Leyenda de Colores

- Amarillo: Para conceptos fundamentales, definiciones y puntos clave.
- Verde: Para ventajas, objetivos, propósitos y características positivas.
- Azul: Para tipos, clasificaciones, componentes, estructuras y ejemplos.
- Rojo/Salmón: Para problemas, inconvenientes, limitaciones o advertencias.
- Gris: Para tecnologías específicas, nombres propios, estándares o secciones explícitamente marcadas como "Contenido Prioritario".

Guía Técnica Avanzada: Consultas Multitabla y Subconsultas en Entornos Relacionales

1. Fundamentos de la Relacionalidad y la Necesidad de Composición

En el diseño de arquitecturas de datos modernas, la normalización no es un objetivo opcional, sino una exigencia técnica para garantizar la integridad y eliminar la redundancia. Esta práctica distribuye la información en múltiples entidades especializadas vinculadas mediante campos comunes. No obstante, la segmentación lógica impone el reto de la recomposición: para recuperar una visión coherente del negocio, el ingeniero de datos debe orquestar consultas multitabla que reconstruyan las relaciones originales sin comprometer el rendimiento.

Distribución de la Información

La clave reside en el uso de campos comunes. Consideremos el ejemplo de las tablas USUARIOS (con clave principal Login) y PARTIDAS (vinculadas por el campo Cod_Creación). Si el sistema requiere un informe que asocie el nombre del jugador con las horas acumuladas en sus sesiones, es imperativo extraer registros de ambas tablas, ya que la identidad reside en la primera y la métrica temporal en la segunda.

El Riesgo del Producto Cartesiano

Desde una perspectiva de sistemas, combinar tablas sin restricciones de unión genera un Producto Cartesiano. Esta operación produce todas las combinaciones posibles entre las filas de las tablas involucradas.

- Impacto Técnico: Si asociamos dos tablas de 10 registros cada una, el resultado será de 10 x 10 = 100 filas.
- Evaluación de Eficiencia: En entornos de producción, el crecimiento exponencial de las filas puede provocar el agotamiento de la memoria volátil y una saturación de las operaciones de entrada/salida (I/O). Esta ineficiencia convierte al producto cartesiano en una operación prohibida para cualquier arquitecto de sistemas, salvo en casos de diseño de pruebas específicos.

Para mitigar este riesgo, es imperativo aplicar mecanismos de discriminación de registros mediante las composiciones internas.

---

2. Composiciones Internas (Inner Joins) y Reglas de Asociación

La composición interna es el método estándar para emparejar registros que poseen valores idénticos en sus columnas de unión. Este proceso asegura que el conjunto de resultados mantenga una integridad referencial estricta, excluyendo cualquier dato que carezca de correspondencia.

Reglas Críticas para Composiciones Profesionales

Un especialista senior debe observar siempre las siguientes directrices:

1. Combinación Múltiple: El motor relacional permite unir tantas tablas como el modelo de datos exija.
2. Criterios de Columnas: La asociación puede fundamentarse en una o varias parejas de campos.
3. Flexibilidad del SELECT: Es posible proyectar columnas de cualquier tabla implicada, formen o no parte de la condición de unión.
4. Resolución de Ambigüedad: Ante columnas con nombres idénticos, es obligatorio el uso de alias para garantizar la claridad sintáctica.

El Rol de la Entidad Asociativa (Bridge)

En esquemas normalizados, a menudo necesitamos una tabla intermedia para conectar entidades que no tienen una relación directa. En nuestro modelo, la tabla HISTORIAL_LABORAL actúa como el puente (bridge) indispensable: vincula un DNI de la tabla EMPLEADOS con un DPTO_COD de la tabla DEPARTAMENTOS. Sin esta pieza, la relación entre el nombre de un empleado y su departamento es inexistente.

Ejemplo de implementación (Sintaxis Clásica):

```
-- Unión de 3 tablas mediante la tabla puente HISTORIAL_LABORAL
SELECT D.NOMBRE_DPTO, E.NOMBRE, E.APELLIDO1, E.APELLIDO2
FROM DEPARTAMENTOS D, EMPLEADOS E, HISTORIAL_LABORAL H
WHERE E.DNI = H.EMPLEADO_DNI
  AND H.DPTO_COD = D.DPTO_COD;
```

---

3. Composiciones Externas: El Operador de Oracle (+)

Para obtener informes de gestión exhaustivos, es vital capturar registros "huérfanos". Un departamento de nueva creación sin personal asignado es un dato valioso que una consulta interna omitiría, sesgando la realidad operativa.

Sintaxis y Posicionamiento del Operador (+)

En la sintaxis tradicional de Oracle, el operador (+) debe colocarse en la cláusula WHERE, específicamente detrás del campo de la tabla que aceptará valores nulos (aquella que "carece" de la información en el emparejamiento).

Caso Práctico: El Departamento "DATAMINING"

Consideremos el departamento DATAMINING (creado mediante INSERT con jefe NULL).

1. Fallo de la Consulta Interna: `WHERE D.JEFE = E.DNI` ignorará a DATAMINING por no haber correspondencia en EMPLEADOS.
2. Éxito de la Consulta Externa: Al utilizar `D.JEFE = E.DNI(+)`, forzamos al sistema a incluir todos los departamentos, aunque el jefe no exista.

```
-- El operador (+) se coloca en E.DNI para preservar el registro de
DEPARTAMENTOS
SELECT D.NOMBRE_DPTO, D.JEFE, E.NOMBRE, E.APELLIDO1
FROM DEPARTAMENTOS D, EMPLEADOS E
WHERE D.JEFE = E.DNI(+);
```

---

4. Modernización de Consultas: El Estándar SQL99

El estándar SQL99 representa un hito en la ingeniería de consultas al separar las condiciones de asociación (lógica de unión) de las de selección (lógica de filtrado), mejorando drásticamente la mantenibilidad del código.

Comparativa de Funciones SQL99

Cláusula	Función Técnica y Estratégica
CROSS JOIN	Genera un producto cartesiano explícito. Útil para generar matrices de datos.
NATURAL JOIN	Une tablas automáticamente por columnas homónimas. Advertencia: Funciona incluso sin PK/FK definidas siempre que los nombres coincidan.
JOIN USING	Permite especificar manualmente la columna común, evitando uniones accidentales en tablas con múltiples campos iguales.
JOIN ON	El estándar para asociaciones complejas o cuando los nombres de columna difieren entre tablas.
LEFT OUTER JOIN	Composición externa que preserva todos los registros de la tabla izquierda.
FULL OUTER JOIN	Operación exhaustiva que devuelve todas las filas de ambas tablas, asociadas o no.

Transición a Código Profesional (SQL99):

```
-- Historial laboral con JOIN ON
SELECT E.NOMBRE, E.APELLIDO1, H.FECHA_INICIO, H.FECHA_FIN
FROM EMPLEADOS E
JOIN HISTORIAL_LABORAL H ON (H.EMPLEADO_DNI = E.DNI);
```

```
-- Departamentos y Jefes con LEFT OUTER JOIN (Sustituye al operador +)
SELECT D.NOMBRE_DPTO, D.JEFE, E.NOMBRE
FROM DEPARTAMENTOS D
LEFT OUTER JOIN EMPLEADOS E ON (D.JEFE = E.DNI);
```

---

5. Operaciones de Conjuntos: Unión, Intersección y Diferencia

El álgebra de conjuntos permite consolidar o contrastar verticalmente información de fuentes diversas, tratando los resultados del SELECT como conjuntos matemáticos.

Operadores Principales

- UNION: Consolida filas de ambos conjuntos eliminando duplicados.
- INTERSECT: Identifica registros comunes presentes en todas las fuentes consultadas.
- MINUS: Extrae la diferencia (registros presentes en el conjunto A pero no en el B).

Requisitos de Arquitectura

Para que estas operaciones sean válidas, es imperativo que las consultas compartan:

1. El mismo número de columnas.
2. Tipos de datos compatibles.
3. El mismo orden de proyección.

Ejemplos de Aplicación:

- INTERSECT: Localizar alumnos inscritos simultáneamente en INGLES, FRANCES y PORTUGUES.
- MINUS: Generar una lista de alumnos de INGLES que no han iniciado el curso de PORTUGUES para campañas de fidelización.

---

6. Subconsultas y Lógica de Comparación Avanzada

Las subconsultas actúan como filtros dinámicos, permitiendo que el resultado de una operación interna sirva de parámetro para la consulta principal, facilitando comparaciones contra datos no conocidos a priori.

Estrategia de Comparación

1. Valor Único: Operadores =, >, <. Error garantizado si la subconsulta devuelve múltiples filas.
2. Múltiples Filas: Requieren instrucciones de conjunto:
    - IN: Pertenencia al conjunto.
    - ANY: La condición es verdadera si se cumple para al menos un elemento.
    - ALL: Validación contra la totalidad del conjunto.

Análisis de Eficiencia con ALL

Para encontrar al empleado con el salario mínimo sin recurrir a funciones de agregado, utilizamos:

```sql
SELECT NOMBRE, SALARIO
FROM EMPLEADOS
WHERE SALARIO <= ALL (SELECT SALARIO FROM EMPLEADOS);
```

El uso de ALL es una técnica robusta de comparación de conjuntos que garantiza que el valor seleccionado sea menor o igual que cualquier otro valor existente en la tabla.

---

7. Apéndice: Resolución de Casos Prácticos

A continuación, se detalla la lógica necesaria para resolver el "Ejercicio 7" utilizando el esquema técnico extraído del sistema (DESC EMPLEADOS y DESC DEPARTAMENTOS).

Caso 1: Proyección de Historial Salarial

Enunciado: Mostrar nombres de empleados, sus salarios históricos y fechas de inicio.

- Lógica de Negocio: Requiere una composición interna entre EMPLEADOS e HISTORIAL_LABORAL.
- Campos Clave: E.NOMBRE, E.APELLIDO1, H.SALARIO_HIST (campo implícito en el historial), H.FECHA_INICIO.
- Punto de Unión: EMPLEADOS.DNI = HISTORIAL_LABORAL.EMPLEADO_DNI.

Caso 2: Auditoría por DNI '12345'

Enunciado: Histórico laboral del DNI '12345' con nombre de puesto y rango salarial.

- Lógica de Negocio:
  1. Filtro restrictivo inicial: `WHERE E.DNI = '12345'`.
  2. Asociación con HISTORIAL_LABORAL para obtener los periodos.
  3. Nota de Arquitectura: Dado que la tabla EMPLEADOS solo contiene datos personales, la obtención del "nombre del puesto" y "rango salarial" requiere un JOIN adicional con una tabla de PUESTOS o CATEGORIAS, asumiendo que el código de puesto reside en el historial del empleado.
- Columnas Resultantes: E.NOMBRE, E.APELLIDO1, P.NOMBRE_PUESTO, P.MIN_SALARIO, P.MAX_SALARIO.

La aplicación rigurosa de estas técnicas asegura una recuperación de información precisa, escalable y alineada con los estándares de excelencia en ingeniería de datos.