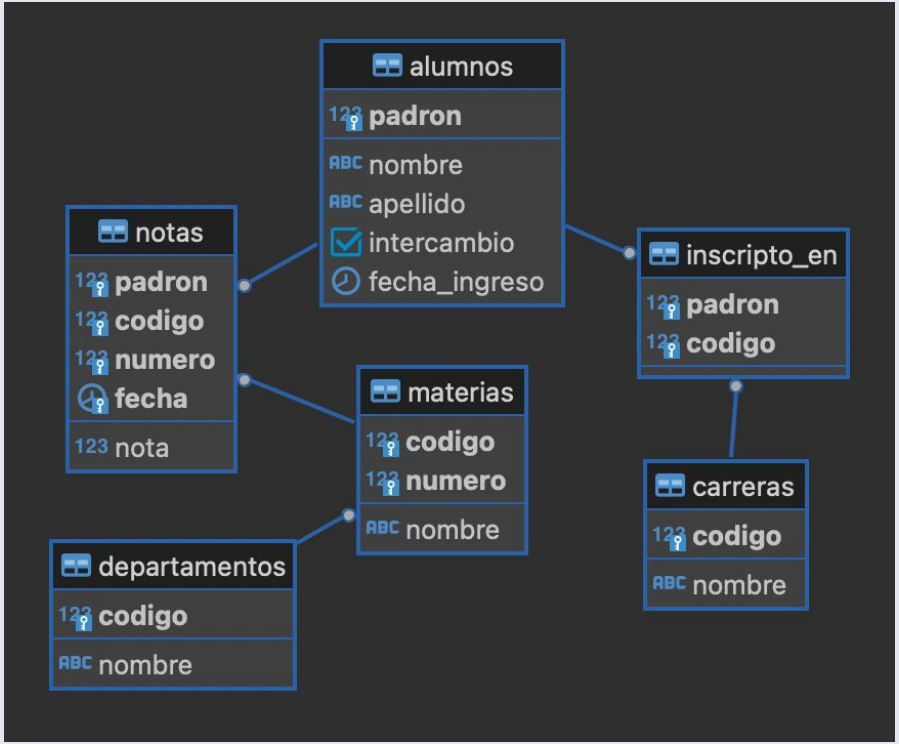
SQL – TALLER IV

SQL - Structured Query Language

- Es un lenguaje declarativo
- Distintas formas de resolver una misma consulta
- Funcionalidad standard y no standard

Vamos a usar la siguiente base: Facultad

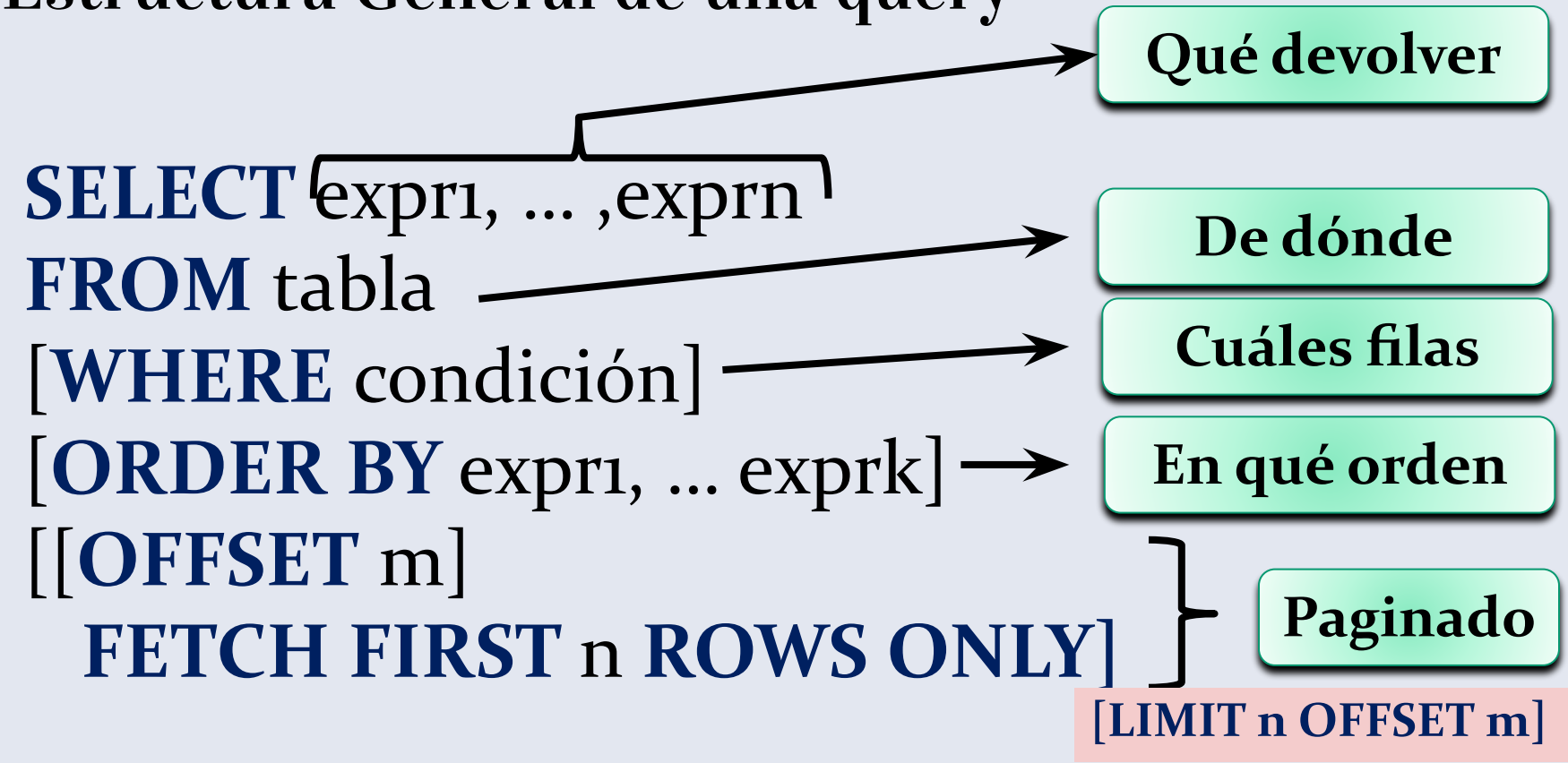
- Alumnos (padrón)
- Departamentos (código)
- Materias (código y número)
- Notas (padrón, código, número y fecha)
- Carreras (código)
- Inscripto_en (padrón y código)



Hagamos algunas queries!!!

(o consultas a la base de datos)

Estructura General de una query



Expresiones

No sólo se puede trabajar con valores de columnas en las filas. También podemos usar:

- Valores constantes:
  - Ej: 10, ‘Hola’, ‘2020-10-21’
- Operadores:
  - Ej: columna1 **+** columna2, ‘Nombre: ’ **||** nombre
  - Importante los operadores logicos: **AND**, **OR** y **NOT**
- Funciones
  - Ej: CURRENT_DATE, to_char(exp, formato), LOWER(‘A’)

Parte A

Resolver ejercicios 1 a 6

Trabajar con CASE sensitive

- Definir un modo de almacenar
  - Una equivocación trae problemas
  - Igual es buena práctica para estandarizar datos
- Comparar con **ILIKE**
  - No aprovecha índices
- Utilizar funciones **UPPER** o **LOWER** para búsquedas
  - Se puede definir índices sobre estas funciones

Agregación

- Funciones de agregación
  - **SUM** y **AVG**
  - **MAX** y **MIN**
  - **COUNT**
    - Opción **DISTINCT**
- 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)

Parte B

Resolver ejercicios 7 a 10

Varias tablas

- Es común necesitar información de varias tablas
  - Para devolver datos de distintas tablas
  - Para ver que se cumplan criterios
- Tres opciones de involucrar más de una tabla
  - Operadores de conjuntos
  - Combinaciones (JOINS)
  - Subconsultas

Operadores de conjunto

- Uso de operadores de conjuntos
  - **UNION**
  - **INTERSECT**
  - **EXCEPT**
- 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
  - Opción ALL para evitar esto`

Parte C

Resolver ejercicios 11 a 12

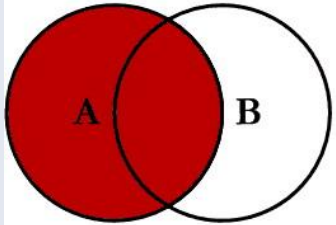
Opciones para JOIN

- Hacer producto cartesiano y selección
  - **FROM** tabla1, tabla2 **WHERE** (Condicion_de_join)
- Usar **INNER JOIN**
  - **FROM** tabla1 **INNER JOIN** tabla2
    ON (Condicion_de_join)
  - **FROM** tabla1 **INNER JOIN** tabla2
    USING (columna1, columna2)
- Usar **NATURAL JOIN**
  - No recomendado en aplicativos

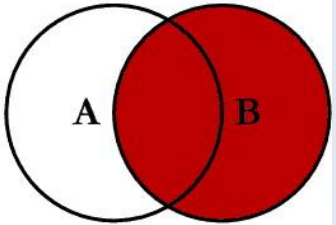
OUTER JOIN

- Similar a **INNER JOIN** pero devuelve filas de una tabla que **no** se hayan vinculado con filas de la otra
  - tabla1 **LEFT OUTER JOIN** tabla2 devuelve siempre todas las filas de tabla1
  - 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

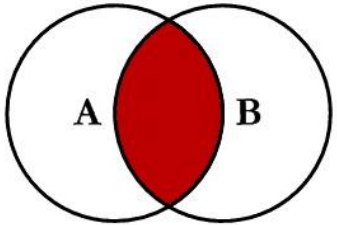
SQL JOINS



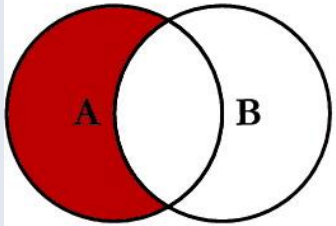

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



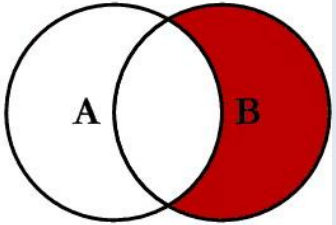

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



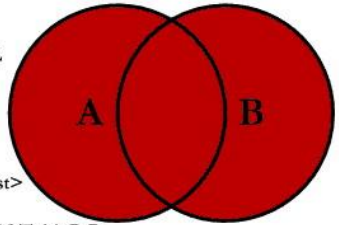

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



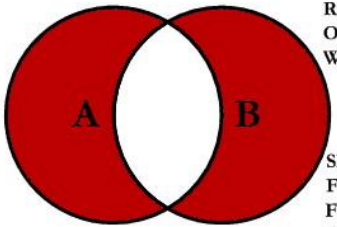

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Subconsultas

- El resultado de una subconsulta puede ser utilizado como si fuera una tabla
- Distintas opciones
  - Como valor
  - Como tabla en el **JOIN**
  - Con operadores de conjunto **ALL**, **ANY** y **SOME**
  - Con **IN** y **EXISTS**

Subconsultas II

- Si siempre devuelve una única columna de una única fila se puede usar como un valor
  - **SELECT** ...
    FROM ...
    WHERE columna = (**SELECT** .... **FROM** ...)
- Se puede usar como tabla en el join
  - **SELECT** ...
    FROM tabla1 **INNER JOIN**
    (**SELECT** ... **FROM** ...) alias tblName

Subconsultas III

- Operadores de conjunto con comparación
- Más comunes:
  - Columna = **ANY** (SELECT …. FROM ….)
  - Columna <> **ALL** (SELECT …. FROM ….)
  - Columna > **ALL** (SELECT …. FROM …..)
- Operador IN
  - Valores fijos: Columna **IN** (valor1, valor2, valor3)
  - Subconsulta: Columna **IN** (SELECT … FROM ….)
  - **IN** equivale a = **ANY**, **NOT IN** equivale a <> **ALL**

Subconsultas IV

- **EXISTS**: Verdadero (true) si la subconsulta devuelve al menos una fila y es Falso (false) si no devuelve nada
  - **WHERE** [**NOT**] **EXISTS** (**SELECT** … **FROM** ….)
  - Como no importa qué devuelve la subconsulta, es común hacer que devuelva un valor fijo (**SELECT** 1 **FROM** …..)
- En general se usan en consultas correlacionadas (su costo es más alto)

Parte D

Resolver ejercicios 13 a 19

Agrupamiento

- Se pueden generar varios grupos para aplicar funciones de agregación en ellos
- Es importante definir qué valores se utilizarán para agrupar
  - Hacer grupos por alumnos? Agrupar por padrón
  - Hacer grupos por materia? Por código y número
- Utilizar **HAVING** para filtrar grupos (condicion de grupo)
  - Si la condición se puede evaluar fila a fila, utilizar **WHERE** que puede aprovechar índices

Parte E

Resolver ejercicios 20 a 22

Quién tiene mayor/menor …

- 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

División en SQL

- Tres formas de encarar una división:

    1. Doble NOT EXISTS
    2. Resta
    3. Con agrupación

1 - Doble NOT EXISTS

- Surge de una equivalencia lógica
  - $\forall x: P(x) \leftrightarrow \neg \exists x: \neg P(X)$
- Si un alumno tiene nota en todas las materias, entonces no existe materia en la que no tenga nota
- Buscar alumnos para los que no exista materia en la que no exista nota de ese alumno en esa materia

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

- Comparar 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

Parte F

Resolver ejercicios 23 a 25