

A faint, abstract background consisting of a network graph with numerous small, light-gray dots connected by thin gray lines, creating a sense of data connections and complexity.

Dominando el Universo de Datos Relacionales

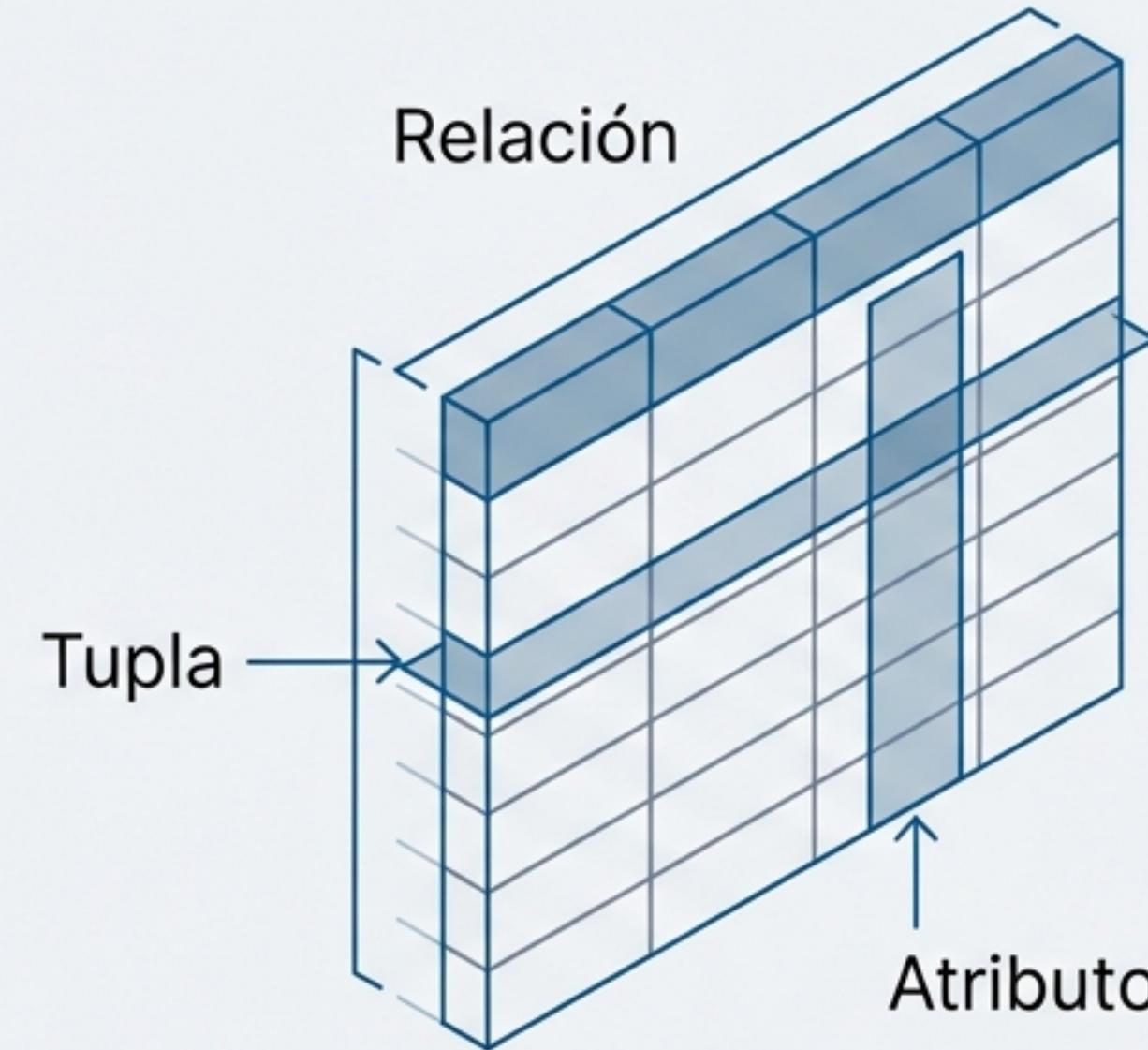
Una Guía Arquitectónica y Práctica de SQL



Blandskron

La Arquitectura que Define los Datos: El Modelo Relacional

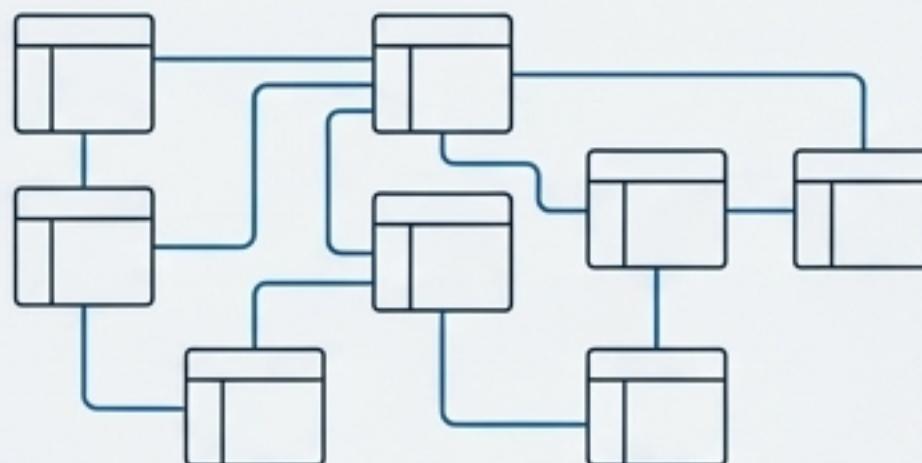
- El Modelo Relacional ha sido el paradigma dominante para datos estructurados desde la década de 1970, proporcionando una base segura, basada en reglas y consistente.
- Su persistencia frente a tecnologías NoSQL radica en la adhesión a las propiedades **ACID** (Atomicidad, Consistencia, Aislamiento, Durabilidad), que garantizan la validez de los datos ante cualquier fallo.
- **Estructura fundamental:** La información se organiza en tablas (relaciones), compuestas por filas (tuplas) y columnas (atributos). Este diseño minimiza la redundancia y maximiza la integridad a través de la normalización.



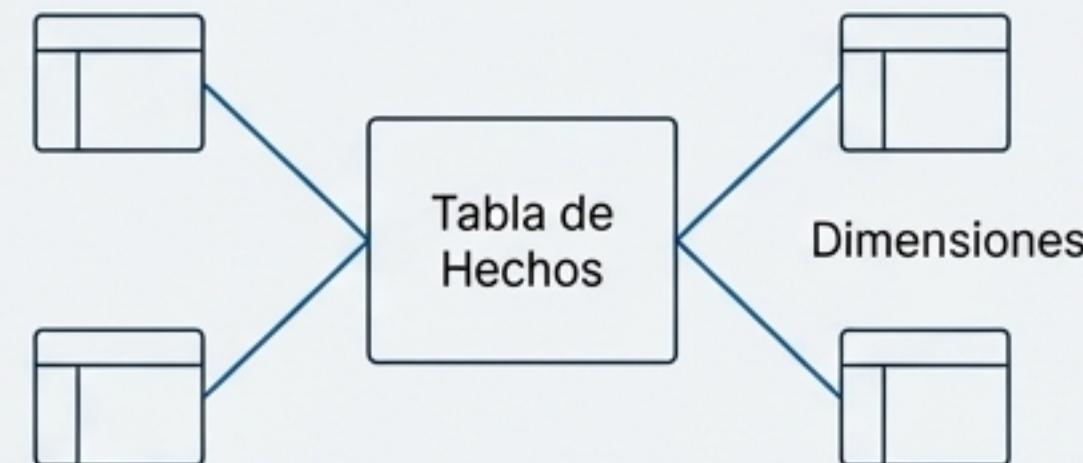
Dos Caras del Mismo Universo: OLTP vs. OLAP

Para usar SQL eficazmente, es imperativo distinguir entre los dos paradigmas de procesamiento de datos: el transaccional y el analítico.

Esquema Normalizado (3NF)



Esquema de Estrella



Característica	OLTP (Procesamiento de Transacciones en Línea)	OLAP (Procesamiento Analítico en Línea)
Propósito	Gestión operativa, transacciones en tiempo real (crear pedidos, actualizar saldos).	Análisis de datos complejos, soporte a decisiones estratégicas.
Modelo de Datos	Altamente normalizado (3NF) para evitar duplicidad y optimizar escrituras.	Desnormalizado (Esquemas de Estrella/Copo de Nieve) para optimizar lecturas y reducir uniones.
Operaciones	Lecturas/Escrituras atómicas y rápidas (` INSERT `, ` UPDATE `).	Lecturas masivas y agregaciones complejas (` SELECT ` masivos).
Volumen	Modesto (Gigabytes), datos actuales.	Masivo (Terabytes/Petabytes), datos históricos.

El Mapa del Territorio: Interpretando el Modelo de Datos

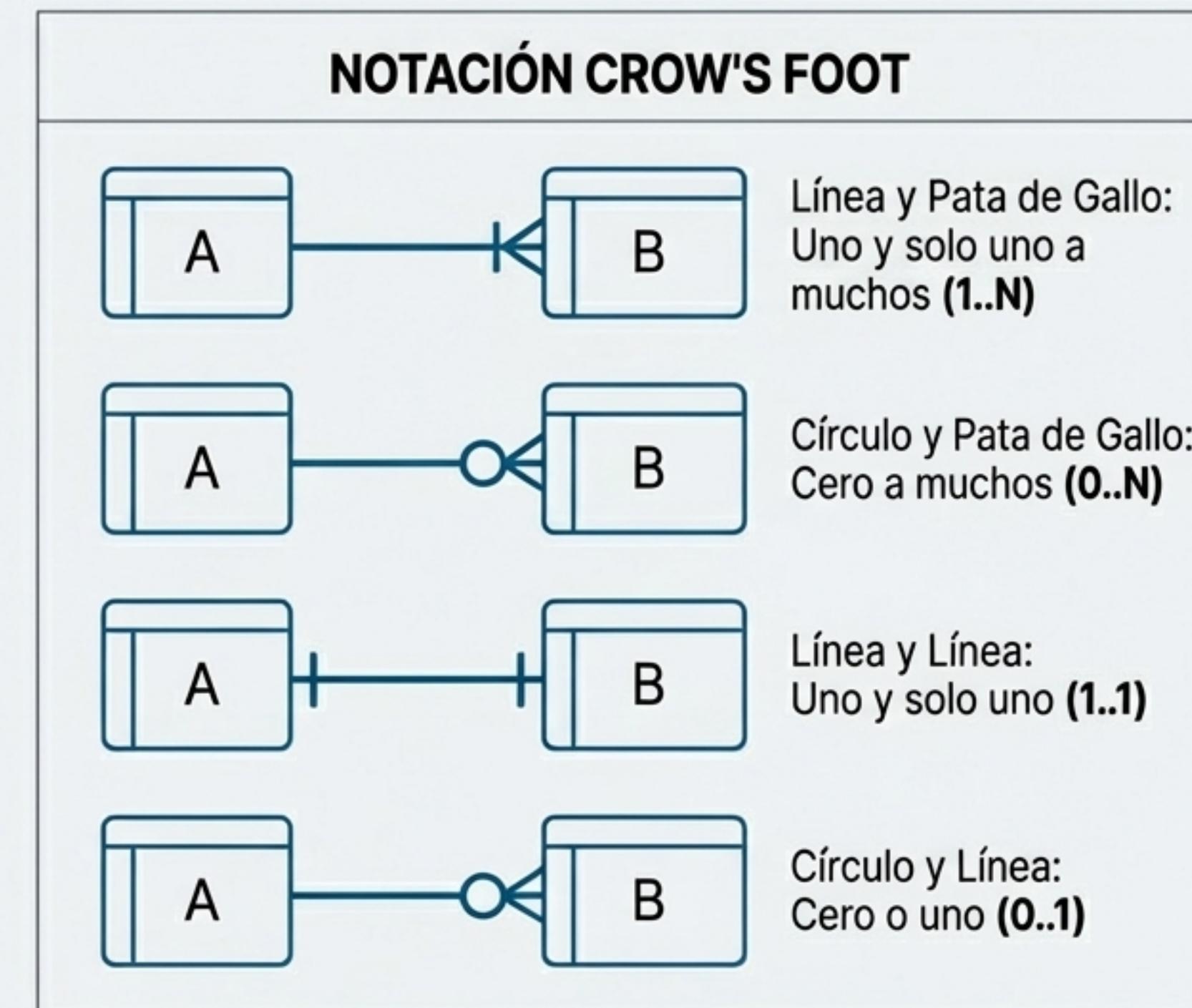
Entendiendo los Diagramas Entidad-Relación (ERD)

Un ERD es una herramienta visual que ilustra la lógica de la base de datos. Sus componentes clave son:

- **Entidades:** Objetos sobre los que se almacenan datos (ej. Clientes, Pedidos).
- **Atributos:** Propiedades de una entidad (ej. Dirección, Precio).
- **Relaciones:** Vínculos lógicos entre entidades (ej. Un cliente *realiza* un pedido).

Notación "Crow's Foot" (Pata de Gallo)

Es el estándar para representar la cardinalidad (cuántos) y la opcionalidad (si es obligatorio).



Los Pilares de la Integridad: Llaves Primarias y Foráneas

Llave Primaria (Primary Key - PK):

- * Una columna (o conjunto) que identifica de forma **única** cada fila.
- * No puede contener valores nulos (NULL).
- * Garantiza la integridad de la entidad (no hay registros duplicados).

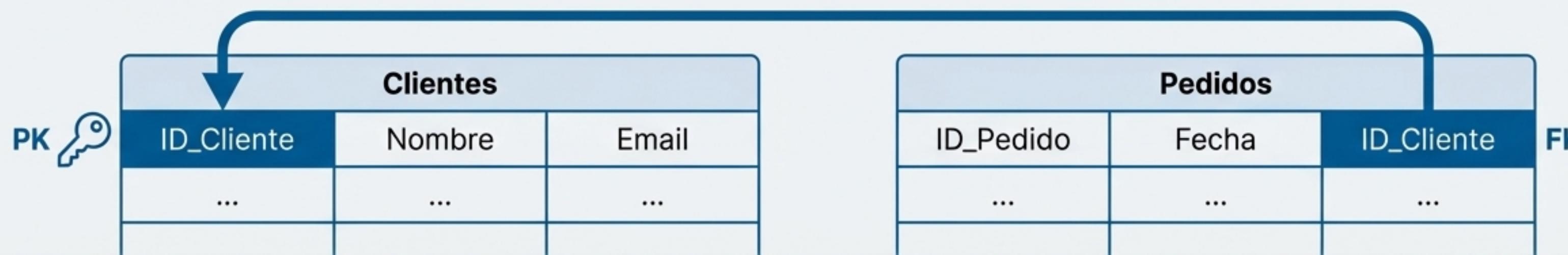
Llave Foránea (Foreign Key - FK):

- * Una columna que hace referencia a la Llave Primaria de otra tabla.
- * Establece la relación física y aplica la integridad referencial.

Integridad Referencial

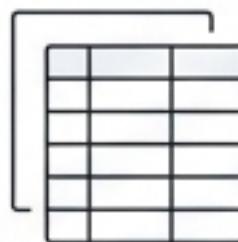
'Un sistema de reglas que asegura que las relaciones entre tablas sean válidas y consistentes.'

Previene "**registros huérfanos**": no puede existir un pedido que refiera a un cliente que no existe. El motor de la base de datos lo impide. Gracias a esto, podemos confiar en que las correspondencias en un `JOIN` son correctas.

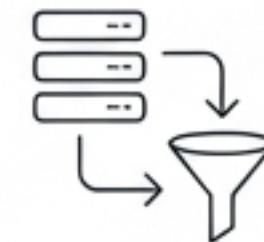


La Herramienta de Exploración Fundamental: `SELECT ... FROM ... WHERE`

SQL es un lenguaje declarativo: especificas *qué* datos deseas, y el motor determina *cómo* recuperarlos eficientemente. El resultado de una consulta siempre es otra tabla.



-- Retorna todas las columnas y filas de la tabla Empleados
`SELECT * FROM Empleados;`



-- Obtiene nombre y salario de los empleados del departamento
-- de 'Ventas' con un salario superior a 3000.
`SELECT nombre, departamento, salario
FROM Empleados
WHERE departamento = 'Ventas' AND salario > 3000;`

Breve mención de **AND** (todas las condiciones deben ser ciertas), **OR** (al menos una debe ser cierta) y **NOT** (invierte la condición).

Técnicas de Filtrado Avanzado

Para resolver problemas de selección más complejos, SQL proporciona operadores especializados para patrones, rangos y conjuntos.

1. `LIKE` (Búsqueda de Patrones)



- Se utiliza para buscar patrones en texto con comodines.
- %: representa cero o más caracteres ('A%' busca textos que empiezan con A).
- _: representa un solo carácter.

Nota Técnica: El uso de comodines al inicio ('%texto') puede impedir el uso de índices y ralentizar la consulta.

2. `BETWEEN` (Rangos)



- Selecciona valores dentro de un rango inclusivo (incluye los límites).
- Aplicable a números, fechas y texto.

3. `IN` (Conjuntos)



- Filtra basándose en una lista de posibles valores.
- Es una alternativa más legible y a menudo más eficiente que múltiples condiciones OR.
- Ejemplo:
`WHERE ID_Categoría IN (1, 5, 9)`

Resumiendo el Paisaje de Datos con `GROUP BY`

A menudo, las preguntas de negocio requieren agrupar datos por categoría y obtener resúmenes. Para esto, SQL combina la cláusula `GROUP BY` con funciones de agregación.

Funciones de Agregación Comunes:

- `COUNT()`: Cuenta el número de filas.
- `SUM()`: Suma valores numéricos.
- `AVG()`: Calcula el promedio.
- `MIN() / `MAX()`: Encuentran el valor mínimo y máximo.

```
-- ¿Cuántos empleados hay en cada departamento?  
SELECT departamento, COUNT(*) AS num_empleados  
FROM Empleados  
GROUP BY departamento;
```



Explicación: `GROUP BY departamento` agrupa las filas. `COUNT(*)` cuenta las filas dentro de cada grupo. El resultado es un resumen por departamento.

El Momento Preciso del Filtro: `WHERE` vs. `HAVING`

El error común es intentar filtrar agregados con `WHERE`. La cláusula `WHERE` actúa *antes* de la agrupación, mientras que `HAVING` actúa *después*.



-- Departamentos con más de 5 empleados, excluyendo
-- a los empleados con salarios menores a 2000.

```
SELECT departamento, COUNT(*)  
FROM Empleados  
WHERE salario >= 2000    -- Filtra filas ANTES de agrupar  
GROUP BY departamento  
HAVING COUNT(*) > 5;    -- Filtra grupos DESPUÉS de agrupar
```

Cruzando Fronteras: Consultas a Múltiples Tablas con `JOIN`

La verdadera potencia de SQL reside en combinar datos de múltiples tablas. La cláusula `JOIN` permite unir filas de dos tablas basándose en una columna común, generalmente una relación de Llave Primaria a Foránea.

```
SELECT  
    E.nombre AS Empleado,  
    D.nombre AS Departamento  
FROM Empleados E  
JOIN Departamentos D ON E.departamento_id = D.id;
```

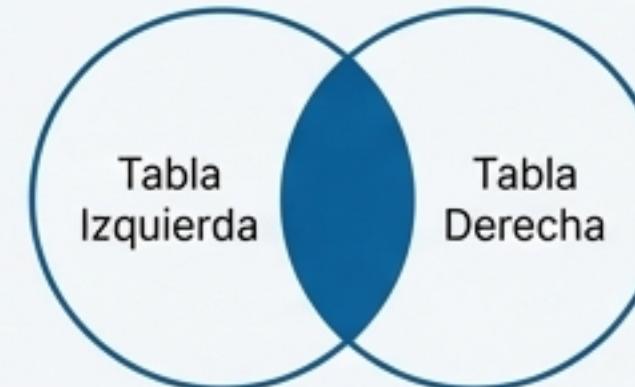


- El motor de la base de datos compara los valores de `departamento_id` en `Empleados` con `id` en `Departamentos`.
- Empareja y combina las filas donde los valores coinciden.
- El `JOIN` por defecto es un `INNER JOIN`: solo muestra filas con coincidencias en *ambas* tablas.

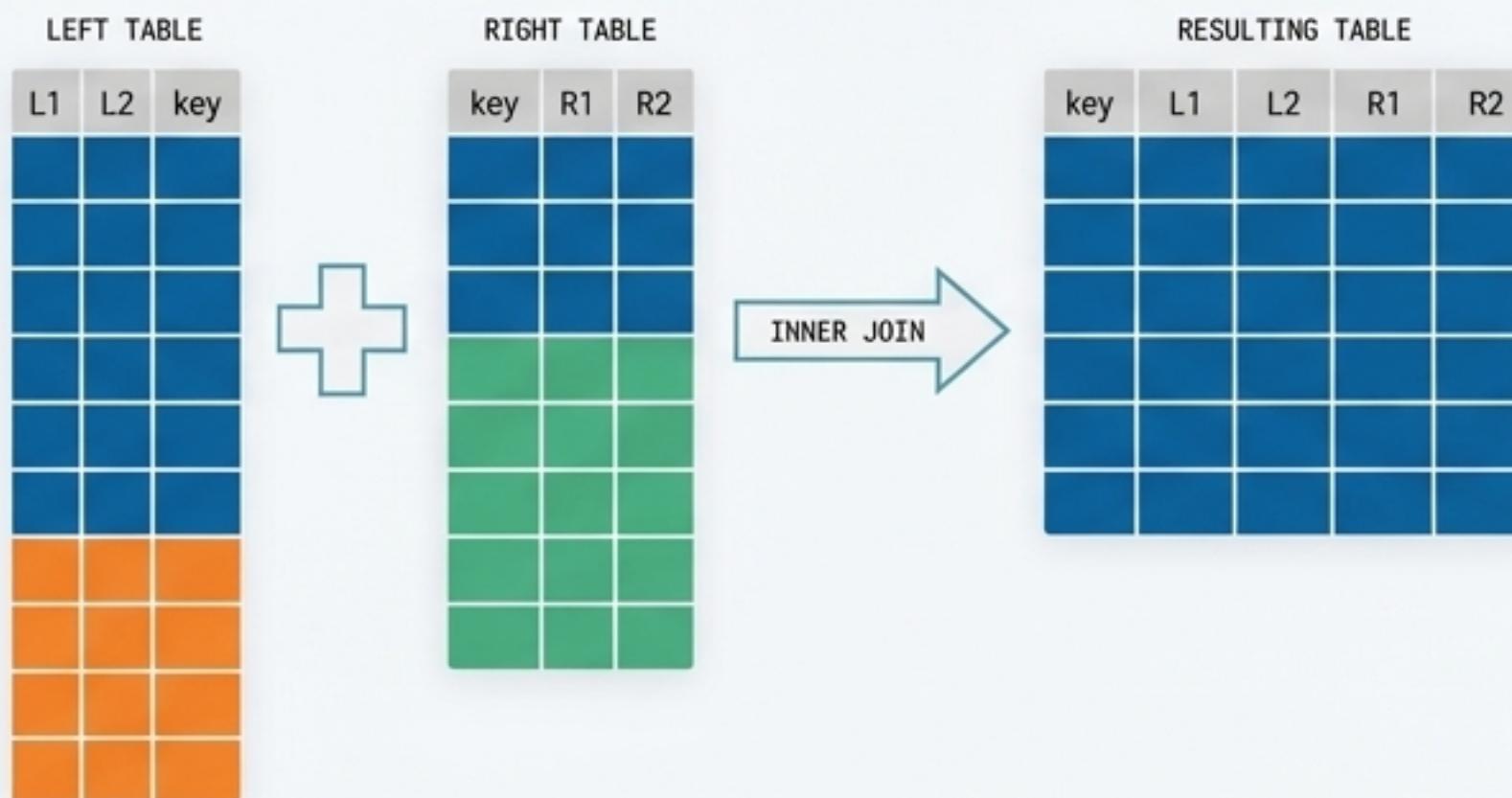
La Familia de `JOINS`: Eligiendo la Unión Correcta (Parte 1)

`INNER JOIN` (Intersección)

Devuelve **solo** las filas donde hay coincidencia en **ambas** tablas. Se excluyen los registros sin pareja.

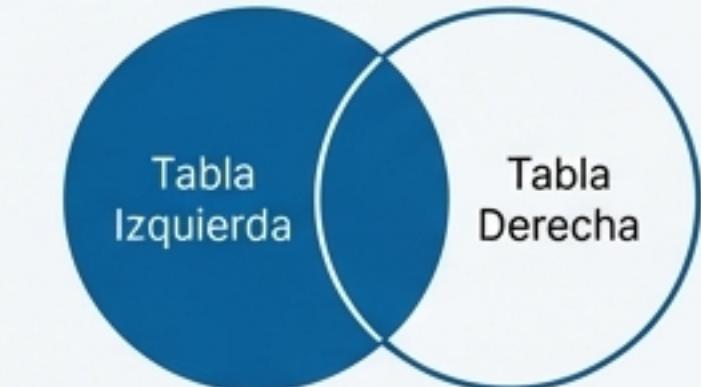


Use Case: 'Mostrar solo los clientes que han realizado pedidos'.

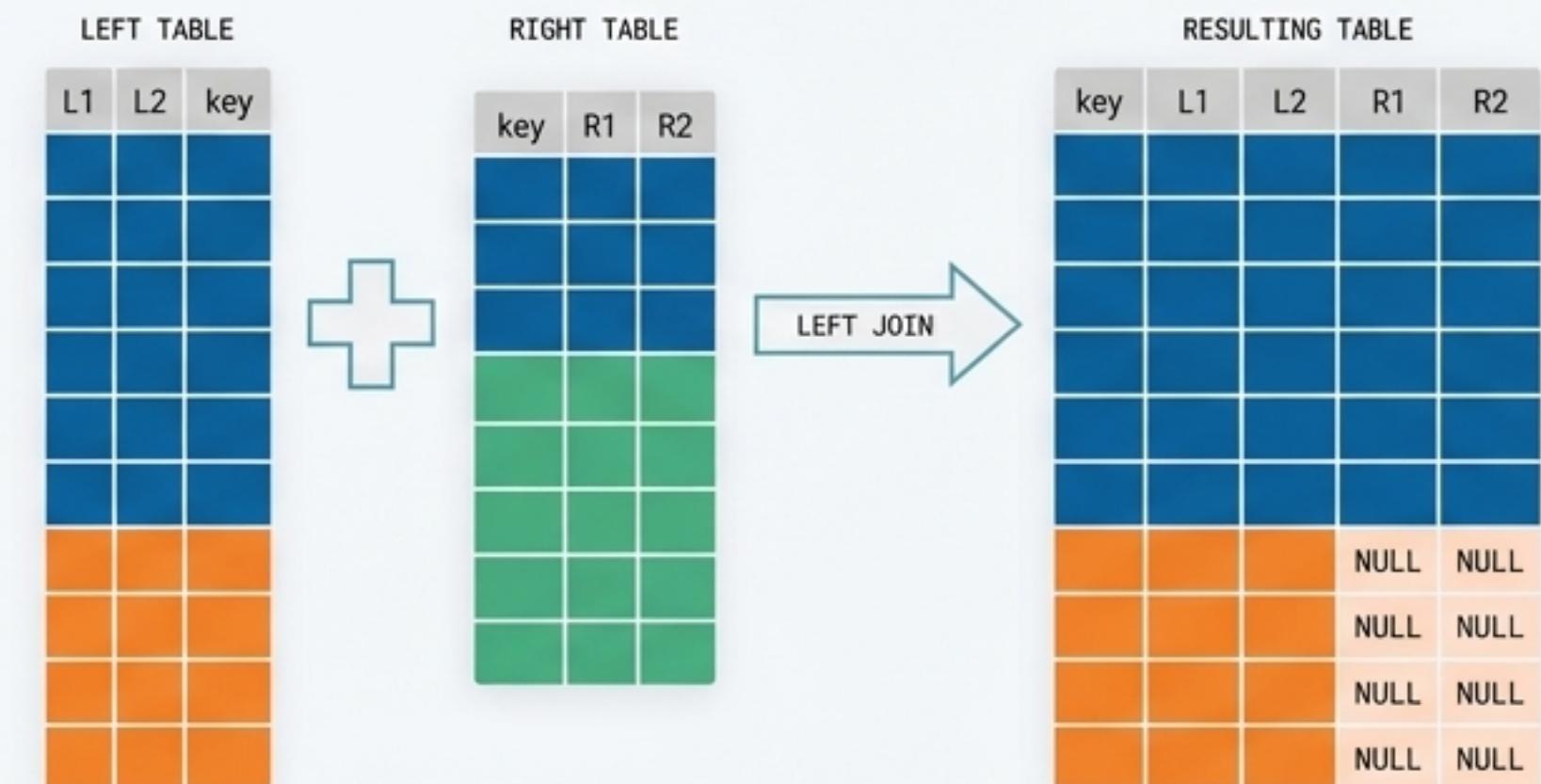


`LEFT JOIN` (Unión Externa Izquierda)

Devuelve **todas** las filas de la tabla izquierda y las coincidencias de la derecha. Si no hay coincidencia, las columnas de la derecha se rellenan con `NULL`.



Use Case: 'Listar **todos** los empleados, mostrando su departamento si lo tienen'.



Guía de Referencia Rápida de los Tipos de `JOIN`

Cada tipo de `JOIN` responde a una pregunta de negocio diferente. Entender sus matices es clave para obtener resultados precisos.

Tipo de Join	Descripción Lógica	Caso de Uso Típico
INNER JOIN	Intersección. Solo filas coincidentes.	Listar pedidos de clientes existentes.
LEFT JOIN	Todo el conjunto izquierdo + coincidencias.	Encontrar clientes sin pedidos; reportes maestro-detalle.
RIGHT JOIN	Todo el conjunto derecho + coincidencias.	Análogo a `LEFT JOIN` invirtiendo las tablas (uso menos frecuente).
FULL JOIN	Unión de ambos conjuntos. Todas las filas de A y B.	Auditoría de datos; encontrar registros no coincidentes en ambas tablas.
CROSS JOIN	Producto Cartesiano (A x B).	Generar todas las combinaciones posibles (ej. cada producto en cada tienda).

Consultas Dentro de Consultas: El Poder de las Subconsultas

Una subconsulta es una sentencia `SELECT` completa anidada dentro de otra consulta. La consulta interna se ejecuta primero y su resultado es utilizado por la consulta externa.

```
-- Listar los empleados cuyo salario es mayor que el  
salario promedio de toda la empresa.
```

```
SELECT nombre, salario  
FROM Empleados  
WHERE salario > (SELECT AVG(salario) FROM Empleados);
```

Paso 1: Se ejecuta la subconsulta. Devuelve un valor único. (Ej: 55000)

Paso 2: El resultado reemplaza a la subconsulta.

```
WHERE salario > 55000
```

Consideración de Rendimiento

- **No Correlacionada:** Se ejecuta una sola vez (como el ejemplo de arriba). Eficiente.
- **Correlacionada:** Se ejecuta una vez por cada fila de la consulta externa. Puede ser muy lenta y a menudo se puede reescribir como un `JOIN` para mejor rendimiento.

Anatomía de una Consulta: El Orden Lógico de Ejecución

Para depurar y optimizar consultas complejas, es vital entender el orden en que el motor de la base de datos procesa las cláusulas. No es el mismo orden en que se escriben.

- 1 `FROM` / `JOIN`** Identifica las tablas fuente y las combina.
- 2 `WHERE`** Filtra las filas individuales.
- 3 `GROUP BY`** Agrupa las filas restantes.
- 4 `HAVING`** Filtra los grupos ya formados.
- 5 `SELECT`** Procesa las expresiones, funciones y selecciona las columnas finales.
- 6 `DISTINCT`** Elimina duplicados del resultado.
- 7 `ORDER BY`** Ordena el conjunto de resultados final.
- 8 `LIMIT` / `TOP`** Restringe el número de filas devueltas.

De la Arquitectura a la Aplicación: La Maestría en SQL

Maestría en SQL

- El dominio de SQL trasciende la memorización de sintaxis.
- Comienza con la capacidad de leer el 'mapa': comprender el modelo relacional, la integridad referencial y las distinciones arquitectónicas como OLTP y OLAP.
- Continúa con la aplicación experta de las 'herramientas': elegir el **JOIN** correcto, filtrar con precisión y agregar datos para revelar insights.
- Un profesional que entiende el **porqué** detrás del **cómo** no solo escribe consultas, sino que diseña soluciones de datos robustas, eficientes y fiables.



Blandskron

```
SELECT default
FROM DGN_id
consumeof last_name
JOIN Jan_iomnu
JOIN conservatment_inMMJ
WHERE conservan_id
GROUP BY = (stow.aboyll.complist );
SELECT scinet
FROM _mra
transmistrinh2&1
JOIN _comebu
data = datas_i01
GROUP BY region
JOIN ventor = (n.compleee.complty)
WHERE swic.data
TUND data = ((onderhwiod.consulias));
SELECT
data = classbi2_k
JOIN URT region
JOIN vendor = cn.contain.longuline
GROUP BY vendor;
```