Universidad De Oriente
Núcleo De Anzoátegui
Escuela De Ingeniería Y Ciencias Aplicadas
Departamento De Computación Y Sistemas

Asignación SQL

Profesor: Guevara, José

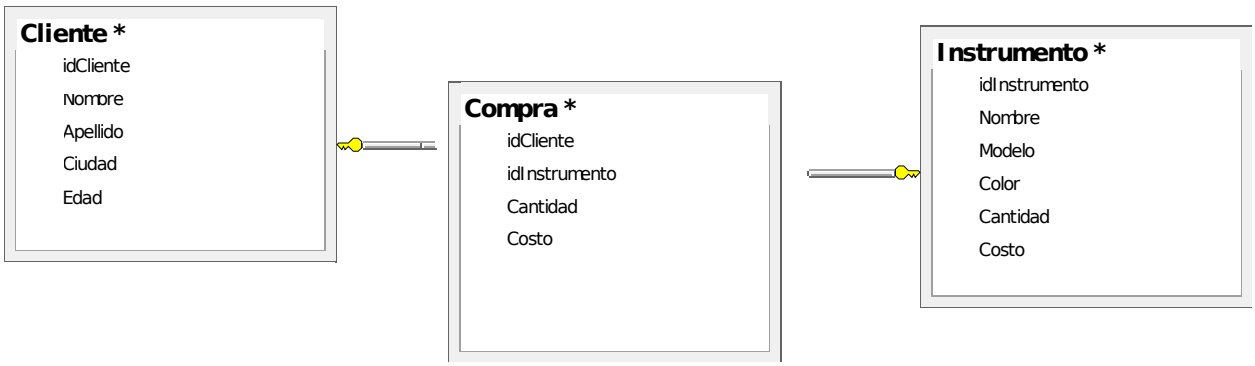
Bachiller:
Delgado, Javier

CI:
19.940.338

Barcelona, Abril de 2012

Base de Datos de una Tienda de música Yamaha que desea almacenar los datos de todos los clientes, los instrumentos que se veden y los registros de compra del cliente.

Cliente (idCliente, Nombre, Apellido, Ciudad, Edad)
Instrumento (idInstrumento, Nombre, Modelo, Color, Cantidad, Costo)
Compra (idCliente,idInstrumento, Cantidad, Costo)



- **Tabla Cliente**

idCliente	Nombre	Apellido	Ciudad	Edad
1	Javier	Delgado	Puerto la cruz	22
2	Carlos	Pérez	Barcelona	25
3	Juan	Rojas	El Tigre	30
4	Pedro	Díaz	Barcelona	20
5	Claudio	Delgado	El Tigre	20

- **Tabla Instrumento**

idInstrumento	Nombre	Modelo	Color	Cantidad	Costo
1	Teclado	PSR-E403	Gris	5	5000,0000
2	Bateria	Yamaha	Gris	1	10000,0000
3	Guitarra	Española	Rojisa	3	2000,0000
4	Piano	Yamaha	Negro	10	100000,0000

- **Tabla Compra**

idCliente	idInstrumento	Cantidad	Costo
1	1	1	5000,0000
1	2	1	10000,0000
4	1	1	5000,0000
3	3	1	2000,0000
5	4	1	100000,0000

- **Uso de la Cláusula DISTINCT**

  Con la cláusula ***distinct*** se especifica que los registros con ciertos datos duplicados sean obviados en el resultado. Por ejemplo, queremos conocer todos los clientes que han comprado un instrumento alguna vez

  Consulta

```
select distinct idCliente from Compra;
```

idCliente
1
2
3
4
5

Resultado

En la Relación Compra se puede reflejar que el idCliente 1 Compra al menos dos veces algún instrumento y al realizar la consulta usando esta cláusula ella obvia ese dato repetido.

- **Operadores de Conjuntos: UNION, INTERSECT, EXCEPT**
  - **UNION**

    El operador UNION permite combinar los resultados de varias instrucciones SELECT en un único conjunto de resultados. Todos los conjuntos de resultados combinados mediante UNION deben tener la misma estructura. Deben tener el mismo número de columnas y las columnas del conjunto de resultados deben tener tipos de datos compatibles.

    Con esta base de datos creada es imposible hacer uso de este operador porque no tenemos al menos dos tablas que sean compatibles para realizar esta consulta, por lo tanto se ha colocado dos tablas de ejemplo para poder ver su uso y funcionamiento.

    Tabla1

ColumnaA	ColumnaB
ABC	1
DEF	2

GHI	3

Tabla2

ColumnaA	ColumnaB
GHI	3
JKL	4
MNO	5

Consulta

```
SELECT * FROM Tabla1
UNION
SELECT * FROM Tabla2
```

Resultado

ColumnaA	ColumnaB
abc	1
def	2
ghi	3
jkl	4
mno	5

Se puede reflejar la unión de las tabla1 y la tabla2 sin repetición de atributos.

- **INTERSECT**
  Este tipo de operador devuelve los valores distintos devueltos por las consultas situadas a los lados izquierdo y derecho del operando INTERSECT.

  Consulta

  ```
  SELECT idCliente FROM Cliente INTERSECT SELECT idCliente FROM Compra
  ```

  Resultado

idCliente
1
3
4
5

El idCliente 2 No está en la tabla de resultado por que existe en la tabla compra.

- **EXCEPT**

  Este tipo de operador devuelve los valores distintos de la consulta inicial que no se devuelven desde la consulta siguiente. Es decir en dependencia de la ubicación de la consulta nos regresara los valores que no se encuentren la siguiente.

  Un Ejemplo seria Obtener los Clientes Registrados que no han realizado ninguna compra de un instrumento

  Consulta

```
SELECT idCliente FROM Cliente
EXCEPT
SELECT idCliente FROM Compra
```

  Resultado

idCliente
2

- **Uso del ANY y del ALL**
  - ANY

    La palabra clave ANY , que debe seguir a un operador de comparación, significa “return TRUE si la comparación es TRUE para ANY (cualquiera) de los valores en la columna que retorna la subconsulta.”

    Ejemplo se desea saber el idCliente que ha comprado el único instrumento que estaba en existencia.

    Consulta

```
SELECT Cantidad  FROM Instrumento WHERE Cantidad = ANY
(SELECT Cantidad FROM Compra);
```

Resultado

idCliente
1

- ALL

Este predicado Devuelve todos los campos de la tabla

Ejemplo Se desea saber todos las cantidad de instrumentos cuya cantidad sean mayores a todas las cantidades de la relación compra

Consulta

```
SELECT Cantidad  FROM Instrumento WHERE Cantidad =ALL
(SELECT Cantidad FROM Compra);
```

Resultado

Cantidad
5
3
10

- **Comparaciones de Sub-Cadenas**
- **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
+	Suma
-	Resta
*	Multiplicación
/	División
%	Modulo

Ejemplo se desea conocer todos aquellos instrumentos cuyo costo sea mayor que 3000 bsf

Consulta

```
SELECT Nombre  FROM Instrumento WHERE Costo >3000
```

Resultado

Nombre
Teclado
Bateria
Piano

- **Uso de la Cláusula ORDER BY**

Permite al usuario ordenar las tupias del resultado de una consulta según los valores de uno o más atributos.

Asc Retorna los valores de atributos de una relación ordenado de menor a mayor

Desc Retorna los valores de atributos de una relación ordenado de mayor a menor.

Ejemplo 1 Se Desea Visualizar el nombre y costo de todos aquellos instrumentos ordenados de menor a mayor.

Consulta

```
SELECT Nombre, Costo FROM Instrumento order by Costo Asc
```

Resultado

Nombre	Costo
Guitarra	2000,00
Teclado	5000,00
Bateria	10000,00
Piano	100000,00

Ejemplo 2 Visualizar el nombre y la edad de todos aquellos clientes ordenados de mayor a menor.

Consulta

```
SELECT Nombre, Edad FROM Cliente order by Edad Desc
```

Resultado

Nombre	Edad
Juan	30
Carlos	25
Javier	22
Pedro	20
Claudio	20

- **Comparacion de Valores nulos**

Si una tabla contiene valores nulos (representados como .VALOR ES NULL.) la siguiente instrucción SELECT siempre devolverá cero registros incluso cuando cFld1 contiene los valores null:

```
SELECT * FROM Cliente WHERE Edad = .NULL.
```

NULL debe utilizarse cuando un valor es irrelevante, desconocidos o ausentes. Cuando cualquier expresión condicional encuentra con un valor nulo, un .VALOR ES NULL. Se devolverá. El valor true o false no se puede devolver si parte de la expresión es desconocido o null. La expresión en el ejemplo anterior, Edad =.NULL., nunca se evaluará como True, en su lugar se evaluará como .VALOR ES NULL. y como resultado no se devolverá ningún registro.

Para localizar .VALOR ES NULL. valores de Edad, utilice la nueva cláusula es nulo del comando SELECT de SQL. La cláusula IS NULL Proporciona un mecanismo para la comparación de campos con .VALOR ES NULL los valores y devolver true o false. Por ejemplo, considere la posibilidad de este comando de selección:

```
SELECT * FROM Cliente WHERE Edad IS NULL
```

- **Especificación de conjuntos explícitos con el operador IN**
- **Consultas anidadas (Operador IN)**

Este operador devuelve aquellos registros cuyo campo indicado coincide con alguno de los en una lista.

Ejemplo Mostrar todos aquellos Clientes que han comprado algún instrumento alguna vez.

Consulta

```
SELECT  idCliente FROM Cliente
WHERE idCliente  IN ( SELECT idCliente  FROM Compra );
```

Resultado

idCliente

1
3
4
5

- **Uso del EXISTS y NOT EXISTS**

  Los operadores "exists" y "not exists" se emplean para determinar si hay o no datos en una lista de valores.

  Estos operadores pueden emplearse con subconsultas correlacionadas para restringir el resultado de una consulta exterior a los registros que cumplen la subconsulta (consulta interior). Estos operadores retornan "true" (si las subconsultas retornan registros) o "false" (si las subconsultas no retornan registros).

  Cuando se coloca en una subconsulta el operador "exists", SQL Server analiza si hay datos que coinciden con la subconsulta, no se devuelve ningún registro, es como un test de existencia; SQL Server termina la recuperación de registros cuando por lo menos un registro cumple la condición "where" de la subconsulta.

  La sintaxis básica es la siguiente:

  where exists (SUBCONSULTA);

  Ejemplo 1 Mostrar si existe un instrumento o más cuyo costo sea igual 2000bsf.

  Consulta

```
SELECT Nombre  FROM Instrumento AS i   WHERE exists  (SELECT
* FROM Instrumento   WHERE i.Costo=2000);
```

  Resultado

Nombre

Guitarra

Ejemplo 2

Mostrar aquellos instrumentos donde no exista un costo de 0bsf

Consulta

```
SELECT Nombre  FROM Instrumento AS i   WHERE not exists  (SELECT * FROM Instrumento WHERE i.Costo=0);
```

Resultado

Nombre
Teclado
Bateria
Guitarra
Piano

Se muestran todos los instrumentos porque no existe alguno cuyo costo sea de 0bsf

- **Representación del operador del álgebra relacional división en SQL**

  Por no tener tablas compatibles para realizar la división, se ha demostrado esta representación basado en dos tablas diferentes a las que ya se había creado previamente.

Opera sobre dos tablas. Si se divide una tabla B por una tabla A, se obtiene una nueva tabla cuyas columnas serán aquellas de la tabla B que no existen en la tabla A, y cuyas filas serán tales que cumplan con estar relacionadas con todas y cada una de las filas de la tabla A. Este caso es más dificil de visualizar, pero se puede entender analizando en detalle el siguiente ejemplo

TABLA A
CODIGO
1425
2000
3000

TABLA B	
CODIGO	INDICE
1425	15%
2000	27%
3000	33%
2128	45%
2121	13%
2000	15%
3000	15%

B DIVIDIDA POR A	
INDICE	15%

La columna INDICE es la única de la tabla B que no existe en la tabla A, y el valor 15% es el único valor de la misma que aparece en filas que se relacionan con todas las filas de la tabla A, es decir, las filas en las que CODIGO toma los valores 1425, 2000 y 3000, que son todos los que aparecen en la tabla A.

En el caso del ejemplo, podría usarse:

```
SELECT DISTINCT X.indice FROM bb X
```

```
WHERE NOT EXISTS (
   SELECT Y.codigo FROM aa Y
   WHERE NOT EXISTS (
      SELECT Z.codigo FROM bb Z
      WHERE Z.codigo=Y.codigo AND Z.indice=X.indice
   )
)
```

- **Cambios de nombres de atributos y uso de seudónimos**
- **Consultas con JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN**

La sentencia join en SQL permite combinar registros de dos o más tablas en una base de datos relacional.

- **LEFT OUTER JOIN**
  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 **NULL** en caso de no correspondencia.

  Consulta

```
SELECT * FROM Cliente LEFT OUTER JOIN Compra ON
Cliente.idCliente = Compra.idCliente
```

  Resultado

idCliente	Nombre	Apellido	Ciudad	Edad	idCliente	Id Instrumento	Cantidad	Costo
1	Javier	Delgado	Puerto la cruz	22	1	1	1	5000,00
1	Javier	Delgado	Puerto la cruz	22	1	2	1	10000,00
2	Carlos	Perez	Barcelona	25	NULL	NULL	NULL	NULL
3	Juan	Rojas	El Tigre	30	3	3	1	2000,00
4	Pedro	Diaz	Barcelona	20	4	1	1	5000,00
5	Claudio	Delgado	El Tigre	20	5	4	1	100000,00

El resultado son todos los registros de la Cliente, y si es posible las coincidencias de la Compra. Si no hay coincidencias, el lado derecho mostrará nulos.

- **RIGHT OUTER JOIN**

Esta operación es inversa a la anterior; el resultado de esta operación siempre contiene todos los registros de la tabla de la derecha (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 **NULL** en caso de no correspondencia.

Consulta

```
SELECT * FROM Cliente RIGHT OUTER JOIN Compra ON Cliente.idCliente = Compra.idCliente
```

Resultado

idCliente	Nombre	Apellido	Ciudad	Edad	idCliente	Id Instrumento	Cantidad	Costo
1	Javier	Delgado	Puerto la cruz	22	1	1	1	5000,00
1	Javier	Delgado	Puerto la cruz	22	1	2	1	10000,00
4	Pedro	Diaz	Barcelona	20	4	1	1	5000,00
3	Juan	Rojas	El Tigre	30	3	3	1	2000,00
5	Claudio	Delgado	El Tigre	20	5	4	1	100000,00

- **Funciones agregadas**

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

Funciones agregadas

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.

First:
muestra el primer valor entre los valores evaluados, teniendo en cuenta el método de ordenación aplicado.

Last:
muestra el último valor entre los valores evaluados, teniendo en cuenta el método de ordenación aplicado.

Stdev:
muestra la desviación estándar de los valores evaluados, a partir de una muestra de la población.

Stdevp:
muestra la desviación estándar de los valores evaluados, teniendo en cuenta toda la población.

Var:
muestra la varianza de los valores evaluados, a partir de una muestra de la población.

Varp:
muestra la varianza de los valores evaluados, teniendo en cuenta toda la población.

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

    Una consulta con una cláusula GROUP BY se denomina consulta agrupada ya que agrupa los datos de la tabla origen y produce una única fila resumen por cada grupo formado. Las columnas indicadas en el GROUP BY se llaman columnas de agrupación.

Ejemplo Mostrar el resultado del descuento de 30% al costo de cada instrumento.

Consulta

```
SELECT Nombre, SUM(Costo-Costo*0.3) AS Nuevo_Costo
FROM Instrumento GROUP BY Nombre
```

Resultado

Nombre	Nuevo_Costo
Batería	7000,00000
Guitarra	1400,00000
Piano	70000,0000
Teclado	3500,00000

- **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 filas o tuplas en el conjunto de resultados representan grupos). La cláusula WHERE se aplica a filas o tuplas individuales, NO a grupos.

Ejemplo 1 Se Desea saber que idCliente ha comprado más de 1 vez la tienda de Yamaha.

Consulta

```
SELECT idCliente, SUM(Cantidad) FROM Compra GROUP BY
idCliente HAVING SUM(Cantidad) > 1;
```

Resultado

idCliente	Veces
1	2

Ejemplo 2 Visualizar Aquellos instrumento que cuya cantidad sea mayor que 1.

Consulta

```
SELECT idInstrumento, SUM(Cantidad) AS Veces FROM Instrumento GROUP BY idInstrumento HAVING SUM(Cantidad) > 1;
```

Resultado

idCliente	Cantidad
1	5
3	3
4	10