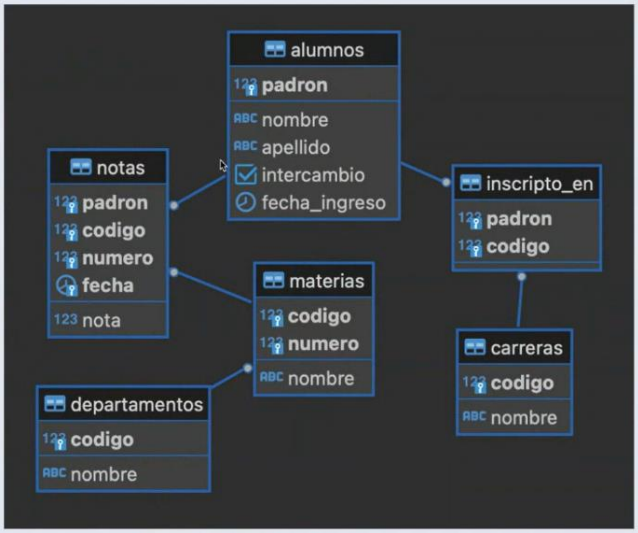
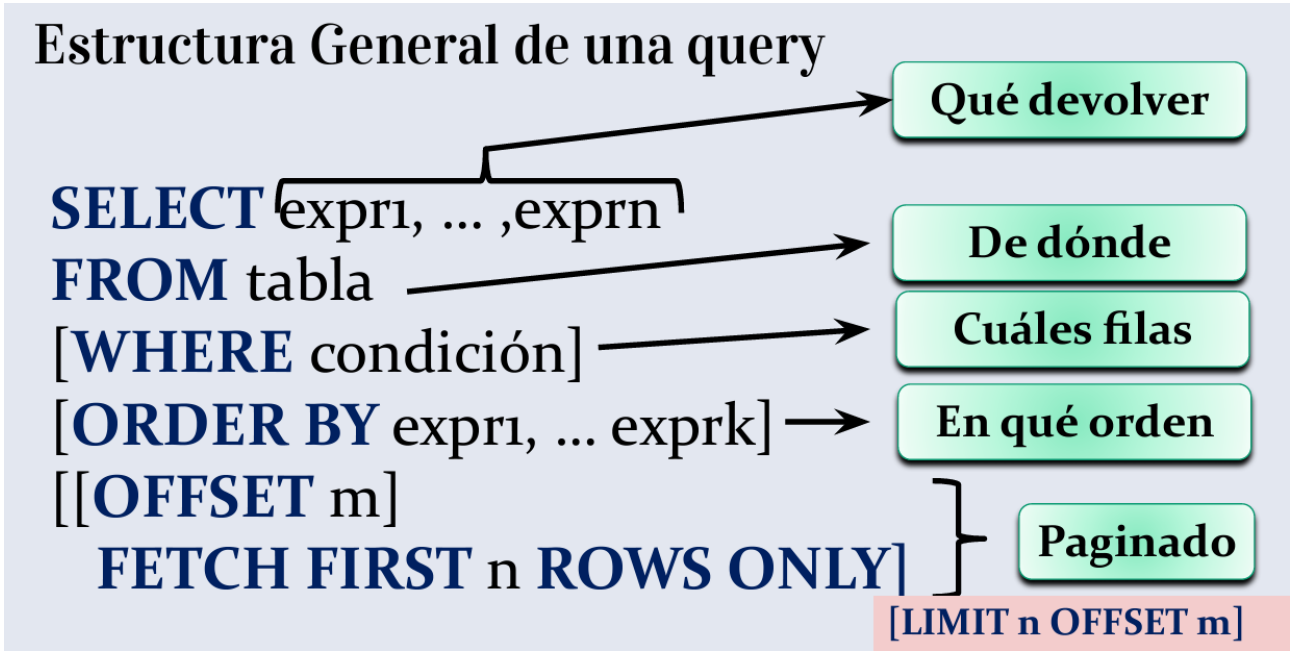
Base de datos a usar:

- Alumnos (padrón)
- Departamentos (código)
- Materias (código y número)
- Notas (padrón, código, número y fecha)
- Carreras (código)
- Inscripto_en (padrón y código)





Expresiones

No solo se trabaja con columnas, sino también con:

- Valores constantes
- Operadores (col1 + col2, operadores lógicos AND, OR y NOT
- Funciones (CURRENT_DATE, to_char(exp, formato), LOWER('A')

Trabajando con CASE sensitive (ej 5)

- Es importante porque puede ser que los datos no estén estandarizados
- Comparamos con el ILIKE, que no usa índices, y para bases pesadas va a tardar mucho
- Para esos casos conviene usar las funciones UPPER o LOWER para búsquedas (se puede definir índices sobre estas funciones)

Funciones de agregación

- SUM y AVG

- MAX y MIN
- COUNT
    - Opción DISTINCT : Si se repiten que las cuente 1 sola vez.
- El utilizarlas cambia cuántos resultados son devueltos
    - Sin GROUP BY, 1 único resultado
    - Con GROUP BY, 1 resultado por cada distinta combinación de valores para las expresiones de agrupación (más adelante)

Joins

Es común necesitar info de varias tablas, para:

- Devolver datos de distintas tablas
- Ver que se cumplan criterios

Para involucrar más de una tabla podemos usar:

1. **Operadores de conjuntos**

Los operadores son:

- UNION
- INTERSECT
- EXCEPT

Para poder usarlos, las dos consultas a combinar deben ser compatibles:

- ✓ Misma cantidad de columnas devueltas
- ✓ Mismo tipo de dato de cada columna

Por defecto, no hay duplicados. Podemos usar la opción ALL para evitar esto.

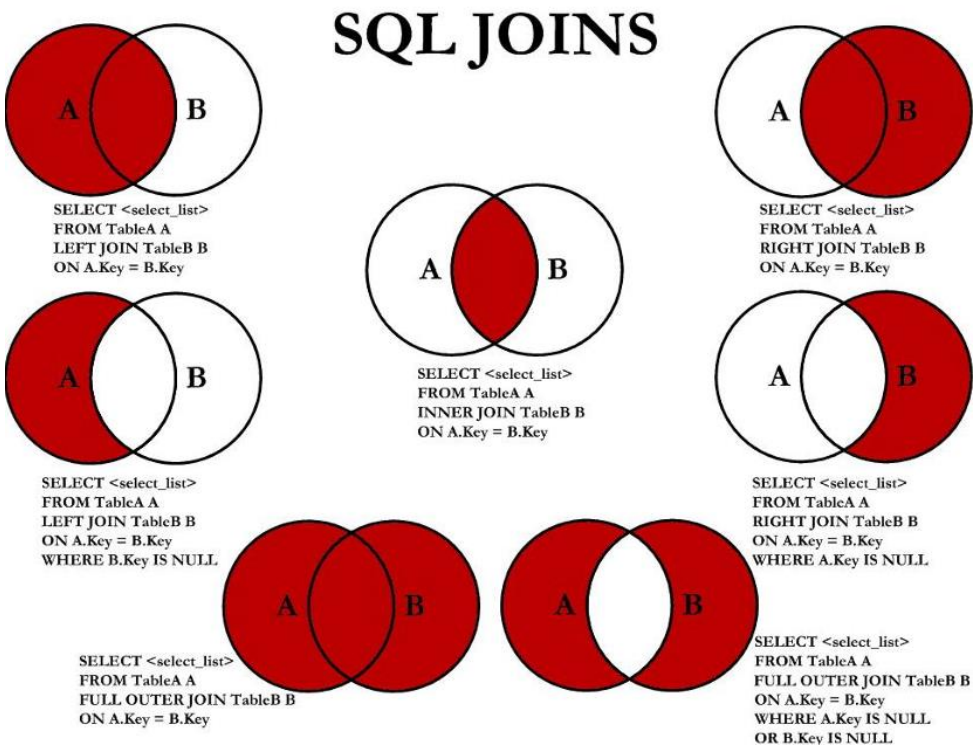
2. **Combinaciones (JOINS)**

Opciones para JOIN:

- Hacer producto cartesiano y selección
    - ***FROM tabla1, tabla2 WHERE (condición_de_join)***
- Usar **INNER JOIN**
    - ***FROM tabla1 INNER JOIN tabla 2***
      ON (condicion_de_join)
    - ***FROM tabla1 INNER JOIN tabla 2***
      USING (col1, col2) a veces es mas claro usar el ON.
- Usar NATURAL JOIN (No recomendado en aplicativos porque puede haber casos en los cuales se pierde info/te da info de más; sobre todo cuando haces joins de texto)

OUTER JOIN

- Es similar al inner join pero devuelve filas de una tabla que **NO** se hayan vinculado con filas de la otra
    - Tabla1 **LEFT OUTER JOIN** table2: devuelve siempre todas las filas de tabla1 y si no hay matcheos con las columnas de la tabla 2, rellena con Null.
    - Tabla1 **RIGHT OUTER JOIN** tabla2: devuelve siempre todas las filas de tabla2
    - Tabla1 **FULL OUTER JOIN** tabla2: devuelve todas las filas de ambas tablas.
- Se devuelve valor NULO para las columnas de la tabla no combinada



3. Subconsultas

Siempre va a ser mas lento usar una subconsulta a que la tabla original. El problema con las subconsultas, es que voy a estar trabajando con cosas no indexadas.

El resultado de una subconsulta puede ser utilizado como si fuera una tabla de las siguientes maneras:

3.1. como valor

Si siempre se devuelve una única columna de una única fila, se puede usar como un valor de la siguiente manera:

SELECT …

FROM …

WHERE columna = (SELECT… FROM…) esto tiene que ser algo que me de UN solo dato, como un SUM, COUNT, AVG, etc.

3.2. como tabla en el JOIN

SELECT …

FROM tabla1 INNER JOIN

(SELECT… FROM…) alias tblName

3.3. con operadores de conjunto ALL, ANY, SOME

las opciones mas comunes son usar operadores de conjunto con comparación:

- columna = **ANY** (SELECT… FROM…) que esté adentro de al menos 1
- columna <> **ALL** (SELECT… FROM…) que sea distinto a todos
- columna > **ANY** (SELECT… FROM…) que sea mayor a todos (muy útil para un max o min)

4. **con IN y EXISTS**

el operador **IN** se puede usar con:

- valores **fijos:** Columna **IN** (valor1, valor2, valor3)
- subconsulta: columna **IN** (SELECT… FROM…)

IN equivale a **= ANY,** NOT IN equivale a **<> ALL** (not all)

El operador **EXISTS** es *true* si la subconsulta devuelve al menos una fila y es *false* si no devuelve nada. En general se usan en consultas correlacionadas (su costo es mas alto)

- **WHERE [NOT] EXISTS (SELECT… FROM…)**
- Como no importa que es lo que devuelve la subconsulta, es común hacer que devuelva un valor fijo: **(SELECT 1 FROM…)**

Agrupamiento

Se pueden generar varios grupos para aplicar funciones de agregación entre ellos. Es importante definir bien qué valores se utilizarán para agrupar.

- Se puede utilizar **HAVING** para filtrar grupos (condición de grupo)
- si la condición se puede evaluar fila a fila, conviene usar **WHERE,** que puede aprovechar índices.

Comparaciones de valores

Comparar valores que surgen de una agregación. Metodología general:

- Consulta que para cada grupo tenga ese valor
- Utilizar dicha consulta también como subconsulta
- Comparar valores

Casos más comunes:

- Que sea mayor/menor o igual a todos
- Que no exista uno mayor

Division en SQL

Se puede encarar de tres maneras:

1. Doble NOT EXISTS

Surge de una equivalencia lógica. Ej: Si un alumno tiene nota en todas las materias, entonces no existe materia en la que no tenga nota.

2. Resta

Surge de una restar dos conjuntos para cada alumno

- Todas las materias del sistema
- Todas las materias en las que tiene notas el alumno

Si un alumno tiene nota en todas las materias, el resultado de la resta es el conjunto vacío (Usar NOT EXISTS para revisar esto)

3. Con agrupación

Se comparan dos cantidades:

- La cantidad total de materias
- La cantidad de materias en las que cada alumno tiene nota

Si un alumno tiene nota en todas las materias, las dos cantidades son iguales