Carrera: Técnico Superior en Programación
Materia: Laboratorio de Computación III
Tema: Consultas de selección - Parte I

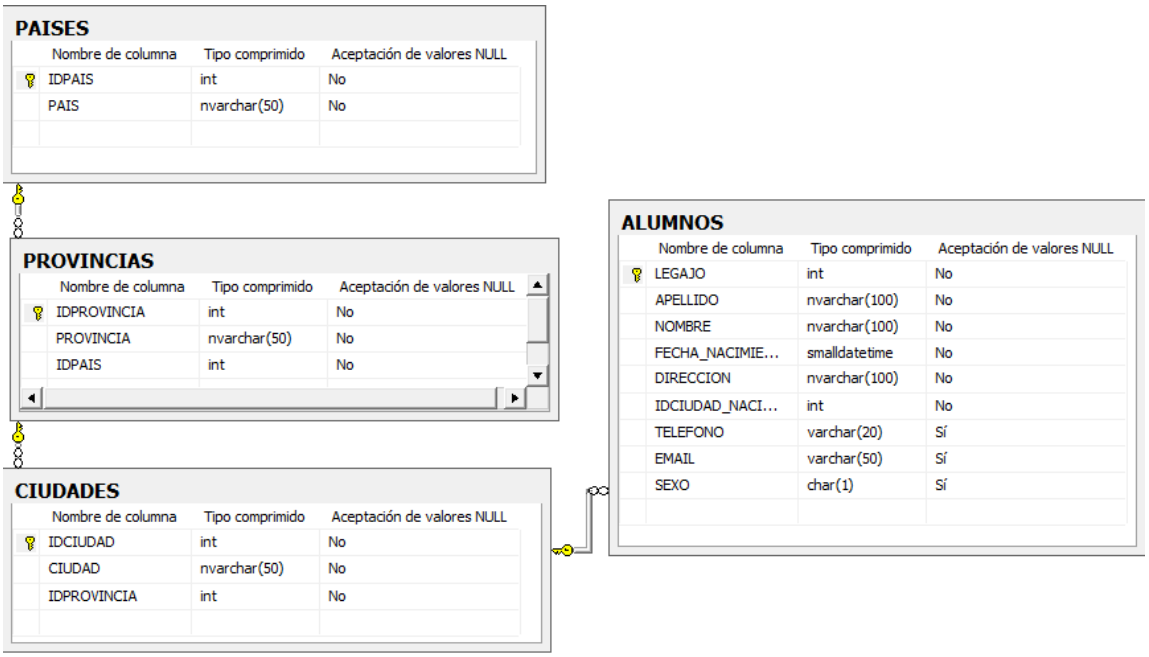
Consultas de Selección - Parte I

Hasta el momento hemos estado utilizando consultas para poder definir y alterar nuestra estructura de datos. Creando mediante código SQL bases de datos, tablas, relaciones y restricciones. Pero, una vez que tengamos la estructura creada correctamente será momento de incorporarle datos a la base de datos y acceder a ellos.
Por ahora, no añadiremos registros a la base de datos mediante código SQL. Lo haremos a través de lo que provee el entorno gráfico de SQL Server. Sin embargo, a continuación analizaremos las diferentes formas de seleccionar datos desde la base de datos mediante código.

La cláusula SELECT

Mediante la sentencia SELECT, vamos a poder obtener datos de nuestra base de datos. La misma permite no sólo aplicar filtros a las columnas sino también a las filas. La cláusula SELECT tiene una serie de elementos y argumentos que se pueden incorporar para facilitar nuestro procesamiento de información al momento de realizar las consultas.

Antes de comenzar con las diferentes alternativas de cómo utilizar las consultas de SELECT, vamos a utilizar una base de datos de ejemplo para poder ubicarlas en el contexto de un caso práctico.



La sentencia para obtener todos los datos de todas las ciudades sería:

```
SELECT IDCIUDAD, CIUDAD, IDPROVINCIA FROM CIUDADES
```

	IDCIUDAD	CIUDAD	IDPROVINCIA
1	1	SAN ISIDRO	1
2	2	VICENTE LOPEZ	1
3	3	CÓRDOBA	2

En el código anterior, se especifican explícitamente las columnas de las cuales se obtendrán los datos. En este caso, la tabla de ciudades está compuesta por las columnas IDCIUDAD, CIUDAD e IDPROVINCIA de manera que se están obteniendo todos los registros de todas las columnas de la tabla.

Por lo tanto, podemos decir que la sintaxis base sería:

```
SELECT [ALL, DISTINCT, TOP] <COLUMNAS SEPARADAS POR COMA>
FROM <NOMBRE DE TABLA>
```

Una alternativa a obtener todos los registros de todas las columnas de la tabla ciudades sería la siguiente:

```
SELECT * FROM CIUDADES
```

	IDCIUDAD	CIUDAD	IDPROVINCIA
1	1	SAN ISIDRO	1
2	2	VICENTE LOPEZ	1
3	3	CÓRDOBA	2

Nótese como en este ejemplo, no se han especificado las columnas sino que se utilizó el símbolo de asterisco. Esta sintaxis es equivalente a que deberá obtener los datos de todas las columnas.

Un ejemplo para la tabla alumnos sería:

```
SELECT * FROM ALUMNOS
```

	LEGAJO	APELLIDO	NOMBRE	FECHA_NACIMIENTO	DIRECCION	IDCIUDAD_NACIMIENTO	TELEFONO	EMAIL	SEXO
1	1000	PEREZ	JUAN	1990-05-05 00:00:00	ITALIA 342	1	NULL	NULL	M
2	1100	MENDEZ	JULIETA	1984-05-10 00:00:00	LIBERTAD 123	2	NULL	NULL	F
3	1300	FERNANDEZ	NATALIA	1984-09-11 00:00:00	JUNIN 12	1	NULL	FERNATALIA@FAKEMAIL.COM	F
4	2000	GUTIERREZ	CARLOS	1991-03-02 00:00:00	BELGRANO 890	3	1234567	NULL	M

Ahora supongamos que queremos obtener todos los registros de la tabla alumnos pero sólo de las columnas legajo, apellido, nombre y fecha de nacimiento. La sintaxis para obtener dichos datos podría ser:

```
SELECT LEGAJO, APELLIDO, NOMBRE, FECHA_NACIMIENTO FROM ALUMNOS
```

	LEGAJO	APELLIDO	NOMBRE	FECHA_NACIMIENTO
1	1000	PEREZ	JUAN	1990-05-05 00:00:00
2	1100	MENDEZ	JULIETA	1984-05-10 00:00:00
3	1300	FERNANDEZ	NATALIA	1984-09-11 00:00:00
4	2000	GUTIERREZ	CARLOS	1991-03-02 00:00:00

SQL también permite incorporar columnas que no son propias de la tabla al momento de realizar la consulta. Supongamos el caso de que querramos obtener el legajo, apellido, nombre y edad de los alumnos. Las primeras tres columnas provienen directamente de la base de datos pero la edad proviene de la base de datos mediante una operación matemática sobre la fecha de nacimiento.

```
SELECT LEGAJO, APELLIDO, NOMBRE, DATEDIFF(YEAR, 0, GETDATE()-FECHA_NACIMIENTO)
AS EDAD FROM ALUMNOS
```

	LEGAJO	APELLIDO	NOMBRE	EDAD
1	1000	PEREZ	JUAN	21
2	1100	MENDEZ	JULIETA	27
3	1300	FERNANDEZ	NATALIA	27
4	2000	GUTIERREZ	CARLOS	21

Aquí podemos observar que en el caso del legajo, apellido y nombre, la consulta no muestra diferencias a la consulta anterior. Sin embargo, sí se puede notar como la cuarta columna que fue nombrada con el alias 'EDAD' mediante el uso de `AS <nombre>`, es el resultado del cálculo de la edad utilizando la función `DATEDIFF` que se encarga de realizar restas entre fechas -la cual se explica en el apunte de SQL-. Esta columna, no es un campo original de la tabla alumnos.

Veamos otro ejemplo:

```
SELECT APELLIDO, 1 AS UNO, 'Hola mundo' AS HOLA, MONTH(FECHA_NACIMIENTO) AS
MES, YEAR(FECHA_NACIMIENTO) AS ANIO FROM ALUMNOS
```

	APELLIDO	UNO	HOLA	MES	ANIO
1	PEREZ	1	Hola mundo	5	1990
2	MENDEZ	1	Hola mundo	5	1984
3	FERNANDEZ	1	Hola mundo	9	1984
4	GUTIERREZ	1	Hola mundo	3	1991

En este ejemplo, que probablemente carezca de sentido. Vemos como obtuvimos los apellidos de la tabla alumnos en la primer columna. La segunda trae la constante 1 y es denominada bajo el alias de UNO. Mismo caso con la tercer columna que sólo trae la cadena 'Hola mundo' con el nombre de HOLA. La cuarta columna denominada MES es el resultante de la función de SQL llamada `MONTH` que a partir de un parámetro del tipo fecha obtiene el mes. Mismo caso para la última columna denominada ANIO sólo que se utiliza la función `YEAR` que devuelve el año del parámetro. En éstas dos últimas columnas los parámetros de MONTH y YEAR son la fecha de nacimiento de cada uno de los registros. De modo que obtendríamos el mes y año de nacimiento de cada alumno.

Por último, se puede observar como se puede obtener registros sin la necesidad de que provengan de una tabla. Son consultas más raras pero igualmente válidas.

```
SELECT GETDATE() AS 'AHORA'
```

	AHORA
1	2012-03-28 01:03:04.167

SELECT ALL, SELECT DISTINCT y SELECT TOP

SELECT ALL

Anteriormente, obteníamos todos los registros de la tabla de la cual seleccionábamos datos. Sin importar si existieran registros duplicados o no. SQL, traerá los datos de ésta manera por defecto. Sin embargo, otra forma de expresarlo sería:

```
SELECT ALL YEAR(FECHA_NACIMIENTO) AS ANIO_NAC FROM ALUMNOS
```

	ANIO_NAC
1	1990
2	1984
3	1984
4	1991

En este caso obtendremos todos los años de nacimiento de la tabla alumnos. Como se puede observar los registros 2 y 3 son duplicados. Sin embargo, como veníamos utilizando las consultas en las páginas anteriores o mediante la utilización de ALL, no solucionaremos el problema de los registros duplicados.

SELECT DISTINCT

Si queremos evitar los registros duplicados al momento de realizar un SELECT, entonces deberemos incluir la expresión `DISTINCT`.

```
SELECT DISTINCT YEAR(FECHA_NACIMIENTO) AS ANIO_NAC FROM ALUMNOS
```

	ANIO_NAC
1	1984
2	1990
3	1991

SELECT TOP

Utilizaremos la expresión TOP para indicar que queremos que sólo un grupo principal de registros. De modo que podríamos obtener menos registros de los que originalmente la consulta pudiera traer.

El primer conjunto de filas que deseamos traer se puede especificar tanto de manera exacta (las 10 primeras filas) como porcentual (el 10% de las filas). Se puede incorporar la sentencia WITH TIES si se desea que la última fila repita los registros que se encuentran igualados.

Ejemplos:

```
SELECT TOP (1) YEAR(FECHA_NACIMIENTO) AS 'FECHA_NAC' FROM ALUMNOS
```

	FECHA_NAC
1	1990

```
SELECT TOP (3) YEAR(FECHA_NACIMIENTO) AS 'FECHA_NAC' FROM ALUMNOS
```

	FECHA_NAC
1	1990
2	1984
3	1984

```
SELECT TOP (50) PERCENT YEAR(FECHA_NACIMIENTO) AS 'FECHA_NAC' FROM ALUMNOS
```

	FECHA_NAC
1	1990
2	1984

```
SELECT TOP (1) WITH TIES YEAR(FECHA_NACIMIENTO) AS 'FECHA NAC' FROM
ALUMNOS ORDER BY YEAR(FECHA_NACIMIENTO) ASC
```

	FECHA NAC
1	1984
2	1984

A continuación, analizaremos cada uno de los ejemplos:
En la primer consulta, se obtiene el primer registro de la consulta que selecciona el año de nacimiento de la tabla alumnos. Como podemos notar en las primeras consultas de selección de la tabla alumnos, el primero de los registros (el del alumno con legajo 1000) tenía fecha de nacimiento con año 1990. Es por eso que obtenemos este valor.
En la segunda, podemos observar al igual que en la anterior como se obtienen un grupo de registros de la consulta original. En este caso se obtienen los primeros tres.
En la tercer consulta, al igual que en las dos primeras se selecciona un conjunto específicos de registros del listado original, con la diferencia de que en lugar de especificarse con un valor entero específico se lo realiza mediante una expresión porcentual. Siendo cuatro registros se obtiene el 50% de los registros (en este caso dos tuplas).
Por último, lo que se realizan dos cosas diferentes a las consultas anteriores. En primer lugar se puede notar como la consulta de selección hace uso de la cláusula ORDER BY, de manera que los registros tendrán un ordenamiento particular. En este caso, se ordena el listado por la columna YEAR(FECHA_NACIMIENTO) de manera ascendiente. De manera que aparecerán primero los registros con menor valor numérico en el año hasta el de mayor valor numérico.
Los años que figuran en la tabla alumnos son 1990, 1984, 1984 y 1991.
La consulta solicita sólo un registro -> TOP (1) pero se especifica la expresión WITH TIES, de modo que como el menor valor entre los años es 1984 y este se encuentra repetido o ‘empatado’ se obtiene dos registros en lugar de uno.

La cláusula ORDER BY

En ocasiones, queremos que nuestra consulta de selección organice los datos de una forma distinta. Una de las formas de organización que se tiene es el ordenamiento. Gracias a ésta cláusula de las consultas de selección podremos hacer que nuestra información se ordene por una o más columnas con criterio ascendente o descendente.
Sin una cláusula de ORDER BY, no se puede garantizar el ordenamiento de una consulta de selección.

```
SELECT [ALL, DISTINCT, TOP] <COLUMNAS SEPARADAS POR COMA>
FROM <NOMBRE DE TABLA> ORDER BY NOMBRE_COLUMNA [ASC, DESC] (, NOMBRE_COLUMNA2
[ASC, DESC], . . .)
```

Ejemplos:

```
SELECT LEGAJO, APELLIDO, NOMBRE, IDCIUDAD_NACIMIENTO FROM ALUMNOS
ORDER BY IDCIUDAD_NACIMIENTO ASC
```

	LEGAJO	APELLIDO	NOMBRE	IDCIUDAD_NACIMIENTO
1	1000	PEREZ	JUAN	1
2	1300	FERNANDEZ	NATALIA	1
3	1100	MENDEZ	JULIETA	2
4	2000	GUTIERREZ	CARLOS	3

Como podemos ver en la consulta anterior, los registros vienen ordenados por IDCIUDAD_NACIMIENTO de manera ascendente.

```
SELECT LEGAJO, APELLIDO, NOMBRE, IDCIUDAD_NACIMIENTO FROM ALUMNOS
ORDER BY IDCIUDAD_NACIMIENTO DESC, APELLIDO ASC
```

	LEGAJO	APELLIDO	NOMBRE	IDCIUDAD_NACIMIENTO
1	2000	GUTIERREZ	CARLOS	3
2	1100	MENDEZ	JULIETA	2
3	1300	FERNANDEZ	NATALIA	1
4	1000	PEREZ	JUAN	1

En el ejemplo anterior, el ordenamiento principal se vuelva a hacer a partir del campo IDCIUDAD_NACIMIENTO sólo que ahora con criterio descendente, luego como ordenamiento secundario (dentro del primer ordenamiento se aplica el segundo -se puede notar en los registros 3 y 4) se clasifican por apellido con criterio ascendente.

La cláusula WHERE

Hasta ahora al conjunto original de filas y columnas le hemos aplicado una serie de filtros al momento de realizar las consultas de selección. En principio hemos visto como se pueden obtener menos columnas de las que originalmente figuran en la tabla o bien aplicarles algún tipo de cálculo u operación para generar una nueva columna.
Con respecto a las filas, también hemos podido filtrar una serie de registros pero bajo ningún

criterio. Sólo los hemos reducido en su cantidad obteniendo un conjunto de datos que, como hemos experimentado, puede especificarse por un valor decimal explícito o por una expresión porcentual.
A continuación, veremos que las filas pueden ser filtradas bajo una o un grupo de condiciones. De manera que sólo pertenecerán al listado de selección aquellas tuplas que cumplan todas las condiciones que se incorporan a la cláusula WHERE. En ella podremos utilizar los operadores lógicos NOT, AND y OR para combinar las condiciones y cualquier tipo de función, expresión o comparación.

Operador BETWEEN
Este operador es utilizado para realizar comparaciones entre rangos. Es decir, cuando queremos saber si un número, cadena o fecha se encuentra dentro de un rango específico.

Ejemplos:
Obtener sólo los registros cuyo legajo se encuentre entre el 1000 y 1200

```
SELECT * FROM ALUMNOS WHERE LEGAJO >= 1000 AND LEGAJO <= 1200
```

	LEGAJO	APELLIDO	NOMBRE	FECHA_NACIMIENTO	DIRECCION	IDCIUDAD_NACIMIENTO	TELEFONO	EMAIL	SEXO
1	1000	PEREZ	JUAN	1990-05-05 00:00:00	ITALIA 342	1	NULL	NULL	M
2	1100	MENDEZ	JULIETA	1984-05-10 00:00:00	LIBERTAD 123	2	NULL	NULL	F

```
SELECT * FROM ALUMNOS WHERE LEGAJO BETWEEN 1000 AND 1100
```

	LEGAJO	APELLIDO	NOMBRE	FECHA_NACIMIENTO	DIRECCION	IDCIUDAD_NACIMIENTO	TELEFONO	EMAIL	SEXO
1	1000	PEREZ	JUAN	1990-05-05 00:00:00	ITALIA 342	1	NULL	NULL	M
2	1100	MENDEZ	JULIETA	1984-05-10 00:00:00	LIBERTAD 123	2	NULL	NULL	F

Como podemos observar en las consultas anteriores, en ambas se obtienen los registros de los alumnos cuyos legajos se encuentren entre 1000 y 1100. En la segunda consulta se hace uso del operador BETWEEN cuyo propósito es especificamente comparar rangos ya sean numéricos, texto o fecha.

```
SELECT LEGAJO, APELLIDO + ', ' + NOMBRE AS APENOM, FECHA_NACIMIENTO FROM
ALUMNOS WHERE FECHA_NACIMIENTO BETWEEN '1/1/1980' AND '1/1/1990'
```

	LEGAJO	APENOM	FECHA_NACIMIENTO
1	1100	MENDEZ, JULIETA	1984-05-10 00:00:00
2	1300	FERNANDEZ, NATALIA	1984-09-11 00:00:00

En esta última consulta se puede apreciar como el operador funciona a la perfección incluso utilizándolo sobre campos que contengan fechas. Por otro lado, se puede observar como se pueden concatenar campos de cadenas de caracteres mediante el uso del operador +

Operador IN
Utilizando el operador IN podremos determinar si un valor especificado se encuentra dentro de una lista de valores.

```
SELECT LEGAJO, APELLIDO, NOMBRE FROM ALUMNOS WHERE LEGAJO IN (1000, 1100, 1101)
```

	LEGAJO	APELLIDO	NOMBRE
1	1000	PEREZ	JUAN
2	1100	MENDEZ	JULIETA

En la consulta anterior, se obtienen los datos de legajo, apellido y nombre de todos los alumnos cuyos legajos coincidan con 1000, 1100 ó 1101.

```
SELECT LEGAJO, APELLIDO, NOMBRE FROM ALUMNOS WHERE LEGAJO NOT IN (1000, 1100, 1101)
```

	LEGAJO	APELLIDO	NOMBRE
1	1300	FERNANDEZ	NATALIA
2	2000	GUTIERREZ	CARLOS

En este caso, se obtienen los datos de los alumnos no coincidan con 1000, 1100 y 1101.

Operador LIKE

Mediante el operador LIKE podremos determinar si una cadena de caracteres específica coincide o no con un patrón determinado. El patrón que se especificará en el LIKE para hacer la comparación puede tener caracteres comunes o comodines.

Comodín	Descripción	Ejemplo
%	Cualquier cadena de cero o más caracteres	WHERE nombre LIKE '%arol%' - busca todos los registros que contengan 'arol' en el nombre. Ej: Carol, Carolina, Carola.
_	Cualquier caracter	WHERE nombre LIKE 'Fernand_' - busca todos los registros que contengan 'Fernand' más un caracter cualquiera. Ej: Fernando, Fernanda, Fernand9.
[ ]	Cualquier carácter individual dentro de un intervalo o conjunto que se haya especificado.	WHERE nombre LIKE '[a-m]ario' - busca todos los registros que contengan un nombre que comience con un caracter entre la 'a' y la 'm' y que luego continúe con la cadena 'ario'. Ej: Dario, Mario. WHERE nombre LIKE '[kcmh]arina' - busca todos los registros que contengan un nombre que comience con 'k','c','m' ó 'h' y que luego continúe con la cadena 'arina'. Ej: Marina, Karina, Carina, etc.
[^]	Cualquier caracter individual que no se encuentre dentro de un intervalo o conjunto que se haya especificado.	WHERE nombre LIKE 'an[^g]%' - busca todos los registros que contengan un nombre que comience con la cadena 'an' que la tercer letra no sea una 'g' y que luego continúe con cualquier cantidad y tipo de caracter. Ej: 'Analía', 'Antonio',

		'Ana' pero NO 'Angel', 'Angie'.

Algunos ejemplos:

```
SELECT LEGAJO, APELLIDO, NOMBRE FROM ALUMNOS WHERE LEGAJO LIKE '1%'
```

	LEGAJO	APELLIDO	NOMBRE
1	1000	PEREZ	JUAN
2	1100	MENDEZ	JULIETA
3	1300	FERNANDEZ	NATALIA

Obtiene todos los datos de legajo, apellido y nombre de los alumnos cuyo legajo comience con 1.

```
SELECT LEGAJO, APELLIDO, NOMBRE FROM ALUMNOS WHERE APELLIDO LIKE '%[A-Z][^R]ez'
```

	LEGAJO	APELLIDO	NOMBRE
1	1100	MENDEZ	JULIETA
2	1300	FERNANDEZ	NATALIA

El ejemplo anterior obtiene los datos de los alumnos cuyo apellido comience con cualquier caracter de la A a la Z y finalice con 'EZ' siempre en cuando la antepenúltima letra no sea la R.

Determinando si un campo contiene NULL

En ocasiones, cuando definimos que una columna de nuestra tabla puede contener un valor núlo es interesante saber cuáles de esos registros contienen dicho valor en tal campo.
Para conocer si un campo contiene null utilizaremos IS NULL.

Ejemplo:

```
SELECT LEGAJO, APELLIDO, NOMBRE, EMAIL FROM ALUMNOS WHERE EMAIL IS NULL
```

	LEGAJO	APELLIDO	NOMBRE	EMAIL
1	1000	PEREZ	JUAN	NULL
2	1100	MENDEZ	JULIETA	NULL
3	2000	GUTIERREZ	CARLOS	NULL

Por supuesto, se puede preguntar si un campo no contiene NULL anteponiendo el operador NOT.

Ejemplo:

```
SELECT LEGAJO, APELLIDO, NOMBRE, EMAIL FROM ALUMNOS WHERE EMAIL IS NOT NULL
```

	LEGAJO	APELLIDO	NOMBRE	EMAIL
1	1300	FERNANDEZ	NATALIA	FERNATALIA@FAKEMAIL.COM

Más recursos sobre este tema

Videotutorial → Consultas de Selección Parte 1A
Explica teóricamente la introducción a las consultas de selección utilizando las cláusulas ORDER BY y el predicado ALL, DISTINCT y TOP.
Duración: 24:48 -- Link online -- Link descarga

Videotutorial → Consultas de Selección Parte 1B
Demuestra de manera práctica las consultas de selección utilizando las cláusulas ORDER BY y el predicado ALL, DISTINCT y TOP.
Duración: XX:XX -- Link Online -- Link descarga

TPS → Consultas de Selección - Parte 1
Ejercicios para poner en práctica las consultas de selección utilizando cláusulas ORDER BY y WHERE. Y el predicado ALL, DISTINCT y TOP.
TP 2: Link online -- Link resolución

Diapositivas → Consultas de Selección - Parte 1
Resume de manera muy sencilla los conceptos principales de las consultas de selección.
Link

Cheatsheet → Consultas de Selección - Parte 1
Contiene términos, sintaxis y gráficos que ejemplifican de manera sencilla y resumida algunos de los términos aprendidos en este tema.
Link

Apunte → Consultas de Selección - Parte 1
Este documento online en su última versión.
Link

Autoevaluación → Consultas de Selección - Parte 1
Test de multiple-choice que evalúa tus conocimientos sobre los contenidos aprendidos en este tema.

Lo podés encontrar en el campus virtual bajo el nombre "Autoevaluación: Consultas de selección - Parte 1"