Curso

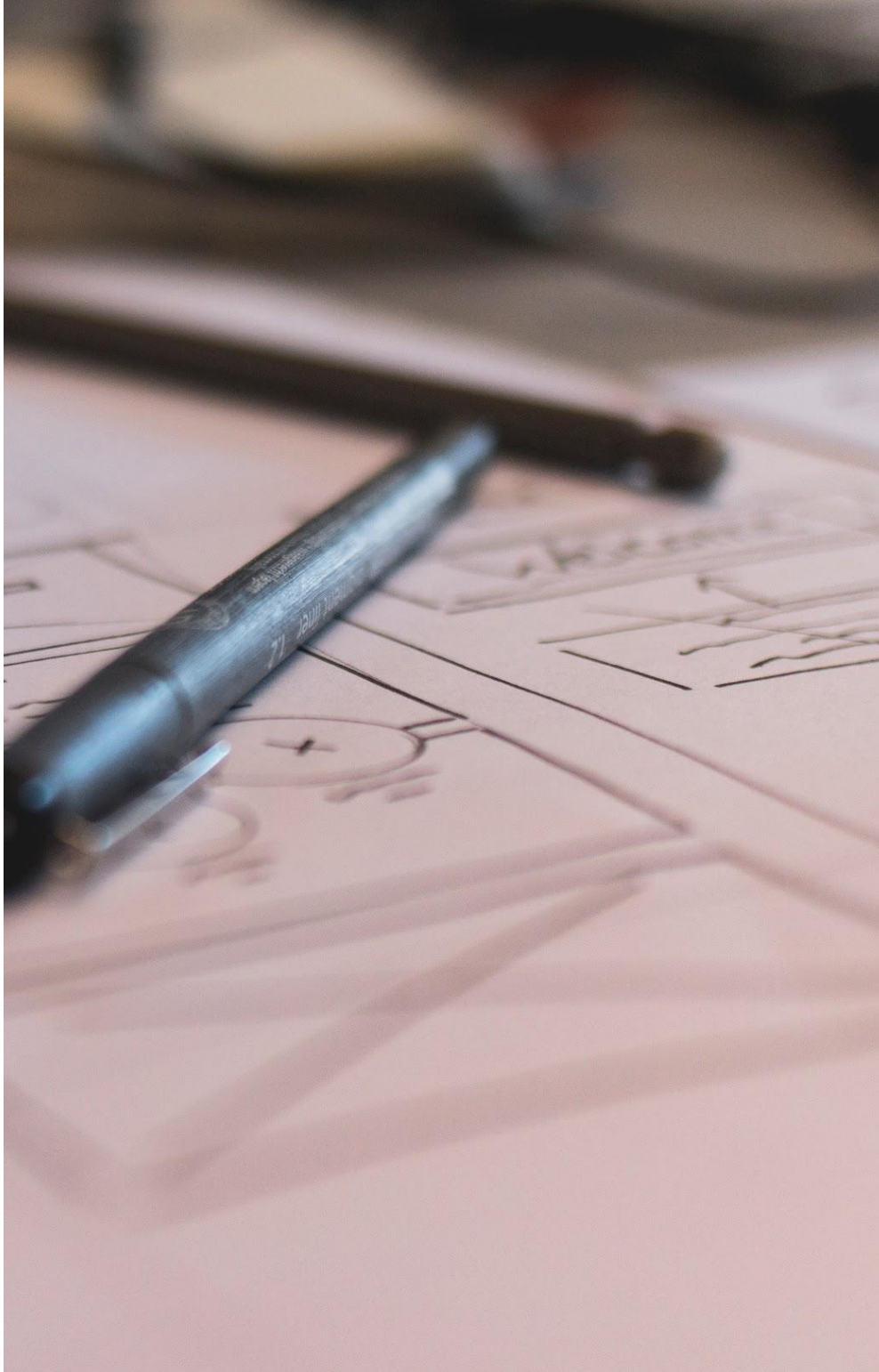
Práctico de SQL

Israel Vázquez

Breve historia de SQL

Bases

Modelo y álgebra relacional de Codd



IBM

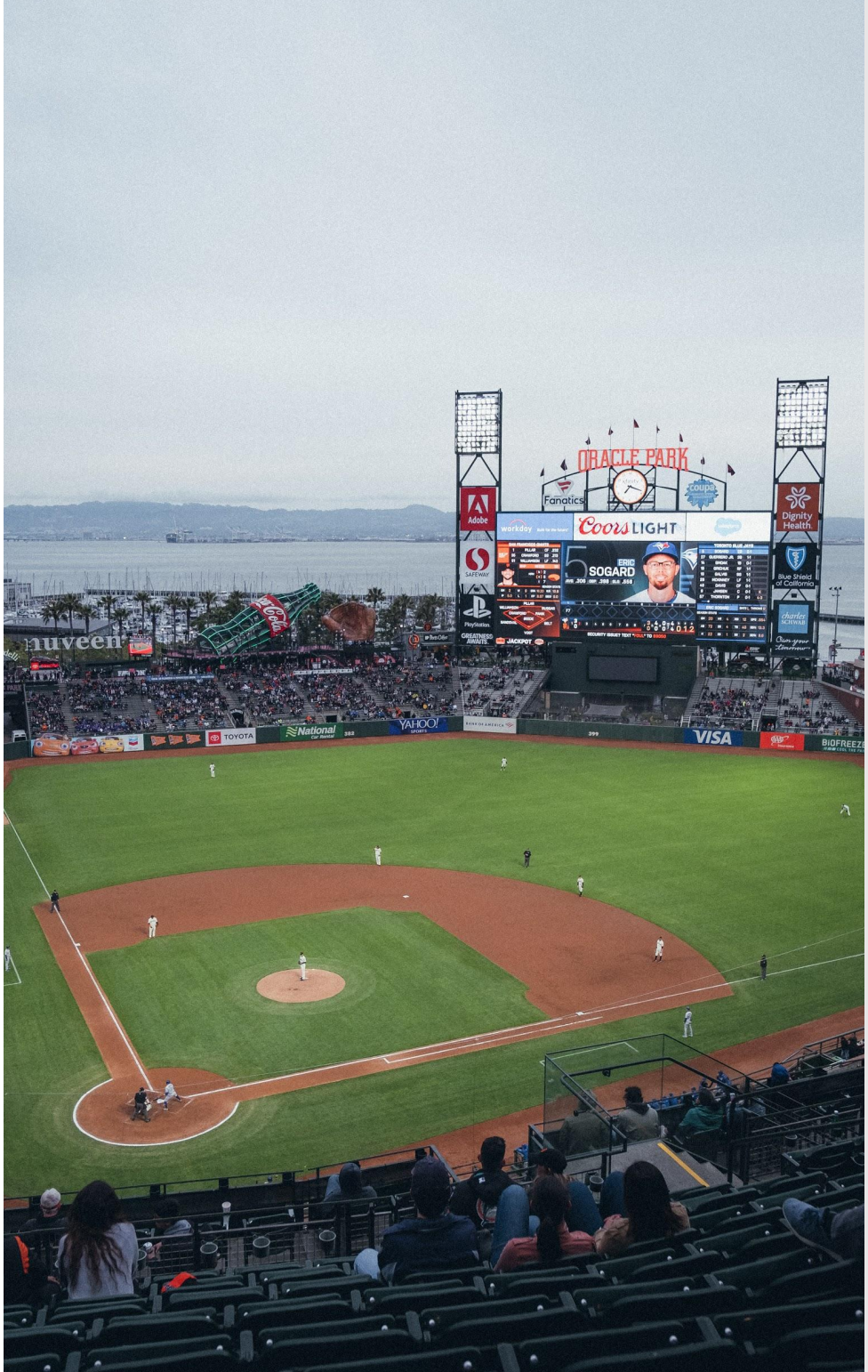
Fue creada en los años 70.

Originalmente llamada SEQUEL.

- System/38 (1979)
- SQL/DS (1981)
- DB2 (1983)

Relational company

Creó el software Oracle V2 en 1979



ANSI / ISO

1989, 1992, 1996,

1999, 2003, 2006,

2008, 2011 and 2016

Álgebra relacional

Proyección

Select

```
SELECT *;
```

```sql
SELECT field AS alias;
```

```sql
SELECT COUNT(id), SUM(quantity),
AVG(age);
```

```sql
SELECT MIN(date), MAX(quantity);
```

```sql
SELECT IF(500<1000, "YES", "NO");
```

```sql
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN "Over 30"
    WHEN Quantity = 30 THEN "Equal 30"
    ELSE "Under 30"
END AS QuantityText
```

Origen

From

```sql
SELECT *
FROM tabla_diaria;
```

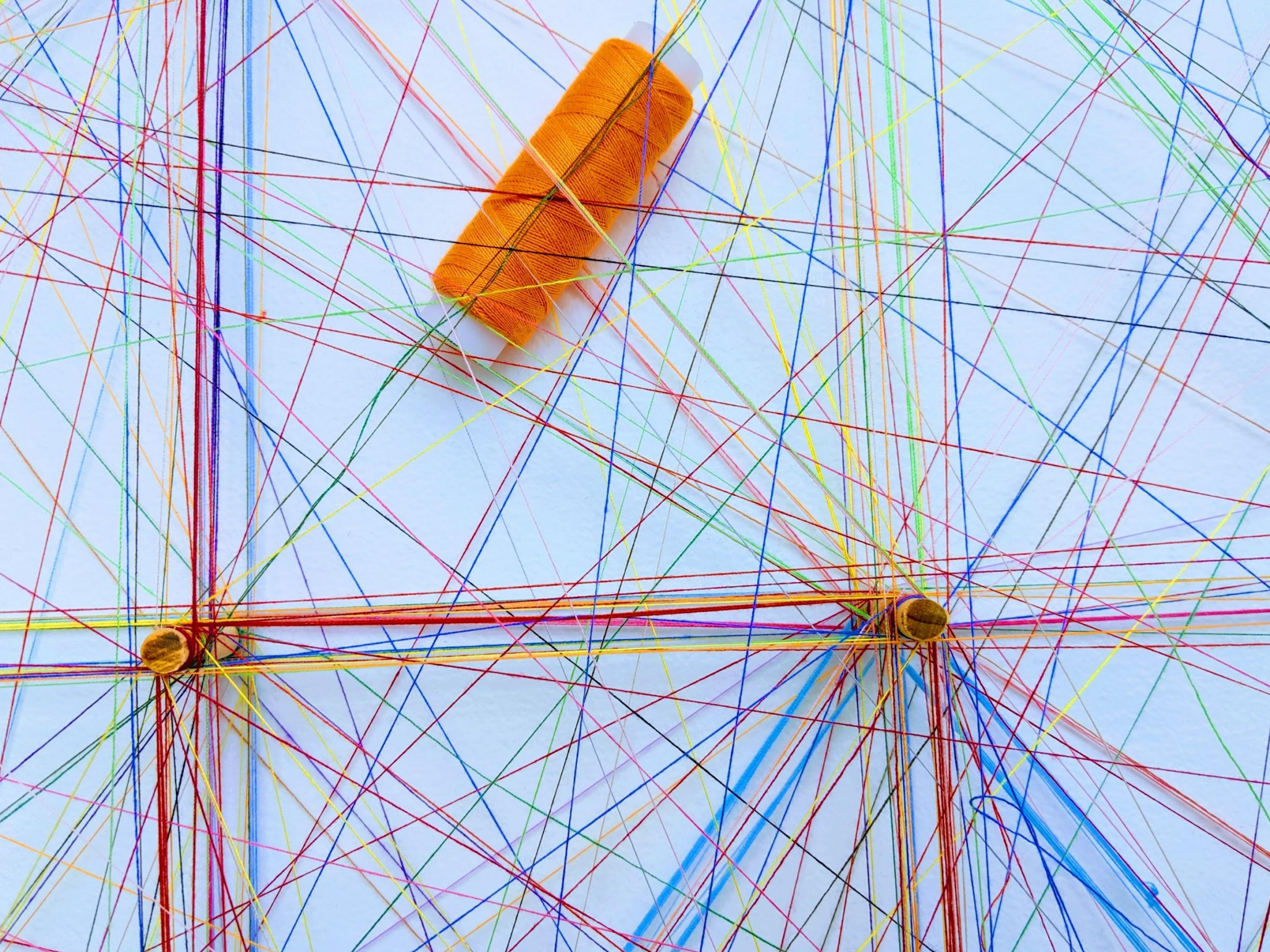
```sql
SELECT *
FROM tabla_diaria AS td
  JOIN tabla_mensual AS tm
  ON td.pk = tm.fk;
```

```
SELECT *
FROM dblink('
  dbname=somedb
  port=5432 host=someserver
  user=someuser
  password=somepwd',
  'SELECT gid, area, perimeter,
          state, county,
          tract, blockgroup,
          block, the_geom
  FROM massgis.cens2000blocks')
  AS blockgroups
```

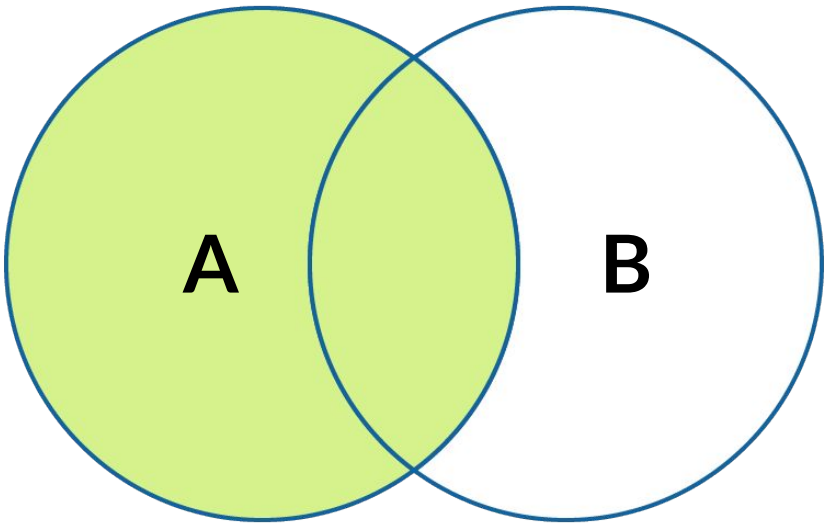
Productos cartesianos

Join

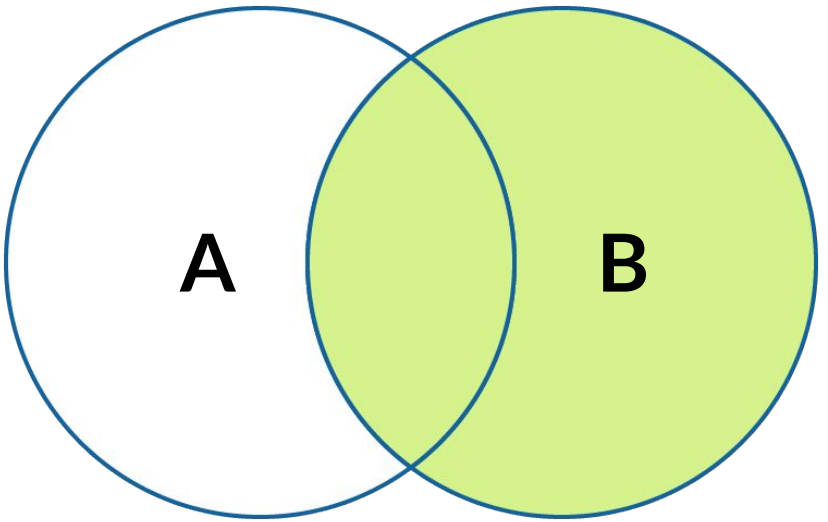
```sql
SELECT *
FROM tabla_diaria AS td
  JOIN tabla_mensual AS tm
  ON td.pk = tm.fk;
```



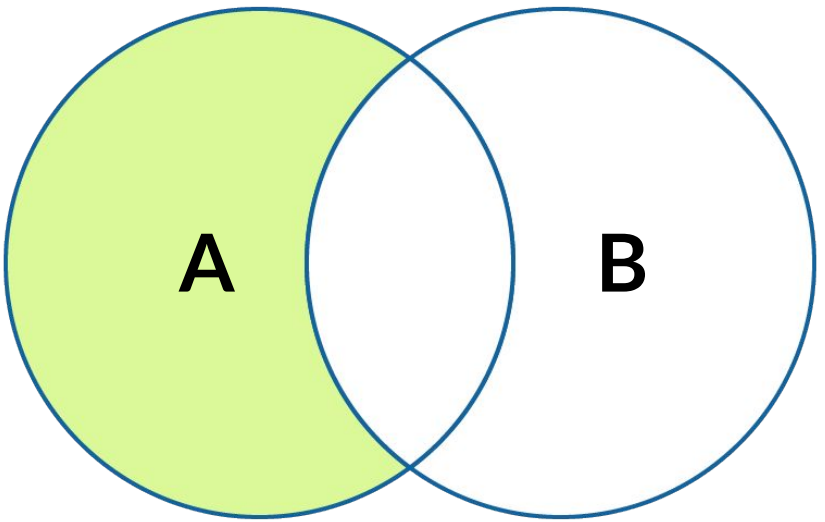
Left Join



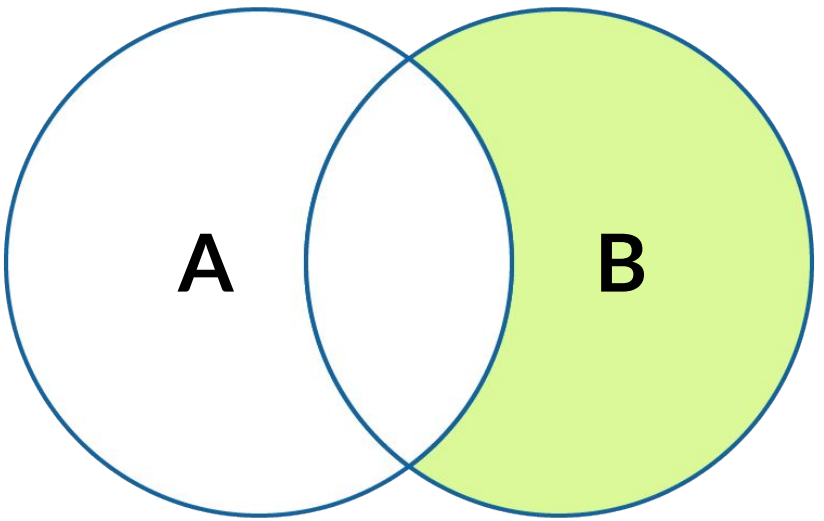
Right Join



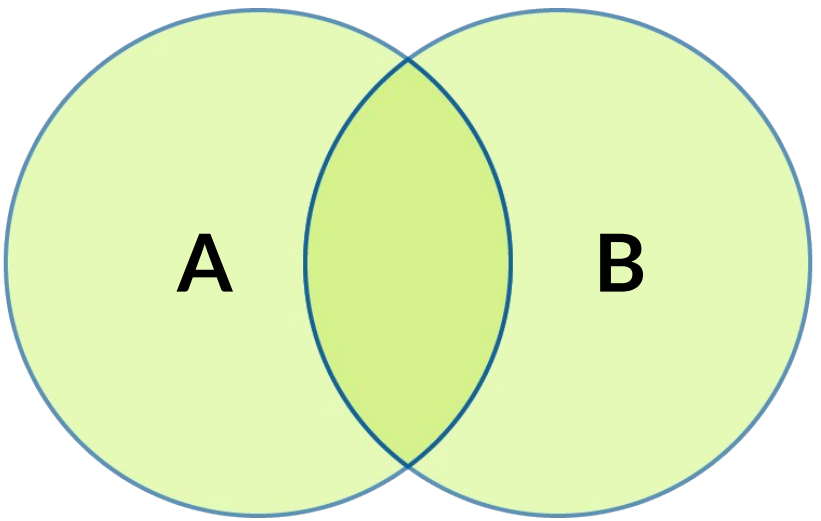
Exclusive Left Join



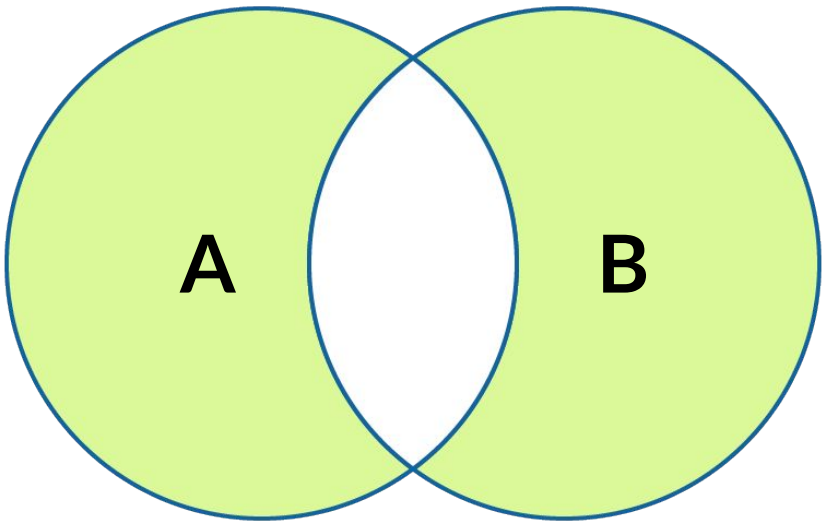
Exclusive Right Join



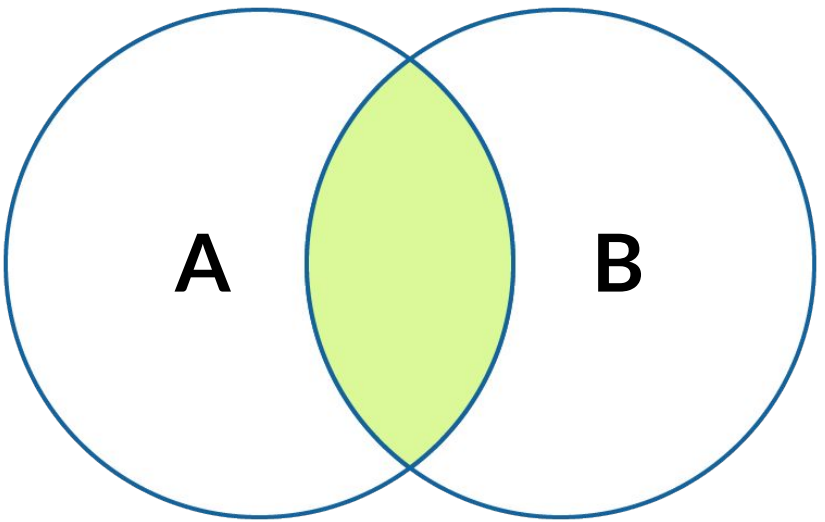
Full Outer Join



Exclusive Full Outer Join



Inner Join



Selección

Where

```sql
SELECT *
FROM tabla_diaria
WHERE id = 1;
```

```sql
SELECT *
FROM tabla_diaria
WHERE cantidad > 10;
```

```sql
SELECT *
FROM tabla_diaria
WHERE cantidad < 100;
```

```sql
SELECT *
FROM tabla_diaria
WHERE  cantidad > 10
   AND cantidad < 100;
```

```sql
SELECT *
FROM tabla_diaria
WHERE  cantidad BETWEEN 10
   AND 100;
```

```sql
SELECT *
FROM users
WHERE name = "Israel"
  AND (
    lastname = "Vázquez"
    OR
    lastname = "López"
  );
```

```sql
SELECT *
FROM users
WHERE name = "Israel"
  AND lastname = "Vázquez"
  OR lastname = "López";
```

```sql
SELECT *
FROM users
WHERE name LIKE "Is%";
```

```sql
SELECT *
FROM users
WHERE name LIKE "Is_ael";
```

```sql
SELECT *
FROM users
WHERE name NOT LIKE "Is_ael";
```

```sql
SELECT *
FROM users
WHERE name IS NULL;


SELECT *
FROM users
WHERE name IS NOT NULL;


SELECT *
FROM users
WHERE name IN ('Israel','Laura',
'Luis');
```

Ordenamiento

Order By

```sql
SELECT *
FROM tabla_diaria
ORDER BY fecha;
```

```sql
SELECT *
FROM tabla_diaria
ORDER BY fecha ASC;
```

```sql
SELECT *
FROM tabla_diaria
ORDER BY fecha DESC;
```

Índices

Excelentes para búsquedas y ordenamientos.

Cuidar para alta transaccionalidad.

Agregación

Group By

```sql
SELECT *
FROM tabla_diaria
GROUP BY marca;
```

```sql
SELECT *
FROM tabla_diaria
GROUP BY marca, modelo;
```

Limitantes

Limit

```sql
SELECT *
FROM tabla_diaria
LIMIT 1500;
```

```sql
SELECT *
FROM tabla_diaria
OFFSET 1500
LIMIT 1500;
```

El primero

El segundo más alto

Seleccionar de un set de opciones

En mis tiempos

Seleccionar por año

Double trouble

Selectores de rango

Eres lo máximo

Egoísta (Selfish)

Resolviendo diferencias

Todas las uniones

Triangulando

Generando rangos

Regularizando expresiones

Bases de datos distribuidas

¿Qué es?

Es una colección de múltiples bases de datos separadas físicamente que se comunican mediante una red informática.

Ventajas

- Desarrollo modular.
- Incrementa la confiabilidad.
- Mejora el rendimiento.
- Mayor disponibilidad.
- Rapidez de respuesta.

Desventajas

- Manejo de seguridad.
- Complejidad de procesamiento.
- Integridad de datos más compleja.
- Costo.

Homogéneas y heterogéneas

- OS
- Sistema de base de datos
- Modelos de datos

Arquitecturas

- Cliente - servidor.
- Par a par
  (Peer 2 Peer).
- Multi manejador
  de base de datos.

Estrategias de diseño

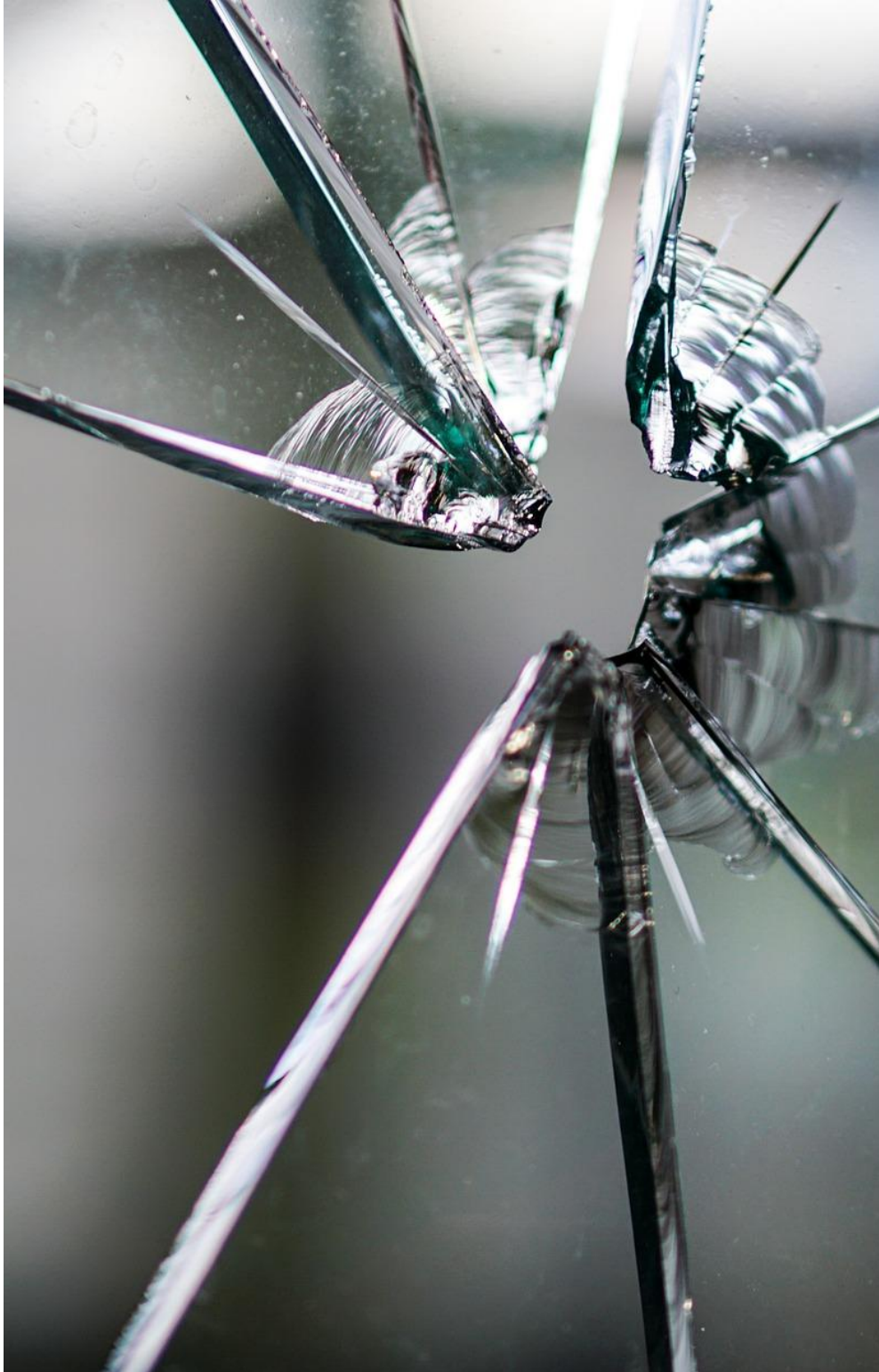
- Top down
- Bottom up

Almacenamiento distribuido

- Fragmentación
- Replicación
- Distribución

Fragmentación

- Horizontal
- Vertical
- Mixta



Replicación

- Completa
- Parcial
- Sin replicación

Distribución de los datos

- Centralizada
- Particionada
- Replicada

Queries distribuidos

Tabla	Columnas		Tuplas	Localización
Proveedores(P)	P#	Ciudad	10,000	Región A
Repuestos (R)	R#	Color	100,000	Región B
Prov-Rep(PR)	P#	R#	1,000,000	Región A

- Cada tupla tiene 25B (200b).
- El query será: obtener los números de proveedor de Bogotá que tienen repuestos de color rojo.
- P >< PR >< R.

Tabla	Columnas		Tuplas	Localización
Proveedores(P)	P#	Ciudad	10,000	Región A
Repuestos (R)	R#	Color	100,000	Región B
Prov-Rep(PR)	P#	R#	1,000,000	Región A

- **Suposiciones de cardinalidad:**
  - Repuestos rojos: 10.
  - Número de pedidos hechos por proveedores de Bogotá: 100,000.

- **Suposiciones de comunicación:**
  - Tasa de transferencia: 50,000 bits por segundo.
  - Latencia de acceso: 0.1s.

Tabla	Columnas		Tuplas	Localización
Proveedores(P)	P#	Ciudad	10,000	Región A
Repuestos (R)	R#	Color	100,000	Región B
Prov-Rep(PR)	P#	R#	1,000,000	Región A

- Retraso total en la comunicación = (retraso total de acceso) + (volumen total de datos/tasa de transferencia).
- Retraso total en la comunicación = (número de mensajes/10) + (número de bits/50,000).

Retraso total en la comunicación =

(número de mensajes/10) + (número de bits/50,000)

#	Técnica	Estimación	Tiempo total
1	Mover R a Región A	(0.1) + (100,000*200)/50,000=400s=6.67min	6.67 min
2	Mover P y PR a Región B	(0.2) + ((10,000+1,000,000)*200)/50,000 = 4,040s = 1.12h	1.12 h
3	Revisar repuestos rojos para cada proveedor de Bogotá	Join de P y PR en A filtrar los proveedores de Bogotá y hacer un join con R en B	5.56 h
4	Por cada repuesto rojo revisar si hay un proveedor en Bogotá	Filtrar R en B y por los 10 resultados hacer un join remoto con PR en A	2 s
5	Mover los pedidos de Bogotá a B	0.1 + (100,000*200)/50,000 = 400s = 6.67min	6.67 min
6	Mover los repuestos rojos a A	0.1 + (10*200)/50,000 = 0.10s	0.10 s

Sharding

Problemas

- Joins entre shards.
- Baja elasticidad
- Reemplaza PK

Window Functions

¿Qué son?

Realizan cálculos en algunas tuplas que se encuentran relacionadas con la tupla actual.

¿Para qué sirven?

Evitan el uso de self joins y reduce la complejidad alrededor de la analítica, agregaciones y uso de cursores.

Window Functions: Particiones y agregación

El futuro de SQL