

PRÁCTICA: 9. EJERCICIOS AVANZADOS**OBJETIVOS:**

Completar los conocimientos de SQL adquiridos en las prácticas anteriores mediante el estudio de varios casos avanzados.

MATERIAL:

ORACLE 9i versión para WINDOWS XP

BIBLIOGRAFIA:

ORACLE: Introduction to ORACLE 9i:SQL. Student Guide. Volume 1 & 2. 2001.

ORACLE 9I: GUIA DE APRENDIZAJE
ABBEY, MICHAEL Y COREY, MIKE Y ABRAMSON, IAN
MCGRAW-HILL / INTERAMERICANA DE ESPAÑA, S.A.

CONTENIDO:

Esquema de Trabajo
Buenas Prácticas
Ejemplos Avanzados

SQL es como una caja de herramientas. En las prácticas anteriores se han presentado las diversas herramientas que ofrece y cómo utilizar cada una de ellas. Pero queda pendiente lo más difícil: saber qué herramienta usar en cada caso y cómo combinarlas para conseguir lo que se busca. En esta práctica se estudian varios casos en los cuales se explica el mecanismo intelectual de razonamiento seguido para su resolución.

1. Esquema de Trabajo

Todos los ejercicios están desarrollados para la base de datos ACADEM, que se ha utilizado desde principio de curso.

Tablas:

areas(codigo, nombre, departamento)

profesores(codigo, apellido1, apellido2, nombre_pila, activo, categoria, dedicacion, area)

departamentos(codigo, nombre)

asignaturas(siglas, nombre, creditos, curso, anualidad, clase, horas_teoría, horas_práctica, grupos_teoría, grupos_práctica, alumnos)

locales(codigo, nombre, docente, capacidad, edificio, situación)

grupos(curso, clase, codigo, nombre)

docencia(id, curso, clase, grupo, siglas, profesor, local, día, hora, periodicidad)

Claves ajenas:

areas.departamento -> departamentos

profesores.area -> areas

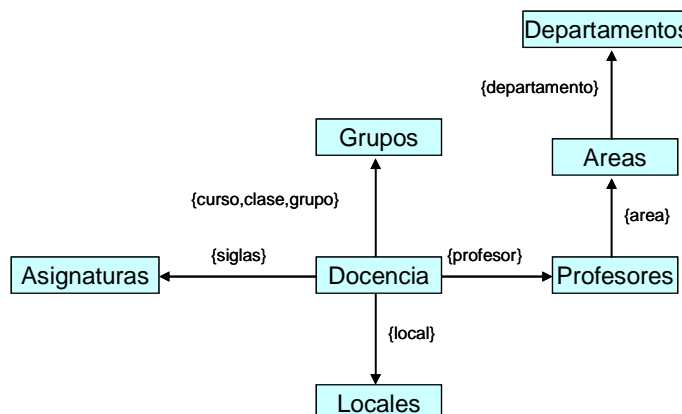
docencia.{curso,clase,grupo} -> grupos

docencia.profesor -> profesores

docencia.local -> locales

docencia.siglas -> asignaturas

La figura siguiente muestra el diagrama referencial, una buena ayuda para “pensar” consultas y otras actividades con bases de datos relacionales.



2. Buenas Prácticas

De manera general, es recomendable seguir los siguientes pasos a la hora de diseñar una consulta, vista o restricción sobre un esquema relacional:

- a) Analizar el enunciado o requisitos solicitados hasta estar seguro de comprenderlo perfectamente.
- b) Identificar las tablas donde están los datos necesarios y la manera en que dichas tablas deberán ser utilizadas (combinación, unión, etc.). En este apso es muy util el diagrama referencial.
- c) Pensar el tipo de orden SQL que sirve para hacer lo que se busca: CREATE TABLE, CREATE VIEW, SELECT, etc.
- d) Pensar las expresiones concretas y cláusulas particulares que deberán utilizarse, por ejemplo, qué debe filtrar la cláusula WHERE de una consulta, o si es necesario realizar un agrupamiento con GROUP BY.
- e) Escribir la orden SQL.

En el caso de acciones que modifican datos (INSERT, UPDATE, DELETE) siempre es recomendable comprobar previamente con una consulta que los datos afectados son los deseados.

Cuando una consulta o vista sea demasiado compleja, es bueno utilizar la modularización, es decir, crear otras vistas más simples que permitan obtener cálculos intermedios.

En el caso de restricciones es aconsejable escribirlas antes en lógica matemática para afianzar su comprensión previa. Si no se domina la lógica matemática se puede hacer en lenguaje natural o pseudocódigo pero con mucho cuidado porque en estos casos es muy fácil la confusión. También es bueno realizar antes una consulta para comprobar a priori que el estado actual de la base de datos cumple la condición evitando un posible error al crearla.

Recordar que se debe estar seguro de la estructura (columnas, tipos y restricciones) de los elementos del esquema a utilizar. En caso de duda, se pueden utilizar las ordenes que incluye ORACLE (por ejemplo, DESCRIBE <tabla>). Igualmente, cuando se quieren conocer los valores que toma una columna C de una tabla T lo más sencillo es hacer una consulta como la siguiente: SELECT C FROM T GROUP BY C;.

3. Ejemplos Avanzados

EJERCICIO 1

Crear una nueva tabla AsiPro(siglas, profesor, horas_semana) que almacene todas las parejas profesor-asignatura, tal que el profesor imparte la asignatura. Profesor es el código de profesor y horas-semana es el número medio de horas semanales de dicha asignatura que

imparte dicho profesor a lo largo del curso, teniendo en cuenta que depende de las horas de clase que da y de la periodicidad de dicha docencia y de la anualidad de la asignatura.

Análisis:

Se trata de crear una tabla bastante sencilla, pero con una dificultad especial en idear la manera de obtener los valores de la columna horas_semana.

Todos los datos a mostrar están en la tabla Docencia. Dicha tabla contiene una fila por cada hora de clase semanal que se imparte, por tanto, para “contar” las horas-semana de cada pareja asignatura-profesor se deberá realizar una consulta con agrupamiento por siglas y profesor.

La columna “Asignatura.anualidad” contiene los siguientes valores: “A” indica que es anual, 1 indica que es de sólo primer semestre, y 2 indica que es de segundo semestre. Por tanto, para obtener el promedio de todo el curso, las horas de clase con A se deben contar enteras, mientras que las que tengan 1 o 2 se deben contar como 0’5.

La columna “Docencia.periodicidad” tiene un 1 o un 2 según la clase se imparte todas las semanas de forma consecutiva o cada dos semanas de forma alterna. De cara al cálculo de horas_semana esto implica que al “contar” el valor de cada hora de clase las que tengan 2 se deben contar sólo como 0’5 horas a la semana.

Procedimiento:

1) Creamos una tabla para contener los datos. Los tipos de cada columna deberán ser los mismos que tienen en la tabla Docencia. La columna horas_semana debe admitir decimales porque una media puede tenerlos. La clave primaria deberá ser la concatenación de las de asignatura y profesor.

2) Antes de almacenar datos en una tabla, es conveniente comprobar que serán los deseados mediante la oportuna consulta. Como no es posible tener una única expresión que contemple los dos casos distintos de anualidad, es necesaria una consulta diferente para la docencia de las asignaturas de cada uno de los casos (anual versus semestral).

Cada una de estas consultas estará basada en la tabla Docencia con agrupamiento por siglas y profesor, combinada con la tabla Asignaturas para conocer el valor de anualidad.

3)

Comprobado que ambas consultas funcionan bien, ya es posible utilizarlas para añadir de forma automática todos sus datos a la tabla creada anteriormente.

EJERCICIO 2

Crear la vista DepProActivos con la relación de profesores activos de cada departamento. Las columnas a incluir son: codigo_dep, codigo_pro, categoria y dedicacion.

Análisis:

Si miramos el diagrama referencial podemos comprobar que la relación entre departamentos y profesores es a través de la tabla Areas, por tanto, los datos que se quieren conseguir se obtienen de las tablas de Profesores (codigo_pro, categoria, dedicacion) y Areas (departamento). No es necesario combinar con la tabla Departamentos porque Areas ya incluye el código de departamento, que es lo que interesa.

Se debe filtrar de manera que sólo se obtengan los profesores en activo (aquellos que tienen un 1 en la columna activo).

Procedimiento

1)

Probamos primero en forma de consulta, combinando datos de las tablas Profesores y Areas, hasta que se consiguen los datos buscados.

2)

La consulta anterior se utiliza como base para crear la vista.

EJERCICIO 3

Hacer una consulta que muestre la media de horas de clase semanales que imparten los profesores en activo de cada departamento. El resultado debe incluir tres columnas: código y nombre del departamento, y media de horas semanales de sus profesores en activo.

Análisis:

La tabla nueva creada en el ejercicio 1 nos facilita las horas semanales de cada profesor en cada asignatura. Sumando todas las asignaturas que da cada profesor (agrupando por profesor) será posible conseguir los horas semanales de cada profesor.

La vista del ejercicio 2 nos permite conocer los profesores en activo de cada departamento. Para obtener el resultado buscado será necesario combinar ambos conjuntos de datos y agrupar por departamento, de forma que los resultados se refieran a cada departamento.

Procedimiento

1)

Creamos una consulta para obtener las horas semanales de cada profesor sumando las de todas sus asignaturas, es decir, agrupando por profesor las filas de la tabla AsiPro.

2)

Utilizamos la consulta anterior para crear una vista llamada ProfesorHoras.

3)

Generamos una consulta combinando la vista ProfesorHoras con la vista DepProActivos, que agrupe por departamento y muestre el departamento y la media de horas de sus profesores. Para mostrar también el nombre del departamento se debe utilizar la tabla Departamentos.

EJERCICIO 4

Listar el código y nombre de las áreas cuyos profesores sólo dan asignaturas en un único curso (1º, 2º, ...).

Análisis:

Se trata de buscar las áreas cuyos profesores no imparten asignaturas de más de un curso, es decir, pueden impartir asignaturas sólo de 1º, o sólo de 2º, pero no unas de 1º y otras de 2º. La docencia (incluido el curso) que imparte cada profesor está en la tabla Docencia.

Procedimiento

1)

Creamos una consulta con todas las combinaciones existentes de area-curso, tal que profesores del área dan alguna docencia de ese curso. Para ello es necesario combinar las tablas Docencia y Profesor (para conocer su área), agrupando por area y curso.

2)

Convertimos dicha consulta en una vista llamada, por ejemplo, AreaCurso.

3)

Utilizando la vista anterior, se obtienen las áreas con un solo curso, es decir, las áreas para las cuales no existe otra fila de esa misma área pero de distinto curso (esto se puede hacer con una subconsulta). Para mostrar el nombre del área además del código, es necesario combinar con la tabla Areas.

EJERCICIO 5

Comprobar que “no existe docencia de una asignatura en un grupo que no sea del mismo curso que el de la asignatura”.

Análisis:

Cada grupo de clase se identifica por tres valores: curso, clase y grupo. Así, 1-T-A es el grupo A de teoría de 1º. Cada hora de clase (tabla Docencia) es de un cierto grupo de clase y,

por tanto, de un cierto curso. Por otro lado, cada asignatura es de un cierto curso.

La condición del enunciado señala que ambos cursos, el de grupo de clase y el de asignatura deben estar en consonancia en todas las horas de clase, es decir, en todas las filas de Docencia.

Para comprobar si se cumple la condición se realiza una consulta que obtenga los posibles resultados que la violan. Si el resultado es vacío entonces se cumple la condición.

Procedimiento

1)

Hacemos una consulta para encontrar los casos de docencia con curso distinto que la asignatura de dicha docencia. Para ello se combinan las tablas Asignaturas y Docencia de forma que se trate de la misma asignatura.

ACLARACIÓN:

Si quisiéramos hacer que la condición anterior se cumpliera siempre de forma automática y ORACLE rechazase cualquier cambio en los datos que la violase, ya no se trataría de una simple comprobación mediante una consulta, sino que debería ser una restricción incorporada al esquema de la base de datos.

Dicha restricción se podría expresar en lógica matemática de la siguiente forma:

$$\forall d(\text{Docencia}(d) \rightarrow \neg \exists a(\text{Asignaturas}(a) \wedge a.\text{curso} \neq d.\text{curso}))$$

En SQL estándar se podría modelar con una cláusula CHECK en la tabla Docencia que utilizaría una subconsulta para comprobar que no existe ninguna asignatura que viola la condición. En ORACLE 9i no es posible hacerlo así porque un CHECK no puede emplear subconsultas.

Otra alternativa sería modelar la restricción de esta otra manera equivalente:

$$\forall d \forall a((\text{Docencia}(d) \wedge \text{Asignaturas}(a)) \rightarrow d.\text{curso} = a.\text{curso})$$

En SQL estándar esta restricción se podría representar mediante un CREATE ASSERTION pero otra vez resulta que en ORACLE 9i no existe dicha posibilidad.

Por tanto, en ORACLE 9i no quedaría más remedio que recurrir a opciones más complejas como algún disparador implementado mediante PL/SQL.