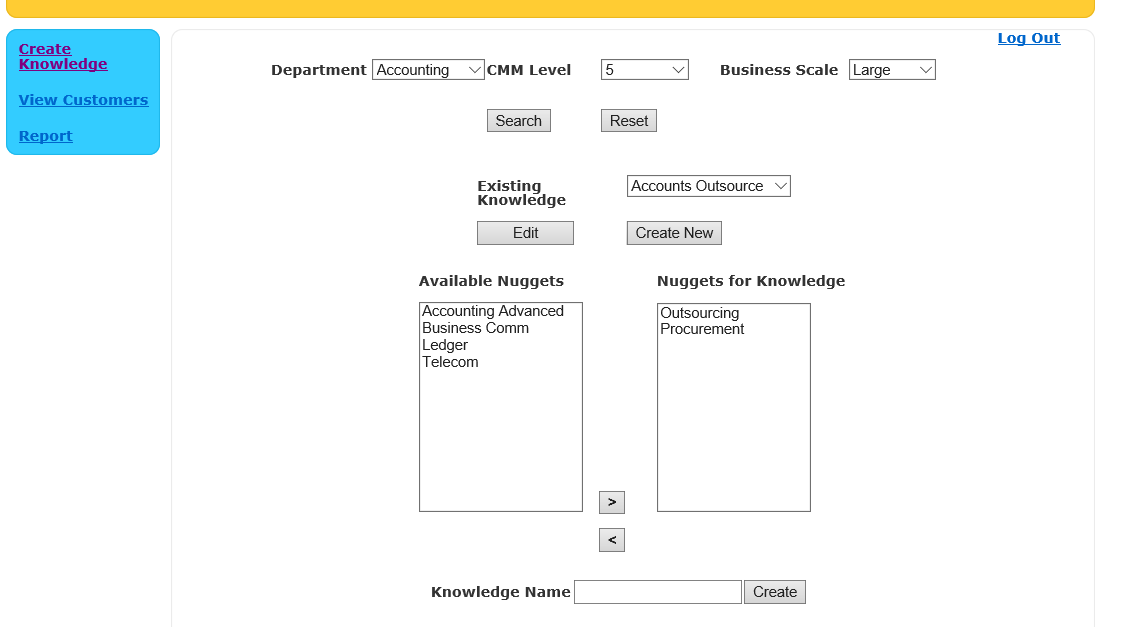
**Password: abc123**



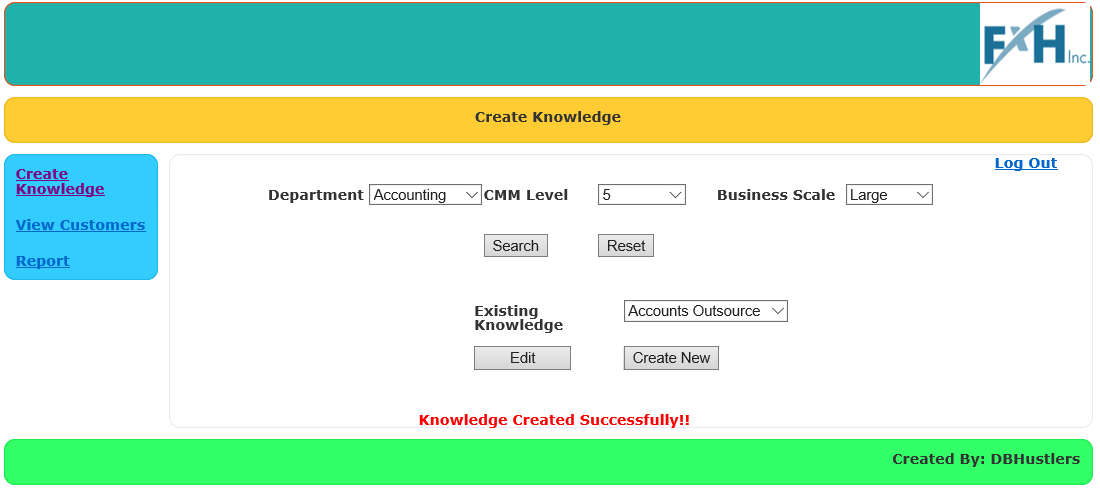
Create Knowledge Page:



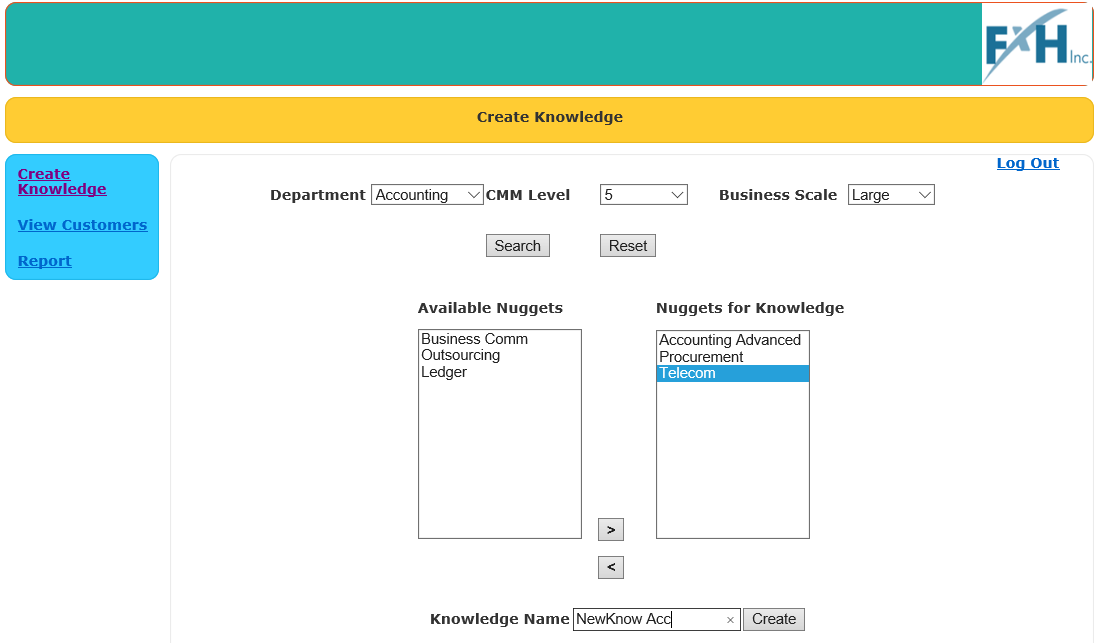
Select Department, CMM Level and Business Scale values. Depending on the selected values Existing Knowledge will be populated. If you want to create a knowledge from one of the existing knowledges, select a knowledge from the dropdown and click on edit.



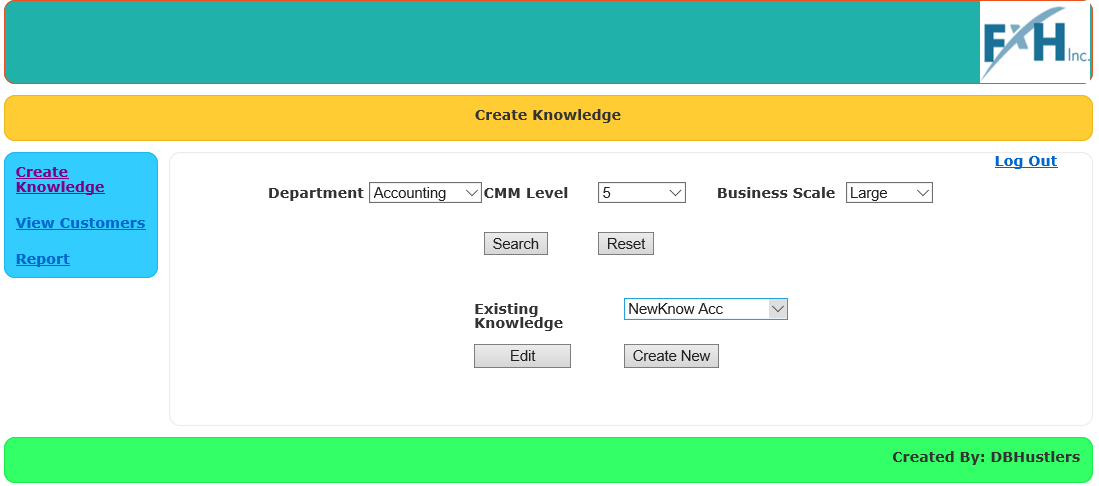
Add or delete nuggets from the listboxes by selecting a nugget and clicking on the arrow buttons. Give new knowledge name and click on Create button. Success message will be displayed.



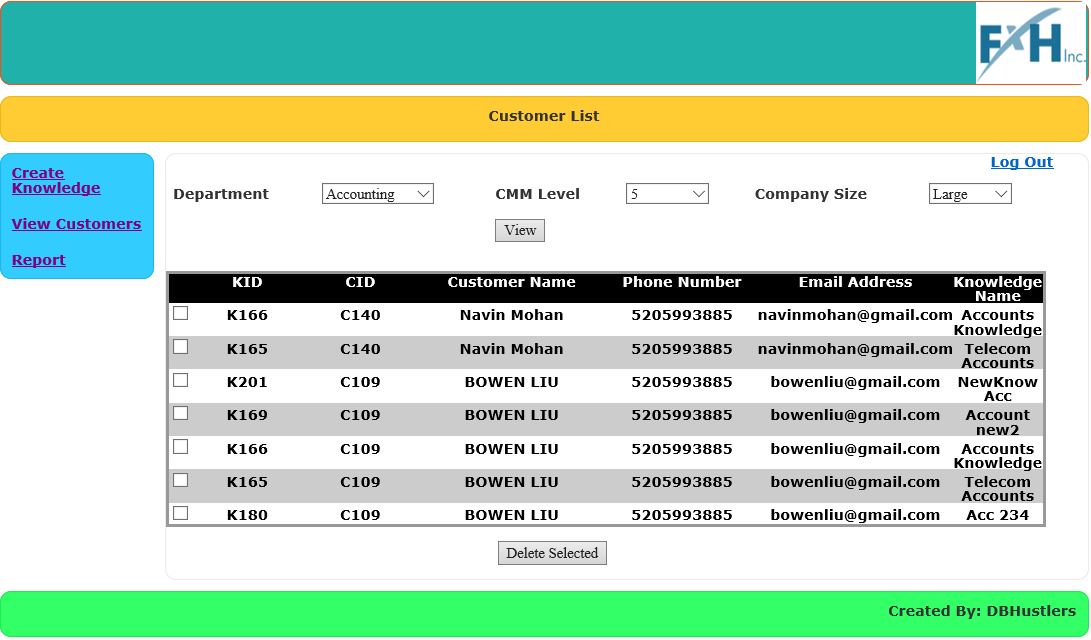
To create a new knowledge without using existing knowledges, click on Create New button.



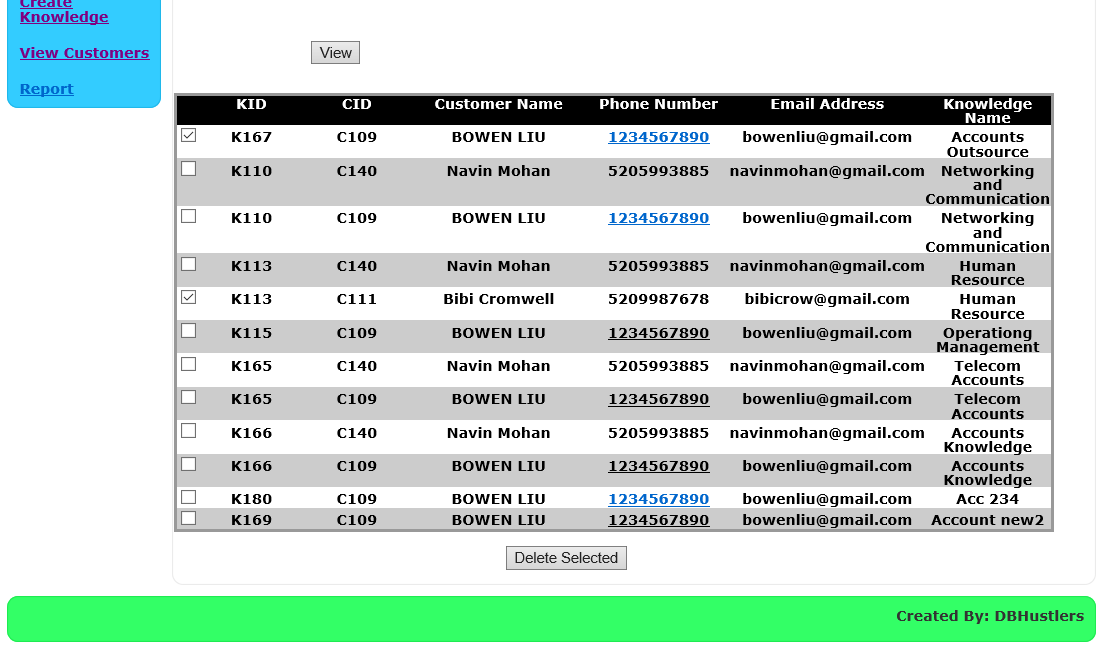
After selecting nuggets and giving a name to the knowledge, click on Create button. Knowledge will be created.



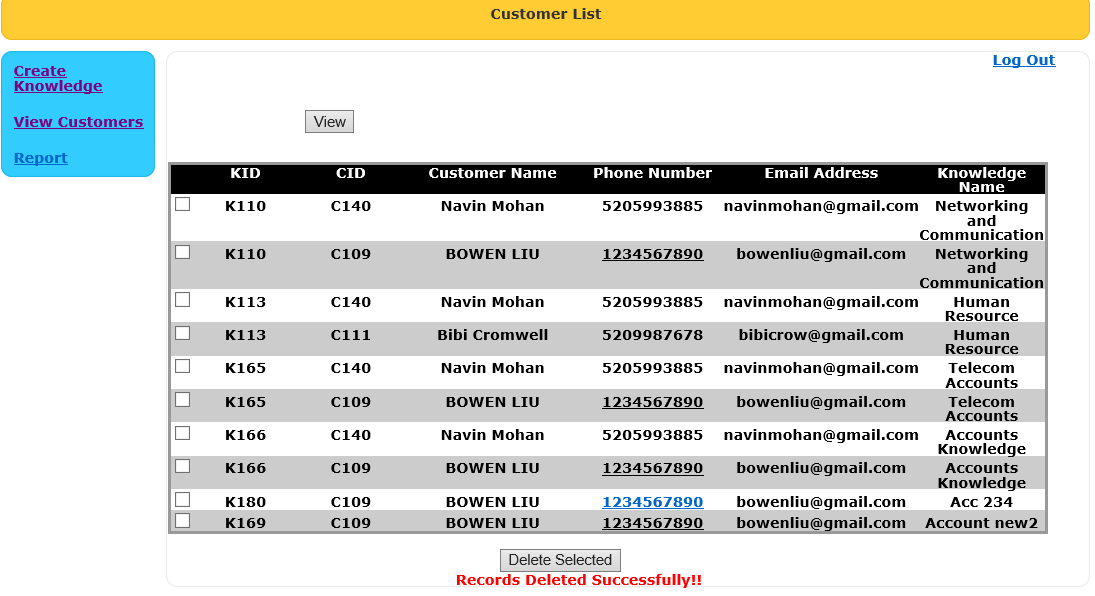
View Customers page: The admin would be able to filter the client list based on the 3 criterions that are displayed in the webpage below : Department, CMM Level and Company Size.



Select the checkboxes to delete knowledge of a particular user.

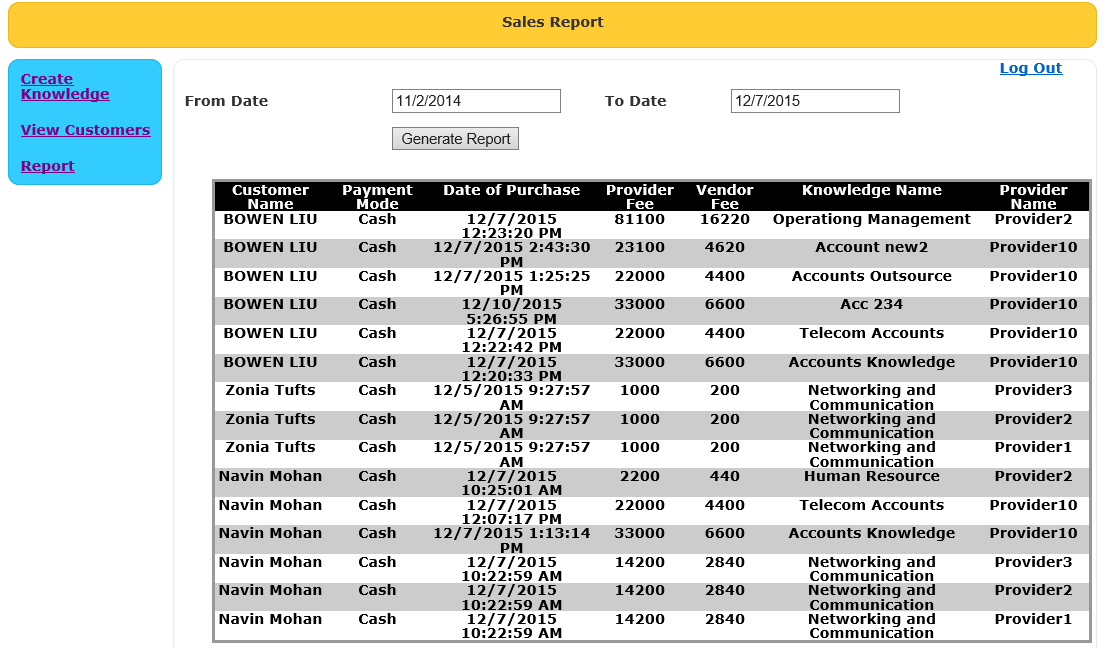


Click on delete Selected button



Report Page:

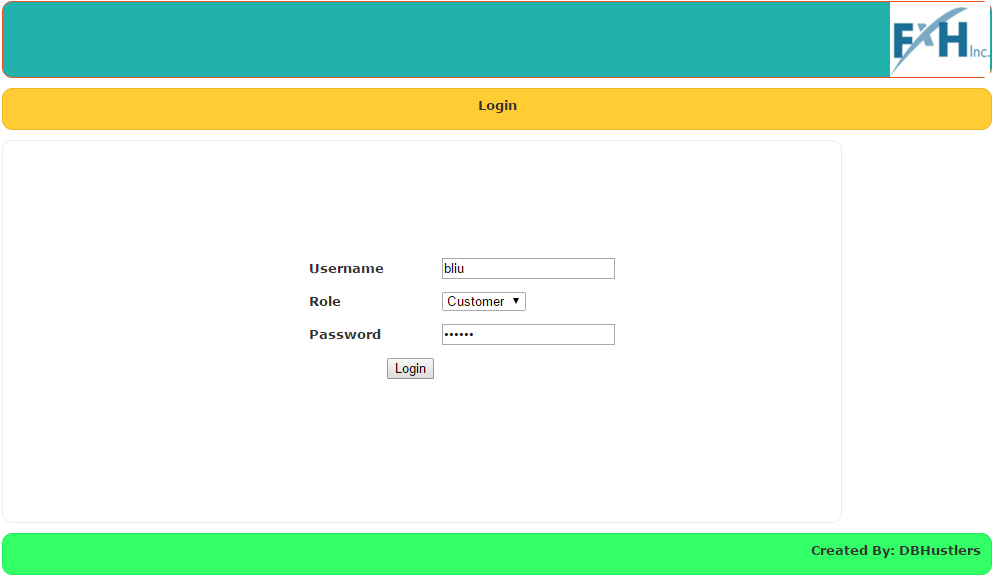
Select to date and from date and click on Generate Report button.



## Customer Login:

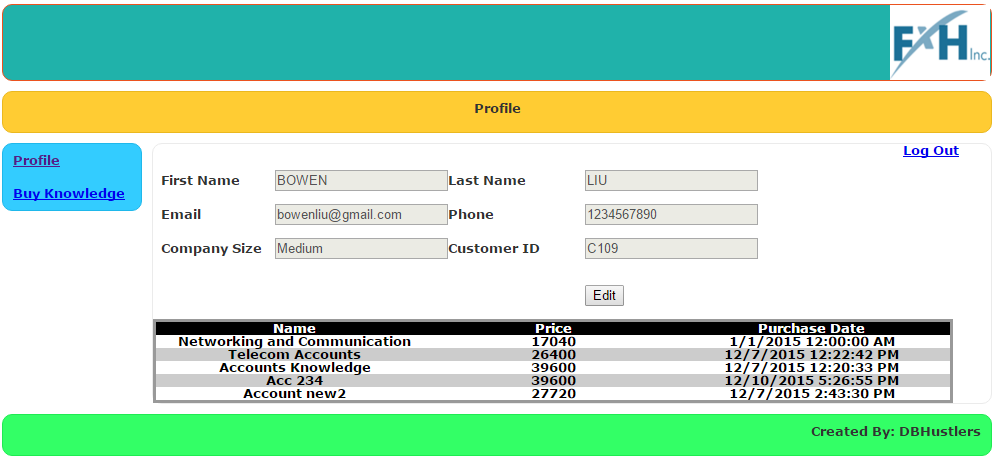
**Username: bliu**

**Password: abc123**

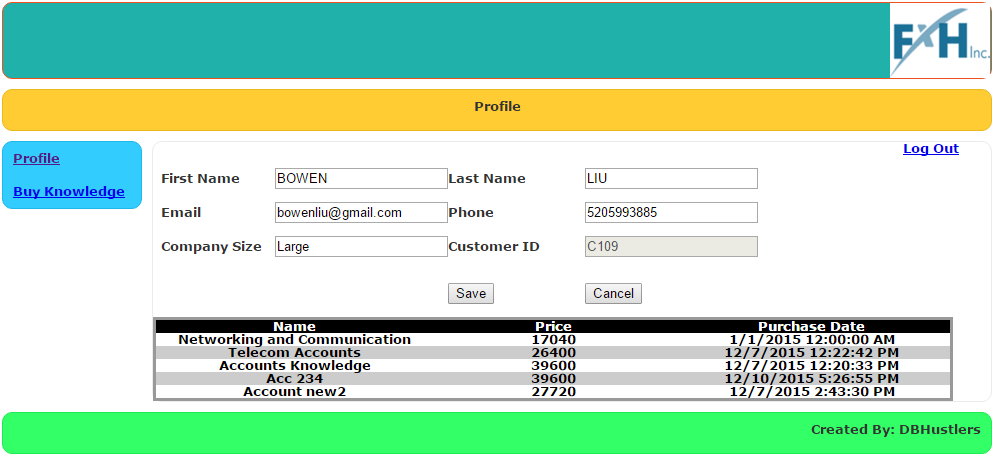


Landing page after login is Customer Profile page:

It shows the customer details along with knowledges bought by him/her.

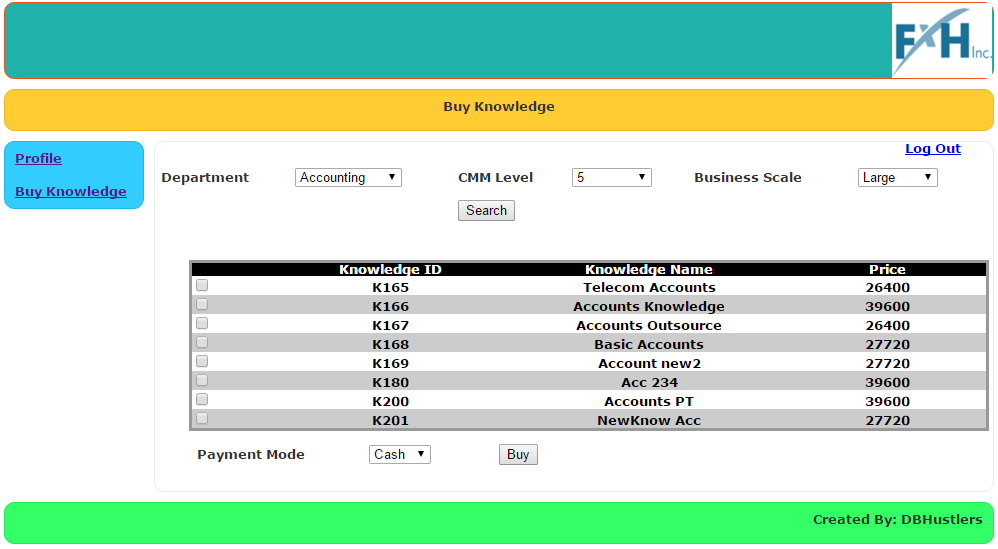


To change customer details click on Edit button. Enter the updated fields in the textbox and click on save button.



Buy Knowledge Page:

Select the dropdown values and click on search button



To buy knowledge(s), check the knowledges to buy, select the payment mode and click on Buy.



# CHAPTER 8: CONCLUSION AND IMPLEMENTATION PLAN

## Implementation Plan

DBHustlers would provide the back end of the database to our client FXH Inc. Although we are also providing the client with the website design as well as the URL for the website, the client looks to make use of our database to create his own system moving forward.

The following would be a part of the docket that would be submitted to the client:

* Web Application
* Source Code Access
* Procedures (Stored)
* Triggers
* SQL Queries
* Insert scripts
* Table creation scripts
* Database dump file (Sample data file)

## Installation Plan

Installation plan for the deployment of the website in a real world scenario

The system would be sequentially installed in 3 phases.

* Phase I would take approximately 3-4 weeks. The long lead time is for the approval of the contract as well as the purchase of the hardware. For purchase of the hardware the client would send out a RFP to understand the market rates for the servers. The servers can be of two types: Database servers as well as Web server. Other than the servers the client would also have to invest in database software licenses and operating system licenses.
* Phase II would include the physical installation of the server. This may be done over a period of 1-2 weeks. This phase also includes the installation of the software on the database server as well as on the web server.
* Phase III would include the deployment of the database system on the server. The website is hosted in the web server. The testing of these both servers are done to check for any errors. Once the system is clear (after quality check) the database is populated with the data manually. Once the database is populated, another check run is done before deployment to the client. This is to ensure that the queries, procedures and triggers that are included in the project run without any hiccup. This also ensures that the reports that the client requires also run without any issue.

## System Cost

If our client does decide to purchase hardware and software for the sake of this project the following table would elaborate on the costs that he would incur. The man hours that are mentioned in the below mentioned table is a rough estimate of the man hours that our team spent on the project. Assumptions were used for the labor rates.

|  |  |  |
| --- | --- | --- |
| Item | Description | Amount |
| Database Server | Server to host database | $ 2500 |
| Web Server | Server to host the website | $ 1500 |
| SSL Certificate | Datafiles that bind the key to a particular organization details | $ 300 per year |
| Software 1 |  |  |
| Software 2 |  |  |
| Installation | Installation of the servers (manual labor of $40/hour and assuming 50 hours) | $ 2000 |
| Total |  |  |

The total cost of the system along with the software licenses as well as the installation would tally up to be close to $XYZ.

## Conclusion

The project was completed meeting all requirements of our client, FXH Inc. and the team has already sent the client the website address, database design, ER diagrams, SQL dump of sample data, Triggers and procedures codes among other critical components of the project. The database shall allow Ryan to gather information on his providers as well as generate reports to better his business that he plans to start in the near future.

## Lessons Learned

The below shall detail the lessons learnt by the team in each of the project processes from Requirement gathering till Implementation of the database on AWS server.

* Requirement Gathering

This process is most critical part of the project and can make or break a project based on the understanding of the client requirement by the team. It is crucial that the client is clear on what he requires the database to do and also ensure that the same is conveyed in clear and concise manner to the consultants (DBHustlers). In our case, since this project was a pet project of Ryan, and he had no idea on how the database would be used the team had to ensure that we met with him on at least a bi-weekly basis to better understand his requirements as well as to note any changes that he may have thought of.

* ER Diagram Design

This step is used to understand and decide how the data would flow in a database. We make use of the ER Diagram to display to the client on the entities within his database and how they interact with one another. Since our client had prior knowledge of an ER Diagram, our team found it easier to send him ER diagrams that were revised each time based on client meetings/email to get a clarity on the client requirement.

* Importance of Meetings

It is important that the team realizes that the requirements for the client are never set in stone and each meeting that we had gave us new insight into his requirement. To make sure that these are captured in the ER as well as while designing our website, we always made sure that we met immediately after the client meeting to quickly discuss the main discussion points of the meeting with the client. This was also recorded in a google drive folder so that people can refer back to the same in the future.

* Presentation Preparations

Our team faced issues while presenting in class. This was due to the fact that we had not prepared before on the facilities that were present in the class. Our team learned a valuable lesson that one must prepare beforehand for such thing to happen and must have a back-up plan to negate such experiences. Luckily we could present using another student’s laptop and use skype as a mode to screen share our live prototype of the website.

* Task distribution based on strengths

The team made sure that that tasks are evenly tasked out, so that not one member of the team is overwhelmed with tasks. While dividing up our tasks, the team made sure that each of the tasks were allocated based on their strengths. For example, one of our team members was really good at creating trigger and procedures so we tasked him the duty of creating them for the project. That said, we also had regular check in points to ensure that the tasks given are proceeding as per plan and in a timely fashion.

# APPENDIX 1: TABLE CREATION

-- SQL for schema: 007 -- 20 Entity tables + 2 multivalue tables + 4 relational tables

-- Drop relationship tables

DROP TABLE Buy CASCADE CONSTRAINTS;

DROP TABLE Cover CASCADE CONSTRAINTS;

DROP TABLE Form\_Salesmen CASCADE CONSTRAINTS;

DROP TABLE Form\_Servicemen CASCADE CONSTRAINTS;

-- Drop Multivalue tables

DROP TABLE Nugget\_Maturity CASCADE CONSTRAINTS;

DROP TABLE Nugget\_Size CASCADE CONSTRAINTS;

-- Drop entity class tables

DROP TABLE Accountants CASCADE CONSTRAINTS;

DROP TABLE Contents CASCADE CONSTRAINTS;

DROP TABLE Contributions CASCADE CONSTRAINTS;

DROP TABLE Clients CASCADE CONSTRAINTS;

DROP TABLE Fields CASCADE CONSTRAINTS;

DROP TABLE Guide CASCADE CONSTRAINTS;

DROP TABLE Knowledge CASCADE CONSTRAINTS;

DROP TABLE Leads CASCADE CONSTRAINTS;

DROP TABLE Nuggets CASCADE CONSTRAINTS;

DROP TABLE Overview CASCADE CONSTRAINTS;

DROP TABLE PPT CASCADE CONSTRAINTS;

DROP TABLE Payments CASCADE CONSTRAINTS;

DROP TABLE Providers CASCADE CONSTRAINTS;

DROP TABLE Salesmen CASCADE CONSTRAINTS;

DROP TABLE Service\_Detail CASCADE CONSTRAINTS;

DROP TABLE Service\_Team CASCADE CONSTRAINTS;

DROP TABLE Servicemen CASCADE CONSTRAINTS;

DROP TABLE VEmployees CASCADE CONSTRAINTS;

DROP TABLE Video CASCADE CONSTRAINTS;

DROP TABLE Worksheet CASCADE CONSTRAINTS;

-- Strong Entity Class

CREATE TABLE Clients (CID Varchar2(20),

FName Varchar2(20) NOT NULL, LName Varchar2(20) NOT NULL,

Cname Varchar2(50) NOT NULL, CmpSize Varchar2(10),

Phone Number(10) NOT NULL, Email Varchar2(50) NOT NULL,

CONSTRAINT Clients\_pk PRIMARY KEY(CID),

CONSTRAINT check\_CmpSize CHECK (CmpSize IN ('Small','Medium','Large')))

;

-- Strong Entity Class for all vendor employees

CREATE TABLE VEmployees (VEID Varchar2(20),

ContractTime Number(2) NOT NULL, FName Varchar2(20) NOT NULL,

LName Varchar2(20) NOT NULL, Salary Number(10) NOT NULL,

Phone Number(10) NOT NULL UNIQUE, Email Varchar2(50) NOT NULL UNIQUE,

CONSTRAINT VEmployees\_pk PRIMARY KEY(VEID))

;

-- Subclass of VEmployees

CREATE TABLE Salesmen (VEID Varchar2(20),

CONSTRAINT Salesmen\_fk FOREIGN KEY(VEID) REFERENCES VEmployees (VEID),

CONSTRAINT Salesmen\_pk PRIMARY KEY(VEID))

;

-- Subclass of VEmployees

CREATE TABLE Servicemen(VEID Varchar2(20),

CONSTRAINT Servicemen\_fk FOREIGN KEY(VEID) REFERENCES VEmployees (VEID),

CONSTRAINT Servicemen\_pk PRIMARY KEY(VEID))

;

-- Subclass of VEmployees

CREATE TABLE Accountants (VEID Varchar2(20),

CONSTRAINT Accountants\_fk FOREIGN KEY(VEID) REFERENCES VEmployees (VEID),

CONSTRAINT Accountants\_pk PRIMARY KEY(VEID))

;

-- Grouping Entity Class

CREATE TABLE Service\_Team (TID Varchar2(20),

Name Varchar2(20) NOT NULL, CONSTRAINT Service\_Team\_pk PRIMARY KEY(TID))

;

-- Strong Entity Class

CREATE TABLE Providers (PID Varchar2(20),

Name Varchar2(50) NOT NULL, CONSTRAINT Providers\_pk PRIMARY KEY(PID))

;

-- Strong Entity Class

CREATE TABLE Leads (LID Varchar2(20), PID Varchar2(20),

FName Varchar2(20) NOT NULL, LName Varchar2(20) NOT NULL,

Phone Number(10)NOT NULL UNIQUE,Email Varchar(50) NOT NULL UNIQUE,

CONSTRAINT Leads\_pk PRIMARY KEY(LID),CONSTRAINT Leads\_fk FOREIGN KEY(PID) REFERENCES Providers (PID))

;

-- Strong Entity Class

CREATE TABLE Nuggets (NID Varchar2(20),Name Varchar2(20) NOT NULL,Path Varchar2(50) NOT NULL,PID Varchar2(20) NOT NULL,Cost Number(10) NOT NULL, CONSTRAINT Nuggets\_pk PRIMARY KEY(NID), CONSTRAINT Nuggets\_fk FOREIGN KEY(PID) REFERENCES Providers (PID))

;

-- Strong Entity Class

CREATE TABLE Fields (

FID Varchar2(20),

Industry Varchar2(20) NOT NULL, Department Varchar2(20) NOT NULL, CONSTRAINT Fields\_pk PRIMARY KEY(FID))

;

-- Multivalued Attribute of Nuggets

CREATE TABLE Nugget\_Maturity (NID Varchar2(20),Maturity Number(1),

CONSTRAINT Maturity\_fk FOREIGN KEY(NID) REFERENCES Nuggets(NID),

CONSTRAINT Maturity\_pk PRIMARY KEY(NID,Maturity),CONSTRAINT check\_maturity CHECK (Maturity BETWEEN 1 AND 5))

;

-- Multivalued Attribute of Nuggets

CREATE TABLE Nugget\_Size(NID Varchar2(20), NuggetSize Number(1),CONSTRAINT Nugget\_Size\_fk FOREIGN KEY(NID) REFERENCES Nuggets(NID),

CONSTRAINT Nugget\_Size\_pk PRIMARY KEY(NID,NuggetSize),

CONSTRAINT check\_NuggetSize CHECK (NuggetSize BETWEEN 1 AND 5))

;

-- Subclass of Nuggets

CREATE TABLE Overview (NID Varchar2(20),CONSTRAINT Overview\_fk FOREIGN KEY(NID) REFERENCES Nuggets (NID), CONSTRAINT Overview\_pk PRIMARY KEY(NID))

;

-- Subclass of Nuggets

CREATE TABLE PPT(NID Varchar2(20),CONSTRAINT PPT\_fk FOREIGN KEY(NID) REFERENCES Nuggets (NID), CONSTRAINT PPT\_pk PRIMARY KEY(NID))

;

-- Subclass of Nuggets

CREATE TABLE Video(NID Varchar2(20),

CONSTRAINT Video\_fk FOREIGN KEY(NID) REFERENCES Nuggets (NID),

CONSTRAINT Video\_pk PRIMARY KEY(NID))

;

-- Subclass of Nuggets

CREATE TABLE Worksheet(NID Varchar2(20),

CONSTRAINT Worksheet\_fk FOREIGN KEY(NID) REFERENCES Nuggets (NID),

CONSTRAINT Worksheet\_pk PRIMARY KEY(NID))

;

-- Subclass of Nuggets

CREATE TABLE Guide(NID Varchar2(20),CONSTRAINT Guide\_fk FOREIGN KEY(NID)

REFERENCES Nuggets (NID),

CONSTRAINT Guide\_pk PRIMARY KEY(NID))

;

-- Subclass of Nuggets

CREATE TABLE Contents(NID Varchar2(20),

CONSTRAINT Contents\_fk FOREIGN KEY(NID) REFERENCES Nuggets (NID),

CONSTRAINT Contents\_pk PRIMARY KEY(NID))

;

-- Strong Entity Class

CREATE TABLE Knowledge (KID Varchar2(20),VEID Varchar2(20),

Startdate DATE NOT NULL, Enddate DATE NOT NULL, Price Number(10,2) NOT NULL,

TotalCost Number(10,2) NOT NULL, Name Varchar2(30) NOT NULL,

CONSTRAINT Knowledge\_pk PRIMARY KEY(KID),

CONSTRAINT Knowledge\_fk FOREIGN KEY(VEID) REFERENCES Servicemen(VEID))

;

-- Strong Entity Class

CREATE TABLE Payments (PID Varchar2(20),Paymode Varchar2(10),Startdate Date NOT NULL,Enddate Date NOT NULL,Fee2provider Number(10,2) NOT NULL, Fee2vendor Number(10,2) NOT NULL, KID Varchar2(20),

VEID Varchar2(20), CID Varchar2(20),CONSTRAINT Payments\_pk PRIMARY KEY(PID),

CONSTRAINT Payments\_fk0 FOREIGN KEY(KID) REFERENCES Knowledge (KID),

CONSTRAINT Payments\_fk1 FOREIGN KEY(VEID) REFERENCES Accountants (VEID),

CONSTRAINT Payments\_fk2 FOREIGN KEY(CID) REFERENCES Clients (CID),

CONSTRAINT check\_paymode CHECK (Paymode IN ('Cheque','Cash','Credit Card')))

;

----------------------------All Weak Classes----------------------------------

Weak Entity Class

CREATE TABLE Contributions (

CID Varchar2(20),

KID Varchar2(20),

NID Varchar2(20),Weight Number(2) NOT NULL,

CONSTRAINT Contributions\_fk0 FOREIGN KEY(KID) REFERENCES Knowledge (KID),

CONSTRAINT Contributions\_fk1 FOREIGN KEY(NID) REFERENCES Nuggets (NID),

CONSTRAINT Contributions\_pk PRIMARY KEY(CID,KID,NID))

;

-- Weak Entity Class

CREATE TABLE Service\_Detail (CID Varchar2(20),

TID Varchar2(20), Servicedate Date, Serivcetype Number(1),

FareNumber(10,2) NOT NULL, Description Varchar2(100) NOT NULL, CONSTRAINT Service\_Detail\_fk0 FOREIGN KEY(CID) REFERENCES Clients (CID),

CONSTRAINT Service\_Detail\_fk1 FOREIGN KEY(TID) REFERENCES Service\_Team (TID),

CONSTRAINT Service\_Detail\_pk PRIMARY KEY(CID,TID,Servicedate),

CONSTRAINT check\_servicetype CHECK (Serivcetype BETWEEN 1 AND 5))

;

-- Base class for aggregate relationship Form

CREATE TABLE Form\_Salesmen(VEID Varchar2(20),TID Varchar2(20),

CONSTRAINT Form\_Salesmen\_fk0 FOREIGN KEY(VEID) REFERENCES Salesmen (VEID),CONSTRAINT Form\_Salesmen\_fk1 FOREIGN KEY(TID) REFERENCES Service\_Team (TID),

CONSTRAINT Form\_Salesmen\_pk PRIMARY KEY(VEID,TID))

;

-- Base class for aggregate relationship Form

CREATE TABLE Form\_Servicemen(VEID Varchar2(20),

TID Varchar2(20),

CONSTRAINT Form\_Servicemen\_fk0 FOREIGN KEY(VEID) REFERENCES Servicemen (VEID),

CONSTRAINT Form\_Servicemen\_fk1 FOREIGN KEY(TID) REFERENCES Service\_Team (TID),

CONSTRAINT Form\_Servicemen\_pk PRIMARY KEY(VEID,TID))

;

-- Binary Many to Many Interaction Relationship

CREATE TABLE Buy (CID Varchar2(20),

KID Varchar2(20),CONSTRAINT Buy\_fk0 FOREIGN KEY(CID) REFERENCES Clients (CID),

CONSTRAINT Buy\_fk1 FOREIGN KEY(KID) REFERENCES Knowledge (KID),

CONSTRAINT Buy\_pk PRIMARY KEY(CID,KID))

;

-- Binary Many to Many Interaction Relationship

CREATE TABLE Cover (FID Varchar2(20),

NID Varchar2(20),CONSTRAINT Cover\_fk0 FOREIGN KEY(FID) REFERENCES Fields (FID),

CONSTRAINT Cover\_fk1 FOREIGN KEY(NID) REFERENCES Nuggets (NID),

CONSTRAINT Cover\_pk PRIMARY KEY(FID,NID))

;

/

# APPENDIX 2: INSERTION OF DATA

--1.For table Payments, buy, provider\_revenue

--Before insert into Payments, please first run the script of creating table, trigger and sequence below:

CREATE SEQUENCE payment\_seq

INCREMENT BY 1

START WITH 1000

MAXVALUE 9999;

Drop table provider\_revenue;

Create table provider\_revenue(

PAYID Varchar2(20), -- paymentID

NID Varchar2(20),

PAYDATE Date,

CONSTRAINT provider\_revenue\_pk PRIMARY KEY(PAYID,NID))

;

create or replace trigger makepay

before insert

on payments

for each row

declare

Cursor C1 is select nid from contributions where kid = :new.kid;

begin

SELECT 'P' || to\_char(payment\_seq.nextval) INTO :new.PID FROM dual;

INSERT INTO BUY VALUES(:new.cid,:new.kid);

select price - totalcost into :new.fee2vendor from knowledge where kid = :new.kid;

select totalcost into :new.fee2provider from knowledge where kid = :new.kid;

FOR res in C1 LOOP

insert into provider\_revenue values(:new.PID,res.nid,:new.startdate);

END LOOP;

end;

--Sample:insert

--INSERT INTO payments(PAYMODE,STARTDATE,ENDDATE,kid,VEID,CID) VALUES('Cash','01-JAN-2015','01-JAN-2019','K001','VE001','C001');

--2. For table client

--Before insert into Payments, please frist run the script of creating table, trigger and sequence below:

CREATE SEQUENCE client\_seq

INCREMENT BY 1

START WITH 100

MAXVALUE 999;

create or replace trigger add\_client

before insert

on clients

for each row

declare

begin

SELECT 'C' || to\_char(client\_seq.nextval) INTO :new.CID FROM dual;

end;

--Sample:insert

--Insert into clients(FNAME,LNAME,CNAME,CMPSIZE,PHONE,EMAIL) VALUES('BOWEN','LIU','DOUBIGONGSI','Small',1234567890,'bowenliu@gmail.com');

--3. For table knowledge:

--Before insert into knowledges, please frist run the script of creating table, trigger and sequence below:

CREATE SEQUENCE knowledge\_seq

INCREMENT BY 1

START WITH 100

MAXVALUE 999;

create or replace trigger add\_knowledge

before insert

on knowledge

for each row

declare

begin

SELECT 'K' || to\_char(knowledge\_seq.nextval) INTO :new.KID FROM dual;

end;

--Sample insert

--insert into knowledge(STARTDATE,ENDDATE,NAME,price,totalcost) VALUES ('02-DEC-2014','02-DEC-2019','Knowledge1',0,0);

the total cost and price will update automatically when you insert data into contribution!

--4.For table provider

--Before insert into provider, please frist run the script of creating table, trigger and sequence below:

CREATE SEQUENCE provider\_seq

INCREMENT BY 1

START WITH 100

MAXVALUE 999;

create or replace trigger add\_provider

before insert

on providers

for each row

declare

begin

SELECT 'P' || to\_char(provider\_seq.nextval) INTO :new.PID FROM dual;

end;

--Sample insert

--insert into providers(NAME) values('Provider1');

--5. For table nugget

--Before insert into nugget, please frist run the script of creating table, trigger and sequence below:

CREATE SEQUENCE nugget\_seq

INCREMENT BY 1

START WITH 100

MAXVALUE 999;

create or replace trigger add\_nuggets

before insert

on nuggets

for each row

declare

begin

SELECT 'N' || to\_char(nugget\_seq.nextval) INTO :new.NID FROM dual;

end;

--Sample insert

--insert into nuggets(NAME,PATH,PID,COST) values('Nuggets11','xxxx','P100',1000);

--

--6. For table Fields

--Before insert into Fields, please frist run the script of creating table, trigger and sequence below:

CREATE SEQUENCE fields\_seq

INCREMENT BY 1

START WITH 100

MAXVALUE 999;

create or replace trigger add\_fields

before insert

on fields

for each row

declare

begin

SELECT 'F' || to\_char(fields\_seq.nextval) INTO :new.FID FROM dual;

end;

--Sample insert

--insert into FIELDS(INDUSTRY, DEPARTMENT) values('IT','MANAGER');

--

--8.For table contribution

--Before insert into contribution, please frist run the script of creating table, trigger and sequence below:

ALTER TABLE CONTRIBUTIONS DROP(weight);

ALTER TABLE CONTRIBUTIONS ADD(weight NUMBER(5,2));

CREATE SEQUENCE contribution\_seq

INCREMENT BY 1

START WITH 100

MAXVALUE 999;

create or replace trigger recal\_weight\_insert

before insert

on contributions

for each row

declare

pre\_rec number(10,0);

temp\_totalcost knowledge.totalcost%type;

temp\_cost nuggets.cost%type;

cursor\_cost nuggets.cost%type;

cursor c1 is select kid, nid

from contributions

where kid = :new.kid

for update of weight;

begin

SELECT 'CON' || to\_char(fields\_seq.nextval) INTO :new.CID FROM dual;

select count(\*) into pre\_rec from buy where kid = :new.kid;

if (pre\_rec = 0)

then

raise\_application\_error('-20001','An transction already made, please create a new knowledge');

rollback;

end if;

select totalcost into temp\_totalcost from knowledge where kid = :new.kid;

select cost into temp\_cost from nuggets where nid = :new.nid;

temp\_totalcost:= temp\_totalcost + temp\_cost;

for rec in c1 loop

select cost

into cursor\_cost

from nuggets

where nid = rec.nid;

update contributions set weight = cursor\_cost/temp\_totalcost\*100 where current of C1;

end loop;

:new.weight := temp\_cost/temp\_totalcost \*100;

update knowledge set totalcost = temp\_totalcost where kid = :new.kid;

update knowledge set price = temp\_totalcost\*1.2 where kid = :new.kid;

exception

when no\_data\_found then

:new.weight := 100;

update knowledge set totalcost = temp\_totalcost where kid = :new.kid;

update knowledge set price = temp\_totalcost\*1.2 where kid = :new.kid;

end;

--Sample insert

--insert into contributions(KID,NID,WEIGHT) values('K001','N0011',0); NOTE:KID and NID should be inputed in advance!!!!