DATA ENGINEERING COURSE

데이터 저장의 진화
RDBMS부터 MongoDB까지

파일 시스템에서 시작된 데이터의 여정과
빅데이터 시대의 새로운 표준, NoSQL 완전 정복

강의 목차 (Agenda)

Part 1. 데이터 저장의 역사

- 태초의 데이터: 파일 시스템
- 파일 시스템의 치명적 문제점
- 1세대: 계층형 데이터베이스 (H-DB)
- 2세대: 관계형 데이터베이스 (RDBMS)
- RDBMS의 구조와 핵심 기능

Part 2. 시대의 변화와 한계

- Web 2.0과 빅데이터의 도래
- RDBMS의 확장성 한계 (Scale-Up)
- 새로운 대안: Scale-Out

Part 3. NoSQL과 MongoDB

- NoSQL의 탄생 배경과 철학
- NoSQL의 다양한 종류
- MongoDB의 핵심 구조 (Document)
- JSON Document의 강력함
- SQL vs MongoDB 문법 비교
- 핵심 요약 (Wrap-up)

PART 01

데이터 저장의 역사

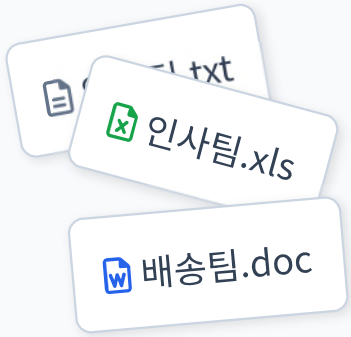
파일에서 시작하여 체계적인 데이터베이스로 발전하는
필연적인 과정에 대하여

1. 태초의 데이터 저장: 파일 시스템

"개인 서랍 속의 무질서"

데이터베이스가 없던 시절, 우리는 운영체제의 파일(txt, csv, xls)에 데이터를 직접 기록했습니다.

- 프로그램이 직접 파일을 열고 닫는 방식.
- 각 부서나 개인이 각자의 파일을 관리.



⚠ 관리가 불가능한 상태

파일 시스템의 치명적 문제점 (Data Silo)

데이터 중복 (Redundancy)

같은 정보가 여러 파일에 흩어져 저장되어 저장 공간이 낭비됩니다.

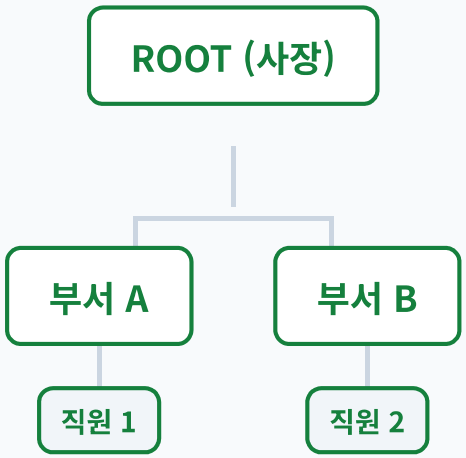
데이터 불일치 (Inconsistency)

한쪽 파일만 수정되면 정보가 서로 달라져 신뢰성이 붕괴됩니다.

데이터 종속성 (Dependency)

파일 형식을 조금만 바꿔도 관련된 모든 프로그램을 수정해야 합니다.

2. 1세대: 계층형 DB (Hierarchical)



트리(Tree) 구조의 등장 (1960s)

NASA의 아폴로 프로젝트 등 거대 자재 관리를 위해 탄생했습니다.

- **구조:** 부모(Parent)와 자식(Child)이 1:N 관계로 연결된 족보 형태.
- **장점:** 경로가 명확하여 검색 속도가 매우 빠름.
- **단점:** **유연성 부족.** 한 자식이 두 부모를 가질 수 없어 복잡한 관계 표현이 불가능.

3. 2세대: 관계형 DB의 혁명 (RDBMS)

"모든 데이터를 표(Table)로 만들자"

1970년, IBM의 E.F. Codd 박사가 제안한 수학적 모델 기반의 데이터베이스입니다.

- **핵심 철학:** 데이터를 잘게 쪼개서(정규화) 중복을 없애자.
- **SQL의 탄생:** 프로그래머가 아니어도 영어 문장처럼 데이터를 다룰 수 있는 표준 언어.
- **표준화:** Oracle, MySQL, PostgreSQL 등 현대 DB의 기반.

Relational Model

Table + Row + Column

RDBMS의 구조: 행과 열

Table: Users

ID (PK)	Name	Dept
101	김철수	개발팀
102	이영희	디자인
103	박민수	영업팀

● Row (행) ● Column (열)

정규화된 테이블 (Normalized Table)

엑셀 스프레드시트와 유사한 2차원 구조입니다.

- **Table (표):** 데이터의 집합 (예: 사원 테이블).
- **Row (행, Record):** 가로줄, 데이터 한 건.
- **Column (열, Field):** 세로줄, 데이터의 속성.
- **Primary Key (기본키):** 각 행을 구분하는 고유한 식별자.

왜 RDBMS가 40년을 지배했을까?

1. 무결성 (Integrity)의 수호자

'스키마(Schema)'라는 엄격한 규칙을 통해 잘못된 데이터 입력을 원천 차단합니다.

2. ACID 트랜잭션 보장

은행 이체처럼 "모두 성공하거나, 아무일도 없었던 것처럼 되돌리는(All or Nothing)" 작업을 완벽하게 보장합니다.

데이터 무결성

중복 및 오류 데이터 방지

트랜잭션 안전성

금융 거래 등에서 필수적

PART 02

시대의 변화와 한계

웹 2.0, 빅데이터 시대의 도래와
기존 RDBMS의 한계 직면

시대의 변화: Web 2.0 & Big Data

데이터 폭발의 시작 (2000년대 후반~)

스마트폰, SNS, IoT의 등장으로 데이터 생성 주체가 '기업'에서 '개인'으로 이동하며 데이터가 폭증했습니다.

과거: 결제 내역, 직원 정보 등 **정형 데이터** 위주.

현재: 댓글, 좋아요, 위치 정보, 사진, 영상 등 **비정형 데이터**가 주류.

빅데이터의 3대 요소 (3Vs)

기존 RDBMS로는 감당하기 힘든 새로운 데이터의 특징입니다.

Volume (양)

GB를 넘어 TB, PB 단위의 거대 데이터. 단일 서버의 저장 한계를 초과합니다.

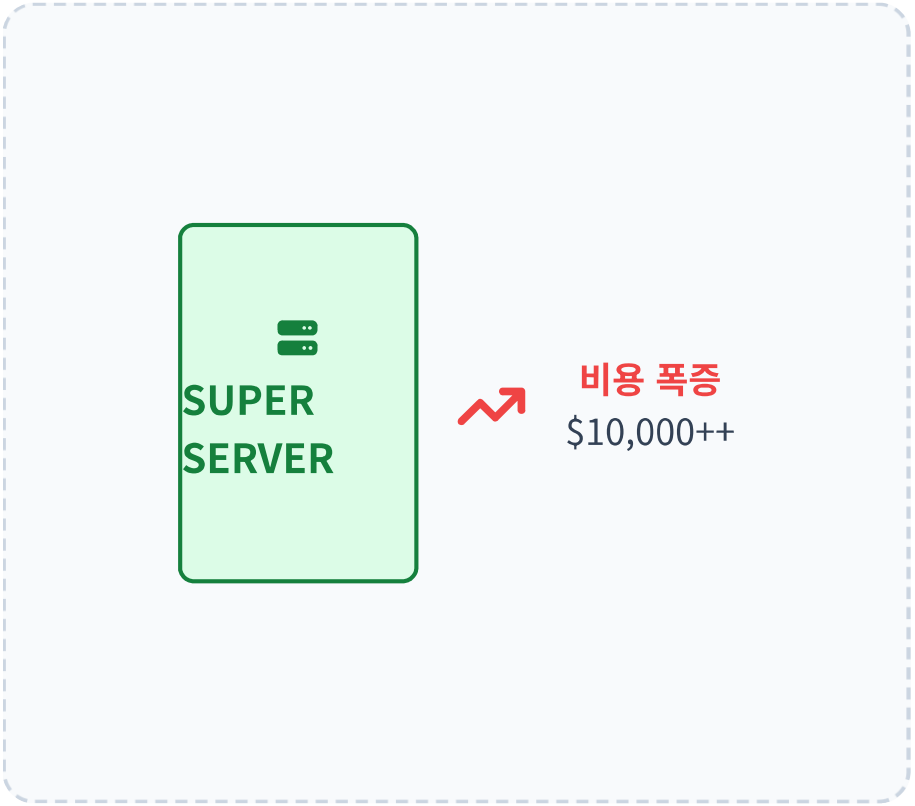
Velocity (속도)

초당 수만 건씩 쏟아지는 실시간 스트리밍 데이터 처리 능력이 필요합니다.

Variety (다양성)

고정된 표(Table)에 넣을 수 없는 사진, 소리, JSON 등 비정형 데이터가 증가합니다.

RDBMS의 한계: Scale-Up (수직 확장)



"더 비싼 컴퓨터를 사자"

RDBMS는 태생적으로 하나의 고성능 서버에서 실행되도록 설계되었습니다.

- **방식:** 기존 서버에 CPU, RAM, SSD 등 더 좋은 부품을 장착합니다.
- **문제점:**
  - 장비 가격이 기하급수적으로 비싸집니다.
  - 언젠가는 더 이상 업그레이드할 수 없는 물리적 한계에 부딪힙니다.

새로운 대안: Scale-Out (수평 확장)

"싼 컴퓨터를 여러 대 연결하자"

구글, 아마존 같은 기업들이 빅데이터를 처리하기 위해 선택한 방식입니다.

- **방식:** 저렴한 일반 서버를 네트워크로 연결하여 하나의 거대한 시스템처럼 사용합니다.
- **장점:**
  - 비용 효율적이며 클라우드 환경에 적합합니다.
  - 이론상 **무한 확장**이 가능합니다.
- **핵심:** 이 방식을 위해 태어난 것이 바로 **NoSQL**입니다.

Cluster (분산 처리)

PART 03

NoSQL과 MongoDB

빅데이터 시대의 새로운 표준,
NoSQL의 개념과 MongoDB 심층 탐구

3세대: NoSQL의 등장

"SQL만으로는 충분하지 않다"

RDBMS의 경직된 스키마와 확장성 한계를 극복하기 위해 탄생했습니다.

- **Schema-less:** 데이터 구조를 미리 정의할 필요가 없습니다.
- **Scale-Out 지향:** 처음부터 분산 환경을 염두에 두고 설계되었습니다.
- **다양한 형태:** 해결하려는 문제에 따라 여러 가지 데이터 모델이 존재합니다.

NoSQL의 종류 (1): Key-Value & Column

Key-Value Store

가장 단순하고 빠른 구조. 헬스장 락커룸처럼 열쇠(Key)로 값(Value)을 바로 찾습니다.

예: Redis, AWS DynamoDB

Column-Family Store

데이터를 행(Row)이 아닌 열(Column) 단위로 묶어서 저장하여 대량 데이터 분석에 유리합니다.

예: Cassandra, HBase

NoSQL의 종류 (2): Document (MongoDB)

가장 대중적인 NoSQL

Key-Value의 유연성과 RDBMS의 검색 기능을 합친 가장 현실적인 대안입니다.

- **데이터 모델:** **JSON 문서(Document)**를 통째로 저장합니다.
- **특징:**
  - 직관적인 데이터 구조.
  - 강력한 인덱싱 및 검색 쿼리 지원.
- **시장 점유율:** 전 세계 NoSQL DB 중 압도적 1위.

```
{
  "name": "김코딩",
  "skills": ["Python", "DB"],
  "active": true
}
```

MongoDB의 구조와 JSON의 장점

구조 비교 (RDBMS vs MongoDB)

RDBMS	MongoDB
Table (표)	**Collection** (꾸러미)
Row (행)	**Document** (문서)
Column (열)	Field (필드)

JSON Document의 강력함

프로그래밍 언어의 객체(Object, Dictionary)와 DB 저장 형식이 완벽하게 일치합니다.

✓ Impedance Mismatch 해결

데이터를 억지로 쪼개고 합치는(Join) 번거로운 과정 없이 개발 속도가 비약적으로 상승합니다.

SQL vs MongoDB 문법 & 핵심 요약

문법(CRUD) 비교

```
[생성 - Create]
SQL: INSERT INTO users (name) VALUES ('Kim');
MQL: db.users.insertOne({ name: 'Kim' });

[조회 - Read]
SQL: SELECT * FROM users WHERE age > 20;
MQL: db.users.find({ age: { $gt: 20 } });
```

핵심 요약 (Key Takeaways)

- **역사의 흐름:** 파일(불편) → RDBMS(정확) → NoSQL(유연/거대)로 진화.
- **MongoDB:** JSON 기반의 유연성과 Scale-Out 능력으로 현대 개발의 표준.
- **적재적소:** 금융은 RDBMS, 빅데이터/로그는 NoSQL. 정답은 상황에 따라 다릅니다 (Polyglot).

수고하셨습니다! 다음 시간은 Python 실습입니다.

```
```