**데이터베이스(최종 수정일 9/11)**

1. **데이터베이스의 이해**

**Information Overload, SCM(Supply Chain Management), CRM(Customer Relationship Management) 등, 데이터의 양적 증가로 DB 필요성 대두**

**초기 파일 시스템의 종속, 중복, 무결성 훼손 문제로 DBMS 등장**

**데이터 독립성:**

1. 논리적 독립성: 데이터의 논리적 구조가 변해도 프로그램의 구조는 변경되지 않는 것
2. 물리적 독립성: 기억장치인 하드웨어가 변해도 프로그램은 그대로 유지되는 것

**DBMS의 등장:** 데이터와 프로그램의 분리, 종속으로 인한 문제 극복

**데이터 중복(Data redundancy) 문제:** 비일관성(Inconsistency) 발생, 동일 데이터 동일 보안이 보장되지 않음

**데이터 무결성(Data Integrity) 문제:** 달리 말하면 데이터 정확성 훼손 문제

**동시 접근 이상:** 협의 없이 동시성을 제어하기란 불가능

**DBMS의 특징**

위와 같은 문제의 해결

**프로그램-데이터 독립성(Program-data Independency):** 프로그램의 형식/구조와 별도로 DBMS 내부의 시스템 카탈로그에 의해 관리. 저장장치 내의 물리적 위치가 아닌 개념적 표현(Conceptual Representation)을 통한 접근 제공. 이러한 데이터 구조화를 위한 데이터 추상화(Data Abstraction) 처리.

**자기 기술성(Self-Describing):** 소스 코드로 기술되는 프로그램 종속 데이터와 달리 DBMS의 데이터는 메타 데이터(시스템 카탈로그/데이터 사전)를 통한 자기 참조적 정의를 한다.

**시스템 카탈로그:** 특정 DB에 저장된 파일의 구조를 파악하기 위해 데이터 타입, 포맷 등 물리적 정보 + 데이터에 대한 설명, 의미 등 논리적 정보를 관리한다.

**값(Value), 메타 데이터, 데이터, 정보:** 수치나 독립된 문자열이 값이다. 값에 의미를 부여하는 것이 메타 데이터이다. 값과 메타 데이터의 합이 데이터이고 데이터에 맥락과 활용 가능한 의미가 부여되면 정보가 된다.

**다중 뷰:** 역할과 권한에 따라 데이터를 임의 편집해 제공하는 것. 전체 데이터는 하나이지만 다양한 관점 제공이 가능하다.

**다수 사용자 요청 처리:** 트랜잭션과 동시성 제어.

**데이터베이스 관리 시스템의 구조**

**데이터 추상화:** 내부/개념/외부 스키마로 추상화. 내부는 원시 단계(데이터 구조와 레코드 유형, 인덱스 유무, 저장된 컬럼의 표현 방식…), 개념 스키마는 데이터의 논리적 관계 기술(보안/무결성 검사는 포함), 외부 스키마는 뷰에 의해 기술.

**단계 간 사상(Mapping):**

1. **외부-개념 사상(External-Conceptual Mapping):** 외부 스키마와 개념 스키마 간 대응관계 정의. 논리적 데이터 독립성 확보
2. **개념**-**내부 사상(Conceptual-Internal Mapping):** 개념 단계의 스키마가 디스크 내의 내부 필드와 어떻게 대응하는가 정의. 물리적 데이터 독립성 확보

**데이터베이스 언어**

**DDL:** 데이터 정의어. 데이터의 논리적 구성/특징을 데이터 모델에 의거해 규정한다. 데이터가 기억장치에 저장되도록 물리적 특성을 규정하고 물리적 규정을 논리적 구성으로 변환할 수 있도록 물리적 구성과 논리적 구성 간의 사상을 규정한다. DDL 명령의 결과는 시스템 카탈로그 또는 데이터 사전에서 관리한다. 시스템 카탈로그는 특수 형태의 테이블로 오직 DBMS에 의해서만 사용/수정된다. DBMS는 실제 데이터를 읽고 쓰기 전에 항상 데이터 사전을 참조한다.

**DML:** 데이터 조작어. DDL에 의해 구조화된 데이터에 CRUD를 제공한다. 크게 절차적 DML과 선언적/비절차적 DML로 나뉜다.

**데이터베이스 시스템 아키텍처**

**중앙집중식 데이터베이스 관리 시스템 구조:** 터미널을 통한 중앙 집중형 데이터베이스 시스템, 병목현상으로 성능 저하, 오류 발생 시 전체 시스템에 영향을 미치는 문제

**클라이언트-서버 구조:** 프로그램 부하를 분산시키고 시스템의 성능을 향상시킬 수 있으며 프로그램 유지보수 비용을 절감하고 이식성을 증가시킬 수 있다.

**2계층 클라이언트-서버 구조:** 2-tier architecture. 소프트웨어 구성요소들이 클라이언트와 서버로 분산되어 2계층 구조라고 한다. 장점 – 단순함, 기존 시스템과의 하위 호환성. 질의 처리와 트랜잭션 기능을 서버 측에 남겨 둬 질의 서버(query server) 또는 트랜잭션 서버(transaction server)라고 한다. 또 대부분의 서버가 SQL언어를 표준으로 사용해 SQL server라고도 한다.

**3계층 클라이언트-서버 구조:** 3-tier architecture. 클라이언트와 데이터베이스 서버 사이에 애플리케이션 서버를 추가한 구조이다. 웹 서버의 경우 웹 애플리케이션 서버(WAS; Web Application Server)라고 부르기도 한다. 데이터베이스 서버에 저장된 데이터에 접근하는 데 사용되는 비즈니스 규칙(프로시저 제약 조건)들을 저장하는 중간 역할을 수행한다. 클라이언트는 일부 비즈니스 규칙을 포함하고 WAS는 클라이언트의 요청을 받아 처리하고 데이터베이스 명령을 DB서버로 보낸 뒤에 서버에서 처리된 데이터를 가시화 작업 후 클라이언트로 보내는 통로 역할을 한다.

**데이터베이스 사용자 및 관리자**

**사용자:** 사용자는 크게 네 가지로 분류하며 유형에 따라 별도의 인터페이스를 제공한다.

1. **일반 사용자:** 사원, 사서 등 DB에 관한 지식 없이 애플리케이션 인터페이스로 작업을 수행한다.
2. **애플리케이션 프로그래머:** 애플리케이션 인터페이스 개발자
3. **전문 사용자:** DBMS 조작 지식이 있는 사용자. 데이터 분석가
4. **특수 사용자:** 전문 사용자 중 전통적인 데이터 처리 프레임워크에 속하지 않은 CAD 시스템, 지식기반 전문가 시스템, 복합 데이터 타입(그래픽, 오디오 등)을 저장하는 시스템과 환경 모델링 시스템 등 특수한 데이터베이스 애플리케이션을 작성하는 이용자.

**관리자:** DBA는 DB 관련 소프트웨어를 관리/감독하는 사람이나 그러한 전문가 집단을 말한다. DBA의 목표는 사용자를 위해 데이터 처리의 효율을 극대화 하는 것이다. 이를 세분화하면 설계, 관리, 운용 및 통제, 성능 측정으로 나눌 수 있다.

1. **설계:** 스키마를 구성(내부, 개념, 외부)하고 데이터 사전도 구성한다. 또한 보안 절차를 수립하고 보안 기능을 사용하려는 사용자의 요구사항을 수용한다. 시스템 고장 시 백업과 회복 절차를 마련해 데이터베이스의 무결성을 유지할 수 있어야 한다.
2. **관리:** 사용자와 소통하기 위한 일이다. 데이터 표현 방법과 시스템 문서화의 기준을 정하고 하드웨어 선택에 관여한다.
3. **운용 및 통제:** 사용자 요구나 데이터 이용 패턴에 따라 최적의 물리적 저장구조와 하드웨어를 선택/유지한다. 무결성을 유지하기 위해 접근과 접근통제 방법을 정의한다.
4. **성능 측정:** 자원 사용 분석, 병목 현상 조사, 데이터 이용 형태 조사, 장치 별 성능측정 등을 수행한다. 이를 바탕으로 물리적 저장구조를 재구성하고 데이터 접근 방식을 갱신하다. 또 경제적 효율성을 최대화하기 위한 스케줄링 알고리즘을 결정한다.

**MySQL과 MySQL 워크벤치**

**MySQL Server:** GPL에 기반한 오픈소스 라이선스와 오라클 서비스와 코드 기밀성이 보장되는 상업 라이선스로 나뉜다. 원래 오픈소스 프로젝트는 마리아DB로 분리됐다. 현재 MySQL은 네 가지 Edition을 제공한다.

1. **MySQL Standard Edition:** MySQL Server에 외부 연결을 위한 다양한 커넥트를 제공한다. 리플리케이션을 통해 저비용 고가용성의 환경 구축이 가능하다. MyISAM 엔진을 많이 사용하며 이의 단점을 보완한 InnoDB엔진도 있다. InnoDB엔진은 트랜잭션과 테이블을 제공한다.
2. **MySQL Enterprise Edition:** 대용량 데이터베이스 성능향상을 위한 테이블 파티셔닝 기능을 제공한다. MySQL Enterprise Monitor 기능을 통해 실시간 모니터링 툴을 제공해 관리 차원에서 다양한 정보를 제공한다. 또 MySQL Dump, MySQL Hot Copy보다 향상된 성능의 백업 기능이 있다.
3. **MySQL Cluster Edition:** 대용량 데이터의 실시간 트랜잭션 처리를 위한 다중 서버 클러스터링을 제공한다. 오픈 소스 데이터베이스와 무정지형(fault tolerant) 데이터베이스 클러스터링 아키텍처를 결합해 99.999% 가용성을 제공한다. 사용자의 니즈에 맞게 퍼포먼스, 신뢰성, 규모 등에 대한 컨트롤을 제공하도록 유연한 아키텍처를 가지고 있다.
4. **MySQL Community Edition:** GPL에 의거한 오픈 소스 배포 버전, Standard Edition과 매우 유사

**+GNU GPL:** 소프트웨어의 실행, 연구, 공유, 수정의 자유를 최종 사용자에게. 사용자는 GNU GQL 라이선스 프로덕트를 통해 개발한 결과물을 반드시 동일한 GNU GQL 라이선스로 배포해야 한다. 때문에 상용 소프트웨어로 쓰기엔 부적합하다. 최신 버전인 GULv3은 수정한 소스 코드를 서버에서만 사용하는 개발자가 해당 프로그램을 배포하지 않을 경우 사용자가 소스 코드를 가질 수 없는 문제를 해결하기 위해 등장했다. 이를 위해 서버에서 프로그램을 실행해 다른 유저와 통신하면 실행하고 있는 프로그램의 소스를 다른 사용자들이 다운로드할 수 있도록 해야한다는 조항이 추가되었다.

**MySQL 워크벤치:** MySQL Server는 GUI 인터페이스를 제공하지 않는다. 워크벤치는 이러한 불편함을 제거하기 위해 오라클 사에서 제공하는 GUI 인터페이스 애플리케이션이다. DB 구조 설계도 시각적으로 지원한다. fabFORCE NET의 DBDesigner 4의 후속판이다. 5.2 버전부터 종합 데이터베이스 GUI 애플리케이션으로 발전했다. 주요 기능은 다음과 같다.

* **GUI DB모델링 지원:** 에러가 발생하기 쉬운 업무를 자동화하고 데이터베이스 디자인과 유지를 간단히 해 DBA와 개발자 사이의 커뮤니케이션을 향상시킨다. 새 ER 다이어그램을 설계할 때나 물리적 데이터베이스를 생성할 때 발생할 실수를 방지한다.
* **Forward & Reverse Engineer 기능 제공:** 물리적 데이터베이스 디자인의 Forward Engineer 기능을 제공해 MySQL서버에서 클릭으로 쉽게 물리적 데이터베이스로 변환할 수 있다. 또 이미 존재하는 DB나 패키지 어플에 대한 리버스 엔지니어링 기능을 제공할 뿐만 아니라 모델 설계를 위한 SQL스크립트를 읽거나 DDL 스크립트로 모델을 내보낼 수 있는 기능을 제공한다.
* **변경 이력 추적 관리:** DB 변경 관리는 어렵고 복잡한 프로세스이다. 스키마의 각 버전을 유지하고 현재 DB를 수동으로 수정하는 것과 관련있다. 워크벤치를 이용해 스키마 동기화와 비교 유틸리티를 포함해 사용자들의 DB 변경 이력을 추적할 수 있다.

1. **데이터베이스 모델링**

**DB모델링의 이해**

요구사항 분석 – 개념 – 논리 – 물리 모델링을 거친다.

개념 모델링은 실세계를 집합적으로 분류 가능한 대상으로 일반/추상화 한다.

논리 모델링은 개념 모델링 된 개념을 특정 DB시스템의 논리 구조에 맞게 규격화 한다.

물리 모델링은 해당 논리 구조를 실제 하드웨어 규격에 맞게 설계하는 일이다.

**사용자 요구분석 단계:** 데이터를 활용하는 목표 수립, 목표를 달성하기 위한 단계별 문서화 작업 -> 명세서(specification) 산출. 업무와 업무 간 데이터의 흐름을 가능한 상세하게 작성한다. 요구사항 도출(요구사항 명세서 도출) – 요구사항 분석(요구사항 정의서 작성) – 요구사항 기록 단계를 거친다.

**데이터 모델링 단계:** 시스템의 대상이 되는 업무를 적절한 표기법(notation)으로 표현하는 것이 모델링이다. 표현 대상은 크게 세 가지 범주로 구분한다.

1. **데이터 관점:** 데이터 간의 관계 중심
2. **프로세스 관점:** 비즈니스 로직을 처리하는 연산 중심
3. **데이터-프로세스 상관 관점:** 프로세스에 따라 영향을 받는 데이터 중심

요구사항 명세서는 요구에 따라 데이터 흐름도, 순서 다이어그램, 시나리오 등 다양한 형태로 기술된다. 데이터베이스 모델링은 이와 같은 명세서를 논리/물리적으로 활용 가능한 DB구축을 위해 구체화하는 것이다.

**사용자 요구분석 -> 개념적 모델링 -> ER 모델 -> 논리적 모델링 -> 관계형 모델 -> 물리적 모델링 -> 물리적 세부사항 -> 내부 스키마 순으로 완성된다.**

1. **개념적 모델링:** 실세계의 대상으로부터 데이터 타입, 속성, 관계, 제약조건을 도출해 추상화한다. 의미 객체 모델, 시맨틱 네트워크 모델 등 다양한 개념적 데이터 모델이 있으나 가장 많이 사용되는 건 ER(Entity-Relation)모델이다. ER모델은 말그대로 개체 간의 관계를 통해 데이터를 표현한다.
2. **논리적 모델링:** 개념적 모델을 특정 DBMS에 맞는 구현체(스키마)로 변환하는 작업이다. 데이터 구조와 데이터 간 관계가 목표 DBMS의 형식으로 변환된다. 구현 데이터 모델은 관계형 모델, 객체지향형 모델, 객체-관계형 모델 등이 있으며 NoSQL 진영에서는 Key/Value 모델, Document 모델 등도 사용한다. 일반적으로는 관계형 모델을 사용하며 최종적으로 목표 DBMS의 데이터 정의 언어(DDL)로 기술된 논리 스키마가 생성된다.
3. **물리적 모델링:** 논리 스키마는 데이터들의 논리적 구성만을 명시한다. 여기에 DB파일의 내부 저장구조, 파일 구성, 인덱스, 접근 경로 등을 더하는 것이 물리적 모델링이다. 물리적 모델링은 따라서 논리 스키마가 실제 하드웨어에 어떻게 적재될지 선택하는 작업이다. 전체 DBMS의 성능에 영향을 미치는 작업이므로 하드웨어 특성을 고려해 처리능력을 향상시키고 요구사항을 최대한 만족시킬 방법을 찾아야 한다. 레코드 저장방법, 데이터 타입 및 저장공간 등도 물리적 모델링에서 설계한다.

**사용자 요구사항 분석 과정**

DBMS를 구축하려는 수탁업체가 **제안 요청서(RFP: Request for Proposal)**를 받아 **요구사항 목록**을 도출해내는 게 첫 단계이다. 수탁업체는 요구사항 목록에서 시스템 요구사항을 주제에 맞게 코드로 분류해야 한다. 분류가 끝나면 유사한 요구사항을 통합해 세부적인 내용을 더한 **요구사항 정의서를** 작성한다. 요구사항 정의서 작성단계는 다음 세 단계를 거친다.

1. 현재 운용 중인 시스템의 사용자 지침서, 인터뷰 결과같은 요구사항 관련 문서를 토대로 기술서 개요를 작성
2. 프로젝트 목적, 기능/비기능 요구사항, 가정과 위험 요소, 용어와 데이터 정의 및 사용자 인터페이스 식별을 명세화
3. 사용자와 내용 합의, 하나의 업무 단위로 수행될 수 있는 단위로 업무를 도출해 업무 내용을 기술한다.

**ER 모델**

1976 첸 박사가 제안. DB의 논리적 구조를 표현하고자 하는 모델링 도구이다. MySQL 워크벤치, Erwin, dbForge Studio for Oracle 등 다수의 DB설계 도구들이 ER모델을 지원한다. ER모델은 크게 개체 집합, 관계 집합, 속성으로 구성된다.

**개체 집합:** 개체 집합은 개체와 개체를 표현하는 속성의 집합이다. 하나의 DB는 다수의 개체 집합으로 이루어진다. 초기 ER 표기법은 명확한 개체, 속성 표현과 제약 조건을 표현하기에 적합했으나 대규모 DB 모델링이 일반화된 이후로 거의 쓰이지 않는다. 현재 MySQL 워크벤치를 비롯한 많은 DBMS 툴은 ER표기법보다 **UML(Unified Modeling Language)표기법**을 사용한다.

UML표기법은 두 개의 직사각형을 이용해 개체 집합을 표현한다. 개체명은 상단 직사각형, 키 속성은 첫 번째 칸에, 이외 속성은 두 번째 칸에 위치한다. UML형식에서는 키와 도메인이 드러난다.

**관계 집합:** 학생과 학과는 전공이라는 관계로 연결할 수 있다. 이런 관계를 **관계 인스턴스(relationship instance)**라고 한다. 일반적으로 개체 집합은 관계를 통해 해석될 때 비로소 정보로서 가치를 가진다. 관계 집합은 같은 유형의 인스턴스 집합이다. 한 개체 집합이 다른 개체 집합과 연관성을 가지는 것을 관계 집합에 **참가(participate)**한다고 표현한다. 관계 집합에 참여하는 개체 집합의 수를 **차수(degree)**라고 부른다. 관계에 참여하는 개체의 기능을 개체의 **역할**이라고 한다. 관계 집합에 참가하는 개체 집합들은 역할 구별이 쉽게 이루어져 역할은 묵시적으로 정해진다. 하지만 동일한 개체 집합이 재귀적으로 다른 역할로 참가하는 **재귀적 관계 집합**의 경우 역할을 명시해야 한다.

**속성:** ER모델에서 기본 요소는 개체 집합이다. 개체 집합은 속성의 집합이다. 속성은 값의 특성에 따라 단순 속성, 복합 속성, 단일값 속성, 다중값 속성, 유도 속성, 저장 속성 등으로 구분된다.

1. **단순 속성과 복합 속성:** 단순 속성은 원자값인 속성이다. 복합 속성은 더 작은 의미단위로 나눌 수 있는 속성이다. 생일같은 속성은 년,월,일,시간 등으로 세분화할 수 있으므로 복합 속성에 해당한다. 모델링 과정에서 작은 구성단위가 반복적으로 사용될 경우 모델링 상에 중복해 포함하기도 한다. 단순 속성과 복합 속성은 들여쓰기로 표현된다.
2. **단일값 속성과 다중값 속성:** 한 개체가 특정 속성에 단 한 가지 값을 가질 경우 단일값 속성이다. 한 사람이 고유한 주민번호를 가지는 것과 같다. 다중값 속성은 한 개체가 한 속성을 다양하게 가질 수 있는 속성이다. 예로 전화번호가 있다.
3. **유도 속성과 저장 속성:** 유도 속성은 타 속성을 통해 유도 가능한 속성이다. 대표적으로 생일을 통해 나이를 유도할 수 있다. 실제 값을 저장해야만 유지되는 속성은 저장 속성이라고 한다. 유도 속성은 함수라는 의미의 괄호 ‘()’를 사용해 표현한다.

**제약조건:** 실세계를 표현하는 데 가장 중요한 추상적 개념 두 가지를 꼽으라면 개체 집합과 관계 집합이다. 관계 집합은 다른 개체 집합으로 개체 간의 대응성, 사상을 의미한다. ER모델링은 집합에 대한 상세 표현을 위해 DB가 준수해야 하는 제약조건(constraints)를 부여할 수 있다. 세 가지 대표적인 제약조건으로 사상수, 참가 제약조건, 키 속성

1. **사상수(mapping cardinality): 1:1, 1:N, N:1, N:N.** **1은 화살표가 있는 실선**, **N은 화살표가 없는 실선**
2. **참가 제약조건(participation constraint):** 관계에 참여하는 개체 집합의 범위를 나타낸다. 모든 개체가 전부 관계에 참여한다면 이는 당연히 **전체적 참가**라고 한다. 일부만 참여해도 되면 **부분적 참가. 전체 참여는 이중 실선, 부분 참여는 하나의 실선**
3. **키 속성:** 최소의 속성으로 개체 집합 내의 모든 개체를 구분할 수 있는 것. ER모델에서는 속성 이름에 밑줄로 표현.

**ER 모델의 기호**

**ER모델**의 논리적 구조는 **ER 다이어그램(ERD; ER diagram)**으로 표현.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

개체와 관계 집합 간 재귀적 속성일 때만 주의하자.

1. **관계형 모델**

**관계형 데이터베이스 구조**

**1969년 코드 박사**가 제안한 데이터 모델로 **술어 논리(predicate logic)**와 **집합론**을 기반으로 한다. ER모델이 개체 집합과 개체 집합을 연결하는 관계 집합으로 현실의 업무를 추상화한다면 관계형 모델은 테이블, 컬럼, 기본키-외래키 참조 관계 등으로 현실 관계를 표현한다. **테이블, 행렬, 기본키-외래키, 참조 등으로 대표된다.**

**릴레이션:** 릴레이션 내부의 한 값이 행과 열의 관계로 결정된다는 의미로 관계형 모델이 되었다. ER모델의 관계는 개체 집합 간의 참여 관계로 정의되었다면 관계형 DB의 관계는 행과 열의 관계이다. 관계형 DB는 행을 **레코드, 튜플** 열을 **칼럼, 필드**라고 한다. 속성은 **도메인**이라고 하는 범위를 가진다. 각 부분은 데이터베이스로서의 기능을 위해서 다음과 같은 특징을 가진다.

**레코드의 유일성:** 기본키가 유일하니 당연하다

**레코드의 무순서성**

**컬럼의 무순서성**

**컬럼값의 원자성:** 컬럼은 분해가 불가능하며 원자적이다.

컬럼의 부분집합은 해당 릴레이션의 레코드를 유일하게 식별하는 역할을 한다. 이러한 컬럼의 부분집합을 키라고 하며 키는 **유일성과 최소성**을 만족시켜야 한다.

릴레이션 R의 키 중 적어도 한 키에 포함되는 컬럼을 릴레이션 R의 주속성(prime attribute)이라고 하고 그렇지 않은 속성을 비주속성(non-prime attribute)라고 한다.

**수퍼키:** 레코드를 고유하게 식별할 수 있는 컬럼의 집합, 여러 개가 될 수도 있다. 수퍼키 중 최소한의 칼럼으로 구성된 집합을 후보키(candidate key)라고 한다. 모든 후보키는 수퍼키이지만 모든 수퍼키가 후보키일 수는 없다. {단과대학, 주소}와 {학과이름}은 수퍼키이지만 {단과대학, 주소}는 후보키가 될 수 없다. 수퍼키 > 후보키 > 기본키. 후보키 중 DBA가 임의로 선택한 키가 기본키가 된다. 기본키는 널값을 가질 수 없지만 외래키나 일반 컬럼은 널 값을 가질 수 있다. 이는 개체 무결성 제약조건 때문이다.

**릴레이션 스키마**

어떤 테이블에서 사용하는 컬럼과 컬럼이 지니는 데이터 타입을 정의한 것을 말한다. 쉽게 얘기해서 속성 모음이다. 인스턴스와의 차이는 인스턴스는 릴레이션 스키마에 구체적인 값을 갖는다는 것이다. 인스턴스는 유동적이고 릴레이션 스키마는 고정적이다. 릴레이션 스키마는 타 테이블의 릴레이션 스키마와 중복관계를 통해 조인 연산을 가능하게 한다.

**관계형 모델의 제약조건(constraints):** 데이터의 무결성과 가용성을 보존하기 위해 거는 조건이다.

영역 제약조건, 키 제약조건, 개체 무결성 제약 조건, 참조 무결성 제약 조건이 있다.

1. **영역 제약조건(domain constraints):** 컬럼이 가질 수 있는 값에 대한 제약조건. 컬럼의 값은 원자적이며 도메인 안에 있어야 한다.
2. **키 제약조건(key constraints):** 같은 테이블 내에 동일한 레코드는 있을 수 없다. 적어도 하나의 키가 모든 레코드를 식별할 수 있어야 한다.
3. **개체 무결성 제약조건(entity integrity constraints):** 기본키는 NULL일 수 없다. 두 개 이상의 기본키가 널값을 가지면 구분할 수 없어 무결성이 훼손되기 때문이다. 일단 NULL로 설정하고 추후에 기본키 값을 갱신하려 해도 이미 NULL값이 들어간 레코드를 특정해 수정할 수 없다.
4. **참조 무결성 제약조건(referential integrity constraints):** 한 릴레이션이 다른 릴레이션을 참조하려면 반드시 참조되는 릴레이션이 참조하는 릴레이션 내에 있어야 한다. 외래키 얘기다.

**외래키를 위한 두 가지 조건**

1. **R1의 외래키의 컬럼들은 R2의 기본키의 속성들과 동일한 영역을 가진다.**
2. **R1의 한 레코드의 외래키는 R2의 어떤 컬럼의 기본키와 일치하거나 널 값을 가져야 한다.**

R1이 R2의 기본키를 외래키로 사용할 때 외래키가 NULL값일 수 있는 경우는 R1의 해당 레코드가 R2와 어떤 연관성도 없는 경우이다.

**논리적 데이터 모델링**

**ER다이어그램의 관계형 모델 변환:** ERD의 모든 표현은 훼손 없이 관계형 모델의 스키마로 변환되어야 한다. 다음 단계를 순차적으로 적용한다.

1. **개체 집합:** 개체 집합을 릴레이션으로 변환한다. 개체 집합의 키 속성 중 하나를 릴레이션의 기본키로 지정한다.
2. **약한 개체 집합:** 독자적으로 존재 불가능하고 관계 속에서만 존재하는 집합이 약한 개체 집합이다. 약한 개체 집합을 릴레이션으로 변환할 경우 지배 개체인 강한 개체 집합의 키 속성을 포함시킨다. 강한 개체란 독자적으로 존재할 수 있는 개체이다. 변환된 릴레이션의 기본키는 강한 개체 집합의 키와 약한 개체 집합의 부분 키를 묶어서 지정한다.
3. **일대일 관계:** 일대일 사상수를 가지는 이항 관계가 있다면 관계 집합에 참여하는 두 릴레이션 중 어느 하나를 다른 릴레이션의 기본키를 참조하는 외래키 컬럼으로 변환시킨다.
4. **일대다 혹은 다대일 관계:** 일대다 이항관계는 ‘다’쪽 릴레이션을 외래키 컬럼으로 변환한다. ‘일’쪽의 기본키를 참조하도록 외래키를 지정한다.
5. **다대다 관계:** 별도의 관계 릴레이션을 생성해야 한다. 관계에 참여하는 두 릴레이션의 기본키를 각각 참조하는 외래키로 복합키 형태의 컬럼을 구성한다.
6. **다중값 속성:** 기존 릴레이션의 기본키와 다중값 속성을 별도의 릴레이션으로 구성한다. 이후 이 둘을 새 릴레이션의 기본키로 지정한다.
7. **관계 집합의 속성:** 각각의 속성을 외래키가 위치한 릴레이션의 컬럼으로 삽입한다.
8. **n항 관계:** n개 이상의 개체 집합이 참여한 관계의 경우, 관계에 참여하는 n개의 릴레이션 키로 구성되는 관계 릴레이션으로 변환한다. 관계 릴레이션의 컬럼들은 참여 릴레이션의 기본키를 참조하는 외래키와 관계 컬럼으로 구성한다.