Database Project

Edgar Buenrostro

CMPS 342

Fall 2010

Table of Contents

Phase I …….. 3

Fact-Finding Techniques and Information Gathering ……...4

Introduction to the Enterprise/Organization……...4

Structure of the Enterprise……..4

Itemized Descriptions of Major Objects…….5

Data Views and Operations for User Groups……6

Entity Set Description…….6

Related Entity Set……12

ER Diagram……...13

Phase II …….. 14

ER Model and the Relational Model……15

ER Database to Relational Database……18

Relation Instances ……27

Queries……..31

Query Representation ……..31

Phase III ……. 37

SQL*Plus……..38

Schema Objects in Oracle…… 38

Schema Objects in This Project……41

SQL Queries……48

Phase IV ……. 55

Common Features in Oracle PL?SQL and Microsoft Transact-SQL……...55

Oracle PL/SQL……55

Oracle PL/SQL SubPrograms……...60

Phase V……... 63

Daily User Activities……64

Relations, Views and Subprograms……..64

Application Screenshots……65

Code Description and GUI Design…….. 68

Development Process……70

Conclusion……70

Phase I:
Fact-Finding, Information Gathering and Conceptual Database Design

Edgar Buenrostro
CMPS 342
Fall 2010

Fact-Finding Techniques and Information Gathering

The database will keep track of data for a small gardening company. The main fact-finding technique used to gather information needed to design the database was interviewing. An interview with the owners of the company was necessary to find out what needs the company has and how a database can help the business.

The first step was to find out exactly what kind of data they would need keep stored. They stored information regarding current employees, suppliers and clients. Once they started imagining how the database can be used for their company, they expressed how important it is to keep other types of information stored. Certain clients have very unique preferences regarding their gardens and flowers and they often like to keep the same arrangements for the same season next year in order to stay consistent and keep the pricing the same. This means that the details of that project needs to be saved as well, so it can be accessed in future years.

Introduction to the Enterprise/Organization

The gardening company in this case is called Buenrostro's Gardening. It's a very small family business that has been in business for about a decade. The services they provide evolved marginally from the initial years. At first, they primarily focused on tending to the flower beds and occasionally the shrubs. Later the services offered expanded to include landscaping and other bigger tasks.

Structure of the Enterprise

The company is actually very small because it is a family business. The two owners are a married couple and are the primary workers in the business. There are also other workers on occasion that also tend to be part of the family. The pay and specific

hours of those workers have to be organized. The company has numerous clients who all receive somewhat different services. There are some clients that are primarily concerned with the care of their flower beds and shrubs. There are still others that receive those services in addition to landscaping and mowing the lawn. Some receive routine weekly maintenance while others only prefer occasional plantings. All of the flowers that are used for the projects are obtained by Buenrostro's Gardening via other suppliers that also need to be saved in the database.

Itemized Descriptions of Major Objects

The main data that has to be organized is supplier data. This is, in some ways, where business starts. It can keep track of all the suppliers that are used for the plants. This will have a relationship with another entity called Plants. This will keep track of those plants that were supplied along with the name, plant type and quantity. This needs a relationship with a Project entity because the specific types of plants and the number of plants has to be saved somewhere so that it is known exactly what services the client received. Employee's information has to be kept in its own. This will be a very small bit of data because there are so few employees, but it is important to keep track of each employee's hours, work days, and pay rate. It is related to the Project entity via a relationship called Works On. This relationship will actually keep track how many hours the employee worked on that specific project. The Project entity has a relationship with an entity called Client. This keeps information on the clients' names, address and the rate that they are charged.

Data Views and Operations for User Groups

All of the information that is going to be stored in the database will be for the benefit of the owners of the company and none of it is created for the customer. The customer will have zero access to it and will not even be able to view it. All the information will be accessible only to the two owners and me. They will have to learn to add, delete, modify and access the information using the user interface that will eventually be created.

Entity Set Description

Supplier

- Description: Contains information regarding the other companies that supply Buenrostro's Gardening with flowers and other plants.
- Candidate keys: Supplier_PK
- Primary Key: Supplier_PK
- Strong/Weak Entity: Strong
- Fields to be indexed: Supplier_PK
- Attributes: Supplier_PK, Name, Address, Phone

Name	**Supplier_PK**	**Name**	**Address**	**Phone**
Description	The primary key	Supplier's name.	Address of the supplier	Supplier's phone number
Domain/Type	Integer	String	String	String
Value – Range	1…n	n/a	n/a	n/a
Default Value	None	None	None	None
Nullable?	No	No	No	No
Unique?	Yes	Yes	Yes	Yes
Single/Multiple	Single	Single	Multiple	Single
Simple/Composite	Simple	Simple	Composite	Simple

Plants

- Description: Keeps track of all the plants that have been obtained from various suppliers. It should keep the name of the plant, color, if applicable and quantity. There will be frequent updating of the records in this table because there will be a constant supply of new plants.
- Candidate keys: Plant_PK
- Primary Key: Plant_PK
- Strong/Weak Entity: Strong
- Fields to be indexed: Plant_PK
- Attributes: Plant_PK, Name, Color, Quantity

Name	**Plant_PK**	**Name**	**Color**	**Quantity**
Description	The primary key	Plant's name.	Plant color if applicable.	How many of that particular plant.
Domain/Type	Integer	String	String	Integer
Value – Range	1…n	n/a	n/a	1…n
Default Value	None	None	Null	None
Nullable?	No	No	Null	No
Unique?	Yes	Yes	No	No
Single/Multiple	Single	Single	Single	Single
Simple/Composite	Simple	Simple	Simple	Simple

Project

- Description: Keeps track of certain projects that will be done for clients. Such projects can have very specific quantities, colors, and types of flowers that were requested by the client. This is meant to keep it stored just in case it's needed for reference in the future and the client wants to keep it the same.
- Candidate keys: Project_PK
- Primary Key: Project_PK
- Strong/Weak Entity: Strong
- Fields to be indexed: Project_PK
- Attributes: Project_PK, StartDate, EndDate, Season, EstimatedPrice, Description

Name	**Project_PK**	**StartDate**	**EndDate**	**Season**	**Estimated Price**	**Description**
Description	The primary key	Project's start date	Project's end date	The season for the project.	The estimated price for the project.	Describing the project.
Domain/Type	Integer	DateTime	DateTime	Integer	Double	String
Value – Range	1…n	n/a	n/a	1…n	0…n	n/a
Default Value	None	None	Null	None	None	None
Nullable?	No	Yes	Yes	No	No	No
Unique?	Yes	No	No	No	No	No
Single/Multiple	Single	Single	Single	Single	Single	Single
Simple/Composite	Simple	Simple	Simple	Simple	Simple	Simple

Employee

- Description: Keeps track of all the employee's information which includes their name and pay rate.
- Candidate keys: Employee_PK
- Primary Key: Employee_PK
- Strong/Weak Entity: Strong
- Fields to be indexed: Employee_PK
- Attributes: Employee_PK, FirstName, LastName, PayRate

Name	**Employee_PK**	**FirstName**	**LastName**	**PayRate**
Description	The primary key	Employee's first name.	Employee's last name.	Employee's hourly pay rate
Domain/Type	Integer	String	String	Double
Value – Range	1…n	n/a	n/a	1…n
Default Value	None	None	None	None
Nullable?	No	No	No	No
Unique?	Yes	Yes	No	No
Single/Multiple	Single	Single	Single	Single
Simple/Composite	Simple	Simple	Simple	Simple

Clients

- Description: Keeps track the clients' information. This includes the name, address and how much they are charged for the services.
- Candidate keys: Client_PK
- Primary Key: Client_PK
- Strong/Weak Entity: Strong
- Fields to be indexed: Client_PK
- Attributes: Client_PK, FirstName, LastName, Address, Rate

Name	**Client_PK**	**FirstName**	**LastName**	**Address**	**Rate**
Description	The primary key	Client's first name.	Client's last name	Client's address.	Rate charged.
Domain/Type	Integer	String	String	String	Double
Value – Range	1…n	n/a	n/a	n/a	1…n
Default Value	None	None	None	None	None
Nullable?	No	No	No	No	No
Unique?	Yes	No	No	No	No
Single/Multiple	Single	Single	Single	Multiple	Single
Simple/Composite	Simple	Simple	Simple	Simple	Composite

Data Views and Operations for User Groups

Supplies

- Description: This is a relationship between the Suppliers entity and the Plants entity. Each supplier supplies some sort of plant to the business.
- Mapping cardinality: M:N
- Participation constraint: The left side has a partial participation constraint because not all of the suppliers will be supplying plants all the time. There will be times where the supplier will be in the database, but they won't be supplying anything currently, but perhaps in the future.

UsedOn

- Description: This is a relationship between Plants and Project that indicates which plants and how many were used on a certain project.
- Mapping cardinality: M:N
- Participation constraint: The left side has a partial participation constraint because not all plants will be used on a project. There are sometimes leftovers. The right side has complete participation because all of the projects require some plants.

WorksOn

- Description: This is a relationship between the Employee entity and the Project entity. It also keeps track of the hours that was worked on that particular project.
- Mapping cardinality: M:N

- Participation constraint: The left side has a complete participation because all the employees work on a project. The right side has complete participation because they all have at least one employee to work on that project.

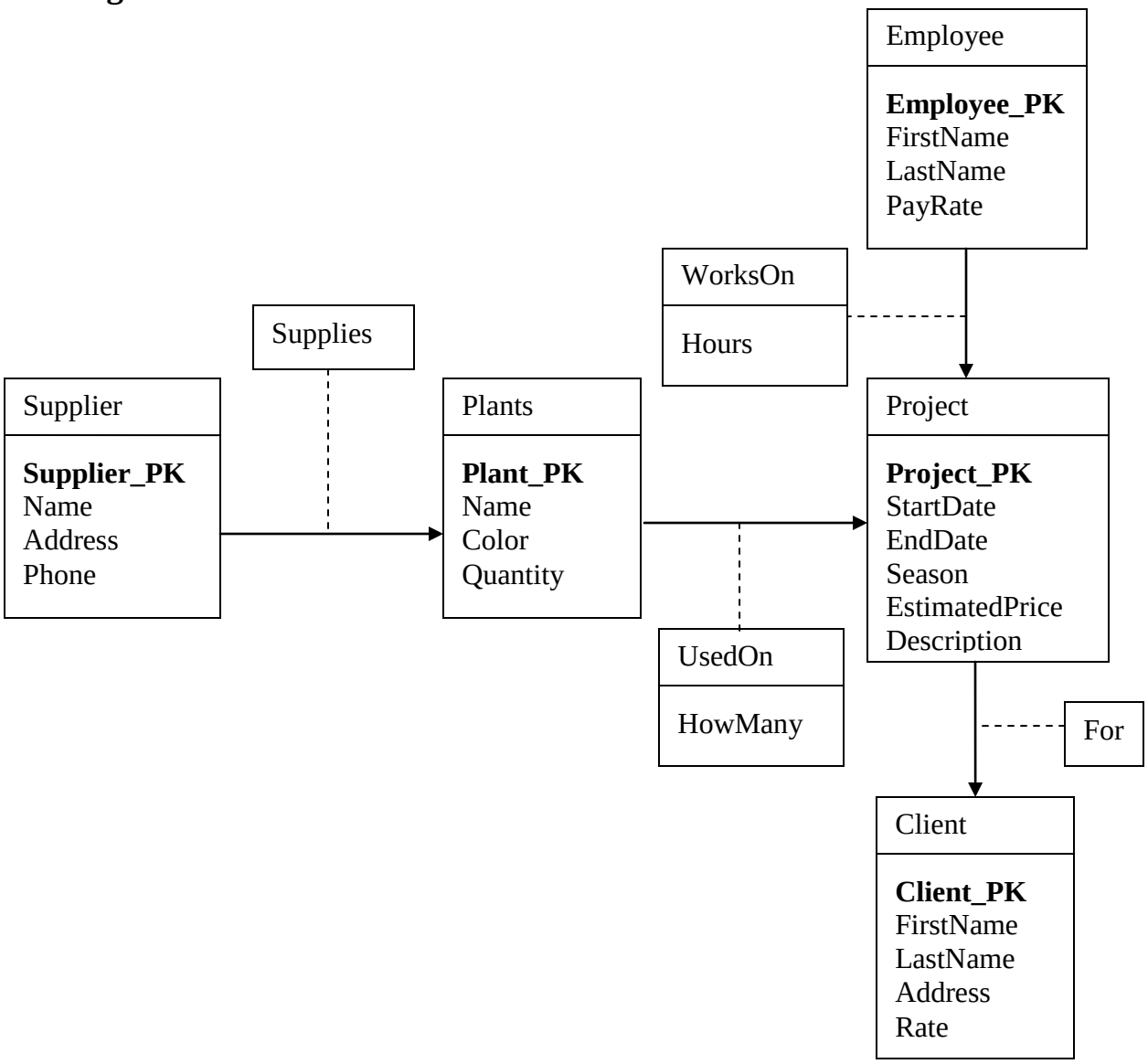
For

- Description: This is a relationship between the Project entity and the Client entity. It specifies which project is done for which client.
- Mapping cardinality: M:N
- Participation constraint: The left side has a partial participation because some of the projects may be done certain years and sometimes it's not done for anybody. The right side also has partial participation because sometimes they don't want an elaborate project. They just might want routine services.

Related Entity Set

The relationships consist of the ones titled Supplies, UsedOn, WorksOn and For. The Supplies describes the relationship between the supplier and the plants it supplies the business. UsedOn is between the plants and the projects they are used on. It also describes how many of each particular plant are used on that project. WorksOn describes the Employee entity and how many hours the person worked on a project. For is a relationship between the Project and the Client it's for.

ER Diagram


Employee
Employee_PK
FirstName
LastName
PayRate
WorksOn
Hours
Supplies
Supplier
Supplier_PK
Name
Address
Phone
Plants
Plant_PK
Name
Color
Quantity
Project
Project_PK
StartDate
EndDate
Season
EstimatedPrice
Description
UsedOn
HowMany
For
Client
Client_PK
FirstName
LastName
Address
Rate

Phase II:
From ER Model to Relational Model

Edgar Buenrostro
CMPS 342
Fall 2010

ER Model and the Relational Model

Description

Now with the ER Model done, it is time to convert the design to the Relational Model. This gives us the opportunity to represent the database as a collection of tables with columns and records. The database becomes a set of relations consisting of those tables. Each row represents a record of a specific entry into that table. It is a tuple that holds the information for a specific instance of that entity or relation. Each column will contain information regarding that attribute.

Comparison

The ER model and the Relational model are both very important steps in designing and eventually implementing a well-built database. The previous phase dealt primarily with the design of the ER model. That model was much more general in nature and does not provide enough detail for the proper implementation of the design. The primary purpose of the ER model is to create the base design at first that can be expanded into a more detailed relational model. It is helpful because of its visual nature and the overall design and structure of the database is more immediately apparent.

The relational model is a bit harsher on the eyes. It doesn't depend on on the visuals and diagramming. While, it doesn't provide as much of an overarching view of the design it allows us to look at the details of the databases and its tables. The relational model is primarily built of various tables and grids that contain the names of the attributes as columns. There is information in each column and there is a more detailed view of the attribute types, constraints and keys. The relational model for this particular database of Buenrostro's Gardening will be shown in this phase.

Conversion from ER Model to Relational Model

In order to convert from an ER model to a relational model, we need to establish certain characteristics for each attribute in each of the tables. The relational model is much more detailed, so we have to be aware of the domains, types, ranges, keys, constraints, foreign keys, candidate keys and any other information that might be needed. First thing, we have to decide which tables will have a primary key attribute. It can either be a single attribute acting as the primary key or it can be a composite primary key where it is composed of two or more attributes that together will give make the attribute unique to any other record in the table. Once it is known which type of primary key if any will be used, actually decide which attribute will be the primary key. In addition to that, the foreign keys have to be fine tuned so that each table that has a relationship with another table uses foreign keys properly. The foreign key has to make a reference to the primary key of the table it is referencing. Also be aware of any other possible candidate keys.

Each attribute also has to have their characteristics defined. What type will it be? What will be its domain and is their any specific range that it has to fall in? The relational model also has to specify which attributes will allow for a NULL to be stored. This is very important especially for attributes that will need to be unique like keys. Allowing for NULL in your attribute means that it allows for multiple records to have that NULL value. That gets rid of its unique characteristic.

During this time, the business logic required for the attributes should be established. Sometimes, a company requires specific things out of the data they store and check constraints allow for making sure the data follows that business logic. Keeping these constraints and all of the other information in mind for the conversion means we'll

have all the information organized in the relational model ready for its eventual implementation.

Constraints

A lot of the data that will be saved into the tables will require some forms of restrictions in order to make sure the information stored makes sense for what the database is needed and for what the system can handle. Implementing constraints into the relational model means we can keep track of those restrictions and keep the exact type of data that we want and need for the project. A constraint can apply to numeric data by restricting the domain of that data. Domain constraints can apply to any attribute by specifying its data type as well as its limited range if there is any. For strings, this can mean that there is a limit on how many characters can be stored for that specific entry.

There are also key constraints that deal with primary keys, foreign keys and candidate keys. The primary key has to be a unique attribute that acts as the way that specific record can be identified and referred to. Another type of key constraint is the foreign key. The foreign key is a way that a table can have a relationship with a different table. The foreign key is an attribute that must be the primary key of the table it is referencing. If the foreign key placed in the table does not exist as a primary key on the other table then it is not valid. Another constraint is whether a NULL should be allowed or not. If the attribute needs to be unique like a primary key then it does not make sense to allow a NULL. If there is no need for uniqueness then sometimes it makes sense to allow for a NULL if no other default value makes sense. Sometimes having zero as the default value doesn't make sense, so a NULL is necessary. Other times it is possible that that specific attribute is not applicable to that particular record so a NULL can be used.

ER Database to Relational Database

Supplier

Attributes

- Supplier_PK
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL
- Name
    - Domain: String must be at least one character. Not NULL.
- Address
    - Domain: String must be composed of numbers or letters. Not NULL.
- Phone
    - Domain: Unsigned integer. Must be ten digits. Not NULL.

Constraints

- Primary key: Supplier_PK is the primary key. It must be unique and not NULL.

Candidate Keys

- Supplier_PK

Order

Attributes

- Order_PK
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL
- Date
    - Domain: Datetime. Not NULL.

- Supplier_FK
    - Domain: Integer that is the primary key of the Supplier table. Not NULL.

Constraints

- Primary key: Order_PK is the primary key. It must be unique and not NULL.

Candidate Keys

- Order_PK

Contains Relation

Attributes

- Quantity
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL
- Price
    - Domain: Double. Not NULL.
- Order_FK
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL. It's the pimary key of the Order table.
- Plants_FK
    - Domain: Integer that's the primary key of the Plants table. Not NULL.
- Color_FK
    - Domain: Integer that's the primary key of the Color table. Not NULL.

Constraints

- Primary key: Order_PK is the primary key. It must be unique and not NULL.
- Foreign key: Order_FK, Plants_FK and Color_FK are foreign keys.

Candidate Keys

- Order_PK

Color

Attributes

- Color_PK
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL
- Name
    - Domain: String. Not NULL.

Constraints

- Primary key: Color_PK is the primary key. It must be unique and not NULL.

Candidate Keys

- Color_PK

Plants

Attributes

- Plants_PK
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL
- Name
    - Domain: String. Not NULL.
- Quantity
    - Domain: Unsigned integer. Not NULL.
- Order_FK

    - Domain: Unsigned integer that is the primary key of the Order key. Not NULL.
- Color_FK
    - Domain: Unsigned integer that is the primary key of the Color table.

Constraints

- Primary key: Color_PK is the primary key. It must be unique and not NULL.
- Foreign key: Color_FK and Order FK are foreign keys.

Candidate Keys

- Color_PK

UsedOn Relation

Attributes

- HowMany
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL
- StartDate
    - Domain: DateTime. Not NULL.
- EndDate
    - Domain: DateTime. Not NULL.
- Price
    - Domain: Double. Not NULL.
- Plants_FK
    - Domain: Unsigned integer that is the primary key of the Plants table. Not NULL.

- Project_FK
    - Domain: Unsigned integer that is the primary key of the Project table. Not NULL.

Constraints

- Foreign key: Plant_FK and Project_FK are foreign keys.

Candidate Keys

- Plant_FK
- Project_FK

Project

Attributes

- Project_PK
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL
- StartDate
    - Domain: DateTime. Not NULL.
- EndDate
    - Domain: DateTime. Not NULL.
- Season
    - Domain: String.
- EstimatedPrice
    - Domain: Double. Not NULL.
- Description
    - Domain: String.

- For_FK
    - Domain: Primary key of a client.

Constraints

- Foreign key: For_FK is a foreign key.

Candidate Keys

- Project_PK

Employee

Attributes

- Employee_PK
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL
- FirstName
    - Domain: String. Not NULL.
- LastName
    - Domain: String. Not NULL.
- PayRate
    - Domain: Double. Not NULL.
- EstimatedPrice
    - Domain: Double. Not NULL.
- Description
    - Domain: String.

Constraints

- Primary Key: Employee_PK is the primary key.

Candidate Keys

- Employee_PK

Client_PK

Attributes

- Hours
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL
- Employee_FK
    - Domain: Unsigned integer must be the primary key of the Employee table. Not NULL.
- Project_FK
    - Domain: Unsigned integer must be the primary key of the Project table. Not NULL.

Constraints

- Foreign Key: Employee_FK and Project_FK are foreign keys.

Candidate Keys

- Employee_FK
- Project_FK

For Relation

Attributes

- Hours
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL

- Employee_FK
    - Domain: Unsigned integer must be the primary key of the Employee table. Not NULL.
- Project_FK
    - Domain: Unsigned integer must be the primary key of the Project table. Not NULL.

Constraints

- Foreign Key: Employee_FK and Project_FK are foreign keys.

Candidate Keys

- Employee_FK
- Project_FK

Client

Attributes

- Client_PK
    - Domain: Unsigned integer: 1 to 2^32 – 1. Cannot be NULL
- FirstName
    - Domain: String. Not NULL.
- LastName
    - Domain: String. Not NULL.
- Address
    - Domain: String. Not NULL.
- Rate

- Domain: Double. Not NULL.

Constraints

- Primary key: Client_PK is the primary key.

Candidate Keys

- Client_PK

Relation Instances

Supplier

Supplier(**Supplier_PK**, Name, Address, Phone)

Supplier_PK	Name	Address	Phone
1	Do Right's Plant Growers	2155 Horton Avenue Los Angeles, CA 90025	8055252155
2	Bolles Nursery	1112 Wible Road Bakersfield, CA 93304	6613988128
3	White Forest Nursery	300 Morning Drive Bakersfield, CA 93306	6613666291
4	Calloway Nursery	2828 Calloway Drive Bakersfield, CA 93312	6615887708

Order

Order(**Order_PK**, Date, Supplier_FK)

Order_PK	Date	Supplier_FK
1	8/15/2010	2155 Horton Avenue Los Angeles, CA 90025
2	9/13/2010	1112 Wible Road Bakersfield, CA 93304
3	9/25/2010	300 Morning Drive Bakersfield, CA 93306
4	10/15/2010	2828 Calloway Drive Bakersfield, CA 93312

Contains

Contains(Quantity, Price, Order_FK, Plant_FK, Color_FK)

Quantity	Price	Order_FK	Plant_FK	Color_FK
15	$9.50	1	1	1
10	$4.00	1	2	2
5	$12.50	1	3	1
10	$3.00	2	4	3
4	$15.00	2	6	NULL
20	$4.00	3	2	5
16	$7.50	3	7	4
8	$7.00	3	5	4
9	$7.50	4	7	1
10	$12.00	4	3	5
2	$14.50	4	1	1

Color

Color(**Color_PK**, Name)

Supplier_PK	Name
1	Red
2	Yellow
3	Pink
4	Purple
5	White

Supplier

Plants (**Plants_PK**, Name, Quantity, Color_FK)

Plants_PK	Name	Quantity	Color_FK
1	Impatiens	20	3
2	Cyclamen	10	4
3	Zinnia	10	2
4	Zinnia	5	1
5	Poppies	9	1
6	Privet	5	NULL
7	Snapdragon	12	3

UsedOn

UsedOn(HowMany, StartDate, EndDate, Price, Plants_FK, Project_FK)

HowMany	StartDate	EndDate	Price	Plants_FK	Project_FK
5	8/15/2010	8/17/2010	$10.00	1	1
10	8/15/2010	8/17/2010	$5.50	2	1
12	8/15/2010	8/17/2010	$3.00	4	1
4	9/15/2010	9/16/2010	$15.00	1	2
5	9/15/2010	9/16/2010	$8.00	5	2
3	9/18/2010	9/18/2010	$8.00	7	3
3	9/18/2010	9/18/2010	$8.00	5	3
2	9/25/2010	9/25/2010	$16.00	6	4
5	9/25/2010	9/25/2010	$8.00	3	4
2	10/16/2010	10/17/2010	$16.00	6	5
20	10/16/2010	10/17/2010	$10.00	1	5

Employee

Employee(**Employee_PK**, FirstName, LastName, PayRate, PayRate)

Employee_PK	FirstName	LastName	PayRate
1	Jose	Peña	$9.00
2	Antonia	Buentello	$10.50
3	Sergio	Valdez	$10.50
4	Victor	Morelos	$9.00
5	Manuela	Villa	$9.00
6	John	Collins	$12.50

WorksOn

WorksOn(Hours, Employee_FK, Project_FK)

Hours	Employee_FK	Project_FK
8	1	1
8	4	1
10	2	2
10	3	2
4	6	3
12	5	4
15	2	5
8	3	5

Project

Project(**Project_PK**, StartDate, EndDate, Season, EstPrice, Description, For_FK)

Project_PK	StartDate	EndDate	Season	EstPrice	Description	For_FK
1	8/15/2010	8/17/2010	Summer	$100.00	NULL	1
2	9/15/2010	9/16/2010	Summer	$120.00	NULL	2
3	9/18/2010	9/18/2010	Summer	$80.00	NULL	3
4	9/25/2010	9/25/2010	Fall	$95.00	NULL	4
5	10/16/2010	10/17/2010	Fall	$110.00	NULL	1

Client

Client(**Client_PK**, FirstName, LastName Address, Rate)

Client_PK	FirstName	LastName	Address	Rate
1	Dwight	Byrum	2135 Newport Drive Bakersfield, CA 93307	$13.00
2	Tracy	Burns	388 Alexis Avenue Bakersfield, CA 93308	$12.00
3	Chris	Sanders	1190 Howley Drive Bakersfield, CA 93308	$13.00
4	Heather	Johns	8865 Aguila Road Bakersfield, CA 93307	$12.50

Queries

- **Select the employees who have worked on projects for Dwight Byrum.**
- **Select projects that require plants other than flowers.**
- **Select the supplier that has sold Impatiens.**
- **Select the suppliers that have supplied the most recent order.**
- **Select the employee who has worked the most hours on a project.**
- **Select the clients that have a fall project.**
- **Select the clients that will have a project in the summer.**
- **Select the project that requires the most Impatiens.**
- **Select the plants that are red.**
- **Select the plant that is most numerous at the moment.**

Query Representation

Select the employees who have worked on projects for Dwight Byrum.

Relational Algebra:

π(e.*) σ(Employee e X WorksOn w X Project p X Client c)
e.Employee_PK = w.Employee_FK ^
w.Project_FK = p.Project_PK ^
p.For_FK = c.Client_PK ^
c.FirstName = “Dwight” ^
c.LastName = “Byrum”

Tuple Relational Calculus:

$$\{ e \mid employee(e) \wedge (\exists w)(WorksOn(w) \wedge e.Employee\_PK = w.Employee\_FK \wedge (\exists p)(Project(p) \wedge p.Project\_PK = w.Project\_FK \wedge (\exists c)(Client(c) \wedge c.Client\_PK = p.For\_FK \wedge c.FirstName = \text{“Dwight”} \wedge c.LastName = \text{“Byrum”})) \}$$

Domain Relational Calculus:

{<e, f, l, p> | Employee(e,f,l,)^(∃p)(WorksOn(_,e,p)^(∃c)(Project(p,_,_,_,_,c)^
Client(c, “Dwight”, “Byrum”,_,_)))

}

Select projects that require plants other than flowers.

Relational Algebra:

π(p.*) σ(Plants pl X UsedOn u X Project p)
u.Project_FK = p.Project_PK ^
u.Plant_FK = pl.Plant_PK ^
pl.color = NULL

Tuple Relational Calculus:

{ p | Project(p) ^ (∃u)(UsedOn(u) ^ u.Project_FK = p.Project_PK ^
(∃pl)(pl.Plant_PK = u.Plant_FK ^ pl.Color = NULL))
}

Domain Relational Calculus:

{<p> | Project(p,_,_,_,_,_,_) ^ (∃pl)(Plant(pl,_,_,NULL) ^ UsedOn(_,_,_,_,pl,p))}

Select the suppliers that have sold Impatiens.

Relational Algebra:

π(s.*) σ(Supplier s X Order o X Contains c X Plants p)
s.Supplier_PK = o.Supplier_FK ^
c.Order_FK = o.Order_PK ^
c.Plant_FK = p.Plant_PK ^
p.Name = “Impatiens”

Tuple Relational Calculus:

{ s | Supplier(s) ^ (∃o)(Order(o) ^ o.Supplier_FK = s.Supplier_PK ^
(∃c)(Contains(c) ^ c.Order_FK = o.Order_PK ^
(∃p)(Plants(p) ^ p.Plant_PK = c.Plant_FK ^
p.Name = "Impatiens")))
}

Domain Relational Calculus:

{ <s> | Supplier(s,_,_,_) ^ (∃o)(Order(o,_,s) ^ (∃p)(Plants(p,_,_,_) ^
(∃c)(Contains(_,_,o,p,_)))
}

Select the suppliers that have supplied the most recent order.

Relational Algebra:

π(s1.*) Supplier s1 - σ(Supplier s X Order o X Order o2)
s.Supplier_PK = o.Supplier_FK ^
o.Date < o2.Date

Tuple Relational Calculus:

{ s | Supplier(s) ^ (∃o)(Order(o) ^ o.Supplier_FK = s.Supplier_PK ^
(∀o2)(Order(o2) → (o2.date <= o.date)))
}

Domain Relational Calculus:

{<s> | Supplier(s,_,_) ^ (∃o)(∃d)(Order(o,d,s) ^ (∀o2)(Order(o2,<d,_))}

Select the employee who has worked the most hours on a project.

Relational Algebra:

π(e1.*) (Employee e1 - σ(Employee e X WorksOn w X WorksOn w2))
e.Employee_PK = w.Employee_FK ^
w.Hours < w2.Hours

Tuple Relational Calculus:

{ e | Employee(e)^(∃w)(WorkedOn(w) ^ w.Employee_FK = w.Employee_PK^
(∀w2)(WorkedOn(w2) → (w2.hours <= w.hours)))
}

Domain Relational Calculus:

{<e> | Employee(e,_,_,_,_)^(∃h)(WorksOn(h,e,_) ^ (∀e2)(Order(<h,e2,_))}

Select the clients that have a fall project.

Relational Algebra:

π(c.*) σ(Client c X Project p)
c.Client_PK = p.For_FK ^
p.Season = "Fall"

Tuple Relational Calculus:

{ c | Client(c) ^ (∃p)(Project(p) ^ p.For_FK = c.Client_PK ^ p.Season = "Fall")}

Domain Relational Calculus:

{<c> | Client(c,_,_,_,_) ^ (∃p)(Project(p,_,_, "Fall",_,_))}

Select the clients that will have a project in the summer.

Relational Algebra:

π(c.*) σ(Client c X Project p)
c.Client_PK = p.For_FK ^
p.Season = "Summer"

Tuple Relational Calculus:

{ c | Client(c) ^ (∃p)(Project(p) ^ p.For_FK = c.Client_PK ^ p.Season =

"Summer")}

Domain Relational Calculus:

{<c> | Client(c,_,_,_,_) ^ (∃p)(Project(p,_,_, "Summer",_,_))

Select the project that requires the most Impatiens

Relational Algebra:

π(p1.*) (Project p1 - σ(Project p X UsedOn u X UsedOn u2 X Plant pa))
e.Employee_PK = w.Employee_FK ^
p.Project_PK = u.Project_FK ^ u.Plant_FK = pa.Plant_PK ^
pa.Name = "Impatiens" ^ u2.Plant_FK = pa.Plant_PK ^
u.HowMany < u2.HowMany

Tuple Relational Calculus:

$\exists \forall$
{ p | Project(p) ^ ($\exists$u)(UsedOn(u) ^ u.Project_FK = p.Project_PK^
($\exists$pl)(Plant(pl) ^ pl.Name = "Impatiens" ^
pl.Plant_PK = u.Plant_FK ^
($\forall$u2)((UsedOn(u2)^u2.Plant_FK = pl.Plant_PK $\rightarrow$
(u2.HowMany<= u.HowMany)))
}

Domain Relational Calculus:

{<p> | Project(p,_,_,_,_,_)^ ($\exists$pa)(Plant(pa, "Impatiens",_) ^
($\exists$u)UsedOn(u,_,_,_pa,p) ^ ($\forall$p2)(UsedOn(<u,_,_,_pa,p2))}

Select the plants that are red.

Relational Algebra:

π(p.*) σ(Plant p X Color c)
p.Color_FK = c.Color_PK ^
c.Name= "Red"

Tuple Relational Calculus:

{ p | Plant(p) ^ ($\exists$c)(Color(c) ^ c.Color_PK = p.Color_FK ^ c.Name = "Red")}

Domain Relational Calculus:

{<p> | ($\exists$c)(Color(c, "Red"), Plant(p,_,_,c))}

Select the plant that is most numerous at the moment.

Relational Algebra:

$$\pi(p1.name)\ (Plant\ p1 - \sigma_{p2.Quantity < p3.Quantity}(Plant\ p2 \times Plant\ p3))$$

Tuple Relational Calculus:

$$\{ p \mid Plant(p) \wedge (\forall p2)(Plant(p2) \rightarrow (p2.Quantity <= p.Quantity))\}$$

Domain Relational Calculus:

$$\{<p,n,q> \mid Plant(p, n,q) \wedge (\forall p2)(Plant(p2,\_,<q)\}$$

Phase III:
Implementation of the Relational Database

Edgar Buenrostro
CMPS 342
Fall 2010

SQL*Plus

The relational model we have created in the previous phase provides us with enough detailed information on the records we'll be saving in the database as well as the structure and characteristics regarding the relationships, primary keys and other constraints needed for the database. To do this, a database management system is needed to keep track of the database and manipulate its records. In order to implement this database, we'll be using the Oracle Database Management System. This also makes use of SQL*Plus which is a command line system that allows us to create, access and manipulate the tables of our database through the use of queries, scripts or other commands. SQL*Plus makes use of SQL, a query language that is used for creating those commands that affect the database.

Schema Objects in Oracle

A schema in oracle can be comprised of various schema objects. Usually a schema is associated with one user on the database and the schema objects are contained within that schema. Those objects are typically structures used to save data related to the database. Examples of schema objects include:

Table

A table is probably the most commonly used schema object. It is the object that stores the information relating to a relation or entity that has been previously outlined in the relational model. Each table consists of attributes or columns. Each one of those attributes should have its own type. The tables can also have primary keys, foreign keys or other constraints.

Views

Views are used when there is a specific query that is frequently used. The view is a common query stored for later use and it returns the resulting tuples from that query. It can be used within other queries and used a bit like a table although it's not really storing any records in it of itself. Each time it is used, it is newly invoked and it might be different every time depending on the query.

Dimensions

These schema objects help reorganize records to make it easier for someone to use or to be used with other queries. They are typically meant for categorizing data.

Sequences

Sequences are objects that helps generate a numbered sequence. This is good for primary keys since each key has to be unique. The sequence helps auto increment to make sure each primary key is different. It can also be used to keep an order of something like commands and it can even be helpful for rollbacks since it needs to be in reverse order.

Synonyms

A synonym is simply another term that can be used for schema objects. A synonym can be created for various objects like tables, views and packages.

Indexes

An index is an object that helps with efficiency of accessing or searching through the records. Retrieval and traversal of the tuples is a little faster, so it's a good idea to apply an index to attributes that are commonly used or retrieved. Since the index needs to

be created there is more space taken up, so the indexing should probably not be applied to too many attributes.

Database links

A database link is a connection between two databases that allows the access of information saved in those databases. It is usually a one way link, so the database being accessed cannot use the same link to access information the opposite way.

Stored Procedures and Functions

These are stored sets of commands that are meant to perform some type of manipulation on the database. They usually consist of SQL commands and queries. Procedures are usually just a set of tasks and they sometimes have their own parameters that are either IN, OUT or IN OUT. Functions are similar, but while procedures don't return a variable, functions do.

Packages

A package is a collection of objects like functions or procedures. The objects it holds together as a collection are usually related. It is divided into parts. One specification part takes care of the declarations, variables and cursors. The other part functions as the body and actually implements all those parts.

Schema Objects in this Project

This database project will primarily make use of tables as its schema objects. To create a table a SQL query has to be written with the following syntax.

```
create table tablename(
        PrimaryKeyName  int PRIMARY KEY,
        attributename   type NOT NULL,
        attributename2  type NULL,
        CONSTRAINT RelationshipName
                FOREIGN KEY(ForeignKeyName) REFERENCES
                TableReferenced(AttributeReferenced)
);
```

Syntax like this was used to create all of the object schemas on this database projects and they are as follows:

- eb_client — Client Relation
- eb_color — Color Relation
- eb_contains — Contains Relation
- eb_employee — Employee Relation
- eb_order — Order Relation
- eb_plant — Plant Relation
- eb_project — Project Relation
- eb_supplier — Supplier Relation
- eb_usedon — UsedOn Relation
- eb_workson — WorksOn Relation

eb_client

```
CS342 SQL> desc eb_client;
 Name                 Null?    Type
 -------------------- -------- ------------------------
 CLIENT_PK            NOT NULL NUMBER(38)
 FIRSTNAME            NOT NULL VARCHAR2(15)
 LASTNAME             NOT NULL VARCHAR2(15)
 ADDRESS              NOT NULL VARCHAR2(50)
 RATE                          NUMBER(5,2)
```

```
CS342 SQL> select * from eb_client;

 CLIENT_PK FIRSTNAME       LASTNAME   ADDRESS                                    RATE
---------- --------------- ---------- ---------------------------------------- ------
         1 Dwight          Byrum      2135 Newport Drive Bakersfield, CA 93307     13
         2 Tracy           Burns      388 Alexis Avenue Bakersfield, CA 93308      12
         3 Chris           Sanders    1190 Howley Drive Bakersfield, CA 93308      13
         4 Heather         Johns      8865 Aguila Road Bakersfield, CA 93307     12.5
```

eb_color

```
CS342 SQL> desc eb_color;
 Name                 Null?    Type
 -------------------- -------- ----------------------------------
 COLOR_PK             NOT NULL NUMBER(38)
 NAME                 NOT NULL VARCHAR2(20)
```

```
CS342 SQL> select * from eb_color;

  COLOR_PK NAME
---------- -------------------------
         1 Red
         2 Yellow
         3 Pink
         4 Purple
         5 White
```

eb_contains

```
CS342 SQL> desc eb_contains;
 Name            Null?    Type
 --------------- -------- --------------------------------
 QUANTITY        NOT NULL NUMBER(38)
 PRICE           NOT NULL NUMBER(5,2)
 ORDER_FK        NOT NULL NUMBER(38)
 PLANT_FK        NOT NULL NUMBER(38)
 COLOR_FK                 NUMBER(38)


CS342 SQL> select * from eb_contains;

  QUANTITY      PRICE   ORDER_FK   PLANT_FK   COLOR_FK
---------- ---------- ---------- ---------- ----------
        15        9.5          1          1          1
        10          4          1          2          2
         5       12.5          1          3          1
        10          3          2          4          3
         4         15          2          6
        20          4          3          2          5
        16        7.5          3          7          4
         8          7          3          5          4
         9        7.5          4          7          1
        10         12          4          3          5
         2       14.5          4          1          1

11 rows selected.
```

eb_employee

```
CS342 SQL> desc eb_employee;
 Name          Null?    Type
 ------------- -------- --------------------------------
 EMPLOYEE_PK   NOT NULL NUMBER(38)
 FIRSTNAME     NOT NULL VARCHAR2(15)
 LASTNAME      NOT NULL VARCHAR2(15)
 PAYRATE       NOT NULL NUMBER(5,2)
```

```
CS342 SQL> select * from eb_employee;

EMPLOYEE_PK FIRSTNAME       LASTNAME           PAYRATE
----------- --------------- --------------- ----------
          1 Jose            Perez                  9
          2 Antonia         Buentello           10.5
          3 Sergio          Valdez              10.5
          4 Victor          Morelos                9
          5 Manuela         Villa                  9
          6 John            Collins             12.5

6 rows selected.
```

eb_order

```
CS342 SQL> desc eb_order;
 Name            Null?    Type
 --------------- -------- --------------------------------
 ORDER_PK        NOT NULL NUMBER(38)
 ORDERDATE       NOT NULL DATE
 SUPPLIER_FK     NOT NULL NUMBER(38)


CS342 SQL> select * from eb_order;

  ORDER_PK ORDERDATE SUPPLIER_FK
---------- --------- -----------
         1 15-AUG-10           1
         2 13-SEP-10           2
         3 25-SEP-10           3
         4 15-OCT-10           4
```

eb_plant

```
CS342 SQL> desc eb_plant;
 Name          Null?    Type
 ------------- -------- --------------------------------
 PLANT_PK      NOT NULL NUMBER(38)
 NAME          NOT NULL VARCHAR2(20)
 QUANTITY      NOT NULL NUMBER(38)
 COLOR_FK               NUMBER(38)
```

```
CS342 SQL> select * from eb_plant;

  PLANT_PK NAME                          QUANTITY   COLOR_FK
---------- ------------------------- ---------- ----------
         1 Impatiens                          20          3
         2 Cyclamen                           10          4
         3 Zinnia                             10          2
         4 Zinnia                              5          1
         5 Poppies                             9          1
         6 Privet                              5
         7 Snapdragon                         12          3

7 rows selected.
```

eb_project

```
CS342 SQL> desc eb_project
 Name          Null?    Type
 ------------ -------- ------------------------------------
 PROJECT_PK    NOT NULL NUMBER(38)
 STARTDATE     NOT NULL DATE
 ENDDATE       NOT NULL DATE
 SEASON        NOT NULL VARCHAR2(7)
 ESTPRICE      NOT NULL NUMBER(5,2)
 DESCRIPTION            VARCHAR2(100)
 FOR_FK        NOT NULL NUMBER(38)


CS342 SQL> select * from eb_project;

PROJECT_PK STARTDATE ENDDATE   SEASON    ESTPRICE DESCRIPTION    FOR_FK
---------- --------- --------- ------- ---------- ------------- -------
         1 15-AUG-10 17-AUG-10 Summer         100                     1
         2 15-SEP-10 16-SEP-10 Summer         120                     2
         3 18-SEP-10 18-SEP-10 Summer          80                     3
         4 25-SEP-10 25-SEP-10 Fall            95                     4
         5 16-OCT-10 17-OCT-10 Fall           110                     1
```

eb_supplier

```
CS342 SQL> desc eb_supplier
 Name          Null?    Type
 ------------ -------- ------------------------------------
 SUPPLIER_PK  NOT NULL NUMBER(38)
 NAME         NOT NULL VARCHAR2(50)
 ADDRESS      NOT NULL VARCHAR2(50)
 PHONE        NOT NULL VARCHAR2(12)

CS342 SQL> select * from eb_supplier;

SUPPLIER_PK NAME                      ADDRESS                   PHONE
----------- ------------------------- ------------------------- ------------
          1 Do Rights Plant Growers   2155 Horton Avenue Los An 805-525-2155
                                      geles, CA 90025

          2 Bolles Nursery            1112 Wible Road Bakersfie 661-398-8128
                                      ld, CA 93304

          3 White Forest Nursery      300 Morning Drive Bakersf 661-366-6291
                                      ield, CA 93306

          4 Calloway Nursery          2828 Calloway Drive Baker 661-588-7708
                                      sfield, CA 93312
```

eb_usedon

```
CS342 SQL> desc eb_usedon
 Name          Null?    Type
 ------------ -------- ------------------------------------
 HOWMANY      NOT NULL NUMBER(38)
 STARTDATE             DATE
 ENDDATE               DATE
 PRICE        NOT NULL NUMBER(5,2)
 PLANT_FK     NOT NULL NUMBER(38)
 PROJECT_FK   NOT NULL NUMBER(38)

CS342 SQL> select * from eb_usedon;

   HOWMANY STARTDATE ENDDATE   PRICE   PLANT_FK PROJECT_FK
---------- --------- --------- ----- ---------- ----------
         5 15-AUG-10 17-AUG-10    10          1          1
        10 15-AUG-10 17-AUG-10   5.5          2          1
        12 15-AUG-10 17-AUG-10     3          4          1
         4 15-SEP-10 16-SEP-10    15          1          2
         5 15-SEP-10 16-SEP-10     8          5          2
         3 18-SEP-10 18-SEP-10     8          7          3
         3 18-SEP-10 18-SEP-10     8          5          3
         2 25-SEP-10 25-SEP-10     6          6          4
```

```
         5 25-SEP-10 25-SEP-10      8          3          4
         2 16-OCT-10 17-OCT-10     16          6          5
        20 16-OCT-10 17-OCT-10     10          1          5

11 rows selected.
```

eb_workson

```
CS342 SQL> desc eb_workson
 Name            Null?    Type
 --------------- -------- --------------------------------
 HOURS           NOT NULL NUMBER(38)
 EMPLOYEE_FK     NOT NULL NUMBER(38)
 PROJECT_FK      NOT NULL NUMBER(38)

CS342 SQL> select * from eb_workson;

     HOURS EMPLOYEE_FK PROJECT_FK
---------- ----------- ----------
         8           1          1
         8           4          1
        10           2          2
        10           3          2
         4           6          3
        12           5          4
        15           2          5
         8           3          5

8 rows selected.
```

SQL Queries

Select the employees who have worked on projects for Dwight Byrum.

```
select e.*
from    eb_employee e, eb_workson w, eb_project p,
eb_client c
where   e.employee_pk = w.employee_fk and
        w.project_fk = p.project_pk and
        p.for_fk = c.client_pk and
        c.firstname = 'Dwight' and c.lastname = 'Byrum'
;
```

REPORT:

EMPLOYEE_PK	FIRSTNAME	LASTNAME	PAYRATE
1	Jose	Perez	9
4	Victor	Morelos	9
2	Antonia	Buentello	10.5
3	Sergio	Valdez	10.5

Select projects that require plants other than flowers.

```
select p.*
from    eb_plant pl, eb_usedon u, eb_project p
where   u.project_fk = p.project_pk and
        u.plant_fk = pl.plant_pk and pl.color_fk IS NULL
;
```

REPORT:

```
PROJECT_PK STARTDATE ENDDATE   SEASON    ESTPRICE DESCRIPTION    FOR_FK
---------- --------- --------- ------- ---------- -------------- ------
         4 25-SEP-10 25-SEP-10 Fall            95                     4
         5 16-OCT-10 17-OCT-10 Fall           110                     1
```

Select the suppliers that have sold Impatiens.

```
select unique s.*
from    eb_supplier s, eb_order o, eb_contains c, eb_plant
p
where   s.supplier_pk = o.supplier_fk and
        c.order_fk = o.order_pk and
        c.plant_fk = p.plant_pk and
        p.name = 'Impatiens'
;
```

REPORT:

```
SUPPLIER_PK NAME                 ADDRESS                 PHONE
----------- -------------------- ----------------------- ------------
          1 Do Rights Plant Grower 2155 Horton Avenue Los A 805-525-2155
            s                    ngeles, CA 90025

          4 Calloway Nursery     2828 Calloway Drive Bake 661-588-7708
                                 rsfield, CA 93312
```

Select the suppliers that have supplied the most recent order.

```
select s.*

from    eb_supplier s, eb_order o

where   s.supplier_pk = o.supplier_fk and

        not exists (select o2.*

                        from eb_order o2

                        where o2.orderdate > o.orderdate

);
```

REPORT:

```
SUPPLIER_PK NAME                 ADDRESS                 PHONE
----------- -------------------- ----------------------- ------------
          4 Calloway Nursery     2828 Calloway Drive Bake 661-588-7708
                                 rsfield, CA 93312
```

Select the employee who has worked the most hours on a project.

```
select e.*
from    eb_employee e, eb_workson w
where   e.employee_pk = w.employee_fk and
        not exists ( select e2.*
                        from eb_employee e2, eb_workson w2
                        where e2.employee_pk = w2.employee_fk and
                                e2.employee_pk != e.employee_pk and
                                w2.hours > w.hours
                )
;
```

REPORT:

```
EMPLOYEE_PK FIRSTNAME       LASTNAME            PAYRATE
----------- --------------- --------------- ----------
          2 Antonia         Buentello             10.5
```

Select the clients that have a fall project.

```
select c.*
from    eb_client c, eb_project p
where   c.client_pk = p.for_fk and p.season = 'Fall'
;
```

REPORT:

CLIENT_PK	FIRSTNAME	LASTNAME	ADDRESS	RATE
4	Heather	Johns	8865 Aguila Road Bakersfield, CA 93307	12.5
1	Dwight	Byrum	2135 Newport Drive Bakersfield, CA 93307	13

Select the clients that will have a project in the summer.

```
select c.*
from    eb_client c, eb_project p
where   c.client_pk = p.for_fk and p.season = 'Summer'
;
```

REPORT:

CLIENT_PK	FIRSTNAME	LASTNAME	ADDRESS	RATE
1	Dwight	Byrum	2135 Newport Drive Bakersfield, CA 93307	13
2	Tracy	Burns	388 Alexis Avenue Bakersfield, CA 93308	12
3	Chris	Sanders	1190 Howley Drive Bakersfield, CA 93308	13

Select the project that requires the most Impatiens

```
select p.*
from    eb_project p, eb_usedon u, eb_plant pa
where   p.project_pk = u.project_fk and u.plant_fk = pa.plant_pk
and
        pa.name = 'Impatiens' and
        not exists (
                select p1.*
                from    eb_project p1, eb_usedon u2, eb_plant pa2
                where   p1.project_pk = u2.project_fk and
                        u2.plant_fk = pa2.plant_pk and
                        pa2.name = 'Impatiens' and
                        u2.howmany > u.howmany
                )
;
```

REPORT:

```
PROJECT_PK STARTDATE ENDDATE   SEASON    ESTPRICE DESCRIPTION    FOR_FK
---------- --------- --------- ------- ---------- -------------- ------
         5 16-OCT-10 17-OCT-10 Fall           110                    1
```

Select the plants that are red.

```
select p.*

from    eb_plant p, eb_color c

where   p.color_fk = c.color_pk and c.name = 'Red'

;
```

REPORT:

```
  PLANT_PK NAME                      QUANTITY   COLOR_FK

---------- ---------------------- ---------- ----------

         4 Zinnia                          5          1

         5 Poppies                         9          1
```

Select the plant that is most numerous at the moment.

```
select p1.name

from eb_plant p1

minus(

select p2.name

from eb_plant p2, eb_plant p3

where p2.quantity < p3.quantity

);
```

REPORT:

```
NAME

----------------------

Impatiens
```

Phase IV:
Stored Procedures

Edgar Buenrostro
CMPS 342
Fall 2010

Common Features in Oracle PL/SQL & Microsoft Trans-SQL

Oracle PL/SQL and Microsoft Transact SQL are two valuable tools that can be used to create stored subprograms and are extensions of SQL. PL/SQL is one of the more important languages needed for Oracle Databases. Its syntax resembles the structure of Ada with a declaration block a main body block and an exception block. By comparison Transact-SQL is also an extension of SQL that is instead used for Microsoft SQL Server. They follow a similar structure with a declaration block and a main body block. It also makes use of the regular flow control tools like if statements.

These languages are used to create stored procedures. These stored subprograms consist of procedures, functions, triggers and other similar objects. For the oracle database used in this project, we make use of PL/SQL for creating the stored procedures. They are able to accept arguments and they can access and manipulate the database. They can simply run queries or do more complex stuff. Procedures and functions are nearly identical, but functions are able to return a value. Triggers can also access tables in a database, but they are invoked automatically when something specific occurs to the database.

Oracle PL/SQL

As previously indicated, we will use PL/SQL for our stored procedures since we are using an Oracle database. All stored procedures using PL/SQL have a structure that divides the code into several blocks.

Declaration: This section is used to declare local variables and cursors

Execution: This part makes use of all the variables and performs the primary tasks of the subprograms

Exception: This section is meant to catch any exceptions that are thrown.

Layout:

```
CREATE OR REPLACE PROCEDURE procedureName(parametername IN datatype)
AS
    variablename variabletype := value;
    CURSOR c IS query;
BEGIN
    -- main body
EXCEPTION
    -- exception things
END procedureName;
```

Variable Types:

Stored procedures are able to use all of the regular datatypes that are supported in Oracle databases. They are also able to use cursors which can store the list of records returned by a query.

Cursor:

A cursor is a specific type used in stored procedures that allows us to name a query like a select statement and have the cursor point to where that sql statement is saved so we can later traverse the rows resulting from that query.

Control Statements:

PL/SQL can also make use of flow control statements like an if statement. It is used like any other programming languages and opens up opportunities to do many

things in our stored procedures. The general syntax of an if statement is as follows:

```
IF <condition>  THEN
        -- body
ELSE
        -- body
END IF;


LOOP
        -- body
END LOOP;


FOR I IN first..last LOOP
        -- body
END LOOP;
```

Exception Handling:

Exception handling is similar to other programming languages. If something in the program goes wrong and it throws an exception the exception block at the bottom is responsible for catching it and displaying the appropriate error.

```
EXCEPTION
        WHEN OTHERS THEN
        -- raise_application_error
```

Stored Procedures:

Stored procedures allow us to access and make use of the data in a database in order to perform more complex tasks that are inconvenient to perform on the SQL*PLUS command line. A procedure can also accept parameters.

```
CREATE OR REPLACE PROCEDURE ProcedureName(name IN datatype)
AS
    -- Declarations
BEGIN
    -- Body
EXCEPTION
    -- exception
END ProcedureName;
```

Stored Functions:

Functions work very similar to stored procedures, but in addition to manipulating the data in the database they can also return a value.

```
CREATE OR REPLACE FUNCTION funcName (name IN datatype) RETURN TYPE
IS
    -- Declarations
BEGIN
    -- Body
END;
```

Triggers:

Triggers are subprograms that are automatically invoked when a specific condition is met. They are ideal for keeping update logs that update automatically. For

example, you can write a trigger that activates when a specific attribute of a table is updated. At that point, the trigger can insert a record into a log table.

```
CREATE OR REPLACE TRIGGER triggerName
AFTER statement
FOR EACH ROW
BEGIN
    -- Body
END;
```

Oracle PL/SQL SubPrograms

Stored Procedures:

InsertClient

This procedure takes in five parameters that correspond to the fields of the eb_client table that will be inserted as a new row.

```
CREATE OR REPLACE PROCEDURE InsertClient(
    clientpk IN number,
    firstname IN varchar2,
    lastname IN varchar2,
    address IN varchar2,
    rate IN number)
    AS
    BEGIN
        insert into eb_client values(
            clientpk,
            firstname,
            lastname,
            address,
            rate);

    EXCEPTION
    WHEN OTHERS THEN
```

```
      ROLLBACK;
    raise_application_error( -40001, 'An error occurred in ' || SQLCODE ||
                    '-ERROR-' || SQLERRM );
    END InsertClient;
/
```

DeleteClient

This procedure takes in only one parameter. It is a number that will correspond to a primary key in the eb_client table. The procedure will declare a cursor that selects all the primary keys of the table and when it finds a record that matches the parameter, it'll delete that record.

```
CREATE OR REPLACE PROCEDURE DeleteClient(clientpk IN number)
    AS
        CURSOR c IS select client_pk from eb_client;
    BEGIN
        FOR rec IN c LOOP
        IF rec = clientpk
        delete from eb_client
        where client_pk = clientpk;
        END LOOP;
    EXCEPTION
    WHEN OTHERS THEN
      ROLLBACK;
    raise_application_error( -40001, 'An error occurred in ' || SQLCODE ||
                    '-ERROR-' || SQLERRM );
    END DeleteClient;
/
```

Stored Functions:

NavgHours

This is a function that takes in one number as a parameter and returns one number as well. The number it accepts is n. The function gets the hours found in the eb_workson table and lists them in descending order it then gets the top n number of hours and averages them. It returns that average.

```
CREATE OR REPLACE FUNCTION NAvgHours( n IN NUMBER ) RETURN
NUMBER
IS
	s number(9,2) := 0.0;
	p number(7,2);
	CURSOR c IS select hours from eb_workson ORDER BY hours DESC;
BEGIN
	open c;
	FOR i IN 1..n LOOP
		fetch c into p;
		s := s + p;
	END LOOP;
	close c;
	RETURN s/n;
EXCEPTION
	WHEN OTHERS THEN
	raise_application_error( -40001, 'An error occurred in ' || SQLCODE ||
				'-ERROR-' || SQLERRM );
END ;
/
```

Phase V:
GUI Design and Implementation

Edgar Buenrostro
CMPS 342
Fall 2010

Daily User Activities

Administrator

The main user that will make use of the interface created for this is the administrator, which can be the owner or one of the owners of the company. The interface will be used to pretty much keep information useful to the company organized. The user should have access to the information so he or she can edit it, add new information or delete records from any of the relations.

Relations, Views and Subprograms

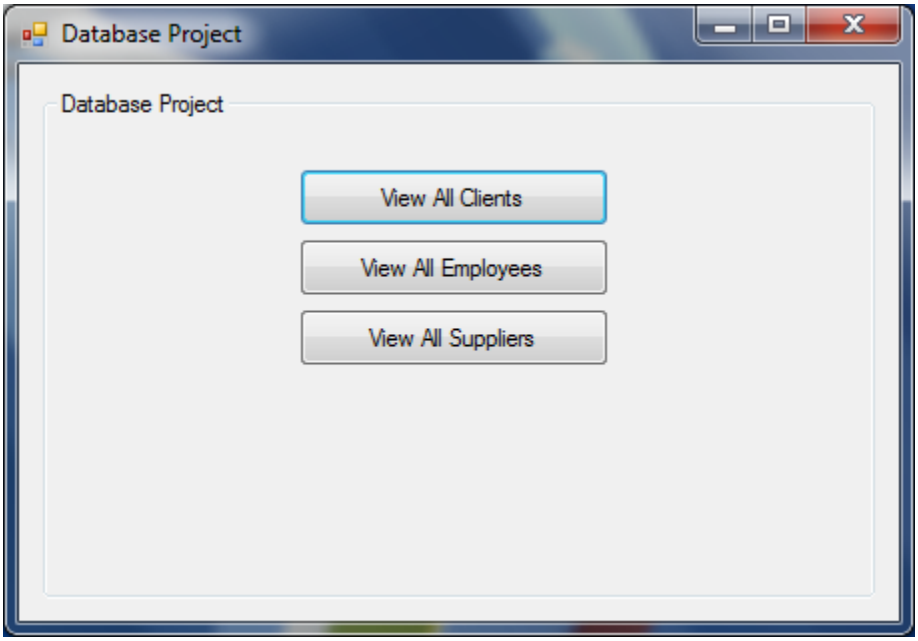
The interface makes use of most of the relations and relationships made for the database. If there are specific types of views that are needed for some of the interface then views can be created on sql*plus for those needs. The entities and relations used on this phase are as follows

- eb_client
- eb_color
- eb_contains
- eb_employee
- eb_logtable
- eb_order
- eb_plant
- eb_project
- eb_supplier
- eb_usedon
- eb_workson

Application Screenshots

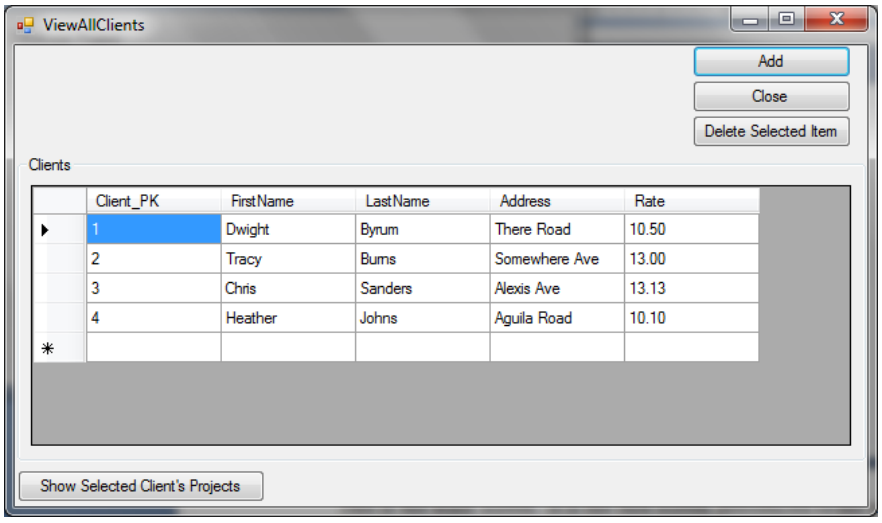
The following will be a few screenshots portraying some of the functionality for the interface created in this phase. It's primarily an admin interface, so it is meant to give you access to the information and being able to manipulate the information.

Main Screen

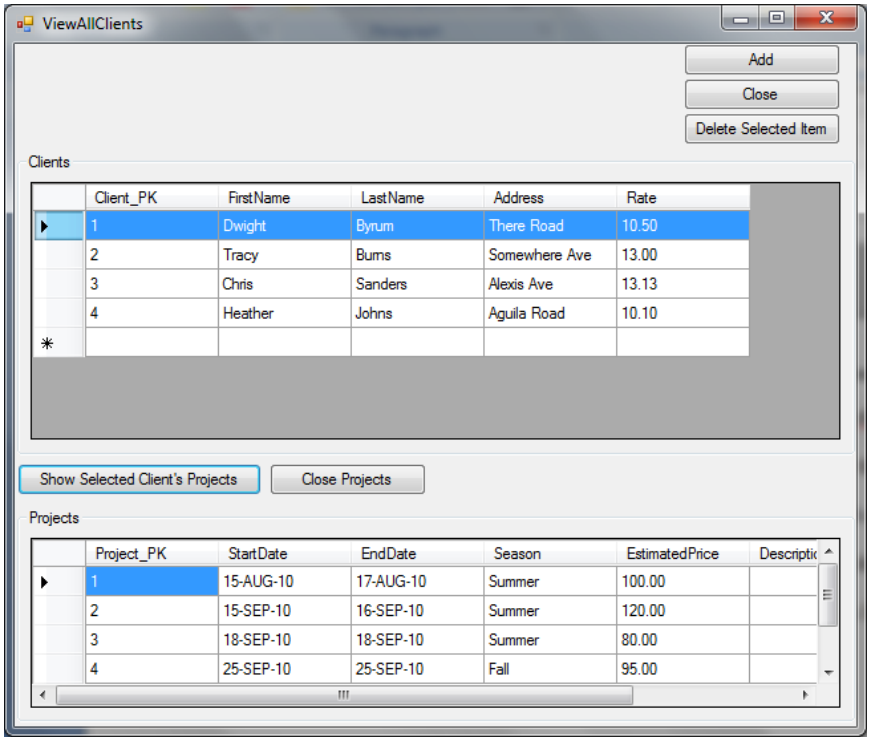


This is the main screen. It is the first screen introduced to the user. From here there are pretty much three simple options that the user can take. They take a look at all the clients, employees or suppliers. Those will take the user to their own screens where they can manipulate the data as they see fit.

Client Screen

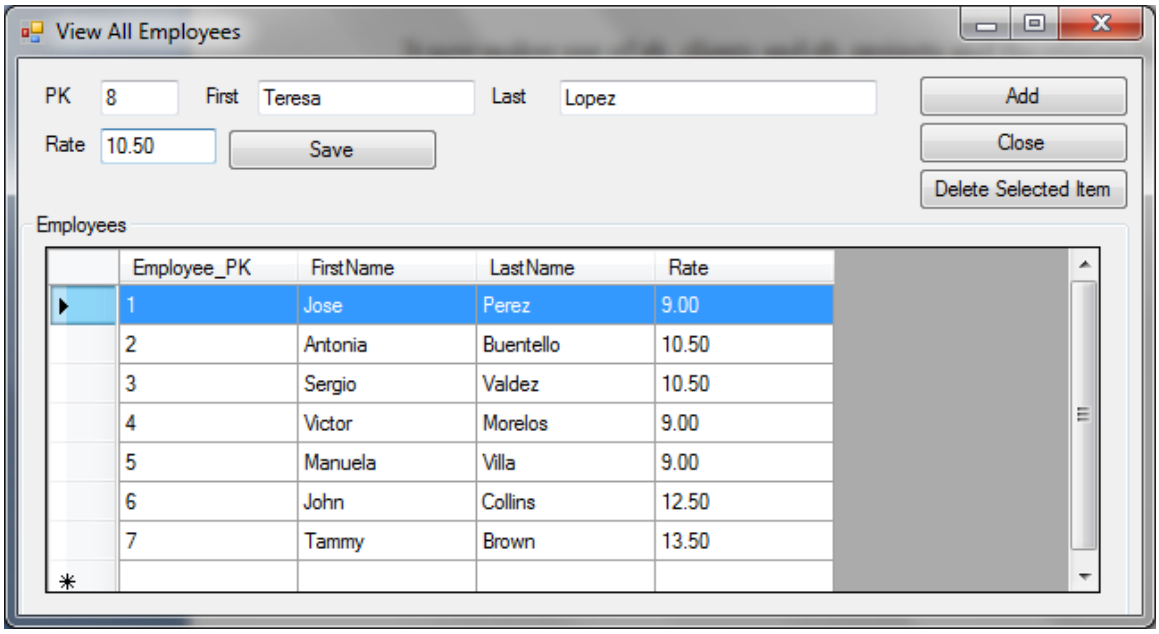


This is the client screen. The datagrid will get the information from the eb_client relation and display it as a list. Each client has several projects that are done for them by the gardening company. In order to find the projects that are associated with a specific client, the user should first click on the client in the datagrid. Then the user clicks on the "Show Selected Client's Projects" button and it expands the window so it looks like this:



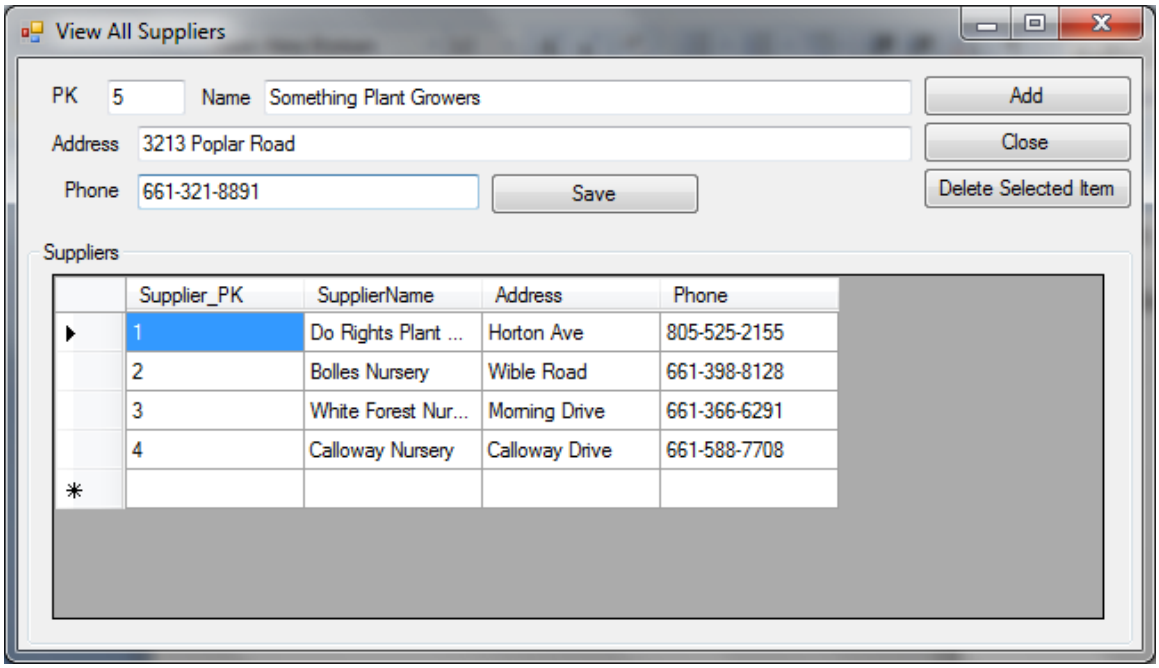
It now makes use of eb_clients and eb_projects and the relationship between that to match up the client's primary key to the project's foreign key.

Employee Screen



This is very similar to the other screen since it shows the information in the datagrid. This screenshot shows off how an employee is added. The user clicks on add and those fields appear where the information can be entered.

Supplier Screen



This functions in the exact same way as the other screens

Code Description and GUI Design

User Interface Design

The primary purpose of the interface design was to aim it towards an administrator or company owner. So the program should allow the user to have access to the information in all the relations. They should be able to view all the relations as well as group the in certain ways. Of course they should also be able to delete, insert or manipulate the data.

Data Access Descriptions

The data needs to be accessed so they can be displayed in each of the datagrids. All of this is done within a single class called DatabaseDB.cs where the data access is accomplished. Each datagrid needs to bind to a datatable to display information. It calls a method in the DatabaseDB class that returns a datatable to bind it to the datagrid. Each

method returns a specific datatable that contains the necessary information. The method in the DatabaseDB uses select statement to get information from relations or views.

Class Descriptions

- **DatabaseDB.cs** This class is used to access information from the database on Oracle. It has multiple methods that return datatables that are meant to bind to datagrids. The methods fill those datatables with information obtained with select statements
- **OCommand.cs** This also communicates with the database, but instead of getting information and returning datatables it is meant for insert and delete functions
- **Form1.cs** This class was the one in charge of the main screen. It was my first windows form class, so I left it with the default name. It is just in charge of those initial buttons.
- **ViewAllClients.cs** This is in charge of the clients screen that is in charge of the datagrids, adding and deleting. It also has to take into account the eb_project relation.
- **ViewAllEmployees.cs** This class is similar to the other class, but it is in charge of the employees screen.
- **ViewAllSuppliers.cs** is also similar to the previous classes in that it takes care of the screen that shows the suppliers and it interacts with the classes that can manipulate and access data from the database.

Development Process

Designing and Implementing the Application

The development process for this phase was extremely difficult. It is a completely different from any of the previous phases. It's a completely different type of task and it was a lot to learn for one phase. The hardest part was trying to figure out how to connect to the database. Actually accessing information from the database proved to be the most difficult. Inserting and deleting was a bit simpler, but using select statements to display the information in the datagrids was quite difficult.

Conclusion

This whole project has been quite a challenge, but it has been a valuable learning experience. I got a lot of understanding from the first four phases and how a database should be built and how to make use of queries, stored procedures and views. The difficult part was actually implementing it and putting it to use in a program. I think I still have a long way to go in that regard and I still have a lot to learn. The interface I created can still have new functionality added to it, and I'm glad I already have some familiarity from these phases, so I can continue to work on it and improve.