DATABASE DESIGN PROJECT

A Business Relationship Model

Internally from People to Product

By: Gerard Taliercio

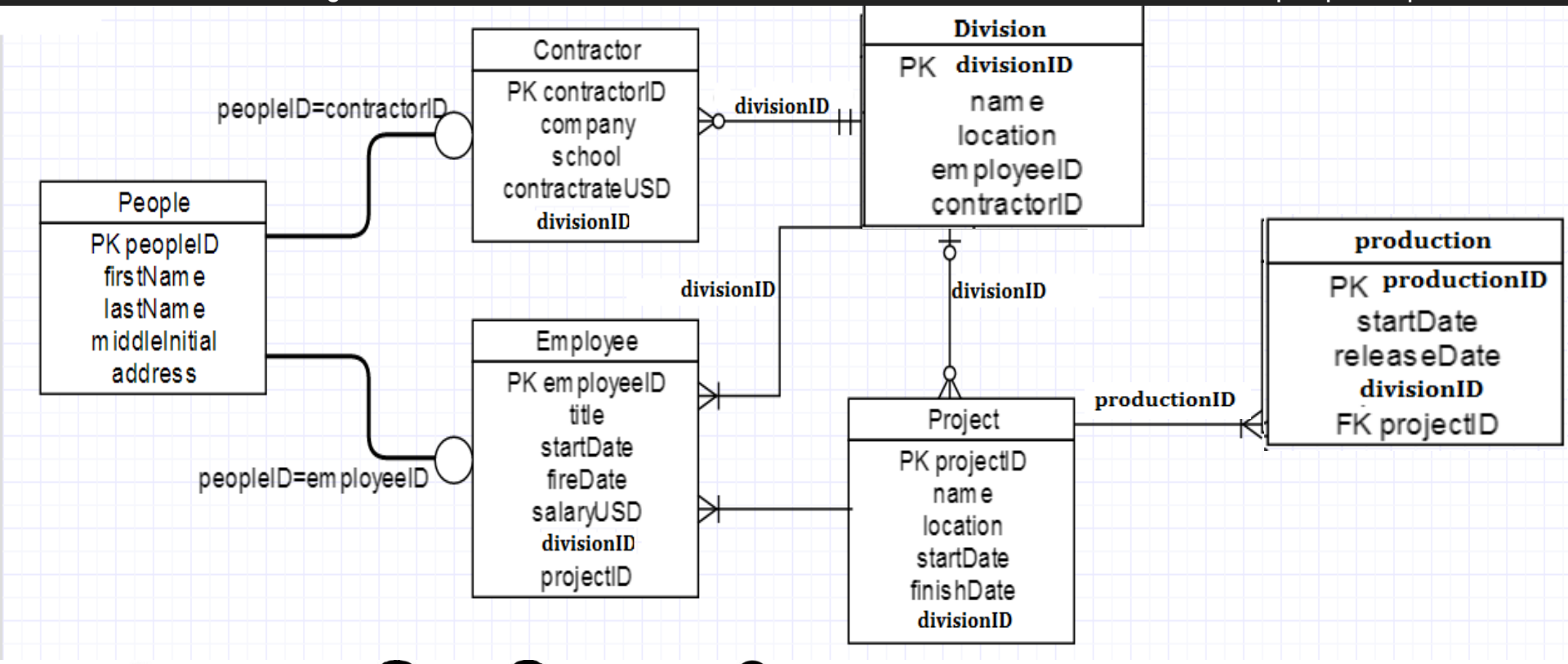
TABLE OF CONTENTS

EXECUTIVE SUMMARY

- This document details the database architecture and specification for a business. It will show the hierarchy structure and relational model between employees, their departments, and products being produced..
- The first part will detail the layout and relationships between all areas of a company and how they work in harmony.
- The second part will examine each area of the business and detail all the members of each object. A series of snapshots and SQL statements will be included for reference.
- The third part will demonstrate the capabilities of the business model through many joins, sub-query, having clause, security, grant, revoke, etc.

ENTITY RELATIONSHIP DIAGRAM

- Below is an ER diagram that details the relations between etities in a business model from people to product.



THE *PEOPLE* TABLE

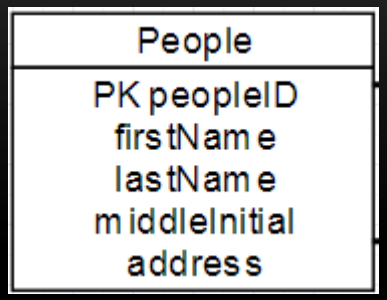
- The *people* table uses peopleID as its primary key
- Functional Dependencies: peopleID → firstName, lastName, middleInitial, address
- *People* shows every person who works or has worked for the business. It details their full name and the where they reside.

Create Statement

```
---------------
-- People --
---------------
drop table if exists people;
create table people (
        peopleID        char (10) not null,
        firstName       varchar (20),
        lastName        varchar (20),
        middleInitial   varchar (1),
        address         varchar (20),
primary key(peopleID)
);
```

THE *PEOPLE* TABLE CONTINUED

- Here is the *people* object from the ER diagram along with sample data from the created.



Sample Data

Data Output | Explain | Messages | History

	peopleid character(10)	firstname character varying(20)	lastname character varying(20)	middleinitial character varying(1)	address character varying(20)
1	002	Bill	Fairbanks	B	Beirut
2	003	Jack	Mason	B	Siberia
3	004	Aiden	Flemmings	B	Beirut
4	005	Stuart	Thomas	B	Greece
5	006	Alec	Trevelyan	B	Russia
6	007	James	Bond	B	Classified
7	010	Le	Chiffe	A	France
8	011	Mr.	Big	R	Russia
9	012	Sir	Hugo	D	London
10	013	Jack	Spang	F	Las Vegas
11	014	Rosa	Klebb	R	Russia
12	015	Emilio	Largo	N	Beirut

THE *CONTRACTORS* TABLE

- The *contractors* table uses contractorID for the primary key.
- Functional Dependencies: contractorID → company, school , contractrateUSD
- *Contractors* categorizes which person from *people* is a contracted employee. A person is outsourced from another company and their educational background is stored for reference and proper placement. The annual rate (USD) to which they are paid is also within this table.
- *divisionID* is a reference to which division the contracted employee has been placed.

Contractor
PK contractorID company school contractrateUSD **divisionID**

CONTRACTORS TABLE CONTINUED

- Here is the create statement for *contractors and the sample data created from it.*

Create Statement

```
----------------
-- Contractors --
----------------

drop table if exists contractors;
create table contractors (
        contractorID        char(10) not null,
        company             varchar (30),
        school              varchar (30),
        contractrateUSD     numeric (12,2),
        divisionID          char (10) not null,
primary key (contractorID),
foreign key (divisionID) references division (divisionID)
);
```

CONTRACTOR TABLE CONTINUED

Sample Data

Data Output | Explain | Messages | History

	contra(charact	company character varying(30)	school character varying(30)	contractrateusd numeric(12,2)	divisionid character(10)
1	1	Apple	University of Stam	44000.00	06
2	2	Microsoft	Columbia Universit	56000.00	05
3	3	Oracle	ITT Tech	78000.00	04
4	4	Goldman Sachs	NYU	65000.00	03
5	5	Morgan Stanley	Vassar College	55000.00	02
6	6	NYSE	University of Alba	80000.00	01

THE *EMPLOYEES* TABLE

- This table uses employeeID for the primary key.
- Functional Dependencies: employeeID → title, startDate, fireDate, salaryUSD
- The employee table also references to the division the employee is working in as well as the project they are working on by using the foreign keys: divisionID and projectID

Employee
PK employeeID title startDate fireDate salaryUSD **divisionID** projectID

EMPLOYEE TABLE CONTINUED

Create Statement with Check Constraint

```
--------------
-- Employee --
--------------

drop table if exists employees;
create table employees (
        employeeID          char(10) not null,
        title               varchar (26),
        startDate           date not null,
        fireDate            date,
        salaryUSD           numeric (12,2) CHECK(salaryUSD > 0),
        divisionID          char (10) not null,
        projectID           char (10) not null,
primary key (employeeID),
foreign key (divisionID) references division (divisionID),
foreign key (projectID)  references projects (projectID)
);
```

EMPLOYEE TABLE CONTINUED

Sample Data

Data Output | Explain | Messages | History

	employeeid character(10	title character varying(26)	startdate date	firedate date	salaryusd numeric(12,2)	divisionid characte	projectid character(10)
1	002	Software Engineer	2012-06-12	2013-12-31	95000.00	01	11
2	003	Systems Engineer	2011-04-14	2013-09-23	55000.00	02	33
3	004	Software Developer	2010-03-15	2013-12-31	118000.00	03	22
4	005	Java Developer	1999-01-01	2013-10-31	68000.00	01	11
5	006	Business Analyst	2012-06-12	2013-12-31	75000.00	02	33
6	007	projects Manager	1993-04-28	2013-12-31	89000.00	05	66

THE *DIVISION* TABLE

- This table uses divisionID as its primary key
- Functional Dependencies: divisionID → name, location
- This table utilizes the foreign keys: employeeID and contractorID to display which employee and or contractor works.

Division	
PK	divisionID
	name
	location
	employeeID
	contractorID

DIVISION TABLE CONTINUED

Create Statement

```
-----------------
-- division    --
-----------------

drop table if exists division;
create table division (
        divisionID          char(10) not null,
        name                varchar (26),
        location            varchar (30),
        employeeID          char (10) not null,
        contractorID        char (10) not null,
primary key (divisionID),
foreign key (employeeID)  references employees   (employeeID),
foreign key (contractorID)references contractors  (contractorID)
);
```

DIVISION TABLE CONTINUED

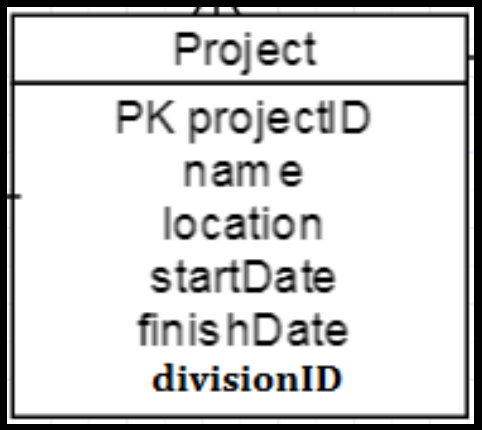
Sample Data

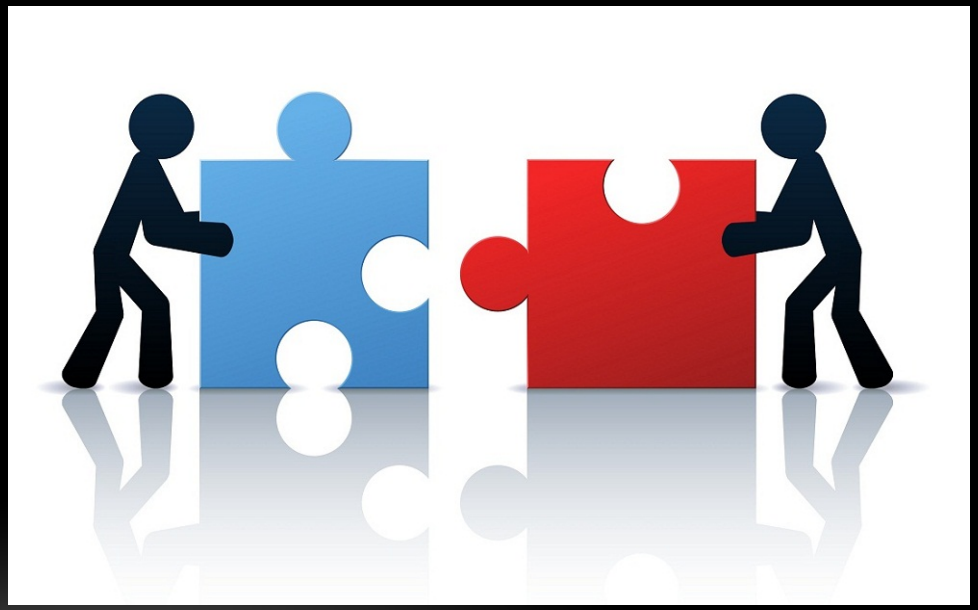
Data Output | Explain | Messages | History

	divisionid character(10)	name character varying(26)	location character varying(30)	employeeid character(10)	contractorid character(10)
1	01	Documentation	New Jersey	002	6
2	02	Customer Communica	New York	003	5
3	03	Help Design	New Jersey	004	4
4	04	Knowledge Transfer	New York	005	3
5	05	Research	New Jersey	006	2
6	06	Top Secret	Connecticut	007	1

THE *PROJECT* TABLE

- This table uses projectID as the primary key.
- Functional Dependencies: projectID → name, location, startDate, finishDate
- The project table references each project belonging to a particular division by using the foreign key divisionID.





PROJECT TABLE CONTINUED

Create Statement

```
--------------
-- projects --
--------------

drop table if exists projects;
create table projects (
        projectID       char(10) not null,
        name            varchar (26),
        location        varchar (30),
        startDate       date not null,
        finishDate      date,
        divisionID      char (10) not null,
primary key (projectID),
foreign key (divisionID) references division (divisionID)
);
```

PROJECT TABLE CONTINUED

Sample Data

Data Output | Explain | Messages | History

	projectid character(10)	name character varying(26)	location character varying(30)	startdate date	finishdate date	divisionid character(10)
1	11	Phone	New York	2010-07-04	2013-11-10	01
2	22	Tablet	New York	2009-02-28	2013-11-10	03
3	33	PC	Connecticut	2012-04-11	2013-11-10	02
4	44	Laptop	New Jersey	2011-12-20	2013-11-10	01
5	55	Holograph Phone	New York	2011-12-20	2013-11-10	04
6	66	Pocket projectsor	New York	2011-12-20	2013-11-10	05
7	77	Glass Translator	Connecticut	2011-12-20	2013-11-10	06

THE *PRODUCTION* TABLE

- The production table utilizes productID as the primary key.
- Functional Dependencies: produciontID → startDate, releaseDate
- Foreign Keys: divisionID references which product came from what division

  projectID references which product came from what project

production
PK productionID
startDate
releaseDate
divisionID
FK projectID

PRODUCTION TABLE CONTINUED

```
--------------                    Create Statement
--Production--
--------------

drop table if exists production;
create table production (
        productionID        char(10) not null,
        releaseDate         date,
        divisionID          char (10) not null,
        projectID           char (10) not null,
primary key (productionID),
foreign key (divisionID) references division (divisionID),
foreign key (projectID)  references projects (projectID)
);
```

PRODUCTION TABLE CONTINUED

Sample Data

Data Output | Explain | Messages | History

	productionid character(10)	releasedate date	divisionid character(10)	projectid character(10)
1	1	2013-11-28	06	77
2	2	2013-11-28	05	66
3	3	2013-11-28	04	55
4	4	2013-11-28	01	44
5	5	2013-11-28	02	33
6	6	2013-11-28	03	22
7	7	2013-11-28	01	11

CROSS JOIN

select *

from employees, contractors

where salaryUSD = 95000.00

Data Output | Explain | Messages | History

	employeeid character(10)	title character varying(26)	startdate date	firedate date	salaryusd numeric(12,2)	divisionid character(	projectid character(10	contractorid character(10	company character varying(30)	school character varying(30)	contractrateusd numeric(12,2)	divisionid character(10)
1	002	Software Engineer	2012-06-12	2013-12-31	95000.00	01	11	1	Apple	University of Stamfor	44000.00	06
2	002	Software Engineer	2012-06-12	2013-12-31	95000.00	01	11	2	Microsoft	Columbia University	56000.00	05
3	002	Software Engineer	2012-06-12	2013-12-31	95000.00	01	11	3	Oracle	ITT Tech	78000.00	04
4	002	Software Engineer	2012-06-12	2013-12-31	95000.00	01	11	4	Goldman Sachs	NYU	65000.00	03
5	002	Software Engineer	2012-06-12	2013-12-31	95000.00	01	11	5	Morgan Stanley	Vassar College	55000.00	02
6	002	Software Engineer	2012-06-12	2013-12-31	95000.00	01	11	6	NYSE	University of Albany	80000.00	01

INNER JOIN

SELECT employeeID, name, division

FROM people INNER JOIN division

ON peopleID = division.employeeID

Data Output | Explain | Messages | History

	employeeid character(10)	name character varying(26)	division division
1	002	Documentation	("01
2	003	Customer Communica	("02
3	004	Help Design	("03
4	005	Knowledge Transfer	("04
5	006	Research	("05
6	007	Top Secret	("06

LEFT OUTER JOIN

SELECT employeeID, NAME, division

FROM people LEFT OUTER JOIN division

ON peopleID = division.employeeID

Data Output | Explain | Messages | History

	employeeid character(10)	name character varying(26)	division division
1	002	Documentation	("01
2	003	Customer Communica	("02
3	004	Help Design	("03
4	005	Knowledge Transfer	("04
5	006	Research	("05
6	007	Top Secret	("06
7			
8			
9			
10			
11			
12			

RIGHT OUTER JOIN

SELECT employeeID, name, division

FROM people RIGHT OUTER JOIN division

ON peopleID = division.employeeID

Data Output | Explain | Messages | History

	employeeid character(10)	name character varying(26)	division division
1	002	Documentation	("01
2	003	Customer Communica	("02
3	004	Help Design	("03
4	005	Knowledge Transfer	("04
5	006	Research	("05
6	007	Top Secret	("06

FULL OUTER JOIN

SELECT employeeID, name, division

FROM people FULL OUTER JOIN division

ON peopleID = division.employeeID

Data Output | Explain | Messages | History

	employeeid character(10)	name character varying(26)	division division
1	002	Documentation	("01
2	003	Customer Communica	("02
3	004	Help Design	("03
4	005	Knowledge Transfer	("04
5	006	Research	("05
6	007	Top Secret	("06
7			
8			
9			
10			
11			
12			

SUB-QUERY

```
SELECT *
FROM people
WHERE peopleID IN (SELECT employeeID
                   FROM employees
                   WHERE salaryUSD < 100000)
```

Data Output | Explain | Messages | History

	peopleid character(10)	firstname character varying(20)	lastname character varying(20)	middleinitial character varying(1)	address character varying(20)
1	002	Bill	Fairbanks	B	Beirut
2	003	Jack	Mason	B	Siberia
3	005	Stuart	Thomas	B	Greece
4	006	Alec	Trevelyan	B	Russia
5	007	James	Bond	B	Classified

SUM CALCULATION

Data Output E

	cost numeric
1	3000.00

SELECT SUM(salaryUSD) AS cost

FROM employees WHERE startDate

BETWEEN '03/15/2010' AND '06/12/2012'

GROUP BY

```
SELECT name
FROM division
GROUP BY division.name
HAVING name <> 'Help Design'
```

Data Output | Explain | Messa

	name character varying(26)
1	Top Secret
2	Knowledge Transfer
3	Research
4	Customer Communica
5	Documentation

HAVING CLAUSE

```
SELECT name
FROM division
GROUP BY name
HAVING count(name) < 9
```

Data Output | Explain | Messa

	name character varying(26)
1	Top Secret
2	Knowledge Transfer
3	Research
4	Help Design
5	Customer Communica
6	Documentation

CREATE VIEW

```
CREATE VIEW people_view
 AS
SELECT peopleID, firstNAME,lastName, address
FROM  people;
----------------------------
select *
 from people_view
-----------------------------
FULL OUTER JOIN FROM VIEW
------------------------------
SELECT peopleID, name, division, location
FROM people FULL OUTER JOIN division
ON peopleID = division.employeeID
```

Data Output | Explain | Messages | History

	peopleid character(10)	name character varying(26)	division division	location character varying(30)
1	002	Documentation	("01	New Jersey
2	003	Customer Communica	("02	New York
3	004	Help Design	("03	New Jersey
4	005	Knowledge Transfer	("04	New York
5	006	Research	("05	New Jersey
6	007	Top Secret	("06	Connecticut
7	012			
8	015			
9	010			
10	013			
11	011			
12	014			

CREATE USER, PASSWORD, GRANT, REVOKE

- CREATE USER Alan WITH PASSWORD 'isawesome'
- GRANT ALL ON people TO Alan
- REVOKE ALL ON COMPANY FROM
- CREATE INDEX Amazing_Index

  ON production (releaseDate)

KNOWN ISSUES AND FUTURE ENHANCEMENTS

Issues

- There is a major lack of security. As it stands anyone could have access to anything within this business.
- There could be more views that would help to implement better structure.
- Joins could be more meaningful.

Enhancements

- Adding much better joins that have more meaning to the business.
- Adding more views to better enhance the functionality of the business
- Adding views can also be used for implementation of tighter security so not everyone has all access everywhere and Top Secret is actually Top Secret.