정보처리기사 실기 준비

응집도 ↑ 결합도 ↓ good!

응집도 (기 순 통 정 시 논 우)

기능적 응집도 functional	* 모든 기능 요소들이 하나의 문제를 해결하기 위해 수행되는 경우 * 모듈은 계층적으로 구성되며 아래로 갈수록 세분화되기 때문에 최하위 모듈의 대부분은 기능적 응집도를 가짐
순차적 응집도 sequential	* 모듈의 기능 수행으로 인한 출력 결과를 다른 모듈의 입력값으로 사용되는 경우
통신적 응집도 communication	* 동일한 입력을 기반으로 수행된 기능의 출력 결과를 이용하여 서로 다른 기능을 수행하는 경우
절차적 응집도 procedural	* 하나의 문제를 해결하기 위해 여러 모듈들이 순차적으로 수행되는 경우
시간적 응집도 temporal	* 각 기능들의 연관성은 없지만 특정 시기에 함께 수행되어야 하는 경우
논리적 응집도 logical	* 유사한 성격이나 형태를 가진 기능을 하나의 모듈에서 수행되도록 하는 경우
우연적 응집도 coincidental	* 모듈 내부의 구성 요소들이 서로 아무런 관련이 없는 경우 * 모듈의 이점이 전혀 없고 유지보수가 어렵기 때문에 모듈 설계를 다시 해야함

결합도 (자 스 제 외 공 내)

자료 결합도 data	* 모듈 간의 인터페이스로 전달되는 인수와 매개변수를 통해서만 상호작용이 일어나는 경우
스탬프 결합도 stamp	* 관련 있는 모듈들이 동일한 자료 구조를 공유하는 경우 * 특정 모듈에서 자료 구조를 변화시켰다면, 관련 있는 다른 모듈에 그 영향이 미침
제어 결합도 control	* 전달 대상 모듈에게 값만 전달하는 것이 아니라 제어 요소를 함께 전달하는 경우 * 전달되는 제어 요소에 따라 대상 모듈의 처리 절차가 달라짐
외부 결합도 external	* 인수의 전달 없이 특정 모듈이 다른 모듈 내부 데이터를 참조하는 경우
공유 결합도 common	* 모듈이 모듈 외부에 선언된 변수를 참조하여 기능을 수행하는 경우 * 외부 변수는 관련 없는 모듈들도 접근할 수 있으므로 문제가 발생할 가능성이 큼
내용 결합도 content	* 모듈이 다른 모듈의 내부 기능과 데이터를 직접적으로 사용하는 경우 * 가장 좋지 않은 결합

생성 구조	Abstract Factory 추상 팩토리	-구체적인 클래스에 의존하지 않고 서로 연관되거나 의존적인 객체들의 조합을 만드는 인터페이스를 제공 -동일한 주제의 다른 팩토리를 묶음
	Builder 빌더	-복잡한 인스턴스를 조립해서 만드는 구조 -복합 객체를 생성할 때 객체를 생성하는 방법과 객체를 구현하는 방법을 분리함으로써 동일한 생성 절차에서 서로 다른 표현 결과를 만들 수 있음 -생성과 표기를 분리해서 복잡한 객체를 생성
	Factory Method 팩토리 메소드	-상위 클래스에서 객체를 생성하는 인터페이스를 정의하고, 하위 클래스에서 인스턴스를 생성하도록 하는 방식 -상위 클래스에서 인스턴스를 만드는 방법만 결정하고 하위 클래스에서는 그 데이터의 생성을 책임지고 조작하는 함수들을 오버로딩하여 인터페이스와 실제 객체를 생성하는 클래스를 분리할 수 있는 패턴 -생성할 객체의 클래스를 국한하지 않고 객체를 생성 -객체 생성 처리를 서브 클래스로 분리해 처리하도록 캡슐화 하는 패턴
	Prototype 프로토타입	-처음부터 일반적인 원형을 만들어 놓고 그것을 복사한 후 필요한 부분만 수정해서 사용하는 패턴 -생성할 객체의 원형을 제공하는 인스턴스에서 생성할 객체들의 타입이 결정되도록 설정하고 객체를 생성할 때 갖추어야할 기본 형태가 있을 때 사용되는 패턴 -기존 객체를 복제함으로써 객체를 생성
	Singleton 싱글톤	-전역 변수를 사용하지 않고 객체를 하나만 생성하도록 하며, 생성된 객체를 어디서든지 참조할 수 있고록 함 -한 클래스에 한 객체만 존재하도록 제한

구조 패턴	Adapter 어댑터	-기존에 생성된 클래스를 재사용할 수 있도록 중간에서 맞춰주는 역할을 하는 인터페이스를 만드는 패턴 -상속을 이용하는 클래스 패턴과 위임을 이용하는 인스턴스 패턴의 두 가지 형태가 있음 -인터페이스가 호환 되짖 않는 클래스들을 함께 이용할 수 있도록, 타 클래스의 인터페이스를 기존 인터페이스에 덧씌움
	Bridge 브릿지	-기능의 클래스 계층과 구현의 클래스 계층을 연결하고, 구현부에서 추상 계층을 분리하여 추상화된 부분과 실제 구현 부분을 독립적으로 확장할 수 있는 디자인 패턴 -구현 뿐만 아니라, 추상화된 부분까지 변경해야하는 경우 활용 -추상화와 구현을 분리해 둘을 각각 따로 발전 시킬 수 있음
	Composite 컴포지트	-객체들의 관계를 트리 구조로 구성하여 부분-전체 계층을 표현하는 패턴 -사용자가 단일 객체와 복합 객체 모두 동일하게 다루도록 함 -0개, 1개 혹은 그 이상의 객체를 묶어 하나의 객체로 이용할 수 있음 -여러 개의 객체들로 구성된 복합 객체와 단일 객체를 클라이언트에서 구별 없이 다루게 해주는 패턴
	Decorator 데코레이터	-기존에 구현도디어 있는 클래스에 필요한 기능을 추가해 나가는 설계 패턴 -기능 확장이 필요할 때 객체 간의 결합을 통해 기능을 동적으로 유연하게 확장 할 수 있게 해주기 때문에 상속의 대안으로 사용됨 -기존 객체의 메서드에 새로운 행동을 추가하거나 오버라이드 할 수 있음
	Facade 퍼싸드	-복잡한 시스템에 대해 단순한 인터페이스를 제공함으로써 사용자의 시스템 간 또는 다른 시스템과의 결합도를 낮추어 시스템 구조에 대한 파악이 쉬움 -오류에 대해서 단위 별로 확인 할 수

		있게 하고 사용자 측면에서 단순한 인터페이스를 제공해 접근성을 높일 수 있음 (많은 분량의 코드에 접근할 수 있는 단순한 인터페이스를 제공) -통합된 인터페이스 제공
	Flyweight 플라이웨이트	-다수으이 객체로 생성될 경우 모두가 갖는 본질적인 요소를 클래스화하여 공유함으로써 메모리를 절약하고 클래스의 경량화를 목적으로 하는 디자인 패턴 -여러 개의 가상 인스턴스를 제공해서 메모리 절감
	Proxy 프록시	-'실제 객체에 대한 대리 객체'로 실제 객체에 대한 접근 이전에 필요한 행동을 취할 수 있게 만들고 이 점을 이용해 미리 할당하지 않아도 상관 없는 것들을 실제로 이용할 때 할당하게 하여 메모리 용량을 아낄 수 있음 -실제 객체가 드러나지 않게 하여 정보 은닉의 역할을 수행 -접근 조절, 비용 절감, 복잡도 감소를 위해 접근이 힘든 객체에 대한 대역을 제공함
행위 패턴	Command 커맨드	-하나의 추상 클래스에 메서드를 만들어 각 명령이 들어오면 그에 맞는 서브 클래스가 선택되어 실행됨 -실행된 기능을 캡슐화함으로써 주어진 여러 기능을 실행 할 수 있는 재사용성이 높은 클래스를 설계하는 패턴 -요구사항을 객체로 캡슐화
	Observer 옵저버	-어떤 클래스에 변화가 일어났을 때, 이를 감지하여 다른 클래스에 통보해주는 것 -한 객체의 상태가 바뀌면 그 객체에 의존하는 다른 객체들에게 연락이 가고 자동으로 내용이 갱신되는 패턴 -일 대 다의 의존성을 가지고 상호작용하는 객체 사이에서는 가능한 느슨하게 결합하는 디자인 패턴

	Templete Method 템플릿 메소드	-상위 클래스에서는 추상적으로 표현하고, 그 구체적인 내용은 하위 클래스에서 결정되는 디자인 패턴 -어떤 작업을 처리하는 일부분을 서브 클래스로 캡슐화해 전체 일을 수행하는 구조는 바꾸지 않으면서 특정 단계에서 수행하는 내역을 바꾸는 패턴(상위 작업의 구조를 바꾸지 않으면서 서브 클래스로 작업의 일부분을 수행)
	Mediator 중재자	-객체지향 설계에서 객체의 수각 너무 많아져 통신이 복잡해지면 느슨한 결합(객체지향에서 중요한 특성)을 해칠 수 있기 때문에 중간에서 이를 통제하고 지시할 수 있는 역할을 중재자 역할을 하는 패턴(중재자에게 요구하여 통신의 빈도를 줄임) -상호작용의 유연한 변경을 지원
	Interpreter 통역사	-언어의 다양한 해석, 구체적으로 구문을 나누고 그 분리된 구문의 해석을 맡는 클래스를 각각 작성해 여러 형태의 언어 구문을 해석할 수 있게 만드는 패턴 -문법 자체를 캡슐화해서 사용
	Iterator	-컬렉션 구현 방법을 노출시키지 않으면서 그 집합체 안에 들어있는 모든 항목에 접근할 방법을 제공하는 패턴 -내부 구조를 노출하짖 않고 복합 객체의 원소를 순차적으로 접근 가능하게 하는 행위 패턴
	State 상태	-객체 상태를 캡슐화 해서 클래스화함으로써 그것을 참조하게 하는 방식 -상태에 따라 다르게 처리할 수 있도록 행위 내용을 변경하고, 변경 시 원시 코드의 수정을 최소화 할 수 있고 유지보수의 편의성도 가짐 -객체의 상태에 따라 행위 내용을 변경
	Visitor 방문자	-각 클래스의 데이터 구조로부터 처리 기능을 분리하여 별도의 클래스를 만들어 놓고 해당 클래스의 메서드가 각 클래스를 돌아다니며 특정 작업을 수

		행하도록 만드는 패턴 -객체의 구조는 변경하지 않으면서 기능만 따로 추가하거나 확장할 때 사용하는 패턴 -특정 구조를 이루는 복합 객체의 원소 특성에 따라 동작을 수행할 수 있도록 지원
	Strategy 전략	-알고리즘 군을 정의하고(추상클래스) 같은 알고리즘을 각각 하나의 클래스로 캡슐화한 후, 필요할 때 서로 교환해서 사용할 수 있게 하는 패턴 -행위 클래스로 캡슐화해 동적으로 행위를 자유롭게 바꿀 수 있게 해주는 패턴
	Memento	-클래스 설계 관점에서 객체의 정보를 저장할 필요가 있을 때 적용하는 디자인 패턴 -Undo 기능을 개발할 때 사용 -객체를 이전 상태로 복구 시켜야 하는 경우 Undo(작업 취소) 요청 기능
	Chain of Responsibility	-정적으로 어떤 기능에 대한 처리의 연결이 하드코딩 되어 있을 때 기능 처리의 연결 변경이 불가능한데, 이를 동적으로 연결한 경우에는 다르게 처리할 수 있도록 하는 디자인 패턴 -한 요청을 2개 이상의 객체에서 처리

대칭키 암호화

DES	* 64 bit (실제 비밀키 길이는 56bit) * 16 라운드 진행 * DES를 3중으로 하여 보안성을 강화한 3DES도 존재
AES	* 128 bit / 가변 길이 키 (128/192/256) * 10/12/14 라운드 진행 * 메모리를 적게 사용하고 속도가 빨라 모바일 장비에서도 사용
IDEA	* 64 bit (키 길이는 128bit) * 8 라운드 진행
SEED	* 128 bit * 16 라운드 진행 * KISA(한국인터넷진흥원)에서 개발
ARIA	* 128 bit / 키 길이 (128/192/256) * 12/14/16 라운드 진행 * 국내 국가보안연구소를 중심으로 하여 공동으로 개발함, 정부 및 공공기관에서 범용적으로 사용
RC5	* 다양한 크기의 키, 블록, 라운드를 가짐 * 단순하고 빠르며 메모리 요구량은 낮음
WEP	* 유선 LAN에서 기대할 수 있는 수준의 무선 LAN 보안 프로토콜
TKIP	* 128 bit * IEEE 802.11 무선랜 보안에 사용된 웹 방식 보완 * WEP의 취약성 보완

공개키 암호화

RSA	* 대규모 소수의 곱셈에 기반한 알고리즘으로 소인수분해 문제의 난해성을 기반으로 함 * 키의 길이가 길고 속도가 느린편 * SSL, 공인인증서 등에 활용
ECC	* 타원 곡선의 이산로그 문제 기반, 이산대수의 난해성에 기반함 * RSA보다 적은 bit수에 동일한 성능, 키 생성 시간은 수십 배 이상 빠름 * 비트코인, 무선 통신 시스템, ElGamal에 사용
DSA	* 이산로그 문제의 난해성 기반 * 미국 NIST에서 전자서명 표준(DSS)에서 사용하기 위해 발표함 * 디지털 서명 프로토콜인 SSL/TLS, SSH, IPSec 등에서 사용

단방향 암호

MDC	* 해시 함수를 사용하여 고유한 체크섬 값을 생성 -> 원래 데이터가 변경되지 않았음을 증명

	* 전송된 파일이 변조되었는 지 여부 검사 예) SHA (NIST에서 개발), MD
MAC	* 주어진 메시지와 키를 입력으로 받아서 고정된 크기의 출력값을 생성

해시 함수

MD5	* 128 bit 해시값 가짐 * MD4는 32 bit 컴퓨터에 최적화된 해시 알고리즘
SHA	* MD를 대체하기 위해 미국 NIST에서 개발한 해시 암호

UML 다이어그램

기본 3요소: 사물(Things), 관계(Relationship), 다이어그램(Diagram)

구조적 다이어그램	Class diagram 클래스 다이어그램	-시스템 내 클래스의 정적 구조를 표현하고 시스템을 구성하는 클래스들 사이의 관계를 표현한다. -클래스와 클래스가 가지는 속성, 클래스 사이의 관계를 표현한다.
	Object diagram 객체 다이어그램	-럼바우 객체지향 분석 기법에서 객체 모델링에 활용된다. -클래스에 속한 사물들 즉 인스턴스를 특정 시점의 객체와 객체 사이의 관계로 표현한다. -객체 정보를 보여준다.
	Composite Structure diagram 복합체 구조 다이어그램	복합 구조의 클래스와 컴포넌트 내부 구조를 표현한다.
	Deployment diagram 배치 다이어그램	-결과물, 프로세스, 컴포넌트 등 물리적 요소들의 위치를 표현한다. -소프트웨어, 하드웨어, 네트워크를 포함한 실행 시스템의 물리 구조를 표현한다.
	Component diagram 컴포넌트 다이어그램	-실제 구현 모듈인 컴포넌트 간의 관계나 컴포넌트 간의 인터페이스를 표현한다. -구현 단계에서 사용한다. -컴포넌트 구조 사이의 관계를 표현한다.
	Package diagram 패키지 다이어그램	클래스나 유스케이스 등을 포함한 여러 모델 요소들을 그룹화해 패키지를 구성하고 패키지들 사이의 관계를 표현한다.
행위 다이어그램	Use Case diagram 유스케이스 다이어그램	-사용자의 요구를 분석하는 것으로 기능 모델링 작업에 사용한다. -사용자 관점에서 시스템 행위를 표현한다.
	Activity diagram 활동 다이어그램	업무 처리 과정이나 연산이 수행되는 과정을 표현한다.
	State Machine diagram 상태 머신 다이어그램	-하나의 객체가 자신이 속한 클래스의 상태 변화 혹은 다른 객체와의 상호 작용에 따라 상태가 어떻게 변화하는 지를 표현한다. -객체의 생명주기를 표현한다. 동적 행위를 모델링하지만 특정 객체만을 다룬다.
	Collaboration diagram	Sequence diagram과 같으며 모델링 공간에 제약이 없어 구조적인 면을 중시한다.
	Sequence diagram 순차 다이어그램	시스템의 동작을 정형화하고 객체의 메시지 교환을 쉽게 표현하고 시간에 따른 메시지 발생 순서를 강조한다. *요소: 생명선 (life line), 통로 (gate), 상호작용

		(Interaction Fragment), 발생 (Occurrence), 실행 (Execution), 상태 불변 (State Invariant), 상호작용(Interaction Use), 메시지 (Messages), 활성 (Activations), 객체 (Entitiy), Actor
	Interaction Overview diagram 상호작용 개요 다이어그램	여러 상호작용 다이어그램 사이의 제어 흐름을 표현한다.
	Communication diagram 통신 다이어그램	객체 사이의 관계를 중심으로 상호작용을 표현한다.
	Timing diagram 타이밍 다이어그램	객체 상태 변화와 시간 제약을 명시적으로 표현한다.

연관 관계 다중성 표현

표기	의미
1	1 개체 연결
* 또는 0..*	0이거나 그 이상 객체 연결
1..*	1이거나 1 이상 객체 연결
0..1	0이거나 1 객체 연결
1, 3, 6	1이거나 3이거나 6 객체 연결
n	n개 객체 연결
n..*	n이거나 n개 이상 객체 연결

테스트

화이트박스 테스트: 각 응용 프로그램의 내부 구조와 동작을 검사하는 소프트웨어 테스트

구문 커버리지 Statement Coverage	모든 명령문을 적어도 한 번 수행
결정 커버리지 (=선택/분기(Branch)커버리지) Decision Coverage	결정 포인트 내의 전체 조건식이 적어도 한 번은 참과 거짓의 결과를 수행
조건 커버리지 Condition Coverage	결정 포인트 내의 각 개별 조건식이 적어도 한 번은 참과 거짓의 결과를 수행
조건/결정 커버리지 Condition/Decision Coverage	전체 조건식분만 아니라 개별 조건식도 참과 거짓의 결과를 수행
변경 조건/결정 커버리지 Modified Condition/Desition Coverage	개별 조건식이 다른 개별 조건식에 영향을 받지않고 전체 조건식에 독립적으로 영향을 주도록 향상시킨 커버리지
다중 조건 커버리지 Multiple Condition Coverage	모든 개별 조건식의 모든 가능한 조합을 100% 보장하는 커버리지
기본 경로 커버리지 Base Path Coverage	수행 가능한 모든 경로를 테스트하는 커버리지
제어 흐름 테스트 Control Flow Testing	프로그램 제어 구조를 그래프 형태로 나타내어 내부 로직을 테스트
데이터 흐름 테스트 Data Flow Testing	제어 흐름 그래프에 데이터 사용현황을 추가한 그래프 테스트

블랙박스 테스트: 프로그램 외부 사용자의 요구사항 명세를 보면서 수행하는 테스트

동등분할 테스트 Equivalence Partitioning Testing	입력 데이터의 영역을 유사한 도메인별로 그룹핑하여 대푯값 테스트 케이스 도출
경곗값 분석 테스트 Boundary Value Analysis Testing	등가 분할 후 경곗값 분석에서 오류 발생확률이 높기 때문에 경곗값을 포함하여 테스트 케이스 도출
결정 테이블 테스트 Decision Table Testing	요구사항 논리와 발생조건을 테이블 형태로 나열하여, 조건과 행위를 모두 조합하는 기법
상태 전이 테스트 State Transition Testing	객체의 상태를 구분하고, 이벤트에 의해 어느 상태에서 다른 상태로 전이되는 경우의 수를 수행하는 테스트 기법

유스케이스 테스트 Use Case Testing	유스케이스를 프로세스 흐름을 기반으로 테스트 케이스를 명세화하여 수행하는 테스트 기법
분류 트리 테스트 Classification Tree Method Testing	SW의 일부 또는 전체를 트리 구조로 분석 및 표현하여 테스트 케이스를 설계하여 테스트하는 기법
페어와이즈 테스트 Pairwise Testing	테스트 데이터 값들 간에 최소한 한 번씩을 조합하는 테스트 방식
원인-결과 그래프 테스트 Cause-Effect Graphing Testing	그래프를 활용하여 입력 데이터 간의 관계 및 출력에 미치는 영향을 분석하여 효용성이 높은 테스트 케이스를 선정하여 테스트하는 기법
비교 테스트 Comparison Testing	여러 버전의 프로그램에 입력값을 넣어서 동일한 결과가 나오는지 비교해 보는 테스트 기법

테스트 목적에 따른 분류

회복 테스트 Recovery	시스템에 고의로 실패하고 정상적 복귀 여부를 테스트하는 기법
안전 테스트 Security	불법적인 소프트웨어가 접근하여 시스템을 파괴하지 못하도록 소스 코드 내의 보안적인 결함을 미리 점검하는 테스트 기법
성능 테스트 Performance	사용자의 이벤트에 시스템이 응답하는 시간, 특정 시간 내에 처리하는 업무량 등을 측정하는 테스트 기법
구조 테스트 Structure	시스템의 내부 논리 경로, 소스 코드의 복잡성을 평가하는 테스트 기법
회귀 테스트 Regression	오류를 제거하거나 수정한 시스템에서 새로 유입된 오류가 없는지 확인하는 일종의 반복 테스트 기법
병행 테스트 Parallel	변경된 시스템과 기존 시스템에 동일한 데이터를 입력 후 결과를 비교하는 테스트 기법

테스트 종류에 따른 분류

명세 기반 테스트 (블랙박스 테스트)	프로그램의 요구사항 명세서를 기반으로 테스트 케이스를 선정하여 테스트하는 기법
구조 기반 테스트 (화이트박스 테스트)	소프트웨어 내부 논리 흐름에 따라 테스트 케이스를 작성하고 확인하는 테스트 기법
경험 기반 테스트 (블랙박스 테스트)	유사 소프트웨어나 유사 기술 평가에서 테스터의 경험을 토대로, 직관과 기술 능력을 기반으로 수행하는 테스트 기법

테스트 레벨 종류

종류	설명	기법
단위 테스트 Unit Test	사용자 요구사항에 대한 단위 모듈, 서브 루틴 등을 테스트하는 단계	인터페이스 테스트 자료 구조 테스트 실행 경로 테스트 오류 처리 테스트
통합 테스트 Integration Test	단위 테스트를 통과한 컴포넌트 간의 인터페이스를 테스트하는 단계	상향식/하향식 테스트 빅뱅 테스트
시스템 테스트 System Test	개발 프로젝트 차원에서 정의된 전체 시스템 또는 제품의 동작에 대해 테스트 하는 단계	기능/비기능 요구사항 테스트
인수 테스트 Acceptance Test	계약상의 요구하상이 만족되었는지 확인하기 위한 테스트 단계	알파 테스트: 사용자-개발자 함께 확인 베타 테스트: 사용자만 확인

1. DDL 데이터정의어

DDL(Data Definition Language)

데이터를 생성, 수정, 삭제하는 등의 데이터의 전체의 골격을 결정하는 역할을 하는 언어이다.

CREATE	생성
ALTER	수정
DROP	삭제
TRUNCATE	오브젝트 내용 모든 데이터 삭제

CASCADE: 제거할 요소를 참조하는 다른 모든 개체를 함께 제거

RESTRICT: 다른 개체가 제거할 요소를 참조 중일 때는 제거를 취소

도메인을 정의하는 명령문:

```
CREATE DOMAIN 도메인명 [AS] CHAR(1)
                [DEFAULT 기본값]
                [CONSTRAINT 제약조건명 CHECK (범위값)];
```

데이터타입: SQL에서 지원하는 데이터 타입

기본값: 데이터를 입력하지 않았을 때 자동으로 입력되는 값

테이블을 정의하는 명령문:

```
CREATE TABLE 테이블명
        (속성명 데이터_타입 [DEFAULT 기본값][NOT NULL], ...
        [, PRIMARY KEY(기본키_속성명, ...)]
        [, UNIQUE(대체키_속성명, ...)]
        [, FOREIGN KEY(외래키_속성명, ...)]
                [REFERENCES 참조테이블(기본키_속성명, ...)]
                [ON DELETE 옵션]
                [ON UPDATE 옵션]
        [, CONSTRAINT 제약조건명][CHECK (조건식)];
```

PRIMARY KEY : 기본키로 사용할 속성을 지정함

UNIQUE : 대체키로 사용할 속성을 지정함, 중복값 가질 수 없음

FOREIGN KEY ~ REFERENCES : 외래키로 사용할 속성을 지정함

ON DELETE 옵션 : 참조 테이블의 튜플이 삭제 되었을 때 기본 테이블에 취해야할 사항을 지정

ON UPDATE 옵션 : 참조 속성 값이 변경 되었을 때 기본테이블에 취해야할 사항을 지정

CONSTRAINT : 제약 조건의 이름을 지정함

CHECK : 속성 값에 대한 제약조건을 정의함

CREATE VIEW

뷰를 정의하는 명령문:

<고객> 테이블에서 '주소'가 안산시인 고객들의 '성명'과 '전화번호'를 '안산고객'이라는 뷰로 정의하라

CREATE VIEW 안산고객(성명, 전화번호)
AS SELECT 성명, 전화번호
FROM 고객
WHERE 주소 = '안산시';

CREATE INDEX

인덱스를 정의하는 명령문:

CREATE [UNIQUE] INDEX 인덱스명
ON 테이블명 (속성명 [ASC|DESC][,속성명 [ASC|DESC]]
[CLUSTER];

UNIQUE
사용된 경우: 중복값이 없는 속성으로 인덱스 생성
생략된 경우: 중복값을 허용하는 속성으로 인덱스 생성
정렬 여부 지정
ASC: 오름차순 정렬
DESC: 내림차순 정렬
생략된 경우: 오름차순으로 정렬됨
CLUSTER: 사용하면 인덱스가 클러스터드 인덱스로 설정됨

ALTER TABLE

테이블에 대한 정의를 변경하는 명령문

ALTER TABLE 테이블명 ADD 속성명 데이터_타입 [DEFAULT '기본값'];
ALTER TABLE 테이블명 ALTER 속성명 [SET DEFAULT '기본값'];
ALTER TABLE 테이블명 DROP COLUMN 속성명 [CASCADE];

ADD: 새로운 속성(열)을 추가할 때 사용
ALTER: 특정 속성의 DEFAULT 값을 변경할 때 사용
DROP: 특정 속성을 삭제할 때 사용함

DROP

스키마, 도메인, 기본 테이블, 뷰 테이블, 인덱스, 제약 조건을 제거하는 명령문

DROP SCHEMA 스키마명 [CASCADE | RESTRICT];
DROP DOMAIN 도메인명 [CASCADE | RESTRICT];
DROP TABLE 테이블명 [CASCADE | RESTRICT];
DROP VIEW 뷰명 [CASCADE | RESTRICT];
DROP INDEX 인덱스명 [CASCADE | RESTRICT];

DROP CONSTRAINT 제약조건명;

CASCADE: 제거할 요소를 참조하는 다른 모든 개체를 함께 제거함
RESTRICT: 다른 개체가 제거할 요소를 참조중일 때는 제거를 취소함

2. DCL

COMMIT	명령에 의해 수행된 결과를 실제 물리적 디스크로 저장
ROLLBACK	데이터베이스 조작작업이 비정상적으로 종료되었을 때 원래 상태로 복구
GRANT	데이터베이스 사용자에게 사용권한을 부여
REVOKE	데이터베이스 사용자의 사용권한을 취소

GRANT/REVOKE
데이터베이스 관리자가 데이터베이스 사용자에게 권한을 부여하거나 취소하기 위한 명령
GRANT 권한_리스트 ON 개체 TO 사용자 [WITH GRANT OPTION];
REVOKE [GRANT OPTION FOR] 권한_리스트 ON 개체 FROM 사용자 [CASCADE];

권한 종류: ALL, SELECT, INSERT, DELETE, UPDATE 등
WITH GRANT OPTION: 부여 받은 권한을 다른 사용자에게 다시 부여할 수 있는 권한 부여
GRANT OPTION FOR: 다른 사용자에게 권한을 부여할 수 있는 권한을 취소
CASCADE: 권한 취소 시 권한을 부여받았던 사용자가 다른 사용자에게 부여한 권한도 연쇄적으로 취소함

INSERT INTO ~
(삽입문) 기본 테이블에 새로운 튜플을 삽입할 때 사용한다.
INSERT INTO 테이블명([속성명1, 속성명2, ...])
VALUES (데이터1, 데이터2, ...);

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

DELETE FROM ~
(삭제문) 기본 테이블에 있는 튜플들 중에서 특정 튜플(행)을 삭제할 때 사용한다.
DELETE
FROM 테이블명
[WHERE 조건];

-모든 레코드를 삭제할 때는 WHERE 절을 생략한다.
-모든 레코드를 삭제하더라도 테이블 구조는 남아 있기 때문에 디스크에서 테이블을 완전히 제거하는 DROP과는 다르다.
-DELETE 문은 테이블 구조나 테이블 자체는 그대로 남겨 두고, 테이블 내의 튜플들만 삭제한다.
-테이블을 완전히 제거하기 위해서는 DROP문을 사용해야 한다.

ROLLBACK

변경되었으나 아직 COMMIT 되지 않은 모든 내용들을 취소하고 데이터베이스를 이전 상태로 되돌리는 명령어
트랙잭션 전체가 성공적으로 끝나지 못하면 일부 변경된 내용만 데이터베이스에 반영되는 비일관성 상태가 될 수 있기 때문에 일부분만 완료된 트렉잭션은 롤백되어야한다.

SAVEPOINT
트랙잭션 내에 ROLLBACK 할 위치인 저장점을 지정하는 명령어
저장점을 지정할 때는 이름을 부여한다.
ROLLBACK 할 때 지정된 저장점까지의 트랙잭션 처리내용이 모두 취소된다.

3. DML
저장된 데이터를 실질적으로 관리하는 데 사용되는 언어

INSERT INTO
(삽입문) 기본적으로 테이블에 새로운 튜플을 삽입할 때 사용한다
INSERT INTO 테이블명([속성명1, 속성명2, ...])
VALUES(데이터1, 데이터2);

DELETE FROM
(삭제문) 기본 테이블에 있는 튜플 중에서 특정 튜플(행)을 삭제할 때 사용한다.
DELETE FROM 테이블명 [WHERE 조건];

UPDATE ~ SET ~
(갱신문) 기본 테이블에 있는 튜플 중에서 특정 튜플의 내용을 변경할 때 사용한다.
UPDATE 테이블명
SET 속성명 = 데이터[, 속성명=데이터, ...]
[WHERE 조건];

SELECT
SELECT [PREDICATE] [테이블명.]속성명 [AS 별칭][, [테이블명.]속성명, ...]
[, 그룹함수(속성명) [AS 별칭]]
[, WINDOW함수 OVER (PARTITION BY 속성명1, 속성명2, ...
ORDER BY 속성명3, 속성명4, ...)]
FROM 테이블명[, 테이블명, ...]
[WHERE 조건]
[GROUP BY 속성명, 속성명, ...]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];

PREDICATE: 검색할 튜플 수를 제한하는 명령어를 기술함
DISTINCT: 중복된 튜플이 있으면, 그 중 첫 번째 한 개만 표시함
속성명: 검색하여 불러올 속성(열) 또는 속성을 이용한 수식을 지정함
AS: 속성이나 연산의 이름을 다른 이름으로 표시하기 위해 사용함
FROM 절: 검색할 데이터가 들어있는 테이블 이름을 기술함
WHERE 절: 검색할 조건을 기술함
ORDER BY 절: 데이터를 정렬하여 검색할 때 사용함

그룹 함수

GROUP BY 절에 지정된 그룹별로 속성의 값을 집계할 때 사용된다.

함수	기능
COUNT(속성명)	그룹별 튜플 수를 구하는 함수
SUM(속성명)	그룹별 합계를 구하는 함수
AVG(속성명)	그룹별 평균을 구하는 함수
MAX(속성명)	그룹별 최대별을 구하는 함수
MIN(속성명)	그룹별 최소값을 구하는 함수
STDDEV(속성명)	그룹별 표준편차를 구하는 함수
VARIANCE(속성명)	그룹별 분산을 구하는 함수
ROLLUP(속성명, 속성명, ...)	-인수로 주어진 속성을 대상으로 그룹별 소계를 구하는 함수 -속성으이 개수가 n개이면 n+1 레벨까지, 하위 레벨에서 상위 레벨 순으로 데이터가 집계됨.
CUBE(속성명, 속성명, ...)	-ROLLUP과 유사한 형태이지만, CUBE는 인수로 주어진 속성을 대상으로 모든 조합의 그룹별 소계를 구함 -속성의 개수가 n개이면 2^n 레벨까지, 상위 레벨에서 하위 레벨 순으로 데이터가 집계됨.

WINDOW 함수

-GROUP BY 절을 이용하지 않고, 함수의 인수로 지정한 속성의 값을 집계한다.

-함수의 인수로 지정한 속성이 집계할 범위가 죄는데, 이를 WINDOW라고 부른다.

*ROW_NUMBER(): 윈도우별로 각 레코드에 대한 일련번호를 반환한다.

*RANK(): 윈도우별로 순위를 반환하며, 공동 순위를 반영한다.

*DENSE_RANK(): 윈도우별로 순위를 반환하며, 공동 순위를 무시하고 순위를 부여한다.

집합연산자를 이용한 통합 질의

UNION: 합집합(중복행 제거)
-두 SELECT 문의 조회 결과를 통합하여 모두 출력함
-중복된 행은 한 번만 출력함
UNION ALL: 합집합(중복된 행 그대로 출력)
-두 SELECT 문의 조회 결과를 통합하여 모두 출력함
-중복된 행도 그대로 출력함
INTERSECT: 교집합
-두 SELECT 문의 조회 결과 중, 공통된 행만 출력함
EXCEPT: 차집합
-첫 번째 SELECT 문의 조회 결과에서 두 번째 SELECT 문의 조회 결과를 제외한 행을 출력함

JOIN
-2개의 릴레이션에서 연관된 튜플들을 결합하여, 하나의 새로운 릴레이션을 반환한다.
-일반적으로 FROM 절에 기술하지만, 릴레이션이 사용되는 곳 어디에나 사용할 수 있다.
-크게 INNER JOIN과 OUTER JOIN으로 구분된다.

INNER JOIN
-일반적으로 EQUI JOIN과 NON-EQUI JOIN으로 구분된다.
-조건이 없는 INNER JOIN을 수행하면 CROSS JOIN과 동일한 결과를 얻을 수 있다.
*CROSS JOIN(교차 조인)
-조인하는 두 테이블에 있는 튜플들의 순서쌍을 결과로 반환한다.
-교차 조인의 결과로 반환되는 테이블의 행의 수는 두 테이블의 행 수를 곱한 것과 같다.

EQUI JOIN
-JOIN 대상 테이블에서 공통 속성을 기준으로 '='(equal) 비교에 의해 같은 값을 가지는 행을 연결하여 결과를 생성하는 JOIN 방법
-EQUI JOIN에서 JOIN 조건이 '='일 때 동일한 속성이 2번 나타나게 되는데, 이 중 중복된 속성을 제거하여 같은 속성을 한 번만 표기하는 방법을 NATURAL JOIN이라고 한다.
-EQUI JOIN에서 연결 고리가 되는 공통 속성을 JOIN 속성이라고 한다.

WHERE 절을 이용한 EQUI JOIN의 표기 형식
SELECT [테이블명1.]속성명, [테이블명2.]속성명, ...
FROM 테이블명1, 테이블명2, ...
WHERE 테이블명1.속성명 = 테이블명2.속성명;

NATURAL JOIN 절을 이용한 EQUI JOIN의 표기형식
SELECT [테이블명1.]속성명, [테이블명2.]속성명, ...
FROM 테이블명1 NATURAL JOIN 테이블명2;

JOIN ~ USING 절을 이용한 EQUI JOIN의 표기 형식
SELECT [테이블명1.]속성명, [테이블명2.]속성명, ...
FROM 테이블명1 JOIN 테이블명2 USING(속성명);

NON-EQUI JOIN
*JOIN 조건에 '='조건이 아닌 나머지 비교 연산자를 사용하는 JOIN 방법
SELECT [테이블명1.]속성명, [테이블명2.]속성명, ...
FROM 테이블명1, 테이블명2, ...
WHERE (NON-EQUI JOIN 조건);

OUTER JOIN
-릴레이션에서 JOIN 조건에 만족하지 않는 튜플도 결과로 출력하기 위한 JOIN 방법
-LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN이 있다.

LEFT OUTER JOIN
-INNER JOIN의 결과를 구한 후, 우측 항 릴레이션의 어떤 튜플과도 맞지 않는 좌측 항의 릴레이션에 있는 튜플들에 NULL 값을 붙여서 INNER JOIN의 결과에 추가한다.

RIGHT OUTER JOIN
-INNER JOIN의 결과를 구한 후, 좌측 항 릴레이션의 어떤 튜플과도 맞지 않는 우측 항의 릴레이션에 있는 튜플들에 NULL 값을 붙여서 INNER JOIN의 결과에 추가한다.

FULL OUTER JOIN
-LEFT OUTER JOIN과 RIGHT OUTER JOIN을 합쳐 놓은 것
-INNER JOIN의 결과를 구한 후, 좌측 항의 릴레이션의 튜플들에 대해 우측 항의 릴레이션의 어떤 튜플과도 맞지 않는 튜플들에 NULL 값을 붙여서 INNER JOIN의 결과에 추가한다.
+)그리고 유사하게 우측 항의 릴레이션의 튜플들에 대해 좌측 항의 릴레이션으이 어떤 튜플과도 맞지 않는 튜플들에 NULL 값을 붙여서 INNER JOIN의 결과에 추가한다.

조건 연산자

비교연산자

연산자	=	<>	>	<	>=	<=
의미	같다	같지 않다	크다	작다	크거나 같다	작거나 같다

논리 연산자
NOT, AND, OR

LIKE 연산자
대표 문자를 이용해 지정된 속성의 값이 문자 패턴과 일치하는 튜플을 검색하기 위해 사용된다.

대표 문자	%	_	#
의미	모든 문자를 대표함	문자 하나를 대표함	숫자 하나를 대표함