Landscape Company Database

Andrew Rokoszak
Professor Labouseur
Database Systems CMPT 308
April 26th, 2016

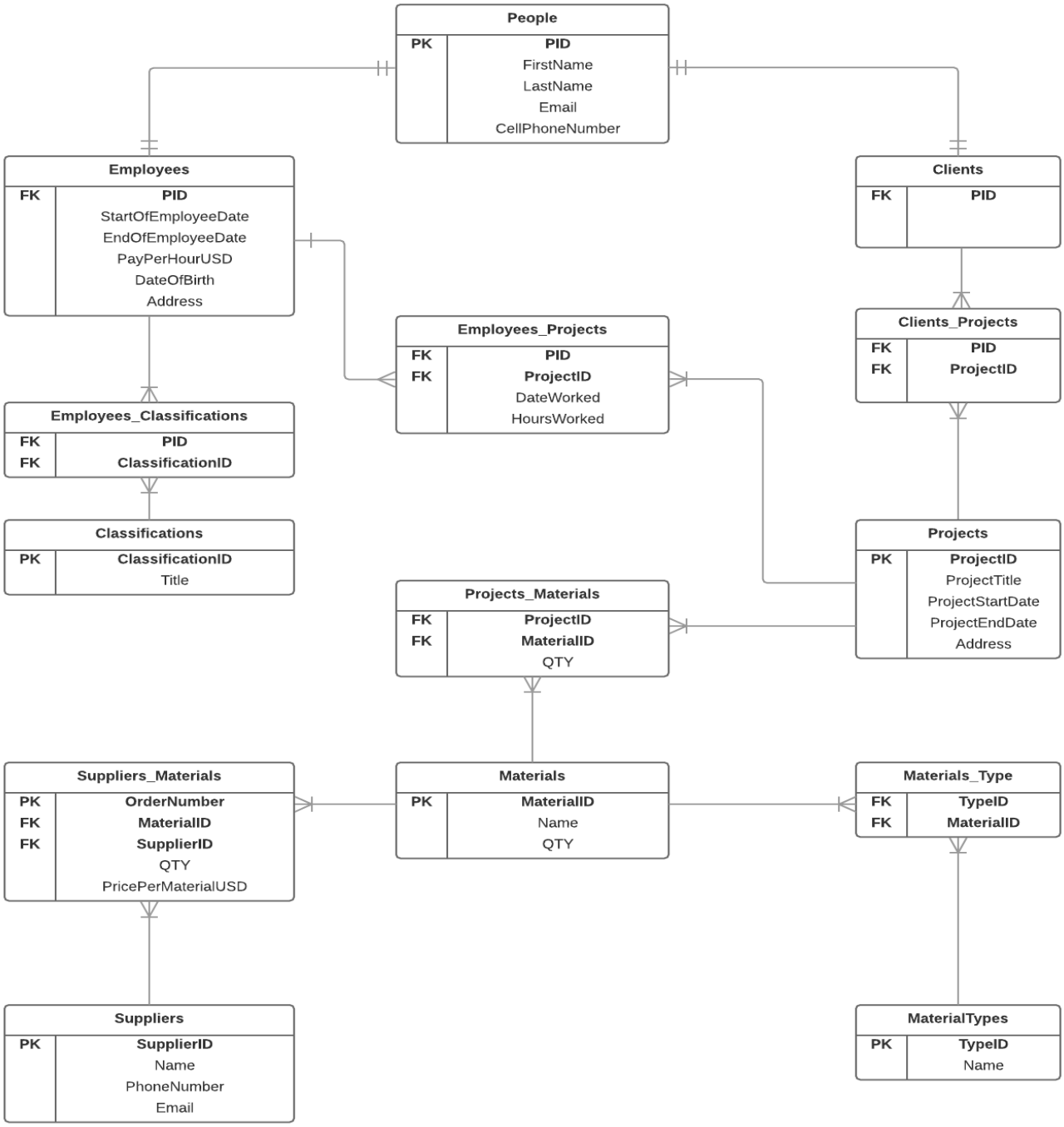
Table of Contents

Executive Summary 2
Entity Relationship Diagram 3
Tables 4
People 5
Clients 6
Employees 7
Classifications 8
Projects 9
Materials 10
MaterialTypes 11
Suppliers 12
Employees_Classifications 13
Clients_Projects 14
Employees_Projects 15
Materials_Type 16
Projects_Materials 17
Suppliers_Materials 18
Views **19**
Current_Employee 19
Present_Inventory 20
Reports **21**
Order History 21
Project History 22
Stored Procedure: Calculate Age **23**
Triggers **24**
Assess_Project 24
Assess_Inventory 25
Database Security **26**
Administrator, Assistant 26
Administrator, Assistant 26
Implementation Notes **29**
Known Problems **29**
Future Enhancements **29**

Executive Summary

The purpose of this database is to assist The Jacobsen Landscape Design Company (JLD) in the management of their employees, clients, materials and projects. Prior to the implementation of this system, the company has utilized a variety of excel spreadsheets to manage their information, but have found it to be both inconsistent and unreliable. This comprehensive database will provide the company with the ability to more easily manage the relationships between the firm and its employees, as well as the logistics of material deliveries to meet the project deadlines of different clients. The database has three security roles to help ensure the personal information of employees and clients is protected. Additionally there are a number of triggers that help to manage projects in an efficient manner and regulate the inventory levels of the company.

Entity Relationship Diagram



Jacobsen

Table: People

The purpose of this table is to store information related to all of the Clients and Employees. It does not specify whether or not an individual is an employee or client as their respective tables determine that.

```
DROP TABLE IF EXISTS          People cascade;
CREATE TABLE                  People
(
PID                     Serial                  Not Null,
FirstName               VARCHAR(255)            Not Null,
LastName                VARCHAR(255)            Not Null,
Email                   VARCHAR(255)            Not Null,
CellPhoneNumber         Char(12)                Not Null,
Primary Key(PID)
);
```

	pid integer	firstname character varying(255)	lastname character varying(255)	email character varying(255)	cellphonenumber character(12)
1	1	Teresa	Dean	tdean0@wisc.edu	366-744-3501
2	2	Alice	Gardner	agardner1@ning.com	627-977-2274
3	3	Craig	Stephens	cstephens2@bandcamp.com	559-578-4557
4	4	Lori	Smith	lsmith3@yahoo.co.jp	841-906-2122
5	5	Tammy	Nguyen	tnguyen4@nps.gov	389-993-7557
6	6	Evelyn	Hughes	ehughes5@sohu.com	280-744-4039
7	7	Henry	Franklin	hfranklin6@army.mil	135-868-0712
8	8	Justin	Stone	jstone7@mlb.com	352-592-6576
9	9	Victor	Stewart	vstewart8@sun.com	547-640-0640

Functional Dependencies

PID ➔ (FirstName, LastName, Email, CellPhoneNumber)

Table Clients

The purpose of this table is to store a PID for each client in the system. This table references the Peoples table to determine who is a client.

```
DROP TABLE IF EXISTS    Clients cascade;
CREATE TABLE            Clients
(
PID                     INT NOT NULL references People (PID),
Primary Key(PID)
);
```

	pid integer
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

Functional Dependencies

There are no functional dependencies for this table.

Table: Employees

This table retains information regarding current and former employees of the firm. This allows the company to manage their employees in a more effective manner.

```
DROP TABLE IF EXISTS          Employees cascade;
CREATE TABLE                  Employees
(
PID                     INT                     Not Null,
StartOfEmployeeDate     Date                    Not Null,
EndOFEmployeeDate       Date                            ,
PayPerHourUSD           Money                   Not Null,
DateOfBirth             Date                    Not Null,
Address                 VARCHAR(255)            Not Null,
Primary Key(PID)
);
```

	pid integer	startofemployeedate date	endofemployeedate date	payperhourusd money	dateofbirth date	address character varying(255)
1	11	2002-10-13		$22.32	1980-06-14	0476 Caliangt Parkway
2	12	2004-09-08	2015-06-29	$21.28	1982-03-23	8 Lakeland Parkway
3	13	2004-11-06		$45.08	1975-04-08	5785 Aberg Way
4	14	2008-08-07	2016-04-13	$49.30	1964-05-13	17356 Charing Cross Pass
5	15	2006-01-19		$25.44	1972-02-16	46308 Susan Center
6	16	2004-02-01	2016-02-07	$28.26	1963-12-25	60004 Village Green Junction
7	17	2001-08-25	2009-06-21	$17.92	1976-06-28	1587 Weeping Birch Place
8	18	2010-06-30		$41.41	1988-12-11	2998 Badeau Junction
9	19	2008-06-05	2015-11-22	$41.41	1984-11-01	19 Coleman Parkway
10	20	2000-02-24		$45.72	1980-03-03	1 West Place
11	21	2003-10-24		$27.92	1972-10-12	325 Iowa Junction
12	22	2011-05-14	2015-08-21	$31.21	1986-12-24	81714 Cody Place
13	23	2007-11-11	2015-04-10	$43.50	1962-11-13	25 Golf Course Terrace
14	24	2009-05-02	2015-01-10	$15.89	1963-09-15	0345 Pankratz Circle
15	25	2000-07-17	2014-07-08	$41.09	1964-06-25	995 Jenifer Trail

Functional Dependencies

PID ➔ (StartOfEmployeeDate, EndOfEmployeeDate, PayPerHourUSD, DateOfBirth, Address)

Table: Classifications

This table allows the business to store different job types that employees can potentially be. In other words, this table stores different job titles that exist at the company.

```
DROP TABLE IF EXISTS            Classifications cascade;
CREATE TABLE                    Classifications
(
ClassificationID        Serial                  Not Null,
Title                   VARCHAR(25)             Not Null,
Primary Key(ClassificationID)
);
```

1	1	Landscaper
2	2	Technician
3	3	Project Manager
4	4	Foreman
5	5	Office Manager
6	6	Office Worker

Functional Dependencies

PID ➔ (Title)

Table: Projects

This table helps to store data about existing and finished projects.

```
DROP TABLE IF EXISTS          Projects cascade;
CREATE TABLE                  Projects
(
ProjectID               Serial                  Not Null,
ProjectTitle            VARCHAR(255)            Not Null,
ProjectStartDate        Date                    Not Null,
ProjectEndDate          Date                    Not Null,
Address                 VARCHAR(255)            Not Null,
Primary Key(ProjectID)
);
```

	projectid integer	projecttitle character varying(255)	projectstartdate date	projectenddate date	address character varying(255)
1	1	Smith Residence	2003-03-04	2015-03-14	040 Rieder Place
2	2	City Park	2013-11-22	2015-02-19	794 Oxford Way
3	3	Tropical Transformation	2007-09-20	2008-10-09	50540 5th Center
4	4	Bond Estate	2003-04-26	2014-08-04	9 Cardinal Parkway
5	5	Schneider Residence	2011-03-18	2015-07-30	8950 Paget Way
6	6	Dominos Pizza	2012-07-29	2015-03-14	89 Pepper Wood Court
7	7	Town Municipal Building	2006-09-10	2016-01-10	07744 Pine View Point
8	8	Jones Residence	2011-09-26	2015-11-05	7139 Spohn Avenue

Functional Dependencies

PID ➔ (ProjectTitle, ProjectStartDate, ProjectEndDate, Address)

Table: Materials

This table stores information regarding the names of the various materials used at the business. It is also used to calculate the Present_Inventory View

```
DROP TABLE IF EXISTS          Materials cascade;
CREATE TABLE                  Materials
(
MaterialID              Serial                  Not Null,
Name                    VARCHAR(255)            Not Null,
QTY                     INT                     Not Null,
Primary Key(MaterialID)
);
```

	materialid integer	name character varying(255)	qty integer
1	1	Brown Mulch	326
2	2	Red Mulch	320
3	3	Gravel	303
4	4	River Stone	479
5	5	Dark Wood Chips	400
6	6	Light Wood Chips	473
7	7	Regular Soil	184
8	8	Premium Soil	319
9	9	Seed	466
10	10	Fertilizer	283

Functional Dependencies

MaterialID ➔ (Name, QTY)

Table: MaterialTypes

This table stores the names of the differing types of materials used. This allows users to view all of the selections in specific categories rather than all of the materials simultaneously.

```
DROP TABLE IF EXISTS          MaterialTypes cascade;
CREATE TABLE                  MaterialTypes
(
TypeID                Serial                Not Null,
Name                  VARCHAR(255)          Not Null,
Primary Key(TypeID)
);
```

	typeid integer	name character varying(255)
1	1	Mulch
2	2	Stone
3	3	Wood Chips
4	4	Lawn Care Products

Functional Dependencies

TypeID ➔ (Name)

Table: Suppliers

The suppliers table stores information about suppliers including all contact information.

```
DROP TABLE IF EXISTS            Suppliers cascade;
CREATE TABLE                    Suppliers
(
SupplierID              Serial                  Not Null,
Name                    VARCHAR(255)            Not Null,
PhoneNumber             Char(12)                Not Null,
Email                   VARCHAR(255)            Not Null,
Primary Key(SupplierID)
);
```

	supplierid integer	name character varying(255)	phonenumber character(12)	email character varying(255)
1	1	Lowes	498-719-3996	ldixon0@lowes.com
2	2	Home Depot	739-143-2477	bwilson1@homedepot.com
3	3	Griffin Stoneworks	677-493-8794	cpalmer2@gmail.com
4	4	Bobs Woodchips	437-710-2904	bob.woodchips@gmail.com

Functional Dependencies

SupplierID ➔ (Name, PhoneNumber, Email)

Table: Employees_Classifications

The purpose of this table is to connect both the Employees and Classifications tables. This enables employees to receive specific job titles.

```
DROP TABLE IF EXISTS            Employees_Classifications cascade;
CREATE TABLE                    Employees_Classifications
(
PID                     INT                     NOT NULL        references Employees(PID),
ClassificationID        INT                     NOT NULL        references Classifications(
ClassificationID),
Primary Key(PID, ClassificationID)
);
```

	pid integer	classificationid integer
1	11	1
2	12	1
3	13	1
4	14	1
5	15	5
6	16	6
7	17	3
8	18	3
9	19	4
10	20	2

Functional Dependencies

PID ➔ (ClassificationID)

Table: Clients_Projects

The purpose of this table is to establish a relationship between specific projects and the respective client.

```
DROP TABLE IF EXISTS          Clients_Projects cascade;
CREATE TABLE                  Clients_Projects
(
PID                   INT                   NOT NULL       references Clients(PID),
ProjectID             INT                   NOT NULL       references Projects(ProjectID),
Primary Key(PID, ProjectID)
);
```

	pid integer	projectid integer
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8

Functional Dependencies

(PID, ProjectID) ➔

Table: Employees_Projects

This table is used to assign specific employees to the various projects going on at the company. This table assists in tracking hours and days worked for employees.

```
DROP TABLE IF EXISTS          Employees_Projects cascade;
CREATE TABLE                  Employees_Projects
(
PID                   INT                   NOT NULL       references Employees(PID),
ProjectID             INT                   NOT NULL       references Projects(ProjectID),
DateWorked            Date                  Not Null,
HoursWorked           INT                   Not Null,
Primary Key(PID, ProjectID, DateWorked)
);
```

	pid integer	projectid integer
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8

Functional Dependencies

(PID, ProjectID, DateWorked) ➔ HoursWorked

Table: Materials_Type

This table connects the Materials and MaterialTypes tables. This helps to categorize materials into various types like mulch, woodchips or stone.

```
DROP TABLE IF EXISTS            Materials_Type cascade;
CREATE TABLE                    Materials_Type
(
MaterialID              INT                     NOT NULL        references Materials(MaterialID),
TypeID                  INT                     NOT NULL        references MaterialTypes(TypeID),
Primary Key(MaterialID, TypeID)
);
```

	materialid integer	typeid integer
1	1	1
2	2	1
3	3	2
4	4	2
5	5	3
6	6	3
7	7	4
8	8	4
9	9	4
10	10	4

Functional Dependencies

(MaterialID, TypeID) ➔

Table: Projects_Materials

This table serves to connect the Projects and Materials Tables. This table is used to calculate the Present_Inventory (current inventory) as well as to show which materials have been used at specific projects.

```
DROP TABLE IF EXISTS          Projects_Materials cascade;
CREATE TABLE                  Projects_Materials
(
ProjectID             INT                   NOT NULL       references Projects(ProjectID),
MaterialID            INT                   NOT NULL       references Materials(MaterialID),
QTY                   INT                   NOT NULL,
Primary Key(ProjectID, MaterialID)
);
```

	projectid integer	materialid integer	qty integer
1	1	1	30
2	1	8	10
3	1	9	25
4	2	10	40
5	2	9	30
6	3	3	20
7	3	4	50
8	4	5	10
9	4	6	15
10	5	7	20

(ProjectID, MaterialID) ➔ QTY

Table: Suppliers_Materials

The purpose of this table is to connect both the Suppliers and Materials Tables. This helps to track orders coming from various suppliers.

```
DROP TABLE IF EXISTS          Suppliers_Materials cascade;
CREATE TABLE                  Suppliers_Materials
(
OrderNumber             Serial                  NOT NULL,
MaterialID              INT                     NOT NULL references Materials(MaterialID),
SupplierID              INT                     NOT NULL references Suppliers(SupplierID),
QTY                     INT                     NOT NULL,
PricePerMaterialUSD     Money                   NOT NULL,
Primary Key(OrderNumber)
);
```

	ordernumber integer	materialid integer	supplierid integer	qty integer	pricepermaterialusd money
1	1	1	1	50	$4.00
2	2	2	1	50	$5.00
3	3	3	3	50	$10.00
4	4	4	3	50	$12.00
5	5	5	4	50	$3.00
6	6	6	4	50	$4.00
7	7	7	2	50	$3.50
8	8	8	2	50	$5.00
9	9	9	2	50	$7.00
10	10	10	2	50	$9.00

Functional Dependencies

OrderNumber ➔ (MaterialID, SupplierID, QTY, PricePerMaterialUSD)

View: Current_Employee

The purpose of this view enables the company to see current workers and also view other information about them.

```
CREATE VIEW Current_Employee AS
Select FirstName, LastName, Title, PayPerHourUSD
From Employees_Classifications, Employees, People, Classifications
Where People.PID = Employees.PID
AND Employees.PID = Employees_Classifications.PID
AND Classifications.ClassificationID = Employees_Classifications.ClassificationID
AND Employees.EndOfEmployeeDate is NULL
--
```

	firstname character varying(255)	lastname character varying(255)	title character varying(25)	payperhourusd money
1	Heather	Wheeler	Landscaper	$22.32
2	Pamela	Harper	Landscaper	$45.08
3	Sara	Gilbert	Office Manager	$25.44
4	Michelle	Ruiz	Project Manager	$41.41
5	Lois	Nelson	Technician	$45.72
6	Kelly	Cunningham	Technician	$27.92

View: Present_Inventory

This view displays the current inventory records of materials that are in stock. This includes the MaterialID, Name and Quantity.

```
CREATE VIEW Present_Inventory AS
Select Materials.MaterialID as TEST, Materials.Name, Materials.QTY - a.ProjectQTY + b.SupplierQTY as QTY
FROM Materials, (SELECT Projects_Materials.MaterialID, SUM (projects_Materials.QTY)
as ProjectQTY FROM Projects_Materials GROUP BY Projects_Materials.MaterialID) as a,
(SELECT Suppliers_Materials.MaterialID , SUM (Suppliers_Materials.QTY) as SupplierQTY FROM
Suppliers_Materials
Group By Suppliers_Materials.MaterialID) as b
WHERE a.MaterialID = Materials.MaterialID
GROUP BY TEST, a.ProjectQTY , b.SupplierQTY;
```

	test integer	name character varying(255)	qty bigint
1	1	Brown Mulch	306
2	2	Red Mulch	358
3	3	Gravel	308
4	4	River Stone	459
5	5	Dark Wood Chips	440
6	6	Light Wood Chips	508
7	7	Regular Soil	214
8	8	Premium Soil	309
9	9	Seed	401
10	10	Fertilizer	293

Report: Order Record

This report displays the order records of the company. It includes the order number, material and cost.

```
SELECT Suppliers_Materials.OrderNumber, Materials.Name, Suppliers.Name, Suppliers_Materials.QTY,
PricePerMaterialUSD, Suppliers_Materials.QTY * PricePerMaterialUSD AS OrderTotal
From Materials, Suppliers, Suppliers_Materials
Where Materials.MaterialID = Suppliers_Materials.MaterialID
AND Suppliers.SupplierID = Suppliers_Materials.SupplierID
ORDER BY OrderNumber ASC
```

	ordernumber integer	name character varying(255)	name character varying(255)	qty integer	pricepermaterialusd money	ordertotal money
1	1	Brown Mulch	Lowes	50	$4.00	$200.00
2	2	Red Mulch	Lowes	50	$5.00	$250.00
3	3	Gravel	Griffin Stoneworks	50	$10.00	$500.00
4	4	River Stone	Griffin Stoneworks	50	$12.00	$600.00
5	5	Dark Wood Chips	Bobs Woodchips	50	$3.00	$150.00
6	6	Light Wood Chips	Bobs Woodchips	50	$4.00	$200.00
7	7	Regular Soil	Home Depot	50	$3.50	$175.00
8	8	Premium Soil	Home Depot	50	$5.00	$250.00
9	9	Seed	Home Depot	50	$7.00	$350.00
10	10	Fertilizer	Home Depot	50	$9.00	$450.00

Report: Project History

The project history report displays the kinds of materials and quantity used at a given project site.

```
SELECT ProjectTitle, Materials.Name, Projects_Materials.QTY
FROM Projects_Materials, Materials, Projects
WHERE Materials.MaterialID = Projects_Materials.MaterialID
AND Projects.ProjectID = Projects_Materials.ProjectID
```

	projecttitle character varying(255)	name character varying(255)	qty integer
1	Smith Residence	Brown Mulch	30
2	Smith Residence	Premium Soil	10
3	Smith Residence	Seed	25
4	City Park	Fertilizer	40
5	City Park	Seed	30
6	Tropical Transformation	Gravel	20
7	Tropical Transformation	River Stone	50
8	Bond Estate	Dark Wood Chips	10
9	Bond Estate	Light Wood Chips	15
10	Schneider Residence	Regular Soil	20

Stored Procedure: Age Calculation

The purpose of this stored procedure is to calculate the age of a given Current_Employee.

```
CREATE OR REPLACE FUNCTION CALCULATE_AGE(EmployeeID INTEGER) RETURNS Integer AS $$
DECLARE
        Age Integer:= (SELECT EXTRACT (YEAR FROM AGE(CURRENT_DATE,(SELECT DateOfBirth From Employees
where
        Employees.pid = employeeID))));
BEGIN
        Return age;
END;
$$ LANGUAGE plpgsql;
```

Trigger: Assess_Project

This trigger prevents anyone from entering a record into the Employees_Projects table for a project that has already been completed. The trigger will post an exception if a Project has been completed.

```
CREATE OR REPLACE FUNCTION Assess_Project() RETURNS TRIGGER AS $Assess_Project$
  BEGIN
    IF
      (SELECT ProjectEndDate
      FROM Projects
      WHERE ProjectID = NEW.ProjectID)
      IS NOT NULL THEN
        Raise Exception 'You cannot enter an employee into a project that has already ended';
      END IF;
    RETURN VIEW;
  END;
$Assess_Project$ LANGUAGE plpgsql;

CREATE TRIGGER Assess_Project
        BEFORE INSERT ON Employees_Projects
        FOR EACH ROW
        EXECUTE PROCEDURE Assess_Project();
```

Trigger: Assess_Inventory

The purpose of this trigger is to determine if inventory levels have fallen below a set quantity. In this case anytime inventory of a specific material falls below 10, a trigger will alert users that there is “Not enough stock in inventory.”

```
CREATE OR REPLACE FUNCTION Assess_Inventory() RETURNS TRIGGER AS $Assess_Inventory$
  DECLARE
        A INTEGER := (SELECT QTY FROM CURRENT_INVENTORY WHERE ID IN (SELECT Materials.MaterialID From
Materials WHERE Materials.MaterialID = New.MaterialID));
        B INTEGER := New.QTY;
        C INTEGER := A - B;
  BEGIN
        IF (C < 5) THEN
                        RAISE EXCEPTION 'Not enough stock in Inventory';
                END IF;
        RETURN NEW;
        END;
$Assess_Inventory$ LANGUAGE plpgsql;

CREATE TRIGGER Assess_Inventory
BEFORE INSERT ON Projects_Materials
FOR EACH ROW EXECUTE PROCEDURE Assess_Inventory();
```

Database Security:

Security is one of the most important considerations in the design for a database that will be used by an wide range of people. There are a number of specific security roles that will act to manage the database including an administrator and a number of assistants. Additionally measures have been taken to ensure various employees cannot manipulate information within the database without administrative approval.

Administrator: The administrator role has the greatest access level to the database. This person has complete control over the system and is not limited access to any areas.

```
CREATE ROLE DBadmin;
GRANT SELECT ON ALL TABLES
IN SCHEMA PUBLIC
TO DBadmin;
```

Assistants: The assistant role has nearly an equal amount of access to the administrator. This role requires the assistant to input and update new information, while being restricted from accessing and deleting specific tables

```
Create Role Assistant
REVOKE DELETE ON Suppliers_Materials          FROM Assistant;
REVOKE DELETE ON Employees_Projects           FROM Assistant;
REVOKE DELETE ON Projects                     FROM Assistant;
```

Employee Security Measures:

Employees are equally responsible for maintaining accurate and useable data for the database. The following example shows the limit of access given to even a project manager within the firm.

Project Manager:

The project manager has very little access to manipulate or change data that exists within the system. They do, however, have access to view more tables within the database.

```
Create Role Project_Manager;
REVOKE ALL PRIVILEGES ON People                          FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Clients                         FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Employees                       FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Classifications                 FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Projects                        FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Materials                       FROM Project_Manager;
REVOKE ALL PRIVILEGES ON MaterialTypes                   FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Suppliers                       FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Employees_Classifications       FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Clients_Projects                FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Employees_Projects              FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Materials_Type                  FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Projects_Materials              FROM Project_Manager;
REVOKE ALL PRIVILEGES ON Suppliers_Materials             FROM Project_Manager;
```

Implementation Notes:

1. In order to best use the Project History Report, it is imperative that the user manually enters the projectID into the query. Without doing so it will not be possible to view the appropriate report.

2. When entering any kind of Phone Number into the system it must be entered in the format of XXX-XXX-XXXX. This will ensure consistency within the system.

Known Problems:

1. One of the problems with the system is that the Projects_Materials Table does not record a PricePerMaterialUSD, which means that it must be calculated outside of the database.

2. Another issue with this database is that one employee can be working on many different projects at once, however this cannot be displayed without inputting an insert statement for each project and each employee.

3. A date can be entered in the Employees_Projects table that does not fall between the projects start and end date. This is an issue of data inconsistency and integrity that must be addressed.

Future Enhancements:

Jacobson Landscape Design is also in the business of construction. This includes renting machines to clients and basic construction projects. Since this is a newer portion of their business, tables should eventually account for the rented equipment (including cost, time rented, and required return dates).

Jacobson Landscape Design has also recently purchased a second building to expand their business. It will be imperative for both sites to have access to the database and that the inventory of both buildings is accounted for.

Lastly, the company is looking to wholesale some additional products by hosting a website for clients. Creating an interface for clients and prospective customers to access the database will allow for increased sales for the company. It is imperative though that the proper security measures are taken once the site is up and running.