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

MEMBERS:

Rui Cheng

Bowen Lui

Rahul Arora

Prateek Trivedi

Aditya Behal

Navin Mohan

Table of Contents

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.

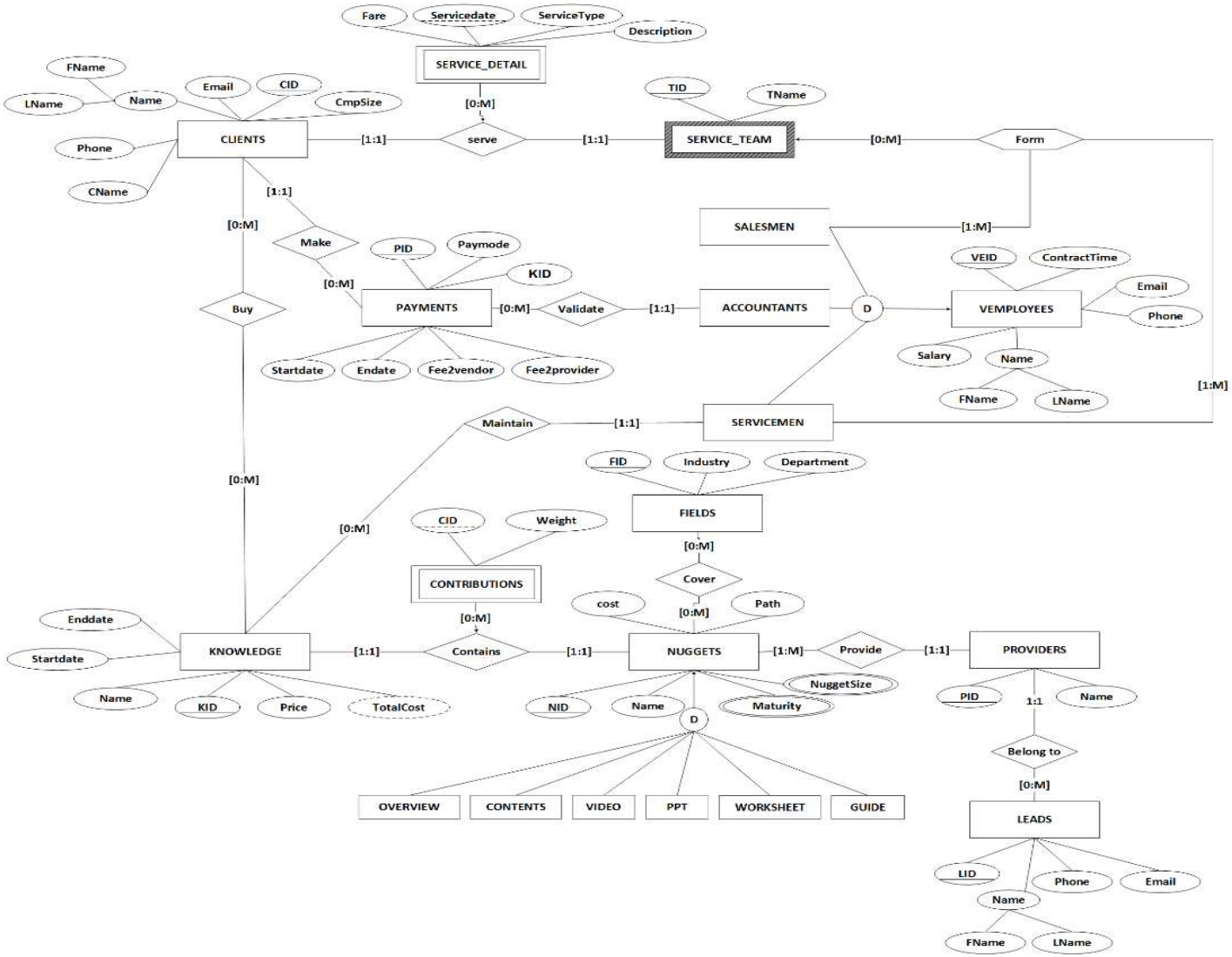
Each provider provides a knowledge nugget and they have a name and ID number associated with each of these providers. For each of the provider we also collect the engineer details who is the person that was responsible for signing the contract with Ryan. Although the vendor buys the knowledge from providers, 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

Schema Construct	Construct Description	Other Information
BELONG TO	Relationship that models Provider Engineers Belonging to Providers	
BUY	Relationship that models customer buying a knowledge	
CLIENTS	Entity to Model Client Information	
CNAME	Name Of Client	Should not be Null
EMAIL	Email of Client	
CID	Identifying number of Client	*Identifying Attribute*
SCALE	Maturity Level of client company	small/medium/large
CONSIST OF	Aggregation relationship that models Customer Service employee and technical service employees	
CONSTRAIN	Relationship that models how knowledge detail like duration constrains Knowledge	
CONTAINS	Relationship that models a knowledge contains many knowledge nuggets	
CONTRIBUTION	Weak entity that models contribution percentage of a particular knowledge nugget to knowledge	
CID	Partial Identifier for this entity	
WEIGHT VALUE	Percentage contribution	will be calculated through a stored procedure
COVER	Relationship that models covering of a particular field by a knowledge nugget	
CUSTOMER SERVICE TEAM	Aggregate entity to model temas that attend customers	
TID	Identifying number of Team	*Identifying Attribute*
SERVE_LEVEL	level of service provided	level-1/2/3
STANDARD_FARE	fare charged for a particular service provided	
FIELD	Entity to Model Field that a knowledge nugget is made to cater to	
FID	identifying number for the field	*Identifying Attribute*
FNAME	Name Of Field	Should not be Null
KNOWLEDGE	Entity to model knowledge piece that a client buys	
KID	Identifying number of Knowledge	*Identifying Attribute*
KNAME	name of knowledge piece	Should not be Null
TOTAL_FARE	total fare charged to client for knowledge provided	
KNOWLEDGE DETAIL	Entity to model details of particular knowledge's validity	
KDID	Identifying number for Knowledge Detail	*Identifying Attribute*
START_DATE	date of start of validity	
END_DATE	date of end of validity	can be derived
DURATION	duration that a knowledge sold is valid for	
KNOWLEDGE NUGGET	Entity to model atomic knowledge pieces that make up a particular knowledge	
KNID	Identifying number for knowledge nugget	*Identifying Attribute*
KNAME	name of knowledge nugget	Should not be Null
MATURE_LEVEL	Maturity Level of company that one nugget is meant for	cantake up numbers 1 through 5
COMPANY_SIZE	scale of the company that it is meant for	small/medium/large
PATH	url address for the nuggets	
MAINTAIN	Relationship that models maintenance of Knowledge by technical service employee	
MAKE	Relationship that models making of payment by a client	
PAYMENT	Entity used to model payment received from a client	
PID	Identifying number of a particular payment	*Identifying Attribute*
PAYMODE	mode of payment	card/cash
KID	knowledge ID that a particular payment is initiated for	Updated through a Trigger Event
START_DATE	Start date of payment	
END_DATE	End date of payment	
FEE2VENDOR	Part of payment that is meant for vendor	
FEE2PROVIDER	Part of payment that is meant for Provider	

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

PROVIDE	Relationship that models knowledge nuggets being provided by providers	
PROVIDER_ENGINEERS	Entity to model SPOC from providers	
EID	Identifying number of Provider engineer	*Identifying Attribute*
NAME	Name of Provider engineer	Should not be Null
FIRST_NAME	First name of provider engineer	
LAST_NAME	last name of provider engineer	
EMAIL	email of provider engineer	can take up multiple values
PHONE	phone number of provider engineer	can take up multiple values
PROVIDERS	Entity to model provider/ firm that provided knowledge piece	
PID	Identifying number of provider/firm	*Identifying Attribute*
PNAME	name of provider/firm	Should not be Null
SERVE	Relationship that models service rendered by customer service team of vendor to clients	
VALIDATE	Relationship that models validation of payment by accounting staff of vendor	
VENDOR EMPLOYEES	Entity to model employees in the vendor's company	
VEID	Identifying number of vendor employee	*Identifying Attribute*
CONTRACT_TIME	Duration of contract for a particular employee	
EMAIL	Email of employee	can take up multiple values
PHONE	phone number of vandor employee	can take up multiple values
NAME	name of vendor employee	Should not be Null
FIRSTNAME	first name of vendor employee	
LASTNAME	last name of vendor employee	
SALARY	salary that one employee is entitled to	
SUBCLASS OF KNOWLEDGE_NUGGET:-	Entities that model subclasses of knowledge nuggets	
CONTENT	entity for opening page/table of contents for a particular knowledge nugget	
LEARNING GUIDE	entity for storing guidance document for a particular knowledge nugget	
OVERVIEW	entity for storing summary of a knowledge nugget	
PPT	entity for storing ppt that may form a part of a knowledge nugget	
PAGES	Number of pages in a ppt	
VIDEO	entity for modelling video that may form a part of knowledge nugget	
LENGTH	length of a video	
WORKSHEET	entity to model worksheet that is a part of a nugget	
ROWCOLUMN	row and columns in a worksheet	
SUBCLASS OF VENDOR_EMPLOYEES:-	Entities that model subclasses of Vendor Employees	
ACCOUNTING STAFF	Entity to model Accounting staff	
CERTIFICATES	Certificates that an accountant holds	Can take up multiple values
CUSTOMER SERVICE EMPLOYEE	Entity to model customer service type vendor employee	
TECHNICAL SERVICE EMPLOYEE	entity to model technical service type of employee	

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

Schema Construct	Data Type	Constraint
BUY	Relation representing the relationship buy between CLIENTS and KNOWLEDGE	
CID	Varchar2(20)	Foreign Key references CUSTOMERS (Part of primary key)
KID	Varchar2(20)	Foreign Key references KNOWLEDGE (Part of primary key), updated with a TRIGGER on buy relationship
CONTRIBUTIONS	Weak Entity between KNOWLEDGE and NUGGETS	
CID	Varchar2(20)	Partial Identifier for CONTRIBUTIONS (Part of primary key)
KID	Varchar2(20)	Foreign Key references KNOWLEDGE (Part of primary key)
NID	Varchar2(20)	Foreign Key references NUGGETS (Part of primary key)
Weight	Number(2)	Should not be null, percentage contribution in whole knowledge
COVER	Relationship between nuggets and fields	
FID	Varchar2(20)	Foreign Key references FIELDS (Part of primary key)
NID	Varchar2(20)	Foreign Key references NUGGETS (Part of primary key)
CLIENTS		
CID	Varchar2(20)	Primary Key
Cname	Varchar2(50)	Should not be null
CmpSize	Varchar2(10)	Should not be null (Check in 'Small', 'Medium', 'Large')
Email	Varchar2(50)	Should not be null, each customer has only one email ID
Fname	Varchar2(20)	Should not be null
Lname	Varchar2(20)	Should not be null
Phone	Number(10)	Should not be null, each client has only one phone number
FIELDS	Entity to model the FIELDS that a knowledge caters to	
FID	Varchar2(20)	Primary Key
Industry	Varchar2(20)	Should not be null
Department	Varchar2(20)	Should not be null
FORM_ST_SALESMEN	Class grouped into service team	
TID	Varchar2(20)	Foreign key references SERVICE TEAM (Part of primary key)
VEID	Varchar2(20)	Foreign key references Salesmen(Part of primary key)
FORM_ST_SERVICEMEN	Class grouped into service team	
TID	Varchar2(20)	Foreign key references SERVICE TEAM (Part of primary key)
VEID	Varchar2(20)	Foreign key references SERVICEMEN(Part of primary key)
KNOWLEDGE	Entity to model KNOWLEDGE entity	
KID	Varchar2(20)	Primary key
Price	Number(10,2)	Should not be null
TotalCost	Number(10,2)	Should not be null
Name	Varchar2(30)	Should not be null
Startdate	DATE	Should not be null
Enddate	DATE	Should not be null
VEID	Varchar2(20)	Foreign key references VEMPLOYEE(VEID)
LEADS		
LID	Varchar2(20)	Primary Key
Fname	Varchar2(20)	Should not be null
Lname	Varchar2(20)	Should not be null
Phone	Number(10)	unique per lead
Email	Varchar2(50)	unique per lead
PID	Varchar2(20)	Foreign key references PROVIDERS(PID)

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

NUGGET_MATURITY	Entity to model multivalued attribute maturity of a nugget	
NID	Varchar2(20)	Foreign Key references NUGGETS (Part of primary key)
Maturity	Number(1)	Maturity level- CHECK between 1 and 5 (Part of primary key)
NUGGET_SIZE	Entity to model multivalued attribute size of a nugget	
NID	Varchar2(20)	Foreign Key references NUGGETS (Part of primary key)
Size	Number(1)	Size- CHECK between 1 and 5 (Part of primary key)
NUGGETS	Entity to model NUGGETS	
NID	Varchar2(20)	Primary Key
Name	Varchar2(20)	Should not be null
Path	Varchar2(50)	Should not be null
PID	Varchar2(20)	Foreign key references PROVIDERS(PID)
PAYMENTS	Entity to model payments by a customer	
PID	Varchar2(20)	Primary Key
Paymode	Varchar2(10)	CHECK in 'Cheque','Cash','Credit Card'
KID	Varchar2(20)	Foreign Key references KNOWLEDGE
Startdate	DATE	Should not be null
Enddate	DATE	Should not be null
Fee2vendor	Number(10,2)	Should not be null
Fee2provider	Number(10,2)	Should not be null
VEID	Varchar2(20)	Foreign key references ACCOUNTANTS(VEID)
CID	Varchar2(20)	Foreign Key references CUSTOMERS
PROVIDERS	Entity to model provider of nuggets	
PID	Varchar2(20)	Primary Key
Name	Varchar2(50)	Should not be null
SERVICE_DETAIL	Weak Entity between Customers and Service Team	
CID	Varchar2(20)	Foreign Key references CUSTOMERS (Part of primary key)
TID	Varchar2(20)	Foreign key references SERVICE_TEAM (Part of primary key)
Servicedate	DATE	Partial Identifier for SERVICE_DETAILS (Part of primary key)
Servicetype	Number(1)	CHECK in (1-5)
Fare	Number(10,2)	Should not be null
Desc	Varchar2(100)	Should not be null
SERVICE_TEAM	Entity to model service team that offers service to customers	
TID	Varchar2(20)	Primary Key
Tname	Varchar2(20)	Should not be null
VEMPLOYEES	Entity to model VENDOR EMPLOYEES	
VEID	Varchar2(20)	Primary Key
Contracttime	Number(2)	Should not be null CHECK <>10 (Years)
Email	Varchar2(50)	Should not be null, unique per employee
Phone	Number(10)	Should not be null, unique per employee
Salary	Number(10)	Should not be null
Fname	Varchar2(20)	Should not be null
Lname	Varchar2(20)	Should not be null

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

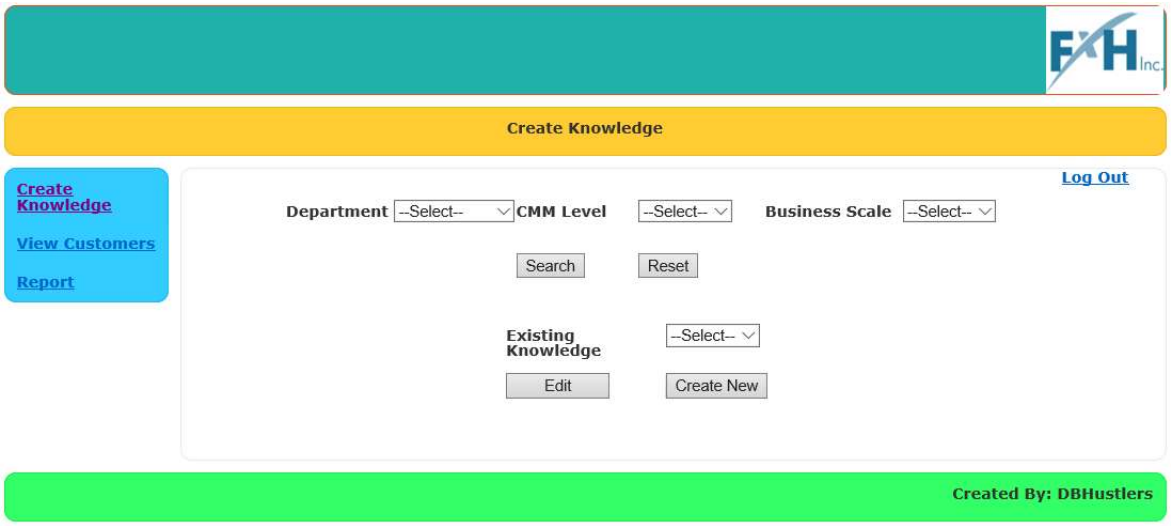
Website Address: 54.213.37.16

ADMIN LOGIN

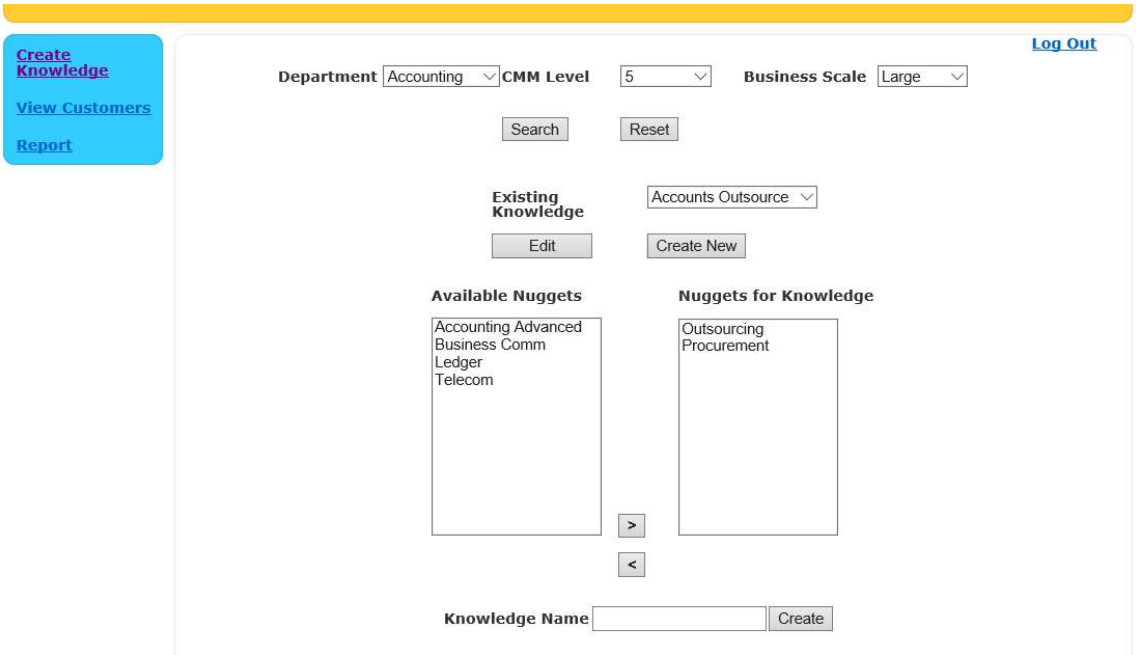
Username: admin

Password: abc123

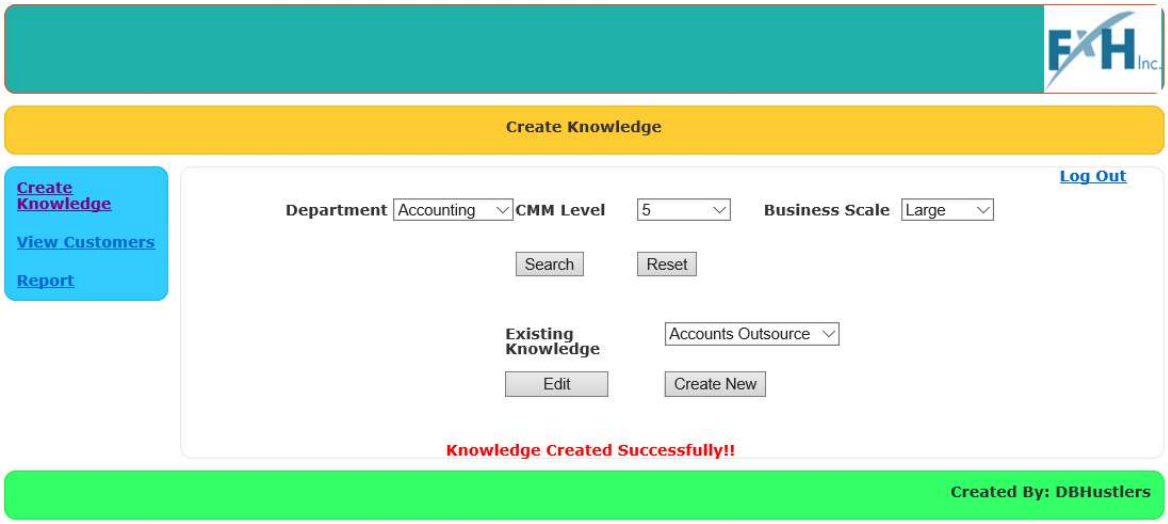
Create Knowledge Page:



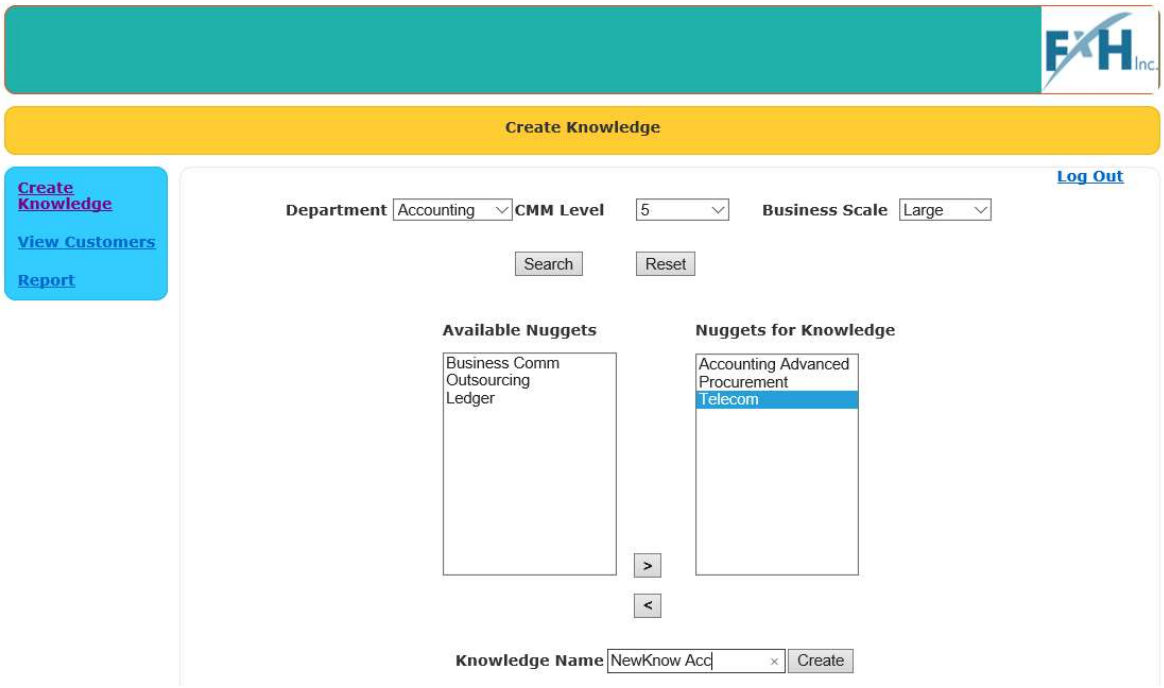
Select Department, CMM Level and Business Scale values. Depending on the selected values Existing Knowledge will be populated. If you want to create a knowledge from one of the existing knowledges, select a knowledge from the dropdown and click on edit.



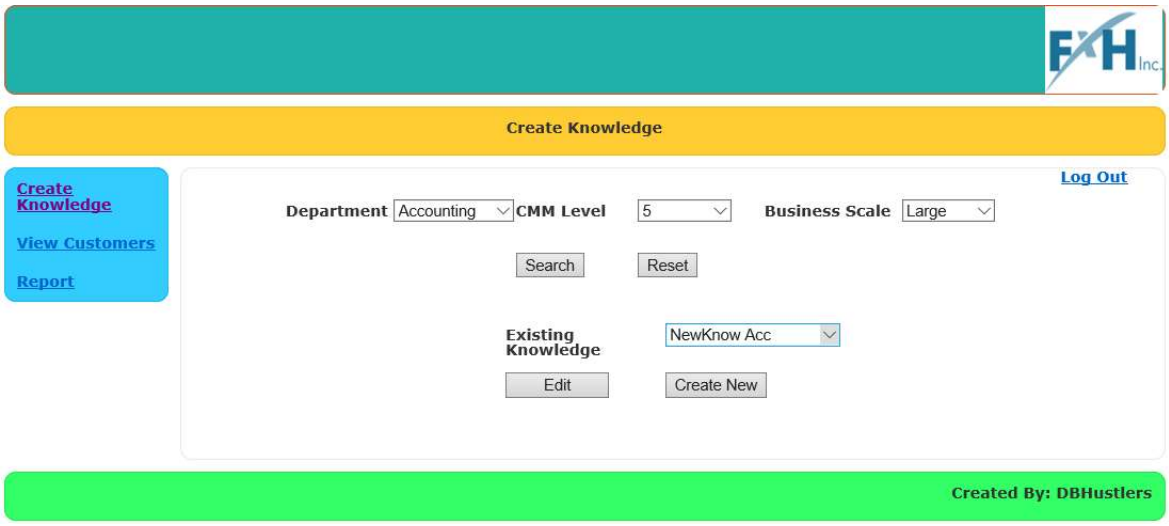
Add or delete nuggets from the listboxes by selecting a nugget and clicking on the arrow buttons. Give new knowledge name and click on Create button. Success message will be displayed.



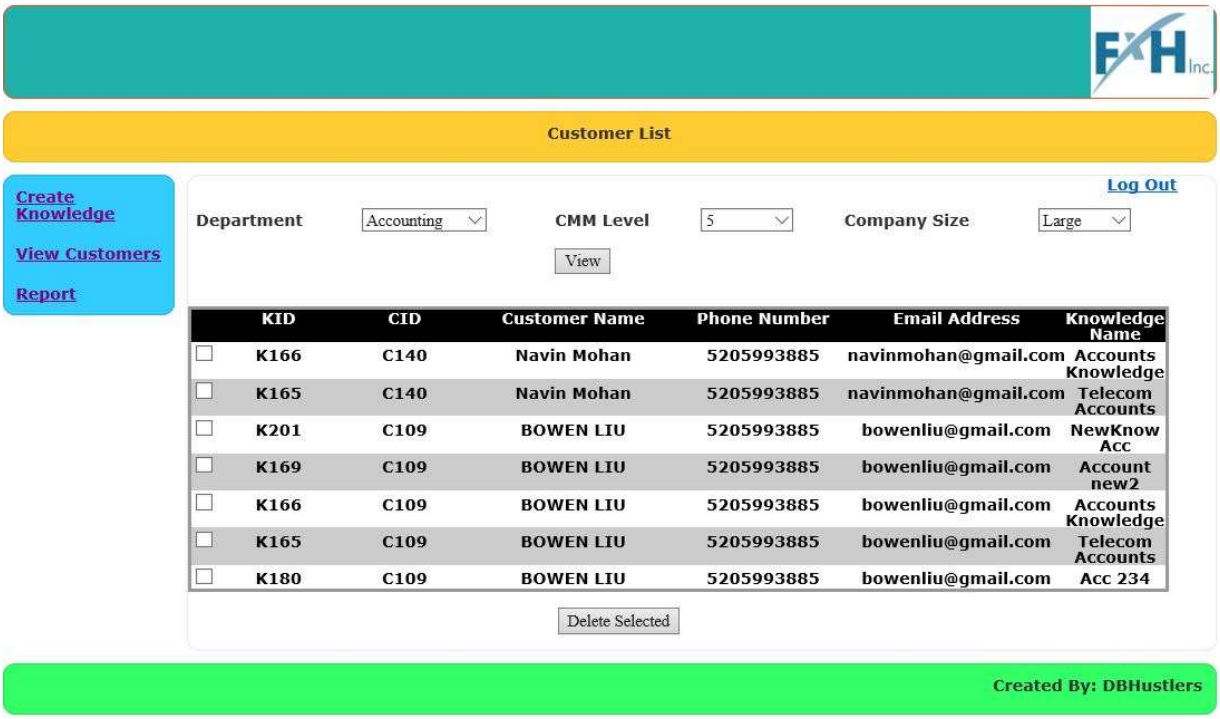
To create a new knowledge without using existing knowledges, click on Create New button.



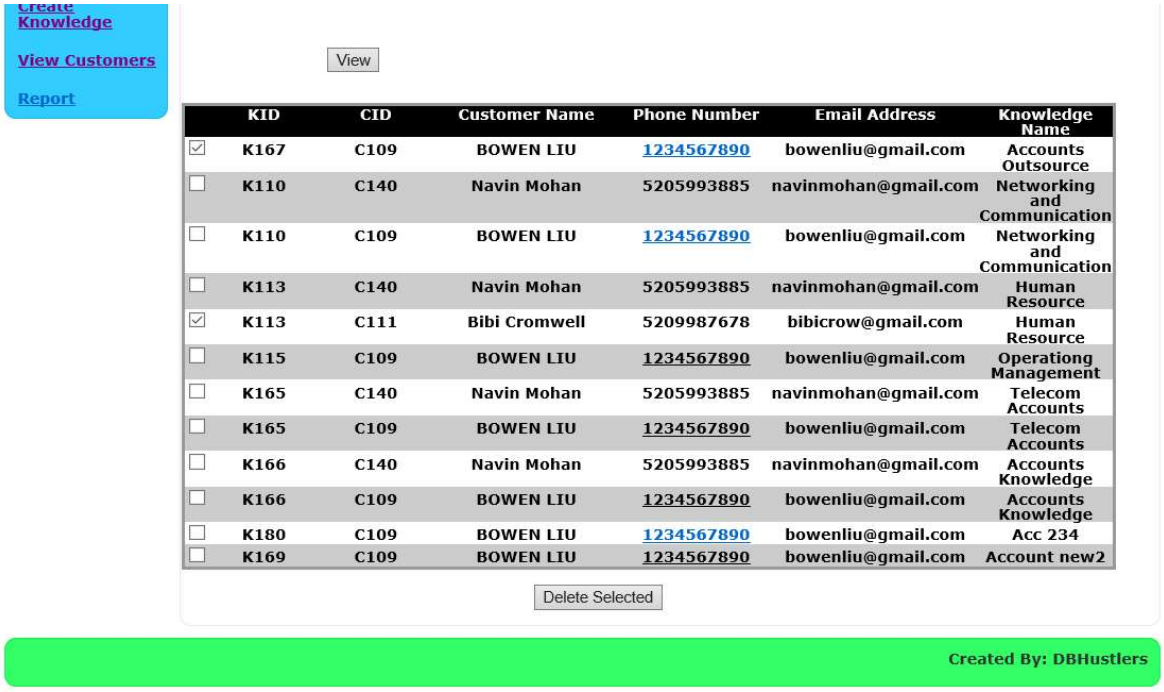
After selecting nuggets and giving a name to the knowledge, click on Create button. Knowledge will be created.



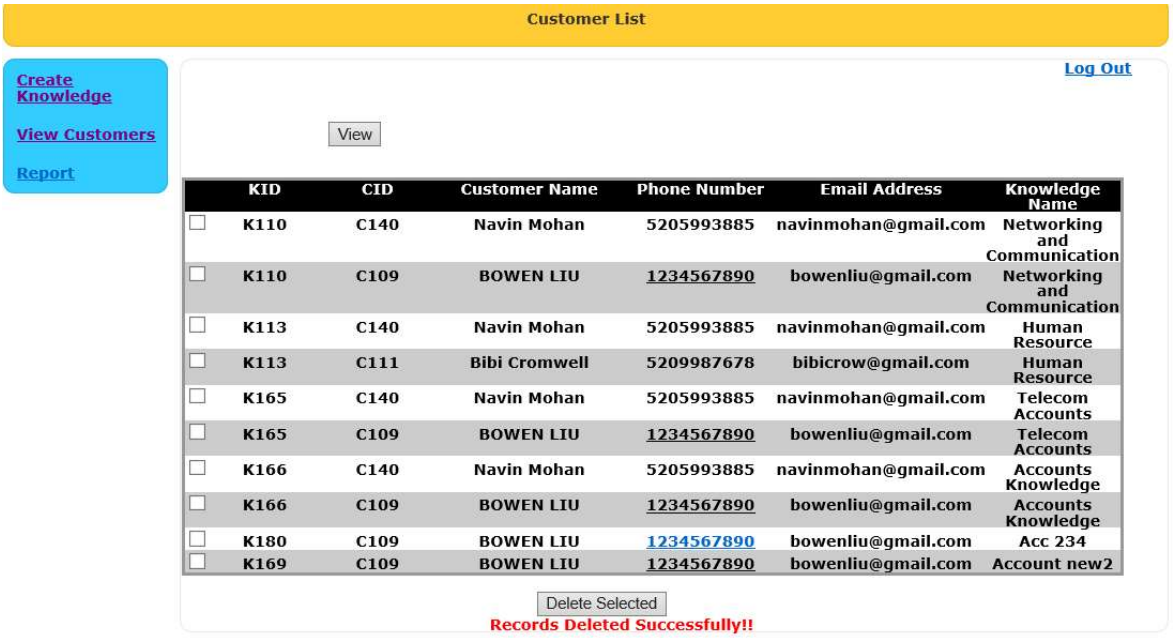
View Customers page: The admin would be able to filter the client list based on the 3 criterions that are displayed in the webpage below : Department, CMM Level and Company Size.



Select the checkboxes to delete knowledge of a particular user.



Click on delete Selected button



KID	CID	Customer Name	Phone Number	Email Address	Knowledge Name
K110	C140	Navin Mohan	5205993885	navinmohan@gmail.com	Networking and Communication
K110	C109	BOWEN LIU	1234567890	bowenliu@gmail.com	Networking and Communication
K113	C140	Navin Mohan	5205993885	navinmohan@gmail.com	Human Resource
K113	C111	Bibi Cromwell	5209987678	bibicrow@gmail.com	Human Resource
K165	C140	Navin Mohan	5205993885	navinmohan@gmail.com	Telecom Accounts
K165	C109	BOWEN LIU	1234567890	bowenliu@gmail.com	Telecom Accounts
K166	C140	Navin Mohan	5205993885	navinmohan@gmail.com	Accounts Knowledge
K166	C109	BOWEN LIU	1234567890	bowenliu@gmail.com	Accounts Knowledge
K180	C109	BOWEN LIU	1234567890	bowenliu@gmail.com	Acc 234
K169	C109	BOWEN LIU	1234567890	bowenliu@gmail.com	Account new2

Report Page:

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

Sales Report

Create Knowledge

View Customers

Report

Log Out

From Date 11/2/2014 To Date 12/7/2015

Generate Report

Customer Name	Payment Mode	Date of Purchase	Provider Fee	Vendor Fee	Knowledge Name	Provider Name
BOWEN LIU	Cash	12/7/2015 12:23:20 PM	81100	16220	Operationg Management	Provider2
BOWEN LIU	Cash	12/7/2015 2:43:30 PM	23100	4620	Account new2	Provider10
BOWEN LIU	Cash	12/7/2015 1:25:25 PM	22000	4400	Accounts Outsource	Provider10
BOWEN LIU	Cash	12/10/2015 5:26:55 PM	33000	6600	Acc 234	Provider10
BOWEN LIU	Cash	12/7/2015 12:22:42 PM	22000	4400	Telecom Accounts	Provider10
BOWEN LIU	Cash	12/7/2015 12:20:33 PM	33000	6600	Accounts Knowledge	Provider10
Zonia Tufts	Cash	12/5/2015 9:27:57 AM	1000	200	Networking and Communication	Provider3
Zonia Tufts	Cash	12/5/2015 9:27:57 AM	1000	200	Networking and Communication	Provider2
Zonia Tufts	Cash	12/5/2015 9:27:57 AM	1000	200	Networking and Communication	Provider1
Navin Mohan	Cash	12/7/2015 10:25:01 AM	2200	440	Human Resource	Provider2
Navin Mohan	Cash	12/7/2015 12:07:17 PM	22000	4400	Telecom Accounts	Provider10
Navin Mohan	Cash	12/7/2015 1:13:14 PM	33000	6600	Accounts Knowledge	Provider10
Navin Mohan	Cash	12/7/2015 10:22:59 AM	14200	2840	Networking and Communication	Provider3
Navin Mohan	Cash	12/7/2015 10:22:59 AM	14200	2840	Networking and Communication	Provider2
Navin Mohan	Cash	12/7/2015 10:22:59 AM	14200	2840	Networking and Communication	Provider1

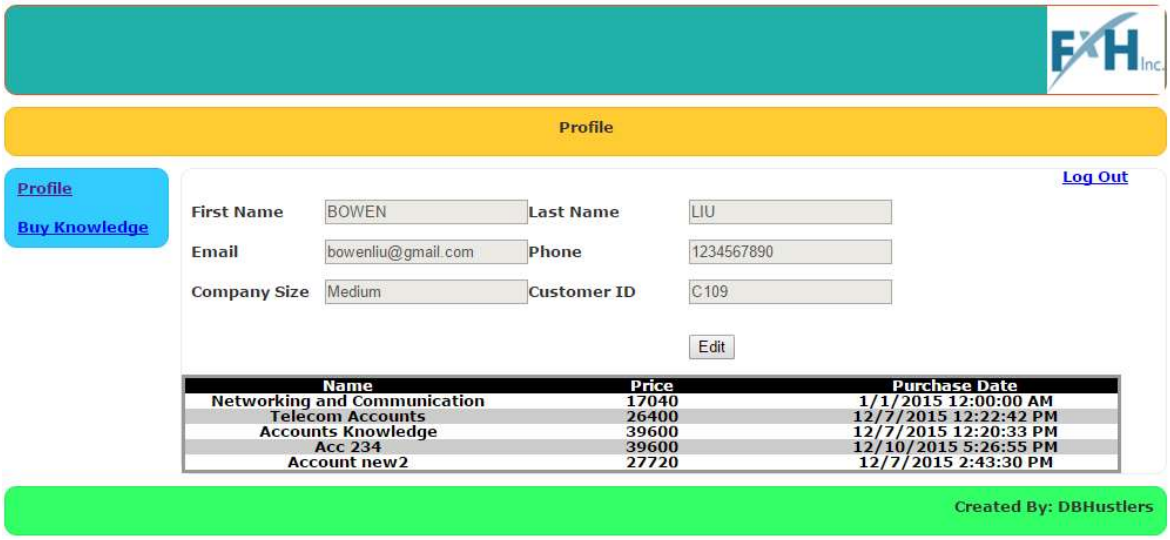
Customer Login:

Username: bliu

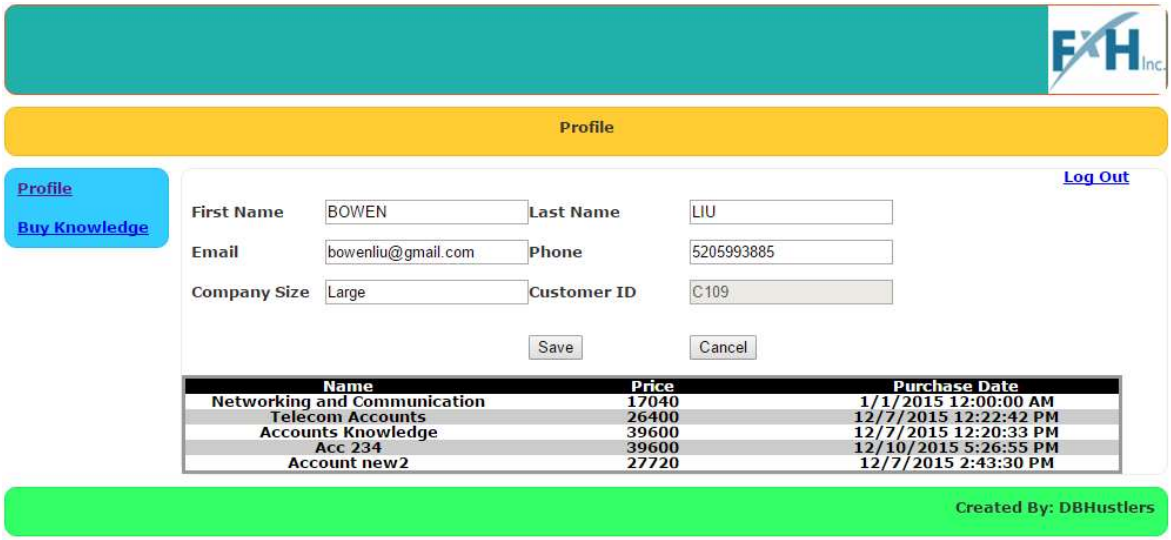
Password: abc123

Landing page after login is Customer Profile page:

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

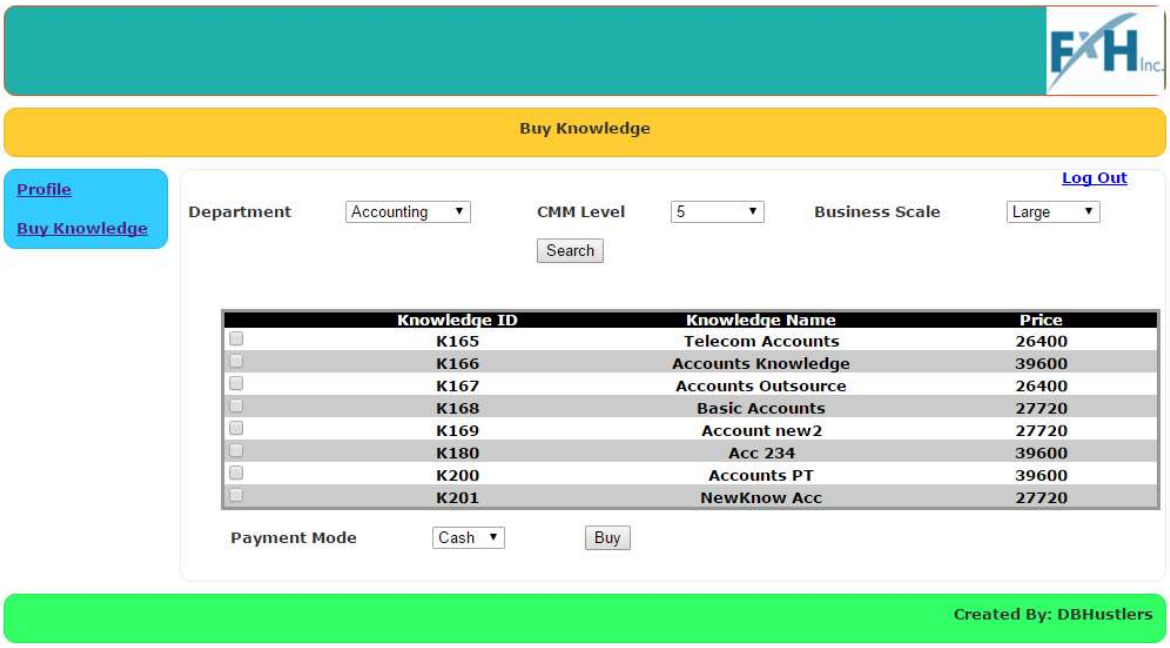


To change customer details click on Edit button. Enter the updated fields in the textbox and click on save button.



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.

Buy Knowledge

Profile

Buy Knowledge

Log Out

Department Accounting CMM Level 5 Business Scale Large

Search

Knowledge added successfully!!

Created By: DBHustlers

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

Phase I	Phase II	Phase III
• Approval of contract. • Hardware purchase.	• Installation of the database in their system. • Software installation including Oracle, SQLDeveloper and Visual Studio.	• Deploy the web application • Create the database to support the application (insert data into the database).

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), ServicedateDate, 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!!!!
```