

## 데이터베이스

**데이터베이스**(영어: database, **DB**)는 여러 사람이 공유하여 사용할 목적으로 체계화해 통합, 관리하는 데이터의 집합이다.<sup>[1]</sup> 작성된 목록으로써 여러 응용 시스템들의 통합된 정보들을 저장하여 운영할 수 있는 공용 데이터들의 묶음이다. 데이터베이스에 속해있는 모델은 다양하다.

## 역사

1950년대에 데이터베이스라는 용어가 미국에서 처음 사용되었으며, 본래는 군비의 집중적·효율적 관리를 위해 컴퓨터를 활용한 도서관 개념을 개발하면서 이를 '데이터의 기지'라는 뜻의 데이터베이스로 일컬었다. 이후 1965년 시스템 디벨로프사가 2차로 개최한 '컴퓨터 중심 데이터베이스 시스템'이라는 심포지엄에서 처음 사용하였다.<sup>[2]</sup>

프로세서, 컴퓨터 메모리, 컴퓨터 스토리지, 컴퓨터 네트워크 분야에서 기술이 진전됨에 따라, 등급 순으로 데이터베이스 및 각 DBMS의 크기, 기능, 성능이 상승해왔다. 데이터베이스 기술의 발전은 데이터 모델이나 구조에 따라 세 개의 시대로 나뉜다(내비게이션 데이터베이스,<sup>[3]</sup> SQL/관계형 데이터베이스, 관계형 이후 데이터베이스).

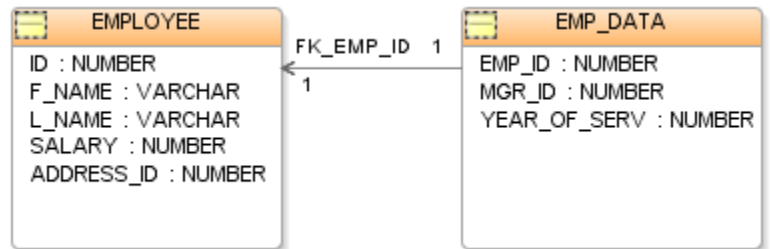
주가 되는 2개의 내비게이션 데이터 모델은 IBM의 IMS 시스템의 전형적 본보기가 되는 계층형 모델, 그리고 IDMS와 같은 수많은 제품들에 구현된 CODASYL 모델 (네트워크 모델)이다.

1970년에 에드거 F. 커드가 처음 제안한 관계형 모델은 응용 프로그램들이 뒷따르는 링크가 아닌 내용을 기준으로 데이터를 검색해야 한다고 주장하면서 이러한 전통에서 출발했다. 관계형 모델은 금전출납부 스타일의 표들의 모임을 이용하며, 각각은 다른 타입의 엔티티를 위해 사용된다. 1980년대 들어서야 컴퓨팅 하드웨어가 비로소 관계형 시스템(DBMS + 애플리케이션)의 폭넓은 배치를 가능케 할 만큼 강력해졌다. 그러나 1990년대 초 들어서 모든 대형 데이터 처리 애플리케이션을 관계형 시스템들이 지배하게 되었으며 2015년 기준으로 여전히 상황은 동일하다. : IBM DB2, 오라클, MySQL, 마이크로소프트 SQL 서버는 최상위 DBMS이다.<sup>[4]</sup> 지배적인 데이터베이스 언어, 곧 관계형 모델을 위한 표준화된 SQL은 다른 데이터 모델의 데이터베이스 언어들에 영향을 미쳤다.

객체 지향 데이터베이스는 객체 지향 임피던스 불일치의 불편함을 극복하고자 1980년대에 개발되었으며, 이로 인해 "관계형 이후"(post-relational)라는 용어가 만들어졌고 하이브리드 객체 관계 데이터베이스의 개발로도 이어졌다.

target_date	target_time	target_time	target_time	target_time	target_time
2016-12-26	02:29:30	1482737402	2	10	10
2016-12-26	02:32:29	1482737502	2	10	10
2016-12-26	02:32:29	1482737502	2	10	10
2016-12-26	02:35:29	1482737502	2	10	10
2016-12-26	02:35:29	1482737502	2	10	10
2016-12-26	02:38:29	1482737502	2	10	10
2016-12-26	02:38:29	1482737502	2	10	10
2016-12-26	02:41:30	1482738123	2	10	10
2016-12-26	02:41:30	1482738123	2	10	10
2016-12-26	02:44:29	1482738502	2	10	10
2016-12-26	02:44:29	1482738502	2	10	10
2016-12-26	02:44:29	1482738502	2	10	10

SQL 데이터베이스 쿼리의 예.



2000년대 말에 관계형 이후의 차세대 데이터베이스는 NoSQL 데이터베이스로 알려지게 되었으며, 고속의 키-값 스토어, 문서 지향 데이터베이스를 도입하였다. NewSQL 데이터베이스라는 경쟁력 있는 차세대 데이터베이스는 상용 관계형 DBMS 대비 NoSQL의 높은 성능에 부합하면서 관계형/SQL 모델을 보유하는 새로운 구현을 시도하였다.

## 데이터베이스의 개념

---

여러 사람이 공유하고 사용할 목적으로 통합 관리되는 정보의 집합이다. 논리적으로 연관된 하나 이상의 자료의 모음으로 그 내용을 고도로 구조화함으로써 검색과 갱신의 효율화를 꾀한 것이다. 즉, 몇 개의 자료 파일을 조직적으로 통합하여 자료 항목의 중복을 없애고 자료를 구조화하여 기억시켜 놓은 자료의 집합체라고 할 수 있다.

공동 자료로서 각 사용자는 같은 데이터라 할지라도 각자의 응용 목적에 따라 다르게 사용할 수 있다.

### ■ 데이터베이스의 특징

1. 실시간 접근성
2. 지속적인 변화
3. 동시 공유
4. 내용에 대한 참조
5. 데이터 논리적 독립성

## 데이터베이스의 장단점

### ■ 데이터베이스 장점


1. 데이터 중복 최소화
2. 데이터 공유
3. 일관성, 무결성, 보안성 유지
4. 최신의 데이터 유지
5. 데이터의 표준화 가능
6. 데이터의 논리적, 물리적 독립성
7. 용이한 데이터 접근
8. 데이터 저장 공간 절약

### ■ 데이터베이스 단점

1. 데이터베이스 전문가 필요
2. 많은 비용 부담
3. 데이터 백업과 복구가 어려움
4. 시스템의 복잡함


## 데이터베이스 모델

---

 이 부분의 본문은 데이터베이스 모델입니다.

실제적인 데이터베이스 구현을 위한 현재 몇몇 개념화된 논리적 데이터 모델은 다음과 같다:

### 관계 데이터 모델

 이 부분의 본문은 관계형 모델입니다.

관계 데이터모델 (relational datamodel)은 데이터 모델 중에서 가장 개념이 간단한 모델이다. IBM 연구소에서 근무하던 코드(E.F.Codd)가 1970년에 제안했다. 이 모델은 상대 수학적 이론을 기반으로 한다. 코드가 수학자였기 때문에 수학 분야, 특히 집합론과 논리분야의 개념을 사용했다. 데이터 모델을 개발하기 위해서 테이블 관계로 묘사하는 이론적 모델 과정이 발생하는데 이를 개체관계모델(영어:entity relational model)이라고 한다. 이 관계 데이터베이스를 위한 설계과정은 이론적으로 관계수학에 기초한 실지 구현이라 보면 된다. 현실 세계는 객체관계 그림(다이어그램)으로 표현되며, 개체와 그 관계는 각기 사각과 선으로 그려진다.

### SQL

개체 관계형 데이터베이스를 지원하기 위해 1974년 IBM 연구소에서 만든 SQL(Structured Query Language)(원어(SEQUEL):Structured English Query Language)가 창안되었으며, 이 언어는 수학적 관계 대수와 관계 논리(relational calculus)에 기반을 두고 있다. 데이터 모델은 데이터를 조작하기 위한 연산 집합을 가져야 한다. 왜냐하면 그것은 데이터베이스 구조와 제약 조건을 정의하기 때문이다. 다시 말해, 관계 데이터모델 연산집합(a set of operations)은 관계대수로 표현되고, 그 연산은 사용자에게 여러 질의를 가능하게 한다.

## 데이터베이스 관리 시스템 선택

---

데이터베이스 설계 후 데이터베이스 관리 시스템을 사용해야 한다. 여러 가지 데이터베이스 관리 시스템 선택 사항(DBMS)이 존재한다.

데이터베이스 관리 시스템으로 현재에는 많이 사용하지 않으나 IBM 메인프레임 환경 하에서 운영되는 'IMS'가 있다. IMS는 정보 관리 시스템(Information Management System)의 약자로 DBMS와 DC 기능을 수행한다. IMS가 관리하는 DB 종류에는 'DEDB'와 'MSDB'가 있다. DEDB는 Data Entry DB로 전통적인 계층형 DB이며 MSDB는 주기억 DB로 시스템 가동시 주 메모리에 상주하는 DB로 빠른 접근이 필요한 업무에 사용된다. 다만, 접근 속도가 매우 빠른 반면 메모리에 상주하기 때문에 용량 및 구조에 제한이 많다.

## DBMS 언어 선택

---


데이터베이스 언어는 다음과 같이 이루어져 있다.

- 데이터 정의 언어(DDL:data definition language) - Create, Alter, Drop등의 명령어
- 데이터 조작 언어(DML:data manipulation language) - Select, Insert, Delete, Update...

- 데이터 제어 언어(DCL:data control language)- Grant, Revoke, Commit, Rollback...

## 트랜잭션

---

 이 부분의 본문은 데이터베이스 트랜잭션입니다.

트랜잭션은 하나의 논리적 단위를 구성하는 데이터베이스 연산의 모임이다. 동시에 여러 트랜잭션이 수행되기 위해서 데이터베이스의 일관성이 보장되어야 하며 이를 위해 동시성 제어(concurrency control)와 회복 제어(recovery control)를 위한 모듈이 있으며 이 둘을 합쳐 트랜잭션 관리 모듈(transaction management module)이라고 한다.

1. 동시성제어 모듈(concurrency control module): 데이터베이스를 일관성 있게 유지하기 위하여 동시에 수행되는 트랜잭션들 사이의 상호작용을 제어한다.
2. 회복제어 모듈(recovery control module): 데이터베이스를 일관성 있게 유지하기 위하여 업데이트를 하는 동안 시스템 장애에도 데이터베이스의 기존 상태가 유지된다.

트랜잭션 스케줄링은 다음과 같은 3가지 개념을 가진다.

1. 직렬 스케줄링(serial scheduling): 트랜잭션 연산들을 각 트랜잭션별로 연속적으로 실행하는 방법
2. 비직렬 스케줄링(nonserial scheduling): 트랜잭션 연산들을 상호적(interleaving)으로 병행 실행하는 방법
3. 직렬 가능 스케줄링(serializable scheduling): 비직렬 스케줄링 S가 항상 직렬 스케줄링 SS에 대해서 같은 결과를 가질 때 "S를 직렬가능"하다고 한다.

직렬가능 트랜잭션을 보장하기 위한 규약(protocol)이 있는데 잠금(locking) 방법과 시간표(timestamp)가 바로 그것이다.

## 데이터베이스 자료구조

---

### 인덱싱

데이터베이스는 흔히 다음과 같은 **ACID 규칙**을 만족해야 한다.

- 원자성 (原子性, Atomicity): 한 트랜잭션의 모든 작업이 수행되든지, 아니면 하나도 수행되지 않아야 한다. 트랜잭션이 제대로 실행되지 않았으면 롤백(roll back)한다.
- 일관성 (一貫性, Consistency): 모든 트랜잭션은 데이터베이스에서 정한 무결성 (無缺性, integrity) 조건을 만족해야 한다.
- 격리성 (隔離性, Isolation): 두 개의 트랜잭션이 서로에게 영향을 미칠 수 없다. 트랜잭션이 실행되는 동안의 값은 다른 트랜잭션이 접근할 수 없어야 한다.
- 내구성 (耐久性, Durability): 트랜잭션이 성공적으로 끝난 뒤에는, (시스템 실패가 일어나더라도) 그 결과가 데이터베이스에 계속 유지되어야 한다.

**병행제어** (並行制御, concurrency control)는 트랜잭션을 안전하게 처리하고 ACID 규칙을 만족시키는 기술이다.

# 데이터베이스의 수요

---

## 대한민국

2012년 한국의 국내 DB산업은 DB구축 시장, DB컨설팅·솔루션 시장, DB서비스 시장 등 모든 분야에서 전년 대비 높은 성장세를 나타내고 있다. 특히 전 산업에서의 정보통신기술(ICT) 융합과 스마트 환경 확산, 빅데이터 관련 수요가 증가하면서 향후 그 성장세는 계속될 것으로 내다봤다. 보고서에서는 DB산업의 성장을 내다보는 주요 요인으로 빅데이터 분석·활용을 위한 기업의 신규 수요 증가, DB자산 가치 인식 증대로 인한 DB구축 투자 증가, 스마트 기반의 모바일 서비스 확산 등을 꼽고 있다.<sup>[5]</sup>

## 각주

---

1. “Database - Definition of database by Merriam-Webster” (<http://www.merriam-webster.com/dictionary/database>). 《merriam-webster.com》 .
2. 한국문헌정보학회. 《최신 문헌정보학의 이해》 . 한국도서관협회. 401쪽. ISBN 9788976781130.
3. Bachman 1973.
4. “TOPDB Top Database index” (<https://web.archive.org/web/20161116070418/http://pypl.github.io/DB.html>). 《pypl.github.io》 . 2016년 11월 16일에 원본 문서 (<http://pypl.github.io/DB.html>)에서 보존된 문서. 2016년 11월 14일에 확인함.
5. 조윤주 기자 (2013년 2월 13일). “국내 DB산업 ‘빅데이터’ 날개 달고 고공행진..지난해 11조원 돌파” ([http://www.fnnews.com/view?ra=Sent0901m\\_View&corp=fnnews&arcid=201302130100104830005865&cDateYear=2013&cDateMonth=02&cDateDay=13](http://www.fnnews.com/view?ra=Sent0901m_View&corp=fnnews&arcid=201302130100104830005865&cDateYear=2013&cDateMonth=02&cDateDay=13)). 파이낸셜뉴스. 2013년 3월 20일에 확인함.

## 참고 문헌

---

- Elmasri, Navathe:Fundamentals of database systems, Addison Wesley.

## 외부 링크

---

- SourceForge의 데이터베이스 관련 프로그램 ([https://web.archive.org/web/20030422200127/http://sourceforge.net/softwaremap/trove\\_list.php?form\\_cat=66](https://web.archive.org/web/20030422200127/http://sourceforge.net/softwaremap/trove_list.php?form_cat=66))

---

원본 주소 "<https://ko.wikipedia.org/w/index.php?title=데이터베이스&oldid=33664760>"