|  |  |
| --- | --- |
| 1-1 | 다 |
| 1-2 | 나 |
| 1-3 | 가 |
| 1-4 | 라 |
| 1-5 | 라 |
| 1-6 | 다 |
| 1-7 | 다 |
| 1-8 | 라 |
| 1-9 | 다 |
| 1-10 | 다 |

2.

가) virtual vs transparency

ANSWER

Virtual: 실제로 존재하지는 않지만 가상의 논리적 공간을 구현함으로써 사용자가 실제 존재하는 것처럼 사용할 수 있도록 하는 것을 virtual이라고 한다. 예를 들어, virtual memory를 들 수 있다. 이는 주기억장치의 프로그램 양이 많아질 때, 보조기억장치 내에 특별한 공간을 만들어서 마치 주기억장치와 같이 사용하는 것이다. 주기억장치의 물리적 공간의 제한으로 프로그램을 수용하기 제한되는 경우에 이러한 방식을 사용하기도 한다.

Trasparency: 분산시스템에서 사용자는 데이터가 물리적으로 어디에 배치되어 있는지 또는 특정 지역사이트에서 데이터를 access할 때 어떻게 access 될 수 있는지 알 필요가 없다. 사용자는 이러한 시스템을 이용할 때 마치 하나의 system을 사용하는 것처럼 사용하게 되는데 이러한 특성을 transparency라고 한다. 이러한 transparency에는 여러 가지 형태가 존재한다.

‣fragmentation transparency – 사용자가 relation이 어떻게 fragmentation 되었는지 알 필요가 없다

‣replication transparency – 사용자들은 각 데이터 객체들이 논리적으로 유일하다고 생각하는 것이다. 분산시스템은 시스템 성능과 데이터의 가용성을 증대시키기 위한 방안으로 데이터를 중복할 수 있고 사용자들은 어떤 데이터가 중복되었는지, 어디에 사본을 두는지 알 수 없다

‣local transparency – 사용자들이 물리적인 위치를 알 필요가 없다는 것이다. 그리고 분산데이터시스템은 데이터 identifier가 자신의 사용자의 transaction에 의해 제공되는 한 어떤 데이터라고 찾을 수 있어야 한다

MARKING SCHEME

만점: 3점

|  |  |  |
| --- | --- | --- |
| 가점기준 | ① Virtual에 대한 예를 들어 설명한 경우 | 1.5점 |
| ② Transparency에 대한 예를 들어 설명한 경우 | 1.5점 |
| 감점기준 | ① Virtual에 대한 설명이 없거나 틀린 경우 | -0.5점 |
| ② Transparency에 대한 설명이 없거나 틀린 경우 | -0.5점 |
| ③ 답의 내용이 전부 틀리거나 다른 내용을 적을 경우 | 0점 |

나) RESTRICT vs CASCADE (DROP TABLE)

ANSWER

테이블을 삭제하는 SQL문은 DROP이며, 사용법은 다음과 같다.

|  |
| --- |
| DROP TABLE <base table name> <behavior>; |

여기에서 <behavior>는 RESTRICT 혹은 CASCADE가 될 수 있다.

RESTRICT - 현재 그 table에 기초한 뷰 정의 등의 참조가 있으면 DROP이 취소되는 것을 의미한다.

CASCADE - DROP 명령이 항상 성공적으로 수행되고, 해당 table을 사용(참조) 중인 모든 것들도 연쇄적으로 DROP됨을 의미한다.

MARKING SCHEME

만점: 4점

|  |  |  |
| --- | --- | --- |
| 가산기준 | ① 답을 적지 않았거나 전혀 다른 내용 | 0점 |
| ② RESTRICT에 대한 설명만 | 2점 |
| ③ CASCADE대한 설명만 | 2점 |
| ④ RESTRICT + CASCADE에 대한 설명 모두 | 4점 |
| 감산기준 | ① RESTRICT에서 "DROP이 fail"된다는 언급은 있으나, "DROP 대상 테이블이 참조되고 있을 때" 라는 언급이 없으면 | -1점 |
| ② CASCADE에서 "DROP이 성공한다"는 언급은 있으나, "테이블을 사용 중인 모든 것들도 DROP된다"는 언급은 없을 때 | -1점 |

다) manual navigation vs automatic navigation

ANSWER

Relation language는 비절차적 언어라고 할 수 있다. 이 말은 사용자가 “어떻게(How)”가 아닌, “무엇을(What)”을 명시한다는 데에 그 기반을 두고 있다. 다시 말해서, 사용자가 어떠한 데이터를 얻기 위한 모든 절차들을 명시하지 않고, 무엇을 원하는지만 선언하면 된다는 것이다.

Manual navigation: non-relational system에 있어서, navigation process는 프로그래머가 명시한 모든 절차에 의해 이루어진다. 즉, 프로그래머가 직접 “What” 뿐만 아니라 “How”의 측면도 구현해야 한다.

Automatic navigation: relational system에 있어서, navigation process가 사용자의 요청을 만족하기 위해서 저장된 데이터에 대해 자동적으로 수행되는 것. “How”에 대한 측면은 system에서 제공해주지만 “What”에 대한 측면은 사용자가 직접 구현해야 한다.

예시) Relational system에서 사용자는 간단한 SQL문을 사용하여 원하는 데이터를 찾아낼 수 있지만, non-relational system에서 사용자는 복잡한 프로그래밍 등을 통해 원하는 데이터를 찾아야 한다.

MARKING SCHEME

만점: 4점

|  |  |  |
| --- | --- | --- |
| 가산기준 | ① 답을 적지 않았거나 전혀 다른 내용 | 0점 |
| ② manual navigation에 대한 설명만 | 1.5점 |
| ③ automatic navigation에 대한 설명만 | 1.5점 |
| ④ manual navigation에 대한 설명과 예 | 2점 |
| ⑤ automatic navigation에 대한 설명과 예 | 2점 |
| ⑥ manual + automatic에 대한 설명과 예 | 4점 |
| 감산기준 | ① “What"에 대한 언급이 없으면 | -1점 |
| ② "How"에 대한 언급이 없으면 | -1점 |

라) serializable

ANSWER

동시에 수행되는(concurrent execution) 다수의 트랜잭션들은 그 트랜잭션들이 어떤 순서대로 한 번에 하나씩 수행된 결과와 같은 결과를 내도록 보장(Isolation)되어야 한다. 이렇게 직렬적으로 수행되는 결과와 같은 결과(equivalent)를 내는 것이 가능할 때 serializable하다고 한다.

MARKING SCHEME

만점: 4점

|  |  |  |
| --- | --- | --- |
| 감점기준 | ① 직렬(serial) 수행에 대한 언급이 없으면 | -1점 |
| ② concurrent execution에 대한 언급이 없으면 | -1점 |
| ③ transaction에 대한 언급이 없으면 | -1점 |
| 0점기준 | ④ serial과 같은 결과를 내야하는 guarantee에 대한  언급이 없으면 | 0점 |
| ⑤ 답을 적지 않았거나 전혀 다른 내용이 적혀있으면 | 0점 |

3. 관계형 시스템의 최적기(optimizer)는 어떠한 사항을 고려하여 사용자의 요청(request)을 구현할 전략을 작성하는가를 다음 예를 사용하여 설명하라. (10)

RESULT:=(EMP WHERE EMP# = "E4" ) {SALARY};

관계형 시스템은 사용자의 요구를 만족시키기 위해서 저장 데이터베이스를 항해(navigation)하게 된다. 일반적으로 이 작업은 사용자에 의해 수동으로 되는 것이 아니라 시스템에 의해 자동으로 수행되며 최적기는 이를 통해 사용자의 요청을 구현할 전략을 작성한다.

이 쿼리를 분석한 결과는 다음과 같다. (EMP WHERE EMP# = "E4" )는 테이블 EMP에서 EMP#가 E4인 행만으로 제한하는 연산이다. 대괄호 안의 열 이름 "SALARY"는 제한한 연산의 결과에 대해 SALARY열만을 projection하는 연산이다. 그리고 그 결과를 테이블 RESULT에 지정한다. 즉 이 쿼리는 요청이 수행된 후 고용인 E4의 급여를 하나의 열과 하나의 행으로 테이블 RESULT에 구성하는 결과를 발생시킨다.

이에 대해 최적기는 두 가지 전략을 채택한다.

1.테이블 EMP에 대해 요구하는 레코드를 찾을 때까지 물리적인 순차 탐색을 한다.

2.테이블 EMP의 EMP#열에 기초한 인덱스가 있다면 그 인덱스를 이용하여 요구된 데이터로 직접 접근한다.

최적기는 두 전략 중에 적용할 전략을 선택할 것이며 다음과 같은 기본적인 고려사항을 기초하여 요청을 구현할 전략을 선택할 것이다.

∎요청에 참조되는 테이블들이 어느 것인가( 조인같은 연산이 포함되어 있다면 하나 이상의 테이블이 포함된다)

∎테이블들이 얼마나 큰 가

∎어떤 인덱스가 존재하는가

∎인덱스의 선별력(selectivity)이 얼마나 뛰어난가

∎디스크상의 데이터 집중화(clustering)이 뛰어난가

∎어떠한 관계형 연산이 포함되는가

∎etc

채점 기준

10점 : 최적기가 채택할 두 가지 전략을 모두 적고 고려해야 할 기본적인 고려사항을 5가지 이상 적었을 경우

8점 : 최적기가 채택할 두 가지 전략을 모두 적고 고려해야 할 기본적인 고려사항을 3가지 이하로 적었을 경우

6점 : 최적기가 채택할 두 가지 전략을 모두 적고 고려해야 할 기본적인 고려사항을 1가지 적었을 경우

4점 : 최적기가 채택할 두 가지 전략을 모두 적었으나 고려해야 할 기본적인 고려사항을 1가지도 적지 못 햇을 경우

2점 : 최적기가 채택할 두 가지 전략 중 1가지만을 적은 경우

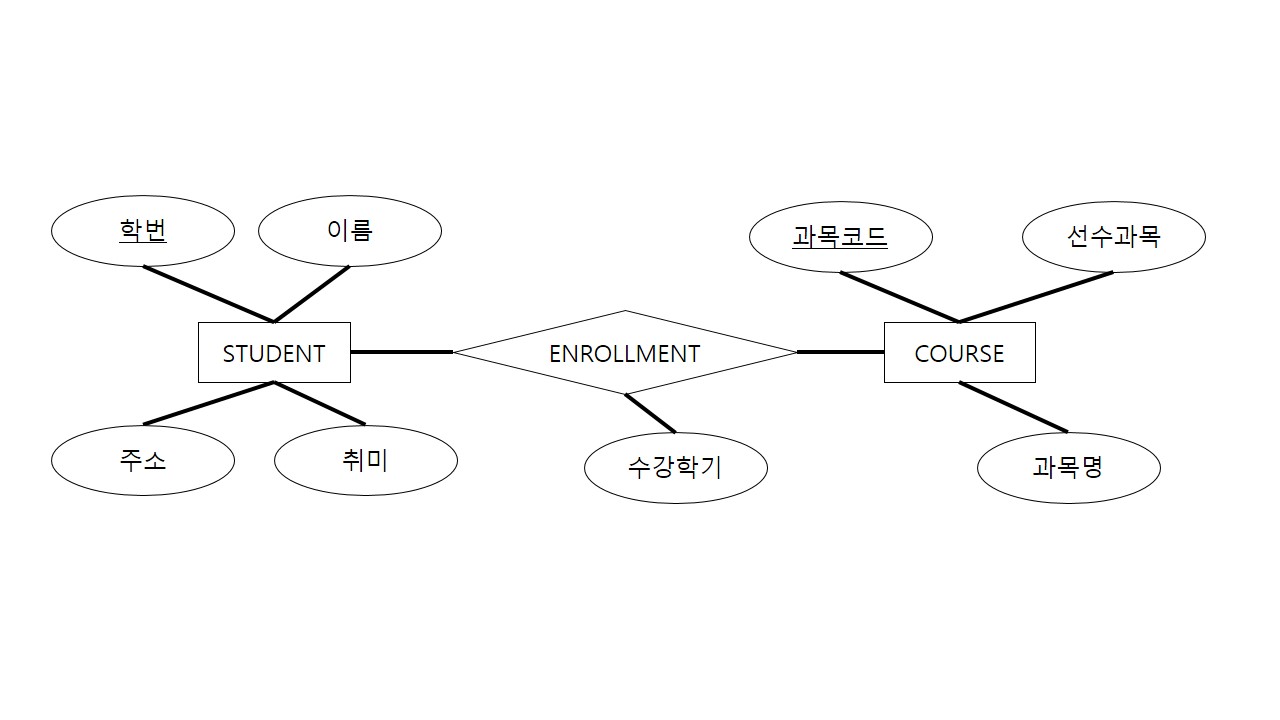
0점 : 답과 관련 없는 내용을 적거나 적지 못 했을 경우

4. 학생(STUDENT)과 과목(COURSE)에 대한 정보를 유지하는 데이터베이스가 있다. 특정한 시점에서 학생과 이들이 수강하는 과목은 다음과 같다. 데이터베이스는 각 학생에 대해서 주소, 취미 등의 상세한 정보를 유지하며, 각각의 수강하는 과목에 대해서도 선수과목, 관계되는 과목코드 등의 여러 가지 정보를 유지한다. 또한 데이터베이스는 각 학생이 (적용되는 경우) 특정 과목을 수강하는 날짜도 기록한다. (15)

|  |  |
| --- | --- |
| STUDENT | COURSE |
| 김갑돌 | PL |
| 홍길동 | OS, PL |
| 허참봉 | DB |
| 신수동 | DB, OS |

가) 이러한 데이타에 대한 개체/관계(Entity/Relationship) 다이어그램을 작성하고, 적절한 관계형 구조(relational structure)를 스케치하라. (4점)

ER 다이어그램



STUDENT

|  |  |  |  |
| --- | --- | --- | --- |
| 학번 | 이름 | 주소 | 취미 |
| 20131234 | 김갑돌 | 서울시 ... | 축구 |
| 20135678 | 홍길동 | 부산시 ... | 농구 |
| 20121234 | 허참봉 | 대전시 ... | 배구 |
| 20125678 | 신수동 | 대구시 ... | 야구 |
| ... | ... | ... | ... |

COURSE

|  |  |  |
| --- | --- | --- |
| 과목코드 | 과목명 | 선수과목 |
| CSE1234 | DB | C프로그래밍 |
| CSE5678 | OS | 자료구조 |
| CSE1357 | PL | 자료구조 |
| ... | ... | ... |

이수과목

|  |  |  |
| --- | --- | --- |
| 학번 | 과목코드 | 수강날짜 |
| 20131234 | CSE1357 | 2013년 봄 |
| 20135678 | CSE5678 | 2013년 봄 |
| 20135678 | CSE1357 | 2013년 봄 |
| 20121234 | CSE1234 | 2013년 봄 |
| 20125678 | CSE1234 | 2013년 봄 |
| 20125678 | CSE5678 | 2013년 봄 |
| ... | ... | ... |

ER 다이어그램: 2점

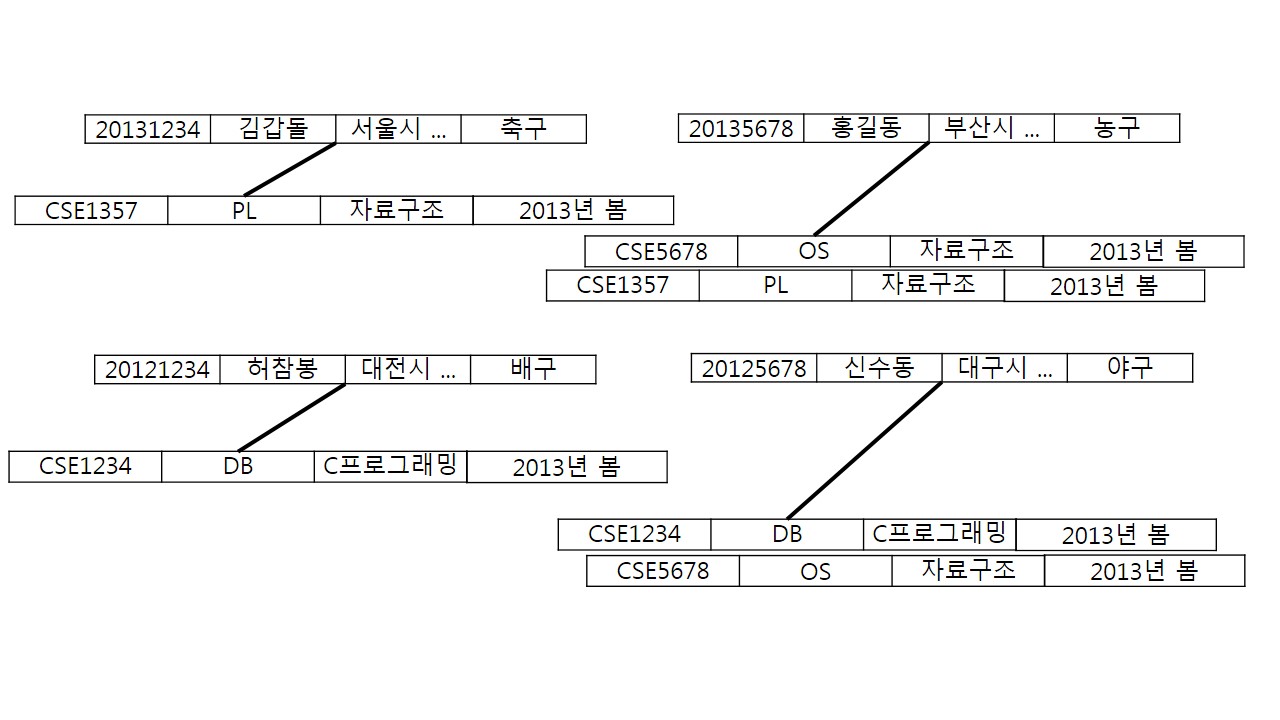
관계형 구조 스케치: 2점

각각 설명이 부족하거나 부정확한 경우 1점씩 감점

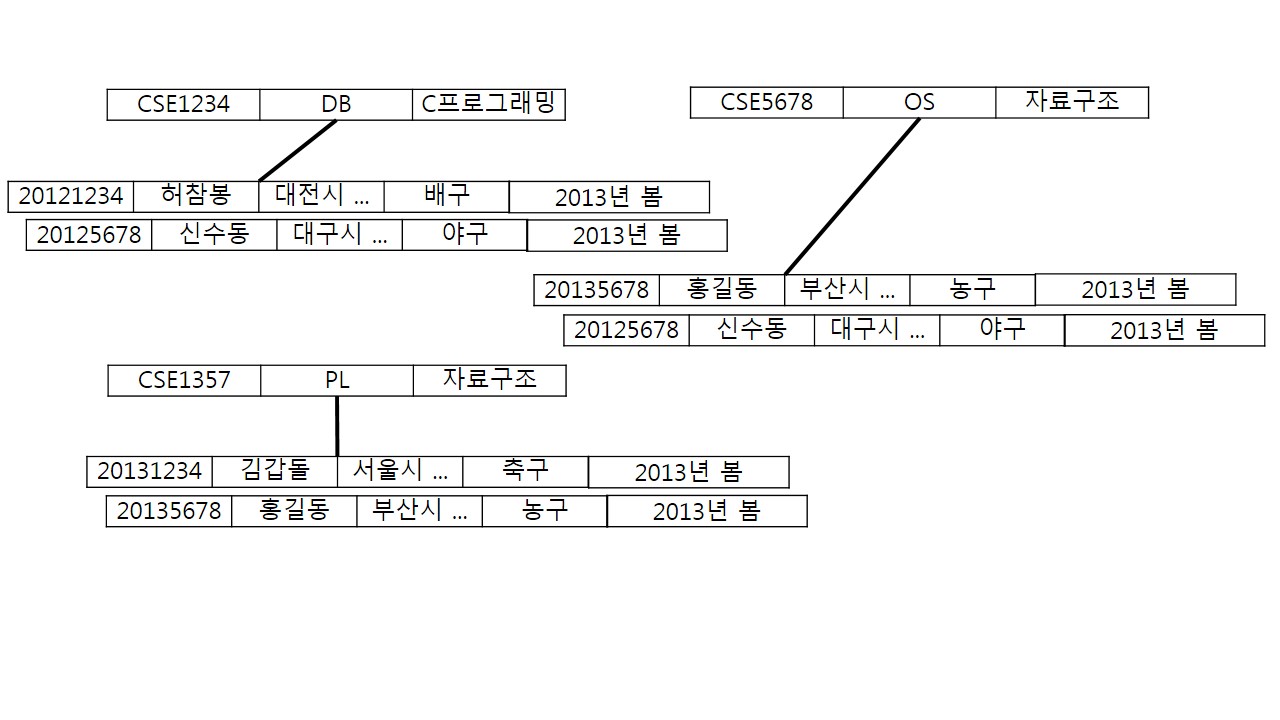
(특정 시점(학기)에 대한 고려가 되지 않았을 경우 등)

나) 이러한 데이터의 두 가지 계층적 구조(hierarchical structure)를 스케치하고, 이 때 발생되는 이상(anomaly)를 설명하라. (4점)

STUDENT가 COURSE의 부모



COURSE가 STUDENT의 부모



Anomaly

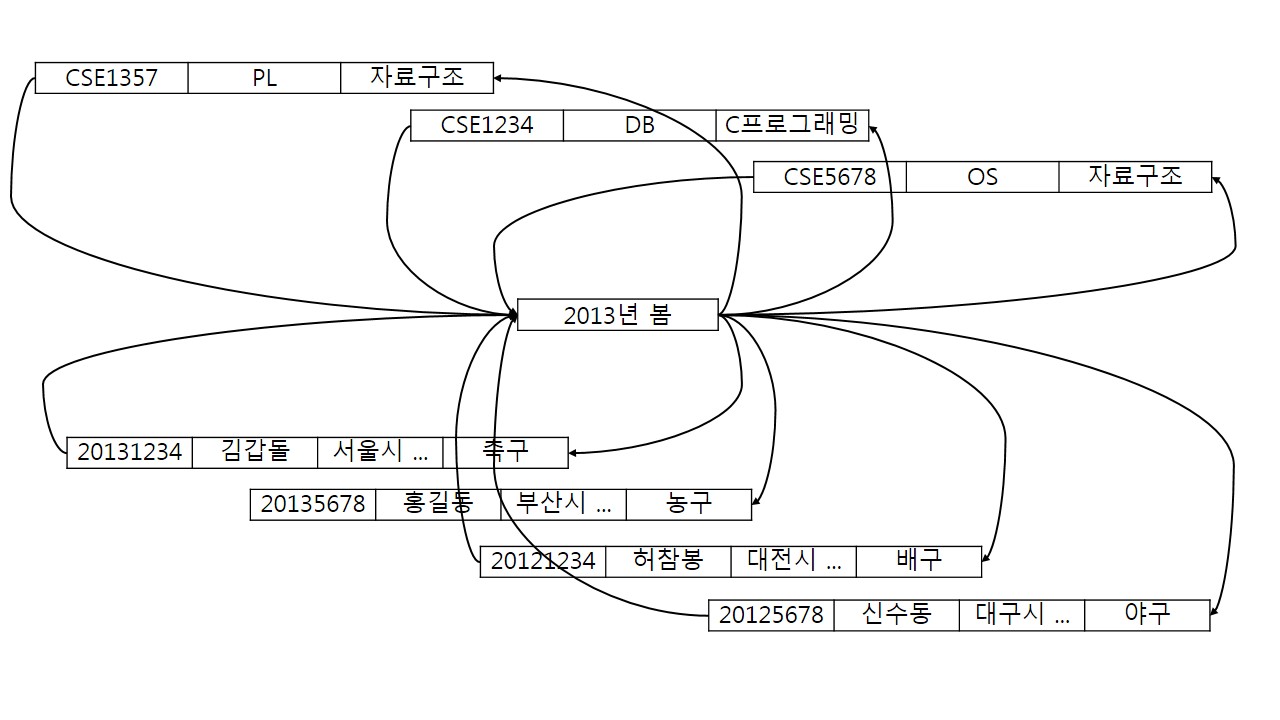
계층적 구조를 사용하는 경우 같은 데이터가 중복되어 저장된다. 같은 데이터가 여러 곳에 중복되어 저장된다면 수정이나 삭제 등의 작업에서 데이터의 무결성을 보장하기 위해 모든 데이터들에 대하여 복잡한 연산을 수행해야 한다. 즉, 데이터를 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는 연쇄 삭제 현상(삭제 이상, Deletion Anomaly)과 데이터의 속성 값을 갱신할 때 일부 데이터의 정보만 갱신되어 정보에 모순이 생기는 현상(갱신 이상, Update Anomaly)이 발생한다.

두 계층적 구조: 2점

Anomaly 설명: 2점

각각 설명이 부족하거나 부정확한 경우 1점씩 감점

다) 이러한 데이타의 네트워크 구조를 스케치하라. (3점)



네트워크 구조가 부정확한 경우 각 1점 감점

라) 위의 세 가지 모델의 각 경우데 다음의 질의들의 처리과정에 대한 차이점을 설명하라

Q1: "Database System을 수강하는 학생의 이름은?" (2점)

관계형 구조

do until no more 이수과목;

get next 이수과목

where 과목코드 = (DB과목코드);

print 이름 in STUDENT

where STUDENT.학번 = 이수과목.학번

end;

계층적 구조

get [next] COURSE where 과목코드 = (DB과목코드)

do until no more STUDENT under this COURSE;

get next STUDENT under this COURSE;

print 이름;

end;

네트워크 구조

get [next] COURSE where 과목코드 = (DB과목코드)

do until no more connectors under this COURSE;

get next connector under this COURSE;

get STUDENT over this connector;

Print 이름;

end;

Q2: "홍길동이 수강하는 과목은?" (2점)

관계형 구조

do until no more 이수과목;

get next 이수과목

where 학번 = (홍길동 학번);

print 과목명 in COURSE

where COURSE.과목코드 = 이수과목.과목코드

end;

계층적 구조

get [next] STUDENT where 학번 = (홍길동 학번)

do until no more COURSE under this STUDENT;

get next COURSE under this STUDENT;

print 과목명;

end;

네트워크 구조

get [next] STUDENT where 학번 = (홍길동 학번)

do until no more connectors under this STUDENT;

get next connector under this STUDENT;

get COURSE over this connector;

Print 과목명;

end;

각 구조에 따른 Q1, Q2의 Symmetric, Asymmetric 질의 차이에 대한 설명이 누락된 경우 0점

그 외 각 구조에 대한 설명이 누락되거나 잘못된 경우 1점씩 감점

5. 클라이언트-데이터베이스 서버 구조를 설명하고 이것이 갖는 장점을 기술하시오. 일반적인 Two-tier 클라이언트/서버 구조에서 발생하는 문제점을 기술하고, Three-tier 구조에서 이러한 문제들의 처리과정을 설명하라. 특히 웹 환경에 적합한 구조를 기술하고 그 이유를 설명하라. (10)

● 채점기준

(1)client-database server architecture 설명 (2 points)

(2)client-database server architecture 장점 기술 (1 point)

(3)client-database server architecture 문제점 기술 (1 point)

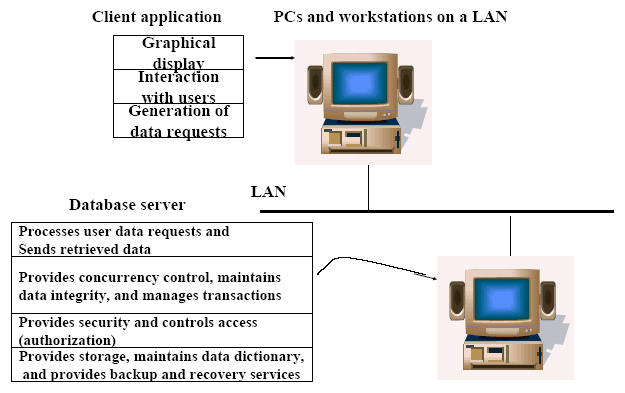
(4)Three-tier architecture에서client-database server architecture의 문제점 처리과정 기술 (2 points)

(5)web 환경에 적합한 구조 기술 (2 points)

(6)web 환경에 적합한 구조의 이유 설명 (2 points)

● 답안

(1)client-database server architecture 설명



client와 server의 임무를 명확히 나누지 못하면 1점 감점

(2)client-database server architecture 장점 기술

- 훨씬 효과적인 일의 분담을 제공한다.

- job에 대한 horizontal, vertical scaling을 제공한다. (작업의 효율적인 분할 가능)

- application은 일반적으로 smaller client computer configuration으로 좋은 performance를 제공한다.

- user들은 자신의 client에서 사용해왔던 tool을 그대로 사용할 수 있다.

- client는 더 많은 data에 접근 가능하다

- valuable data의 손실이나 부적절한access로부터 보호를 해준다.

3가지 이하 기술 시0.5점 감점

(3)client-database server architecture 문제점 기술

- 동시에 다수의 client가 server에 접속을 할 경우 서버에 부하가 생기며 시스템 전체에 정체 현상을 유발시킨다.

- fat client 문제 발생

- server로 사용되는 mainframe의 설치, 유지, 보수 비용이 크다.

- 개별 user의 독립성과 개인적인 system 사용이 어렵다.

- security cost가 높아진다.

2가지 이하 기술 시0.5점 감점

(4)Three-tier architecture에서 client-database server architecture의 문제점 처리과정 기술

- application server를 여러 개 배치하여 server의 load balancing이쉬워진다.

- fat client의 임무를 application server에 두어 client를 가볍게 한다.

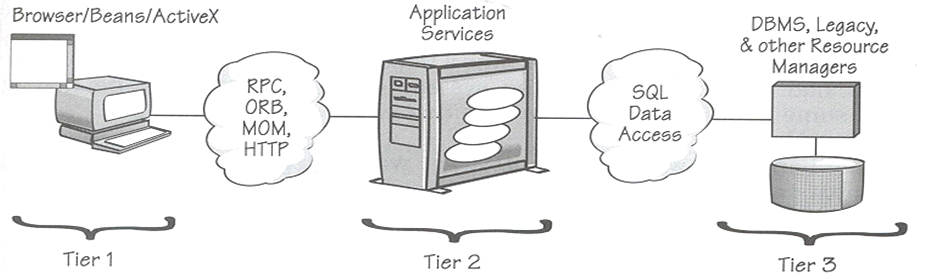
- application server의 설치, 유지, 보수 비용은 mainframe보다 저렴하기 때문에 incremental scale이 가능하다.

- application server의 적절한 변경을 통해 user의 요구 충족 기간이 2-tier보다 짧아진다.

- security를 위한 server를 따로 두어 관리한다.

2가지 이하 기술 시 1점 감점

(5)web 환경에 적합한 구조 기술



- Tier 1 (client) : user interface, perhaps performing some simple logic processing (e.g. input validation)

- Tier 2 (application server) : business logic, data processing logic

- Tier 3 (database server) : data validation, database access

3-tier에 크게 벗어나면 1점 감점

(6)web 환경에 적합한 구조의 이유 설명

web 환경에서 user는 헤아릴 수 없을 정도로 많기 때문에 기존의 2-tier 방식은 server에 상당한 부하를 주고 있다. 이에 비해 위에서 제안한 구조는 다음과 같은 장점이 있다.

- server의 기능이 대폭 줄어 비용감소 및 관리가 쉬워진다.

- 많은 user에 대한 business logic을 하나의 application server에 중앙 집중화 함으로써 application의 중앙집중화를 유도한다. (S/W의 분산에 대한 문제 제거)

- 하나의 tier가 다른 tier에 영향을 끼치지 않으면서 대체되거나 수정될 수 있다.

- core business를database function으로부터 분리시킴으로써 load balancing을 구현하기 쉽다.

위의 장점들은 3-tier가 web 환경에 적합하다는 근거이다.

한 개당 0.5점씩 감점

6. 부서-고용인(dept-employee) 데이터베이스를 가정한다.

DEPT(DEPT#, DNAME, BUDGET)

EMP(EMP#, ENAME, DEPT#, SALARY)

(a) 이에 대한 카탈로그(TABLES, COLUMNS)의 개략적인 구조를 스케치 하라. (3점)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TABLES   |  |  |  | | --- | --- | --- | | TABNAME | COLCOUNT | ROWCOUNT | | DEPT | 3 |  | | EMP | 4 |  | | COLUMNS   |  |  | | --- | --- | | TABNAME | COLNAME | | DEPT | DEPT# | | DEPT | DNAME | | DEPT | BUDGET | | EMP | EMP# | | EMP | ENAME | | EMP | DEPT# | | EMP | SALARY | |

설명이 부족한 경우 1점 감점

(b) Type(도메인)을 갖기 위해 카탈로그의 구조가 어떻게 확장되어야 하겠는가? (3점)

각 타입에 대한 엔트리를 포함하는 새로운 카탈로그 릴레이션 변수(TYPE)가 필요하다. 그리고 릴레이션 변수의 각각의 속성의 타입이 주어진 ATTRIBUTE 릴레이션 변수에 있는 새로운 속성(TYPENAME)도 필요하다.

|  |  |
| --- | --- |
| TYPENAME | ㆍㆍㆍ |

TYPE

|  |  |  |  |
| --- | --- | --- | --- |
| TABNAME | COLCOUNT | ROWCOUNT | ㆍㆍㆍ |

TABLES

|  |  |  |  |
| --- | --- | --- | --- |
| TABNAMES | COLNAME | TYPENAME | ㆍㆍㆍ |

COLUMNS

설명이 부족한 경우 1점 감점

(c) column EMP#을 갖는 모든 relvar 이름을 구하기 위한 질의를 작성하라. (3점)

SELECT TABNAME FROM COLUMNS WHERE COLNAME = "EMP#"

정확한 질의가 아니면 0점

(d) 릴레이션 PART\_STRUCTURE(MAJOR\_P#, MINOR\_P#, QTY)을 추가할 때 필요한 relvar 정의와 type 정의 집합을 CREATE문으로 작성하라. (4점)

TYPE P# POSSREP ( CHAR );

TYPE QTY POSSREP ( INTEGER );

CREATE TABLE PART\_STRUCTURE (

MAJOR\_P\_# P#,

MINOR\_P\_# P#,

QTY QTY,

PRIMARY KEY (MAJOR\_P\_#, MINOR\_P\_#))

TYPE 정의 1점

CREATE문 3점

일부 누락된 경우 1점씩 감점

(e) 이 relvar이 부서-고용인 데이터베이스에 포함되어 있다고 가정하자. 시스템이 카탈로그가 문제 (d)에 대한 답을 반영하도록 하기 위한 갱신을 보여라. (4점)

TYPE

|  |  |
| --- | --- |
| TYPENAME | ㆍㆍㆍ |
| P# | ㆍㆍㆍ |
| QTY | ㆍㆍㆍ |

TABLES

|  |  |  |  |
| --- | --- | --- | --- |
| TABNAME | COLCOUNT | ROWCOUNT | ㆍㆍㆍ |
| PART\_STRUCTURE | 3 | 0 | ㆍㆍㆍ |

COLUMNS

|  |  |  |  |
| --- | --- | --- | --- |
| TABNAME | COLNAME | TYPENAME | ㆍㆍㆍ |
| PART\_STRUCUTURE | MAJOR\_P# | P# | ㆍㆍㆍ |
| PART\_STRUCUTURE | MINOR\_P# | P# | ㆍㆍㆍ |
| PART\_STRUCUTURE | QTY | QTY | ㆍㆍㆍ |

잘못된 내용으로 갱신된 카탈로그를 보이면 각 1점씩 감점

(f) 다음의 질의를 Embedded SQL로 작성하라 (3점)

Q: "Get all employee's names whose BUDGET is given by the host variable BUDGET"

EXEC SQL SELECT EMP.ENAME

INTO :ENAME

FROM EMP, DEPT

WHERE EMP.DEPT# = DEPT.DEPT#

AND DEPT.BUDGET = :BUDGET

host variable을 표기 하지 않으면 2점 감점

기타 정확한 질의가 아니면 0점

7. ERD에 적합한 릴레이션 스킴들을 만드는 경우를 고려한다.

가) ERD 상의 composite attribute와 multivalued attribute는 어떻게 처리하는 것이 적당한가?

해답

Composite attribute는 각각의 composite attribute를 생성하여 나누어 처리할 수 있고, entity E의 multi-value attribute M의 경우 스키마 EM으로 구분하여 표현한다. 예를 들어 customer entity의 composite attribute인 name을 first\_name과 last\_name 의 composite attribute로 나누어 name.first\_name과 name.last\_name 으로 사용한다. Employee\_dependent\_names = (employee\_id, dname) 과 같이 각각의 multi-value attribute는 분리된 EM에 맵핑된다.

채점기준

두 개의 attribute의 처리방법을 적절하게 기술했다 (5점)

한 개의 attribute의 처리방법을 적절하게 기술했다 (3점)

문제의 발생, 해결방법, 그 이점에 대해 기술하면 (+1점)

나) ERD 상의 specialization을 어떻게 처리하는 것이 적당한가?

해답

ISA 컴포넌트를 통해 specialization을 처리한다. 이 때, 하위 레벨의 엔티티 셋이 상위 레벨의 엔티티 셋의 모든 속성을 물려받도록 처리한다. 예를 들어 고용인, 고객의 엔티티는 모두 ‘사람’이라는 엔티티의 ISA 관계로 설정되어 사람의 속성을 모두 가지도록 한다.

채점기준

Specialization을 위한 처리방안(질의 set에 따라 전부/primary key를 물려받아 처리하는 과정의 장단점)을 정확히 기술했다. (5점)

Specialization을 위한 처리방안을 일부 기술했다. (3점)