

Base de Datos I

Dr. Edward Hinojosa C.

Dr. Edgar Sarmiento C.

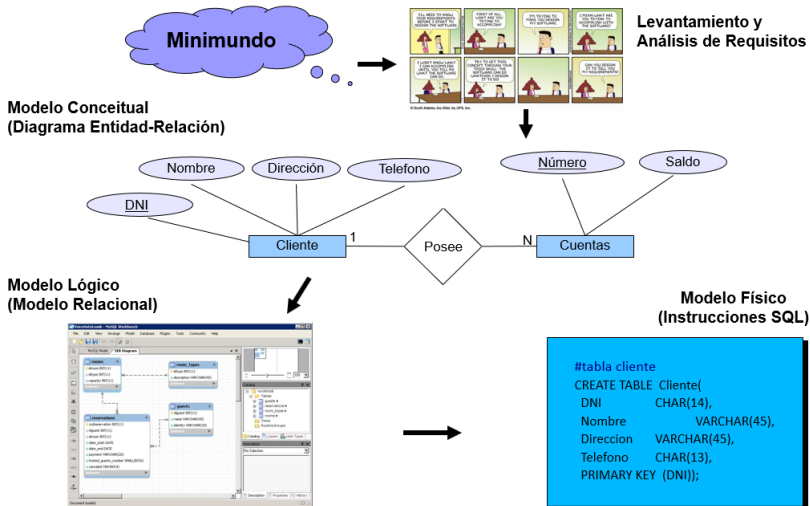
Universidad Nacional de San Agustín de Arequipa

2020/Semestre Par

Índice

- 1 Fases del Proyecto de BD
- 2 Lenguaje de Manipulación de Datos en BDR
- 3 Algebra Relacional
- 4 Cálculo Relacional
- 5 SQL - SELECT - DRL
- 6 SQL - SELECT - Funciones de Agregación

Fases del Proyecto de BD



Lenguajes de consulta en BDR

- Vamos a estudiar tres lenguajes de consulta en bases de datos relacionales:
 - Álgebra Relacional.
 - Cálculo Relacional.
 - SQL (Structured Query Language).

Algebra Relacional y Cálculo Relacional

- Álgebra y cálculo relacional son dos formalismos lógico-matemáticos para escribir consultas.
- Álgebra y cálculo relacional fueron propuesto por Codd como lenguajes de consulta en BDR. Ambos son equivalentes.
- SQL se diseñó basándose en álgebra y cálculo relacional.
- Hasta cierto punto equivalentes a SQL pero permitiendo asegurar la consistencia matemática.

Algebra Relacional y Cálculo Relacional

- Los lenguajes de consulta son los lenguajes en el que los usuarios solicitan información de la base de datos.
- Estos lenguajes son generalmente de más alto nivel que los lenguajes de programación.
- Los lenguajes de consulta pueden clasificarse como procedimentales y no procedimentales.

Algebra Relacional y Cálculo Relacional

- Los lenguajes de consulta son los lenguajes en el que los usuarios solicitan información de la base de datos.
- Estos lenguajes son generalmente de más alto nivel que los lenguajes de programación.
- Los lenguajes de consulta pueden clasificarse como procedimentales y no procedimentales.

Algebra Relacional y Cálculo Relacional

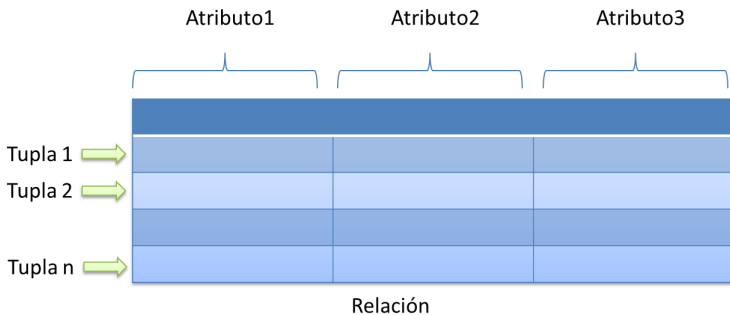
- Los lenguajes de consulta procedimentales requieren que el usuario especifique que datos se necesitan y cómo obtenerlos.
- El álgebra relacional es un lenguaje de consulta procedural.
- El álgebra relacional define operadores que funcionan sobre las tablas, por ejemplo, unión, intersección, etc, para llegar al resultado deseado o información.

Algebra Relacional y Cálculo Relacional

- Los lenguajes de consulta no procedimentales requieren que el usuario especifique que datos se necesitan sin especificar cómo obtenerlos.
- El cálculo relacional es un lenguaje de consulta no procedimental.
- El cálculo relacional define el resultado o información a ser obtenida sin dar un procedimiento específico para obtener dicha información.

Algebra Relacional

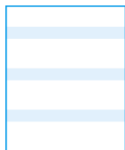
- Álgebra Relacional se define como un conjunto de operaciones que se ejecutan sobre las relaciones (tablas) para obtener un resultado, el cual es otra relación.



Algebra Relacional

- Las operaciones se pueden dividir en dos:
 - Operaciones tradicionales: unión, intersección, diferencia y producto cartesiano.
 - Operaciones especiales: selección, proyección, entre otras.

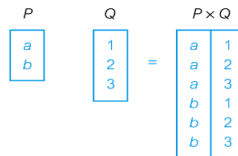
Algebra Relacional



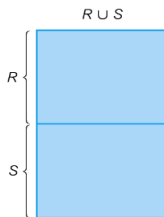
(a) Selection



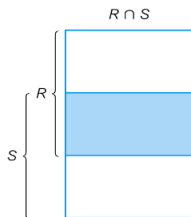
(b) Projection



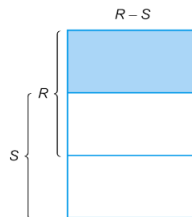
(c) Cartesian product



(d) Union



(e) Intersection



(f) Set difference

Algebra Relacional

Empleado:

CodEmpleado	Nombres	PrimerApellido	SegundoApellido	FechaNacimiento	Direccion	Sexo	Salario	Supervisor	CodDepto
123456789	Juan	Perez	Rodriguez	1965-01-09	Calle Numero A 1	M	300	333445555	5
333445555	Frank	Velazquez	Flores	1955-12-08	Calle Numero B 2	M	4000	888665555	5
999887777	Alice	Jimenez	Portugal	1968-07-19	Calle Numero C 3	F	2500	987654321	4
987654321	Luisa	Santos	Ferrel	1951-06-20	Calle Numero D 4	F	430	888665555	4
666884444	Pedro	Lima	Maldonado	1952-09-15	Calle Numero E 5	M	1200	333445555	5
453453453	Daniela	Acco	Olivares	1962-07-31	Calle Numero F 6	F	2500	333445555	5
987987987	Mateo	Vela	Marruecos	1979-03-29	Calle Numero H 7	M	2500	987654321	4
888665555	Francisco	Linares	Gomez	1957-11-10	Calle Numero I 8	M	5500	NULL	1

Departamento:

CodDepto	NombreDepto	CodGerente	FechaInicioGerencia
1	Direccion	888665555	2001-06-19
4	Administracion	987654321	1995-01-01
5	Investigacion	333445555	1998-05-22

Localizacion_Depto:

CodDepto	Localizacion
1	Lima
4	Arequipa
5	Puno
5	Trujillo
5	Cuzco

Algebra Relacional

Trabaja_en:

CodEmpleado	CodProyecto	Horas
123456789	1	32
123456789	2	7
666884444	3	40
453453453	1	20
453453453	2	20
333445555	2	10
333445555	3	10
333445555	10	10
333445555	20	10
999887777	30	30
999887777	10	10
987987987	10	35
987987987	30	5
987654321	30	20
987654321	20	15
888665555	20	NULL

Proyecto:

CodProyecto	NombreProyecto	Localizacion	CodDepto
1	Proyecto X	Puno	5
2	Proyecto Y	Trujillo	5
3	Proyecto Z	Cuzco	5
10	Proyecto A	Arequipa	4
20	Proyecto B	Lima	1
30	Proyecto C	Arequipa	4

Dependiente:

CodEmpleado	NomDependiente	Sexo	FechNacimiento	Parentesto
333445555	Ana	F	1976-04-03	Hija
333445555	Victor	M	1973-10-25	Hijo
333445555	Juana	F	1958-05-05	Conyugue
987654321	Igor	M	1952-02-29	Conyugue
123456789	Michael	M	1988-01-21	Hijo
123456789	Anabel	F	1998-12-31	Hija
123456789	Elizabeth	F	1957-05-05	Conyugue

Algebra Relacional: Selección

- La operación de Selección es utilizada para seleccionar un conjunto de tuplas de una relación:

$$\sigma_{\langle cond \rangle}(\langle R \rangle)$$

- Donde $\langle cond \rangle$ es una condición de selección y $\langle R \rangle$ es el nombre de una relación.

Algebra Relacional: Selección

- Ejemplo: Seleccionar todos los empleados que trabajan en el departamento 5.

$$\sigma_{CodDepto=5}(Empleado)$$

CodEmpleado	Nombres	PrimerApellido	SegundoApellido	FechaNacimiento	Direccion	Sexo	Salario	Supervisor	CodDepto
123456789	Juan	Perez	Rodriguez	1965-01-09	Calle Numero A 1	M	300	333445555	5
333445555	Frank	Velazquez	Flores	1955-12-08	Calle Numero B 2	M	4000	888665555	5
666884444	Pedro	Lima	Maldonado	1952-09-15	Calle Numero E 5	M	1200	333445555	5
453453453	Daniela	Acco	Olivares	1962-07-31	Calle Numero F 6	F	2500	333445555	5

Algebra Relacional: Selección

- Es una operación unitaria (realizada en una única relación).
- El grado (número de atributos) de la relación resultante es el mismo de la relación original.
- Se puede combinar un conjunto de operaciones de Selección en una única operación de Selección.
- La operación de selección es conmutativa:

$$\sigma_{\langle cond1 \rangle}(\sigma_{\langle cond2 \rangle}(\langle R \rangle)) = \sigma_{\langle cond2 \rangle}(\sigma_{\langle cond1 \rangle}(\langle R \rangle))$$

Algebra Relacional: Proyección

- La operación de Proyección es utilizada para seleccionar un conjunto de atributos de una relación:

$$\pi_{\langle atributos \rangle}(\langle R \rangle)$$

- Donde $\langle atributos \rangle$ es una lista de atributos dentro de los atributos de la relación R y $\langle R \rangle$ es el nombre de un relación.

Algebra Relacional: Proyección

- Ejemplo: Listar el nombre, primer apellido y el salario de todos los empleados.

$\pi_{\text{Nombres, PrimerApellido, Salario}}(\text{Empleado})$

Nombres	PrimerApellido	Salario
Juan	Perez	300
Frank	Velazquez	4000
Alice	Jimenez	2500
Luisa	Santos	430
Pedro	Lima	1200
Daniela	Acco	2500
Mateo	Vela	2500
Francisco	Linares	5500

Algebra Relacional: Proyección

- Es una operación unitaria (realizada en una única relación).
- Caso la lista de atributos incluya solamente atributos que no sean claves de R , es posible que ocurran tuplas duplicadas:
 - La operación Proyección elimina tuplas duplicadas de tal forma que el resultado sea una relación válida.
 - Con ello, el número de tuplas en la relación resultante es siempre menor o igual al número de tuplas de la relación R .
- La operación de proyección no es conmutativa.

Algebra Relacional: Secuencia de Operaciones

- Es común aplicar diversas operaciones del álgebra relacional, una después de la otra (secuencia de operaciones).
- Se puede escribir las operaciones en la forma de una única expresión o aplicar una operación secuencialmente, creando relaciones de resultado intermedio, en ese último caso, se debe nombrar las relaciones implicadas.

Algebra Relacional: Secuencia de Operaciones

- Ejemplo: Listar el nombre, primer apellido y el salario de todos los empleados que trabajan en el Departamento número 5.

$$\pi_{Nombres, PrimerApellido, Salario}(\sigma_{CodDepa=5}(Empleado))$$

Equivalente a:

$$Dep5Empleado \leftarrow \sigma_{CodDepa=5}(Empleado)$$

$$Resultado \leftarrow \pi_{Nombres, PrimerApellido, Salario}(Dep5Empleado)$$

Nombres	PrimerApellido	Salario
Juan	Perez	300
Frank	Velazquez	4000
Pedro	Lima	1200
Daniela	Acco	2500

Algebra Relacional: Secuencia de Operaciones

- Se puede utilizar la técnica de secuencia de operaciones para renombrar los atributos en la relaciones intermedias y del resultado: debemos listar los nombres de los nuevos atributos entre paréntesis juntamente con los nombres de las nuevas relaciones:

$$Dep5Empleado \leftarrow \sigma_{CodDepa=5}(Empleado)$$

$$Resultado(Nom, ApePaterno, Sueldo) \leftarrow$$

$$\pi_{Nombres, PrimerApellido, Salario}(Dep5Empleado)$$

Algebra Relacional: Teoría de Conjuntos

- El álgebra relacional posee un grupo patrón de operaciones matemáticas sobre conjuntos:
 - Las operaciones son binarias, es decir, envuelven dos relaciones.
 - Para algunas operaciones, las relaciones deben poseer el mismo tipo de tuplas, siendo consideradas compatibles para la unión, intersección y diferencia.
- Dos relaciones $R(A_1, A_2, \dots, A_n)$ y $S(B_1, B_2, \dots, B_n)$ son compatibles para la unión, intersección y diferencia si poseen el mismo grado "n" y si $dom(A_i) = dom(B_i)$ para $1 \leq i \leq n$.

Algebra Relacional: Teoría de Conjuntos

- Las operaciones de teoría de conjuntos que exigen relaciones compatibles son:
 - Unión: Denotada por $R \cup S$, genera una relación que incluye todas las tuplas que están en R o en S o en ambas.
 - Intersección: Denotada por $R \cap S$, genera una relación que incluye todas las tuplas que están en R y en S .
 - Diferencia: Denotada por $R - S$, genera una relación que incluye todas las tuplas que están en R , pero no en S .

Algebra Relacional: Teoría de Conjuntos

- La relación resultante de las operaciones posee los mismos nombres de atributos de la primera relación (R) envueltas en las operaciones.
- Las operaciones de unión e intersección son conmutativas y asociativas.

$$R \cup S = S \cup R \quad R \cap S = S \cap R$$

$$R \cup (S \cap T) = (R \cup S) \cap T \quad R \cap (S \cup T) = (R \cap S) \cup T$$

Algebra Relacional: Ejemplo - Unión

- Listar el código de todos los empleados que trabajan en el departamento 5 o que supervisan a algún empleado que trabaje en el departamento 5.

$$Dep5Empleados \leftarrow \sigma_{codDepto=5}(Empleado)$$

$$Resultado1 \leftarrow \pi_{CodEmpleado}(Dep5Empleados)$$

$$Resultado2(CodEmpleado) \leftarrow \pi_{supervisor}(Dep5Empleados)$$

$$Resultado(CodEmpleado) \leftarrow Resultado1 \cup Resultado2$$

CodEmpleado	Nombres	PrimerApellido	SegundoApellido	FechaNacimiento	Direccion	Sexo	Salario	Supervisor	CodDepto
123456789	Juan	Perez	Rodriguez	1965-01-09	Calle Numero A 1	M	300	333445555	5
333445555	Frank	Velazquez	Flores	1955-12-08	Calle Numero B 2	M	4000	888665555	5
999887777	Alice	Jimenez	Portugal	1968-07-19	Calle Numero C 3	F	2500	987654321	4
987654321	Luisa	Santos	Ferrel	1951-06-20	Calle Numero D 4	F	430	888665555	4
666884444	Pedro	Lima	Maldonado	1952-09-15	Calle Numero E 5	M	1200	333445555	5
453453453	Daniela	Acco	Olivares	1962-07-31	Calle Numero F 6	F	2500	333445555	5
987987987	Mateo	Vela	Marruecos	1979-03-29	Calle Numero H 7	M	2500	987654321	4
888665555	Francisco	Linares	Gomez	1957-11-10	Calle Numero I 8	M	5500	NULL	1

Algebra Relacional: Teoría de Conjuntos

- La operación de conjunto binaria Producto Cartesiano, representada por \times , es utilizada para combinar tuplas de dos relaciones de forma combinatoria.
- Las relaciones no necesitan ser compatibles como para la unión.
- El resultado $R(A_1, A_2, \dots, A_n) \times S(B_1, B_2, \dots, B_m)$ es una relación $Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$ con $n + m$ atributos.
- La relación Q posee una tupla para cada combinación de tuplas de las relaciones implicadas: Si R posee n_R tuplas y S posee n_S tuplas, entonces Q poseerán $n_R * n_S$ tuplas.

Algebra Relacional: Teoría de Conjuntos

- No es una operación muy usual porque genera tuplas que no hacen sentido. Siempre se usa seguida por una Selección que combina valores de atributos en las relaciones implicadas.
- Ejemplo: Mostrar el Nombre, Apellido Paterno y Código de todos los empleados de sexo femenino que tengan dependientes:

$$EmpMujeres \leftarrow \sigma_{sexo='F'}(Empleado)$$

$$NomEmpleado(Nom, ApePaterno, CodigoEmp)$$

$$\leftarrow \pi_{Nombres, PrimerApellido, CodEmpleado}(EmpMujeres)$$

$$DepenEmple \leftarrow NomEmpleado \times Dependiente$$

$$DepenCorrectos \leftarrow \sigma_{codigoEmp=CodEmpleado}(DepenEmple)$$

$$Resultado \leftarrow \pi_{Nomb, ApePaterno, CodigoEmp}(DepenCorrectos)$$

Algebra Relacional: Teoría de Conjuntos

CodEmpleado	Nombres	PrimerApellido	SegundoApellido	FechaNacimiento	Direccion	Sexo	Salario	Supervisor	CodDepto
123456789	Juan	Perez	Rodriguez	1965-01-09	Calle Numero A 1	M	300	333445555	5
333445555	Frank	Velazquez	Flores	1955-12-08	Calle Numero B 2	M	4000	888665555	5
999887777	Alice	Jimenez	Portugal	1968-07-19	Calle Numero C 3	F	2500	987654321	4
987654321	Luisa	Santos	Ferrel	1951-06-20	Calle Numero D 4	F	430	888665555	4
666884444	Pedro	Lima	Maldonado	1952-09-15	Calle Numero E 5	M	1200	333445555	5
453453453	Daniela	Acco	Olivares	1962-07-31	Calle Numero F 6	F	2500	333445555	5
987987987	Mateo	Vela	Marruecos	1979-03-29	Calle Numero H 7	M	2500	987654321	4
888665555	Francisco	Linares	Gomez	1957-11-10	Calle Numero I 8	M	5500	NULL	1

CodEmpleado	NomDependiente	Sexo	FechNacimiento	Parentesto
333445555	Ana	F	1976-04-03	Hija
333445555	Victor	M	1973-10-25	Hijo
333445555	Juana	F	1958-05-05	Conyugue
987654321	Igor	M	1952-02-29	Conyugue
123456789	Michael	M	1988-01-21	Hijo
123456789	Anabel	F	1998-12-31	Hija
123456789	Elizabeth	F	1957-05-05	Conyugue

Algebra Relacional: Teoría de Conjuntos

- Una vez que la operación Producto Cartesiano, seguida de la operación Selección, es usada con frecuencia, fue definida una operación especial, denominada Yunción, para especificar tal secuencia como una única operación.
- La operación Yunción es utilizada para combinar tuplas relacionadas de dos relaciones en un única tupla:

$$R \bowtie_{\langle cond \rangle} S$$

- Donde R y S son relaciones y $\langle cond \rangle$ es una condición de yunción entre las relaciones.

Algebra Relacional: Teoría de Conjuntos

- $R(A_1, A_2, \dots, A_n) \bowtie_{\langle \text{cond} \rangle} S(B_1, B_2, \dots, B_m)$ es una relación $Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$ con $n + m$ atributos.
- La relación Q posee una tupla para cada combinación de tuplas de las relaciones implicadas, siempre que la combinación satisfaga la condición de yunción.
- Una condición general de yunción es: $\langle \text{cond}_1 \rangle$ y $\langle \text{cond}_2 \rangle$ y ... y ... $\langle \text{cond}_N \rangle$, donde cada condición es de la forma $A_i \theta B_j$:
- A_i es el atributo de R , B_j es el atributo de S del mismo dominio de A_i , y θ es un operador de comparación $=, <, >, \leq, \geq, \neq$.

Algebra Relacional: Teoría de Conjuntos

- Ejemplo: Mostrar el Nombre, Apellido Paterno y Código de todos los empleados de sexo femenino que tengan dependientes:

$$EmpMujeres \leftarrow \sigma_{\text{sexo}='F'}(Empleado)$$

$$NomEmpleado(Nom, ApePaterno, CodigoEmp) \leftarrow \pi_{Nombres, PrimerApellido, CodigoEmp}(Empleado)$$

$$DepenCorrectos \leftarrow NomEmpleado \bowtie_{\text{codigoEmp}=\text{CodEmpleado}} (Dependiente)$$

$$Resultado \leftarrow \pi_{Nomb, ApePaterno, CodigoEmp}(DepenCorrectos)$$

Algebra Relacional: Teoría de Conjuntos

- La operación Yunción más común, denominada Equiyunción, envuelve apenas condiciones de yunción con comparaciones de igualdad.
- Una equiyunción donde dos atributos de la comparación tienen el mismo nombre es llamada de Yunción Natural, siendo definida por $*$; en ese caso, apenas uno de los atributos de la comparación aparece en la relación resultante y la condición de yunción no es especificada.
- Ejemplo: Listar, para cada empleado del sexo femenino, los nombres de sus dependientes:

Algebra Relacional: Teoría de Conjuntos

$$EmpMujeres \leftarrow \sigma_{\text{sexo}='F'}(Empleado)$$
$$DepenCorrectos \leftarrow EmpMujeres * Dependiente$$
$$Resultado \leftarrow \pi_{Nomb, ApePaterno, CodigoEmp}(DepenCorrectos)$$

Cálculo Relacional

- Al igual que el Álgebra Relacional (AR), el Cálculo Relacional de Tuplas (CRT) es un lenguaje de consulta asociado al Modelo Relacional.
- CRT es un lenguaje declarativo o no procedimental: Describe cuáles tuplas se deben devolver pero no como se calculan.
- Cualquier consulta escrita en AR puede ser expresada en CRT y viceversa, es decir, ambos tienen la misma expresividad.

Cálculo Relacional

- El CRT posee una base firme en la lógica matemática.
- El CRT se basa sobre la especificación de variables tupla.
- Cada variable tupla se extiende a lo largo de una relación y puede tomar como valor cualquier tupla de esa relación.
- El lenguaje de consulta estándar (SQL) tiene muchos de sus fundamentos en el CRT. Veremos ejemplos de selección, proyección y yunción (también podemos usar unión, intersección y diferencia pero no veremos ejemplos de ello).

Cálculo Relacional

- La Fórmula General de Expresión puede ser definida de la siguiente manera:

$$\{t|F(t)\}$$

Conjunto de tuplas t tal que $F(t)$ es verdadera.

- Considere la siguiente BD para expresar fórmulas del CRT:

Cálculo Relacional

PROVEEDORES

P#	PNOMBRE	CATEGORIA	CIUDAD
P1	CARLOS	20	SEVILLA
P2	JUAN	10	MADRID
P3	JOSE	30	SEVILLA
P4	INMA	20	SEVILLA
P5	EVA	30	CACERES

COMPONENTES

C#	CNOMBRE	COLOR	PESO	CIUDAD
C1	X3A	ROJO	12	SEVILLA
C2	B85	VERDE	17	MADRID
C3	C4B	AZUL	17	MALAGA
C4	C4B	ROJO	14	SEVILLA
C5	VT8	AZUL	12	MADRID
C6	C30	ROJO	19	SEVILLA

Cálculo Relacional

ENVIOS

P#	C#	T#	CANTIDAD
P1	C1	T1	200
P1	C1	T4	700
P2	C3	T1	400
P2	C3	T2	200
P2	C3	T3	200
P2	C3	T4	500
P2	C3	T5	600
P2	C3	T6	400
P2	C3	T7	800
P2	C5	T2	100
P3	C3	T1	200
P3	C4	T2	500
P4	C6	T3	300
P4	C6	T7	300
P5	C2	T2	200
P5	C2	T4	100
P5	C5	T4	500
P5	C5	T7	100
P5	C6	T2	200
P5	C1	T4	100
P5	C3	T4	200
P5	C4	T4	800
P5	C5	T5	400
P5	C6	T4	500

ARTICULOS

T#	TNOMBRE	CIUDAD
T1	CLASIFICADORA	MADRID
T2	PERFORADORA	MALAGA
T3	LECTORA	CACERES
T4	CONSOLA	CACERES
T5	MEZCLADORA	SEVILLA
T6	TERMINAL	BARCELONA
T7	CINTA	SEVILLA

Cálculo Relacional

- **PROVEEDORES** .- Representa los datos de proveedores de componentes para la fabricación de artículos y su ciudad de residencia.
- **COMPONENTES**.- Indica la información de piezas utilizadas en la fabricación de diferentes artículos, indicándose el lugar de fabricación de dichos componentes.
- **ARTICULOS**.- Información sobre los diferentes artículos que se fabrican y el lugar de montaje del mismo.
- **ENVIOS**.- Suministros realizados por los diferentes proveedores de determinadas cantidades de componentes asignadas para la elaboración del artículo correspondiente.

Cálculo Relacional: Relación de Intervalo (Range)

- Ejemplo: Listar todas las tuplas de la tabla proveedores:

$$\{t \mid t \in \text{proveedores}\}$$

P#	PNOMBRE	CATEGORIA	CIUDAD
P1	CARLOS	20	SEVILLA
P2	JUAN	10	MADRID
P3	JOSE	30	SEVILLA
P4	INMA	20	SEVILLA
P5	EVA	30	CACERES

Cálculo Relacional: Selección

- Ejemplo: Listar las tuplas de la tabla artículos que tengan esten en la Ciudad Caceres.

$$\{t \mid t \in \text{articulos} \wedge t.Ciudad = 'Caceres'\}$$

T#	TNOMBRE	CIUDAD
T3	LECTORA	CACERES
T4	CONSOLA	CACERES

Cálculo Relacional: Proyección

- Ejemplo: Mostrar las columnas PNombre y Categoría de la tabla proveedores.

$$\{t | \exists s \in \text{proveedores} (t.PNombre = s.PNombre \wedge t.Categoría = s.Categoría)\}$$

PNOMBRE	CATEGORIA
CARLOS	20
JUAN	10
JOSE	30
INMA	20
EVA	30

Cálculo Relacional: Yunción

- Ejemplo: Mostrar el nombre del proveedor y su ciudad de todos los proveedores que envían el componente X3A. (No se eliminan resultados iguales).

$$\{t | \exists pr \in \text{proveedores} \exists co \in \text{componentes} \exists en \in \text{envios} \\ (t.PNombre = pr.PNombre \wedge t.Ciudad = pr.Ciudad \\ \wedge pr.P = en.P \wedge co.C = en.C \wedge co.CNombre = 'X3A')\}$$

PNOMBRE	CIUDAD
CARLOS	SEVILLA
CARLOS	SEVILLA
EVA	CACERES

SQL - Data Retrieval Language (DRL) - SELECT

- Para obtener información de una BDR podemos utilizar el comando SELECT

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [, select_expr] ...
[into_option]
[FROM table_references
  [PARTITION partition_list]]
[WHERE where_condition]
[GROUP BY {col_name | expr | position}, ... [WITH ROLLUP]]
[HAVING where_condition]
[WINDOW window_name AS (window_spec)
  [, window_name AS (window_spec)] ...]
[ORDER BY {col_name | expr | position}
  [ASC | DESC], ... [WITH ROLLUP]]
[LIMIT [{offset,} row_count | row_count OFFSET offset]]
[into_option]
[FOR {UPDATE | SHARE}
  [OF tbl_name [, tbl_name] ...]
  [NOWAIT | SKIP LOCKED]
  | LOCK IN SHARE MODE]
[into_option]

into_option: {
  INTO OUTFILE 'file_name'
    [CHARACTER SET charset_name]
    export_options
  | INTO DUMPFILE 'file_name'
  | INTO var_name [, var_name] ...
}

```

SQL - SELECT - Selección - BDR Universidad

- Muestre al estudiante con código 666884444.

The screenshot shows a SQL query editor with the following code:

```

1 • USE universidad;
2 • SELECT * FROM estudiante WHERE estu_id = 666884444;
3
4
5

```

Below the query editor is a "Result Grid" showing the results of the query. The grid has the following columns: estu_id, nombres, prim_apel, segu_apel, depa_id, and prof_id. The first row shows the student with ID 666884444, named Pedro, with last names Lima and Maldonado, in department 1, and taught by professor 2. The second row shows NULL values for all columns.

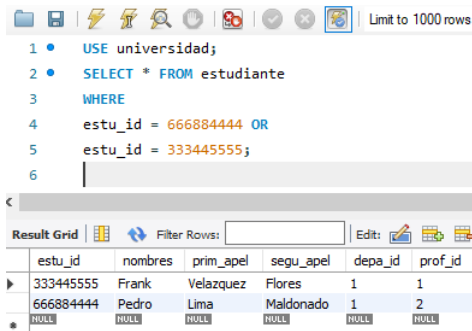
	estu_id	nombres	prim_apel	segu_apel	depa_id	prof_id
▶	666884444	Pedro	Lima	Maldonado	1	2
*	NULL	NULL	NULL	NULL	NULL	NULL

SQL - SELECT - Selección - BDR Universidad

- Como hemos visto, el comando WHERE puede utilizarse para seleccionar datos condicionalmente de una tabla.
- Esta condición puede ser una condición simple (como la vimos anteriormente), o puede ser una condición compuesta. Las condiciones compuestas están formadas por múltiples condiciones simples conectadas por AND u OR.
- No existe límites en el número de condiciones simples que pueden presentarse en una sola instrucción SQL.

SQL - SELECT - Selección - BDR Universidad

- Muestre los estudiantes con código 666884444 o 333445555



The screenshot shows a SQL IDE interface. At the top, there is a toolbar with various icons and a 'Limit to 1000 rows' button. Below the toolbar, a query is entered in a text area:

```

1 • USE universidad;
2 • SELECT * FROM estudiante
3   WHERE
4     estu_id = 666884444 OR
5     estu_id = 333445555;
6

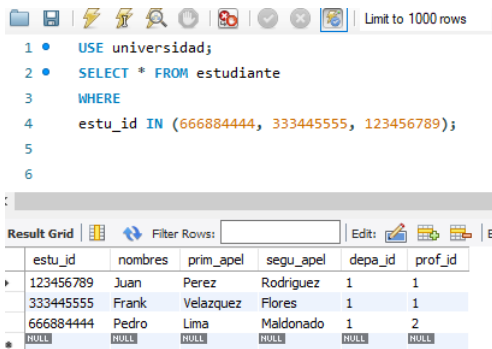
```

Below the query editor, the 'Result Grid' is displayed. It shows a table with the following data:

estu_id	nombres	prim_apel	segu_apel	depa_id	prof_id
333445555	Frank	Velazquez	Flores	1	1
666884444	Pedro	Lima	Maldonado	1	2
NULL	NULL	NULL	NULL	NULL	NULL

SQL - SELECT - Selección - BDR Universidad

- Muestre los estudiantes con código 666884444 o 333445555 O 123456789.



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```

1 • USE universidad;
2 • SELECT * FROM estudiante
3   WHERE
4     estu_id IN (666884444, 333445555, 123456789);
5
6

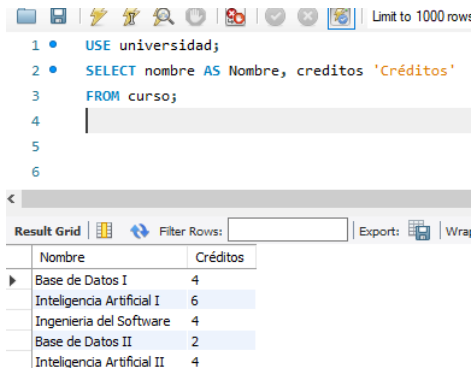
```

Below the query, the results are displayed in a table titled "Result Grid". The table has 7 columns: estu_id, nombres, prim_apel, segu_apel, depa_id, and prof_id. The results are as follows:

estu_id	nombres	prim_apel	segu_apel	depa_id	prof_id
123456789	Juan	Perez	Rodriguez	1	1
333445555	Frank	Velazquez	Flores	1	1
666884444	Pedro	Lima	Maldonado	1	2
NULL	NULL	NULL	NULL	NULL	NULL

SQL - SELECT - Proyección - BDR Universidad

- Muestre el nombre y los créditos de cada curso.



The screenshot shows a SQL query editor with the following code:

```

1 • USE universidad;
2 • SELECT nombre AS Nombre, credits 'Créditos'
3 • FROM curso;
4
5
6

```

Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The table has two columns: 'Nombre' and 'Créditos'.

Nombre	Créditos
Base de Datos I	4
Inteligencia Artificial I	6
Ingeniería del Software	4
Base de Datos II	2
Inteligencia Artificial II	4

SQL - SELECT - Proyección - BDR Universidad

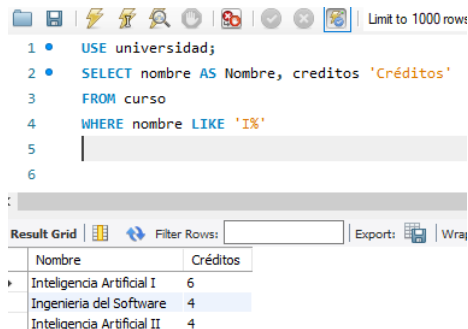
- LIKE es otro comando que se utiliza en la cláusula WHERE.
- Básicamente, LIKE nos permite hacer una búsqueda basada en un patrón en vez de especificar exactamente lo que se desea.
- Tarea: Estudiar las forma de búsqueda de textos.

SQL - SELECT - Proyección - BDR Universidad

- patrón generalmente consiste en comodines. Por ejemplo:
- 'A_Z': Toda línea que comience con 'A', otro carácter y termine con 'Z'. Por ejemplo, 'ABZ' y 'A2Z' deberían satisfacer la condición, mientras 'AKKZ' no debería (debido a que hay dos caracteres entre A y Z en vez de uno).
- 'ABC %': Todas las líneas que comienzan con 'ABC'. Por ejemplo, 'ABCD' y 'ABCABC' ambas deberían satisfacer la condición.
- '%XYZ': Todas las líneas que terminan con 'XYZ'. Por ejemplo, 'WXYZ' y 'ZZXYZ' ambas deberían satisfacer la condición.
- '%AN %': Todas las líneas que contienen el patrón 'AN' en cualquier posición. Por ejemplo, 'LOS ANGELES' y 'SAN FRANCISCO' ambos deberían satisfacer la condición.

SQL - SELECT - Proyección y Selección - BDR Universidad

- Muestre el nombre y los créditos de cada curso que empiezen con 'I'.



The screenshot shows a SQL query editor with the following query:

```

1 • USE universidad;
2 • SELECT nombre AS Nombre, credits 'Créditos'
3   FROM curso
4  WHERE nombre LIKE 'I%'
5
6

```

Below the query editor, the 'Result Grid' is displayed, showing the results of the query:

Nombre	Créditos
Inteligencia Artificial I	6
Ingenieria del Software	4
Inteligencia Artificial II	4

SQL - SELECT - BDR Universidad

- Hemos visto cómo obtener datos de una tabla utilizando los comandos SELECT y WHERE.
- Con frecuencia, necesitamos mostrar el resultado en un orden particular. Esto podría ser en orden ascendente, en orden descendente, o podría basarse en valores numéricos o de texto. En tales casos, podemos utilizar el comando ORDER BY.

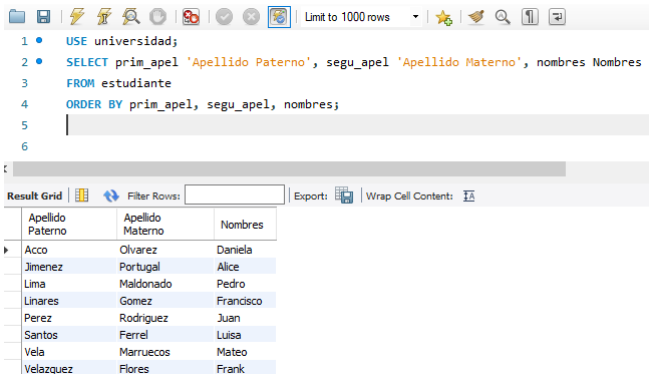
SQL - SELECT - BDR Universidad

- Hemos visto cómo obtener datos de una tabla utilizando los comandos SELECT y WHERE.
- Con frecuencia, necesitamos mostrar el resultado en un orden particular. Esto podría ser en orden ascendente, en orden descendente, o podría basarse en valores numéricos o de texto. En tales casos, podemos utilizar el comando ORDER BY.
- La cláusula ORDER BY ASC significa que los resultados se mostrarán en orden ascendente, y DESC significa que los resultados se mostrarán en orden descendente. Si no se especifica ninguno, la configuración predeterminada es ASC.
- Es posible ordenar por más de una columna.

SQL - SELECT - Proyección y Selección - BDR

Universidad

- Muestre el nombre, primer apellido y segundo apellido de todos los estudiantes ordenados de forma alfabética según el primer apellido, seguido del segundo apellido y por último el nombre.



```

1 • USE universidad;
2 • SELECT prim_apel 'Apellido Paterno', segu_apel 'Apellido Materno', nombres Nombres
3 FROM estudiante
4 ORDER BY prim_apel, segu_apel, nombres;
5
6

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: ☐

Apellido Paterno	Apellido Materno	Nombres
Acco	Olvarez	Daniela
Jimenez	Portugal	Alice
Lima	Maldonado	Pedro
Linares	Gomez	Francisco
Perez	Rodriguez	Juan
Santos	Ferrel	Luisa
Vela	Marruecos	Mateo
Velazquez	Flores	Frank

SQL - SELECT - Yunción

- En SQL la yunción es representada por el comando JOIN el cual se utiliza para combinar filas de dos o más tablas, en función a una columna relacionada entre ella.
- Tenemos diferentes tipos:
 - Cross Join (FULL JOIN)
 - Inner Join (Más utilizado)
 - Left Join
 - Right Join

SQL - SELECT - Yunción

- Para demostrar el funcionamiento de los métodos o comandos de yunción (joins) vamos a crear dos tablas entre las cuales se debe realizar algún relacionamiento para “cruzar” datos.
- Veremos algunos ejemplos en relación a la Base de Datos Universidad.

SQL - SELECT - Cross Join - BDR Universidad

Limit to 1000 rows

```

1 • USE universidad;
2 • SELECT c.nombre Curso, d.nombre Departamento
3 FROM curso c
4 CROSS JOIN
5 departamento d;
6

```

Result Grid | Filter Rows: | Export: | Wri

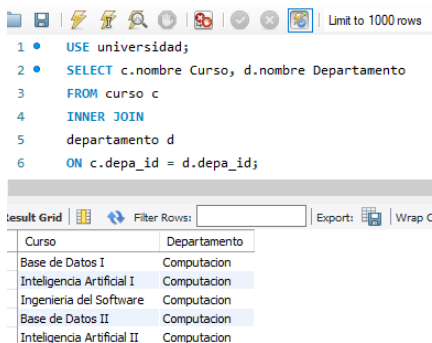
Curso	Departamento
Base de Datos I	Computacion
Base de Datos I	Ingenieria
Base de Datos I	Biomedicas
Inteligencia Artificial I	Computacion
Inteligencia Artificial I	Ingenieria
Inteligencia Artificial I	Biomedicas
Ingenieria del Software	Computacion
Ingenieria del Software	Ingenieria
Ingenieria del Software	Biomedicas
Base de Datos II	Computacion
Base de Datos II	Ingenieria
Base de Datos II	Biomedicas
Inteligencia Artificial II	Computacion
Inteligencia Artificial II	Ingenieria
Inteligencia Artificial II	Biomedicas

SQL - SELECT - INNER JOIN

- El comando Inner Join (Yunción Interna) realiza la yunción de tablas basándose en un condición de yunción (o punto en común).
- Veremos varios ejemplos de este comando.

SQL - SELECT - INNER JOIN

- Muestre el código y nombre de los cursos, así como los departamentos al que pertenece cada curso.



The screenshot shows a SQL query editor with the following code:

```

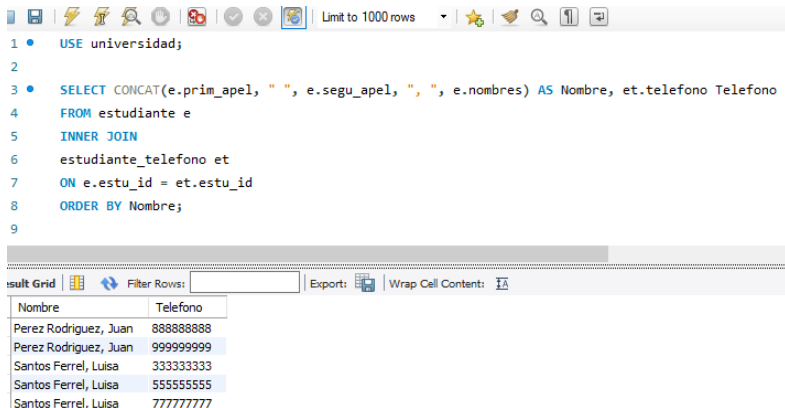
1 • USE universidad;
2 • SELECT c.nombre Curso, d.nombre Departamento
3   FROM curso c
4   INNER JOIN
5   departamento d
6   ON c.depa_id = d.depa_id;
  
```

Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The grid has two columns: 'Curso' and 'Departamento'. The results are as follows:

Curso	Departamento
Base de Datos I	Computacion
Inteligencia Artificial I	Computacion
Ingenieria del Software	Computacion
Base de Datos II	Computacion
Inteligencia Artificial II	Computacion

SQL - SELECT - INNER JOIN

- Muestre a todos los estudiantes con sus teléfonos (en caso tengan teléfono).



The screenshot shows a SQL query editor with the following code:

```

1 • USE universidad;
2
3 • SELECT CONCAT(e.prim_apel, " ", e.segu_apel, ", ", e.nombres) AS Nombre, et.telefono Telefono
4 FROM estudiante e
5 INNER JOIN
6 estudiante_telefono et
7 ON e.estu_id = et.estu_id
8 ORDER BY Nombre;
9

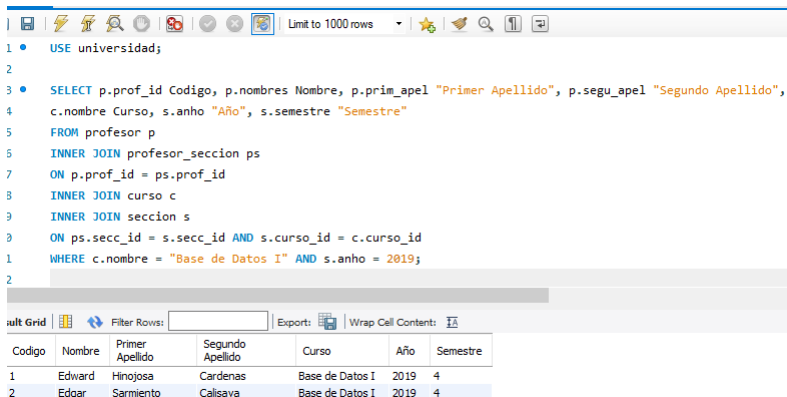
```

Below the query editor, the results are displayed in a table with the following data:

Nombre	Telefono
Perez Rodriguez, Juan	888888888
Perez Rodriguez, Juan	999999999
Santos Ferrel, Luisa	333333333
Santos Ferrel, Luisa	555555555
Santos Ferrel, Luisa	777777777

SQL - SELECT - INNER JOIN

- Muestre todos los profesores que han dictado el curso de Base de Datos I en el año 2019.



The screenshot shows a SQL IDE interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, a SQL query is entered in a text area. The query uses multiple INNER JOINs to filter professors who have taught 'Base de Datos I' in 2019. Below the query editor, there's a 'Result Grid' section with a table of results. The table has columns for Codigo, Nombre, Primer Apellido, Segundo Apellido, Curso, Año, and Semestre. Two rows of data are displayed.

```

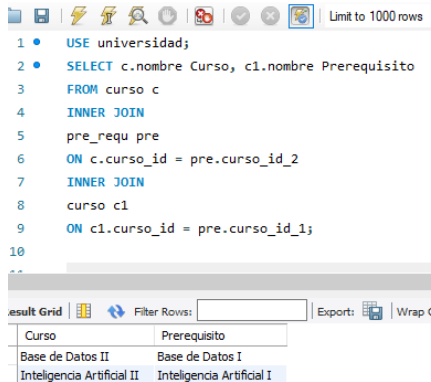
1 • USE universidad;
2
3 • SELECT p.prof_id Codigo, p.nombres Nombre, p.prim_apel "Primer Apellido", p.segu_apel "Segundo Apellido",
4     c.nombre Curso, s.anho "Año", s.semestre "Semestre"
5 FROM profesor p
6 INNER JOIN profesor_seccion ps
7 ON p.prof_id = ps.prof_id
8 INNER JOIN curso c
9 INNER JOIN seccion s
10 ON ps.secc_id = s.secc_id AND s.curso_id = c.curso_id
11 WHERE c.nombre = "Base de Datos I" AND s.anho = 2019;
12

```

Codigo	Nombre	Primer Apellido	Segundo Apellido	Curso	Año	Semestre
1	Edward	Hinojosa	Cardenas	Base de Datos I	2019	4
2	Edgar	Sarmiento	Calisaya	Base de Datos I	2019	4

SQL - SELECT - INNER JOIN

- Muestre los cursos con sus pre-requisitos (lo cursos que tienen pre-requisito).



The screenshot shows a SQL IDE interface. At the top, there is a toolbar with various icons and a text 'Limit to 1000 rows'. Below the toolbar, a SQL query is written in a text editor. The query is as follows:

```

1 • USE universidad;
2 • SELECT c.nombre Curso, c1.nombre Prerequisito
3 FROM curso c
4 INNER JOIN
5 pre_requ pre
6 ON c.curso_id = pre.curso_id_2
7 INNER JOIN
8 curso c1
9 ON c1.curso_id = pre.curso_id_1;
10

```

Below the query editor, there is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap' checkbox. The results are displayed in a table with two columns: 'Curso' and 'Prerequisito'.

Curso	Prerequisito
Base de Datos II	Base de Datos I
Inteligencia Artificial II	Inteligencia Artificial I

SQL - SELECT - INNER JOIN

- Muestre los cursos, año y semestre donde el alumno con código 987654321 se ha matriculado.

```

1 • USE universidad;
2 • SELECT e.estu_id 'Código', e.nombres Nombres, e.prim_apel 'Apellido Paterno', e.segu_apel 'Apellido Materno',
3     c.nombre Curso, s.anho 'Año', s.semestre 'Semestre'
4 FROM estudiante e
5 INNER JOIN
6     estudiante_seccion es
7 ON e.estu_id = es.estu_id
8 INNER JOIN
9     curso c
10 INNER JOIN
11     seccion s
12 ON es.secc_id = s.secc_id AND s.curso_id = c.curso_id
13 WHERE e.estu_id = 987654321
14

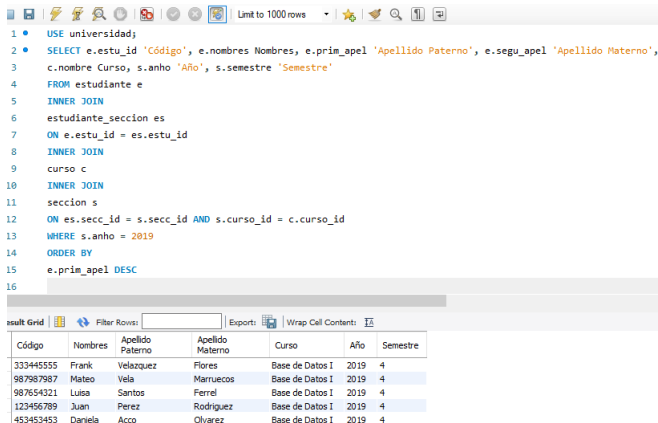
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: ☐

Código	Nombres	Apellido Paterno	Apellido Materno	Curso	Año	Semestre
987654321	Luisa	Santos	Ferrel	Base de Datos I	2019	4
987654321	Luisa	Santos	Ferrel	Ingeniería del Software	2020	6
987654321	Luisa	Santos	Ferrel	Base de Datos II	2020	5

SQL - SELECT - INNER JOIN

- Muestre los cursos, año y semestre para todos los alumnos matriculados en el año 2019 ordenados de forma descendente según su primer apellido.



```

1 • USE universidad;
2 • SELECT e.estu_id 'Código', e.nombres Nombres, e.prim_apel 'Apellido Paterno', e.segu_apel 'Apellido Materno',
3   c.nombre Curso, s.año 'Año', s.semestre 'Semestre'
4   FROM estudiante e
5   INNER JOIN
6     estudiante_seccion es
7   ON e.estu_id = es.estu_id
8   INNER JOIN
9     curso c
10  INNER JOIN
11    seccion s
12  ON es.secc_id = s.secc_id AND s.curso_id = c.curso_id
13  WHERE s.año = 2019
14  ORDER BY
15    e.prim_apel DESC
16

```

Result Grid

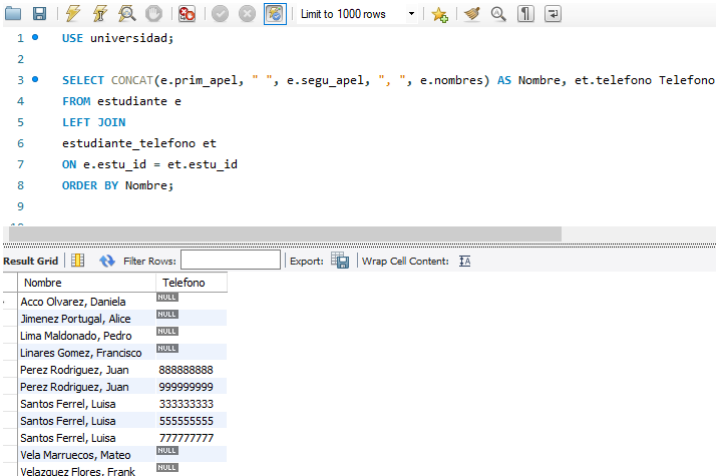
Código	Nombres	Apellido Paterno	Apellido Materno	Curso	Año	Semestre
333445555	Frank	Velazquez	Flores	Base de Datos I	2019	4
987987987	Mateo	Vela	Marruecos	Base de Datos I	2019	4
987654321	Luisa	Santos	Ferrel	Base de Datos I	2019	4
123456789	Juan	Perez	Rodriguez	Base de Datos I	2019	4
453453453	Daniela	Acco	Olvarez	Base de Datos I	2019	4

SQL - SELECT - LEFT JOIN

- El comando LEFT JOIN (Yunción a la izquierda) hará yunción entre las tablas dando de preferencia a los registro de la tabla más a la izquierda en el código, mostrando todos sus registros.
- Cuando no hubiera correspondencia será mostrado el valor NULL.

SQL - SELECT - LEFT JOIN

- Muestre a todos los alumnos con sus teléfonos.



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Limit to 1000 rows' dropdown. The SQL editor contains the following query:

```

1 • USE universidad;
2
3 • SELECT CONCAT(e.prim_apel, " ", e.segu_apel, " ", e.nombres) AS Nombre, et.telefono Telefono
4 FROM estudiante e
5 LEFT JOIN
6 estudiante_telefono et
7 ON e.estu_id = et.estu_id
8 ORDER BY Nombre;
9
10

```

Below the editor is the 'Result Grid' section, which includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with two columns: 'Nombre' and 'Telefono'.

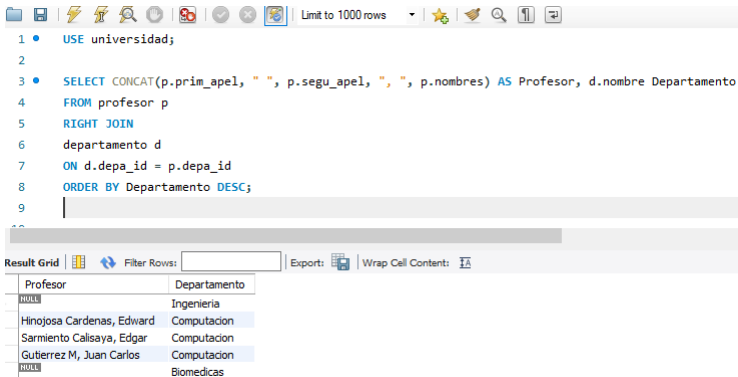
Nombre	Telefono
Acco Olvarez, Daniela	NULL
Jimenez Portugal, Alice	NULL
Lima Maldonado, Pedro	NULL
Linares Gomez, Francisco	NULL
Perez Rodriguez, Juan	888888888
Perez Rodriguez, Juan	999999999
Santos Ferrel, Luisa	333333333
Santos Ferrel, Luisa	555555555
Santos Ferrel, Luisa	777777777
Vela Marruecos, Mateo	NULL
Velazquez Flores, Frank	NULL

SQL - SELECT - RIGHT JOIN

- El comando RIGHT JOIN (Yunción a la derecha) hará yunción entre las tablas dando de preferencia a los registro de la tabla más a la derecha en el código, mostrando todos sus registros.
- Cuando no hubiera correspondencia será mostrado el valor NULL.

SQL - SELECT - RIGHT JOIN

- Muestre a todos los profesores por departamento.



The screenshot shows a SQL IDE interface. At the top, there is a toolbar with various icons and a dropdown menu set to "Limit to 1000 rows". Below the toolbar, the SQL query is written in a text editor:

```

1 • USE universidad;
2
3 • SELECT CONCAT(p.prim_apel, " ", p.segu_apel, ", ", p.nombres) AS Profesor, d.nombre Departamento
4 FROM profesor p
5 RIGHT JOIN
6 departamento d
7 ON d.depa_id = p.depa_id
8 ORDER BY Departamento DESC;
9

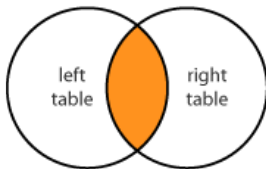
```

Below the query editor, the "Result Grid" is displayed. It has a "Filter Rows:" field and an "Export:" button. The results are shown in a table with two columns: "Profesor" and "Departamento".

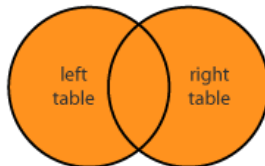
Profesor	Departamento
NULL	Ingenieria
Hinojosa Cardenas, Edward	Computacion
Sarmiento Calisaya, Edgar	Computacion
Gutierrez M, Juan Carlos	Computacion
NULL	Biomedicas

SQL - SELECT - INNER, FULL, LEFT, RIGHT JOIN

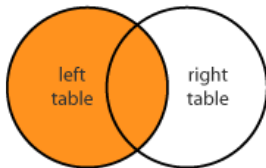
INNER JOIN



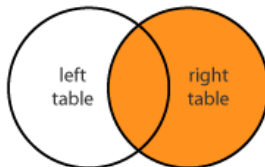
FULL JOIN



LEFT JOIN



RIGHT JOIN



SQL - SELECT - Funciones de Agregación

- Las funciones de agregadas son funciones que toman una colección (un conjunto o conjunto múltiple) de valores como entrada y devuelven un solo valor.
- SQL ofrece cinco funciones agregadas integradas estándar:
 - Total: SUM
 - Average: AVG
 - Minimum: MIN
 - Maximum: MAX
 - Count: COUNT
- La entrada para SUM y AVG debe ser una colección de números.
- Las otras funciones trabajan con colecciones de tipos de datos no numéricos, como cadenas de texto.

SQL - SELECT - SUM

- Obtener la suma (SUM) de los presupuestos de los departamentos.

The screenshot shows a SQL query editor interface. At the top, there is a toolbar with various icons and a 'Limit to 1000 rows' button. The query text is as follows:

```

1 • USE universidad;
2
3 • SELECT SUM(presupuesto) AS "Suma Presupuestos"
4 FROM departamento;
5
6

```

Below the query editor, the 'Result Grid' is displayed. It shows a single row with the column header 'Suma Presupuestos' and the value '27981.74'.

Suma Presupuestos
27981.74

SQL - SELECT - AVG

- Obtener el promedio (average) de los presupuestos de los departamentos.

The screenshot shows a SQL query editor with the following code:

```

1 • USE universidad;
2
3 • SELECT AVG (presupuesto) "Promedio Presupuesto"
4   FROM departamento;
5
6

```

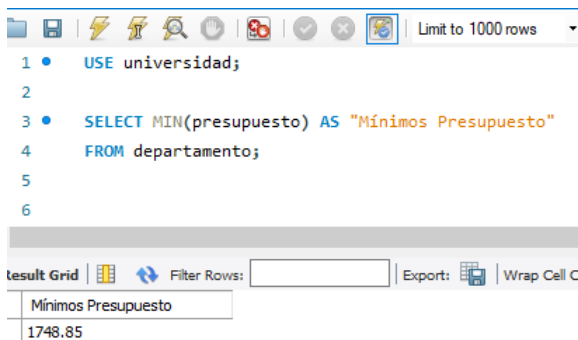
Below the code, a "Result Grid" is displayed with the following data:

Promedio Presupuesto
9327.246667

The interface includes a toolbar at the top with icons for file operations, execution, and a "Limit to 1000 rows" dropdown. The bottom of the window shows a status bar with navigation icons and the page number 75 / 84.

SQL - SELECT - MIN

- Obtener el mínimo (MIN) de los presupuestos de los departamentos.



The screenshot shows a SQL query editor with the following code:

```

1 • USE universidad;
2
3 • SELECT MIN(presupuesto) AS "Mínimos Presupuesto"
4   FROM departamento;
5
6

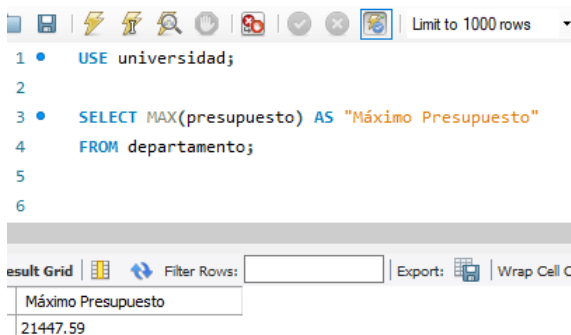
```

Below the query editor, the result grid is displayed with the following data:

Mínimos Presupuesto
1748.85

SQL - SELECT - MAX

- Obtener el máximo (MAX) de los presupuestos de los departamentos.



The screenshot shows a SQL IDE interface. At the top, there is a toolbar with various icons and a dropdown menu set to "Limit to 1000 rows". Below the toolbar, the SQL query is displayed in a text editor:

```
1 • USE universidad;  
2  
3 • SELECT MAX(presupuesto) AS "Máximo Presupuesto"  
4 FROM departamento;  
5  
6
```

Below the query editor, there is a "result Grid" section. It includes a "Filter Rows:" input field, an "Export:" button, and a "Wrap Cell C" option. The result grid displays the following data:

Máximo Presupuesto
21447.59

SQL - SELECT - COUNT

- La función COUNT permite contar el número de filas en una tabla determinada.
- Obtener la cantidad de Departamentos.

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' button. The SQL editor contains the following code:

```

1 • USE universidad;
2
3 • SELECT COUNT(depa_id) AS "N Departamentos"
4   FROM departamento;
5
6

```

Below the editor is the 'Result Grid' section. It has a 'Filter Rows' input field and an 'Export' button. The result is displayed in a table with one column 'N Departamentos' and one row with the value '3'.

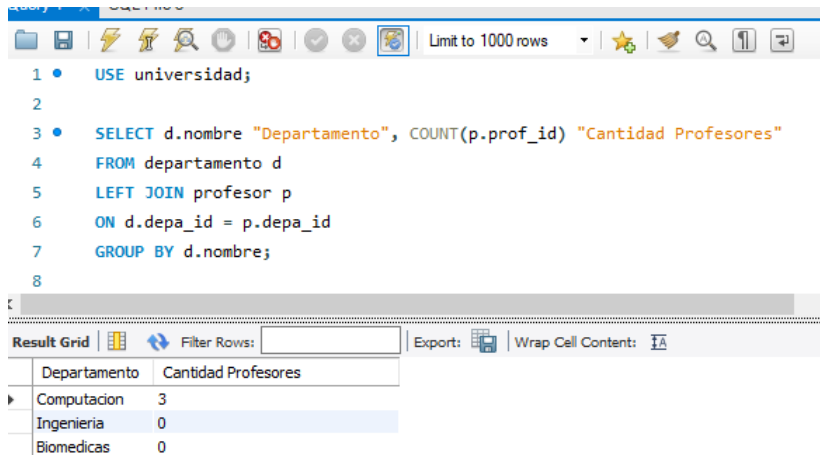
N Departamentos
3

SQL - SELECT - GROUP BY

- La cláusula GROUP BY es un comando SQL que se usa para agrupar filas que tienen los mismos valores
- La cláusula GROUP BY se utiliza en la instrucción SELECT.
- Opcionalmente se usa junto con funciones agregadas para producir informes resumidos de la base de datos.

SQL - SELECT - GROUP BY

- Muestre la cantidad de profesores por departamento.



The screenshot shows a SQL IDE interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, the SQL query is displayed in a text editor:

```

1 • USE universidad;
2
3 • SELECT d.nombre "Departamento", COUNT(p.prof_id) "Cantidad Profesores"
4 FROM departamento d
5 LEFT JOIN profesor p
6 ON d.depa_id = p.depa_id
7 GROUP BY d.nombre;
8

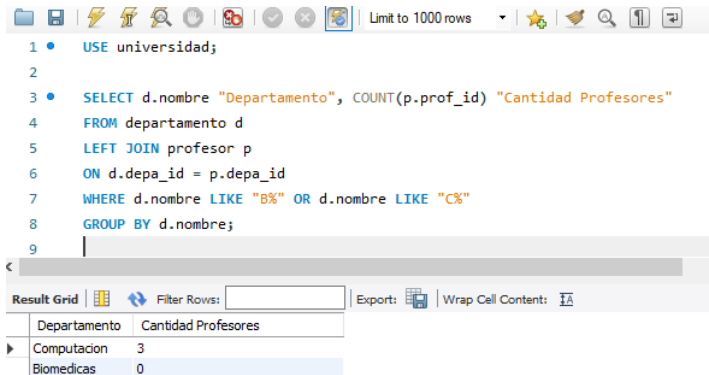
```

Below the query editor, there's a 'Result Grid' section. It includes a 'Filter Rows:' input field, an 'Export:' button, and a 'Wrap Cell Content:' checkbox. The result grid itself is a table with two columns: 'Departamento' and 'Cantidad Profesores'.

Departamento	Cantidad Profesores
Computacion	3
Ingenieria	0
Biomedicas	0

SQL - SELECT - GROUP BY

- La cláusula GROUP BY se puede utilizar con el comando WHERE.
- Muestre la cantidad de profesores por departamento que comienzan con B o C.



The screenshot shows a SQL IDE interface. At the top, there is a toolbar with various icons and a dropdown menu set to "Limit to 1000 rows". Below the toolbar, the SQL query is displayed in a text editor:

```

1 • USE universidad;
2
3 • SELECT d.nombre "Departamento", COUNT(p.prof_id) "Cantidad Profesores"
4 FROM departamento d
5 LEFT JOIN profesor p
6 ON d.depa_id = p.depa_id
7 WHERE d.nombre LIKE "B%" OR d.nombre LIKE "C%"
8 GROUP BY d.nombre;
9

```

Below the query editor, the "Result Grid" is visible, showing the results of the query. It has two columns: "Departamento" and "Cantidad Profesores". The results are as follows:

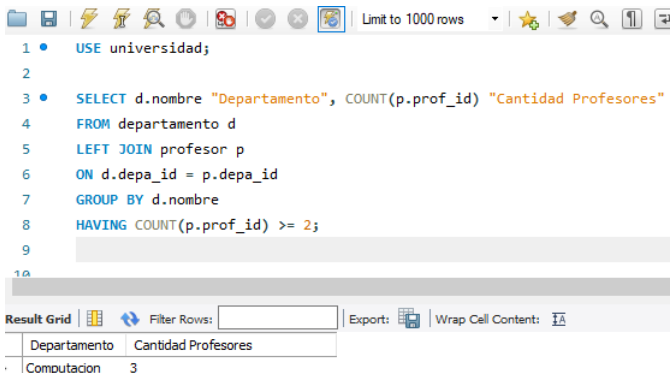
Departamento	Cantidad Profesores
Computacion	3
Biomedicas	0

SQL - SELECT - HAVING

- La función HAVING se utiliza para incluir condiciones con alguna función de agregación de SQL.
- Como la cláusula WHERE no se puede utilizar con funciones de agregación SQL, entonces se puede utilizar HAVING.

SQL - SELECT - HAVING

- Muestre los departamentos que tengan una cantidad mayor o igual a 2 profesores.



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following query:

```

1 • USE universidad;
2
3 • SELECT d.nombre "Departamento", COUNT(p.prof_id) "Cantidad Profesores"
4   FROM departamento d
5  LEFT JOIN profesor p
6    ON d.depa_id = p.depa_id
7  GROUP BY d.nombre
8  HAVING COUNT(p.prof_id) >= 2;
9
10

```

Below the editor, the 'Result Grid' is displayed with the following data:

Departamento	Cantidad Profesores
Computacion	3

¡GRACIAS!

