



Prepared by group 1

Modelo Relacional de las Bases de Datos

19 Marzo 2025

Aquino Lozada Gabriela
Soriano Barrera María Elena



Introducción

- Antes de 1970, las bases de datos se organizaban de manera jerárquica o en redes
- E. F. Codd, un investigador de IBM, propuso un nuevo enfoque: el modelo relacional.
- Este modelo revolucionó la forma en que almacenamos y manipulamos la información

Information Retrieval

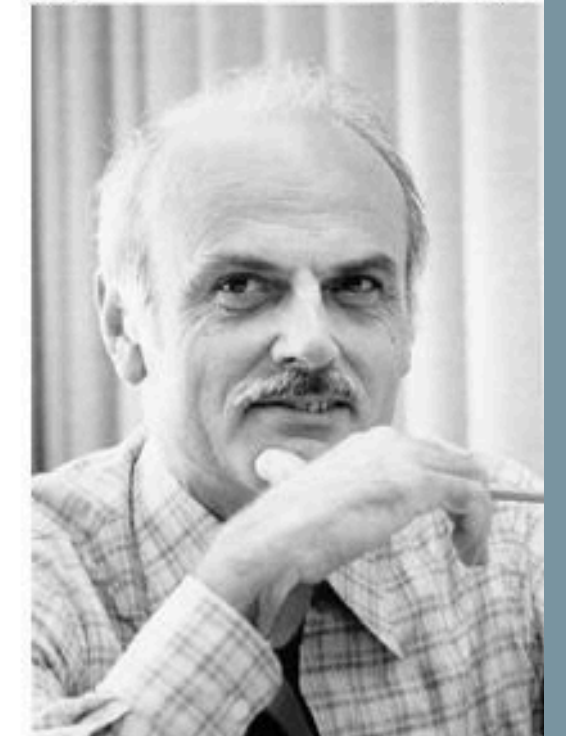
A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network

The relational view (or model Section 1 appears to be superior in graph or network model [3, 4] pre



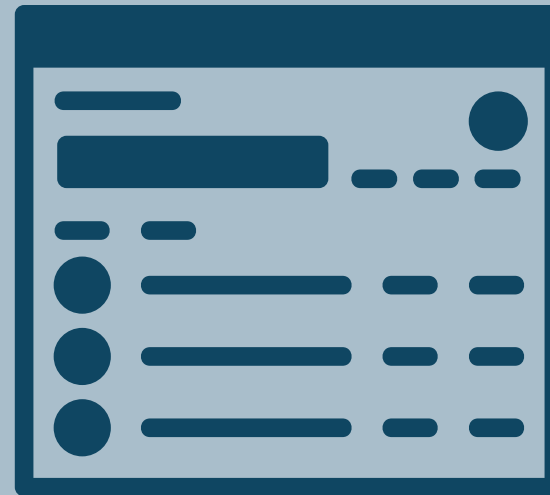
Problemas de los Modelos

Anteriores



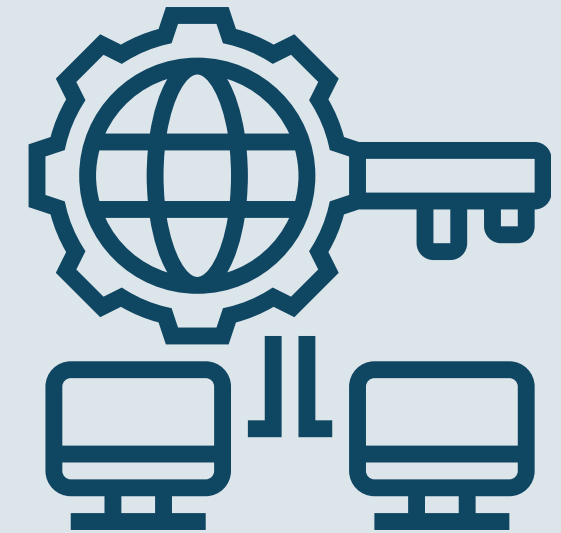
Dependencia del Orden

- Tener los datos almacenados en un orden específico.
- Al cambiar el orden, muchas aplicaciones dejaban de funcionar.



Dependencia de Índice

- Se usaban índices fijos para localizar datos, lo que dificultaba modificar la estructura sin afectar las aplicaciones.



Dependencia de Rutas de Acceso

- Los datos solo podían accederse siguiendo caminos predefinidos, lo que limitaba la flexibilidad para consultas nuevas.

Ejemplo:



Una empresa almacena información sobre empleados y departamentos en una estructura jerárquica:

- cada departamento tiene varios empleados asignados.
- si queremos encontrar a que departamento pertenece un empleado, debemos recorrer toda la estructura.
- si la empresa reorganiza los departamentos, tendríamos que reescribir muchas consultas.



Modelo Relacional de Codd



Almacenar la información en
tablas (relaciones), donde
cada fila representara una
entidad y cada columna un
atributo



Ejemplo:

● ● ● ● ● implementamos tablas

Tabla EMPLEADOS

ID	Nombre	Departamento_ID
1	Juan	10
2	María	20
3	Pedro	10

Tabla DEPARTAMENTOS

ID	Nombre
10	Ventas
20	Contabilidad

● ● ● ● ●

Ejemplo:

● ● ● ● ● implementamos tablas

Tabla EMPLEADOS

ID	Nombre	Departamento_ID
1	Juan	10
2	María	20
3	Pedro	10

Tabla DEPARTAMENTOS

ID	Nombre
10	Ventas
20	Contabilidad

● ● ● ● ●

Ejemplo:

● ● ● ● ● implementamos tablas

Tabla DEPARTAMENTOS

```
CREATE TABLE departamentos (  
    id SERIAL PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL  
);
```

insertamos datos en tabla

```
INSERT INTO departamentos (id, nombre) VALUES  
(10, 'Ventas'),  
(20, 'Contabilidad');
```

● ● ● ● ●

Ejemplo:

● ● ● ● ● implementamos tablas

Tabla EMPLEADOS

```
CREATE TABLE empleados (  
  id SERIAL PRIMARY KEY,  
  nombre VARCHAR(50) NOT NULL,  
  departamento_id INT NOT NULL,  
  FOREIGN KEY (departamento_id) REFERENCES departamentos(id)  
);
```

insertamos datos en tabla

```
INSERT INTO empleados (id, nombre, departamento_id) VALUES  
(1, 'Juan', 10),  
(2, 'María', 20),  
(3, 'Pedro', 10);
```

● ● ● ● ●



Ejemplo:

● ● ● ● ● implementamos tablas

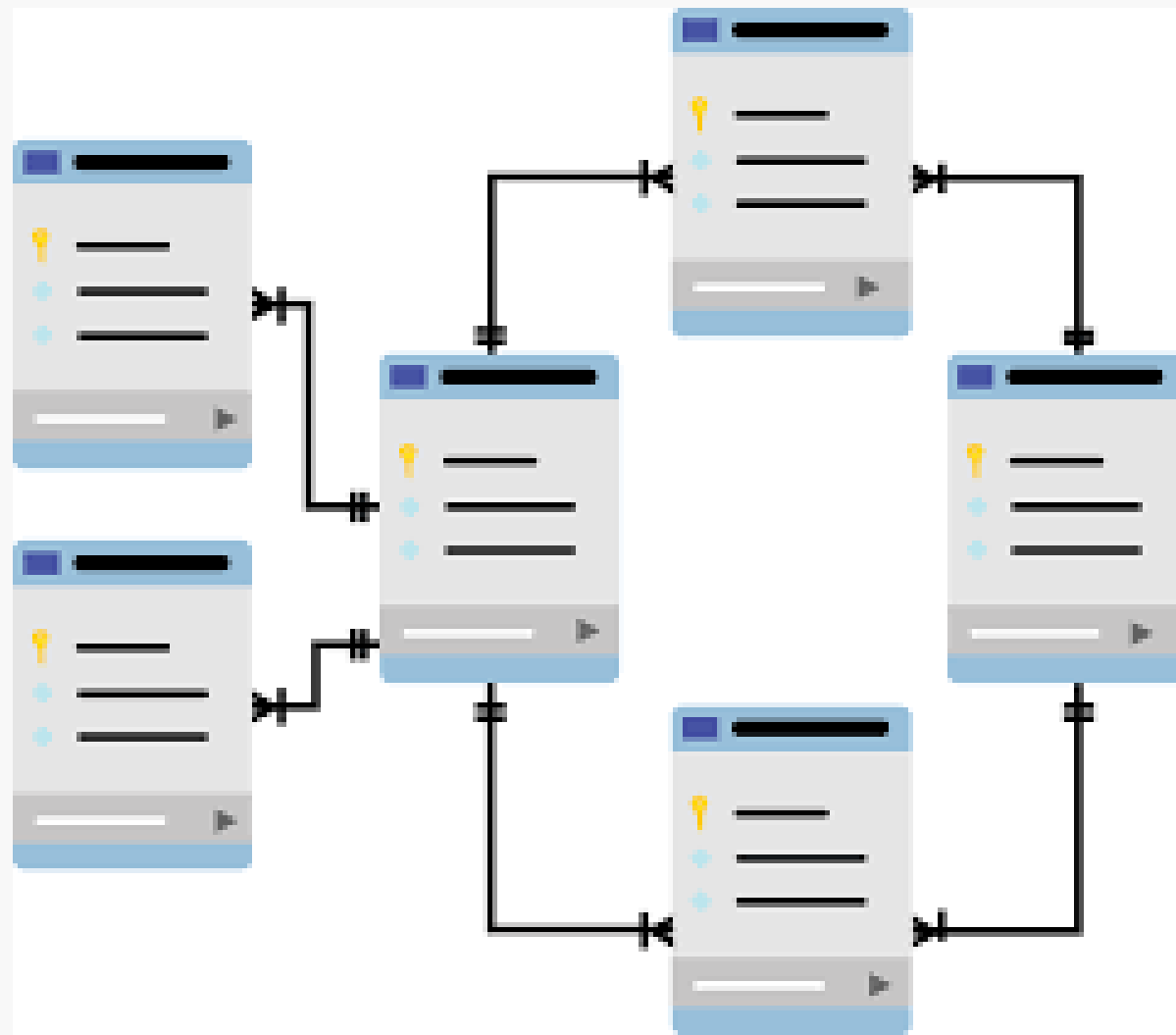
Consulta SQL

```
SELECT EMPLEADOS.Nombre, DEPARTAMENTOS.Nombre
FROM EMPLEADOS
JOIN DEPARTAMENTOS ON EMPLEADOS.Departamento_ID = DEPARTAMENTOS.ID
WHERE EMPLEADOS.Nombre = 'María';
```

Lo que devuelve la consulta

	nombre character varying (50) 	nombre character varying (50) 
1	María	Contabilidad

● ● ● ● ●



PRINCIPIOS DEL MODELO RELACIONAL

INDEPENDENCIA DE LOS DATOS:

- Las aplicaciones pueden acceder a los datos sin preocuparse por cómo están almacenados físicamente.

ELIMINACIÓN DE REDUNDANCIA:

- Se evita repetir datos innecesarios mediante formas normales.

OPERACIONES RELACIONALES:

- Se pueden hacer operaciones matemáticas sobre los datos, como:
 - Selección: Obtener solo ciertos registros.
 - Proyección: Extraer ciertas columnas.
 - Unión (JOIN): Combinar tablas relacionadas.

Redundancia

Cod definía la redundancia en dos tipos:

Redundancia Fuerte:

- La misma información se repite innecesariamente.

Ejemplo:

Tablas de EMPLEADOS

ID_Empleado	Nombre	Departamento_ID	Nombre_Departamento
1	Juan Perez	10	Ventas
2	Ana Gomez	10	Ventas
3	Luis Ramirez	20	Contabilidad

Tablas de EMPLEADOS

ID_Empleado	Nombre	Departamento_ID
1	Juan Perez	10
2	Ana Gomez	10
3	Luis Ramirez	20

Tablas de DEPARTAMENTOS

ID_Departamento	Nombre
10	Ventas
20	Contabilidad

Redundancia

Redundancia Débil:

- No se repite información, pero se puede deducir

Ejemplo:

Tablas de EMPLEADOS_PROYECTOS

ID_Empleado	Proyecto_ID
1	A1
2	A2
3	B1

Tablas de PROYECTOS_DEPARTAMENTOS

Proyecto_ID	Departamento_ID
A1	10
A2	10
B1	20

Tablas de DEPARTAMENTOS

ID_Departamento	Nombre
1	Ventas
2	Ventas
3	Contabilidad

Código SQL:

```
SELECT EMPLEADOS_PROYECTOS.ID_Empleado, DEPARTAMENTOS.Nombre
FROM EMPLEADOS_PROYECTOS
JOIN PROYECTOS_DEPARTAMENTOS ON EMPLEADOS_PROYECTOS.Proyecto_ID = PROYECTOS_DEPARTAMENTOS.Proyecto_ID
JOIN DEPARTAMENTOS ON PROYECTOS_DEPARTAMENTOS.Departamento_ID = DEPARTAMENTOS.ID_Departamento;
```

Restricciones de Integridad en el Modelo Relacional

ID_Empleado (PK)	Nombre	Edad	Departamento_ID (FK)
1	Juan Pérez	30	10
2	Ana Gómez	25	10
3	Luis Ramírez	40	20

ID_Departamento (PK)	Nombre
10	Ventas
20	Contabilidad
30	Recursos Humanos

Integridad de Entidad

- Evita filas duplicadas asegurando que cada registro tenga un identificador único (PRIMARY KEY)
- Ejemplo: La columna ID_Empleado en la tabla EMPLEADOS.

Integridad Referencial

- Evita referencias inválidas asegurando que una clave foránea (FOREIGN KEY) siempre apunte a un valor existente.
- Ejemplo: Departamento_ID en la tabla EMPLEADOS debe existir en la tabla DEPARTAMENTOS.

Integridad de Dominio

- Garantiza que los valores sean válidos según su tipo de dato y restricciones (CHECK, NOT NULL).
- Ejemplo: No permitir edades negativas en una columna Edad.



```
-- Crear la tabla DEPARTAMENTOS
CREATE TABLE DEPARTAMENTOS (
    ID_Departamento INT PRIMARY KEY,
    Nombre VARCHAR(50) NOT NULL
);

-- Crear la tabla EMPLEADOS con PRIMARY KEY y FOREIGN KEY
CREATE TABLE EMPLEADOS (
    ID_Empleado INT PRIMARY KEY,
    Nombre VARCHAR(50) NOT NULL,
    Edad INT CHECK (Edad >= 18), -- Restricción de Dominio
    Departamento_ID INT,
    FOREIGN KEY (Departamento_ID) REFERENCES DEPARTAMENTOS(ID_Departamento) -- Restricción Referencial
);
```





Referencias

- E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, June 1970, doi: 10.1145/362384.362685.
- Colaboradores de los proyectos Wikimedia. "Modelo relacional - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. Accedido el 19 de marzo de 2025. [En línea]. Disponible: https://es.wikipedia.org/wiki/Modelo_relacional
- E. F. Codd, "Relational Completeness of Data Base Sublanguages," en *Database Systems*, R. Rustin, Ed. Prentice-Hall, 1972, pp. 65–98.
- "Jorge Sánchez. Manual de Gestión de Bases de Datos. Modelo Relacional". Jorge Sanchez, profesor de informática. Manuales, ejercicios y documentos sobre cursos de informática. Accedido el 19 de marzo de 2025. [En línea]. Disponible: <https://jorgesanchez.net/manuales/gbd/modelo-relacional.html#:~:text=Según%20Codd,%20los%20datos%20se,almacenamiento%20real%20en%20la%20computadora.>





Gracias :)

