**정보처리기사 실기 2장. 데이터 입출력 구현**

***최종수정일 – 2023/04/03***

**중요사항(암기사항)**은 **볼드체**

**시나공 정보처리기사 실기2023 요약.**

**29. 데이터베이스 개요**

정보를 컴퓨터에서 사용하기 위해 논리적 구조로 조직화하고, 다시 그 논리적 구조를 물리적으로 조직화해 물리적 저장장소에 저장한 게 데이터베이스이다.

**데이터베이스**

여러 사람들이 **공동**으로 사용할 **정보**에서 **중복을 제거**하고 **통합**한뒤 **저장장치**에 저장해 항상 사용할 수 있도록 **운영**하는 시스템이다.

다음과 같이 구분된 정의를 가진다.

* **통합된 데이터(Integrated Data): 중복을 제거**한 **데이터 모임**
* **저장된 데이터(Stored Data): 컴퓨터로 접근**할 수 있는 **저장 매체**에 저장된 자료
* **운영 데이터(Operational Data): 조직을 운영**하는데 있어 **반드시 필요한** 자료
* **공용 데이터(Shared Data): 여러 시스템**들이 **공동으로 소유한** 자료

**DBMS(Database Management System; 데이터베이스 관리 시스템)**

기존의 파일 시스템이 갖는 데이터의 종속성과 중복성의 문제를 해결하기 위해 제안된 시스템으로 사용자가 바라는 대로 데이터베이스에 정보를 생성하고 이를 관리해주는 시스템이다. **DBMS**로 활용되기 위해서는 다음과 같은 필수 기능 세 가지(**DMC**)를 갖춰야 한다.

**D**efinition(정의 기능): 데이터의 type과 구조에 대한 정의, 이용 방식, 제약 조건 등을 명시

**M**anipulation(조작 기능): 데이터 검색, 갱신, 삽입, 삭제를 위한 인터페이스 제공

**C**ontrol(제어 기능): 데이터의 무결성, 보안, 권한 검사, 병행 제어를 제공하는 기능

**DMC(정의, 조작, 제어) 기억하자.**

**데이터의 독립성**

독립성 🡨----🡪 종속성, 논리적 독립성과 물리적 독립성이 있다.

**논리적 독립성 –** 응용프로그램과 데이터베이스의 분리를 말한다. 이렇게 하면 데이터베이스의 논리적 구조를 변경해도 응용프로그램에 영향을 미치지 않는다.

**물리적 독립성 –** 응용프로그램과 보조기억장치 같은 물리적 장치를 독립시켜 디스크를 추가/변경해도 응용 프로그램은 영향을 받지 않는다.

**스키마(Schema)**

데이터베이스의 구조와 제약조건에 관한 전반적인 명세를 기술한 것이다.

**종류.**

**외부 스키마:** 사용자나 응용 프로그래머가 각 개인의 입장에서 필요로 하는 데이터 베이스의 논리적 구조를 정의한 것

**개념 스키마:** 데이터베이스의 논리적 구조, 응용 프로그램과 사용자들이 필요로 하는 데이터를 종합한 조직 전체의 데이터베이스이다. 단 하나 뿐이다.

**내부 스키마:** 물리적 저장장치 관점에서 데이터베이스, 실제 저장될 레코드 형식, 저장 데이터 항목 표현 방법, 내부 레코드의 물리적 순서 표시

**데이터베이스 설계**

소프트웨어의 요구 공학처럼 데이터베이스도 사용자의 요구를 분석해 컴퓨터에 저장할 데이터베이스의 구조에 맞게 변형한 후 DBMS로 데이터베이스를 구현한다.

**데이터베이스 설계 시 고려사항**

* **무결성 :** 삽입, 삭제, 갱신 등 연산 후에도 데이터가 정해진 제약 조건을 항상 만족해야 한다.
* **일관성 :** 특정 데이터에 접근하거나 질의를 할 때 응답에 변함 없어야 한다.
* **회복 :** 시스템장애가 발생하면 장애 이전 상태로 복구 가능해야 한다.
* **보안 :** 허가되지 않은 데이터의 노출, 변경, 손실로부터 데이터를 보호할 수 있어야 한다.
* **효율성 :** 응답시간 단축, 시스템 생산성, 저장 공간의 최적화가 가능해야 한다.
* **데이터베이스 확장 :** 데이터베이스 운영에 영향을 주지 않으면서 지속적으로 데이터를 추가할 수 있어야 한다.

**데이터베이스 설계 순서**

**요구조건 분석 – 개념적 설계 – 논리적 설계 – 물리적 설계 – 구현**

**개념적 설계(정보 모델링, 개념화)**

개념 스키마 모델링과 트랜잭션 모델링을 병행 수행한다. 요구 분석에서 나온 요구 조건 명세를 DBMS에 독립적인 E-R 다이어그램으로 작성함으로 DBMS에 독립적인 개념 스키마를 설계한다.

**\*개념 스키마는 DBMS에 독립적이다.**

**논리적 설계(데이터 모델링)**

현실의 개념을 특정 DBMS에 맞는 논리적 자료 구조로 변환(mapping)하는 과정이다. 데이터를 필드로 기술된 데이터 타입과 데이터 타입 간의 관계로 표현되는 논리적 구조로 모델링한다. 개념적 설계가 개념 스키마를 설계하는 단계라면 논리적 설계는 개념 스키마를 평가, 정제하고 DBMS에 따라 서로 다른 논리적 스키마를 설계하는 단계이다. 트랜잭션의 인터페이스를 설계한다.

**물리적 설계(데이터 구조화)**

논리적 구조로 표현된 데이터를 하드웨어에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정이다. 파일 저장경로, 액세스 경로 등을 설정한다. 또 저장 레코드의 형식, 순서, 접근 경로, 조회 집중 레코드 등의 정보를 사용해 데이터가 컴퓨터에 어떻게 저장될지 결정한다.

**데이터베이스 구현**

논리 스키마와 물리 스키마를 실제 파일로 생성하는 과정이다. 사용 DBMS의 데이터 질의어(DDL)을 통해 데이터베이스 스키마를 기술하고 컴파일해 데이터베이스 파일을 생성한다. 응용 프로그램을 위한 트랜잭션을 생성하고 데이터베이스에 접근하기 위한 응용 프로그램을 구현한다.

**31. 데이터 모델의 개념**

**데이터 모델**

구성요소: 개체 속성 관계

종류: 개념적 모델, 논리적 모델, 물리적 모델

데이터 모델에 표시할 요소: 구조, 연산, 제약조건

**개념적 데이터 모델**

**정보모델, E-R 다이어그램으로 표현, 결과적으로 데이터를 개체, 속성, 관계로 구분하는 일**

**속성은 행렬에서 열의 구분기준이 되고 개체는 행의 기준이 된다. 관계는 개체의 속성 type과 이 type들 간의 관계로 설명된다.**

**논리적 데이터 모델**

**단순히 데이터 모델**이라고 하면 **논리적 데이터 모델**을 의미한다. **개념적 테이터 모델을 컴퓨팅 논리에 적합하게 변형**시킨 모델이다. **테이블, 레코드, 필드, 엔티티** 등을 통해 **현실을 추상화**한 데이터 모델로 변형시킨 모델이 논리 데이터 모델이다. DBMS에 맞는 논리 데이터 모델 하나를 선정해 사용한다. 데이터 간의 관계를 어떻게 표현하냐에 따라 관계 모델, 계층 모델, 네트워크 모델로 구분한다.

**참고. 테이블=행렬, 레코드=행, 필드=열, 엔티티=현실세계의 존재를 논리 모델로 성형한 것**

**데이터 모델에 표시할 요소**

* **구조:** 논리적 모델링을 거친 개체 타입들 간의 관계를 말한다. 데이터 구조의 정적 성질을 표현한다.
* **연산:** DBMS에서 데이터 처리 방식을 명세화한 것이다. 데이터 조작 도구를 말한다.
* **제약조건:** DBMS에 저장될 수 있는 데이터의 논리적 제약조건이다.

**32. 데이터 모델의 구성요소**

**개체(Entity)**

현실 속 유, 무형의 대상을 컴퓨팅 환경에서 다루기 위해 개념적으로 만들어낸 모델이다. 유일한 식별자(Unique Identifier)에 의해 구분되며 다른 개체와 하나 이상의 관계에 있다.

개체는 객체가 아님에 유의하자

**속성(Attribute)**

데이터베이스를 구성하는 **가장 작은 논리단위**, **데이터 필드(열)을 구분하는 기준이라고 생각하면 된다.** 속성의 수를 **디그리 또는 차수**라고 한다. 속성은 속성의 특성과 개체 구성 방식에 따라 분류한다.

**속성의 특성에 따른 분류**

**기본 속성: 속성 중 가장 많고 일반적이다. 도메인 영역**에서 뽑아낸 속성이다. **하지만 도메인에서 뽑아냈어도 업무상 코드로 정의된 속성이라면 기본 속성에서 제외된다.**

**설계 속성:** 도메인에 없고 설계, 데이터 모델링 과정에서 나오는 속성.

**파생 속성:** 다른 속성에서 계산, 변경 등 영향 받아 생기는 속성. 적을수록 좋다.

**속성의 개체 구성 방식에 따른 분류**

**기본키:** 개체를 식별할 유일한 속성

**외래키:** 다른 개체와의 관계에서 포함된 속성

**일반 속성:** 개체에는 포함되어 있으나 기본키, 외래키에 포함되지 않는 속성

**관계(Relationship)**

개체와 개체 사이의 논리적 연결

**관계의 형태**

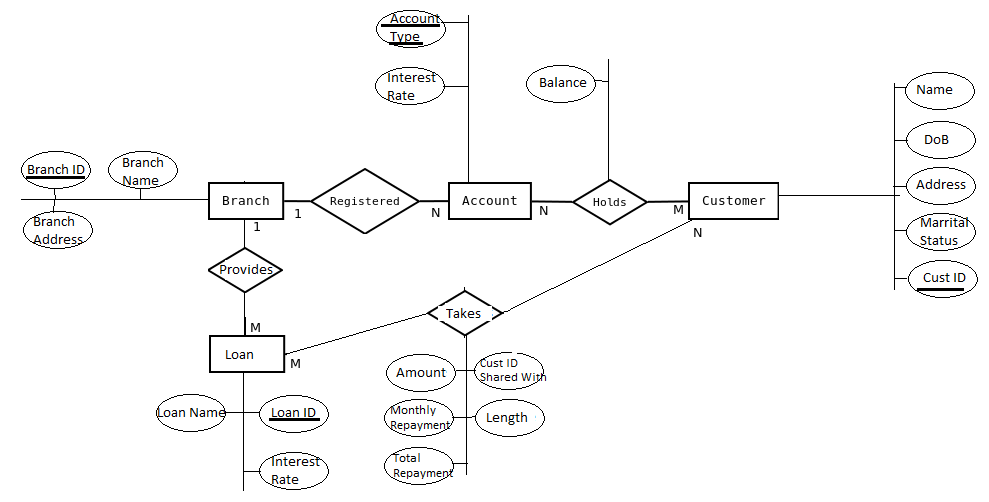
**1:1, 1:N, N:M**

**관계의 종류**

**종속 관계, 중복 관계(없을수록 좋지만 반정규화 등 필요에 의해 쓰기도 함), 재귀 관계, 배타 관계**

**33. E-R(개체-관계) 모델**

엔터티-릴레이션쉽 모델. 현실세계의 무질서를 개체의 관계 속에서 사용 가능한 논리 데이터로 표현한다. 개체, 관계, 속성으로 데이터를 표현하며 1:1, 1:N, N:M 등의 관계 유형을 사용할 수 있다. 개념적 모델의 대표격



알 수 없는 작성자 님의 [이 사진](http://stackoverflow.com/questions/20313095/bank-er-notation-example)에는 [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/) 라이선스가 적용됩니다.

사각형 = 개체

마름모 = 관계

타원 = 속성

이중타원 = 다중값 속성(복합 속성)

밑줄 타원 = 기본키 속성

복수 타원 = 복합 속성

네모-마름모-네모 = 관계

선 = 개체 타입과 속성 타입을 연결

**32. 관계형 데이터베이스의 구조/관계형 데이터 모델**

관계형 DB는 2원 표(Table)을 이용해 데이터의 관계를 정의하는 데이터베이스이다. 1970년 IBM의 코드(E.F.Codd)에 의해 제안되었다. 개체와 관계를 릴레이션이라는 표로 표현해 개체를 표현하는 개체 릴레이션과 관계를 표현하는 관계 릴레이션이 존재한다.

**장점:** 간결하고 보기 편하다. 데이터베이스로 변환이 용이하다.

**단점:** 상대적으로 성능이 떨어진다.

**관계형 데이터베이스의 릴레이션 구조**

릴레이션이란 2차원 테이블이다. 어트리뷰트(= 디그리, 칼럼, 열)의 집합인 릴레이션 스키마와 실제 값들인 릴레이션 인스턴스(= 카디널리티, 튜플, 행)로 구성된다. 2차원 행렬을 행과 열의 관계 관점에서 말하면 릴레이션이다.

**튜플 = 행**

릴레이션을 구성하는 행이다. 튜플의 수를 카디널리티, 기수, 대응수, 속성의 모임이라고 한다. 파일 구조에서 레코드와 같은 의미이다.

**속성(Attribute)**

데이터베이스를 구성하는 가장 작은 논리적 단위. 데이터 항목 또는 필드에 해당된다. 개체의 특성을 나타내며 디그리 또는 차수라고 한다.

**도메인(Domain)**

하나의 속성이 취할 수 있는 원자 값의 집합. 한 속성이 20세 이상/미만일 때 값으로는 20세 이상과 미만만 올 수 있다. 이때 (20세 이상, 미만)이 해당 속성의 도메인이 된다.

**릴레이션의 특징**

튜플과 속성(어트리뷰션)에는 순서가 없다. 동일한 튜플은 존재하지 않는다. 이 점은 집합과 같다. 또 릴레이션은 튜플의 삽입, 삭제로 인해 시간에 따라 변한다. 속성의 이름은 모두 달라야 하지만 속성을 구성하는 도메인은 서로 같을 수 있다. 튜플을 식별하기 위한 속성의 부분집합을 키로 설정한다. 속성값은 논리적으로 쪼개질 수 없는 원자값만을 지정한다.

**관계형 데이터 모델(Relational Data Model)**

2차원 테이블을 통해 데이터 상호 관계를 정의하는 DB구조. 가장 널리 사용되는 데이터 모델이다. 기본키와 이를 참조하는 외래키로 데이터간 관계를 표현하며 계층 모델과 망 모델의 복잡한 구조를 단순화한 모델이다. 관계형 모델의 대표는 SQL이고 1:1, 1:N, N:M.

**관계형 데이터베이스의 제약 조건 – 키(key)**

조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 기준이 되는 속성.

**종류.**

* **후보키(Candidate Key)**
* **기본키(Primary Key)**
* **대체키(Alternate Key)**
* **슈퍼키(Super Key)**
* **외래키(Foreign Key)**

**후보키(Candidate Key)**

튜플을 유일하게 식별하기 위해 사용되는 속성들의 부분집합, 기본키로 사용할 수 있으나 사용하지 않는 속성들이다. 후보키는 유일성과 최소성을 모두 만족하는 키를 말한다. 유일성은 말 그대로 하나의 키로 하나의 튜플을 유일하게 식별할 수 있는 성질을 말하고 최소성은 키를 구성하는 속성 하나를 제거하면 더이상 키를 식별할 수 없어야 하는 속성을 말한다.

**기본키(Primary key)**

후보키 중에 특별히 선정된(Main key = Primary key)이다. 중복된 값을 가질 수 없고 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성이다. 키본키는 널값을 가질 수 없다.

**대체키(Alternate key)**

후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키, 보조키라고도 한다.

**슈퍼키(Super Key)**

속성들의 집합으로 구성된 키. 슈퍼키는 모든 튜플에 대해 유일성은 만족하지만 최소성은 만족하지 못한다.

**외래키(Foreign Key)**

다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합

**36. 관계형 데이터베이스의 제약조건 – 무결성(Intergrity)**

데이터에 문제가 없을 경우 무결하다.

**무결성의 종류**

* **개체 무결성**: 기본키의 속성은 NULL과 중복값을 가질 수 없다.
* **참조 무결성**: 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다.
* **도메인 무결성**: 속성값은 도메인 값을 벗어날 수 없다.
* **사용자 정의 무결성**: 속성값은 사용자가 정의한 제약조건을 만족시켜야 한다.
* **NULL 무결성**: 릴레이션의 특정 속성 값이 NULL이 될 수 없도록 하는 규정
* **고유 무결성**: 튜플은 고유해야 한다.(릴레이션의 특정 속성에 대해 튜플이 갖는 속성값들이 서로 달라야 한다.)
* **키 무결성**: 하나의 릴레이션에는 적어도 하나의 키가 존재해야 한다.
* **관계 무결성**: 튜플들 사이 관계의 적절성 여부

**데이터 무결성 강화**

애플리케이션, 데이터베이스 트리거, 제약 조건을 이용해 강화할 수 있다.

애플리케이션 – 데이터 생성, 수정, 삭제 시 무결성을 검증하는 코드를 추가한다.

데이터베이스 트리거 – 트리거 이벤트에 무결성 조건을 실행하는 SQL을 추가한다.

제약 조건 – 제약조건 설정으로 무결성을 유지한다.

**37. 관계대수 및 관계해석**

관계대수란 관계형 데이터베이스에서 원하는 정보와 그 정보를 찾기 위해 사용하는 절**차적 언어**이다. **관계대수는 해를 구하기 위한 연산의 순서를 명시한다**. 관계형 데이터베이스에서 활용하기 위해 개발한 **순수 관계 연산자**와 집합 이론에서 사용하는 **일반 집합 연산자**가 있다.

**순수 관계 연산자**

* **Select: σ**(시그마 기호) 사용. 조건을 만족하는 튜플의 부분집합을 구해 새 릴레이션을 만든다. 행을 구하는 것이므로 수평 연산이라고도 한다.
* **Project: π(파이 사용)** 조건을 만족하는 속성의 부분 집합을 추려 새 릴레이션을 만든다. 중복이 발생하면 중복을 제거한다. 열에 해당하는 속성을 추출하므로 수직 연산자라고도 한다.
* **Join: ⋈로 표시**, 공통 속성을 중심으로 두 개의 릴레이션을 하나로 합쳐 새 릴레이션을 만든다. Join의 결과는 Cartesian Product(교차곱)을 수행한 뒤 Select를 수행한 결과와 같다.
* **Division: ÷기호 사용, X⊃Y**인 두 개의릴레이션 R(X)와 S(Y)가 있을 때 R의 속성이 S의 속성값을 모두 가진 튜플에서 S가 가진 속성을 제외한 속성만을 구한다.

**일반 집합 연산자**

수학 집합 이론에서 사용하는 연산자를 말한다. 합집합, 교집합, 차집합을 처리하기 위해서는 합병 조건을 만족해야 한다. 합병 조건이랑 합치려는 두 릴레이션의 디그리가 같고 대응되는 속성별로 도메인이 같아야 한다는 조건이다.

튜플의 합병과 관련된 일반 합집합, 교집합, 차집합을 떠올리면 된다.

교차곱(Cartesian Product)만 유의하자.

**교차곱 –** 두 릴레이션에 있는 튜플의 순서쌍을 구한다. 합병 조건을 만족하는 두 릴레이션 R과 S가 있을 때, R x S = {r.s| r ∈ R ∧ s ∉ S}이 된다. 교차곱의 디그리는 두 릴레이션의 디그리를 더한 것과 같고 카디널리티는 두 릴레이션의 카디널리티를 곱한 것과 같다.

**관계해석(Relational Calculus)**

관계 데이터의 연산을 표현하는 방법이다. 코드(E. F. Codd)가 수학의 Predicate Calculus(술어 해석)에 기반을 제안했다. 원하는 정보를 정의하기 위해 계산 수식을 활용한다.

**!관계대수와 관계해석의 차이 -> 관계해석은 원하는 게 뭔지에 대한 것만 정의하는 비절차적 특성, 관계대수는 해를 구하기 위한 연산의 절차를 명시**

**38. 이상/함수적 종속**

**이상**이란 테이블에서 일부 속성들의 종속으로 데이터의 중복이 발생할 때 이 중복으로 인해 조작에 문제가 발생하는 것을 말한다. 삽입 이상, 삭제 이상, 갱신 이상 등이 있다.

**함수적 종속(Functional Dependency)**

테이블 R에 대해 X와 Y가 R의 속성 부분집합이라 하자. 시간에 관계없이 항상 속성 X의 값에 대해 속성 Y의 값이 하나만 연결되어 있을 때 Y는 X에 함수적 종속 되었다고 한다. 또는 X가 Y를 함수적으로 결정한다고 말하고. X -> Y로 표기한다.

X를 결정자라고 하고 Y를 종속자라고 한다.

**39. 정규화(Normalization)**

속성들이 상호 종속적인 관계임을 이용해 테이블을 손실없이 분해하는 과정이다. 쉽게 말해 중복을 제거해 삽입, 삭제, 갱신 이상을 줄이는 일이다. 제1정규형부터 제5정규형까지 총 다섯 가지 정규화가 있으며 위로 갈수록 정규화 정도가 높아진다.

보통 3단계까지만 한다.

**정규화 과정**

* **제1정규형:** 테이블에 속한 모든 속성의 도메인이 원자값이어야 한다.
* **제2정규형:** 기본키가 아닌 모든 속성이 기본키에 대해 완전 함수적 종속을 만족하는 정규형
* **제3정규형:** 기본키가 아닌 모든 속성이 기본키에 대해 이행적 함수적 종속을 만족하지 않는 정규형
* **BCNF:** 모든 결정자가 후보키(Candidate Key)인 정규형
* **제4정규형:** 다중 값 종속(MVD; Multi Valued Dependency) A ->-> B가 존재할 경우 모든 속성이 A에 함수적 종속 관계를 만족하는 정규형
* **제5정규형:** 모든 조인 종속(JD; Join Dependency)이 후보키를 통해서만 성립되는 정규형

**도부이결다조**

**도메인이 원자값(1형) -> 부분적 함수 종속 제거(2형) -> 이행적 함수 종속 제거(3형)**

* **결정자이면서 후보키가 아닌 것 제거(BCNF) -> 다치 종속 제거(4형) -> 조인 종속성 이용(5형)**

**40. 반정규화(Denormalization)**

시스템의 성능 향상, 개발 및 운영의 편의성 향상을 위해 의도적으로 통합, 중복, 분리해 정규화 원칙을 위배하는 행위이다. 과도하게 하면 오히려 성능이 저하된다.

**방법**

* **테이블 통합**
* **테이블 분할**
* **중복 테이블 추가**
* **중복 속성 추가**

**테이블 통합**

두 테이블이 조인되는 경우가 많을 경우 하예 하나의 테이블로 만들어 사용하는 것.

두 개의 테이블에서 동일 프로세스를 자주 사용하거나 항상 두 개의 테이블을 이용해 조회하는 경우 통합을 고려할 수 있다. 1:1 통합, 1:N 통합, 슈퍼타입/서브타입 테이블 통합 등이 있다.

**테이블 분할**

레코드의 사용 빈도 차이가 너무 크면 수평 분할, 하나의 테이블에 속성이 너무 많을 경우 수직 분할 등을 고려한다.

**중복 테이블 추가**

작업의 효율성을 위해 중복 테이블을 추가하는 것이다. 여러 테이블에서 데이터를 추출해야 하거나 다른 서버에 저장된 테이블을 이용해야 하는 경우 중복 테이블을 추가한다. 집계 테이블의 추가(트리거), 진행 테이블의 추가(이력관리), 특정 부분만을 포함하는 테이블의 추가 등이 있다.

**중복 속성 추가**

조인해서 데이터를 처리할 때 데이터 조회 경로를 단축하기 위해 자주 사용하는 속성 하나를 추가하는 것이다. 중복 속성을 추가하면 데이터 무결성 확보가 어렵고 디스크 공간이 추가로 필요하다.

**중복 속성 추가하는 경우**

* **조인이 자주 발생하는 속성인 경우**
* **접근 경로가 복잡한 속성인 경우**
* **액세스의 조건으로 자주 사용되는 속성인 경우**
* **기본키의 형태가 적절하지 않거나 여러 개의 속성으로 구성된 경우**

**41. 시스템 카탈로그**

시스템과 관련있는 다양한 객체들의 정보를 포함하는 시스템 데이터베이스이다. 카탈로그들은 데이터 사전에 저장된다. 이때문에 좁은 의미에서 카탈로그를 데이터 사전이라고도 한다.

**메타 데이터**

데이터를 정의하는 데이터를 말한다. 시스템 카탈로그에 저장된 정보를 말한다.

* 데이터베이스 객체 정보: 테이블, 인덱스, 뷰 등의 구조 및 통계 정보
* 사용자 정보: 아이디 패스워드, 권한
* 테이블의 무결성 제약 조건 정보: 기본키, 외래키, 널값 허용 여부 등
* 함수 , 프로시저, 트리거 정보

**데이터 디렉터리**

데이터 사전에 수록된 데이터에 접근하는 데 필요한 정보를 관리 유지하는 시스템

**트랜잭션 분석/CRUD 분석**

1. 트랜잭션은 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 수행되어야 할 일련의 연산들을 의미한다.
2. 병행 제어 및 회복 작업 시 처리되는 작업의 논리적 단위로 사용된다.
3. 사용자의 요구 명령에 대한 상태 변환 과정의 작업 단위이다.

**트랜잭션의 특성**

**원자성(반영 혹은 롤백), 일관성(실행에 대한 결과는 언제나 같게), 독립성=격리성=순차성(둘 이상의 트랜잭션이 동시 실행되는 경우 하나의 트랜잭션의 동작 중에 다른 트랜잭션이 끼어들 수 없다.), 영속성=지속성(성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나도 영구적으로 반영되어야 한다.)**

**CRUD(CREATE, READ, UPDATE, DELETE) 분석**

프로세스와 테이블 간에 CRUD 매트릭스를 만들어 트랜잭션을 분석하는 일이다. 많은 트랜잭션이 몰리는 테이블을 파악할 수 있게 해준다.

**CRUD 매트릭스**

행은 프로세스 열은 테이블, 행렬이 만나는 위치에는 CRUD 중 프로세스가 테이블에 발생시킨 변화를 적는다. 복수의 변화가 적용될 때는 C > D > U > R을 우선순위로 하여 적거나 우선순위가 높은 한 가지만 적는다. 매트릭스 완성 후 CRUD 중 어느 것도 적히지 않은 행이나 열, C나 R이 없는 열을 확인해 불필요하거나 누락된 테이블 프로세스를 찾을 수 있다.

**트랜잭션 분석**

매트릭스를 분석해 테이블에 저장되는 데이터의 양을 유추하고 이를 근거로 데이터 베이스의 용량 산적과 최적화를 진행한다. 업무 개발 담당자가 수행하며 프로세스가 과도하게 접근하는 테이블을 여러 디스크에 분산 배치해 입출력 성능 향상을 기대해볼 수 있다.

**트랜잭션 분석서**

프로세스와 CRUD 매트릭스를 통해 작성한다.

**43. 인덱스**

키값과 포인터 쌍으로 구성되는 데이터 구조로 데이터 레코드에 빠르게 접근하기 위해 만든다. 인덱스 개수를 최소로 유지하는 게 효율적

**인덱스의 종류**

* **트리 기반 인덱스:** 인덱스를 저장하는 블록들이 트리 구조를 이룬다.
* **비트맵 인덱스:** 인덱스 컬럼의 데이터를 0, 1로 변환해 인덱스 키로 사용
* **함수 기반 인덱스:** 컬럼값 대신 함수나 수식을 적용해 산출된 값을 사용
* **비트맵 조인 인덱스:** 다수의 조인된 객체로 구성된 인덱스
* **도메인 인덱스:** 필요한 인덱스를 직접 만들어 사용

**클러스터드/넌클러스터드 인덱스**

* **클러스터드 인덱스:** 인덱스 키 순서에 따라 데이터가 정렬되어 저장. 실제 데이터가 순서대로 저장되어 있어 인덱스를 검색하지 않아도 원하는 데이터를 빠르게 찾을 수 있다.
* **넌클러스터드 인덱스:** 인덱스 키 값만 정렬되어 있고 실제 데이터는 정렬되지 않는다. 데이터 삽입, 삭제 발생 시 순서를 유지하기 위해 데이터를 재정렬해야 한다.

**44. 뷰/클러스터**

**뷰**는 접근 권한이나 필요에 따라 기존 테이블에서 일부를 떼어낸 **가상 테이블**이다. **저장장치에 물리적으로 존재하지 않는다**. 가상 테이블이기에 **뷰가 정의된 기본 테이블 또는 뷰를 삭제하면 해당 뷰와 관련 있는 다른 뷰도 자동삭제된다**. **클러스터**는 데이터 **저장 시 액세스 효율을 증가시키기 위해 동일한 성격의 데이터를 동일한 데이터 블록에 저장**하는 **물리적 저장 방법**이다. 클러스터링은 데이터 조회 속도를 향상시키지만 입력, 수정, 삭제 작업 성능을 저하시킨다. 데이터의 분포도가 넓을수록 유리하기도 하다.

**뷰의 장/단점**

* 장점: 논리적 데이터에 독립성 제공, 동일 데이터에 접근하는 여러 사용자의 응용이나 요구를 지원, 데이터 관리를 간단하게 해주며 접근 제어를 통해 자동 보안 제공
* 단점: 인덱스를 가질 수 없고 뷰의 정의를 변경할 수 없다. 삽입, 삭제, 갱신에 제약이 따른다.

**파티션**

대용량의 테이블이나 인덱스를 작은 논리적 단위로 나누는 것. 테이블 단위로 데이터 처리를 한다.

**파티션의 장/단점**

장점 – 액세스 범위를 줄여 쿼리 성능이 향상된다. 파티션별로 데이터가 분산되어 저장되므로 디스크 성능이 향상된다. 속도가 빠르고 시스템 장애시 데이터 손상 정도를 최소화할 수 있다.

단점 – 하나의 테이블을 쪼개서 파티션으로 관리하므로 관리에 보다 신경써야 한다. 테이블간 조인 비용이 증가하고 용량이 작은 테이블을 파티셔닝하면 오히려 성능이 떨어진다.

**파티션의 종류**

* 범위 분할: 지정한 컬럼의 값을 기준으로 분할한다.
* 해시 분할: 해시 함수를 적용한 결과값에 따라 데이터를 분할. 데이터가 고른 컬럼의 데이터 분산에 유리
* 조합 분할: 범위 분할 + 해시 분할, 범위 분할의 파티션이 너무 커서 관리가 어려울 때 유용하다.

**46. 분산 데이터베이스 설계**

용량 설계는 데이터가 저장될 공간을 정의한다. 데이터 양과 인덱스, 클러스터 등이 차지하는 공간을 예측해 반영한다. 확장성 가용성을 높이고 병목현상을 최소화하기 위해 한다.

**분산 데이터베이스 설계**

논리적으로는 하나이지만 물리적으로는 네트워크를 통해 연결된 여러 개의 사이트(Site)에 분산된 데이터베이스를 말한다. 데이터 처리나 이용이 많은 지역에 DB를 위치시킨다. 애플리케이션이나 사용자가 분산되어 저장된 데이터에 접근하게 하는 것을 목표로 한다.

**분산 데이터베이스의 목표**

**위치 투명성(Location Transparency)**: 데이터의 물리적 위치를 알 필요없이 논리적 명칭만으로 데이터에 액세스할 수 있는 투명성을 말한다.

**중복 투명성(Replication Transparenct):** 동일 데이터가 여러 곳에 중복되어 있어도 사용자는 하나의 데이터만 존재하는 것처럼 사용하고 시스템은 여러 자료에 대한 작업을 수행한다.

**병행 투명성(Concurrency Transparency):** 다수의 트랜잭션이 동시에 실현되어도 그 트랜잭션의 결과는 영향을 받지 않는다.

**장애 투명성(Failure Transparency):** 트랜잭션, DBMS, 네트워크, 컴퓨터 장애에도 트랜잭션을 정확히 처리한다.

**분산 설계 방법**

테이블 위치 분산: 테이블을 각기 다른 서버에 분산해 배치

분할: 완전성, 재구성, 상호 중첩 배제의 규칙을 따라 수평 또는 수직 분할

할당: 동일한 분할을 여러 개의 서버에 생성, 중복 포함 할당과 중복 없는 할당을 나뉜다.

**데이터베이스 이중화/ 서버 클러스터링**

이중화는 동일한 데이터를 복제해 관리하는 것이다. 하나의 데이터베이스에 문제가 발생하면 복제된 데이터베이스를 이용해 즉시 문제를 해결할 수 있다. 애플리케이션을 여러 데이터베이스에 분산해 처리하므로 데이터베이스의 부하를 줄일 수 있어서 쓴다.

**데이터베이스 이중화 분류**

**Eager 기법**: 트랜잭션 수행 중에 데이터 변경이 발생하면 이중화된 모든 데이터 베이스에 즉시 전달해 변경 내용이 즉시 적용되도록 한다.

**Lazy 기법:** 트랜잭션 수행이 종료되면 변경 사실을 새로운 트랜잭션에 작성해 각 데이터베이스에 전달한다. 데이터베이스마다 새로운 트랜잭션이 수행되는 것으로 간주한다.

**데이터베이스 이중화 구성 방법**

**활동-대기 방법(Active – Strandby):** 한 디비가 활성 상태면 다른 디비는 대기하다가 장애 발생시 대기하던 디비가 자동으로 모든 서비스를 대시 수행한다. 구성과 관리가 쉽다.

**활동-활동(Active – Active):** 두 개의 디비가 서로 다른 서비스를 제공하다가 한 쪽에 장애가 발생하면 다른 디비가 서비스를 제공한다. 처리율이 높지만 구성이 복잡

**클러스터링**

두 개 이상의 서버를 하나처럼 운영하는 기술, 서버 이중화, 공유 스토리지, 고가용성

**고가용성 클러스터링:** 하나의 서버에 장애가 발생하면 다른 노드(서버)가 받아 처리해 서비스 중단을 방지한다. 일반적인 클러스터링이 고가용성 클러스터링이다.

**병렬 처리 클러스터링:** 전체 처리율을 높이기 위해 하나의 작업을 여러 개의 서버에서 분산하여 처리하는 방식이다.

**RTO/RPO**

**RTO(Recovery Time Objective; 목표 복구 시간):** 비상사태 이후 복구까지 얼마나 걸렸는지

**RPO(Recovery Point Objective;** **목표 복구 시점):** 비상사태 이후 데이터를 복구할 수 있는 기준점(ex. 어제 오후 여섯 시 기준으로 데이터 복구 가능)

**데이터베이스 보안**

보안은 이용자들의 액세스 권한을 관리하는 일이다.

**암호화(Encryption)**

암호화 과정과 복호화 과정으로 나뉘며 크게 두 가지 암호화 기법이 사용된다.

* 개인키 암호화 방식
* 공개키 암호화 방식

**접근통제**

액세스 통제, 접근통제는 3요소를 외우면 된다.  **접근통제 정책, 메커니즘, 보안모델**

**접근통제 기술**

**임의 접근통제(DAC, Discretionary Access Control):** 사용자의 신원에 따라 접근 권한을 부여한다. 데이터 소유자가 권한을 지정하고 제어한다.

**강제 접근통제(MAC, Mandatory Access Control):** 시스템이 접근 권한을 지정한다. 주체와 객체의 등급을 비교해 접근 권한을 부여하며 사용자별로 인가 등급을 부여할 수 있다.

**역할기반 접근통제(RBCA, Role Based Access Control):** 사용자의 역할에 따라 접근 권한을 부여한다. 중앙관리자가 권한을 지정하고 다중 프로그래밍에 최적화된 방식이다.

**접근통제 정책**

신분기반, 규칙기반, 역할기반 정책 등이 있다. 또 보안 모델로는 기밀성 모델(군사용, 수학적 모델), 무결성 모델(무결성 기반), 접근통제 모델 등이 있다.

**감사추적**

사용자나 애플리케이션이 데이터베이스에 접근해 수행한 모든 활동을 기록한다. 데이터베이스를 복구하거나 부적절한 데이터 조작을 파악하기 위해 활용한다.

**49. 데이터베이스 백업**

로그 파일을 기준으로 Undo(복귀-트랜잭션 취소) or Redo(회복-트랜잭션 재실행)하여 데이터베이스를 복구하기 위한 작업을 말한다.

**데이터베이스 복구 알고리즘**

NO-UNDO/REDO: 데이터베이스 버퍼의 내용을 비동기적으로 갱신한 경우 복구 알고리즘

NO-UNDO: 트랜잭션 완료 전에 변경 내용이 데이터베이스에 기록되지 않으므로 취소할 필요가 없다.

REDO: 트랜잭션 완료 후 데이터베이스 버퍼에는 기록되어 있고 저장매체에는 기록되지 않았으므로 트랜잭션 내용을 다시 실행해야 한다.

UNDO/NO-REDO: 데이터베이스 버퍼의 내용을 동기적으로 갱신한 경우 복구 알고리즘

UNDO: 트랜잭션 완료 전에 시스템이 파손되었다면 변경 내용을 취소한다.

NO-REDO: 트랜잭션 완료 전에 데이터베이스 버퍼 내용을 이미 저장 매체에 기록했으므로 트랜잭션 내용을 다시 실행할 필요가 없다.

UNDO/REDO: 버퍼의 내용을 동기/비동기적으로 갱신한 경우 복구 알고리즘

NO-UNDO/NO-REDO: 버퍼의 내용을 동기적으로 저장 매체에 기록하지만 데이터베이스와는 다른 영역에 기록한 경우의 복구 알고리즘

**백업 종류**

물리 백업과 논리 백업으로 나뉜다.

**물리 백업:** 데이터베이스 파일을 백업하는 방법이다. 백업 속도가 빠르고 작업이 단순하지만 문제 발생 시 원인 파악이 어렵다.

**논리 백업:** DB내의 논리적 객체들을 백업하는 방법이다. 원인 파악/ 문제 해결이 수월하지만 백업/복원 시 시간이 많이 소요된다.

**50. 스토리지**

대용량의 데이터를 저장하기 위해 서버와 저장장치를 연결하는 기술이다. DAS, NAS, SAN등의 스토리지가 있다.

**DAS(Direct Attached Storage)**

서버와 저장장치를 전용 케이블로 직접 연결하는 방식.

**NAS(Network Attached Storage)**

서버와 저장장치를 네트워크를 통해 연결하는 방식

**SAN(Storage Area Network)**

서버와 저장장치를 연결하는 전용 네트워크를 별도로 구성한다. 광 채널 스위치를 이용하며 여러 개의 저장장치를 활용해 확장성, 유연성, 가용성이 뛰어나다.

**51. 논리 데이터 모델의 변환**

**엔티티 – 테이블**

논리 데이터 모델에서 정의한 엔티티를 물리 데이터 모델의 테이블로 변환한다.

엔티티 – 테이블

속성 – 컬럼

주 식별자 – 기본키

외부 식별자 – 외래키

관계 – 관계

왼편이 논리 설계(데이터 모델링) 오른편이 물리 설계

**슈퍼타입/서브타입을 테이블로 변환**

슈퍼타입 기준 테이블 변환, 서브타입 기준 테이블 변환, 개별타입 기준 테이블 변환 등의 방식으로 논리 데이터 모델을 물리 데이터 모델로 변환한다.

**속성을 컬럼으로 변환**

**일반 속성 변환, Primary UID를 기본키로 변환 등의 방법이 있다.**

**52. 자료구조(자료구조 파트는 키워드 별로 디테일한 학습이 필요하다. 구글링)**

**선형 자료구조- 배열, 선형리스트(연결, 연속 리스트), 스택(LIFO), 큐(FIFO), 덱(스택+큐)**

**비선형 자료구조- 트리, 그래프**

**53. 트리**

정점과 간선으로 구성된 그래프의 특수한 형태. 노드, 근 노드(루트 노드), 디그리(가지수), 단말 노드(자식이 하나도 없는 노드), 비단말노드(디그리가 0이 아닌 노드), 조상 노드(임이의 노드 근처에서 근 노드에 이르는 경로상에 있는 노드들), 레벨(루트 노드를 레벨1로 가정하고 깊이에 따라 레벨 증가), 깊이(노드가 가질 수 있는 최대 레벨)

**54. 이진 트리**

차수가 2 이하인 노드들로 구성된 트리가 이진 트리이다. 트리의 운행법에 따라 **프리오더, 인오더, 포스트오더** 등이 있다. 또 표기법에 따라 **전위, 중위, 후위 표기**가 있다. 모두 시험에 나온다.

**55. 정렬**

**삽입, 선택, 버블, 쉘, 퀵, 힙, 2-way 합병, 기수 정렬 출제. 정렬 개념과 시간 복잡도를 물어본다.**