5.1. Las consultas

Las bases de datos no tendrían ninguna utilidad si no se pudiera acceder a ellos, hacer operaciones y analizarlos para tomar decisiones. Las **consultas** son las operaciones que permiten mostrar los registros de las tablas, así como hacer operaciones sobre estos registros. Las consultas son las operaciones que más se efectúan en la vida de la base de datos. Los desarrolladores de software que almacenan los datos en bases de datos, y que acceden a ellos a través de sistemas gestores de bases de datos, tienen que embeber en los programas que desarrollan infinidad de consultas asociadas a los diferentes elementos que se pueden encontrar en sus formularios, botones, listas desplegables, cuadros de textos, cuadrículas, etcétera.

Figura 5.1. *Los procesos para la implementación de las* Queries, selects *o consultas se pueden llegar a convertir en plenas tareas profesionales. Estos profesionales abordan cómo desarrollar consultas especializadas altamente eficientes.*

5.1.1. El comando SELECT

El comando que se usa para la consulta de datos en la mayoría de las bases de datos relacionales es SELECT. La sintaxis del comando es compleja, pero se podría comenzar reduciéndola como sigue:

```
SELECT [{DISTINCT|UNIQUE}|ALL] {*|columna[,columnas]}
FROM tabla[,tablas]
[WHERE condicion_where]
[ORDER BY columna {DESC|ASC} [,columnas [{DESC|ASC}]];
```

En cualquier consulta, las cláusulas **SELECT** y **FROM** son obligatorias. La primera se usa para indicar las columnas que se necesitan proyectar y la segunda para indicar cuáles son las tablas que deben participar para proyectar dichas columnas y hacer operaciones con dichos campos. Cualquier tabla se incluirá, inevitablemente, si posee alguna columna que se requiere proyectar a través del SELECT. Las tablas incluidas en la cláusula FROM deben ser las suficientes y necesarias, ni una más ni una menos, ya que cada una implica aumentar el número de registros en la selección.

La cláusula **WHERE** también es muy común, en pocas consultas no aparecerá. Esta cláusula permite seleccionar, de todos los registros que se combinan con las tablas declaradas en FROM, aquellos que interesan para la operación que se persigue. Tendrá siempre una operación lógica que filtra los registros que van a retornar.

La cláusula **ORDER BY** permite ordenar la proyección a través de una o más columnas y de forma ascendente o descendente.

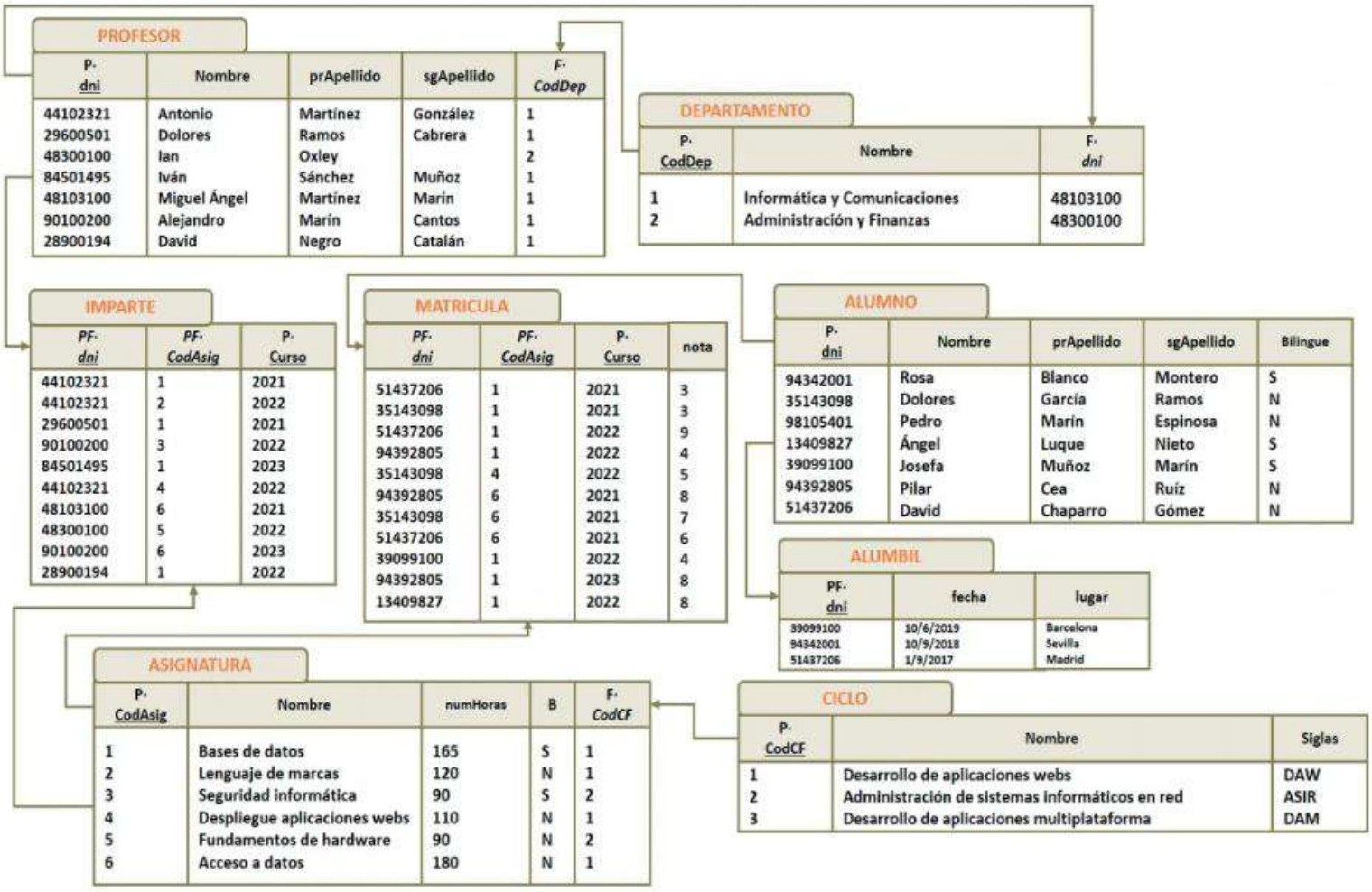

Figura 5.2. *Esquema relacional del caso de uso «Centro de formación bilingüe versión 4.0».*

La consulta siguiente muestra las columnas *Nombre* y *numHoras* de la tabla *Asignatura:*

```
SELECT Nombre, NumHoras FROM Asignatura;
```

Para mostrar todas las columnas de una tabla se puede usar el carácter *, así, de este modo, no se tiene que indicar cada una de las columnas. Por ejemplo, de los profesores de la base de datos ejecutaríamos la consulta

```
SELECT * FROM Profesor;
```

lo que equivale a

```
SELECT dni, Nombre, PrApellido, SgApellido, CodDep FROM Profesor;
```

Si quisiéramos que los registros se mostrasen ordenados por una columna, se usará la cláusula ORDER BY. Así, al ejecutar la consulta `SELECT * FROM Profesor ORDER BY PrApellido;` se mostrarían todos los profesores ordenados ascendentemente por el primer apellido. Se puede usar más de una columna, por ejemplo

```
SELECT * FROM Profesor ORDER BY PrApellido, SgApellido;
```

mostraría los profesores ordenados por el primer apellido y el segundo apellido. También se puede indicar el número de la columna atendiendo al orden en el que se ha indicado en la proyección en el SELECT. Un ejemplo es

```
SELECT dni, nombre, prApellido, sgApellido, bilingue
FROM Alumno
ORDER BY 4;
```

donde la salida estaría ordenada por la columna cuarta, es decir, por el campo SgApellido. Si se usa el símbolo *, no se puede añadir ninguna expresión en el SELECT.

5.1.2. Cálculos en el predicado SELECT

Después de la palabra SELECT se pueden realizar cálculos aritméticos con los valores de las columnas numéricas de las tablas declaradas en la cláusula FROM. No se pueden hacer operaciones lógicas ni relacionales. Los operadores aritméticos son +, -, *, % y /. El símbolo * es el que se usa para multiplicar dos valores numéricos y % para obtener el resto de una división.

La siguiente consulta muestra las asignaturas con los créditos de las asignaturas. Si se considera que un crédito son 10 horas, la consulta podría ser la siguiente:

```
SELECT Nombre, numHoras/10 FROM Asignatura;
```

Si se usa el símbolo * para mostrar todas las columnas de las tablas declaradas en FROM, no se pueden declarar cálculos. Por ejemplo, la siguiente sentencia sería errónea:

```
SELECT *, numHoras/10 FROM Asignatura;
```

Cuando se hace una operación aritmética sobre el valor NULL el resultado es NULL, es decir, si hubiera algún registro cuyo valor en la columna *numHoras* fuese NULL, la operación NULL/10 sería NULL, no mostrando ningún valor para dicho registro.

Para estas operaciones es útil usar un mecanismo que consiste en pedir que en dicha consulta una columna se muestre con otro nombre. A esto se le llama **alias en columnas** y se coloca *as* detrás de la columna, seguido del texto que quiere que se muestre. Esto se puede usar en cualquier tipo de consulta y en Oracle la palabra ***as*** es opcional. Los siguientes son ejemplos de uso de alias:

```
SELECT Nombre as "Nombre Asignatura", numHoras/10 as "Créditos" FROM Asignatura;
SELECT Nombre "Nombre Asignatura", numHoras/10 "Créditos" FROM Asignatura;
SELECT dni, Nombre, prApellido "Primer apellido", sgApellido "Segundo apellido", CodDep "Código de departamento" FROM Profesor;
```

El predicado **DISTINCT** omite las filas que se repiten tras hacer el filtrado de la consulta. Por ejemplo, la siguiente consulta muestra el DNI de los profesores que imparten la asignatura con código CodAsig=1. Como estas relaciones se pueden repetir para diferentes cursos, el DNI se mostrará varias veces:

```
SELECT dni FROM Imparte WHERE CodAsig=1;
```

La siguiente consulta nos muestra los valores de *dni* repetidos al hacer uso del predicado DISTINCT:

```
SELECT DISTINCT dni FROM Imparte WHERE CodAsig=1;
```

5.1.3. Seleccionar los registros que cumplen una condición

La cláusula WHERE se usa para seleccionar aquellos registros en caso de cumplir una condición. Esta condición estará compuesta por una o más expresiones lógicas o relacionales. Cuando se ejecuta una sentencia SELECT se recorre toda la tabla registro a registro y para cada uno de ellos se verifica si se cumple la condición. Aquellos que la cumplan serán seleccionados. Es por eso que en la condición solo pueden usarse columnas de las tablas que se declaran en la cláusula FROM.

En la ejecución de una consulta el procesador del SGBD revisa registro a registro en todas las tablas declaradas en el FROM en busca de aquellos que cumplen la condición definida en la cláusula WHERE.

Por ejemplo, si se quisiera mostrar las asignaturas de más de 100 horas de duración, se ejecutaría la siguiente sentencia:

```
SELECT CodAsig, Nombre FROM Asignatura WHERE numHoras>100;
```

No es necesario que las columnas que participen en la condición estén también en la proyección en el SELECT. Lo que se tiene que tener en cuenta es que no se puede usar una columna que no esté en las tablas.

Atendiendo a las columnas que se quieren proyectar y los filtros que se quieren realizar, lo más difícil es saber cuál o cuáles son las tablas que hay que consultar. Por ejemplo, si se quisiera mostrar el *dni* de los alumnos que han sacado más de un 7, la pregunta que habría que hacerse es ¿en qué tabla se almacena la calificación obtenida por un alumno? La respuesta es fácil: la columna nota de la tabla matrícula almacena la calificación que un alumno obtuvo, por su *dni,* en una asignatura, por su *codasig.* Como solo es necesario mostrar el *dni* no hay que cruzar más columnas. La consulta quedaría como sigue:

```
SELECT dni FROM Matricula WHERE nota>7;
```

En la Tabla 5.1 se muestran varios ejemplos usando la cláusula WHERE.

Tabla 5.1. Algunos ejemplos usando la cláusula WHERE

Operación	Consulta SQL
Mostrar las asignaturas que pertenecen al ciclo con código 1.	`SELECT nombre FROM Asignatura WHERE codCF=1;`
Mostrar las asignaturas que tienen una duración de horas entre 50 y 150.	`SELECT nombre FROM Asignatura WHERE numHoras>=50 AND numHoras<=150;`