UNIDAD 1: ESCRITURA DE SENTENCIAS SQL SELECT BASICAS

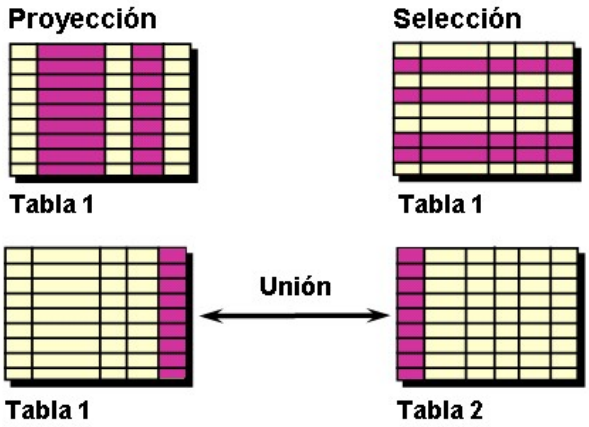
INTRODUCCION

Para extraer datos de la base de datos, es necesario utilizar la sentencia SELECT de lenguaje estructurado de consulta (SQL). En esta unidad se describen las capacidades de las sentencias SQL SELECT y a realizar de forma correcta la escritura de una sentencia SELECT básica.

CAPACIDADES DE LAS SENTENCIAS SQL SELECT

Una sentencia SELECT recupera información de la base de datos. Mediante una sentencia SELECT se puede realizar lo siguiente:

- **Proyección:** se puede utilizar la capacidad de proyección de SQL para seleccionar las columnas de una tabla que desee que su consulta le devuelva. Puede elegir tantas columnas de la tabla como sea necesario.
- **Selección:** se puede utilizar la capacidad de selección de SQL para seleccionar las filas de una tabla que desee que una consulta le devuelva. Puede usar diferentes criterios para restringir las filas visibles.
- **Unión:** se puede utilizar la capacidad de unión de SQL para recopilar datos que están almacenados en diferentes tablas creando un enlace entre ellos.



SENTENCIAS SELECT BASICAS

En su forma más simple, una sentencia SELECT debe incluir lo siguiente:

- Una cláusula SELECT, que especifica las columnas que se han de mostrar
- Una cláusula FROM, que especifica la tabla que contiene las columnas listadas en la cláusula SELECT

En la sintaxis:

```
SELECT    *|{[DISTINCT] column|expression [alias],...}
FROM      table;
```

SELECT	es una lista de una o más columnas
*	selecciona todas las columnas
DISTINCT	suprime los duplicados
column|expresión	selecciona la expresión o columna especificada
alias	asigna cabeceras diferentes a las columnas seleccionadas
FROM *table*	especifica la tabla que contiene las columnas

SELECCION DE TODAS LAS COLUMNAS DE TODAS LAS FILAS

Se pueden visualizar todas las columnas de datos en una tabla si escribe un asterisco (*) detrás de la palabra clave SELECT. En el siguiente ejemplo la tabla DEPARTMENT contiene cuatro columnas: DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID y LOCATION_ID. La tabla contiene siete filas, una para cada departamento:

```
SELECT *
FROM   departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

También se pueden visualizar todas las columnas de la tabla si enumeran todas las columnas después de la palabra clave SELECT. Por ejemplo, la siguiente sentencia SQL, al igual que el ejemplo anterior, muestra todas las columnas y filas de la tabla DEPARTMENTS:

```
SELECT  department_id, department_name, manager_id, location_id
FROM    departments;
```

SELECCION DE COLUMNAS ESPECIFICAS DE TODAS LAS FILAS

Se puede utilizar la sentencia SELECT para visualizar columnas específicas de la tabla si se indican los nombres de las columnas separados por comas. En el ejemplo se muestran todos los números de departamento y la ubicación de los departamentos:

```
SELECT department_id, location_id
FROM   departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

En la cláusula SELECT, se especifican las columnas en el orden en que aparecerán en el resultado. Por ejemplo, para mostrar la ubicación, antes que el número de departamento, de izquierda a derecha, se utiliza la siguiente sentencia:

```
SELECT location_id, department_id
FROM   departments;
```

ESCRITURA DE SENTENCIAS SQL

Utilizando las siguientes reglas sencillas se pueden construir sentencias válidas que sean fáciles de leer como de editar:

- Las sentencias SQL no son sensibles a mayúsculas/minúsculas a menos que se indique.
- Las sentencias SQL se pueden introducir en una o más líneas.
- Las palabras clave no se pueden dividir entre líneas ni abreviar.
- Normalmente las cláusulas están colocadas en líneas separadas por motivos de legibilidad y facilidad de edición.
- Los sangrados se deben utilizar para que los códigos sean más legibles.
- Generalmente, las palabras clave se introducen en mayúsculas; todas las demás palabras, como los nombres de tabla y columnas se introducen en minúsculas.

EXPRESIONES ARITMETICAS

Se puede necesitar modificar la forma en la que se muestra la información, realizar cálculos, o examinar supuestos hipotéticos. Todo ello es posible si se utilizan las expresiones aritméticas. Una expresión aritmética puede contener nombres de columna, valores numéricos constantes y operadores aritméticos.

La tabla siguiente muestra los operadores aritméticos disponibles en SQL. Se puede utilizar operadores aritméticos en cualquier cláusula de una sentencia SQL excepto en la cláusula FROM:

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División

USO DE LOS OPERADORES ARITMETICOS

En el ejemplo siguiente se utiliza el operador suma para calcular un incremento del salario de $300 para todos los empleados y muestra una nueva columna SALARY+300 en el resultado.

Observar que la columna calculada resultante SALARY+300 no es una columna nueva de la tabla EMPLOYEES, sino sólo de visualización. Por defecto, el nombre de una columna nueva surge del cálculo que la generó, en este caso, salary+300.

```
SELECT last_name, salary, salary + 300
FROM   employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300
...		
Hartstein	13000	13300
Fay	6000	6300
Higgins	12000	12300
Gietz	8300	8600

20 rows selected.

PRIORIDAD DE OPERADOR

Si una expresión aritmética contiene más de un operador, la multiplicación y la división se evalúan en primer lugar. Si los operadores incluidos en una expresión son de idéntica prioridad, entonces la evaluación se hace de izquierda a derecha. Se puede utilizar paréntesis para forzar que la expresión incluida entre paréntesis se evalúe en primer lugar.

El siguiente ejemplo muestra el apellido, el salario y la remuneración anual de los empleados. La remuneración anual se calcula multiplicando 12 por el salario mensual, más una bonificación única de $100. Observar que la multiplicación se realiza antes que la suma.

Se utilizan los paréntesis para forzar el orden de prioridad estándar y para mejorar la claridad. Por ejemplo, la expresión del ejemplo se puede escribir como (12*salary)+100 sin que haya ningún cambio en el resultado.

```
SELECT last_name, salary, salary + 300
FROM   employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300
...		
Hartstein	13000	13300
Fay	6000	6300
Higgins	12000	12300
Gietz	8300	8600

20 rows selected.

USO DE PARENTESIS

Se pueden alterar las reglas de prioridad utilizando paréntesis, para especificar el orden de ejecución de los operadores.

En el siguiente ejemplo se muestra el apellido, el salario y la remuneración anual de los empleados. La remuneración anual se calcula como salario mensual más una bonificación mensual de $100 multiplicado por 12. Debido a los paréntesis, la suma tiene prioridad sobre la multiplicación.

```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200
Hunold	9000	109200
Ernst	6000	73200
...		
Hartstein	13000	157200
Fay	6000	73200
Higgins	12000	145200
Gietz	8300	100800

20 rows selected.

DEFINICION DEL VALOR NULO

Si una fila no tiene un valor de dato para una columna determinada, se dice que ese valor es *nulo*, o que contiene un null.

Un valor nulo es un valor no disponible, no asignado, desconocido, o no aplicable. Un valor nulo no es lo mismo que cero ni que un espacio. Cero es un número y un espacio es un carácter.

Las columnas de cualquier tipo de dato pueden contener valores nulos. Sin embargo, algunas restricciones, NOT NULL y PRIMARY KEY, evitan que se utilicen valores nulos en la columna.

En la columna COMMISSION_PCT de la tabla EMPLOYEES, observar que sólo el director de ventas o el representante de ventas puede percibir una comisión. Los demás empleados no tienen derecho a ganar comisiones. Un valor nulo representa este hecho.

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
...			
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2
...			
Gietz	AC_ACCOUNT	8300	

20 rows selected.

VALORES NULOS EN EXPRESIONES ARITMETICAS

Si un valor de columna de una expresión aritmética es nulo, el resultado es null. Por ejemplo, si se realiza una división de un número por cero, aparecerá un mensaje de error. Sin embargo, si se divide un número por un valor nulo, el resultado será nulo o desconocido.

En el siguiente ejemplo, el empleado King no obtiene ninguna comisión. Debido a que la columna COMMISSION_PCT de la expresión aritmética es nula, el resultado es null.

```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```

LAST_NAME	12*SALARY*COMMISSION_PCT
King	
Kochhar	
...	
Zlotkey	25200
Abel	39600
Taylor	20640
...	
Gietz	

20 rows selected.

DEFINICION DE UN ALIAS DE COLUMNA

Cuando se muestra el resultado de una consulta, iSQL*Plus normalmente utiliza el nombre de la columna seleccionada como la cabecera de columna. Es posible que esta cabecera no sea descriptiva y por tanto, sea difícil de comprender. Se puede cambiar la cabecera de una columna mediante un alias de columna.

Se especifica el alias tras la columna en la lista SELECT utilizando un espacio como separador. Por defecto, las cabeceras de alias aparecen en mayúsculas. Si el alias contiene espacios o caracteres especiales (como # o $), o es sensible a mayúsculas/minúsculas, escriba el alias entre comillas dobles (" ").

USO DE ALIAS DE COLUMNA

Este ejemplo muestra los nombres y los porcentajes de las comisiones de todos los empleados. Observar que la palabra clave opcional AS se ha usado delante del nombre alias de columna. El resultado de la consulta es el mismo se utilice la palabra clave AS o no. Observar también que la sentencia SQL tiene los nombres, las comisiones y los alias de columna en minúsculas, mientras que el resultado de la consulta muestra la cabecera de columna en mayúsculas. Como se mencionó anteriormente, las cabeceras de columna aparecen en mayúsculas por defecto.

```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

...

20 rows selected.

El segundo ejemplo muestra los apellidos y los salarios anuales de todos los empleados. Debido a que Annual Salary contiene un espacio, se ha escrito entre comillas dobles. Observar que la cabecera de columna en el resultado es exactamente igual que el alias de columna.

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM   employees;
```

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

...

20 rows selected.

OPERADOR DE CONCATENACION

Se pueden enlazar columnas a otras columnas, expresiones aritméticas o valores constantes para crear una expresión de caracteres mediante el operador de concatenación (||). Las columnas a cada lado del operador se combinan para hacer una única columna de resultados.

En el siguiente ejemplo, LAST_NAME y JOB_ID están concatenados y se les ha asignado el alias Employees. Observar que el código del puesto y el apellido del empleado están combinados para formar una única columna de resultados. La palabra clave AS delante del alias hace que la cláusula SELECT sea más fácil de leer.

```
SELECT   last_name||job_id AS "Employees"
FROM     employees;
```

Employees
KingAD_PRES
KochharAD_VP
De HaanAD_VP
HunoldIT_PROG
ErnstIT_PROG
LorentzIT_PROG
MourgosST_MAN
RajsST_CLERK

...

20 rows selected.

CADENAS DE CARACTERES LITERALES

Un literal es un carácter, un número o una fecha que está incluido en la lista SELECT y que no es un nombre de columna o un alias de columna. Un literal está impreso para cada fila devuelta. Las cadenas literales de texto de formato libre se pueden incluir en el resultado de la consulta y son tratados de la misma manera que una columna en la lista SELECT. los literales de caracteres y fecha *se deben* escribir entre comillas simples (' '), los literales numéricos no.
El siguiente ejemplo muestra los códigos de los cargos y los apellidos de todos los empleados. La columna tiene la cabecera Employee Details. Observar los espacios entre las comillas simples en la sentencia SELECT. Los espacios mejoran la legibilidad del resultado.

```
SELECT last_name  ||' is a '||job_id
       AS "Employee Details"
FROM   employees;
```

Employee Details
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP
Hunold is a IT_PROG
Ernst is a IT_PROG
Lorentz is a IT_PROG
Mourgos is a ST_MAN
Rajs is a ST_CLERK

...

20 rows selected.

FILAS DUPLICADAS

A menos que se indique lo contrario, iSQL*Plus muestra los resultados de una consulta sin eliminar filas duplicadas. El siguiente ejemplo muestra todos los números de departamento de la tabla EMPLOYEES. Observar que los números de departamento están repetidos

```
SELECT department_id
FROM   employees;
```

DEPARTMENT_ID
90
90
90
60
60
60
50
50
50

...

20 rows selected.

Para eliminar filas duplicadas en el resultado, se debe incluir la palabra clave DISTINCT en la cláusula SELECT inmediatamente detrás de la palabra clave SELECT. En el ejemplo siguiente, la tabla EMPLOYEES contiene en realidad 20 filas, pero sólo hay siete únicos números de departamento en la tabla.

Se pueden especificar múltiples columnas detrás del cualificador DISTINCT. Este cualificador afecta a todas las columnas seleccionadas y el resultado es cada distinta combinación de columnas.

```
SELECT DISTINCT department_id
FROM   employees;
```

DEPARTMENT_ID
10
20
50
60
80
90
110

8 rows selected.