# 관계형 데이터베이스의 역사

| 태그 | Database |
| --- | --- |



## 관계란?

- 복수의 대상이 서로 관련하여 이루는 특성
- 관계형 모델에서는 동일한 구조로 이루어진 튜플의 집합

## 관계형 모델

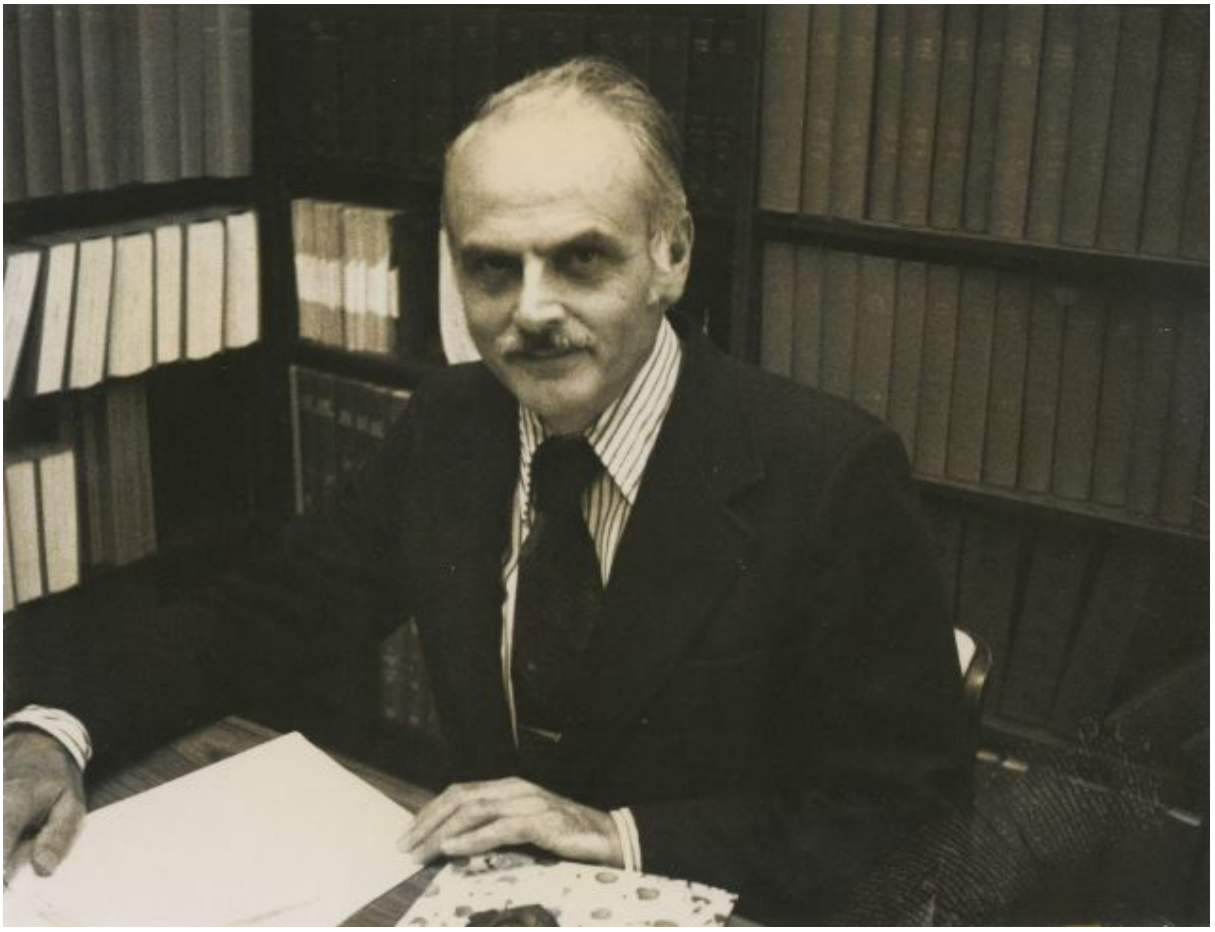
- 실제 세계의 데이터를 **관계라는 개념을 사용**해 표현한 데이터 모델

## 관계형 모델의 역사

### 왜 관계형 모델이 나왔을까?

- 1960년 대에 방대한 양의 데이터를 다루기 시작함.
- 초기의 데이터베이스에는 자기 테이프에 있는 데이터들을 물리적으로 연결하거나 중첩함.
- 그래서 **엄청나게 복잡하고 큰 비용을 야기**하고 이를 다루기 위해서는 **컴퓨터 전문가**와 복잡한 과정이 필요.

## E.F. Codd, “A Relational Model for Large Shared Data Banks”



**“그의 놀라운 비전과 지적 천재성은 오늘날 기술 세계를 형성한 완전히 새로운 혁신의 영역을 열었습니다.”**

- IBM 연구소에 근무하는 수학자 출신 **E.F Codd**의 논문
- 1981년에 튜링상 수상
- 기존과는 다르게 **전문 지식 없이도** 데이터에 액세스 할 수 있음.
- 값 사이의 관계를 **튜플**과 **속성**을 이용하여 나타냄.

| *supply* | (*supplier* | *part* | *project* | *quantity*) |
|---|---|---|---|---|
| | 1 | 2 | 5 | 17 |
| | 1 | 3 | 5 | 23 |
| | 2 | 3 | 7 | 9 |
| | 2 | 7 | 5 | 4 |
| | 4 | 1 | 1 | 12 |

    - 데이터 간의 관계는 추가적인 정보가 아닌 해당 관계가 가지고 있는 속성의 값을 이용하여 연결
- 데이터베이스와 이를 질의하는 방법이 간결하고 수학적 모델로 설명됨.
- 이 논문 발표 후 추가 연구와 구현이 활발하게 진행됨.

### 관계형 모델의 장점

1. **추가적인 표현 없이도** 관계만으로 자연스러운 설명이 가능
2. **중복성, 일관성, 새로운 파생 관계**를 처리하기에 유리
3. 고수준의 데이터 언어를 제공
    a. 이후에 **SQL**이 만들어지고 채택됨.
4. 데이터베이스에 대한 물리적인 지식이 존재하지 않아도 데이터에 액세스 할 수 있음.

## 이후

- 관계형 모델에 기반하여 관계형 데이터베이스가 만들어지기 시작함.

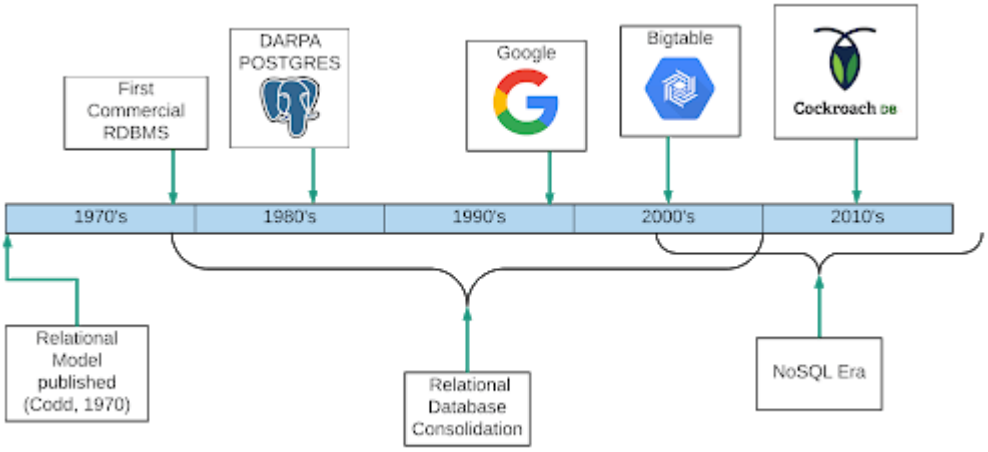
### 구현

- 저장 공간을 늘리고 리소스를 적게 만들도록 하는 최적화 프로그램 개발
- 데이터 쿼리 계획을 저장할 수 있는 컴파일러 개발
- 고수준의 쿼리를 효율적인 실행 계획으로 자동 변환하는 SQL과 시스템 개발
- Oracle, IBM이 선두주자로 여러가지 상용 관계형 데이터베이스를 만듬.

# 관계형 데이터베이스

- 각각 고유한 이름을 가진 **테이블(릴레이션)의 모임**으로 구성됨.
  - **튜플**과 **속성**으로 이루어짐.
  - 튜플은 속성의 집합
  - 속성은 값의 집합(**도메인**)
  - 속성 사이의 관계를 속성이 가진 값으로 표현
  - 릴레이션 간의 관계를 릴레이션이 가진 값으로 표현
- 정적인 스키마, 제약 조건, 키를 통한 관계 설정과 같은 특징들이 존재
- **SQL**을 통한 손쉬운 정보 획득 가능
- 다양한 관계 대수 연산 지원
  - selection, projecton, rename, join, union, intersection,cartesian product, set-difference, assignment, equivalent

# 관계형 데이터베이스의 역사




## 첫 상용 RDBMS

- codd의 논문을 기반하여 프로젝트들이 시작됨.
  - 효율적인 RDBMS를 만들기 시작함.
- Oracle, IBM 등이 상용 RDBMS를 만들어서 출시
- Oracle이 1979년에 첫 출시

- 이후에 DB2, SAP, Informix가 이어서 출시됨.

## RDBMS의 부흥

1. **인덱스**를 통한 효율적 쿼리 지원
2. **조인**을 통한 복잡한 조회 연산 지원
3. **트랜잭션**을 통한 일관성 보장
4. **SQL**을 통한 쉬운 질의 지원
5. **DBMS의 구현을 알지 못해도 사용**할 수 있음.

위와 같은 이점들로 DBMS에서 RDBMS는 **시장의 선두**를 차지함.

## 여러가지 RDBMS의 출시

- MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database, SQLite 등

## RDBMS의 한계

- 단일 서버와 RDBMS로는 **너무 많은 데이터는 처리할 수 없음**.
  - **수직 스케일 업에는 한계**가 존재
- 단일 서버와 RDBMS에는 **장애 대응이 어려움**.
- 요구사항이 너무 고도화되고 빠르게 변하기에 **유연성이 필요**함.
  - 정적이고 고정된 스키마를 가지는 RDBMS에는 한계가 존재

## NoSQL

- **유연한 스키마**
- **분산 처리**를 통한 **대용량 데이터 처리**에 용이
- **여러 노드**를 통한 **장애 대응**
- **여러 노드**를 통한 **수평 스케일 업**
- BigTable, HDFS, Cassandra, MongoDB 등

## Distributed SQL

- **여러 서버에 데이터를 복제하는 단일 논리적 DBMS**
- 지리적인 특성을 넘어서는광범위한 일관성 보장

- 행과 열로 구성된 테이블, 확장, 일관성, 복제, 샤딩, 분산 트랜잭션, SQL 지원
- Google Spanner, Amazon Aurora, CockroachDB 등

# 참조문헌


- https://www.ibm.com/history/relational-database
- https://sis.binus.ac.id/2023/07/05/the-relational-model-history-and-its-language/
- https://www.cockroachlabs.com/blog/history-of-databases-distributed-sql/
- https://en.wikipedia.org/wiki/Relational_model
- Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377-387. https://doi.org/10.1145/362384.362685
- https://www.cockroachlabs.com/blog/what-is-distributed-sql/
- https://en.wikipedia.org/wiki/Distributed_SQL
- https://www.cockroachlabs.com/blog/history-of-databases-distributed-sql/