UNIVERSIDAD DE ORIENTE.

NUCLEÓ ANZOÁTEGUI.

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE INGENIERIAS.

MATERIA: SISTEMAS DE BASES DE DATOS



ASIGNACIÓN SQL

**PROFESOR:**

**JOSE GUEVARA**

**ALUMNO:**

**Ing. Computación**

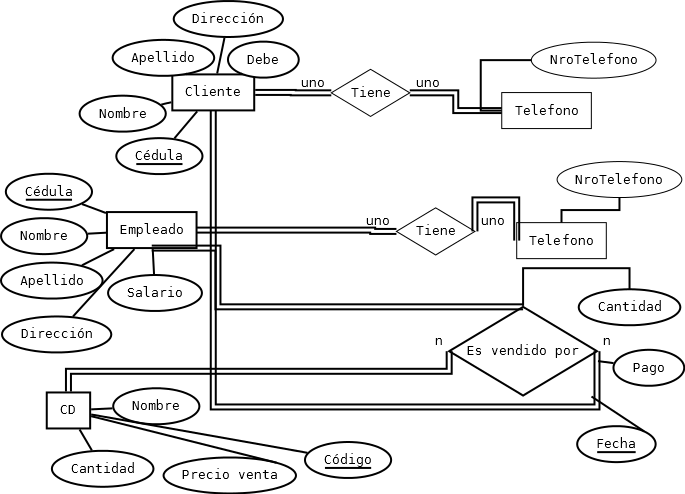
**LUIS CORREA CI: 19.840.230**

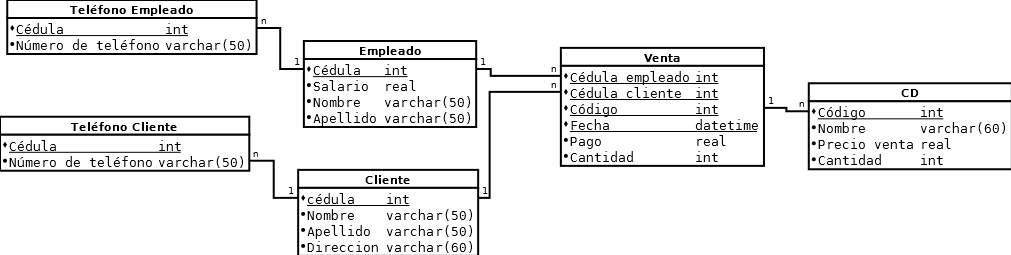
Barcelona, 06 de Mayo del 2014 SECCION­:01

## Desarrollo

1.-Base de datos relacional de una Tienda que vende cd de música que almacena datos de cds, empleados, clientes, y de las ventas.

**Esquemas:**

****



**Instancias:**

|  |  |  |  |
| --- | --- | --- | --- |
| CLIENTE | | | |
| Cedula | Nombre | Apellido | Dirección |
| 19888888 | Luis | Correa | Hospital |
| 19888889 | Ana | Correa | Hospital |
| 18888887 | Patricia | Correa | Hospital |
| 21888887 | Jorge | Garcia | Lecheria |
| 12342161 | Adel | Diaz | Pto la Cruz |
| 21808880 | Cesar | Diaz | Morro Humbolt |

|  |  |  |  |
| --- | --- | --- | --- |
| CD | | | |
| Codigo | Nombre | Precio | Cantidad |
| 123 | Guitarra Electrica | 300 | 50 |
| 124 | Violin | 300 | 50 |
| 125 | Cuatro | 300 | 50 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EMPLEADO | | | | |
| Cedula | Nombre | Apellido | Direccion | Salario |
| 10000000 | Manuel | Martinez | Lecheria | 8000 |
| 11000000 | Gabriela | Hurtado | Iglesia San Jorge | 8000 |

|  |  |
| --- | --- |
| TELEFONOC | |
| Teléfono | Cedula |
| 02812389456 | 19888888 |
| 02812345861 | 19888889 |
| 02812345862 | 19888887 |
| 02812342863 | 21888887 |
| 02818488787 | 12345161 |
| 02812320191 | 21808880 |

|  |  |
| --- | --- |
| TELEFONOE | |
| Teléfono | Cedula |
| 02812345880 | 10000000 |
| 02812345870 | 11000000 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Venta | | | | | |
| Fecha | CedulaC | CedulaE | Código | Pago | Cantidad |
| 2013-01-12 12:21:02 | 19888887 | 10000000 | 123 | 300 | 5 |
| 2013-01-12 12:23:03 | 19888888 | 10000000 | 123 | 300 | 5 |
| 2013-01-12 12:24:04 | 19888889 | 10000000 | 123 | 300 | 5 |
| 2013-01-12 12:25:01 | 21808880 | 11000000 | 125 | 300 | 5 |
| 2013-01-12 12:26:01 | 21888887 | 10000000 | 125 | 300 | 5 |
| 2013-01-12 12:27:01 | 21888887 | 10000000 | 123 | 300 | 5 |
| 2013-01-12 12:21:02 | 19888887 | 10000000 | 123 | 300 | 5 |

**Dependencias Funcionales:**

TelefonoC:

F = {Cedula → Número de teléfono}

TelefonoE:

F = {Cédula → Número de teléfono}

Empleado:

F = {Cedula → Salario, Nombre, Apellido, Dirección, Salario}

Cliente:

F = {Cédula →Nombre, Apellido, Dirección}

Venta:

F = {(Fecha, CedulaE, CedulaC, Código) → Pago, Cantidad}

CD:

F = {Codigo → Nombre, Precio, Cantidad}

**Cerradura de conjuntos:**

De TelefonoC tenemos Cedula como clave primaria

Cerradura <- Cedula

Anterior <- Cedula

Cerradura<- Cedula, Número de teléfono

Anterior <- Cedula, Número de teléfono

De TelefonoE tenemos Cedula como clave primaria

Cerradura <- Cedula

Anterior <- Cedula

Cerradura<- Cedula, Número de teléfono

Anterior <- Cedula, Número de teléfono

Empleado tenemos Cedula como clave primaria

Cerradura <- Cedula

Anterior <- Cedula

Cerradura <- Cedula, Salario, Nombre, Apellido, Teléfono

Anterior <- Cedula, Salario, Nombre, Apellido, Teléfono

Cliente tenemos Cedula como clave primaria

F = {Cédula →Nombre, Apellido, Dirección}

Cerradura <- Cedula

Anterior <- Cedula

Cerradura <- Cedula, Nombre, Apellido, Dirección

Anterior <- Cedula, Nombre, Apellido, Dirección

Venta tenemos Fecha, CedulaE, CedulaC, Código como súperclave

F = {(Fecha, CedulaE, CedulaC, Código) → Pago, Cantidad}

Cerradura <- Fecha, CedulaE, CedulaC, Código

Anterior <- Fecha, CedulaE, CedulaC, Código

Cerradura <- Fecha, CedulaE, CedulaC, Código, Pago, Cantidad

Anterior <- Fecha, CedulaE, CedulaC, Código, Pago, Cantidad

CD tenemos Código como clave primaria

Cerradura <- Código, Nombre, Precio, Cantidad

Anterior <- Código, Nombre, Precio, Cantidad

Cerradura <- Código, Nombre, Precio, Cantidad

Anterior <- Código, Nombre, Precio, Cantidad

**Documentación:**

En cd almacenamos código, nombre, precio, cantidad de cd de música instrumental, in cd puede ser vendido a varios clientes, vendido por varios empleados, siempre serán vendidos todos los cd instrumentales.

En empleados almacenamos cedula, salario, nombre, apellido, dirección, salario, los empleados tienen muchos teléfonos y además venden todos los cd´s instrumentales.

En teléfono de empleados almacenamos cédula, número de teléfono, todos los teléfonos tienen su cedula asociada a todos los empleados, los empleados tienen muchos teléfonos o al menos un teléfono.

En clientes almacenamos cédula, nombre, apellido, dirección

En teléfono de clientes almacenamos cedula, número de teléfono, todos los teléfonos tienen su cedula asociada a todos los clientes, los clientes tienen muchos teléfonos o al menos un teléfono.

En venta almacenamos fecha, cédula de empleado, cédula de cliente, Código de cd de instrumento, pago, cantidad de cd´s comprados

**DISTINCT:**

Sirve para mostrar datos de las consultas si repertir datos, por ejemplo todos los apellidos de un empleado sin repetir la consulta seria así, SELECT DISTINCT Apellido FROM Cliente.

Respuesta seria:

|  |
| --- |
| Apellido |
| Correa |
| Garcia |
| Diaz |

**UNION:**

Permite trabajar las columnas de las tablas, por ejemplo al realizar la unión de dos columnas como la de los nombres de empleados con los nombres de los clientes mostraria los nombres de los clientes y empleados sin mostrar nombres repetidos, para que funcione las columnas deben tener el mismo tipo de dato,

use Tienda;

SELECT Empleado.Nombre FROM Empleado UNION SELECT Cliente.Nombre FROM Cliente;

Respuesta seria:

|  |
| --- |
| Nombre |
| Manuel |
| Gabriela |
| Adel |
| Patricia |
| Luis |
| Ana |
| Cesar |
| Jorge |

**INTERSECT:**

Permite mostrar datos en comun en una serie de columnas a intersectar, para que funcione las columnas que se intersecten deben tener el mismo tipo de dato por ejemplo al intersectar los nombres de los clientes y los empleados obtendriamos solo los nombres que aparecen en ambas columnas sin repetir se haria así:

use Tienda;

SELECT Cliente.Nombre FROM Cliente INTERSECT SELECT Empleado.Nombre FROM Empleado;

Respuesta seria:

|  |
| --- |
| Nombre |

No arroja resultado por que no tenemos ningun nombre en comun en la tabla empleados y estudiantes.

**EXCEPT:**

Permite mostrar los datos de una columna que no sean iguales a los que estan en otra columna para que funcione las columnas deben tener el mismo tipo de datos, podemos mostrar los nombres de los Clientes que no sean iguales a los de los Empleados así:

Use Tienda;

SELECT Nombre FROM Cliente EXCEPT SELECT Nombre FROM Empleado;

Respuesta seria:

|  |
| --- |
| Nombre |
| Luis |
| Ana |
| Patricia |
| Jorge |
| Adel |
| Cesar |

**ANY:**

Compara un valor escalar con un conjunto de valores de una sola columna. SOME y ANY son equivalentes, por ejemplo para saber si existe un empleado con una cedula, hariamos una consulta así:

Use Tienda;

IF 4 = ANY (SELECT Cedula FROM Empleado)

PRINT 'existe un Empleado con esa cedula’

ELSE

PRINT 'existen empleados con esa cedula' ;

Respuesta seria: existe un Empleado con ese id

**ALL:**

Compara un valor escalar con un conjunto de valores de una sola columna, por ejemplo para saber si todos los empleados tienen cedula 10000000, hariamos una consulta así:

IF 10000000 = ALL (SELECT idEmpleado FROM Empleado)

PRINT 'todos los empleados tienen esa Cedula'

ELSE

PRINT 'no todos los empleados tienen esa cedula' ;

Respuesta seria: no todos los empleados tienen esa cedula

**Comparación de sub-cadenas:**

Un ejemplo de comparar cadenas podría ser:

IF 'Luis Correa' = (SELECT Nombre+' '+Apellido FROM Cliente WHERE Cedula=19888888)

PRINT 'son iguales'

ELSE

PRINT 'no son iguales' ;

Respuesta seria: son iguales

Otro ejemplo podria ser usando LIKE.

Definición de LIKE:

Determina si una cadena de caracteres específica coincide con un patrón especificado.  Por ejemplo para lograr en una consulta que me muestre los nombres de los empleados con apellido que empiece por C seria así:

Use Tienda;

SELECT Nombre FROM Cliente WHERE Apellido LIKE '%C%';

Respuesta seria:

|  |
| --- |
| Nombre |
| Luis |
| Ana |
| Patricia |

**Operadores aritméticos:**

Los operadores aritméticos realizan operaciones matemáticas con dos expresiones de uno o más de los tipos de datos de la categoría de tipos de datos numéricos.

|  |  |
| --- | --- |
| **Operador** | **Significado** |
| [+ (sumar)](http://msdn.microsoft.com/es-es/library/ms178565.aspx) | Suma |
| [- (restar)](http://msdn.microsoft.com/es-es/library/ms189518.aspx) | Resta |
| [\* (multiplicar)](http://msdn.microsoft.com/es-es/library/ms176019.aspx) | Multiplicación |
| [/ (dividir)](http://msdn.microsoft.com/es-es/library/ms175009.aspx) | División |
| [% (Módulo)](http://msdn.microsoft.com/es-es/library/ms190279.aspx) | Devuelve el resto entero de una división. Por ejemplo, 12 % 5 = 2 |

DECLARE @X INT;

SET @X = 1;

DECLARE @Y INT;

SET @Y = 2;

DECLARE @Z INT;

SET @Z = @X+@Y;

PRINT @Z

**Clausula ORDER BY:**

Ordenar el conjunto de resultados de una consulta por la lista de columnas especificada y opcionalmente limitar las filas devueltas a un intervalo especificado. El orden en que se devuelven las filas en un conjunto de resultados no se puede garantizar, a menos que se especifique una cláusula ORDER BY. Un ejemplo podría ser mostrar los nombres de los clientes por orden alfabético

Use Tienda;

SELECT Nombre FROM Cliente ORDER by Nombre

Respuesta seria:

|  |
| --- |
| Nombre |
| Adel |
| Ana |
| Cesar |
| Jorge |
| Luis |
| Patricia |

**Comparación de valores nulos:**

El valor NULL significa que el valor de los datos de la columna no se conoce o no está disponible. NULL no es sinónimo de cero, ni de una cadena de longitud cero ni de valor en blanco. Los valores NULL permiten distinguir entre una entrada con el valor cero (columnas numéricas) o en blanco (columnas de caracteres) y valores sin entrada (NULL tanto para columnas numéricas como de caracteres). Por ejemplo no obtendríamos ningún resultado de esta consulta debido a que las Cedulas de los empleados no admiten valores nulos.

Use Tienda;

SELECT Cedula, Nombre, Apellido FROM Empleado WHERE Cedula IS NULL;

Resultado: No mostraria nada debido a que no hay empleados con una Cedula null.

**Uso del operador IN:**

Determina si un valor especificado coincide con algún valor de una sub-consulta o una lista, un ejemplo seria la consulta de los clientes.

IF ('Luis Correa' IN (SELECT Nombre+' '+Apellido FROM Cliente))

PRINT 'encontró un valor igual'

ELSE

PRINT 'todos no tienen el mismo valor';

Respuesta seria: encontró un valor igual

**Uso del EXISTS y NOT EXISTS:**

Especifica una sub-consulta para probar la existencia de filas. [NOT] EXISTS (orden SELECT): El operador EXISTS nos informa de si una sub-consulta ha obtenido algún resultado: devuelve verdadero si hay al menos una fila en la relación derivada y falso si la relación derivada es vacía. Por ejemplo usando exist para obtener el nombre y apellido de los clientes que no son empleados de la tienda.

SELECT a.Nombre, a.Apellido FROM Cliente a

WHERE EXISTS (SELECT \* FROM Empleado b

WHERE a.Cedula != b.Cedula);

La respuesta seria así:

|  |  |
| --- | --- |
| Nombre | Apellido |
| Luis | Correa |
| Ana | Correa |
| Patricia | Correa |
| Jorge | Garcia |
| Adel | Diaz |
| Cesar | Diaz |

**División con consulta anidada:**

La división como operador definido en el álgebra relacional nos da como resultado "algo que se relaciona con todos los que cumplen una condición". Por ejemplo la cedula, nombres y apellidos de todos los clientes que han comprado todos los diferentes tipos de cd en la tienda.

use tienda;

select distinct cedula, nombre, apellido from cliente, venta as c

where c.cedulac=cedula and not exists (

select codigo from cd

except

select codigo from venta as e where c.cedulac=e.cedulac

);

**Cambios de nombres de atributos y uso de seudónimos:**

Se puede cambiar el nombre de una columna de la tabla mediante una consulta SQL, un ejemplo podría ser el de cambiar el atributo cedula con id y luego devolverlo a su estado original así:

EXEC sp\_rename 'Cliente.Cedula', 'Id', 'COLUMN';

SELECT \* FROM Cliente

Respuesta seria:

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Nombre | Apellido | Dirección |

EXEC sp\_rename 'Persona.Id', 'Cedula', 'COLUMN';

SELECT \* FROM Persona

Respuesta seria:

|  |  |  |  |
| --- | --- | --- | --- |
| Cedula | Nombre | Apellido | Dirección |

**Consultas con JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN:**

Uso JOIN:

Realizar una reunión para obtener el nombre y apellido de los clientes junto con sus compras en la tienda, solo los clientes que han comprado:

use tienda;

select Cliente.Nombre, Apellido, CD.Nombre

from (Cliente Join Venta on Cedula=CedulaC),CD where Venta.Codigo=CD.Codigo;

|  |  |  |
| --- | --- | --- |
| Nombre | Apellido | Nombre |
| Adel | Diaz | Cuatro |
| Cesar | Diaz | Cuatro |
| Jorge | Garcia | Cuatro |
| Patricia | Correa | Guitarra Electrica |
| Luis | Correa | Guitarra Electrica |
| Ana | Correa | Guitarra Electrica |

LEFT OUTER JOIN: Modificar

Retorna la pareja de todos los valores de la tabla izquierda con los valores de la tabla de la derecha correspondientes, o retorna un valor nulo en caso de no correspondencia. Por ejemplo si queremos mostrar la tabla de los clientes que han realizado una compra lo haríamos de la siguiente manera:

SELECT \* FROM Cliente LEFT OUTER JOIN Venta ON

Cliente.Cedula = Venta.CedulaC;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cedula | Nombre | Apellido | Dirección | Fecha | CedulaC | CedulaE | Codigo | Pago | Cantidad |
| 12342161 | Adel | Diaz | Pto la Cruz | 2013-01-12 12:20:01 | 12342161 | 11000000 | 125 | 300 | 5 |
| 19888887 | Patricia | Correa | Hospital | 2013-01-12 12:21:02 | 19888887 | 10000000 | 123 | 300 | 5 |
| 19888888 | Luis | Correa | Hospital | 2013-01-12 12:23:03 | 19888888 | 10000000 | 123 | 300 | 5 |
| 19888889 | Ana | Correa | Hospital | 2013-01-12 12:24:04 | 19888889 | 10000000 | 123 | 300 | 5 |
| 21808880 | Cesar | Diaz | Morro Humbolt | 2013-01-12 12:25:01 | 21808880 | 11000000 | 125 | 300 | 5 |
| 21888887 | Jorge | Garcia | Lecheria | 2013-01-12 12:26:01 | 21888887 | 10000000 | 125 | 300 | 5 |

Respuesta nos daría todos los datos de los clientes que han realizado compras con las fechas que realizaron las compras al lado.

RIGHT OUTER JOIN: Modificar

Esta operación es inversa a la anterior; el resultado de esta operación siempre contiene todos los registros de la segunda tabla que se menciona en la consulta, aun cuando no exista un registro correspondiente en la tabla de la izquierda para uno de la derecha.

La sentencia RIGHT OUTER JOIN retorna la pareja de todos los valores de la tabla derecha con los valores de la tabla de la izquierda correspondientes, o retorna un valor nulo en caso de no correspondencia. Por ejemplo:

SELECT \* FROM Cliente RIGHT OUTER JOIN Venta ON

Cliente.Cedula = Venta.CedulaC;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cedula | Nombre | Apellido | Dirección | Fecha | CedulaC | CedulaE | Codigo | Pago | Cantidad |
| 12342161 | Adel | Diaz | Pto la Cruz | 2013-01-12 12:20:01 | 12342161 | 11000000 | 125 | 300 | 5 |
| 19888887 | Patricia | Correa | Hospital | 2013-01-12 12:21:02 | 19888887 | 10000000 | 123 | 300 | 5 |
| 19888888 | Luis | Correa | Hospital | 2013-01-12 12:23:03 | 19888888 | 10000000 | 123 | 300 | 5 |
| 19888889 | Ana | Correa | Hospital | 2013-01-12 12:24:04 | 19888889 | 10000000 | 123 | 300 | 5 |
| 21808880 | Cesar | Diaz | Morro Humbolt | 2013-01-12 12:25:01 | 21808880 | 11000000 | 125 | 300 | 5 |
| 21888887 | Jorge | Garcia | Lecheria | 2013-01-12 12:26:01 | 21888887 | 10000000 | 125 | 300 | 5 |

Respuesta nos daría todos los datos de los clientes que han realizado compras con las fechas que realizaron las compras al lado.

**Funciones agregadas:**

El lenguaje SQL incorpora una serie de funciones que nos permiten obtener distintos resultados numéricos al ejecutar consultas. Son las siguientes:

Count:

Muestra el número de registros que devuelve una consulta.

Avg:

Muestra el promedio de los valores evaluados.

Sum:

Muestra la suma de los valores evaluados.

Max:

Muestra el valor más alto de los valores evaluados.

Ejemplo de count: Se quiere mostrar el número de empleados en la tienda, el promedio del salario de los empleados, la suma de los salarios de los empleados de la tienda

select sum(salario) from empleado, el salario más alto de los empleados en la tienda

use tienda;

select count(cedula) as numero\_de\_empleados

,avg(salario) as promedio

,sum(salario) as suma\_salarios

,max(salario) as salario\_maximo from empleado;

|  |  |  |  |
| --- | --- | --- | --- |
| Numero\_empleados | promedio | Suma\_salarios | Salario\_maximo |
| 2 | 8000 | 16000 | 8000 |

**Uso de la cláusula GROUP BY y HAVING:**

GROUP BY:

Especifica la agrupación que se da a los datos. Se usa siempre en combinación con funciones agregadas. Las columnas indicadas en el GROUP BY se llaman columnas de agrupación.

Ejemplo Mostrar los clientes y cuantas compras han realizado.

select Cliente.Nombre,Count(CedulaC) as compro from Cliente join Venta

on Cedula=CedulaC group by Cedula;

|  |  |
| --- | --- |
| Nombre | Compro |
| Adel | 1 |
| Patricia | 1 |
| Luis | 1 |
| Ana | 1 |
| Cesar | 1 |
| Jorge | 2 |

HAVING:

Es utilizada junto con SELECT para especificar una condición de búsqueda para un grupo. HAVING se comporta como WHERE, pero se aplica a grupos (las columnas o filas en el conjunto de resultados representan grupos). Ejemplo se desea saber que cliente ha comprado más de 1 vez en la tienda se haría la consulta así:

SELECT Cedula,Nombre, count(\*) AS Total FROM Venta join Cliente

on CedulaC=Cedula

GROUP BY Cedula HAVING (count(\*) >= 2);

|  |  |  |
| --- | --- | --- |
| Cedula | Nombre | Compro |
| 21888887 | Jorge | 2 |

**Objetivos:**

Asignación SQL (Valor: 2 ptos del II Parcial)

Lo primero que usted debe realizar es plantear una base de datos relacional que sea creada por usted mismo (producto de su imaginación) en donde se muestren los esquemas de las relaciones junto con una instancia de esas relaciones. Adicionalmente usted debe explicar brevemente cada uno de los siguientes puntos y basándose en el modelo planteado, dar ejemplos de consultas que refuercen la explicación que está dando. Para cada consulta se debe mostrar el resultado en función de las instancias de relación especificadas:

* Uso de la cláusula DISTINCT
* Operadores de Conjuntos: UNION, INTERSECT, EXCEPT
* Uso del ANY y del ALL
* Comparaciones de Sub-Cadenas
* Operadores Aritméticos
* Uso de la Cláusula ORDER BY
* Comparación de valores nulos
* Especificación de conjuntos explícitos con el operador IN
* Consultas anidadas (Operador IN)
* Uso del EXISTS y NOT EXISTS
* Representación del operador del álgebra relacional división en SQL
* Cambios de nombres de atributos y uso de seudónimos
* Consultas con JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN
* Funciones agregadas
* Uso de la cláusula GROUP BY y HAVING

NOTA: No se aceptarán modelos relacionales y consultas que estén documentadas en libros o en Internet. Esta asignación es INDIVIDUAL.

Fecha de Entrega: miércoles 09 de julio de 2014