

Customer Relation Management System with Project Management Tracker

Course Section: CS605.441.81
Spring, 2016

Chin-Ting Ko
April 2016

Database Design Project Document

TABLE OF CONTENTS

List Of Figures.....
Figure1: ERD Diagram

1. Introduction

- Customer Relationship Management (CRM) are very common database application to record customer contact and account information. Some particular industry such as hardware or software high tech company the product deliverable are mainly project based rather than simply buy and sell. A project management system is required to track, monitor, and record project status and action items follow up. Due to business requirement, here is one proposed database solution which developed to meet this business environment.

1.1. Scope and Purpose of Document conceptual design:

For this DB project, "CRM with project tracker", the requirement is basically traditional CRM system plus project management tool. The Entities include Account (Customer), Customer Contact, Order, Project, Issue Tracker, Product, Employee (Sales, PM etc.)

Logical design:

For each entity attribute, most data is character type except for PK (id), which is integer type.

For relationships, typical are below:

Account to Project (0/1 to Many)

Account to Contact (1 to Many)

Account to Order (0/1 to Many)

Project to Employee (Many to Many), uses WorksOn as intersection Entity.

Project to Product (Many to Many), uses Requirement as intersection Entity.

Implementation database:

Used MySQL with MySQL workbench as database implementation.

1.2. Project Objective -

The project objective is to provide a database system implementation to combine traditional CRM and project management tracker for business users, and it includes different users such as sales rep, sales manager, product manager, direct customer, and external ecosystem partners.

2. System Requirements

- 2.1. Hardware Requirements - the minimum hardware requirement is same as modern RDBMS application requirement. This program doesn't need must have hardware requirement, however we assume the user's computer should with 2 CPU cores 1.2GHz, 1GB RAM, 1GB storage, and I/O device. Recommend hard ware requirements are 4 CPU cores 2.0 GHz, 4GB RAM, 8GB storage space, or more.
- 2.2. Software Requirements - modern 32bit operating system such as windows 7.0 or later. Internet connection and IE explorer 9.0 or later is suggested if need to access data from client side web browser.
- 2.3. Functional Requirements -
This DB can support traditional CRM system with Project Management tracker.
CRM: able to query customer contact information, customer account information, order status etc.
PM tracker: able to track project status, issue tracker, and product requirement.
- 2.4. Database Requirements -
Used MySQL with MySQL workbench 6.3 for OS X version as database implementation.

3. Database Design Description

- 3.1. Design Rationale -The main database starts with CRM, which is customer name and customer contact. The projects are from customer, and it might not be project available, so I put non-identifying relationships. Issue tracker, order, competitors are also weak entities and non-identifying relationship with project/account.

3.2. E/R Model

3.2.1. Entities

Contact: Customer contact information

- idContact: identifier for Contact, PK
- LastName:
- FirstName:
- Title: Job title of contact
- Tel:
- Email:

Account: Customer corporate information

- idAccount: identifier for customer corporate information, PK

- Acct Name: Customer corporate name
- State:
- City:
- Address:

Order: Sales Order

- idOrder: Sales order#, PK
- OrderTotal: Order total amount
- Shipdate: order delivery date

Project : Ongoing project with customer

- idProject: identifier for project, PK
- Project Name: project code name
- EAU: estimated annual units, project scope
- StartDate: project start date
- EndDate: project end date

Competitor: Project bid competitor

- idCompetitor: identified for competitor, PK
- ProductName: competitor product for this project
- Price: competitor price for this project

IssueTracker: project issue tracker

- idIssue: identifier for this issue, PK
- IssueName: issue content
- IssueOwner: issue owner, can be customer or sales person etc.
- DueDate: issue due date

Product: particular product for project or order

- idProduct: identifier for product, can be part number, PK
- Model Name: product series name

OrderProduct: intersection entity

- Order_idOrder: PK
- Product_idProject: PK
- Product: product name
- Qty: product quantity
- Price: product price

Requirement: project requirement, intersection entity

- Product_idProject: PK
- Project_idProject: PK
- Price: project price

Employee: project stakeholder, can be sales, Product manger, or general manager etc.

- idEmployee: identifier for employee, PK
- LastName:
- FirstName:
- Department: sales department, pm department, factory etc.
- Department_idDepartment: PK

3.2.2. Relationships -

Account to Contact: one to many relationships
(Each account must have one contact information)

Account to Order: zero/one to many relationships
(One customer can without order, or multiple orders)

Account to Project: zero/one to many relationships
(One customer can without project, or multiple projects)

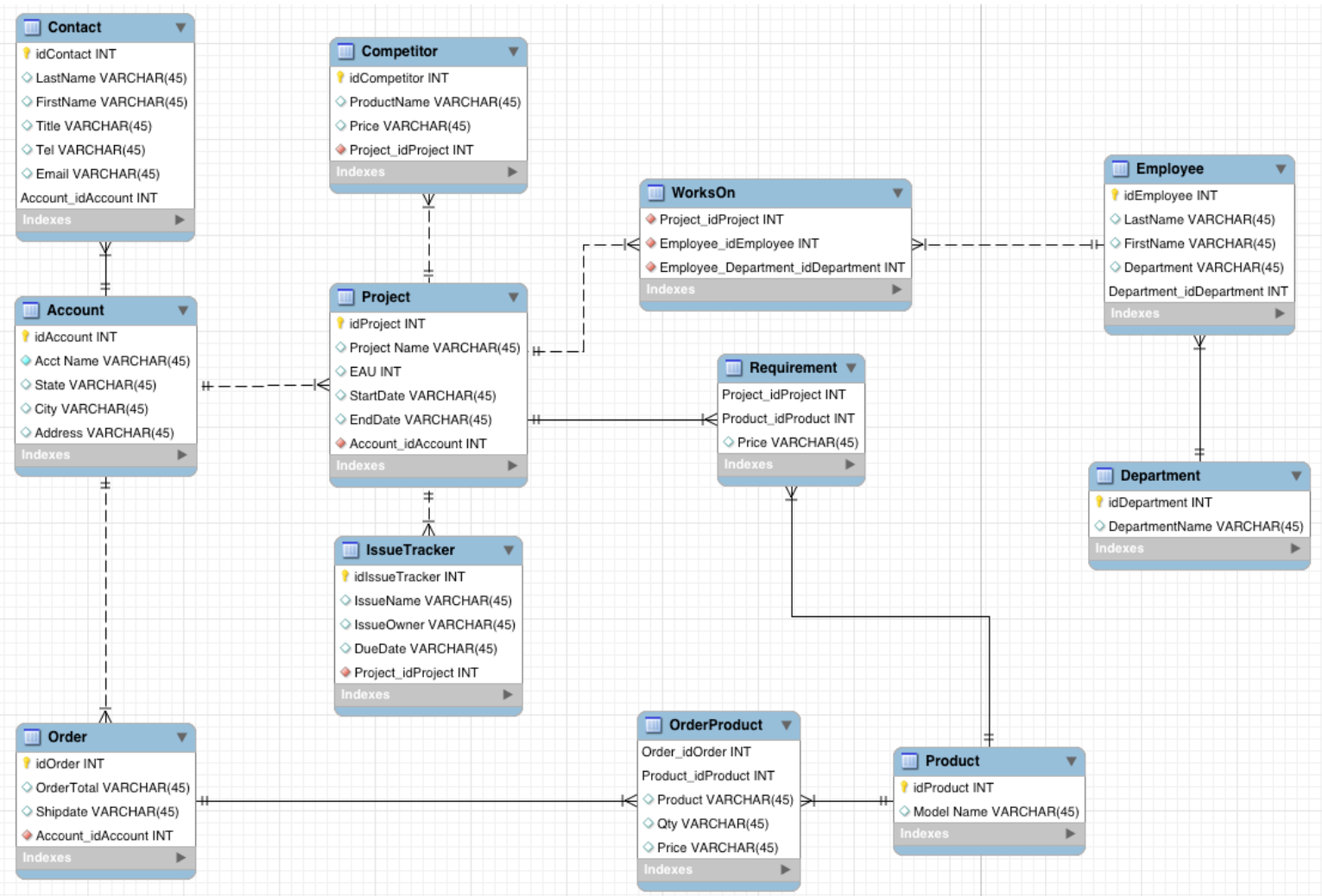
Project to competitor: zero/one to many relationships
(One project bid can without competitor, or multiple competitors)

Project to IssueTracker: zero/one to many relationships
(One project bid can without issues, or multiple issues)

Department to Employee: one to many relationships
(Each employee should with only one department, one department can with multiple employee)

3.2.3. E/R Diagram -

Figure 1



3.3. Relational Model

1. Data Dictionary

Account

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
idAccount	Primary key	INT			y	
Act Name		varchar	45		y	

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
State		varchar	45		N	
City		varchar	45		N	
Address		varchar	45		N	

Contact

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
idContact	Primary key	INT			Y	
LastName		varchar	45		N	
FirstName		varchar	45		N	
Title		varchar	45		N	
Tel		varchar	45		N	
Email		varchar	45		N	
Account_id Account	FK	INT			Y	

Order

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
idOrder	Primary key	INT			Y	
OrderTotal		varchar	45		N	
Shipdate		datetime			N	
Account_id Account	FK	INT			Y	

Project

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
idProject	Primary key	INT			Y	
ProjectName		varchar	45		N	

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
EAU		INT			N	
StartDate		datetime			N	
EndDate		datetime			N	
Account_id Account	FK	INT			Y	

Competitor

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
idCompetitor	Primary key	INT			Y	
ProductName		varchar	45		N	
Price		INT			N	
Project_idP roject	FK	INT			Y	

IssueTracker

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
idIssueTracker	Primary key	INT			Y	
IssueName		varchar	45		N	
IssueOwner		varchar	45		Y	
DueDate		datetime			N	
Project_idP roject	FK	INT			Y	

WorksOn

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
Project_idProject	FK	INT			y	
Employee_idEmployee	FK	INT			y	
Employee_Department	FK	INT			y	

Requirement

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
Project_idProject	FK	INT			y	
Product_idProduct	FK	INT			y	
Price		varchar	45		N	

OrderProduct

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
Order_idOrder	FK	INT			y	
Product_idProduct	FK	INT			y	
Product		varchar	45		N	
Qty		INT			N	
Price		INT			N	

Employee

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
id_Employee	PK	INT			y	

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
Department_idDepartment	FK	INT			Y	
LastName		varchar	45		N	
FirstName		varchar	45		N	
Department		varchar	45		N	
Bonus		INT			N	

Department

Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values
id_Empolyee	PK	INT			Y	
Department Name		varchar	45		N	

3.3.2. Integrity Rules -

Mandatory:

each Primary Key is not null. It's to ensure each insert dat must include an identifier.

Data formatting:

Most of data are string type, but use date time type for column such as start date, due date, end date etc.

Valid values:

As above, most constrain are on date time format.

Referential integrity:

I defined foreign key constraint between tables to maintain consistency among tuples of the two relations. Such as Foreign key Account_idAccount in table ORDER, PROJECT. Foreign Key Project_idProject in table COMPETITOR and ISSUETRACKER etc.

3.3.3. Operational Rules - As the topic, the main operations are on Projects and Accounts actives

Retrieve Project Status:

For each sales rep, can only retrieve each person's projects

For sales manager and PM, there is no such constraint.

Create Issue tracker:

each Issue must with corresponding project

each Issue must with issue owner

Create new Project:

each project must with requirement and corresponding product

Create new Account:

each account must with customer contact information

Enter Order:

each order must with corresponding product

3.3.4. Operations -

For example a sales manager or product manager, would like to retrieve "Project/account EAU is more than 1000 with open issue"

select table PROJECT where EAU is more than 1000

select table ACCOUNT where corresponding the project

select table EMPLOYEE where which employee is working on this project

select table ISSUETRACKER where if any open issues

create a VIEW for sales manager include data above

3.4. Security - This project security feature is mainly on access control, which is username/password management. Each user set different authority such as admin/manager/regular employee etc. to control data access and data update authority.

3.5. Database Backup and Recovery - This project doesn't implement backup and recovery mechanism. In the long run, deferred update and immediate update might be options for future recovery. Such as recovery algorithm NO-UNDO/REDO, or UNDO/REDO, UNDO/NO-REDO etc. For physical database storage, RAID or distributed database system might be another solution to enhance backup and recovery feature.

3.6. Using Database Design or CASE Tool -

I used MySQL workbench as main CASE tool, it combines E/R diagram, database mapping, create a database schema, create database table, define PK, FK constraint, user query etc. I consider several tool such as visual paradigm and ERWIN, but later on decided to use MySQL workbench. The reason is visual paradigm is mainly on ER diagram but lack of database management tool features, and ERWIN is good for windows system but not for MAC OS X.

3.7. Other Possible E/R Relationships - Not consider at this time, but might be other suitable E/R relationships for future consideration.

4. Implementation Description -

4.1. Data Dictionary -

Account

Field	Type	Null	Key	Default	Extra
idAccount	int(11)	NO	PRI	NULL	
Acct Name	varchar(45)	NO		NULL	
State	varchar(45)	YES		NULL	
City	varchar(45)	YES		NULL	
Address	varchar(45)	YES		NULL	

Competitor

Field	Type	Null	Key	Default	Extra
idCompetitor	int(11)	NO	PRI	NULL	
ProductName	varchar(45)	YES		NULL	
Price	int(11)	YES		NULL	
Project_idProject	int(11)	NO	MUL	NULL	

Contact

Field	Type	Null	Key	Default	Extra
idContact	int(11)	NO	PRI	NULL	
LastName	varchar(45)	YES		NULL	
FirstName	varchar(45)	YES		NULL	
Title	varchar(45)	YES		NULL	
Tel	varchar(45)	YES		NULL	
Email	varchar(45)	YES		NULL	
Account_idAccount	int(11)	NO	PRI	NULL	

Department

Field	Type	Null	Key	Default	Extra
idDepartment	int(11)	NO	PRI	NULL	
DepartmentName	varchar(45)	YES		NULL	

Employee

Field	Type	Null	Key	Default	Extra
idEmployee	int(11)	NO	PRI	NULL	
LastName	varchar(45)	YES		NULL	
FirstName	varchar(45)	YES		NULL	
Department	varchar(45)	YES		NULL	
Department_idDepartment	int(11)	NO	PRI	NULL	
Bonus	int(11)	YES		NULL	

IssueTracker

Field	Type	Null	Key	Default	Extra
idIssueTracker	int(11)	NO	PRI	NULL	
IssueName	varchar(45)	YES		NULL	
IssueOwner	varchar(45)	NO		NULL	
DueDate	datetime	YES		NULL	
Project_idProject	int(11)	NO	MUL	NULL	

Field	Type	Null	Key	Default	Extra
idOrder	int(11)	NO	PRI	NULL	
OrderTotal	varchar(45)	YES		NULL	
Shipdate	varchar(45)	YES		NULL	
Account_idAccount	int(11)	NO	MUL	NULL	

OrderProduct

Field	Type	Null	Key	Default	Extra
Order_idOrder	int(11)	NO	PRI	NULL	
Product_idProduct	int(11)	NO	PRI	NULL	
Product	varchar(45)	YES		NULL	
Qty	int(11)	YES		NULL	
Price	int(11)	YES		NULL	

Product

Field	Type	Null	Key	Default	Extra
idProduct	int(11)	NO	PRI	NULL	
Model Name	varchar(45)	YES		NULL	

Project

Field	Type	Null	Key	Default	Extra
idProject	int(11)	NO	PRI	NULL	
Project Name	varchar(45)	YES		NULL	
EAU	int(11)	YES		NULL	
StartDate	datetime	YES		NULL	
EndDate	datetime	YES		NULL	
Account_idAccount	int(11)	NO	MUL	NULL	

Requirement

Field	Type	Null	Key	Default	Extra
Project_idProject	int(11)	NO	PRI	NULL	
Product_idProduct	int(11)	NO	PRI	NULL	
Price	int(11)	YES		NULL	

WorkOn

Field	Type	Null	Key	Default	Extra
Project_idProject	int(11)	NO	MUL	NULL	
Employee_idEmployee	int(11)	NO	MUL	NULL	
Employee_Department_idDepartment	int(11)	NO		NULL	

4.2. Advanced Features -

Trigger:

This DB project only designed with trigger. When creating a project, or project status move from "initial" to "design win" or "mass production", the bonus column from employee automatically triggered to add certain amount as bonus. Others similar business rule apply to trigger such as order price amount reach to retain revenue, trigger the bonus update from employee table.

Views: (restrict access)

This DB project set roles authority to access each table. For example, west coast sales person can only access west coast customer contact information, and order status. Project managers can access each account contact information and project details, but not on each employees bonus column etc.

Views: (Presenting tables to users in various forms)






Project/Sales review View: provide general manager or sales director a new view combine account table and project table together for business review and discussion

4.3. Queries -

CRM related:

- Customer contact information for particular Account-

```
SELECT * FROM mydb.Contact where Account_idAccount=5;
```

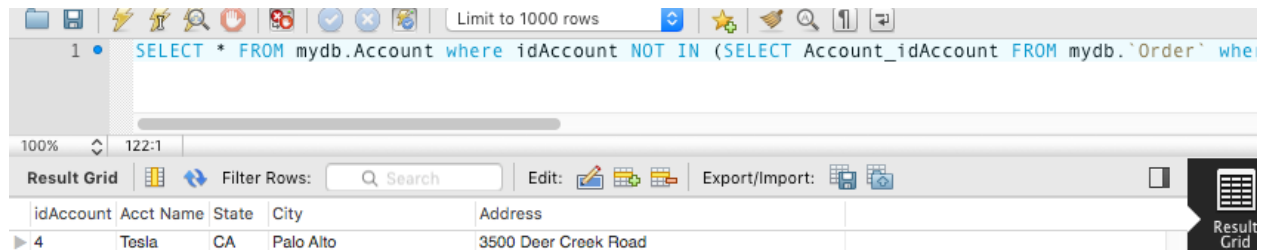
Result Grid						
Filter Rows: <input type="text" value="Search"/>						
Edit:   						
Export/Import:  						
idContact	LastName	FirstName	Title	Tel	Email	Account_idAccount
8	H	Hayden	Procurement Dr.	9998887777	HaydenH@yahoo.com	5
9	I	Ilionis	Egnineer	8887776666	Ilionil@yahoo.com	5
10	J	Jonh	BDM	6665554444	JohnJ@yahoo.com	5

- Sales history for particular Account-

```
SELECT * FROM mydb.`Order` where Account_idAccount=2;
```

Result Grid			
Filter Rows: <input type="text" value="Search"/>			
	idOrder	OrderTotal	Shipdate
▶	3	3000	03/03/2016
	4	4000	04/04/2016

- List of inactive Account-
`SELECT * FROM mydb.Account where idAccount NOT IN (SELECT Account_idAccount FROM mydb.`Order` where idOrder is not null);`

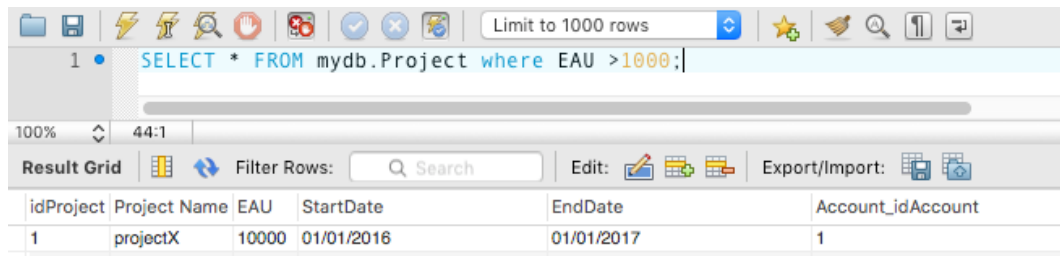


idAccount	Acct Name	State	City	Address
4	Tesla	CA	Palo Alto	3500 Deer Creek Road

PM related:

- Ongoing project by EAU more than X

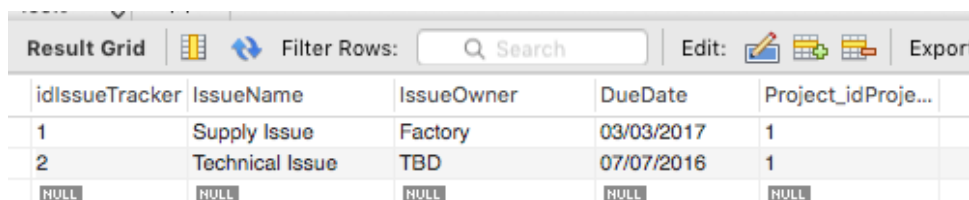
`SELECT * FROM mydb.Project where EAU >1000;`



idProject	Project Name	EAU	StartDate	EndDate	Account_idAccount
1	projectX	10000	01/01/2016	01/01/2017	1

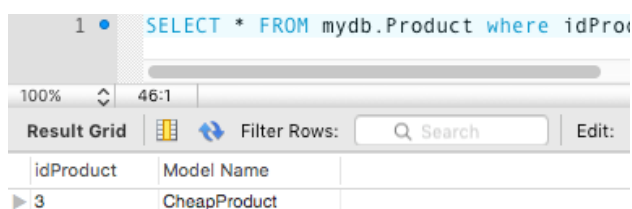
- Open Issue

`SELECT * FROM mydb.IssueTracker where Project_idProject=1;`

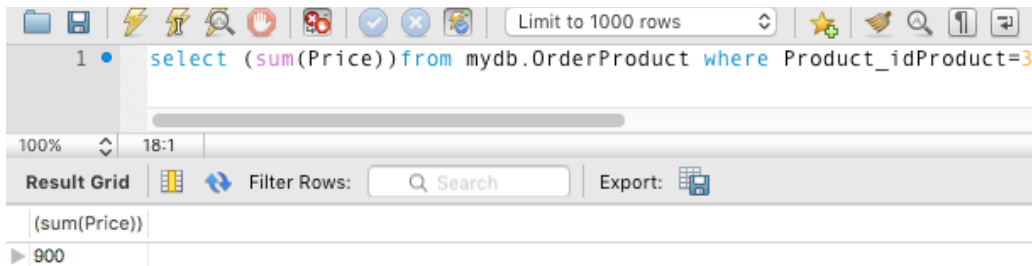


idIssueTracker	IssueName	IssueOwner	DueDate	Project_idProject
1	Supply Issue	Factory	03/03/2017	1
2	Technical Issue	TBD	07/07/2016	1

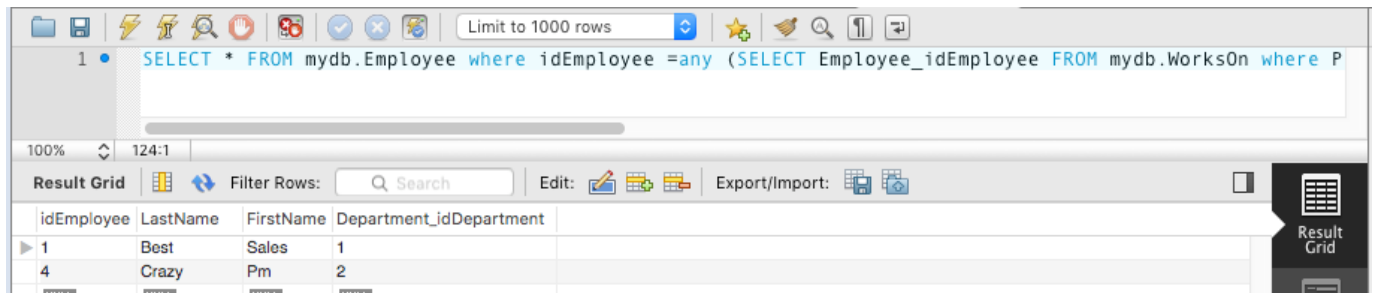
- See Product Revenue



idProduct	Model Name
3	CheapProduct



- List of employee working on project X
`SELECT * FROM mydb.Employee where idEmployee =any (SELECT Employee_idEmployee FROM mydb.WorksOn where Project_idProject=1);`



5. CRUD Matrix -

5.1. List of Entity Types

- E1: Account
- E2: Competitor
- E3: Contact
- E4: Department
- E5: Employee
- E6: IssueTracker
- E7: Order
- E8: OrderProduct
- E9: Product
- E10: Project
- E11: Requirement
- E12: WorksOn

5.2. List of Functions

- F1: Customer contact information for particular Account-
- F2: Open order for particular Account-
- F3: Sales history for particular Account-

F4: List of inactive Account-
 F5: Ongoing project by EAU more than X
 F6: Open Issue
 F7: Best Selling Product
 F8: List of employee working on project X
 F9: create/update/delete a new project
 F10: competitor on particular project

5.2. CRUD

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12
F1	R		R									
F2			R				R					
F3			R				R					
F4			R				R			R		
F5			R							R		
F6			R			R				R		
F7			R				R		R			
F8				R	R					R		R
F9			R						R	CUD	R	
F10		R								R		

6. Concluding Remarks - At end of the semester, I am glad have this final project assignment to help understand the lecture content more thoroughly. If without this project, I probably skim some of the content and might not have impression on all of the database design knowledge. To implement a RDMS project and hands-on experience help me practical knowledge on DB design. However, this project documentation takes much more time than expected. I am thinking it might be more suitable for a 2~3 people team project, to have more accurate and thorough quality report. Some of advanced feature such as security, backup recovery, will enhance for future work.

Appendices –

A. DDL, INSERT, SELECT Statements –

```
CREATE TABLE `mydb`.`Account` (
  `idAccount` INT NOT NULL,
  `Account Name` VARCHAR(45) NULL,
  `State` VARCHAR(45) NULL,
  `City` VARCHAR(45) NULL,
  `Address` VARCHAR(45) NULL,
  PRIMARY KEY (`idAccount`));
```

```
INSERT INTO `mydb`.`Account` (`idAccount`, `Acct Name`, `State`, `City`, `Address`) VALUES ('1',
'FB', 'CA', 'MenloPark', '1 Hacker Way');
```

```
INSERT INTO `mydb`.`Account` (`idAccount`, `Acct Name`, `State`, `City`, `Address`) VALUES ('2',
'Google', 'CA', 'Mountain View', '1600 Amphitheatre Pkwy');
```

```
INSERT INTO `mydb`.`Account` (`idAccount`, `Acct Name`, `State`, `City`, `Address`) VALUES ('3',
'Apple', 'CA', 'Cupertino', '1 Infinite Loop');
```

```
INSERT INTO `mydb`.`Account` (`idAccount`, `Acct Name`, `State`, `City`, `Address`) VALUES ('4',
'Tesla', 'CA', 'Palo Alto', '3500 Deer Creek Road');
```

```
INSERT INTO `mydb`.`Account` (`idAccount`, `Acct Name`, `State`, `City`, `Address`) VALUES ('5',
'Yahoo', 'CA', 'Sunnyvale', '701 1st Ave');
```

```
INSERT INTO `mydb`.`Contact` (`idContact`, `LastName`, `FirstName`, `Title`, `Tel`, `Email`,
`Account_idAccount`) VALUES ('1', 'A', 'Abby', 'PM', '2234445555', 'AbbyA@fb.com', '1');
```

```
INSERT INTO `mydb`.`Contact` (`idContact`, `LastName`, `FirstName`, `Title`, `Tel`, `Email`,
`Account_idAccount`) VALUES ('2', 'B', 'Bob', 'Sales', '1112223333', 'BobB@fb.com', '1');
```

```
INSERT INTO `mydb`.`Contact` (`idContact`, `LastName`, `FirstName`, `Title`, `Tel`, `Email`,
`Account_idAccount`) VALUES ('3', 'C', 'Cathie', 'Procurement', '3334445555', 'CatheiC@google.com', '2');
```

```
INSERT INTO `mydb`.`Contact` (`idContact`, `LastName`, `FirstName`, `Title`, `Tel`, `Email`,
`Account_idAccount`) VALUES ('4', 'D', 'Doglas', 'Engineer', '4445556666', 'DoglasD@google.com', '2');
```

```
INSERT INTO `mydb`.`Contact` (`idContact`, `LastName`, `FirstName`, `Title`, `Tel`, `Email`,
`Account_idAccount`) VALUES ('5', 'E', 'Edward', 'Sales', '5556667777', 'EdwardE@apple.com', '3');
```

```
INSERT INTO `mydb`.`Contact` (`idContact`, `LastName`, `FirstName`, `Title`, `Tel`, `Email`,
`Account_idAccount`) VALUES ('6', 'F', 'Fisher', 'Engineer', '6667778888', 'FisherF@apple.com', '3');
```

```
INSERT INTO `mydb`.`Contact` (`idContact`, `LastName`, `FirstName`, `Title`, `Tel`, `Email`,
`Account_idAccount`) VALUES ('7', 'G', 'Gladia', 'GM', '7778889999', 'GladiaG@tesla.com', '4');
```

```
INSERT INTO `mydb`.`Contact` (`idContact`, `LastName`, `FirstName`, `Title`, `Tel`, `Email`,
`Account_idAccount`) VALUES ('8', 'H', 'Hayden', 'Procurement Dr.', '9998887777', 'HaydenH@yahoo.com',
'5');
```

```
INSERT INTO `mydb`.`Contact` (`idContact`, `LastName`, `FirstName`, `Title`, `Tel`, `Email`,
`Account_idAccount`) VALUES ('9', 'I', 'Ilionis', 'Egnineer', '8887776666', 'IllioniI@yahoo.com', '5');
```

```
INSERT INTO `mydb`.`Contact` (`idContact`, `LastName`, `FirstName`, `Title`, `Tel`, `Email`,
`Account_idAccount`) VALUES ('10', 'J', 'Jonh', 'BDM', '6665554444', 'JohnJ@yahoo.com', '5');
```

```
INSERT INTO `mydb`.`Order` (`idOrder`, `OrderTotal`, `Shipdate`, `Account_idAccount`) VALUES
('0001', '1000', '01/01/2016', '1');
```

```
INSERT INTO `mydb`.`Order` (`idOrder`, `OrderTotal`, `Shipdate`, `Account_idAccount`) VALUES ('0002', '2000', '02/02/2016', '1');
INSERT INTO `mydb`.`Order` (`idOrder`, `OrderTotal`, `Shipdate`, `Account_idAccount`) VALUES ('0003', '3000', '03/03/2016', '2');
INSERT INTO `mydb`.`Order` (`idOrder`, `OrderTotal`, `Shipdate`, `Account_idAccount`) VALUES ('0004', '4000', '04/04/2016', '2');
INSERT INTO `mydb`.`Order` (`idOrder`, `OrderTotal`, `Shipdate`, `Account_idAccount`) VALUES ('0005', '500', '05/05/2016', '3');
INSERT INTO `mydb`.`Order` (`idOrder`, `OrderTotal`, `Shipdate`, `Account_idAccount`) VALUES ('0006', '600', '06/06/2015', '4');
INSERT INTO `mydb`.`Order` (`idOrder`, `OrderTotal`, `Shipdate`, `Account_idAccount`) VALUES ('0007', '700', '07/07/2015', '5');
INSERT INTO `mydb`.`Order` (`idOrder`, `OrderTotal`, `Shipdate`, `Account_idAccount`) VALUES ('0008', '800', '08/08/2014', '5');
```

```
INSERT INTO `mydb`.`Project` (`idProject`, `Project Name`, `EAU`, `StartDate`, `EndDate`, `Account_idAccount`) VALUES ('1', 'projectX', '10000', '01/01/2014', '01/01/2017', '1');
INSERT INTO `mydb`.`Project` (`idProject`, `Project Name`, `EAU`, `StartDate`, `EndDate`, `Account_idAccount`) VALUES ('2', 'projectZ', '500', '01/01/2015', '01/01/2016', '2');
```

```
INSERT INTO `mydb`.`Competitor` (`idCompetitor`, `ProductName`, `Price`, `Project_idProject`) VALUES ('1', 'CoolProduct', '100', '1');
INSERT INTO `mydb`.`Competitor` (`idCompetitor`, `ProductName`, `Price`, `Project_idProject`) VALUES ('2', 'FantasyProduct', '200', '1');
```

```
INSERT INTO `mydb`.`Department` (`idDepartment`, `DepartmentName`) VALUES ('1', 'Sales');
INSERT INTO `mydb`.`Department` (`idDepartment`, `DepartmentName`) VALUES ('2', 'Product Management');
INSERT INTO `mydb`.`Department` (`idDepartment`, `DepartmentName`) VALUES ('3', 'Excute ');
INSERT INTO `mydb`.`Department` (`idDepartment`, `DepartmentName`) VALUES ('4', 'Procurement');
INSERT INTO `mydb`.`Department` (`idDepartment`, `DepartmentName`) VALUES ('5', 'IT');
```

```
INSERT INTO `mydb`.`Employee` (`idEmployee`, `LastName`, `FirstName`, `Department_idDepartment`) VALUES ('1', 'Best', 'Sales', '1');
INSERT INTO `mydb`.`Employee` (`idEmployee`, `LastName`, `FirstName`, `Department_idDepartment`) VALUES ('2', 'Handsome', 'Sales', '1');
INSERT INTO `mydb`.`Employee` (`idEmployee`, `LastName`, `FirstName`, `Department_idDepartment`) VALUES ('3', 'Beauty', 'Sales', '1');
INSERT INTO `mydb`.`Employee` (`idEmployee`, `LastName`, `FirstName`, `Department_idDepartment`) VALUES ('4', 'Crazy', 'Pm', '2');
INSERT INTO `mydb`.`Employee` (`idEmployee`, `LastName`, `FirstName`, `Department_idDepartment`) VALUES ('5', 'Good', 'CEO', '3');
INSERT INTO `mydb`.`Employee` (`idEmployee`, `LastName`, `FirstName`, `Department_idDepartment`) VALUES ('6', 'Cheap', 'Procure', '4');
INSERT INTO `mydb`.`Employee` (`idEmployee`, `LastName`, `FirstName`, `Department_idDepartment`) VALUES ('7', 'Best', 'IT', '5');
```

```
INSERT INTO `mydb`.`IssueTracker` (`idIssueTracker`, `IssueName`, `IssueOwner`, `DueDate`, `Project_idProject`) VALUES ('1', 'Supply Issue', 'Factory', '05/05/2016', '1');
```

```
INSERT INTO `mydb`.`IssueTracker` (`idIssueTracker`, `IssueName`, `IssueOwner`, `DueDate`,
`Project_idProject`) VALUES ('2', 'Technical Issue', 'TBD', '06/06/2016', '1');
```

```
INSERT INTO `mydb`.`WorksOn` (`Project_idProject`, `Employee_idEmployee`,
`Employee_Department_idDepartment`) VALUES ('1', '1', '1');
INSERT INTO `mydb`.`WorksOn` (`Project_idProject`, `Employee_idEmployee`,
`Employee_Department_idDepartment`) VALUES ('1', '4', '2');
INSERT INTO `mydb`.`WorksOn` (`Project_idProject`, `Employee_idEmployee`,
`Employee_Department_idDepartment`) VALUES ('2', '2', '1');
```

```
INSERT INTO `mydb`.`Product` (`idProduct`, `Model Name`) VALUES ('1', 'CoolProduct');
INSERT INTO `mydb`.`Product` (`idProduct`, `Model Name`) VALUES ('2', 'FantasyProduct');
INSERT INTO `mydb`.`Product` (`idProduct`) VALUES ('3');
```

```
INSERT INTO `mydb`.`OrderProduct` (`Order_idOrder`, `Product_idProduct`, `Qty`, `Price`) VALUES
('1', '1', '10', '100');
INSERT INTO `mydb`.`OrderProduct` (`Order_idOrder`, `Product_idProduct`, `Qty`, `Price`) VALUES
('2', '1', '10', '100');
INSERT INTO `mydb`.`OrderProduct` (`Order_idOrder`, `Product_idProduct`, `Qty`, `Price`) VALUES
('3', '1', '10', '100');
INSERT INTO `mydb`.`OrderProduct` (`Order_idOrder`, `Product_idProduct`, `Qty`, `Price`) VALUES
('4', '2', '20', '200');
INSERT INTO `mydb`.`OrderProduct` (`Order_idOrder`, `Product_idProduct`, `Qty`, `Price`) VALUES
('5', '2', '20', '200');
INSERT INTO `mydb`.`OrderProduct` (`Order_idOrder`, `Product_idProduct`, `Qty`, `Price`) VALUES
('6', '3', '30', '300');
INSERT INTO `mydb`.`OrderProduct` (`Order_idOrder`, `Product_idProduct`, `Qty`, `Price`) VALUES
('7', '3', '30', '300');
INSERT INTO `mydb`.`OrderProduct` (`Order_idOrder`, `Product_idProduct`, `Qty`, `Price`) VALUES
('8', '3', '30', '300');
```

```
INSERT INTO `mydb`.`Requirement` (`Project_idProject`, `Product_idProduct`, `Price`) VALUES ('1', '1',
'10');
INSERT INTO `mydb`.`Requirement` (`Project_idProject`, `Product_idProduct`, `Price`) VALUES ('2', '2',
'200');
```

```
SELECT * FROM mydb.Contact where Account_idAccount=5;
SELECT * FROM mydb.`Order` where Account_idAccount=2;
SELECT * FROM mydb.Account where idAccount NOT IN (SELECT Account_idAccount FROM mydb.`Order`
where idOrder is not null);
SELECT * FROM mydb.Project where EAU >1000;
SELECT * FROM mydb.Employee where idEmployee =any (SELECT Employee_idEmployee FROM mydb.WorksOn
where Project_idProject=1);
```

References -

https://developer.salesforce.com/docs/atlas.en-us.fundamentals.meta/fundamentals/adg_database_concepts_relational.htm