Modelo Relacional

Agradecimientos

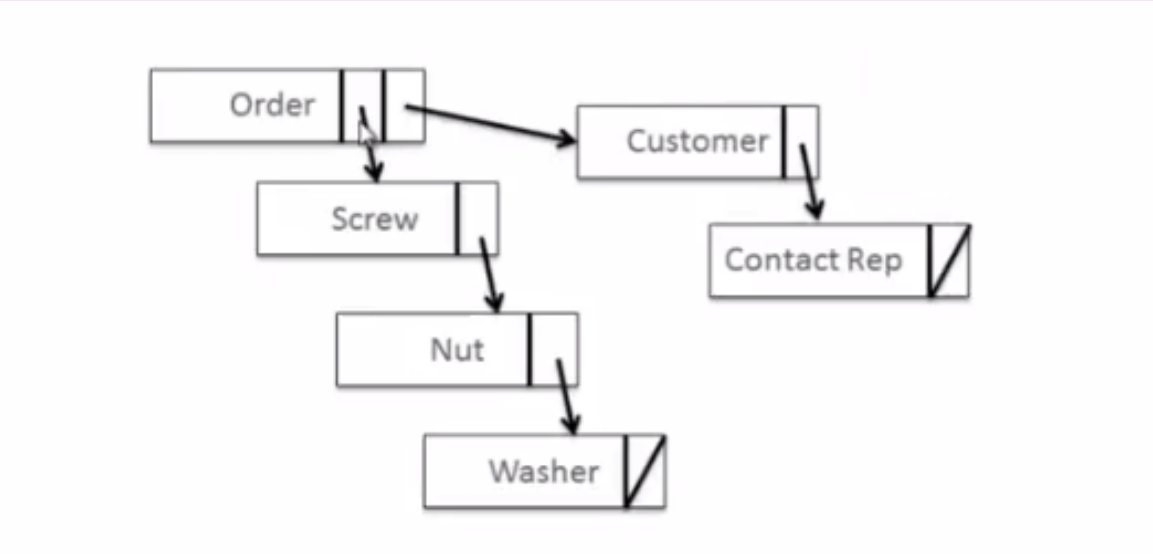
- Este curso es una versión libre del curso on-line Introducción a la Ciencia de Datos (Bill Howe, Univ. de Wasshington) https://www.coursera.org/course/datasci

¿Para qué queremos una Base de Datos?

- Compartir información
  - Permitir accesos simultáneos de lectores y escritores
- Escalabilidad
  - Operar sobre volúmenes de datos que superen el tamaño de la memoria
- Flexibilidad
  - Usar la información de modos novedosos no previstos

Motivación histórica

- Antes de las bases de datos relacionales la información estaba organizada en archivos con punteros
- Problemas
  - Los punteros se rompían si agregabamos un campo nuevo a un registro
  - La información se organizaba según el tipo de acceso que se requería de antemano (no se podía reusar la información de modos no previstos)



Independencia de Datos

El objetivo principal que dio origen al modelo relacional fue el de garantizar la *independencia de datos*.

La noción de *independencia de datos* es que los datos y los programas que acceden a los datos sean independientes. Es decir, que la modificación de la estructura de los datos no requiera que los programas y aplicaciones tengan que modificarse.

Ejemplo: agregar una nueva columna a una tabla.

Motor de una Base de Datos Relacional

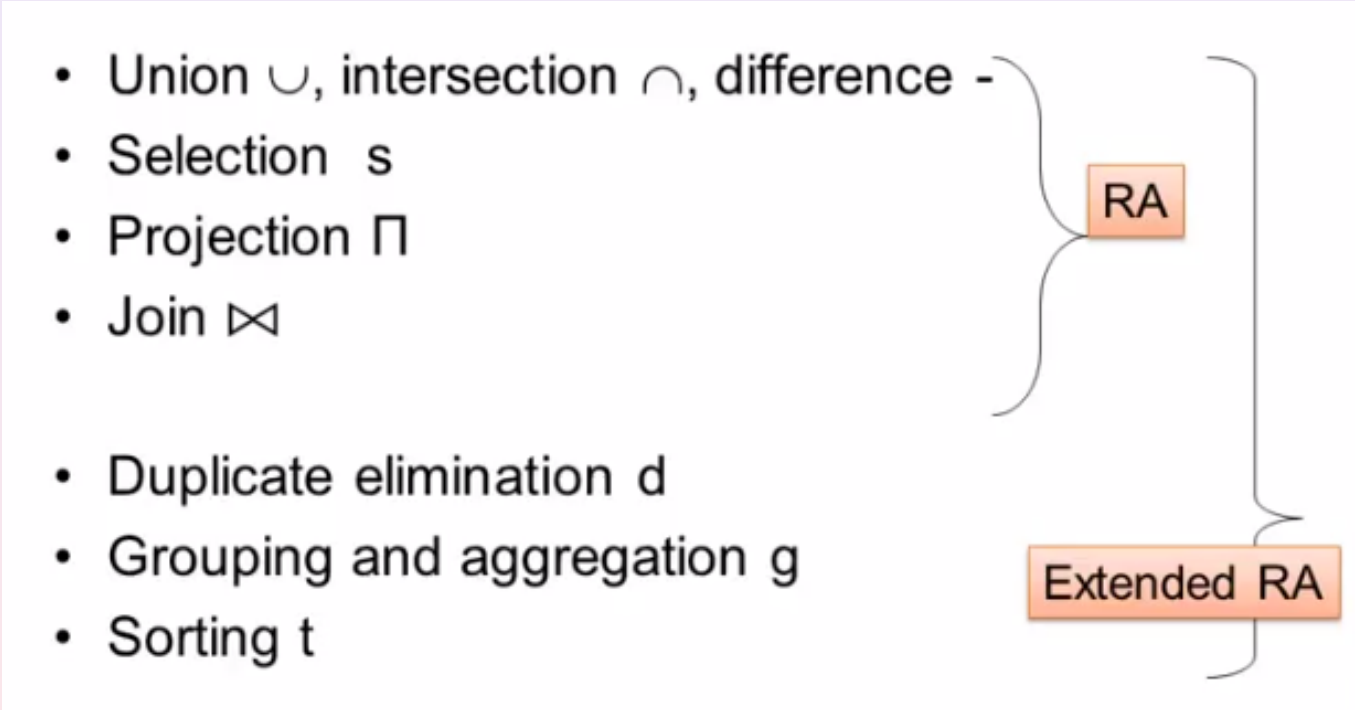
- El motor de una base de datos relacional es el software que administra la organización de la información
- Los programas se conectan con el motor para solicitarle la gestión de los datos (crear o borrar una tabla, insertar, modificar o borrar registros, etc.)
- El motor separa a las aplicaciones de la organización de los datos (independencia de datos)

SQL

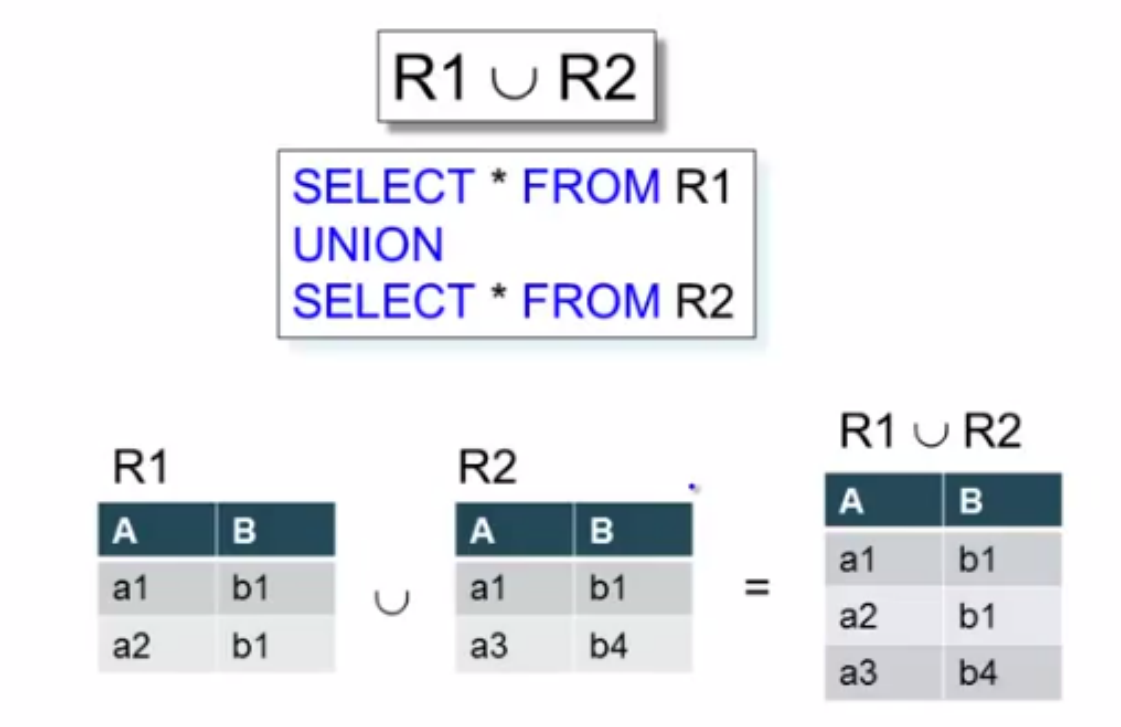
- El lenguaje estandar que emplean los motores de bases de datos relacionales es SQL (*Structured Query Language*)
- SQL se traduce directamente a expresiones de *álgebra relacional*
- SQL es un *lenguaje declarativo*
  - El motor de la base de datos traduce la consulta SQL a expresiones equivalentes y elige la mejor (Plan de Ejecución)
  - Refuerza la independecia de datos por que no hace explícito el modo en que los datos deben accederse

Álgebra Relacional

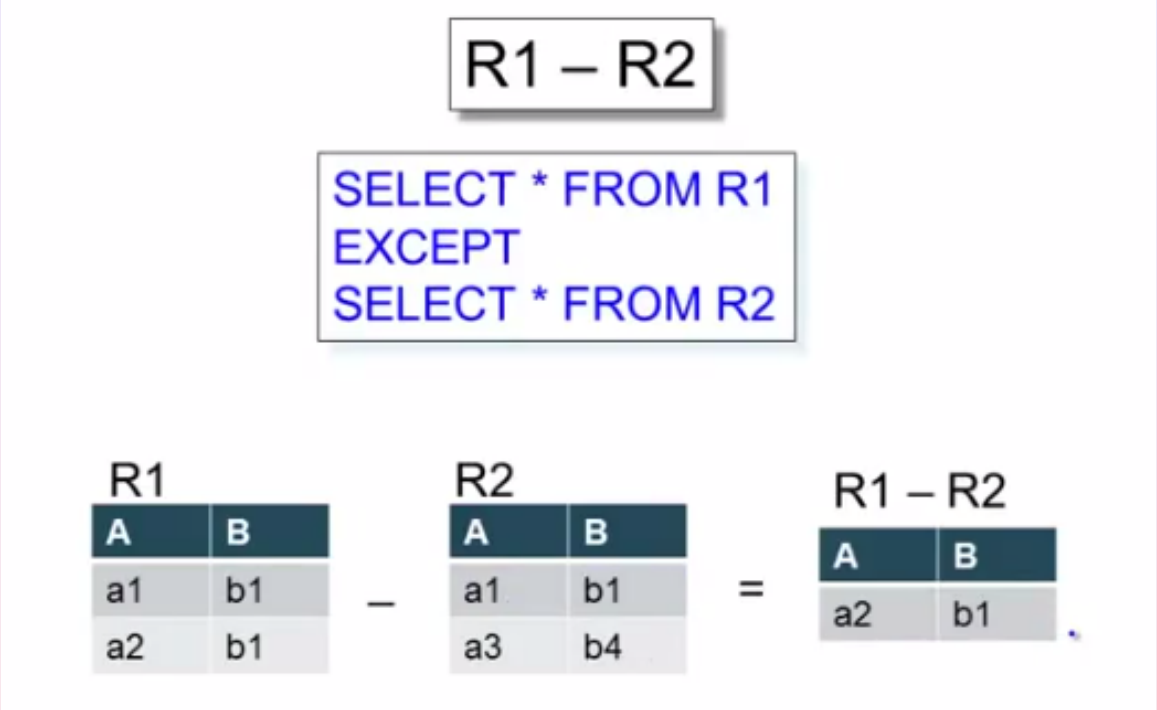
El *álgebra relacional* es el formalismo teórico del modelo relacional. Es una extensión de la teoría de conjuntos.



Unión


R1 ∪ R2
SELECT * FROM R1
UNION
SELECT * FROM R2
R1
A B
a1 b1
a2 b1
∪
R2
A B
a1 b1
a3 b4
=
R1 ∪ R2
A B
a1 b1
a2 b1
a3 b4

Diferencia


R1 – R2
SELECT * FROM R1
EXCEPT
SELECT * FROM R2
R1
A B
a1 b1
a2 b1
–
R2
A B
a1 b1
a3 b4
=
R1 – R2
A B
a2 b1

Intersección

- Derived operator using minus

$$R1 \cap R2 = R1 - (R1 - R2)$$

- Derived using join (will explain later)

$$R1 \cap R2 = R1 \bowtie R2$$

Selección

- Returns all tuples which satisfy a condition

$$\sigma_c(R)$$

- Examples
  - sSalary > 40000 (Employee)
  - sname = “Smith” (Employee)
- The condition c can be =, <, ≤, >, ≥, <>

Selección

Employee

SSN	Name	Salary
1234545	John	20000
5423341	Smith	60000
4352342	Fred	50000

$\sigma_{\text{Salary} > 40000}(\text{Employee})$

SSN	Name	Salary
5423341	Smith	60000
4352342	Fred	50000

Proyección

Projection

- Eliminates columns

$$\Pi_{A1,\ldots,An}(R)$$

- Example: project social-security number and names:
  - P SSN, Name (Employee)
  - Answer(SSN, Name)

Proyección

Employee

SSN	Name	Salary
1234545	John	20000
5423341	John	60000
4352342	John	20000

$\Pi_{\text{Name,Salary}}$ (Employee)

Name	Salary
John	20000
John	60000
John	20000

Name	Salary
John	20000
John	60000

Set semantics

Producto Cartesiano

- Each tuple in R1 with each tuple in R2

$$R1 \times R2$$

Employee

Name	SSN
John	999999999
Tony	777777777

Dependent

EmpSSN	DepName
999999999	Emily
777777777	Joe

Employee □ Dependent

Name	SSN	EmpSSN	DepName
John	999999999	999999999	Emily
John	999999999	777777777	Joe
Tony	777777777	999999999	Emily
Tony	777777777	777777777	Joe

Join

$$R1 \bowtie_{A=B} R2 \quad = \quad \sigma_{A=B} (R1 \times R2)$$

```
SELECT *
FROM R1, R2
WHERE R1.A = R2.B
```

```
SELECT *
FROM R1 JOIN R2
ON R1.A = R2.B
```

- Find all hospitals within 5 miles of a school

$$\Pi_{name}(Hospitals \bowtie_{distance(location,location) < 5} Schools)$$

```
SELECT DISTINCT h.name
FROM Hospitals h, Schools s
WHERE distance(h.location, s.location) < 5
```

Outer Join

- Outer join
  - Include tuples with no matches in the output
  - Use NULL values for missing attributes

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu
33	98120	lung

AnnonJob J

job	age	zip
lawyer	54	98125
cashier	20	98120

$P \ltimes V$

age	zip	disease	job
54	98125	heart	lawyer
20	98120	flu	cashier
33	98120	lung	null

Diseño del Esquema de una Base de Datos

- El objetivo es evitar la redundancia y las anomalías
- El proceso de normalización permite depurar las relaciones