

시험에 나오는 것만 공부한다!

시나공



기출문제집

길벗알앤디, 강윤석, 김용갑,
김우경 지음

나 는 시 험 에 나 오 는 것 만 공 부 한 다 !

정보처리기사 필기





1권

핵심 요약

- 1과목 · 데이터베이스
- 2과목 · 전자계산기 구조
- 3과목 · 운영체제
- 4과목 · 소프트웨어 공학
- 5과목 · 데이터 통신

불합격 방지용
안전장치

기억상자



틀린 문제만 모아 오답 노트를 만들고
까먹기 전에 다시 한 번 복습하고
싶다고요?

지금 당장 QR 코드를 스캔하거나 www.membox.co.kr에 접속해 보세요.



1과목 · 데이터베이스

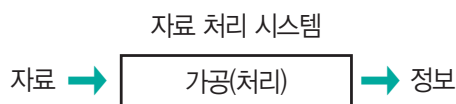
001 정보 시스템

정보 시스템

- 조직체에 필요한 Data를 수집, 저장해 두었다가 필요 시에 처리해서 의사 결정에 유용한 정보를 생성하고 분배하는 수단이다.
- 사용하는 목적에 따라 경영 정보 시스템, 군사 정보 시스템, 인사 행정 정보 시스템, 의사 결정 지원 시스템 등으로 사용된다.

정보와 자료

- 자료(Data) : 현실 세계에서 관찰이나 측정을 통해 수집한 단순한 사실이나 결과값으로, 가공되지 않은 상태
- 정보(Information) : 의사 결정에 도움을 줄 수 있는 유용한 형태로, 자료를 가공(처리)해서 얻는 결과물



- 자료 처리 시스템 : 정보 시스템이 사용할 자료를 처리하는 정보 시스템의 서브 시스템으로, 처리 형태에 따라 일괄 처리 시스템, 온라인 실시간 처리 시스템, 분산 처리 시스템으로 분류
- 데이터웨어 하우스(DataWare House) : 조직이나 기업체의 중심이 되는 주요 업무 시스템에서 추출되어 새로이 생성된 데이터베이스로서 의사 결정 지원 시스템을 지원하는 주체적, 통합적, 시간적 데이터의 집합체

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요? 까먹기 전에 다시 한 번 복습하고 싶다고요? 지금까지 공부한 내용을 안전하게 시험장까지 가져가는 완벽한 방법이 있습니다. 지금 당장 QR 코드를 스캔해 보세요.



www.membox.co.kr을 직접 입력해도 접속할 수 있습니다.

002 데이터베이스의 정의

- 통합된 데이터(Integrated Data) : 자료의 중복을 배제한 데이터의 모임
- 저장된 데이터(Stored Data) : 컴퓨터가 접근할 수 있는 저장 매체에 저장된 자료
- 운영 데이터(Operational Data) : 조직의 업무를 수행하는데 있어서 존재 가치가 확실하고 없어서는 안 될 반드시 필요한 자료
- 공용 데이터(Shared Data) : 여러 응용 시스템들이 공동으로 소유하고 유지하는 자료

003 데이터베이스의 특징

- 실시간 접근성(Real Time Accessibility) : 수시적이고 비정형적인 질의(조회)에 대하여 실시간 처리(Real-Time Processing)에 의한 응답이 가능함
- 계속적인 변화(Continuous Evolution) : 새로운 데이터의 삽입(Insertion), 삭제(Deletion), 갱신(Update)으로 항상 최신의 데이터를 유지함
- 동시 공유(동시 공유)(Concurrent Sharing) : 여러 사용자가 동시에 자기가 원하는 데이터를 이용할 수 있음
- 내용에 의한 참조(Content Reference) : 데이터베이스에 있는 데이터를 참조할 때 데이터 주소나 위치에 의해서가 아니라 사용자가 요구하는 데이터 내용으로 데이터를 찾음

004 기존의 파일 처리 방식에서의 문제점

종속성으로 인한 문제점

- 종속성이란 응용 프로그램과 데이터 파일이 상호 의존적인 관계를 말한다.
- 데이터 파일이 보조기억장치에 저장되는 방법이나 저장된 데이터의 접근 방법을 변경할 때는 응용 프로그램도 같이 변경해야 한다.

중복성으로 인한 문제점

- 일관성 : 중복된 데이터 간에 내용이 일치하지 않는 상황이 발생하여 일관성이 없어짐



- 보안성 : 중복되어 있는 모든 데이터에 동등한 보안 수준을 유지하기가 어려움
- 경제성 : 저장공간의 낭비와 동일한 데이터의 반복 작업으로 인한 비용의 증가
- 무결성 : 제어의 분산으로 인해 데이터의 정확성을 유지할 수 없음

핵심 005 DBMS의 필수 기능

- 정의(조직)(Definition)
 - 데이터의 형(Type)과 구조, 데이터가 DB에 저장될 때의 제약조건 등을 명시하는 기능이다.
 - 데이터와 데이터의 관계를 명확하게 명세할 수 있어야 하며, 원하는 데이터 연산은 무엇이든 명세할 수 있어야 한다.
- 조작(Manipulation)
 - 데이터 검색(요청), 갱신(변경), 삽입, 삭제 등을 체계적으로 처리하기 위해 데이터 접근 수단 등을 정하는 기능이다.
 - 사용자와 데이터베이스 사이의 인터페이스 수단을 제공한다.
- 제어(Control)
 - 데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되어 데이터의 무결성이 유지되도록 제어해야 한다.
 - 정당한 사용자가 허가된 데이터만 접근할 수 있도록 보안(Security)을 유지하고 권한(Authority)을 검사할 수 있어야 한다.
 - 여러 사용자가 데이터베이스를 동시에 접근하여 데이터를 처리할 때 처리 결과가 항상 정확성을 유지하도록 병행 제어(Concurrency Control)를 할 수 있어야 한다.

핵심 006 DBMS의 장·단점

장 점	단 점
<ul style="list-style-type: none"> • 데이터의 중복을 피할 수 있음 • 저장된 자료를 공동으로 이용할 수 있음 • 데이터의 일관성을 유지할 수 있음 • 데이터의 무결성을 유지할 수 있음 • 보안을 유지할 수 있음 • 데이터를 표준화할 수 있음 • 데이터를 통합하여 관리할 수 있음 • 항상 최신의 데이터를 유지함 • 데이터의 실시간 처리가 가능함 • 데이터의 논리적·물리적 독립성이 보장 	<ul style="list-style-type: none"> • 데이터베이스 전문가 부족 • 전산화 비용이 증가함 • 대용량 디스크로의 집중적인 Access로 과부하(Overhead)가 발생함 • 파일의 예비(Backup)와 회복(Recovery)이 어려움 • 시스템이 복잡함

논리적 독립성과 물리적 독립성

- 논리적 독립성 : 응용 프로그램과 데이터베이스를 독립 시킴으로써, 데이터의 논리적 구조를 변경시키더라도 응용 프로그램은 변경되지 않음
- 물리적 독립성 : 응용 프로그램과 보조기억장치 같은 물리적 장치를 독립시킴으로써, 데이터베이스 시스템의 성능 향상을 위해 새로운 디스크를 도입하더라도 응용 프로그램에는 영향을 주지 않고 데이터의 물리적 구조만을 변경함

핵심 007 스키마(Schema)의 정의

- 데이터베이스의 구조와 제약조건에 관한 전반적인 명세(Specification)를 기술(Description)한다.
- 데이터베이스를 구성하는 데이터 개체(Entity), 속성(Attribute), 관계(Relationship) 및 데이터 조작 시 데이터 값들이 갖는 제약조건 등에 관해 전반적으로 정의한다.
- 스키마는 사용자의 관점에 따라 외부(External) 스키마, 개념(Conceptual) 스키마, 내부(Internal) 스키마로 나뉜다.
- 스키마(Schema)는 데이터 사전에 저장되며, 다른 이름으로 메타 데이터(Meta-Data)라고도 한다.



핵심 18.3, 17.3, 16.5, 15.3, 14.8, 14.5, 14.3, 13.8, 13.6, 12.8, 12.5, 12.3, 11.3, 10.9, 10.5, 10.3, 09.8, 06.9, ...

008 스키마의 3계층

외부 스키마(External Schema) = 서브 스키마 = 사용자 뷰(View)

- 사용자나 응용 프로그래머가 각 개인의 입장에서 필요로 하는 데이터베이스의 논리적 구조를 정의한다.
- 전체 데이터베이스의 한 논리적인 부분으로 볼 수 있으므로 서브 스키마(Subschema)라고도 한다.
- 하나의 데이터베이스 시스템에는 여러 개의 외부 스키마가 존재할 수 있으며, 하나의 외부 스키마를 여러 개의 응용 프로그램이나 사용자가 공유할 수 있다.
- 같은 데이터베이스에 대해서도 서로 다른 관점을 정의할 수 있도록 허용한다.
- 일반 사용자는 질의어(SQL)를 사용하여 DB를 사용한다.

개념 스키마(Conceptual Schema) = 전체적인 뷰(View)

- 데이터베이스의 전체적인 논리적 구조로서, 모든 응용 프로그램이나 사용자들이 필요로 하는 데이터를 통합한 조직 전체의 데이터베이스로 하나만 존재한다.
- 개념 스키마는 개체 간의 관계와 제약조건을 나타내고 데이터베이스의 접근 권한, 보안 및 무결성 규칙에 관한 명세를 정의한다.
- 단순히 스키마(Schema)라고 하면 개념 스키마를 의미한다.
- 기관이나 조직체의 관점에서 데이터베이스를 정의한 것이다.
- 데이터베이스 관리자에 의해서 구성된다.

내부 스키마(Internal Schema)

- 물리적 저장장치의 입장에서 본 데이터베이스 구조로, 물리적인 저장장치와 밀접한 계층이다.
- 실제로 데이터베이스에 저장될 레코드의 물리적인 구조를 정의하고, 저장 데이터 항목의 표현 방법, 내부 레코드의 물리적 순서 등을 나타낸다.
- 시스템 프로그래머나 시스템 설계자가 보는 관점의 스키마이다.
- 데이터베이스의 물리적 구조를 정의한다.
- 데이터의 실제 저장 방법을 기술한다.
- 물리적인 저장장치와 밀접한 계층이다.

핵심 08.9, 07.9, 07.3, 06.9, 06.5, 05.3, 03.8, 03.5, 02.9, 02.3, 01.9, 01.6, 01.3, 99.8

009 데이터베이스 언어(Database Language)

데이터 정의 언어(DDL ; Data Definition Language)

- DB 구조, 데이터 형식, 접근 방식 등 DB를 구축하거나 수정할 목적으로 사용하는 언어이다.
- 번역한 결과가 데이터 사전(Data-Dictionary)이라는 특별한 파일에 여러 개의 테이블로 저장된다.
- 데이터 정의 언어의 기능
 - 외부 스키마 명세 정의
 - 데이터베이스의 논리적 데이터 구조와 물리적 데이터 구조의 정의 및 수정
 - 논리적 데이터 구조와 물리적 데이터 구조 간의 사상 정의
 - 스키마에 사용되는 제약조건에 대한 명세 정의
 - 데이터의 물리적 순서 규정

데이터 조작 언어(DML ; Data Manipulation Language) = 서브 언어

- 사용자로 하여금 데이터를 처리할 수 있게 하는 도구로서 사용자(응용 프로그램)와 DBMS 간의 인터페이스를 제공한다.
- 응용 프로그램을 통하여 사용자가 DB의 데이터를 실질적으로 조작할 수 있도록 하기 위해 C, COBOL 등의 호스트 언어에 DB 기능을 추가시켜 만든 언어이다.
- 대표적인 데이터 조작용어(DML)에는 질의어가 있으며, 질의어는 터미널에서 주로 이용하는 비절차적(Non Procedural) 데이터 언어이다.

데이터 제어 언어(DCL ; Data Control Language)

- 무결성, 보안 및 권한 제어, 회복 등을 하기 위한 언어이다.
- 데이터를 보호하고 데이터를 관리하는 목적으로 사용된다.
- 데이터 제어 언어의 기능
 - 불법적인 사용자로부터 데이터를 보호하기 위한 데이터 보안(Security)
 - 데이터의 정확성을 위한 무결성(Integrity) 유지
 - 시스템 장애에 대비한 데이터 회복과 병행수행 제어



010 데이터베이스 사용자

DBA(DataBase Administrator)

데이터베이스 시스템의 모든 관리와 운영에 대한 책임을 지고 있는 사람이나 그룹을 의미한다.

- 데이터베이스 구성 요소 결정
- 개념 스키마 및 내부 스키마 정의
- 데이터베이스의 저장 구조 및 접근 방법 정의
- 보안 및 데이터베이스의 접근 권한 부여 정책 수립
- 장애에 대비한 예비(Back Up) 조치와 회복(Recovery)에 대한 전략 수립
- 무결성을 위한 제약조건의 지정
- 데이터 사전의 구성과 유지관리
- 사용자의 요구와 불평의 청취 및 해결
- 변화 요구에 대한 적응과 성능 향상에 대한 감시
- 시스템 감시 및 성능 분석
- 데이터 사용 추세, 이용 형태 및 각종 통계 등을 종합, 분석

응용 프로그래머

- 응용 프로그래머는 일반 호스트 언어로 프로그램을 작성할 때 데이터 조작어를 삽입해서 일반 사용자가 응용 프로그램을 사용할 수 있게, 인터페이스를 제공할 목적으로 데이터베이스를 접근하는 사람들이다.
- 응용 프로그래머는 C, COBOL, PASCAL 등의 호스트 언어와 DBMS가 지원하는 데이터 조작어에 능숙한 컴퓨터 전문가이다.

일반 사용자

일반 사용자는 보통 터미널을 이용하여 데이터베이스에 있는 자료를 활용할 목적으로 질의어나 응용 프로그램을 사용하여 데이터베이스에 접근하는 사람들이다.

011 데이터 모델에 표시할 사항

- 구조(Structure) : 논리적으로 표현된 개체 타입들 간의 관계로서 데이터 구조 및 정적 성질을 표현함
- 연산(Operation) : 데이터베이스에 저장된 실제 데이터를 처리하는 방법을 표시하는 것으로서 데이터베이스를 조작하는 기본 도구임

- 제약조건(Constraint) : 데이터베이스에 저장될 수 있는 실제 데이터의 논리적인 제약조건을 표시함

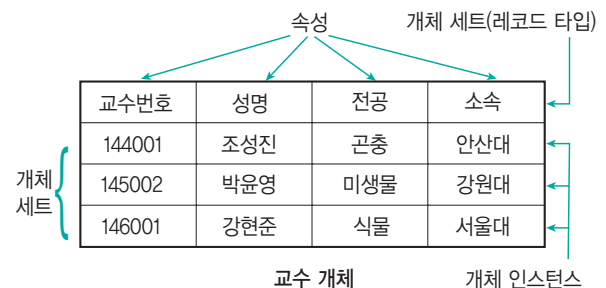
012 데이터 모델의 구성 요소

개체(Entity)

- 데이터베이스에 표현하려는 것으로, 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상체이다.
- 유형, 무형의 정보로서 서로 연관된 몇 개의 속성으로 구성된다.
- 파일 시스템의 레코드에 대응하는 것으로, 어떤 정보를 제공하는 역할을 수행한다.
- 실세계에 독립적으로 존재하거나 그 자체로서도 구별이 가능하다.

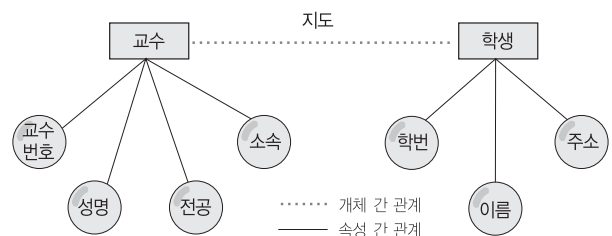
속성(Attribute)

- 데이터의 가장 작은 논리적 단위로서 파일 구조의 데이터 항목 또는 데이터 필드에 해당된다.
- 개체를 구성하는 항목이다.



관계(Relationship)

- 개체 간의 관계 또는 속성 간의 관계
- 다음 그림의 관계는 교수가 학생을 지도하는 관계이다.

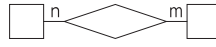



교수 개체의 구성 요소

- 속성 : 개체가 가지고 있는 특성, 교수번호, 성명, 전공, 소속



- 개체 타입 : 속성으로만 기술된 개체의 정의
- 개체 인스턴스 : 개체를 구성하고 있는 각 속성들이 값을 가져 하나의 개체를 나타내는 것으로 개체 어커런스(Entity Occurrence) 라고도 함
- 개체 세트 : 개체 인스턴스의 집합

	관계	1:1, 1:N, N:M 등의 개체 관계에 대해 선 위에 대응수 기술
	선, 링크	개체 타입과 속성 연결

핵심 015 계층형 데이터 모델

- 데이터의 논리적 구조도가 트리 형태이며, 개체가 트리를 구성하는 노드 역할을 한다.
- 개체 집합에 대한 속성 관계를 표시하기 위해 개체를 노드로 표현하고 개체 집합들 사이의 관계를 링크로 연결한다.
- 개체 간의 관계를 부모와 자식 간의 관계로 표현한다.
- 개체 타입 간에는 상위와 하위 관계가 존재하며, 일 대 다(1:N) 대응 관계만 존재한다.
- 레코드 삭제 시 연쇄 삭제(Triggered Delete)가 된다.
- 개체 타입들 간에는 사이클(Cycle)이 허용되지 않는다.
- 계층형 모델에서는 개체(Entity)를 세그먼트(Segment)라 부른다.
- 대표적인 DBMS는 IMS이다.

핵심 016 망(그래프, 네트워크)형 데이터 모델

- CODASYL이 제안한 것으로, CODASYL DBTG 모델이라고도 한다.
- 그래프를 이용해서 데이터 논리 구조를 표현한 데이터 모델이다.
- 상위와 하위 레코드 사이에서 다 대 다(N:M) 대응 관계를 만족하는 구조이다.
- 상위의 레코드를 Owner, 하위의 레코드를 Member라 하여 Owner-Member 관계라고도 한다.
- 레코드 타입 간의 관계는 1:1, 1:N, N:M이 될 수 있다.
- 대표적인 DBMS : DBTG, EDBS, TOTAL 등

핵심 013 개체-관계(Entity-Relationship) 모델

- 개념적 데이터 모델의 가장 대표적인 것으로, 1976년 Peter Chen에 의해 제안되었다.
- 개체 타입(Entity Type)과 이들 간의 관계 타입(Relationship Type)을 이용해 현실 세계를 개념적으로 표현한다.
- 데이터를 개체(Entity), 관계(Relationship), 속성(Attribute)으로 묘사한다.
- E-R 다이어그램으로 표현한다.
- 특정 DBMS를 고려한 것이 아니기 때문에 관계 표현에 제한이 없다.

핵심 014 E-R 다이어그램

- E-R 모델의 기본적인 아이디어를 시각적으로 표현하기 위한 도구이다.
- 개체 간의 관계는 물론 시스템 내의 역할을 하는 모든 개체들, 즉 조직, 부서, 사용자, 프로그램, 데이터를 모두 표시한다.

기 호	기호 이름	의 미
	사각형	개체(Entity) 타입
	마름모	관계(Relationship) 타입
	타원	속성(Attribute)
	이중 타원	다중값 속성
	밑줄 타원	기본키 속성
	복수 타원	복합 속성 예) 성명은 성과 이름으로 구성



핵심 18.4, 17.8, 17.5, 17.3, 15.8, 15.3, 14.8, 14.3, 13.8, 13.6, 12.5, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, ...

017 데이터베이스 설계

데이터베이스 설계 시 고려사항

- 데이터의 무결성 유지 : 삽입, 삭제, 갱신 등의 연산 후에도 데이터베이스에 저장된 데이터가 정해진 제약조건을 항상 만족해야 함
- 데이터의 일관성 유지 : 데이터베이스에 저장된 데이터들 사이나, 특정 질의에 대한 응답이 처음부터 끝까지 변함없이 일정해야 함
- 데이터의 회복성 유지 : 시스템에 장애가 발생했을 때 장애 발생 직전의 상태로 복구할 수 있어야 함
- 데이터의 보안성 유지 : 불법적인 데이터의 노출 또는 변경이나 손실로부터 보호할 수 있어야 함
- 데이터의 효율성 유지 : 응답시간의 단축, 시스템의 생산성, 저장 공간의 최적화 등이 가능해야 함
- 데이터베이스의 확장성 유지 : 데이터베이스 운영에 영향을 주지 않으면서 지속적으로 데이터를 추가할 수 있어야 함

개념적 설계(정보 모델링, 개념화)

- 정보의 구조를 얻기 위하여 현실 세계의 무한성과 계속성을 이해하고, 다른 사람과 통신하기 위하여 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정이다.
- 스키마 모델링과 트랜잭션 모델링을 병행하여 수행한다.
- 요구 분석 단계에서 나온 결과(요구 조건 명세)를 DBMS에 독립적인 E-R 다이어그램(개체 관계도)으로 작성한다.
- DBMS에 독립적인 개념 스키마를 설계한다.

논리적 설계(데이터 모델링)

- 현실 세계에서 발생하는 자료를 컴퓨터가 처리할 수 있는 물리적 저장장치에 저장할 수 있도록 변환하기 위해 특정 DBMS가 지원하는 논리적 자료 구조로 변환시키는 과정이다.
- 개념 세계의 데이터를 필드로 기술된 데이터 타입과 이 데이터 타입들 간의 관계로 표현되는 논리적 구조의 데이터로 모델화한다.
- 개념적 설계가 개념 스키마를 설계하는 단계라면 논리적 설계에서는 개념 스키마를 평가 및 정제하고 특정 DBMS에 종속적인 논리적 스키마를 설계하는 단계이다.
- 트랜잭션의 인터페이스를 설계한다.
- 관계형 데이터베이스라면 테이블을 설계하는 단계이다.

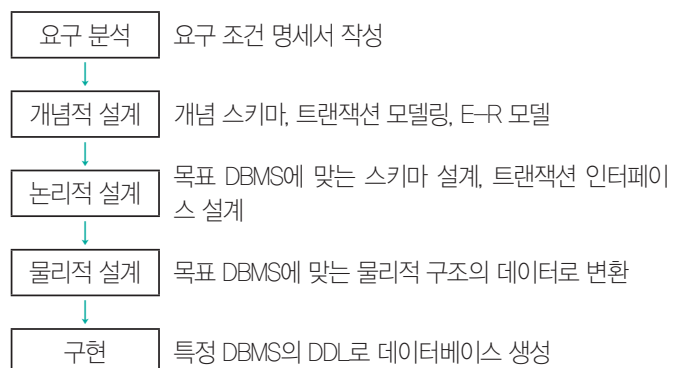
물리적 설계(데이터 구조화)

- 논리적 설계 단계에서 논리적 구조로 표현된 데이터를 디스크 등의 물리적 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정이다.
- 데이터베이스 파일의 저장 구조, 레코드의 형식, 접근 경로와 같은 정보를 사용하여 데이터가 컴퓨터에 저장되는 방법을 묘사한다.
- 트랜잭션을 작성한다.
- 물리적 설계 단계에 꼭 포함되어야 할 것은 저장 레코드의 양식 설계, 레코드 집중의 분석 및 설계, 접근 경로 등이다.
- 물리적 설계 시 고려사항
 - 인덱스의 구조
 - 레코드의 크기 및 개수
 - 파일에 대한 트랜잭션의 갱신과 참조 성향
 - 성능 향상을 위한 개념 스키마의 변경 여부 검토
 - 빈번한 질의와 트랜잭션들의 수행속도를 높이기 위한 고려
 - 시스템 운용 시 파일 크기의 변화 가능성
- 물리적 설계 옵션 선택 시 고려 사항
 - 반응시간(Response Time) : 트랜잭션 수행을 요구한 시점부터 처리 결과를 얻을 때까지의 경과시간
 - 공간 활용도(Space Utilization) : 데이터베이스 파일과 액세스 경로 구조에 의해 사용되는 저장공간의 양
 - 트랜잭션 처리량(Transaction Throughput) : 단위 시간 동안 데이터베이스 시스템에 의해 처리될 수 있는 트랜잭션의 평균 개수

핵심

16.3, 15.5, 14.8, 14.5, 13.6, 13.3, 12.5, 12.3, 11.6, 09.5, 08.9, 06.9, 06.5, 05.9, 05.5, 05.4, 04.9, 04.5, ...

018 데이터베이스 설계 순서



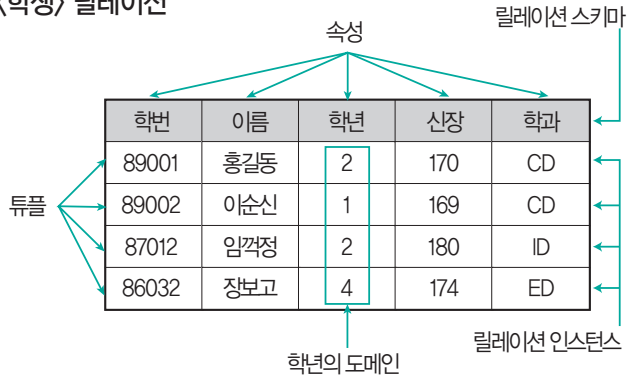


핵심 18.4, 18.3, 16.5, 16.3, 15.8, 15.5, 14.8, 14.3, 12.5, 12.3, 11.8, 09.5, 08.5, ...

019 관계 데이터베이스의 Relation 구조

릴레이션은 데이터들을 표(Table)의 형태로 표현한 것으로, 구조를 나타내는 릴레이션 스키마와 실제 값들인 릴레이션 인스턴스로 구성된다.

〈학생〉 릴레이션



튜플(Tuple)

- 릴레이션을 구성하는 각각의 행
- 속성의 모임으로 구성된다.
- 파일 구조에서 레코드와 같은 의미이다.
- 튜플의 수 = 카디널리티(Cardinality) = 기수 = 대응수

속성(Attribute, 애트리뷰트)

- 릴레이션을 구성하는 각각의 열
- 데이터베이스를 구성하는 가장 작은 논리적 단위이다.
- 파일 구조 상의 데이터 항목 또는 데이터 필드에 해당된다.
- 개체의 특성을 기술한다.
- 속성의 수 = 디그리(Degree) = 차수

도메인(Domain)

- 하나의 애트리뷰트가 취할 수 있는 같은 타입의 원자(Atomic)값들의 집합
- 실제 애트리뷰트 값이 나타날 때 그 값의 합법 여부를 시스템이 검사하는 데에도 이용된다.
- 예 성별 애트리뷰트의 도메인은 '남'과 '여'로, 그 외의 값은 입력될 수 없다.

릴레이션 인스턴스(Relation Instance)

데이터 개체를 구성하고 있는 속성들에 데이터 타입이 정의되어 구체적인 데이터 값을 갖고 있는 것을 말한다.

핵심 17.5, 17.3, 16.3, 15.8, 15.5, 15.3, 14.5, 14.3, 13.8, 13.6, 12.8, 12.5, 12.3, ...

020 릴레이선의 특징

〈학생〉 릴레이션

학번	이름	학년	신장	학과
89001	홍길동	2	170	CD
89002	이순신	1	169	CD
87012	임꺽정	2	180	ID
86032	장보고	4	174	ED

- 한 릴레이션에 포함된 튜플들은 모두 상이하다.
- 예 〈학생〉 릴레이션을 구성하는 홍길동 레코드는 홍길동에 대한 학적사항을 나타내는 것으로 〈학생〉 릴레이션 내에서는 유일하다.
- 한 릴레이션에 포함된 튜플 사이에는 순서가 없다.
- 예 〈학생〉 릴레이션에서 홍길동 레코드와 임꺽정 레코드의 위치가 바뀌어도 상관없다.
- 튜플들의 삽입, 삭제 등의 작업으로 인해 릴레이션은 시간에 따라 변한다.
- 예 〈학생〉 릴레이션에 새로운 학생의 레코드를 삽입하거나 기존 학생에 대한 레코드를 삭제함으로써 테이블은 내용 면에서나 크기 면에서 변하게 된다.
- 릴레이션 스키마를 구성하는 속성들 간의 순서는 중요하지 않다.
- 예 학번, 이름 등의 속성을 나열하는 순서가 이름, 학번 순으로 바뀌어도 데이터 처리에는 아무런 영향을 미치지 않는다.
- 속성의 유일한 식별을 위해 속성의 명칭은 유일해야 하지만, 속성을 구성하는 값은 동일한 값이 있을 수 있다.
- 예 각 학생의 학년을 기술하는 속성인 '학년'은 다른 속성명들과 구분되어 유일해야 하지만 '학년' 속성에는 2, 1, 2, 4 등이 입력된 것처럼 동일한 값이 있을 수 있다.
- 릴레이션을 구성하는 튜플을 유일하게 식별하기 위해 속성들의 부분집합을 키(Key)로 설정한다.
- 예 〈학생〉 릴레이션에서는 '학번'이나 '이름'이 튜플들을 구분하는 유일한 값인 키가 될 수 있다.
- 속성은 더 이상 쪼갤 수 없는 원자값만을 저장한다.
- 예 '학년'에 저장된 1, 2, 4 등은 더 이상 세분화할 수 없다.



핵심 18.4, 16.5, 15.8, 15.5, 14.5, 13.8, 13.3, 12.8, 12.3, 11.3, 10.9, 10.5, ...

021 키(Key)의 개념 및 종류

키(Key)는 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 다른 튜플들과 구별할 수 있는 유일한 기준이 되는 애트리뷰트(속성)이다.

〈학생〉 릴레이션

학번	주민번호	성명	성별
1001	810429-1231457	김형석	남
1002	800504-1546781	김현천	남
1003	811215-2547842	류기선	여
1004	801225-2201248	홍영선	여

〈수강〉 릴레이션

학번	과목명
1001	영어
1001	전산
1002	영어
1003	수학
1004	영어
1004	전산

후보키 (Candidate Key)	<ul style="list-style-type: none"> 릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별하기 위해 사용하는 속성들의 부분집합, 즉 기본키로 사용할 수 있는 속성들을 말함 모든 릴레이션은 반드시 하나 이상의 후보키를 가져야 함 릴레이션에 있는 모든 튜플에 대해서 유일성과 최소성을 만족시켜야 함 <p>예 〈학생〉 릴레이션에서 '학번'이나 '주민번호'는 다른 레코드를 유일하게 구별할 수 있는 기본키로 사용할 수 있으므로 후보키이다.</p>
기본키 (Primary Key)	<ul style="list-style-type: none"> 후보키 중에서 선택한 주키(Main Key) 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성 Null 값을 가질 수 없음 기본키로 정의된 속성에는 동일한 값이 중복되어 저장될 수 없음 <p>예 〈학생〉 릴레이션에서는 '학번'이나 '주민번호'가 기본키가 될 수 있고, 〈수강〉 릴레이션에서는 '학번'+'과목명'으로 조합해야 기본키가 만들어진다.</p> <p>예 '학번'이 〈학생〉 릴레이션의 기본키로 정의되면 이미 입력된 '1001'은 다른 튜플의 '학번' 속성의 값으로 입력할 수 없다.</p>

대체키 (Alternate Key)	<ul style="list-style-type: none"> 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키들을 말함 보조키라고도 함 <p>예 〈학생〉 릴레이션에서 '학번'을 기본키로 정의하면 '주민번호'는 대체키가 된다.</p>
슈퍼키 (Super Key)	<ul style="list-style-type: none"> 슈퍼키는 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키로서 릴레이션을 구성하는 모든 튜플 중 슈퍼키로 구성된 속성의 집합과 동일한 값은 나타나지 않는다. 릴레이션을 구성하는 모든 튜플에 대해 유일성은 만족시키지만, 최소성은 만족시키지 못함 <p>예 〈학생〉 릴레이션에서는 '학번', '주민번호', '학번'+'주민번호', '주민번호'+'성명', '학번'+'주민번호'+'성명' 등으로 슈퍼키를 구성할 수 있다.</p>
외래키 (Foreign Key)	<ul style="list-style-type: none"> 관계(Relationship)를 맺고 있는 릴레이션 R1, R2에서 릴레이션 R1이 참조하고 있는 릴레이션 R2의 기본키와 같은 R1 릴레이션의 속성 외래키는 참조되는 릴레이션의 기본키와 대응되어 릴레이션 간에 참조 관계를 표현하는데 중요한 도구임 외래키로 지정되면 참조 테이블의 기본키에 없는 값은 입력할 수 없음 <p>예 〈수강〉 릴레이션이 〈학생〉 릴레이션을 참조하고 있으므로 〈학생〉 릴레이션의 '학번'은 기본키이고, 〈수강〉 릴레이션의 '학번'은 외래키이다.</p> <p>예 〈수강〉 릴레이션의 '학번'에는 〈학생〉 릴레이션의 '학번'에 없는 값은 입력할 수 없다.</p>

잠깐만요 !

널 값(NULL Value)

데이터베이스에서 아직 알려지지 않았거나 모르는 값으로서 '해당 없음' 등의 이유로 정보 부재를 나타내기 위해 사용하는, 이론적으로 아무것도 없는 특수한 데이터입니다.

최소성과 유일성

'학번'+'주민번호'를 사용하여 슈퍼키를 만들면 다른 튜플들과 구분할 수 있는 유일성은 만족하지만, '학번'이나 '주민번호' 하나만 가지고도 다른 튜플들을 구분할 수 있으므로 최소성은 만족시키지 못합니다.

핵심 18.4, 18.3, 17.5, 17.3, 16.8, 15.3, 13.6, 11.6, 10.3, 08.5, 08.3, 07.9, ...

022 무결성(Integrity)

- 개체 무결성 : 릴레이션에서 기본키를 구성하는 속성은 널(NULL) 값이나 중복값을 가질 수 없음
- 예 〈학생〉 릴레이션에서 '학번'이 기본키로 정의되면 튜플을 추가할 때 '주민번호'나 '성명' 필드에는 값을 입력하지 않아도 되지만 '학번' 속성에는 반드시 값을 입력해야 한다. 또한 '학번' 속성에는 이미 한 번 입력한 속성값을 중복하여 입력할 수 없다.



- 참조 무결성 : 외래키 값은 NULL이거나 참조 릴레이션의 기본키 값과 동일해야 함, 즉 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없음

예 <수강> 릴레이션의 '학번' 속성에는 <학생> 릴레이션의 '학번' 속성에 없는 값은 입력할 수 없다.

- 도메인 무결성 : 특정 속성의 값이 그 속성이 정의된 도메인에 속한 값이어야 한다는 규정

예 성별 속성의 도메인은 '남'과 '여'로, 그 외의 값은 입력할 수 없다.

Division	$X \supset Y$ 인 2개의 릴레이션 $R(X)$ 와 $S(Y)$ 가 있을 때, R 의 속성이 S 의 속성값을 모두 가진 튜플에서 S 가 가진 속성을 제외(분리)한 속성만을 구하는 연산
----------	---

핵심 17.8, 17.5, 15.5, 15.3, 14.3, 13.6, 12.5, 10.5, 10.3, 09.8, 08.9, 06.5, 05.5, 04.9, 02.9, 00.3, 99.4

025 관계해석

- 코드(E. F. Codd)가 수학의 Predicate Calculus(술어 해석)에 기반을 두고 관계 데이터베이스를 위해 제안했다.
- 관계해석은 원하는 정보가 무엇이라는 것만 정의하는 비절차적 특성을 지닌다.
- 원하는 정보를 정의할 때는 계산 수식을 사용한다.
- 튜플 관계해석과 도메인 관계해석이 있다.
- 기본적으로 관계해석과 관계대수는 관계 데이터베이스를 처리하는 기능과 능력 면에서 동등하다.
- 질의어로 표현한다.

핵심 18.3, 17.8, 16.5, 16.3, 15.3, 14.5, 13.8, 13.3, 12.5, 12.3, 11.8, 10.9, 10.5, 09.8, 09.3, 08.5, 07.5, 06.5, ...

026 정규화(Normalization)

정규화의 개요

- 함수적 종속성 등의 종속성 이론을 이용하여 잘못 설계된 관계형 스키마를 더 작은 속성의 세트로 쪼개어 바람직한 스키마로 만들어 가는 과정이다.
- 정규형에는 제1정규형, 제2정규형, 제3정규형, BCNF형, 제4정규형, 제5정규형이 있으며, 차수가 높아질수록 만족시켜야 할 제약 조건이 늘어난다.
- 정규화는 데이터베이스의 논리적 설계 단계에서 수행한다.
- 정규화는 논리적 처리 및 품질에 큰 영향을 미친다.

정규화의 목적

- 데이터 구조의 안정성을 최대화한다.
- 어떠한 릴레이션이라도 데이터베이스 내에서 표현 가능하게 만든다.
- 효과적인 검색 알고리즘을 생성할 수 있다.
- 중복을 배제하여 삽입, 삭제, 갱신 이상의 발생을 방지한다.
- 데이터 삽입 시 릴레이션을 재구성할 필요성을 줄인다.

023 관계대수의 개요

핵심 18.4, 18.3, 15.8, 14.5, 11.8, 11.6, 11.3, 10.3, 09.8

- 관계형 데이터베이스에서 원하는 정보와 그 정보를 어떻게 유도하는가를 기술하는 절차적인 언어이다.
- 릴레이션을 처리하기 위해 연산자와 연산규칙을 제공하는 언어로 피연산자가 릴레이션이고, 결과도 릴레이션이다.
- 질의에 대한 해를 구하기 위해 수행해야 할 연산의 순서를 명시한다.
- 순수 관계 연산자와 일반 집합 연산자가 있다.
- 순수 관계 연산자 : Select, Project, Join, Division
- 일반 집합 연산자 : UNION(합집합), INTERSECTION(교집합), DIFFERENCE(차집합), Cartesian Product(교차곱)

핵심 17.5, 17.3, 16.8, 16.3, 14.8, 08.5, 07.5, 07.3, 05.3, 04.5, 03.8, 02.9, ...

024 순수 관계 연산자

관계 데이터베이스에 적용할 수 있도록 특별히 개발된 관계 연산자이다.

연산자	특징
Select	<ul style="list-style-type: none"> 릴레이션에 존재하는 튜플 중에서 선택 조건을 만족하는 튜플의 부분집합을 구하여 새로운 릴레이션을 만들 릴레이션의 행(가로)에 해당하는 튜플을 구하는 것이므로 수평 연산이라고도 함 연산자의 기호는 그리스 문자 시그마(σ)를 사용함
Project	<ul style="list-style-type: none"> 주어진 릴레이션에서 속성 List에 제시된 Attribute만을 추출하는 연산 릴레이션의 열(세로)에 해당하는 Attribute를 추출하는 것이므로 수직 연산자라고도 함 연산자의 기호는 그리스 문자 파이(π)를 사용함
Join	<ul style="list-style-type: none"> 공통 속성을 중심으로 2개의 릴레이션을 하나로 합쳐서 새로운 릴레이션을 만드는 연산 연산자의 기호는 \bowtie를 사용함



027 Anomaly(이상)의 개념 및 종류

이상(Anomaly)의 개념

- 정규화(Normalization)를 거치지 않은 데이터베이스 내에 데이터들이 불필요하게 중복되어 릴레이션 조작 시 발생하는 예기치 못한 곤란한 현상이다.
- 애트리뷰트들 간에 존재하는 여러 종속 관계를 하나의 릴레이션에 표현하기 때문에 이상이 발생한다.

이상의 종류

삽입 이상 (Insertion Anomaly)	릴레이션에 데이터를 삽입할 때 의도와는 관계없이 원하지 않은 값들도 함께 삽입되는 현상
삭제 이상 (Deletion Anomaly)	릴레이션에서 한 튜플을 삭제할 때 의도와는 관계없는 값들도 함께 삭제되는 연쇄 삭제 현상
갱신 이상 (Update Anomaly)	릴레이션에서 튜플에 있는 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상

028 정규화 과정

비정규 릴레이션
↓ 도메인이 원자값
1NF
↓ 부분적 함수 종속 제거
2NF
↓ 이행적 함수 종속 제거
3NF
↓ 결정자이면서 후보키가 아닌 것 제거
BCNF
↓ 다치 종속
4NF
↓ 조인 종속성 이용
5NF

정규화 단계 암기 요령

정규화라는 출소자가 말했다.

두부이겨다줘 ≡ 도부이결다조

도메인이 원자값

부분적 함수 종속 제거

이행적 함수 종속 제거

결정자이면서 후보키가 아닌 것 제거

다치 종속

조인 종속성 이용

함수적 종속 관계

어떤 릴레이션 R에서 X와 Y를 각각 R의 애트리뷰트 집합의 부분 집합이라고 할 경우, 애트리뷰트 X의 값 각각에 대해 시간에 관계없이 항상 애트리뷰트 Y의 값이 오직 하나만 연관되어 있을 때 Y는 X에 함수 종속적이라고 하며, $X \rightarrow Y$ 와 같이 표기한다.

예 <수강> 릴레이션이 (학번, 이름, 과목명)으로 되어 있을 때, '학번'이 결정되면 '과목명'에 상관없이 '학번'에는 항상 같은 이름이 대응된다. '학번'에 따라 '이름'이 결정될 때 '이름'을 '학번'에 함수 종속적이라고 하며 '학번 \rightarrow 이름'과 같이 표기한다.

이행적 종속 관계

$A \rightarrow B$ 이고 $B \rightarrow C$ 일 때 $A \rightarrow C$ 를 만족하는 관계이다.

029 SQL의 분류

DDL(데이터 정의어)

- SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의하거나 변경 또는 삭제할 때 사용하는 언어이다.
- 데이터베이스 관리자나 데이터베이스 설계자가 사용한다.
- 데이터 정의어(DDL)의 3가지 유형

명령어	기능
CREATE	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의함
ALTER	TABLE에 대한 정의를 변경하는 데 사용함
DROP	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 삭제함



DML(데이터 조작어)

- 데이터베이스 사용자가 응용 프로그램이나 질의어를 통하여 저장된 데이터를 실질적으로 처리하는 데 사용하는 언어이다.
- 데이터베이스 사용자와 데이터베이스 관리 시스템 간의 인터페이스를 제공한다.
- 데이터 조작어(DML)의 4가지 유형

명령어	기능
SELECT	테이블에서 조건에 맞는 튜플을 검색함
INSERT	테이블에 새로운 튜플을 삽입함
DELETE	테이블에서 조건에 맞는 튜플을 삭제함
UPDATE	테이블에서 조건에 맞는 튜플의 내용을 변경함

DCL(데이터 제어어)

- 데이터의 보안, 무결성, 데이터 회복, 병행수행 제어 등을 정의하는 데 사용하는 언어이다.
- 데이터베이스 관리자가 데이터 관리를 목적으로 사용한다.

데이터 제어어(DCL)의 종류

명령어	기능
COMMIT	명령에 의해 수행된 결과를 실제 물리적 디스크로 저장하고, 데이터베이스 조작 작업이 정상적으로 완료되었음을 관리자에게 알려줌
ROLLBACK	데이터베이스 조작 작업이 비정상적으로 종료되었을 때 원래의 상태로 복구함
GRANT	데이터베이스 사용자에게 사용 권한을 부여함
REVOKE	데이터베이스 사용자의 사용 권한을 취소함

핵심 18.4, 17.3, 16.8, 16.5, 16.3, 14.3, 09.5, 08.5, 08.3, 05.9, 05.3, 04.9, ...

030 Select문

테이블을 구성하는 튜플(행)들 중에서 전체 또는 조건을 만족하는 튜플(행)을 검색하여 주기억장치 상에 임시 테이블로 구성시키는 명령문이다.

```
SELECT Predicate [테이블명].속성명1, [테이블명].속성명2, ...
FROM 테이블명1, 테이블명2, ...
[WHERE 조건]
[GROUP BY 속성명1, 속성명2, ...]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];
```

1. SELECT절

- Predicate : 불리올 튜플 수를 제한할 명령어를 기술함
 - ALL : 모든 튜플을 검색할 때 지정하는 것으로, 주로 생략함
 - DISTINCT : 중복된 튜플이 있으면 그 중 첫 번째만 검색함
 - DISTINCTROW : 중복된 튜플을 검색하지만 선택된 속성의 값이 아닌, 튜플 전체를 대상으로 함
- 속성명 : 검색하여 불리올 속성(열) 및 수식들을 지정함
 - 기본 테이블을 구성하는 모든 속성을 지정할 때는 '*'를 기술한다.
 - 두 개 이상의 테이블을 대상으로 검색할 때는 반드시 테이블명.속성명으로 표현해야 한다.

2. FROM절 : 질의에 의해 검색될 데이터들을 포함하는 테이블명을 기술함

3. WHERE절 : 검색할 조건 기술

잠깐만요 !

IN / BETWEEN 연산자의 사용

- 직위가 '과장', '팀장', '사원'인 자료만 검색
예 WHERE 직위 IN('과장', '팀장', '사원')
- 생일이 '01/09/69'에서 '10/22/73' 사이인 자료만 검색
예 WHERE 생일 BETWEEN #01/09/69# AND #10/22/73#

NULL 값의 사용

- 주소가 NULL인, 즉 주소가 입력되지 않은 자료만 검색
예 WHERE 주소 IS NULL
- 주소가 NULL이 아닌, 즉 주소가 입력된 자료만 검색
예 WHERE 주소 IS NOT NULL

4. GROUP BY절

- 특정 속성을 기준으로 그룹화하여 검색할 때 그룹화할 속성을 지정함
- 일반적으로 GROUP BY절은 그룹 함수와 함께 사용된다.
- 그룹 함수의 종류

- COUNT(속성명) : 그룹별 튜플 수를 구하는 함수
- MAX(속성명) : 그룹별 최대값을 구하는 함수
- MIN(속성명) : 그룹별 최소값을 구하는 함수
- SUM(속성명) : 그룹별 합계를 구하는 함수
- AVG(속성명) : 그룹별 평균을 구하는 함수

5. HAVING절 : GROUP BY와 함께 사용되며, 그룹에 대한 조건을 지정함

6. ORDER BY절 : 특정 속성을 기준으로 정렬하여 검색할 때 사용함



- 속성명 : 정렬의 기준이 되는 속성명을 기술함
- [ASC|DESC] : 정렬 방식으로서 'ASC'는 오름차순, 'DESC'는 내림차순임, 생략하면 오름차순으로 지정

핵심 031 삽입, 삭제, 갱신문

삽입문(INSERT INTO ~)

- 기본 테이블에 새로운 튜플을 삽입할 때 사용한다.

```
INSERT
INTO 테이블명(속성명1, 속성명2, ...)
VALUES (데이터1, 데이터2, ...);
```

- 대응하는 속성과 데이터는 개수와 데이터 형식이 일치해야 한다.
- 기본 테이블의 모든 속성을 사용할 때는 속성명을 생략할 수 있다.
- SELECT문을 사용하여 다른 테이블의 검색 결과를 삽입할 수 있다.

삭제문(DELETE FROM ~)

- 기본 테이블에 있는 튜플들 중에서 특정 튜플을 삭제할 때 사용한다.

```
DELETE
FROM 테이블명
WHERE 조건;
```

- 모든 레코드를 삭제할 때는 WHERE절을 생략한다.
- 모든 레코드를 삭제하더라도 테이블 구조는 남아 있기 때문에 디스크에서 테이블을 완전히 제거하는 DROP과는 다르다.

갱신문(UPDATE ~ SET ~)

기본 테이블에 있는 튜플들 중에서 특정 튜플의 내용을 변경할 때 사용한다.

```
UPDATE 테이블명
SET 속성명1 = 데이터1[, 속성명2 = 데이터2]
WHERE 조건;
```

핵심 032 내장 SQL(Embedded SQL)

- 응용 프로그램이 실행될 때 함께 실행되도록 호스트 프로그램 언어로 만든 프로그램에 삽입된 SQL이다.
- 내장 SQL 실행문은 호스트 언어에서 실행문이 나타날 수 있는 곳이면 프로그램의 어느 곳에서나 사용할 수 있다.
- 일반 SQL문은 수행 결과로 여러 개의 튜플을 반환하는 반면, 내장 SQL은 단 하나의 튜플만을 반환한다.
- 내장 SQL문에 의해 반환되는 튜플은 일반 변수를 사용하여 저장할 수 있다.
- Host Program의 컴파일 시 내장 SQL문은 선행 처리기에 의해 분리되어 컴파일된다.
- 호스트 변수와 데이터베이스 필드의 이름은 같아도 된다.
- 내장 SQL문에 사용된 호스트 변수의 데이터 타입은 이에 대응하는 데이터베이스 필드의 SQL 데이터 타입과 일치하여야 한다.
- 내장 SQL문이 실행되면 SQL의 실행 상태가 SQL 상태 변수에 전달된다.
- 호스트 언어의 실행문과 SQL문을 구분시키는 방법
 - 명령문의 구분 : C/C++에서 내장 SQL문은 \$와 세미콜론(;) 문자 사이에 기술하고, Visual BASIC에서는 내장 SQL문 앞에 'EXEC SQL'을 기술함
 - 변수의 구분 : 내장 SQL에서 사용하는 호스트 변수는 변수 앞에 콜론(:) 문자를 붙임

핵심 033 뷰(View)

- 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된 가상 테이블이다.
- 저장장치 내에 물리적으로 존재하지 않지만, 사용자에게는 있는 것처럼 간주된다.
- 데이터 보정작업, 처리과정 시험 등 임시적인 작업을 위한 용도로 활용된다.

뷰(View)의 특징

- 기본 테이블로부터 유도된 테이블이기 때문에 기본 테이블과 같은 형태의 구조를 가지며, 조작도 기본 테이블과 거의 같다.



- 가상 테이블이기 때문에 물리적으로 구현되어 있지 않다.
- 필요한 데이터만 뷰로 정의해서 처리할 수 있기 때문에 관리가 용이하고 명령문이 간단해진다.
- 조인문의 사용을 최소화하여 사용상의 편의성을 최대화한다.
- 뷰를 통해서만 데이터에 접근하게 하면 뷰에 나타나지 않는 데이터를 안전하게 보호할 수 있다.
- 기본 테이블의 기본키를 포함한 속성(열) 집합으로 뷰를 구성해야만 삽입, 삭제, 갱신 연산이 가능하다.
- 정의된 뷰는 다른 뷰의 정의에 기초가 될 수 있다.
- 하나의 뷰를 삭제하면 그 뷰를 기초로 정의된 다른 뷰도 자동으로 삭제된다.

뷰의 장점

- 논리적 데이터 독립성을 제공한다.
- 동일 데이터에 대해 동시에 여러 사용자의 상이한 응용이나 요구를 지원해준다.
- 사용자의 데이터 관리를 간단하게 해준다.
- 접근 제어를 통한 자동 보안이 제공된다.

뷰의 단점

- 독립적인 인덱스를 가질 수 없다.
- 뷰의 정의를 변경할 수 없다.
- 뷰로 구성된 내용에 대한 삽입, 삭제, 갱신 연산에 제약이 따른다.

뷰 정의문

```
CREATE VIEW 뷰이름[(속성이름[,속성이름])]  
AS SELECT문;
```

- SELECT문을 부질의로 사용하여 SELECT문의 결과로서 뷰를 생성한다.
- 부질의로서의 SELECT문에는 UNION이나 ORDER BY절을 사용할 수 없다.
- 속성 이름을 기술하지 않으면 SELECT문의 속성 이름이 자동으로 사용된다.

뷰 삭제문

```
DROP VIEW 뷰이름 {RESTRICTED | CASCADE};
```

- RESTRICTED : 뷰를 다른 곳에서 참조하고 있으면 삭제가 취소된다.
- CASCADE : 뷰를 참조하는 다른 뷰나 제약 조건까지 모두 삭제된다.

핵심

17.8, 17.5, 17.3, 15.8, 15.5, 15.3, 14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.9, ...

034 시스템 카탈로그

- 시스템 그 자체에 관련이 있는 스키마 및 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스이다.
- 데이터베이스에 포함되는 모든 데이터 객체에 대한 정의나 명세에 관한 정보를 유지관리하는 시스템 테이블이다.
- 데이터 정의어의 결과로 구성되는 기본 테이블, 뷰, 인덱스, 패키지, 접근 권한 등의 데이터베이스 구조 및 통계 정보를 저장한다.
- 카탈로그들이 생성되면 자료 사전(Data Dictionary)에 저장되기 때문에 좁은 의미로는 카탈로그를 자료 사전이라고도 한다.
- 카탈로그에 저장된 정보를 메타 데이터(Meta-Data)라고 한다.

시스템 카탈로그의 특징

- 카탈로그 자체도 시스템 테이블로 구성되어 있어 일반 이용자로 SQL을 이용하여 내용을 검색해 볼 수 있다.
- INSERT, DELETE, UPDATE문으로 갱신하는 것은 허용하지 않는다.
- DBMS가 스스로 생성하고 유지한다.
- 카탈로그는 사용자가 SQL문을 실행시켜 기본 테이블, 뷰, 인덱스 등에 변화를 주면 시스템이 자동으로 갱신된다.

핵심

16.8, 16.5, 14.8, 13.6, 12.5, 09.3, 08.9, 07.5, 02.9, 00.7

035 트랜잭션의 정의

- 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미한다.
- 데이터베이스 시스템에서 복구 및 병행 수행 시 처리되는 작업의 논리적 단위이다.
- 하나의 트랜잭션은 Commit되거나 Rollback된다.
- 트랜잭션은 일반적으로 회복의 단위가 된다.



핵심 18.3, 17.8, 17.5, 17.3, 16.5, 16.3, 15.8, 15.5, 15.3, 14.8, 14.5, 13.8, 13.6, 12.8, 12.5, 11.8, 11.6, 11.3, ...
036 트랜잭션의 특성

Atomicity (원자성)	<ul style="list-style-type: none"> 트랜잭션의 연산은 데이터베이스에 모두 반영되든지 아니면 전혀 반영되지 않아야 함 트랜잭션 내의 모든 명령은 반드시 완벽히 수행되어야 하며, 모두가 완벽히 수행되지 않고 어느 하나라도 에러가 발생하면 트랜잭션 전부가 취소되어야 함
Consistency (일관성)	<ul style="list-style-type: none"> 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환함 시스템이 가지고 있는 고정 요소는 트랜잭션 수행 전과 트랜잭션 수행 완료 후의 상태가 같아야 함
Isolation (독립성, 격리성)	<ul style="list-style-type: none"> 둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행중에 다른 트랜잭션의 연산이 끼어들 수 없음 수행중인 트랜잭션은 완전히 완료될 때까지 다른 트랜잭션에서 수행 결과를 참조할 수 없음
Durability (영속성, 지속성)	성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 함

핵심 18.3, 17.8, 16.8, 05.9, 03.3, 99.4
037 Commit, Rollback 연산

- Commit 연산 : 하나의 논리적 단위(트랜잭션)에 대한 작업이 성공적으로 끝났고, 데이터베이스가 다시 일관된 상태에 있을 때 이 트랜잭션이 행한 갱신 연산이 완료된 것을 트랜잭션 관리자에게 알려주는 연산
- Rollback 연산 : 하나의 트랜잭션 처리가 비정상적으로 종료되어 데이터베이스의 일관성을 깨뜨렸을 때, 이 트랜잭션의 일부가 정상적으로 처리되었더라도 트랜잭션의 원자성을 구현하기 위해 이 트랜잭션이 행한 모든 연산을 취소(Undo)시키는 연산으로, 해당 트랜잭션을 재시작하거나 폐기함

핵심 17.5, 12.3, 10.9, 01.6, 01.3, 99.10
038 회복(Recovery)

- 회복은 트랜잭션들의 처리를 수행하는 도중 장애가 발생하여 데이터베이스가 손상되었을 때 손상되기 이전의 정상 상태로 복구시키는 작업이다.
- 장애의 유형
 - 트랜잭션 장애 : 입력 데이터 오류, 불명확한 데이터, 시스템 자원 요구의 과다 등 트랜잭션 내부의 비정상적인 상황으로 인하여 프로그램 실행이 중지되는 현상

- 시스템 장애 : 데이터베이스에 손상을 입히지는 않으나 하드웨어 오동작, 소프트웨어의 손상, 교착 상태 등에 의해 모든 트랜잭션의 연속적인 수행에 장애를 주는 현상
- 미디어 장애 : 저장장치인 디스크 블록의 손상이나 디스크 헤드의 충돌 등에 의해 데이터베이스의 일부 또는 전부가 물리적으로 손상된 상태

• 회복 관리기(Recovery Management)

- DBMS의 구성 요소이다.
- 트랜잭션 실행이 성공적으로 완료되지 못하면 트랜잭션이 데이터베이스에 만들었던 모든 변화를 취소(Undo)시키고 트랜잭션 수행 이전의 원래 상태로 복구하는 역할을 담당한다.
- 메모리 덤프, 로그(Log)를 이용하여 수행한다.

• 회복 기법의 종류

- 연기 갱신 기법(Deferred Update)
- 즉각 갱신 기법(Immediate Update)
- 그림자 페이지 대체 기법(Shadow Paging)
- 검사점 기법(Check Point)

핵심 18.4, 18.3, 17.8, 17.5, 16.8, 16.5, 16.3, 15.8, 15.5, 15.3, 14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 11.8, ...
039 병행 제어(Concurrency Control)

- 다중 프로그램의 이점을 활용하여 동시에 여러 개의 트랜잭션을 병행 수행시킬 때, 동시에 실행되는 트랜잭션들이 데이터베이스의 일관성을 파괴하지 않도록 트랜잭션 간의 상호작용을 제어하는 것이다.
- 병행 수행의 문제점
 - 갱신 분실(Lost Update) : 2개 이상의 트랜잭션이 같은 자료를 공유하여 갱신할 때 갱신 결과의 일부가 없어지는 현상
 - 비완료 의존성(Uncommitted Dependency) : 하나의 트랜잭션 수행이 실패한 후 회복되기 전에 다른 트랜잭션이 실패한 갱신 결과를 참조하는 현상
 - 모순성(Inconsistency) : 두 개의 트랜잭션이 병행 수행될 때 원치 않는 자료를 이용함으로써 발생하는 문제
 - 연쇄 복귀(Cascading Rollback) : 병행 수행되던 트랜잭션들 중 어느 하나에 문제가 생겨 Rollback하는 경우 다른 트랜잭션도 함께 Rollback되는 현상



- 병행 제어의 목적
 - 데이터베이스의 공유를 최대화한다.
 - 시스템의 활용도를 최대화한다.
 - 데이터베이스의 일관성을 유지한다.
 - 사용자에게 대한 응답 시간을 최소화한다.
- 로킹(Locking)
 - 로킹은 주요 데이터의 액세스를 상호 배타적으로 하는 것이다.
 - 트랜잭션들이 어떤 로킹 단위를 액세스하기 전에 Lock(잠금)을 요청해서 Lock이 허락되어야만 그 로킹 단위를 액세스할 수 있도록 하는 기법이다.
- 로킹 단위(Locking Granularity)
 - 병행 제어에서 한꺼번에 로킹할 수 있는 데이터 단위이다.
 - 데이터베이스, 파일, 레코드, 필드 등은 로킹 단위가 될 수 있다.
 - 로킹 단위가 크면 로크 수가 작아 관리하기 쉽지만 병행성 수준이 낮아지고, 로킹 단위가 작으면 로크 수가 많아 관리하기 복잡하지만 병행성 수준이 높아진다.

- 비밀키는 제3자에게는 노출시키지 않고, 데이터베이스 사용 권한이 있는 사용자만 나누어 가진다.
- 대칭 암호 방식 또는 단일키 암호화 기법이라고도 한다.
- 대표적으로 DES(Data Encryption Standard)가 있다.
- 장점 : 암호화/복호화 속도가 빠르며, 알고리즘이 단순하고 파일 크기가 작음
- 단점 : 사용자의 증가에 따라 관리해야 할 키의 수가 상대적으로 많아짐
- DES는 56Bit의 16개 키를 이용하여 64Bit의 평문 블록을 16회의 암호 계산 단계를 거쳐 64Bit의 암호문을 얻는다.
- 공개키 암호 방식(Public Key Encryption)
 - 서로 다른 키로 데이터를 암호화하고 복호화한다.
 - 데이터를 암호화할 때 사용하는 키(공개키, Public Key)는 데이터베이스 사용자에게 공개하고, 복호화할 때의 키(비밀키, Secret Key)는 관리자가 비밀리에 관리하는 방법이다.
 - 비대칭 암호 방식이라고도 하며, 대표적으로 RSA (Rivest Shamir Adleman)가 있다.
 - 장점 : 키의 분배가 용이하고, 관리해야 할 키의 개수가 적음
 - 단점 : 암호화/복호화 속도가 느리며, 알고리즘이 복잡하고 파일 크기가 큼

핵심 040 보안(Security)

데이터베이스 보안의 개요

- 데이터베이스의 일부분 또는 전체에 대해서 권한이 없는 사용자가 액세스를 수행하는 것을 금지하기 위해 사용되는 기술이다.
- 데이터베이스 사용자들은 일반적으로 서로 다른 객체에 대하여 다른 접근권리 또는 권한을 갖게된다.

무결성(Integrity)과 보안(Security)

- 무결성은 권한이 있는 사용자로부터 데이터베이스를 보호하는 것이고, 보안은 권한이 없는 사용자로부터 데이터베이스를 보호하는 것을 말한다.
- 보안은 데이터베이스 사용자들이 데이터베이스를 사용하고자 할 때 언제든지 사용할 수 있도록 보장하는 것이고, 무결성은 정확하게 사용할 수 있도록 보장하는 것을 말한다.

암호화 기법

- 개인키 암호 방식(Private Key Encryption) = 비밀키 암호 방식
 - 동일한 키로 데이터를 암호화하고 복호화한다.

핵심 041 권한 부여 기법

- 권한 부여 기법은 일반적으로 사용자들이 서로 다른 객체에 대하여 서로 다른 접근 권한을 갖게 설정한다.
- 권한 부여 기법에서 보안을 위한 데이터 단위는 테이블 전체나 특정한 행, 열에 있는 데이터 값이 될 수 있다.
- GRANT : 권한 부여 명령
- REVOKE : 권한 취소 명령
- 사용자 등급 지정 및 해제

- GRANT 사용자 등급 TO 사용자 ID 리스트 [IDENTIFIED BY 암호 리스트];
- REVOKE 사용자 등급 FROM 사용자 ID 리스트;

- 사용자 등급의 종류
 - DBA : 데이터베이스 관리 책임자



- RESOURCE : 데이터베이스 및 테이블 생성 가능자
- CONNECT : 단순 사용자

예제 1 GRANT RESOURCE TO KORA;

사용자 ID가 KORA인 사람에게 데이터베이스 및 테이블을 생성할 수 있는 권한을 부여한다.

예제 2 GRANT CONNECT TO STAR;

사용자 ID가 STAR인 사람에게 단순히 데이터베이스에 있는 정보를 검색할 수 있는 권한을 부여한다.

- 테이블 및 속성에 대한 권한 부여 및 취소

- GRANT 권한 ON 데이터 개체 TO 사용자 [WITH GRANT OPTION];
- REVOKE [GRANT OPTION FOR] 권한 ON 데이터 개체 FROM 사용자 [CASCADE];

- 권한 종류 : ALL, SELECT, INSERT, DELETE, UPDATE, INDEX, ALTER 등
- WITH GRANT OPTION : 부여받은 권한을 다른 사용자에게 다시 부여할 수 있는 권한을 부여한다.
- GRANT OPTION FOR : 다른 사용자에게 권한을 부여할 수 있는 권한을 취소한다.
- CASCADE : 권한 해제 시 권한을 부여받았던 사용자가 다른 사용자에게 부여한 권한도 연쇄적으로 해제한다.

예제 3 GRANT ALL ON 고객 TO JULIA WITH GRANT OPTION;

사용자 ID가 JULIA인 사람에게 고객 테이블에 대한 모든 권한과 다른 사람에게도 권한을 부여할 수 있는 권한까지 부여한다.

예제 4 REVOKE UPDATE ON 고객 FROM JULIA;

사용자 ID가 JULIA인 사람에게 부여한 권한 중 갱신(UPDATE) 권한을 취소한다.

• 분산 데이터베이스의 4대 목표

위치 투명성 (Location Transparency)	액세스하려는 데이터베이스의 실제 위치를 알 필요 없이 단지 데이터베이스의 논리적인 명칭만으로 액세스할 수 있음
중복(복제) 투명성 (Replication Transparency)	동일 데이터가 여러 곳에 중복되어 있더라도 사용자는 마치 하나의 데이터만 존재하는 것처럼 사용하고, 시스템은 자동으로 여러 자료에 대한 작업을 수행함
병행 투명성 (Concurrency Transparency)	분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실행되더라도 그 트랜잭션의 결과는 영향을 받지 않음
장애 투명성 (Failure Transparency)	트랜잭션, DBMS, 네트워크, 컴퓨터 장애에도 불구하고 트랜잭션을 정확하게 처리함

핵심 14.3, 11.8, 10.5, 10.3, 08.9, 08.5, 07.5, 06.9, 06.3, 05.4, 04.5, 03.8, 02.5, 02.3, 01.9, 00.10

043 분산 데이터베이스의 장 · 단점

장 점	단 점
<ul style="list-style-type: none"> • 지역 자치성이 높음 • 자료의 공유성이 향상 • 분산 제어가 가능함 • 시스템 성능이 향상 • 효율성과 융통성이 높음 • 신뢰성 및 가용성이 높음 • 점증적 시스템 용량 확장이 용이함 	<ul style="list-style-type: none"> • DBMS가 수행할 기능이 복잡함 • 데이터베이스 설계가 어려움 • 소프트웨어 개발 비용이 증가함 • 처리 비용이 증가함 • 잠재적 오류가 증가함

핵심 17.8, 16.8, 16.5, 13.8, 13.3, 12.8, 12.5, 07.5, 07.3, 06.9, 06.5, 05.5, 04.5, 03.8, 03.3, 01.6, 01.3

044 자료 구조의 분류

- 선형 구조 : 선형 리스트(배열), 연결 리스트, 스택, 큐, 데크
- 비선형 구조 : 트리, 그래프

핵심 07.9, 02.9, 01.9, 00.10, 00.3

045 연결 리스트(Linked List)

- 연결 리스트는 자료들을 임의의 기억공간에 기억시키되, 자료 항목의 순서에 따라 노드의 포인터 부분을 이용하여 서로 연결시킨 자료 구조이다.
- 노드의 삽입, 삭제 작업이 용이하다.
- 기억공간이 연속적으로 놓여있지 않아도 저장이 가능하다.
- 연결을 위한 링크(포인터) 부분이 필요하기 때문에 순차 리스트에 비해 기억공간 이용 효율이 좋지 않다.

핵심 11.3, 09.8, 07.9, 06.9, 05.9, 05.3, 04.9, 04.3, 03.8, 03.3, 01.9

042 분산 데이터베이스

- 분산 데이터베이스는 논리적으로는 하나의 시스템에 속하지만 물리적으로는 네트워크를 통해 연결된 여러 개의 컴퓨터 사이트(Site)에 분산되어 있는 데이터베이스를 의미한다.



- 접근 속도가 느리다.
- 연결 리스트 중에서 중간 노드의 연결이 끊어지면 그 다음 노드를 찾기 힘들다.
- 희소 행렬을 링크드 리스트로 표현하면 기억장소가 절약된다.

잠깐만요 ! 희소 행렬(Sparse Matrix)

행렬의 요소 중 많은 항들이 0으로 되어 있는 형태로, 기억장소를 절약하기 위해 링크드 리스트를 이용하여 저장합니다.

핵심 요약 17.8, 15.3, 14.8, 13.8, 12.5, 11.6, 10.9, 09.5, 09.3, 08.9, 08.5, 08.3, 07.5, 07.3, 06.5, 06.3, 05.9, 04.3, 03.8
046 스택(Stack)

- 리스트의 한쪽 끝으로만 자료의 삽입, 삭제 작업이 이루어지는 자료 구조이다.
- 가장 나중에 삽입된 자료가 가장 먼저 삭제되는 후입선출(LIFO; Last-In, First-Out) 방식으로 자료를 처리한다.
- TOP : Stack으로 할당된 기억공간에 가장 마지막으로 삽입된 자료가 기억된 위치를 가리키는 요소, 스택 포인터라고도 함
- Bottom : 스택의 가장 밑바닥임
- PUSH : 스택에 자료를 입력하는 명령
- POP : 스택에서 자료를 출력하는 명령

Stack의 용도

- 부프로그램 호출 시 복귀주소를 저장할 때
- 함수 호출의 순서 제어
- 인터럽트가 발생하여 복귀주소를 저장할 때
- 후위 표기법(Postfix Notation)으로 표현된 산술식을 연산할 때
- 0 주소지정방식 명령어의 자료 저장소
- 재귀(Recursive) 프로그램의 순서 제어
- 컴파일러를 이용한 언어 번역 시

잠깐만요 ! 재귀(Recursion) 프로그램

재귀(Recursion) 프로그램이란 한 루틴(Routine)이 자기를 다시 호출하여 실행하는 프로그램을 말합니다.

핵심 요약 18.3, 17.8, 17.5, 16.5, 16.3, 15.5, 15.3, 14.5, 13.8, 12.8, 11.8, 11.6, 10.5, ...
047 스택의 삽입(Push)과 삭제(Pop)

삽입(Push)

Top = Top + 1	스택 포인터(Top)를 1 증가시킨다.
If Top > M Then Overflow	스택 포인터가 스택의 크기보다 크면 Overflow
Else X(Top) ← Item	그렇지 않으면 Item이 가지고 있는 값을 스택의 Top 위치에 삽입한다.

- M : 스택의 크기
- Top : 스택 포인터
- X : 스택의 이름
- Overflow : 스택으로 할당받은 메모리 부분의 마지막 주소가 M번지라고 할 때, Top Pointer의 값이 M보다 커지면 스택의 모든 기억장소가 꽉 채워져 있는 상태이므로 더 이상 자료를 삽입할 수 없어 Overflow를 발생시킨다.

삭제(Pop)

If Top = 0 Then Underflow	스택 포인터가 0이면 스택의 바닥이어서 더 이상 삭제할 자료가 없으므로 Underflow를 처리한다.
Else Item ← X(Top) Top = Top - 1	그렇지 않으면 Top 위치에 있는 값을 Item으로 옮기고 스택 포인터를 1 감소시킨다.

- Underflow : Top Pointer가 주소 0을 가지고 있다면 스택에는 삭제할 자료가 없으므로 Underflow를 발생시킨다.

잠깐만요 ! Stack에 기억되어 있는 자료를 삭제시킬 때는 제일 먼저 삭제할 자료가 있는지 없는지부터 확인해야 합니다.

핵심 요약 12.8, 12.3, 08.5, 07.9, 06.5, 05.9, 03.5, 03.3, 02.5, 02.3, 00.10
048 큐(Queue)

- 선형 리스트의 한쪽에서는 삽입 작업이 이루어지고 다른 한쪽에서는 삭제 작업이 이루어지도록 구성된 자료 구조이다.
- 시작(Front 또는 Head)과 끝(Rear 또는 Tail)을 표시하는 두 개의 포인터가 있다.
- 가장 먼저 삽입된 자료가 가장 먼저 삭제되는 선입선출(FIFO; First-In, First-Out) 방식으로 처리한다.
- 프론트(F, Front) 포인터 : 가장 먼저 삽입된 자료의 기억공간을 가리키는 포인터로, 삭제 작업을 할 때 사용함
- 리어(R, Rear) 포인터 : 가장 마지막에 삽입된 자료가 위치한 기억장소를 가리키는 포인터로, 삽입 작업을 할 때 사용함



Queue를 이용하는 예

- 창구 업무처럼 서비스 순서를 기다리는 등의 대기 행렬의 처리에 사용한다.
- 운영체제의 작업 스케줄링에 사용한다.

핵심 049 데크(Deque)

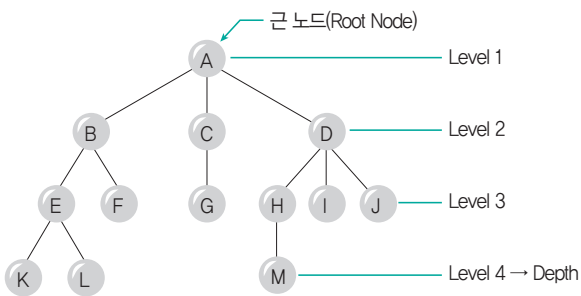
18.3, 07.9, 06.3, 05.4

- 삽입과 삭제가 리스트의 양쪽 끝에서 모두 발생할 수 있는 자료 구조이다.
- Double Ended Queue의 약자이다.
- Stack과 Queue의 장점만 따서 구성한 것이다.
- 입력이 한쪽에서만 발생하고 출력은 양쪽에서 일어날 수 있는 입력 제한과 입력은 양쪽에서 일어나고 출력은 한쪽에서만 이루어지는 출력 제한이 있다.
- 입력 제한 데크 : Scroll
- 출력 제한 데크 : Shelf

핵심 050 트리(Tree)

18.4, 18.3, 17.5, 16.8, 16.5, 16.3, 15.8, 14.8, 14.5, 14.3, 13.6, 11.8, 10.9, ...

정점(Node, 노드)과 선분(Branch, 가지)을 이용하여 사이클을 이루지 않도록 구성한 Graph의 특수한 형태로 가족의 계보(족보), 연산 수식, 회사 조직 구조도, 힙(Heap) 등을 표현하기에 적합하다.



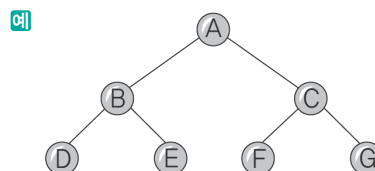
- 노드(Node) : 트리의 기본 요소로서 자료 항목과 다른 항목에 대한 가지(Branch)를 합친 것
예 A, B, C, D, E, F, G, H, I, J, K, L, M
- 근 노드(Root Node) : 트리의 맨 위에 있는 노드 예 A
- 디그리(Degree, 차수) : 각 노드에서 뻗어나온 가지의 수
예 A = 3, B = 2, C = 1, D = 3

- 트리의 디그리 : 노드들의 디그리 중에서 가장 많은 수
예 노드 A나 D가 3개의 디그리를 가지므로 위 트리의 디그리는 3이다.
- 단말 노드(Terminal Node) = 잎 노드(Leaf Node) : 자식이 하나도 없는 노드, 즉 디그리가 0인 노드
예 K, L, F, G, M, I, J
- 비단말 노드(Non-Terminal Node) : 자식이 하나라도 있는 노드, 즉 디그리가 0이 아닌 노드
예 A, B, C, D, E, H
- 자식 노드(Son Node, Children Node) : 어떤 노드에 연결된 다음 레벨의 노드들
예 D의 자식 노드 : H, I, J
- 부모 노드(Parent Node) : 어떤 노드에 연결된 이전 레벨의 노드
예 E, F의 부모 노드는 B
- 형제 노드(Brother Node, Sibling) : 동일한 부모를 갖는 노드들
예 H의 형제 노드는 I, J
- Level : 근 노드의 Level을 1로 가정한 후 어떤 Level이 L이면 자식 노드는 L+1이다.
예 H의 레벨은 3
- 깊이(Depth, Height) : 어떤 Tree에서 노드가 가질 수 있는 최대의 레벨
예 위 트리의 깊이는 4

핵심 051 이진 트리의 특성

17.8, 17.3, 01.6, 01.3

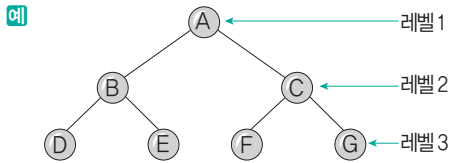
- 이진 트리의 레벨 i에서 최대 노드의 수는 2^{i-1} 이다.
- 이진 트리에서 Terminal Node수가 n_0 , 차수가 2인 노드 수가 n_2 라 할 때 $n_0 = n_2 + 1$ 이 된다.



- 레벨 3에서 최대 노드의 수는 $2^{3-1} = 4$ 이다.
- Terminal 노드의 개수가 4개이고, 차수가 2인 노드가 3개 이므로 $4 = 3 + 1$ 에 의해 $n_0 = n_2 + 1$ 이 성립된다.



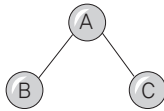
- 정이진 트리는 깊이(Depth)가 k 일 때 전체 노드의 수가 $2^k - 1$ 개의 노드이고, 레벨 i 마다 2^{i-1} 개의 노드들로 짝 찬 트리를 말한다.



- 전체 Node의 수 : $k = 3, 2^3 - 1 = 8 - 1 = 7$
- 레벨 1에서 $2^{1-1} = 2^0 = 1$
- 레벨 2에서 $2^{2-1} = 2^1 = 2$
- 레벨 3에서 $2^{3-1} = 2^2 = 4$

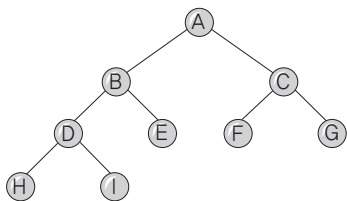
핵심 18.4, 17.8, 17.5, 16.8, 16.3, 15.8, 15.5, 15.3, 14.8, 14.5, 14.3, 13.8, 13.3, 12.8, ...

052 이진 트리의 운행법



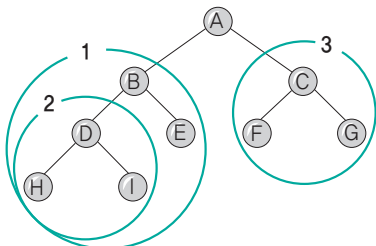
- Preorder(전위) 운행 : Root \rightarrow Left \rightarrow Right 순으로 운행함 A, B, C
- Inorder(중위) 운행 : Left \rightarrow Root \rightarrow Right 순으로 운행함 B, A, C
- Postorder(후위) 운행 : Left \rightarrow Right \rightarrow Root 순으로 운행함 B, C, A

예 다음 트리를 Inorder, Preorder, Postorder 방법으로 운행했을 때 각 노드를 방문한 순서는?



〈Preorder 운행법의 방문 순서〉

서브 트리를 하나의 노드로 생각할 수 있도록 다음과 같이 서브 트리 단위로 묶는다. Preorder, Inorder, Postorder 모두 공통으로 사용한다.



① Preorder는 Root \rightarrow Left \rightarrow Right이므로 A13이 된다.

② 1은 B2E이므로 AB2E3이 된다.

③ 2는 DHI이므로 ABDHIE3이 된다.

④ 3은 CFG이므로 ABDHIECFG가 된다.

\therefore 방문 순서 : ABDHIECFG

〈Inorder 운행법의 방문 순서〉

① Inorder는 Left \rightarrow Root \rightarrow Right이므로 1A3이 된다.

② 1은 2BE이므로 2BEA3이 된다.

③ 2는 HDI이므로 HDIBEA3이 된다.

④ 3은 FCG이므로 HDIBEAFCG가 된다.

\therefore 방문 순서 : HDIBEAFCG

〈Postorder의 방문 순서〉

① Postorder는 Left \rightarrow Right \rightarrow Root이므로 13A가 된다.

② 1은 2EB이므로 2EB3A가 된다.

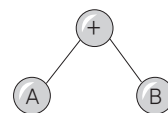
③ 2는 HID이므로 HIDEB3A가 된다.

④ 3은 FGC이므로 HIDEBFGCA가 된다.

\therefore 방문 순서 : HIDEBFGCA

핵심 18.4, 18.3, 16.5, 14.5, 12.3, 09.5, 09.3, 07.9, 07.3, 06.5, 06.3, 05.5, 05.4, ...

053 수식의 표기법



- 전위 표기법(Prefix) : 연산자 \rightarrow Left \rightarrow Right, +AB
- 중위 표기법(Infix) : Left \rightarrow 연산자 \rightarrow Right, A+B
- 후위 표기법(Postfix) : Left \rightarrow Right \rightarrow 연산자, AB+

Infix 표기를 Postfix로 바꾸기

Infix로 표기된 수식에서 연산자를 해당 피연산자 2개의 뒤(오른쪽)에 오도록 이동하면 Postfix가 된다.

$$X = A / B * (C + D) + E \rightarrow X A B / C D + * E + =$$

① 연산 우선순위에 따라 괄호로 묶는다.

$$(X = ((A / B) * (C + D)) + E)$$



② 연산자를 해당 괄호의 뒤로 옮긴다.

$$(X = ((A / B) * (C + D)) + E))$$



$$(X ((AB) / (CD) +) * E) +) =$$

③ 괄호를 제거한다.

$$X A B / C D + * E + =$$

Infix 표기를 Prefix로 바꾸기

Infix로 표기된 수식에서 연산자를 해당 피연산자 2개의 앞(왼쪽)에 오도록 이동하면 Prefix가 된다.

① 연산 우선순위에 따라 괄호로 묶는다.

$$(X = ((A / B) * (C + D)) + E))$$

② 연산자를 해당 괄호의 앞으로 옮긴다.

$$(X = ((A / B) * (C + D)) + E))$$



$$=(X + (* (/ (A B) + (C D)) E))$$

③ 괄호를 제거한다.

$$=X + * / A B + C D E$$

Postfix 표기를 Infix로 바꾸기

Postfix는 Infix 표기법에서 연산자를 해당 피연산자 2개의 뒤(오른쪽)로 이동한 것이므로 연산자를 다시 해당 피연산자 2개의 가운데로 옮기면 된다.

$$A B C - / D E F + * + \rightarrow A / (B - C) + D * (E + F)$$

① 먼저 인접한 피연산자 2개와 오른쪽의 연산자를 괄호로 묶는다.

$$((A (B C -) /) (D (E F +) *) +)$$

② 연산자를 해당 피연산자의 가운데로 이동시킨다.

$$((A (B C -) /) (D (E F +) *) +)$$



$$((A / (B - C)) + (D * (E + F)))$$

③ 필요 없는 괄호를 제거한다.

$$A / (B - C) + D * (E + F)$$

054 주요 정렬 알고리즘의 이해

삽입 정렬

예제 8, 5, 6, 2, 4를 삽입 정렬로 정렬하시오.

• 초기 상태 :

8	5	6	2	4
---	---	---	---	---

• 1회전 :

8	5	6	2	4
---	---	---	---	---

 →

5	8	6	2	4
---	---	---	---	---

두 번째 값 5를 첫 번째 값과 비교하여 첫 번째 자리에 삽입하고 8을 한 칸 뒤로 이동시킨다.

• 2회전 :

5	8	6	2	4
---	---	---	---	---

 →

5	6	8	2	4
---	---	---	---	---

세 번째 값 6을 첫 번째, 두 번째 값과 비교하여 8자리에 삽입하고 8은 한 칸 뒤로 이동시킨다.

• 3회전 :

5	6	8	2	4
---	---	---	---	---

 →

2	5	6	8	4
---	---	---	---	---

네 번째 값 2를 처음부터 비교하여 맨 처음에 삽입하고 나머지를 한 칸씩 뒤로 이동시킨다.

• 4회전 :

2	5	6	8	4
---	---	---	---	---

 →

2	4	5	6	8
---	---	---	---	---

다섯 번째 값 4를 처음부터 비교하여 5자리에 삽입하고 나머지를 한 칸씩 뒤로 이동시킨다.

버블 정렬

예제 8, 5, 6, 2, 4를 버블 정렬로 정렬하시오.

• 초기 상태 :

8	5	6	2	4
---	---	---	---	---

• 1회전 :

5	8	6	2	4
---	---	---	---	---

 →

5	6	8	2	4
---	---	---	---	---

5	6	2	8	4
---	---	---	---	---

 →

5	6	2	4	8
---	---	---	---	---

• 2회전 :

5	6	2	4	8
---	---	---	---	---

 →

5	2	6	4	8
---	---	---	---	---

→

5	2	4	6	8
---	---	---	---	---

• 3회전 :

2	5	4	6	8
---	---	---	---	---

 →

2	4	5	6	8
---	---	---	---	---

• 4회전 :

2	4	5	6	8
---	---	---	---	---



선택 정렬

예제 8, 5, 6, 2, 4를 선택 정렬로 정렬하시오.

- 초기 상태 :

8	5	6	2	4
---	---	---	---	---
- 1회전 :

5	8	6	2	4
---	---	---	---	---

 →

5	8	6	2	4
---	---	---	---	---

→

2	8	6	5	4
---	---	---	---	---

 →

2	8	6	5	4
---	---	---	---	---
- 2회전 :

2	6	8	5	4
---	---	---	---	---

 →

2	5	8	6	4
---	---	---	---	---

→

2	4	8	6	5
---	---	---	---	---
- 3회전 :

2	4	6	8	5
---	---	---	---	---

 →

2	4	5	8	6
---	---	---	---	---
- 4회전 :

2	4	5	6	8
---	---	---	---	---

2-Way 합병 정렬(Merge Sort)

예제 71, 2, 38, 5, 7, 61, 11, 26, 53, 42를 2-Way 합병 정렬로 정렬하시오.

- 1회전 : 2개씩 묶은 후 각각의 묶음 안에서 정렬합니다.
(71, 2) (38, 5) (7, 61) (11, 26) (53, 42)
↓
(2, 71) (5, 38) (7, 61) (11, 26) (42, 53)
- 2회전 : 묶여진 묶음을 2개씩 묶은 후 각각의 묶음 안에서 정렬합니다.
((2, 71) (5, 38)) ((7, 61) (11, 26)) (42, 53)
↓
(2, 5, 38, 71) (7, 11, 26, 61) (42, 53)
- 3회전 : 묶여진 묶음을 2개씩 묶은 후 각각의 묶음 안에서 정렬합니다.
((2, 5, 38, 71) (7, 11, 26, 61)) (42, 53)
↓
(2, 5, 7, 11, 26, 38, 61, 71) (42, 53)
- 4회전 : 묶여진 묶음 2개를 하나로 묶은 후 정렬합니다.
((2, 5, 7, 11, 26, 38, 61, 71) (42, 53))
↓
2, 5, 7, 11, 26, 38, 42, 53, 61, 71

핵심

18.3, 16.8, 13.6, 11.3, 06.3, 05.5, 04.9, 03.3, 02.5, 00.7

055 이분 검색(이진 검색)

- 제어 검색의 일종인 이분 검색은 반드시 순서화된 파일이어야 검색할 수 있다.
- 탐색 효율이 좋고 탐색 시간이 적게 소요된다.
- 전체 파일을 두 개의 서브 파일로 분리해 가면서 Key 레코드를 검색하기 때문에 검색회수를 거듭할 때마다 검색 대상이 되는 데이터의 수가 절반으로 줄어든다.
- 찾고자 하는 Key 값을 파일의 중간 레코드 Key 값과 비교하면서 검색한다.
- 중간 레코드 번호(M) : $\frac{F+L}{2}$ (단, F : 첫 번째 레코드 번호, L : 마지막 레코드 번호)

핵심

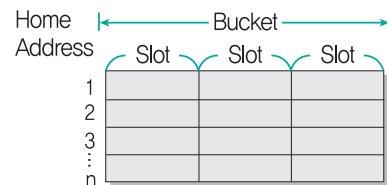
17.5, 16.8, 16.5, 15.8, 15.3, 13.3, 08.5, 07.5, 06.3, 04.5, 04.3, 01.3, 99.4

056 해싱(Hashing)

- Hash Table이라는 기억공간을 할당하고, 해시 함수(Hash Function)를 이용하여 레코드 키에 대한 Hash Table 내의 Home Address를 계산한 후 주어진 레코드를 해당 기억장소에 저장하거나 검색 작업을 수행하는 방식이다.
- DAM(직접접근방법) 파일을 구성할 때 해싱이 사용되며, 접근 속도는 빠르지만 기억공간이 많이 요구된다.
- 여러가지 검색 방식 중 검색 속도가 가장 빠르다.
- 삽입, 삭제 작업의 빈도가 많을 때 유리한 방식이다.
- 키-주소 변환 방법이라고도 한다.

해시 테이블(Hash Table)

- 레코드를 1개 이상 보관할 수 있는 Home Bucket들로 구성된 기억공간으로, 보조기억장치에 구성할 수도 있고 주기억장치에 구성할 수도 있다.



* n크기의 3개의 슬롯으로 구성된 버킷을 가진 해시표

- 버킷(Bucket) : 하나의 주소를 갖는 파일의 한 구역을 의미하며, 버킷의 크기는 같은 주소에 포함될 수 있는 레코드 수를 의미함



- 슬롯(Slot) : 1 개의 레코드를 저장할 수 있는 공간으로 n 개의 슬롯이 모여 하나의 버킷을 형성함
- Collision(충돌 현상) : 서로 다른 2개 이상의 레코드가 같은 주소를 갖는 현상
- Synonym : 같은 Home Address를 갖는 레코드들의 집합
- Overflow
 - 계산된 Home Address의 Bucket 내에 저장할 기억공간이 없는 상태
 - Bucket을 구성하는 Slot이 여러 개일 때는 Collision은 발생해도 Overflow는 발생하지 않을 수 있음

- 레코드를 참조할 때는 색인을 탐색한 후 색인이 가리키는 포인터(주소)를 사용하여 직접 참조할 수 있다.
- 일반적으로 자기 디스크에 많이 사용되며, 자기 테이프에서는 사용할 수 없다.

색인 순차 파일의 구성

- 기본 구역(Prime Area) : 실제 레코드들을 기록하는 부분으로, 각 레코드는 키 값 순으로 저장
- 색인 구역(Index Area) : 기본 구역에 있는 레코드들의 위치를 찾아가는 색인이 기록되는 부분으로, 트랙 색인 구역, 실린더 색인 구역, 마스터 색인 구역으로 구분할 수 있음
- 오버플로 구역(Overflow Area) : 기본 구역에 빈 공간이 없어서 새로운 레코드의 삽입이 불가능할 때를 대비하여 예비적으로 확보해 둔 부분

실린더 오버플로 구역(Cylinder Overflow Area)	각 실린더마다 만들어지는 오버플로 구역으로, 해당 실린더의 기본 구역에서 오버플로 된 데이터를 기록함
독립 오버플로 구역(Independent Overflow Area)	실린더 오버플로 구역에 더 이상 오버플로 된 데이터를 기록할 수 없을 때 사용할 수 있는 예비 공간으로, 실린더 오버플로 구역과는 별도로 만들어짐

색인 순차 파일의 장점

- 순차 처리와 랜덤 처리가 모두 가능하므로, 목적에 따라 융통성 있게 처리할 수 있다.
- 효율적인 검색이 가능하고 레코드의 삽입, 삭제, 갱신이 용이하다.

색인 순차 파일의 단점

- 색인 구역과 오버플로우 구역을 구성하기 위한 추가 기억 공간이 필요하다.
- 파일이 정렬되어 있어야 하므로 추가, 삭제가 많으면 효율이 떨어진다.
- 색인을 이용한 액세스를 하기 때문에 액세스 시간이 랜덤 편성 파일보다 느리다.

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요?
끼떡기 전에 다시 한 번 복습하고 싶다고요?
지금 당장 QR 코드를 스캔해 보세요.



핵심 16.3, 15.5, 14.8, 13.6, 12.3, 11.3, 10.9, 10.5

057 순차 파일(Sequential File) = 순서 파일

- 입력되는 데이터들을 논리적인 순서에 따라 물리적 연속 공간에 순차적으로 기록하는 방식이다.
- 급여 관리 등과 같이 변동 사항이 크지 않고 기간별로 일괄 처리를 주로 하는 경우에 적합하다.
- 주로 순차 접근이 가능한 자기 테이프에서 사용된다.

순차 파일의 장점

- 기록 밀도가 높아 기억공간을 효율적으로 사용할 수 있다.
- 레코드가 키 순서대로 편성되어 취급이 용이하다.
- 매체 변환이 쉬워 어떠한 매체에도 적용할 수 있다.
- 레코드를 기록할 때 사용한 키 순서대로 레코드를 처리하는 경우, 다른 편성법보다 처리 속도가 빠르다.

순차 파일의 단점

- 파일에 새로운 레코드를 삽입, 삭제, 수정하는 경우 파일 전체를 복사해야 하므로 시간이 많이 소요된다.
- 데이터 검색 시 처음부터 순차적으로 하기 때문에 검색 효율이 낮다.

핵심 18.4, 17.8, 15.8, 14.5, 14.3, 13.3, 10.3, 09.5, 09.3, 08.9, 05.3, 03.3, 02.9, 02.3, 00.5, 00.3, 99.6

058 색인 순차 파일(Indexed Sequential File)

- 순차 처리와 랜덤 처리가 모두 가능하도록 레코드들을 키 값 순으로 정렬(Sort)시켜 기록하고, 레코드의 키 항목만을 모은 색인을 구성하여 편성하는 방식이다.
- 색인을 이용한 순차적인 접근 방법을 제공하여 ISAM(Index Sequential Access Method)이라고도 한다.