34. 자료 구조
자료 구조
자료 구조의 분류

선형 구조
리스트
선형 리스트
연결 리스트
비선형 구조
트리
그래프
자료 구조의 활용

정렬
집합된 데이터 레코드를 일정 기준으로 재배열하는 것.
오름차순, 내림차순
검색
저장된 데이터 레코드 중 원하는 값을 빠르게 찾는 것
인덱스(Index)
데이터베이스 성능에 많은 영향을 주는 DBMS 의 구성 요소로, 테이블과 클러스터에 연관되어 독립적인 저장 공간을 보유하며, 데이터베이스에 저장된 자료를 더욱 빠르게 조회하기 위하여 별도로 구성한 순서 데이터를 말한다. (ex. 책의 맨 뒤에 빠르게 찾기 등)
B-트리 인덱스는 분기를 목적으로 하는 Branch Block 을 가지고 있다.
BETWEEN 등 범위(Range) 검색에 활용될 수 있다.
파일 편성
파일에서 레코드의 물리적인 배열 방법이다.
선형 자료 구조
리스트

선형 리스트(Linear List)
배열(Array)과 같이 연속되는 기억 장소에 저장되는 리스트이다.
가장 간단한 데이터 구조 중 하나로 데이터 항목을 추가 삭제하는 것이 불편하다.
연결 리스트(Linked List)
노드의 포인터 부분을 서로 연결시킨 리스트로, 연속적인 기억 공간이 없어도 저장이 가능하다.

노트의 삽입/삭제가 용이하며 포인터를 위한 추가 공간이 필요하므로 기억 공간이 많이 소요된다.

스택

리스트의 한쪽 끝에서만 자료의 삽입과 삭제가 이루어 지는 자료 구조이다.
가장 나중에 삽입된 자료가 가장 먼저 삭제되는 후입선출(LIFO, Last In First Out) 방식이다.(PUSH and POP)
마지막 삽입된 자료의 위치를 Top 이라 하고, 가장 먼저 삽입된 자료의 위치를 Bottom 이라고 한다.
스택 가능 : 메모리상에서 프로그램의 복귀 주소와 변수 사이에 특정 값을 저장해 두었다가 그 값이 변경되었을 경우 오버플로우 상태로 가정하여 프로그램 실행을 중단하는 기술이다.
스택 응용 분야
인터럽트 처리, 수식의 계산, 0-주소 지정 방식
재귀호출, 후위 표현(Post-fix expression)의 연산, 깊이 우선 탐색
스택의 삽입 알고리즘

```
if TOP >= n then call Stack-Full;
else TOP = TOP + 1;
Stack(TOP) = Data;
end Insert
```

스택의 삭제 알고리즘

```
if TOP = 0
then Underflow
Else
Remove S(TOP)
TOP = TOP - 1
```

스택의 오버플로 알고리즘

```
TOP <- TOP + 1
if TOP > n then goto AA
else Stack(TOP) <- item
```

큐(Queue)

자료의 삽입 작업은 선형 리스트의 한쪽 끝에서, 제거 작업은 다른 쪽 끝에서 수행되는 자료 구조이다.
가장 먼저 삽입된 자료가 가장 먼저 삭제되는 선입선출(FIFO: First In First Out) 방식이다.
큐의 응용 분야 : 운영체제의 작업 스케줄링 등에서 응용된다.

덱(Deque)

자료의 삽입과 삭제가 리스트의 양쪽 끝에서 이루어지므로 두 개의 포인터를 사용하는 자료 구조이다.
스택과 큐를 복합한 형태이다.
입력 제한 덱을 Scroll, 출력 제한 덱을 Shelf 라고 한다.
35. 비선형 구조
트리
트리(TREE)의 정의

그래프(Graph)의 특수한 형태로써 노드(Node)와 브랜치(Branch)를 이용하여 사이클을 이루지 않도록 구성한 자료 구조이다.
트리 관련 용어

노드(Node) 트리의 기본 구성 요소
근노드(Root Node) 가장 상위에 위치한 노드
레벨(Level) 근노드를 기준으로 특정 노드까지의 경로 길이
조상 노드
(Ancestors Node) 어떤 노드에서 근노드에 이르는 경로상의 모든 노드
부모 노드
(Parent Node) 어떤 노드에 연결된 이전 레벨의 노드
자식 노드
(Child Node) 어떤 노드에 연결된 다음 레벨의 노드
형제 노드
(Brother Node) 같은 부모를 가진 노드
깊이
(Depth) 트리의 최대 레벨
차수
(Degree) 어떤 노드에 연결된 자식 노드의 수
단말 노드
(Terminal Node) 트리의 제일 마지막에 위치한 노드(차수=0)
트리의 차수(Degree) 트리의 노드 중 가장 큰 차수
이진 트리(Binary Tree)

차수(Degree)가 이 이하인 노트들로만 구성된 트리
이 트리의 레벨 n 에서 최대 노드의 수 : 2^n-1
깊이(레벨)가 4 인 트리의 최대 노드 수는 2^4-1 로, 15 이다.
이진 트리의 구조

정이진 트리 : 첫 번째 레벨부터 마지막 레벨까지 모두 2 개씩 채워진 트리를 말한다.
전이진 트리 : 전이진 트리에서 한쪽 방향 노드가 아예 존재하지 않는 트리를 말한다.
사향 이진 트리 : 근노드로부터 한쪽 방향으로만 기울어진 트리를 말한다.
이진 트리의 운행법(Traversal)

전위(Preorder) 운행 : Root => Left => Right
중위(Inorder) 운행 : Left => Root => Right
후위(Postorder) 운행 : Left => Right => Root
수식의 표기법

표기 순서 비고
전위(Prefix) 표기법 연산자=>피연산자=>피연산자 +AB
중위(Infix) 표기법 피연산자=>연산자=>피연산자 A+B
후위(Postfix) 표기법 피연산자=>피연산자=>연산자 AB+
그래프

정점(Vertex)과 간선(Edge)의 집합으로 이루어지는 자료 구조.
표현 방법 : 인접 행렬(Adjacency Matrix)
신장 트리(Spanning Tree) : 간선들이 사이클을 이루지 않도록 정점들을 연결시킨 그래프이다.
종류 : 방향 그래프, 무방향 그래프, 완전 그래프, 부 그래프
n 개의 노드로 구성된 무방향 그래프의 최대 간선 수는 n(n-1)/2 개다.
제어 흐름 스래프에서 순환 복잡도 : V(G) - E(화살표 수) - N(노드 수) + 2
인접 행렬(Adjacency Matrix)

방향 그래프에서 ViVj 관계를 나타내는 행렬의 원소를 Aij 라고 할 때, 방향 간선이 있으면 행렬의 Aij = 1, 방향 간선이 없으면 행렬의 Aij = 0 으로 나타낸다.
무방향 그래프에서 Vi 와 Vj 가 서로 인접하면 Aij = 1, 서로 인접하지 않으면 Aij = 0 으로 나타낸다.
여기서 i, j 는 첨자 (A_3의 3 과 같음)
36. 정렬
정렬(Sort)
정렬 알고리즘 선택 시 고려사항 : 데이터의 양, 초기 데이터의 배열 상태, 키 값들의 분포 상태, 사용 컴퓨터 시스템의 특성
종류 : 내부 정렬(장치에서 정렬이 이루어짐), 외부 정렬(보조 기억 장치에서 정렬이 이루어짐)

내부 정렬
삽입 정렬(Insertion Sort)

정렬된 파일에 새로운 하나의 레코드를 순서에 따라 삽입시켜 정렬하는 방법
선택 정렬(Selection Sort)

n 개의 레코드 중에서 최소값(또는 최대값)을 찾아 1st 레코드 위치에 놓고, 나머지 (n-1) 개의 레코드 중에서 최소값(또는 최대값)을 찾아 2nd 레코드 위치에 놓는 방법을 반복하여 정렬하는 방법이다.
최대, 최소, 평균 시간 복잡도 : $O(n^2)$
병합 정렬(2-Way Merge Sort)

두 개의 키들을 한 쌍으로 하여 각 쌍에 대해 순서를 정한다.
순서대로 정렬된 각 쌍의 키들을 합병하여 하나의 정렬된 서브 리스트로 만든다.
최대, 최소, 평균 시간 복잡도 : $O(n \log_2 n)$
퀵 정렬(Quick Sort)

레코드의 많은 자료 이동을 없애고 하나의 파일을 부분적으로 나누어가면서 정렬하는 방법으로, 키를 기준으로 작은 값은 왼쪽에 큰 값은 오른쪽에 모이도록 서로 교환시키는 부분 교환 정렬법이다.
최대, 최소, 평균 시간 복잡도 : $O(n \log_2 n)$, 약=$O(n^2)$
힙 정렬(Heap Sort)

전이진 트리를 이용하여 정렬하는 방법이다.
정렬한 입력 레코드들로 힙을 구성하고 가장 큰 키값을 갖는 루트 노드를 제거하는 과정을 반복하여 정렬하는 기법이다.
평균 수행 시간 복잡도는 $O(n \log_2 n)$이고, 최악의 수행 시간 복잡도는 $O(\log_2 n)$이다.
입력 자료의 레코드를 완전 이진 트리(Complete Binary Tree)로 구성한다.
37. 검색과 해싱
검색
검색(Search)의 정의

기억 공간 내 기억된 자료 중에서 주어진 조건을 만족하는 자료를 찾는 것이다.
검색 방식의 종류

이분 검색(Binary Search, 이진 검색)
이분 검색을 실행하기 위한 전제 조건은 자료가 순차적으로 정렬되어 있어야 한다.
탐색 효율이 좋고 탐색 시간이 적게 소요된다.

비교 횟수를 거듭할 때마다 검색 대상이 되는 데이터의 수가 절반으로 줄어든다.
선형 검색(Linear Search)
순차 검색(Sequential Search)이라고도 한다.
주어진 자료에서 원소를 첫 번째 레코드부터 순차적으로 비교하면서 해당키 값을 가진 레코드를 찾아내는 가장 간단한 검색 방법이다.
데이터를 특별히 조직화할 필요가 없고 다양한 상황에서도 사용될 수 있는 장점이 있지만 N 개의 입력 자료에 대해서 평균적으로 (n+1)/2 번의 비교를 해야 하므로 비효율적이다.
피보나치 검색(Fibonacci Search)
이진 검색과 비슷한 원리로, 비교 대상 기준을 피보나치 수열로 결정한다.
피보나치 수열 : 1,2,3,5,8,11~로 앞의 두 수의 합이 다음 번 값이 된다.
블록 검색
전체 레코드를 일정한 블록으로 분리한 뒤 각 블록 내의 키값을 순서대로 비교하여 원하는 값을 찾는 기법이다.
이진 트리 검색
레코드를 2 진 트리로 구성하여 검색하는 방식으로 데이터를 입력하는 순서대로 첫 번째 값을 근노드로 지정하고 근노드보다 작으면 왼쪽, 크면 오른쪽에 연결하여 구성한다.
해싱
해싱(Hashing)의 정의

해싱 함수(Hashing Function)를 이용하여 레코드키에 대한 해시 테이블(Hash Table) 내의 홈 주소(Home Address)를 계산하여 주어진 레코드에 접근하는 방식이다.
직접 접근(Direct Access Method) 파일을 구성할 때 사용된다.
속도는 가장 빠르지만 충돌 현상 시 오버플로우 해결의 부담이 가중되며, 많은 기억 공간을 요구한다.
해싱 함수의 종류

제산 방법(Division Method)
해싱 함수 기법에서 키값을 양의 정수인 소수로 나누어 나머지를 홈주소로 취하는 방법이다.
중간 제곱 방법(Mid-Square Method)
레코드 키값을 제곱하고 나서 그 중간 부분의 값을 주소로 계산하는 방법이다.
해시 테이블의 크기에 따라서 중간 부분의 적당한 자릿수를 선택할 수 있다.
비트 단위로 n 자릿수를 중간 위치 자릿수로 가정하면 해시 테이블의 크기는 2n 이다.
중첩 방법(Folding Method)
해싱 함수 중 주어진 키를 여러 부분으로 나누고, 각 부분의 값을 더하거나 배타적 논리합(XOR: Exclusive OR) 연산을 통하여 나온 결과로 주소를 취하는 방법이다.

기수 변환 방법(Radix Conversion Method)
해싱 함수 기법 중 어떤 진법으로 표현된 주어진 레코드 키값을 다른 진법으로 간주하고 키값을 변환하여 홈주소로 취하는 방식이다.
계수 분석 방법(Digit Analysis Method)
주어진 모든 키값들에서 그 키를 구성하는 자릿수들의 분포를 조사하여 비교적 고른 분포를 보이는 자릿수들을 필요한 만큼 택하는 방법을 취하는 해싱 함수 기법이다.
오버플로 해결 방법

선형 개방 주소법(Linear Open Addressing)
선형 검색 방식이라고도 한다.
해싱에서 충돌이 일어난 자리에서 그다음 버킷들을 차례로 하나씩 검색하여 최초로 나오는 빈 버킷에 해당 데이터를 저장하는 방법으로 저장할 데이터가 적을 때 유리하다.
포인터와 추가적 저장 공간이 필요 없다. 삽입/삭제 시 오버헤드가 적다.
폐쇄 주소 방법
버킷 내에 연결 리스트(Linked List)를 할당하여, 버킷에 데이터를 삽입하다가 해시 충돌이 발생하면 연결 리스트로 데이터들을 연결하는 방식이다.
해시 체이닝(Hash Chainning) 기법이라고도 한다.
연결 리스트만 사용하면 되기 때문에 개방 주소법과 계산식을 사용할 필요성이 낮다.
해시 테이블이 채워질수록 Lookup 성능 저하가 발생할 수 있다.
재해싱
충돌이 발생하면 새로운 해시 함수를 적용하여 새로운 홈 주소를 계산한다.
해싱 관련 용어
동의어(Synonym)

해싱에서 동일한 홈주소로 인하여 충돌이 일어난 레코드들의 집합을 의미한다.
슬롯(Slot)

한 개의 레코드를 저장할 수 있는 공간으로 n 개의 슬롯이 모여 하나의 버킷을 형성한다.
충돌(Collision)

레코드를 삽입할 때 2 개의 상이한 레코드가 똑같은 버킷으로 해싱되는 것을 의미한다.
해싱은 충돌(Collision)이 발생하면 항상 오버플로우(Overflow)가 발생한다.
버킷(Bucket)이 여러 개의 슬롯(Slot)으로 구성될 때에는 충돌(Collision)이 발생하여도 오버플로우(OverFlow)가 발생하지 않을 수 있다.
38. 인덱스 구조와 파일 편성
인덱스
인덱스를 통하여 레코드를 빠르게 접근할 수 있다.

데이터베이스의 물리적 구조와 밀접한 관계가 있다.
레코드의 삽입/삭제가 자주 발생할 경우 인덱스의 개수를 최소화하는 것이 효율적이다.
인덱스 구성 방법
B 트리

근노드와 단말 노드를 제외한 모든 노드가 최소 m/2, 최대 m 개의 서브 트리를 가지는 구조이다.
한 노드에 있는 키값은 오름차순을 유지한다.
근노드로부터 탐색, 추가, 삭제가 이루어진다.
B+ 트리

B 트리의 추가, 삭제 시 발생하는 노드의 분열과 합병 연산 과정을 줄일 수 있는 구조이다.
가장 널리 사용되는 인덱스 구조이고 레코드 삽입, 삭제 시에도 성능이 보장된다.
트라이(Trie) 색인

키 탐색을 위해 키값을 직접 표현하는 것이 아니라 키를 구성하는 문자나 숫자 자체의 순서로 키값을 구성하는 구조이다.
삽입, 삭제 시 노드의 분열, 병합이 발생하지 않는다.
문자의 함수로 트라이 차수의 키값을 표현한다.
파일 편성 방법
순차 파일(Sequentail File)

입력되는 데이터의 논리적 순서에 따라 물리적으로 연속된 위치에 순차적으로 기록하는 방식이다.
색인 순차 파일의 구성

기본 영역 : 데이터 레코드를 지정하는 부분
색인(Index) 영역 : 기본 영역에 인덱스가 저장되는 부분
구성
트랙 인덱스
트랙 인덱스
마스터 인덱스
오버플로우 영역
한 블록 내에 레코드들이 모두 영역을 차지하여 추가적인 레코드 입력을 처리할 수 없을 때 블록을 할당받아 이를 연결시키는 부분이며 실린더 오버플로우 영역과 독립 오버플로우 영역으로 구성된다.

VSAM 파일(Virtual Storage AccessMethod File) - 동적 인덱스 방법을 이용한 색인 순차 파일이다. - 기본 영역과 오버플로우 영역을 구분하지 않는다. - 레코드를 삭제하면 그 공간을 재사용할 수 있다. - 레코드 저장은 제어 구간에서 이루어진다. - 제어 구간 단위별 그룹을 제어 영역이라 한다. - 제어 영역에 대한 인덱스 저장은 순차 세트, 순차 세트의 상위 인덱스, 인덱스 세트 등이 있다.

직접 파일(Direct File)

해싱 함수를 계산하여 물리적 주소에 직접 접근하는 방식으로 레코드를 임의 물리적 기억 공간에 기록한다.
특정 레코드에 접근하기 위해서 디스크의 물리적 주소로 변환할 수 있는 해싱 함수를 사용하는 방식이다.
속도가 빠르고, 랜덤 처리에 적합하다.
기억 공간 효율이 떨어진다.
역파일(Inverted File)

특정 파일을 여러 개의 색인으로 만들고 항목별 특성에 맞게 작업하도록 구성한 구조이다.
파일 또는 데이터베이스에서 레코드를 빨리 검색하기 위해 별도 인덱스 파일을 만들어 두며 인덱스 파일에는 키 필드의 값과 그 키값을 가지는 레코드에 대한 포인터들이 저장된다.
검색 속도가 빠르다.
데이터 파일에 접근하지 않아 질의응답 시간이 줄어들고, 처리가 비교적 쉽다.
정적 인덱싱과 동적 인덱싱
정적 인덱싱-색인 순차 파일 방식

데이터 파일에 레코드가 삽입, 삭제되면 인덱스 내용은 변하지만 인덱스 구조는 정적으로 변하지 않는 구조를 말한다.
인덱스 부분과 데이터 부분을 별개의 파일로 구성한다.
동적 인덱싱-가상 기억 접근 방식

데이터 파일에 레코드가 삽입되면서 삽입될 레코드를 위해 미리 빈 공간을 준비하는 방법을 말한다.
레코드가 블록에 가득 차면 동적으로 분열된다.
인덱스 부분과 데이터 부분을 별개의 파일로 구성한다.
39. 데이터베이스의 개념과 DBMS
자료 처리
자료와 정보

자료(Data) : 현실 세계로부터 단순한 관찰이나 측정을 통하여 수집된 사실이나 값
정보(Information) : 자료를 처리하여 얻은 결과로써, 의사 결정을 하기 위한 값이다.
정보 시스템

한 조직체의 데이터를 바탕으로 의사 결정에 필요한 정보를 추출하고 생성하는 시스템이다.
사용 목적에 따라 인사 정보 시스템, 행정 정보 시스템 등으로 구분된다.
자료 처리 시스템의 종류

일괄 처리 시스템 : 일정 시간 동안 수집된 변동 자료를 컴퓨터의 입력 자료로 만들었다가 필요한 시점에 이 자료들을 입력하여 실행한 후 그 결과를 출력시켜 주는 방식의 시스템이다. (ex. 급여 관리, 세무 관리 등)
온라인 처리 시스템 : 자료 발생 즉시 해당 자료를 처리하여 결과를 출력시켜 주는 방식의 시스템이다. (ex. 좌석 예약, 주식 거래 등)
분산 처리 시스템 : 물리적으로 분리된 각각의 데이터베이스를 네트워크로 연결하여 실사용자들이 각 시스템이 하나인 것처럼 사용할 수 있도록 자원해 주는 시스템이다.
데이터베이스(Database)의 정의

통합된 데이터(Integrated Data) : 각 사용자의 데이터를 한 곳에 모아 통합한 데이터이다.
저장된 데이터(Stored Data) : 데이터베이스는 컴퓨터 하드웨어 저장 장치에 저장되어 있는 데이터이다.
운영 데이터(Operational Data) : 데이터베이스는 어떤 조직의 고유 기능을 수행하기 위해 반드시 필요한 데이터이다.
공용 데이터(Shared Data) : 데이터베이스를 여러 사용자가 공동 소유/관리/활용하는 데이터이다.
데이터베이스의 특성

실시간 접근성(Real Time Accessibility) : 수시적이고 비정형적인 질의에 대하여 실시간 처리로 응답할 수 있어야 한다.
내용에 의한 참조(Content Reference) : 데이터베이스의 데이터는 그 주소나 위치에 의해 참조되는 것이 아니라 내용을 참조한다.
동시 공유(Concurrent Sharing) : 같은 내용의 데이터를 여러 사람이 동시에 공용할 수 있다.
계속적 변화(Continuous Evolution) : 데이터베이스는 데이터의 삽입, 삭제, 갱신으로 내용이 계속적으로 변한다.
데이터베이스 시스템의 구성

DBMS, 스키마, 데이터베이스 언어, 데이터베이스 사용자.
DBMS(DataBase Management System, 데이터베이스 관리 시스템)의 정의

종속성과 중복성의 문제를 해결하기 위해 제안된 시스템이다.
응용 프로그램과 데이터의 중재자로서 모든 응용 프로그램들이 데이터베이스를 공유할 수 있도록 관리한다.
데이터베이스의 구성, 접근 방법, 관리 유지에 대한 모든 책임을 진다.
DBMS 의 필수 기능

정의 기능(Definition Facility)
데이터베이스 구조를 정의한다.
데이터의 논리적 구조와 물리적 구조 사이에 변환이 가능하도록 두 구조 사이의 사상(Mapping)을 명시한다.
조작 기능(Manipulation Facility)
데이터베이스에 접근하여 데이터의 검색/삽입/삭제/갱신 등의 연산 적업을 하기 위한 사용자와 데이터베이스 사이의 인터페이스 수단을 제공한다.
제어 기능(Control Facility)
데이터베이스에 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되어 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 제어를 할 수 있도록 한다.
DBMS 의 장/단점

장점
데이터의 중복 및 종속성 최소화
데이터 공유
데이터 무결성 및 일관성 유지
데이터 보안 보장 용이
단점
예비와 회복 기법이 어려움
데이터베이스 전문가 부족
시스템이 복잡하고, 전산화 비용 증가
데이터 웨어하우스

기간 업무 시스템에서 추출되어 새로이 생성된 데이터베이스로서 의사결정 지원을 위한 주제 지향적, 통합적, 시계열적(Historical), 비휘발적인 데이터의 집합이다.
OLAP(On-Line Analytical Processing) : 대용량 데이터를 고속으로 처리하며 쉽고 다양한 관점에서 추출, 분석할 수 있도록 지원하는 데이터 분석 기술이다.
OLAP 연산 종류 : Roll-Up, Drill-Down, Dicing, Slicing
데이터베이스 용어

빅데이터 : 데이터의 생성 양, 주기, 형식 등이 기존 데이터에 비해 매우 크기 때문에, 종래의 방법으로는 수집/저장/검색/분석이 어려운 방대한 데이터이다.
데이터 마이닝 : 데이터웨어하우징에서 수집되고 분석된 자료를 사용자에게 제공하기 위해 분류 및 가공되는 요소 기술이다.
Hadoop : 일반 컴퓨터로 가상화된 대형 스토리지를 구현한다. 그 안에 보관된 거대한 데이터 세트를 병렬로 처리할 수 있도록 빅데이터 분산 처리를 돕는 자바 소프트웨어 오픈소스 프레임워크이다.
40. 데이터베이스의 구성
데이터베이스의 구성
스키마(Schema)

데이터베이스의 구조(개체, 속성, 관계)에 대한 정의이다.
스키마의 3 계층

외부 스키마(External Schema)
사용자나 응용 프로그래머가 접근할 수 있는 정의르 기술한다.
개념 스키마(Conceptual Schema)
데이터베이스 전체를 정의한 것으로 데이터 개체, 관계, 제약조건, 접근 권한, 무결성 규칙 등을 명세한 것이다.
범기관적 입장에서 데이터베이스를 정의한다.
내부 스키마(Internal Schema)
데이터의 실제 저장 방법을 기술한다.
물리적 저장 장치의 입장에서 본 데이터베이스 구조로써 실제로 데이터베이스에 저장될 레코드의 형식을 정의하고 저장 데이터 항목의 표현 방법, 내부 레코드의 물리적 순서 등을 나타낸다.
데이터베이스 언어(Database Language)

데이터 정의어(DDL : Data Definition Language)
데이터베이스의 객체들, 즉 테이블, 뷰, 인덱스 등에 대한 구조인 스키마를 정의하고 변경하며 삭제할 수 있는 기능이 있다.

논리적 데이터 구조와 물리적 데이터 구조 간의 사상 정의이다.
번역한 결과가 데이터 사전에 저장된다.
데이터 조작어(DML : Data Manipulation Language)
사용자와 데이터베이스 관리 시스템 간의 인터페이스를 제공한다.
데이터의 검색/삽입/삭제/변경을 구행한다.
데이터 제어어(DCL : Data Control Language)
불법적인 사용자로부터 데이터를 보호한다.
무결성을 유지한다.
데이터 회복 및 병행 제어를 수행한다.
데이터베이스 사용자

데이터베이스 관리자(DBA : Database Administrator)
데이터베이스를 구축하는 책임자이다.
DBMS 를 관리한다.
사용자 요구 정보 결정 및 데이터를 효율적으로 관리한다.
백업 및 회복 전략을 정의한다.
행정적 책임을 가지고 있다.
시스템 감시 및 성능을 분석한다.
데이터 사전을 구성한다.
데이터 접근 권한과 회복 절차를 수립한다.
데이터베이스의 구성 요소 결정과 내장 저장 구조를 정의 및 수정한다.
응용 프로그래머
DBA 가 설계한 데이터베이스를 기반으로 소프트웨어 개발 툴을 이용하여 사용자에게 제공할 소프트웨어를 작성하는 업무를 담당한다.
COBOL, PASCAL, C, JAVA 등의 개발 언어를 사용한다.
일반 사용자
응용 프로그램, 질의어 등을 통하여 데이터베이스에 직접 접근하여 자원을 사용한다.
데이터베이스 모델의 종류
데이터 모델의 개념

현실 세계를 데이터베이스에 표현하는 중간 과정, 즉 데이터베이스 설계 과정에서 데이터의 구조를 표현하기 위해 사용되는 도구이다.
데이터 모델의 구성 요소

데이터 구조(Structure) : 데이터 구조 및 정적 성질을 표현한다.

연산(Operations) : 데이터의 인스턴스에 적용 가능한 연산 명세와 조작 기법을 표현한다.
제약조건(Constraints) : 데이터의 논리적 제한 명시 및 조작의 규칙이다.
데이터 모델의 구분

데이터베이스 모델
개념적 모델
ERD(Entity Relationship Diagram)
논리적 모델
계층형
네트워크(망)형
관계형
객체지향형
개념적 데이터 모델

속성들로 기술된 개체 타입과 이 개체 타입 간의 관계를 이용하여 현실 세계를 표현하는 방법이다.
E-R 모델(Entity-Relationship 모델 , 개체-관계 모델)
대표적인 개념적 데이터 모델이다.
개체 타입과 이들 간의 관계 타입을 이용하여 현실 세계를 개념적으로 표현한 방법이다.
E-R 다이어그램 : E-R 모델을 그래프 방식으로 표현하였다.
기호 의미
사각형 개체(Entity)
마름모 관계(Relationship)
타원 속성(Attribute)
실선 개체 타입과 속성을 연결
논리적 데이터 모델

필드로 기술된 데이터 타입과 이 데이터 타입 간의 관계 현실 세계를 표현하는 방법이다.
종류
관계형 데이터 모델 : 데이터베이스를 테이블(Table)의 집합으로 표현한다.
계층형 데이터 모델 : 데이터베이스를 트리(Tree) 구조로 표현한다.
네트워크형 데이터 모델 : 데이터베이스를 그래프(Graph) 구조로 표현(owner-member 관계)하며, CODASYL DBTG 모델이라고도 한다.
41. 관계형 데이터베이스 모델
관계형 데이터베이스 모델의 개요
관계형 데이터베이스 모델의 정의

관계형 데이터베이스를 구성하는 개체나 관계를 릴레이션(Relation)으로 표현한다.
관계형 데이터베이스 모델 구조

속성(Attribute)	학번	이름	학과	학년
튜플(Tuple)	2024010	A	심리학과	1
튜플	2025015	B	전자공학과	2
튜플	2026016	C	컴퓨터공학과	3

튜플(Tuple)
테이블의 행(Row)에 해당하며 파일 구조의 레코드(Record)와 같은 의미이다.
카디널리티(Cardinality) : 튜플의 수(기수)
한 릴레이션의 튜플들의 값은 모두 상이하며, 튜플 간 순서가 없다.
속성(Attribue)
테이블의 열(Column)에 해당하며 파일 구조의 항목(Item), 필드(Field)와 같은 의미이다.
차수(Degress) : 속성의 수
한 릴레이션의 속성은 원자값이며, 속성 간 순서가 없다.
도메인(Domain) : 하나의 속성이 가질 수 있는 원자값들의 집합이다.
릴레이션의 특징

튜플의 유일성 : 모든 튜플은 서로 다른 값을 갖는다.
튜플의 무순서성 : 하나의 릴레이션에서 튜플의 순서는 없다.
속성의 원자성 : 속성은 원자값을 갖는다.
속성의 무순서성 : 각 속성은 릴레이션 내에서 유일한 이름을 가지며, 속성의 순서는 큰 의미가 없다.
키(Key)의 종류와 무결성
키의 분류

학번	주민번호	이름	나이	<-학생---수강->	학번	과목
1	123123-1	A	10	---	1	운영체제
2	123123-1	B	20	---	2	소프트웨어공학
3	123123-1	C	30	---	3	C 언어

슈퍼키(Super Key)

두 개 이상의 속성으로 구성된 키 또는 혼합키를 의미한다.
모든 튜플에 대해 유일성은 만족하지만, 최소성은 만족하지 않는다.
외래키(Foreign Key)

다른 테이블의 기본키로 사용되는 속성이다.
<수강> 테이블에서 <학생> 테이블을 참조할 때 <학생> 테이블의 학번은 참조키, <수강> 테이블의 학번이 외래 키가 된다.
무결성(Integrity)

릴레이션 무결성 규정(Relation Integrity Rules)은 릴레이션을 조작하는 과정에서의 의미전 관계 (Semantic Relationship)를 명세한 것으로 정의 대항으로 도메인, 키, 종속성 등이 있다.
개체 무결성 : 기본키의 값은 Null 값이나 중복 값을 가질 수 없다는 제약조건이다.
참조 무결성 : 릴레이션 R1 에 속성 조합인 외래키를 변경하려면 이를 참조하고 있는 릴레이션 R2 의 기본키도 변경해야 한다. 이때 참조할 수 없는 외래키 값을 가질 수 없다는 제약조건이다.
도메인 무결성 : 릴레이션 중 하나의 속성은 반드시 원자 값이어야 한다는 것을 보장하는 제약조건이다.
42. 데이터베이스 설계와 구조화
데이터베이스 설계 단계
요구조건 분석

데이터베이스 사용자로부터 요구조건 수집과 요구조건 명세서를 작성한다.
개념적 설계

목표 DBMS 에 독립적인 개념 스키마를 설계한다.
개념 스키마 모델링(ERD)과 트랜잭션 모델링을 병행 수행한다.
논리적 설계

목표 DBMS 에 종속적인 논리적 스키마를 설계한다.
스키마의 평가 및 정제를 한다.
논리적 데이터 모델로 변환 및 트랜잭션 인터페이스를 설계한다.
물리적 설계

목표 DBMS 에 종속적인 물리적 구조를 설계한다.
저장 레코드 양식 설계와 레코드 집중의 분석/설계, 엑세스 경로 인덱싱, 클러스터링, 해싱 등의 설계가 포함된다.
접근 경로 설계 및 트랜잭션 세부 설계를 한다.
데이터베이스 구현

목표 DBMS 의 DDL 로 스키마를 작성한다.

데이터베이스에 등록 후 트랜잭션을 작성한다.
데이터베이스 정규화
정규화(Normalization)의 개념

함수적 종속성 등의 종속성 이론을 이용하여 잘못 설계된 관계형 스키마를 더 작은 속성의 세트로 쪼개어 바람직한 스키마로 만들어 가는 과정이다.
좋은 데이터베이스 스키마를 생성하고 불필요한 데이터의 중복을 방지하여 정보 검색을 용이하게 할 수 있도록 허용한다.
정규화의 목적

데이터 구조의 안정성 최대화
중복 데이터의 최소화
수정 및 삭제 시 이상 현상 최소화
테이블 불일치 위험 간소화
이상 현상(Anomaly)

릴레이션 조작 시 데이터들이 불필요하게 중복되어 예기치 않게 발생하는 곤란한 현상을 의미한다.
종류 : 삽입 이상, 삭제 이상, 갱신 이상
수강 테이블

학번	과목코드	성적	학년
100	C413	A	4
200	C123	B	1
300	C312	B	3
400	C312	C	2
500	C324	A	2
600	E412	C	2

단, 학번, 과목코드가 하나로 묶여 기본키가 되는 혼합 속성이다.

삽입 이상(Insertion Anomaly)

데이터를 삽입할 때 불필요한 데이터가 함께 삽입되는 현상이다.
<수강> 릴레이션에 학번이 600 이고, 학년이 2 인 학생 값을 새롭게 삽입하려 할 때, 이 학생이 어떤 과목을 등록해서 과목번호를 확보하지 않는 한 이 삽입은 성공할 수 없다.(개체무결성 위반).
삭제 이상(Deletion Anomaly)

릴레이션의 한 튜플을 삭제함으로써 연쇄 삭제로 인해 정보의 손실을 발생시키는 현상이다.

<수강> 릴레이션에서 학번이 200 인 학생이 과목 'C123'의 등록을 취소한다고 할 때, 자연히 학번이 200 인 튜플에서 과목번호 C123 을 삭제해야 하는데 과목번호는 기본키에 포함되어 있기 때문에 과목번호만 삭제하지 못한다.
갱신 이상(Update Anomaly)

튜플 중에서 일부 속성을 갱신함으로써 정보의 모순성이 발생하는 현상이다.
<수강> 릴레이션에 학번이 400 인 학생의 학년을 2 에서 3 으로 변경시키려 할 때, 이 변경을 위해서는 이 릴레이션에 학번 400 이 나타나 있는 튜플 3 개 모두에 대해 학년의 값을 갱신시켜야 한다. 그렇게 하지 않고 일부 튜플만 변경시키게 되면 학번 400 인 학생의 학년이 2 와 3, 즉 두 가지 값을 갖게 되어 일관성이 없게 된다.
함수적 종속
함수적 종속

개체 내에 존재하는 속성 간의 관계를 종속적인 관계로 정리하는 방법이다.
데이터 속성들의 의미와 속성 간의 상호 관계로부터 도출되는 제약조건이다.
기준값을 결정자(Determinant)라 하고 종속되는 값을 종속자(Dependent)라고 한다.
속성 Y 는 속성 X 에 함수적 종속이라 하고 표현은 X->Y 로 표현한다. 이때 X 를 결정자, Y 를 종속자라고 부른다.
완전 함수적 종속

복합 속성 X 에 대하여 X->Y 가 성립할 때이다.
이행 함수적 종속

속성 X, Y, Z 가 주어졌을 때 X->Y, Y->Z 하면 X->Z 가 성립된다는 것이다.
43. 정규화
정규화 과정
비정규 릴레이션 => 1NF(도메인이 원자값) => 2NF(부분적 함수 종속 제거) => 3NF(이행적 함수 종속 제거) => BCNF(결정자이면서 후보키가 아닌 함수 종속 제거) => 4NF(다치 종속 제거) => 5NF(조인 종속성 제거)

1 정규형

어떤 릴레이션에 속한 모든 도메인이 원자값(Atomic Value)만으로 되어 있는 릴레이션이다.
하나의 속성만 있어야 하고 반복되는 속성은 별도 테이블로 분리한다.

국가	도시	1 정규형->	국가	도시
대한민국	서울,부산		대한민국	서울

미국 워싱턴 뉴욕 대한민국 부산
중국 베이징 미국 워싱턴
미국 뉴욕
중국 베이징
2 정규형

1 정규형을 만족하고, 내재된 부분 함수적 종속을 제거한다.
기본키가 아닌 애트리뷰트 모두가 기본키에 완전 함수 종속이 되도록 부분 함수적 종속에 해당하는 속성을 별도의 테이블로 분리한다.
3 정규형

1, 2 정규형을 만족하고, 이행적 함수 종속(A->B,B->A->C)을 제거한다.
BCNF(Boyce-Code Normal Form - 보이스/코드)정규형

1, 2, 3 정규형을 만족하고, 결정자가 후보키가 아닌 함수 종속이 제거되면 보이스/코드 정규형에 속한다.
후보키를 여러 개 가지고 있는 릴레이션에서 발생할 수 있는 이상 현상을 해결하기 위해 3 정규형보다 좀 더 강력한 제약조건을 적용한다.
보이스/코드 정규형에 속하는 모든 릴레이션은 3 정규형에 속하지만, 3 정규형에 속하는 모든 릴레이션이 보이스/코드 정규형에 속하지는 않는다.
4 정규형

1, 2, 3, BCNF 정규형을 만족하고, 다가(다치) 종속을 제거한다.
5 정규형

1, 2, 3, BCNF, 4 정규형을 만족하고, 후보키를 통하지 않은 조인 종속을 제거한다.
반정규화
정규화를 통하여 정합성과 데이터 무결성이 보장되지만, 테이블의 개수가 증가함에 따라 테이블 간의 조인이 증가하여 조회 성능이 떨어질 수 있는데, 이렇게 정규화된 엔티티, 속성, 관계에 대해 시스템의 성능 향상과 개발(Development) 및 운영(Maintenance)의 단순화를 위해 중복, 통합, 분리 등을 수행하는 데이터 모델링의 기법을 의미한다.
반정규화 기법 : 테이블 반정규화, 컬럼 반정규화, 관계 반정규화
테이블 반정규화 기법 : 테이블 병합, 테이블 분할, 테이블 추가
테이블 추가 반정규화 유형 : 중복 테이블 추가, 집계 테이블 추가, 진행 테이블 추가, 부분 테이블 추가
테이블 병합

기법 설명
1:1 관계 테이블 병합 1:1 관계를 통합하여 성능을 향상시킨다.
1:M 관계 테이블 병합 1:M 관계를 통합하여 성능을 향상시킨다.
슈퍼/서브 타입 테이블 병합 슈퍼/서브를 통합하여 성능을 향상시킨다.
테이블 분할(파티셔닝)

테이블을 여러 부분으로 분할하는 것을 의미한다.
대표적으로 분산 데이터베이스 분할로 각 파티션은 여러 노드로 분산 배치되어 사용자가 각 노드에서 로컬 트랜잭션을 수행할 수 있다.
파티션 각각이 작은 데이터베이스가 되도록 분할하는 방법과 하나의 테이블만 같이 선택된 요소만 분리하는 방법이 있다.
행/열 분할 기법

수직 분할
트랜잭션의 처리 유형을 파악하고 컬럼(열) 단위의 테이블을 저장 장치의 I/O 분산 처리를 위하여 테이블을 1:1 로 분리하여 성능을 향상시킨다.
수평 분할
row(행) 단위로 집중 발생되는 트랜잭션을 분석하여 저장 장치의 I/O 및 데이터 접근의 효율성과 성능 향상을 위해 row 단위로 테이블을 분할한다.
분할 키 기준 분할 기법

분할 범위
분할 키 값이 범위 내에 있는지 여부로 구분한다. 예를 들어, 우편번호를 분할 키로 수평 분할하는 경우이다.(일, 월, 분기 등 순차 데이터를 관리하는 테이블에 많이 사용한다.)
목록 분할
값 목록에 파티션의 할당 분할 키 값을 그 목록에 비추어 파티션을 선택한다. 예를 들어, Country 라는 컬럼의 값이 Iceland, Norway, Sweden, Finland, Denmark 중 하나에 있는 행을 빼낼 때 북유럽 국가 파티션을 구축할 수 있다.
해시 분할
해시 함수의 값에 따라 파티션에 포함할지를 결정한다. 예를 들어, 4 개의 파티션으로 분할하는 경우 해시 함수는 0~3 의 정수를 돌려준다.
합성 분할
범위, 목록, 해시 분할을 결합하여 사용한다. 예를 들면 먼저 범위 분할하고, 다음에 해시 분할 같은 것을 생각할 수 있다. 컨시스턴트 해시법은 해시 분할 및 목록 분할의 합성으로 간주될 수 있고 키 공간을 해시 축소함으로써 일람할 수 있게 한다.
라운드로빈 분할

라운드로빈 분할로 회전하면서 새로운 행이 파티션에 해당된다.
파티션에 행의 고른 분포를 원할 때 사용한다.
기본키가 필수가 아니며, 해시 분할과 다르게 분할 컬럼을 명시하지 않아도 된다.
데이터베이스 클러스터링

두 대 이상의 서버를 하나의 서버처럼 운영하는 기술로, 서버 이중화 및 공유 스토리지를 사용하여 서버의 가용성을 높이는 기술이다.
병렬 처리 클러스터링 : 처리율을 높이기 위한 목적으로 단위 작업을 여러 서버에서 분산 처리한다.
고가용성 클러스터링 : 하나의 서버에 장애가 발생하면 다른 서버가 작업을 이어받아 처리하여 서비스 중단을 방지한다.
테이블 추가

중복 테이블 추가
업무가 다르거나 서버가 분리된 경우 같은 테이블을 중복으로 추가하여 원격조인을 제거하는 방법을 통하여 성능을 향상시킨다.
집계 테이블 추가
합계, 평균 등 통계 계산을 미리 수행하여 계산해두어 조회 시 성능을 향상한다.
이력 테이블 추가
이력 테이블에 레코드를 중복 저장하여 성능을 향상시킨다.
부분 테이블 추가
하나의 테이블을 전체 컬럼 중 자주 이용하는 집중화된 컬럼이 있을 경우, 디스크 I/O 를 줄이기 위해 해당 컬럼들을 모아놓은 별도의 반정규화된 테이블을 생성한다.
진행 테이블
검색 조건이 여러 테이블에 걸쳐 다양하게 사용되어 복잡하고 처리량이 많은 경우 사용한다.
컬럼 반정규화

중복 컬럼 추가
조인 시 성능 저하를 예방하기 위해, 중복된 컬럼을 추가하여 조인 횟수를 감소시킨다.
파생 컬럼 추가
트랜잭션이 처리되는 시점에 계산 때문에 발생하는 성능 저하를 예방하기 위해 미리 계산된 값을 저장하는 파생 컬럼을 추가한다.
이력 테이블 컬럼 추가
대량의 이력 데이터를 처리할 때 임의의 날짜 조회나 최근 값을 조회할 때 발생하는 성능 저하를 예방하기 위해 최근값 여부, 시작일, 종료일 등의 기능성 컬럼을 추가한다.

PK(Primary Key)에 의한 컬럼 추가
복합 의미가 있는 PK 를 단일 속성으로 구성했을 때 발생하며 PK 안에 데이터가 존재하지만, 성능 향상을 위해 일반 컬럼으로 추가한다.
응용 시스템 오작동을 위한 컬럼 추가
업무적으로는 의미가 없으나, 데이터 처리 시 오류로 인해 원래 값으로 복구하길 원하는 경우 이전 데이터를 임시로 중복 보관하는 컬럼을 추가한다.
관계 반정규화

중복 관계 추가
데이터 처리 시 여러 경로를 거쳐 조인할 수 있지만, 이때 발생할 수 있는 성능 저하를 방지하기 위해 추가적인 관계 설정을 통하여 성능을 향상할 수 있다.
44. 관계 대수와 연산자
관계 대수와 관계 해석
관계 대수(Relational Algebra)

원하는 정보와 그 정보를 어떻게 유도하는가를 기술하는 절차적인 방법이다.
주어진 릴레이션 조작을 위한 연산의 집합이다.
일반 집합 연산과 순수 관계 연산으로 구분된다.
질의에 대한 해를 구하기 위해 수행해야 할 연산의 순서를 명시한다.
관계 해석(Relational Calculus)

원하는 정보가 무엇이라는 것만 정의하는 비절차적인 방법이다.
순수 관계 연산자
순수 관계 연산자의 종류

연산자 속성
Select(∑) 튜플 집합을 검색한다.
Project(∏) 속성 집합을 검색한다.
Join(▷◁) 두 릴레이션의 공통 속성을 연결한다.
Division(÷) 두 릴레이션에서 특정 속성을 제외한 속성만 검색한다.
Select(선택)

릴레이션의 행에 해당하는 튜플을 선택하는 것이므로 수평적 연산이라고도 한다.
연산자의 기호는 시그마(∑)를 사용한다.
Project(추출)

Project(추출)은 릴레이션의 열에 해당하는 속성을 추출하는 것이므로 수직적 연산이라고도 한다.
연산자의 기호는 파이(∏)를 사용한다.
Join(연결)

공통 속성을 기준으로 두 릴레이션을 합하여 새로운 릴레이션을 만드는 연산이다.
연산자의 기호는 ▷◁를 사용한다.
Division(나누기)

Division 에서 나누어지는 릴레이션(학생 릴레이션)은 나누는 릴레이션(학과 릴레이션)의 모든 속성을 전부 포함하고 있다.
연산자의 기호는 ÷를 사용한다.
집합 연산자
일반 집합 연산자의 종류

집합	연산자	속성
합집합	∪	두 릴레이션의 튜플의 합집합을 구하는 연산이다.
교집합	∩	두 릴레이션의 튜플의 교집합을 구하는 연산이다.
차집합	—	두 릴레이션의 튜플의 차집합을 구하는 연산이다.
교차곱	X	두 릴레이션의 튜플들의 교차곱(순서쌍)을 구하는 연산이다.

Union(합집합)

Union(합집합)은 두 개의 릴레이션을 합쳐 하나의 릴레이션을 생성한다.
Intersection(교집합)

Intersection(교집합)은 연관성이 있는 두 개의 릴레이션에서 중복되는 레코드를 선택하여 릴레이션을 생성한다.
Difference(차집합)

Difference(차집합)은 연관성이 있는 두 개의 릴레이션에서 중복되는 레코드를 제거하여 릴레이션을 생성한다.
Cartesian Product(교차곱)

두 릴레이션의 튜플을 교차 곱하여 생성한다.
45. SQL, DDL, DCL, View
SQL(Structured Query Language)

의미 : 관계형 데이터베이스의 표준 질의어이다.
종류 : DDL, DML, DCL

DDL(Data Definition Language, 데이터 정의어)

데이터베이스의 정의/변경/삭제에 사용되는 언어이다.
논리적 데이터 구조와 물리적 데이터 구조로 정의할 수 있다.
논리적 데이터 구조와 물리적 데이터 구조 간의 사상을 정의한다.
번역한 결과가 데이터 사전에 저장된다.
종류
CREATE : 스키마, 도메인, 테이블, 뷰 정의
ALTER : 테이블 정의 변경(필드 추가, 삭제, 갱신)
DROP : 스키마, 도메인, 테이블, 뷰 삭제
CREATE 문 문법 구조

CREATE TABLE : 테이블을 생성하는 명령문이다.
CREATE TABLE 기본테이블
```
    (
        {열이름 데이터_타입 [NOT NULL] [DEFALUT 값]}
        {[PRIMARY KEY(열이름_리스트)]},
        {[UNIQUE(열이름_리스트,...)]},
        {[FOREIGN KEY(열이름_리스트)]
        REFERENCES 기본테이블[(기본키_열이름)]
        [ON DELETE 옵션]
        [ON UPDATE 옵션]}
        [CHECK(조건식)]
    );
```
{}는 중복 가능한 부분
NOT NULL 은 특정 열에 대해 NULL 값을 허용하지 않을 때 기술
PRIMARY KEY 는 기본키를 구성하는 속성을 지정할 때 사용된다.
FOREIGN KEY 는 외래키로 어떤 릴레이션의 기본키를 참조하는지를 기술한다.
Alter 문 문법 구조

ALTER TABLE : 테이블 구조(필드 추가, 삭제, 변경) 변경문이다.
ALTER TABLE 테이블_이름 ADD 열_이름 데이터_타입 DEFAULT 값;
ALTER TABLE 테이블_이름 ALTER 열_이름 SET DEFAULT 값;
ALTER TABLE 테이블_이름 DROP 열_이름 CASCADE;
ADD : 새로운 열(속성)을 추가할 때 사용한다.
ALTER : 특정 열(속성)의 디폴트 값을 변경할 때 사용한다.
DROP : 특정 열(속성)을 제거할 때 사용한다.

DROP 문 문법 구조

DROP : 테이블 삭제문
DROP SCHEMA 스키마_이름 [CASCADE | RESTRICT];
DROP DOMAIN 도메인_이름 [CASCADE | RESTRICT];
DROP TABLE 테이블_이름 [CASCADE | RESTRICT];
DROP INDEX 인덱스_이름;
CASCADE : 옵션을 사용하면 삭제할 요소가 다른 개체에서 참조 중이라도 삭제가 수행된다.
RESTRICT : 옵션을 사용하면 삭제할 요소가 다른 개체에서 참조 중이라면 삭제가 수행되지 않는다.
DCL(Data Control Language)
DCL(데이터 제어어)의 기능

데이터 제어 정의 및 기술에 사용되는 언어이다.
불법적인 사용자로부터 데이터를 보호한다.
무결성을 유지하고 데이터 복구 및 병행 제어를 한다.
종류
COMMIT : 명령어로 수행된 결과를 실제 물리적 디스크로 저장하고, 명령어로 수행을 성공적으로 완료하였음을 선언한다.
ROLLBACK : 명령어로 수행에 실패하였음을 알리고, 수행된 결과를 원상 복귀시킨다.
GRANT : 데이터베이스 사용자에게 사용 권한을 부여한다.
REVOKE : 데이터베이스 사용자로부터 사용 권한을 취소한다.
뷰(View)

사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 기본 테이블에서 유도되는 가상 테이블이다.
뷰(View) 특징

뷰의 생성 시 CREATE 문, 검색 시 SELECT 문을 사용한다.
뷰의 정의 변경 시 ALTER 문을 사용할 수 없고 DROP 문을 이용한다.
뷰를 이용한 또 다른 뷰의 생성이 가능하다.
하나의 뷰 제거 시 그 뷰를 기초로 정의된 다른 뷰도 함께 삭제된다.
뷰에 대한 조작에서 삽입, 갱신, 삭제 연산은 제약이 따른다.
뷰가 정의된 기본 테이블이 제거되면 뷰도 자동적으로 제거된다.
뷰(View) 장단점

장점

논리적 데이터 독립성 제공, 사용자 데이터 관리 편의성을 제공한다.

접근 제어를 통한 보안을 제공한다.

단점

ALTER VIEW 문으로 뷰의 정의 변경이 불가능하다.

삽입, 갱신, 삭제 연산에 제약이 따른다.

시스템 카탈로그(System Catalog)

시스템 자신이 필요로 하는 여러 가지 객체(기본 테이블, 뷰, 인덱스, 데이터베이스, 패키지, 접근 권한 등)에 관한 정보를 포함하고 있는 시스템 데이터베이스이다.

데이터 사전(Data Dictionary), 메타 데이터(Meta Data)라고도 한다.

시스템 카탈로그 자체도 시스템 테이블로 구성되어 있어 SQL 문을 이용하여 내용 검색이 가능하다.

사용자가 시스템 가탈로그를 직접 갱신할 수는 없으나 SQL 문으로 여러 가지 객체에 변화를 주면 시스템이 자동으로 갱신된다.

46. 데이터베이스 조작어(DML)

DML

DML(Data Maniqulation Language, 데이터 조작어)의 개념

데이터의 검색/삽입/삭제/변경에 사용되는 언어이다.

사용자와 DBMS 간의 인터페이스를 제공한다.

종류

SELECT

튜플 검색 명령어이다.

기본 구조

```
SELECT 속성명[ALL | DISTINCT]
FROM 릴레이션명
WHERE 조건
[GROUP BY 속성명1, 속성명2, ... ]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]]
```

ALL : 모든 튜플을 검색(생략 가능)

DISTINCT : 중복된 튜플 생략

INSERT

튜플 삽입 명령어

기본 구조

```
INSERT INTO 테이블명(속성명1, 속성명2,...)
```

VALUES(데이터 1, 데이터 2 ...);

DELTETE

튜플 삭제 명령어이다.

기본 구조

```
DELETE
FROM 테이블명
WHERE 조건;
```

UPDATE

튜플의 내용 변경 명령어이다.

기본 구조

```
UPDATE 테이블명
SET 속성명 = 데이터
WHERE 조건;
```

NoSQL

NoSQL

"Not only SQL"로, SQL 만을 사용하지 않는 데이터베이스 관리 시스템(DBMS)을 지칭하며, 다양한 유형의 데이터베이스를 사용하는 것을 의미한다.

데이터를 저장하는데 SQL 외에도 다른 방법도 있다는 개념하에 비정형 데이터의 저장을 위해 유연한 데이터 모델을 지원한다.

전통적인 관계형 데이터베이스 관리 시스템과는 다른 비관계형(Non-Relational) DBMS 이다.

그룹 함수, 하위 질의

그룹 함수의 종류(집계 함수)

종류 설명

COUNT - 테이블의 행 수를 계산할 때

- 표현식: COUNT(*)

SUM - 하나 또는 여러 개의 열 합계를 구할 때

- 표현식 : SUM(열 이름)

AVG - 하나 또는 여러 개의 열 평균을 구할 때

- 표현식 : AVG(열 이름)

MAX - 해당 열의 최댓값을 구할 때

- 표현식 : MAX(열 이름)

HAVING 절을 사용한 조회 검색

GROUP BY 절에 의해 선택된 그룹의 탐색 조건을 지정할 수 있으며 SUM, AVG, COUNT, MAX, MIN 등의 그룹 함수와 함께 사용할 수 있다.

ORDER BY 절을 이용한 정렬 검색

특정 항목을 기준으로 검색 테이블의 행들을 오름차순(ASC) 또는 내림차순(DESC)으로 정렬할 때 사용한다. 생략하면 ASC 가 디폴트 값이 되어 오름차순으로 정렬된다.
하위 질의(Sub Query)

질의를 1 차 수행한 다음, 반환값을 다른 릴레이션의 WHERE 절에 포함시켜 사용하는 것이다.
47. 트랜잭션, 병행 제어
트랜잭션
트랜잭션의 정의(Transaction)

하나의 논리적 기능을 수행하기 위한 작업 단위이다.
데이터베이스에서 일어나는 연산의 집합이다.
트랜잭션의 특성

원자성(Atomicity)
완전하게 수행이 완료되지 않으면 전혀 수행되지 않아야 한다.
연산은 Commit, Rollback 을 이용하여 적용 또는 취소로 한꺼번에 완료되어야 한다.
중간에 하나의 오류가 발생되더라도 취소가 되어야 한다.
일관성(Consistency)
시스템의 고정 요소는 트랜잭션 수행 전후가 같아야 한다.
트랜잭션 결과는 일관성을 유지해야 한다.
격리성(Isolation, 고립성)
트랜잭션 실행 시 다른 트랜잭션의 간섭을 받지 않아야 한다.
영속성(Durability, 지속성)
트랜잭션의 완료 결과가 데이터베이스에 영구히 기억된다.
은행계좌에서 100 원 중 10 원을 인출했을 때 계좌에는 90 원이 남아 있어야 한다.
CRUD Matrix

데이터베이스에 영향을 주는 생성, 읽기, 갱신, 삭제 연산으로 프로세스와 테이블 간에 매트릭스를 만들어서 트랜잭션을 분석하는 도구이다.
업무 프로세스와 데이터 간의 상관관계 분석을 위한 것으로 업무 프로세스와 엔티티 타입을 행과 열로 구분하여 행과 열이 만나는 교차점 이용에 대한 상태를 표시한다.
즉각 갱신법

데이터를 갱신하면 트랜잭션이 완료되기 전에 실제 데이터베이스에 반영하는 방법이다.
회복 작업을 위해서 갱신 내용을 별도 Log 로 기록해야 한다.

Redo, Undo 모두 사용 가능하다.
트랜잭션의 연산

Commit 연산 : 트랜잭션 실행이 성공적으로 종료되었음을 선언한다.
Rollback 연산 : 트랜잭션 실행히 실패하였음을 선언한다.
Recovery 연산 : 트랜잭션을 수행하는 도중 장애로 인해 손상된 데이터베이스를 손상되기 이전의 정상적인 상태로 복구시키는 작업이다.
트랜잭션의 상태

실행 시작

활동
부분 완료 -> 완료
실패 -> 철회
활동 : 트랜잭션이 현재 실행 중인 상태를 말한다.

부분 완료 : 트랜잭션이 마지막 처리를 실행한 뒤 데이터베이스에 그 처리 내용을 적용하기 직전의 상태이다.

완료 : 부분 완료 상태에서 정상적인 트랜잭션 터리가 이루어져 데이터베이스에 트랜잭션 처리를 적용 완료한 상태이다.

실패 : 트랜잭션 실행 중 오류로 인해 정상적인 처리가 되지 않아 원자성과 일관성에 문제가 발생하여 더 이상 처리가 불가능한 상태이다.

철회 : 트랜잭션 처리 실패를 확인하고 처음 상태로 돌아가는 상태이다.

병행 제어
병행 제어(Concurrency Control)

동시에 수행되는 트랜잭션들을 일관성 있게 처리하기 위해 제어하는 것이다.
목적
데이터베이스의 공유를 최대화한다.
데이터베이스의 일관성을 최대화한다.
시스템 활용도를 최대화한다.
사용자에 대한 응답 시간을 최소화한다.
병행 수행의 문제점 : 갱신 분실, 비완료 의존성, 모순성, 인쇄 복귀가 있다.
종류 : 로킹, 최적 병행 수행, 타임스탬프, 다중 버전 기법

타임스탬프

트랜잭션이 DBMS 로부터 유일한 타임스탬프(시간 허가 인증 도장)를 부여받는다.
동시성 제어를 위한 직렬화 기법으로 트랜잭션 간의 순서를 미리 정하는 방법이다.
병행 제어 기법

로킹(Locking)의 의미 : 하나의 트랜잭션이 데이터를 액세스하는 동안 다른 트랜잭션이 그 데이터 항목을 액세스할 수 없도록 하는 병행 제어 기법이다.
로킹(Locking) 특징

로킹 단위가 커지면 로크의 수가 적어 관리가 쉬워지지만 병행성 수준은 낮아진다.
로킹 단위가 작으면 로크의 수가 많아 관리가 어려워지지만 병행성 수준은 높아진다.
로킹의 대상이 되는 객체(파일, 테이블, 필드, 레코드)의 크기를 로킹 단위라고 한다.
2 단계 로킹(2-Phase Locking)
직렬성(트랜잭션을 순서대로 처리하는 것)은 보장하지만 교착상태 예방은 불가능하다.
확장 단계와 축소 단계의 두 단계(Phase)가 있다.
각 트랜잭션의 로크 요청과 해제 요청을 2 단계로 실시한다.
48. 보안, 분산 데이터베이스
보안
보안(Security)의 개념

권한이 없는 사용자로부터 데이터베이스를 보호하는 것이다.
암호화(Encryption)

네트워크를 통하거나 컴퓨터 내부에 자료를 저장할때 권한을 가진 사람 외에는 데이터를 보지 못하도록 하는 것
일반 평문을 다양한 방식의 암호화 기법으로 가공하여 저장하거나 전송 권한이 있는 사용자에 의해 복호화되어 사용한다.
암호화 과정 : 평문 => 암호화(암호키) => 암호문 => 복호화(복호키) => 평문

암호화 기법
비밀키(Private Key, 대칭키) 암호화 기법

비밀키 암호화 기법을 동일한 키로 데이터를 암호화하고 복호화한다.
암호화, 복호화 키가 같아서 키를 공개하면 타인이 알게 된다.
암호화와 복호화 속도가 빠르다.
공개키(Public Key, 비대칭키) 암호화 기법

공개키 암호화 기법은 각기 다른 키로 데이터를 암호화하고 복호화한다.
암호화, 복호화 키가 다르므로 키는 공개되어도 된다.
암호화 및 복호화 속도가 느리다.
권한 부여 기법
GRANT

데이터베이스 사용자에게 사용 권한을 부여한다.
기본 구조

```
GRANT 권한 ON 데이터 객체 TO 사용자 [WITH GRANT OPTION];
```

WITH GRANT OPTION : 사용자가 부여받은 권한을 다른 사용자에게 다시 부여할 수 있는 권한을 부여한다.
부여 가능한 권한 : Update, Delete, Insert, Select
REVOKE

데이터베이스 사용자로부터 사용 권한을 취소한다.
기본 구조

```
REVOKE [GRANT OPTION FOR] 권한 ON 데이터 객체 FROM 사용자 [CASCADE];
```

GRANT OPTION FOR : 다른 사용자에게 권한을 부여할 수 있는 권한을 취소한다.
CASCADE : 권한을 부여받았던 사용자가 다른 사용자에게 부여한 권한도 연쇄 취소한다.
부여 가능한 권한 : Update, Delete, Insert, Select
트리거(Trigger)

연쇄 반응을 의미한다. 즉 일정 작업을 수행할 때 이에 부수적으로 자동 처리되도록 하는 것을 말한다.
분산 데이터베이스
분산 데이터베이스의 개념

네트워크를 통하여 연결된 여러 개의 컴퓨터에 데이터가 분산된 데이터베이스이다.
데이터 처리와 비용이 큰 곳에 별도의 데이터베이스 서버를 확충하는 것을 의미한다.
분산 데이터베이스의 목표

위치 투명성(Location Transparency) : 하드웨어와 소프트웨워의 물리적 위치를 사용자가 알 필요가 없다.
중복(복제) 투명성(Replication Transparency) : 사용자에게 통지할 필요 없이 시스템 안에 파일들과 자원들의 부가적인 복사를 자유롭게 할 수 있다.
병행 투명성(Concurrency Transparency) : 다중 사용자들이 자원들을 자동으로 공유할 수 있다.

장애 투명성(Failure Transparency) : 사용자들은 어느 위치의 시스템에 장애가 발생했는지 알 필요가 없다.
분산 데이터베이스 시스템의 구성 요소

분산 처리기 : 지리적으로 분산된 시스템을 통합하여 각각의 트랜잭션을 처리한다.
분산 데이터베이스 : 각 지역에 설치되는 데이터베이스 시스템이다.
통신 네트워크 : 지역적으로 분산된 데이터베이스 시스템을 통신 회선으로 연결한다.
분산 데이터베이스의 장/단점

장점
질의 처리 시간의 단축
데이터 공유성, 신뢰성, 가용성 향상
점진적 시스템 용량 확장이 용이
지역 자치성 향상으로 지역 상황에 맞는 시스템 구축 용이
단점
소프트웨어 개발 비용 증가
오류 발생 가능성 증가
통신망 성능에 따라 전체적인 시스템 성능 저하
하드웨어 구매 비용 증가
지역 자치성과 병렬 처리

지역 자치성 : 데이터베이스 중앙 관리자 외에 각 지역에 담당 관리자를 두는 것을 말한다.
병렬 처리 : 하나 이상의 처리를 분산되어 있는 데이터베이스에 분산 처리하여 처리 속도를 높이는 것을 말한다.