데이터 모델링

문제5) 개념적 데이터 모델링, 논리적 데이터 모델링, 물리적 데이터모델링

개념적 데이터 모델링	전사적 데이터 모델링을 수행할 때 많이 하며, 추상화 수준이 높고 업무중심적이며 포괄적인 수준의 모델링을 진행한다.
논리적 데이터 모델링	시스템으로 구축하고자 하는 업무에 대해 Key, 속성, 관계 등을 정확하게 표현하며 재사용성이 높다.
물리적 데이터모델링	실제 데이터베이스에 이식할 수 있도록 성능, 저장 등의 물리적인 성격을 고려한 데이터 모델링.

문제6) 외부스키마, 개념스키마, 내부스키마 및 독립성

외부스키마 (사용자 관점)	각 사용자가 보는 데이터베이스의 스키마를 정의
개념스키마 (통합된 관점)	모든 사용자가 보는 데이터베이스의 스키마를 통합하여 전체 데이터베이스를 나타내는 것. 데이터 베이스에 저장되는 데이터들을 표현하고 데이터들 간의 관계를 나타낸다
내부스키마 (물리적 관점)	물리적인 저장구조를 나타낸다. 실직적인 데이터의 저장 구조나 컬럼 정의, 인덱스 등이 포함된다.

논리적 독립성	개념스키마가 변경되어도 외부 스키마는 영향받지 않는다.
물리적 독립성	내부 스키마가 변경되어도 외부/개념 스키마는 영향을 받지 않는다.

문제 8) 엔터티 배치

중요한 엔터티를 오른쪽 상단에 배치
추가 발생되는 엔터티를 왼쪽편과 하단에 배치하는 것이 원칙

문제 10, 15) 엔터티의 특징

영속적으로 존재하는 '두개 이상'의 인스턴스 집합이어야 한다.
하나의 엔터티는 '두개 이상'의 속성을 갖는다.

문제 21) 존재적 관계, 행위적 관계

		UML 표기법	ERD 표기법
존재적 관계	소속한다, 소속된다 등	실선	구분 없이 단일화 된 표기법
행위적 관계	주문한다, 출석한다 등	점선	

문제 23) 두개의 엔티티 사이에서 관계를 도출할 때 체크할 사항

- 두 개의 엔터티 사이에 *관심 있는 연관규칙이 존재* 하는가?
- 두 개의 엔터티 사이의 *정보의 조합이 발생*되는가?
- 업무기술서, 장표에 관계 *연결을 가능하게 하는 동사(Verb)* 가 있는가?
- 업무기술서, 장표에 관계 *연결에 대한 규칙이 서술* 되어 있는가?

문제12) 엔티티의 분류

유/무형에 따른 엔티티 분류	
유형 엔터티	물리적인 형태가 있고 안정적이며 지속적으로 활용되는 엔터티 업무로부터 엔터티를 구분하기가 가장 용이함 ex) 사원, 물품, 강사
개념 엔터티	물리적인 형태 존재X 관리해야 할 개념적 정보로 구분이 되는 엔터티 ex) 조직, 보험상품
사건 엔터티	업무를 수행함에 따라 발생되는 엔터티 비교적 발생량이 많으며 각종 통계자료에 이용 ex) 주문, 청구, 미납

발생시점에따른 엔티티 분류	
기본 엔터티	업무에 원래 존재하는 정보 다른 엔터티와 관계에 의해 생성되지 않고 독립적으로 생성이 가능 (자신은 타 엔터티의 부모 역할) 타 엔터티로부터 주식별자를 상속받지 않고 자신의 고유한 주식별자를 가짐 ex) 사원, 부서, 고객, 상품, 자재
중심 엔터티	기본 엔터티로부터 발생 업무의 중심 역할 데이터의 양이 많이 발생되고 다른 엔터티와의 관계를 통해 많은 행위 엔터티 생성 ex) 계약, 사고, 예금원장, 청구, 주문, 매출
행위 엔터티	두 개 이상의 부모 엔터티로부터 발생 자주 내용이 바뀌거나 데이터량이 증가함 상세 설계단계나 프로세스와 상관모델링을 진행하면서 도출 ex) 주문목록, 사원변경이력

문제26) 식별자

분류	식별자	내용	
대표성 여부	**주식별자**	엔터티 내에서 각 어커런스를 구분할 수 있는 구분자이며, 타 엔터티와 참조 관계를 연결할 수 있는 식별자	·유일성, 최소성, 불변성, 존재성을 가진 대표 식별자 **·다른 엔터티와 참조관계로 연결**
	보조식별자	엔터티 내에서 각 어커런스를 구분할 수 있는 구분자이나 대표성을 가지지 못해 참조관계 연결을 못함	·인스턴스를 식별할 수는 있지만 대표 식별자가 아님 **·다른 엔터티와 참조 관계로 연결되지 않음**
스스로 생성 여부	**내부식별자**	엔터티 내부에서 스스로 만들어지는 식별자	
	외부식별자	타 엔터티와의 관계를 통해 타 엔터티로부터 받아오는 식별자	다른 엔터티와 연결고리 역할 (FK)
속성의 수	**단일식별자**	하나의 속성으로 구성된 식별자	
	복합식별자	둘 이상의속성으로 구성된 식별자	
대체 여부	**본질(원조)식별자**	업무에 의해 만들어 지는 식별자	가공되지 않은 원래의 식별자
	인조(대리)식별자	업무적으로 만들어지지는 않지만 원조 식별자가 복잡한 구성을 가지고 있기 때문에 인위적으로 만든 식별자	주식별자의 속성이 두 개 이상인 경우 그 속성들을 하나로 묶어서 사용하는 식별자

문제 32, 41, 42) 반정규화 기법 및 슈퍼/서브타입 예

<테이블의 반정규화>

테이블 병합, 분할, 추가 기법이 있다.

병합 : 1:1, 1:M, 서브/슈퍼 타입 → 3가지 유형의 테이블 병합

분할 : 수직, 수평 → 2가지 유형의 테이블 분할

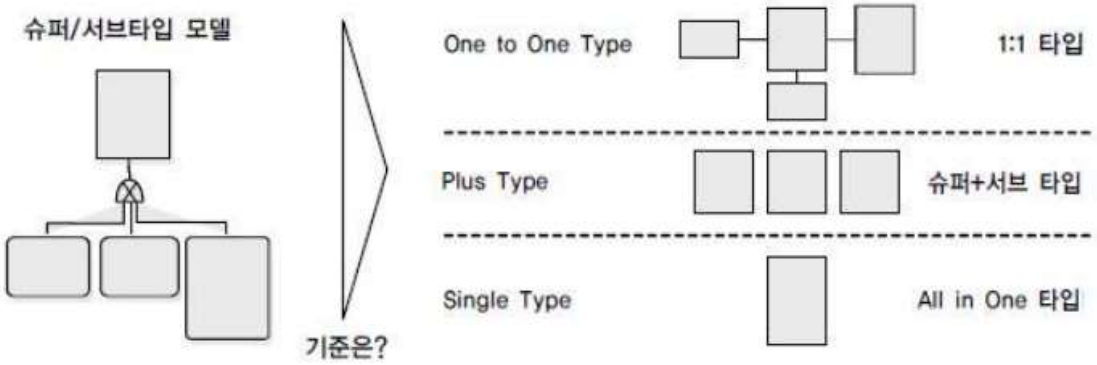
추가 : 중복테이블, 통계테이블, 이력테이블, 부분테이블 → 3가지 유형의 테이블 추가

기법 분류	반정규화 기법	내용
중복컬럼 추가	**1:1 관계 테이블 병합**	1:1 관계를 통합하여 성능향상
	1:M 관계 테이블 병합	1:M 관계를 통합하여 성능향상
	슈퍼/서브타입 테이블 병합	슈퍼/서브 관계를 통합하여 성능 향상
테이블 분할	**수직분할**	컬럼단위의 테이블에서 디스크 I/O를 분산처리 하기 위해 테이블을 1:1로 분리하여 성능 향상 (트랜잭션의 처리되는 유형을 파악하는 것이 선행되어야 한다.)
	수평분할	로우 단위로 집중 발생되는 트랜잭션을 분석하여 디스크 I/O 및 데이터접근의 효율성을 높여 성능을 향상하기 위해 로우단위로 테이블을 쪼갠다. (관계가 없음)
테이블 추가	**중복 테이블 추가**	다른 업무이거나 서버가 다른 경우 동일한 테이블 구조를 중복하여 원격 조인을 제거하여 성능을 향상
	통계 테이블 추가	SUM, AVG 등을 미리 수행하여 계산해 둠으로써 조회시 성능을 향상
	이력 테이블 추가	이력 테이블 중에서 마스터 테이블에 존재하는 레코드를 중복하여 이력 테이블에 존재하는 방법
	부분 테이블 추가	하나의 테이블의 전체 컬럼 중 자주 이용하는 집중화된 컬럼들이 있을 때 디스크 I/O를 줄이기 위해 해당 컬럼들을 모아놓은 별도의 반정규화된 테이블을 생성

<컬럼의 반정규화> 중복컬럼, 파생컬럼, 이력테이블컬럼, PK에 의한 컬럼, 응용시스템 오작동을 위한 컬럼 → 5가지 유형

반정규화 기법	내용
중복컬럼 추가	조인에 의해 처리할 때 성능 저하를 예방하기 위해 즉, 조인을 감소시키기 위해 중복된 컬럼을 위치시킴 *조인 감소를 위해 여러 테이블에 동일한 컬럼을 갖도록 한다.*
파생 컬럼 추가	트랜잭션이 처리되는 시점에 계산에 의해 발생되는 성능저하를 예방하기 위해 미리 값을 계산하여 컬럼에 보관한다. (Derived Column 이라고 함) *조회 성능을 우수하게 하기 위해 미리 계산된 컬럼을 갖도록 한다.*
이력테이블 컬럼 추가	대량의 이력데이터를 처리할 때 불특정 날 조회나 최근 값을 조회할 때 나타날 수 있는 성능 저하를 예방하기 위해 이력 테이블에 기능성 컬럼(최근값 여부, 시작과 종료일자 등) 을 추가함 *최신값을 처리하는 이력의 특성을 고려하여 기능성 컬럼을 추가한다.*
PK에 의한 컬럼 추가	복합의미를 갖는 PK를 단일 속성으로 구성하였을 경우 발생. 단일 PK안에서 특정 값을 별도로 조회하는 경우 성능 저하 발생 가능성 존재 이미 PK안에 데이터가 존재하지만 성능 향상을 위해 일반 속성으로 포함하는 방법
응용시스템 오작동을 위한 컬럼 추가	업무적으로는 의미가 없지만 사용자가 데이터처리를 하다가 잘못 처리하여 원래 값으로 복구하기를 원하는 경우 이전 데이터를 임시적으로 중복하여 보관하는 기법. 컬럼으로 이것을 보관하는 방법은 오작동 처리를 위한 임시적인 기법이지만 이것을 이력데이터 모델로 풀어내면 정상적인 데이터 모델의 기법이 될 수 있음.

<슈퍼/서브타입 테이블 병합> 예시



슈퍼/서브타입을 성능을 고려한 물리적인 데이터모델로 변환하는 기준
→ 데이터 양과 해당 테이블에 발생되는 트랜잭션의 유형에 따라 결정

구분	One To One Type	Plus Type	Single Type
특징	개별 테이블 유지	슈퍼 + 서브타입 테이블	하나의 테이블
확장성	우수	보통	나쁨
조인성능	나쁨	나쁨	우수함
I/O량 성능	좋음	좋음	나쁨
관리 용이성	나쁨	나쁨	좋음(1개)
트랜잭션 유형에 따른 선택 방법	개별 테이블로 접근이 많은 경우	슈퍼+서브 형식으로 데이터를 처리하는 경우	전체를 일괄적으로 처리하는 경우

문제 33) 성능을 고려한 데이터 모델링의 순서

1) 데이터 모델링을 할 때 정규화를 정확하게 수행한다.
2) 데이터베이스 용량산정을 수행한다.
3) 데이터베이스에 발생되는 트랜잭션의 유형을 파악한다.
4) 용량과 트랜잭션의 유형에 따라 반정규화를 수행한다.
5) 이력모델의 조정, PK/FK조정, 슈퍼타입/서브타입 조정 등을 수행한다.

문제 34) 정규화

테이블 분해로 많은 조인 연산시 성능저하가 발생할 수 있으나 항상은 아니다.
중복 데이터 제거로 갱신 이상 등의 성능저하 현상을 방지할 수 있다.

문제38) 1차 정규화, 2차 정규화

	상황	관계 차수
1차 정규화	하나의 속성에 단 하나의 값만 존재 (도메인이 원자값)	?
	컬럼단위에서 중복된 테이블 (중복성이 있는 속성에 대해)	1:M관계로 분류
2차 정규화	Composite Key를 단일 키로 분해 연관 종속 컬럼을 포함하여 하나의 엔터티 테이블로 분해한다. (부분 함수적 종속성 제거)	1:M관계로 분류

문제 40) 빌링의 잔액 반정규화 대상 이유

반정규화 정보에 대한 재현의 적시성으로 판단한다. 예를들어, 빌링의 잔액(balance)은 다수 테이블에 대한 다량의 조인이 불가피하므로 데이터 제공의 적시성 확보를 위한 필수 반정규화 대상 정보이다.

빌링의 잔액 데이터를 조회하기 위해서는 다수 테이블에 의해 다량의 조인이 필요하므로

문제 45) [로우체이닝] 컬럼수가 많은 테이블에 대한 설명

로우 체이닝	로우의 길이가 너무 길어서 데이터 블록 하나에 데이터를 모두 저장하지 않고 두 개 이상의 블록에 걸쳐서 하나의 로우가 저장되는 형태
로우 마이그레이션	데이터 블록에서 수정이 발생하면 수정된 데이터를 해당 데이터 블록에서 저장하지 못하고 다른 블록의 빈 공간을 찾아 저장하는 방식

문제 48) 논리모델의 슈퍼타입과 서브타입 데이터 모델을 물리적인 테이블 형식으로 변환할 때 성능저하 이유

1. 트랜잭션은 항상 일괄로 처리하는데 테이블은 개별로 유지되어 Union연산에 의해 성능이 저하될 수 있다.

2. 트랜잭션은 항상 서브타입 개별로 처리하는데 테이블은 하나로 통합되어 있어 불필요하게 많은 양의 데이터가 집약되어 있어 성능이 저하되는 경우가 있다.

3. 트랜잭션은 항상 슈퍼+서브 타입을 공통으로 처리하는데 개별로 유지되어 있거나 하나의 테이블로 집약되어 있어 성능이 저하되는 경우가 있다.

- 트랜잭션 항상 일괄처리

→ 테이블 개별로 유지 → UNION연산에 의해 성능저하

- 트랜잭션 항상 서브타입 개별로처리

→ 테이블 하나로 통합 → 불필요하게 많은양의 데이터 집약으로 성능 저하

- 트랜잭션 항상 슈퍼/서브타입 공통처리

→ 개별 or 하나 → 성능 저하

슈퍼/서브타입을 성능을 고려한 물리적인 데이터모델로 변환하는 기준

→ 데이터 양과 해당 테이블에 발생되는 트랜잭션의 유형에 따라 결정

SQL 기본

문제 19) 참조무결성

표준 SQL에서 테이블 생성시 참조관계를 정의하기 위해 외래키를 선언한다. 관계형 데이터베이스에서 Child Table의 FK 데이터 생성시 Parent Table에 PK가 없는 경우, Child Table 데이터 입력을 허용하지 않는 참조동작인 것은?

DELETE(/UPDATE)	
CASCADE	Master 삭제시 Child 같이 삭제
SET NULL	Master 삭제시 Child 해당 필드 NULL
SET DEFAULT	Master 삭제시 Child 해당 필드 Default값으로 설정
RESTRICT	Child 테이블에 PK 값이 없는 경우만 Master 삭제 허용
NO ACTION	참조 무결성을 위반하는 삭제/수정 액션을 취하지 않음

INSERT (일반적으로 외래키 제약조건은 삭제 시에 동작을 지정하는 것이 주로 사용함. 따라서 INSERT 작업을 제한하려면 애플리케이션 수준에서 해당 로직을 구현하여 제한 조건을 처리하는 것이 일반적)	
AUTOMATIC	Master 테이블에 PK가 없는 경우 Master PK를 생성 후 Child입력
SET NULL	Master 테이블에 PK가 없는 경우 Child 외부키를 Null 값으로 처리
SET DEFAULT	Master 테이블에 PK가 없는 경우 Child 외부키를 지정된 기본값으로 입력
DEPENDENT	Mater 테이블에 PK가 존재할 때만 Child 입력을 허용
NO ACTION	참조 무결성을 위반하는 입력 액션을 취하지 않음.

문제 23) DROP TRUNCATE DELETE

	로그여부	Auto Commit	삭제 및 수행 내용
DROP	x	o	테이블 구조 완전 삭제
DELETE	o	x	테이블 내 데이터만 삭제
TRUNCATE	x	o	테이블 내 모든행 제거, 저장공간 해제(재사용 가능하도록)

문제 28) 데이터베이스 트랜잭션에 대한 격리성이 낮은 경우 발생할 수 있는 문제점

Dirty Read, Non-repeatable Read, Phantom Read, Lost Update

Dirty Read	하나의 트랜잭션이 아직 커밋되지 않은 다른 트랜잭션의 변경사항을 읽는 경우, 잘못된 데이터를 읽을 수 있다. 이로 인해 커밋 되지 않은 변경 사항이 롤백되면 읽은 데이터는 실제로 존재하지 않는 데이터가 된다.
Non-repeatable Read	동일한 트랜잭션 내에서 동일한 쿼리를 반복해서 실행할 때, 다른 트랜잭션이 해당 데이터를 변경하거나 삭제하는 경우, 처음과 다른 결과를 얻게 된다. 이로 인해 데이터의 일관성이 깨질 수 있다.
Phantom Read	동일한 쿼리를 반복해서 실행할 때, 다른 트랜잭션이 해당 데이터를 삽입 또는 삭제하는 경우, 처음과 다른 결과를 얻게 된다. 이로 인해 조회 결과에 예상치 못한 데이터가 추가되거나 제거될 수 있다.
Lost Update	두 개 이상의 트랜잭션이 동일한 데이터를 동시에 수정하는 경우, 마지막으로 커밋된 트랜잭션의 변경사항이 덮어 씌워져 이전 변경사항이손실될 수 있다.

문제 10) UNIQUE

테이블 내에서 중복되는 값이 없으며 NULL입력이 가능하다.
(NULL을 방지하기 위해서는 primary key로 지정하거나 NOT NULL 제약조건을 추가해야 한다.)

문제 29) DDL AUTO COMMIT

ORACLE : 자동 / SQL SERVER : 수동
ex) CREATE, DROP, ALTER 등

문제37) VARCHAR 컬럼에 '' 공백 문자열 INSERT시

ORACLE : null로 입력됨 / SQL SERVER : '' 공백 그대로 입력됨

문제40) 다중행에서의 단일행 함수 적용

(다) 1:1 관계의 두 테이블을 조인할 경우 M쪽에 다중행이 출력되므로 단일행 함수는 사용할 수 없다.
풀이 : M쪽에 다중행이 출력되더라도 다중행의 각 행(row)별로 단일행 함수가 적용된다.

문제 44) CASE WHEN 구문

SEARCH : CASE WHEN 컬럼명 = '값' THEN '값'
SIMPLE : CASE 컬럼명 WHEN '값' THEN '값'
(ELSE 생략시 DEFAULT값은 NULL이다.)

문제 46) 단일행 NULL관련 함수

NULLIF(a, b)	a와 b가 같으면 NULL 다르면 a반환
NVL / ISNULL(a, b)	a가 NULL이면 b 출력 NULL이 아니면 a출력
COALESCE(a, b, ...)	인수 중 NULL이 아닌 최초값 출력

문제 52) 인라인 뷰

FROM절에 선언한 인라인 뷰 서브쿼리는 FROM절에 선언한 다른 테이블을 직접 참조할 수 없다.
(WHERE절에 오는 연관 서브쿼리는 가능하다.)

문제 56) GROUP BY 메모리 적재

쿼리 유형	메모리 적재 방식
GROUP BY 하지 않은 쿼리	테이블에 있는 전체 컬럼을 로드
GROUP BY 가 있는 쿼리	GROUP BY에 지정한 컬럼 SELECT에 지정한 컬럼 ORDER BY에 명시한 집계함수의 결과값

문제 63) Non Equi Join

[마] 대부분 Non Equi Join을 수행할 수있지만, 때로는 설계상의 이유로 수행이 불가능한 경우도 있다.
일반적으로 거의 다 된다.

Chat GPT 답변

1. 데이터 모델링의 무결성 제약 조건: 일부 데이터 모델링 방법론에서는 테이블 간의 조인에 제약을 두는 경우가 있습니다. 예를 들어, 개체-관계 모델에서 일대다 관계에 있는 테이블을 조인할 때 등호(=)를 사용하는 것이 권장되는 경우가 있습니다. 이 경우 non equi join은 모델링 규칙에 어긋나므로 허용되지 않을 수 있습니다.
2. 성능 제약: 일부 데이터베이스 관리 시스템에서는 non equi join을 처리하는 것이 성능상 불리할 수 있습니다. 비교 연산자에 의해 생성되는 조인 조건이 복잡하거나 인덱스를 활용하기 어려운 경우, 데이터베이스 시스템은 비효율적인 실행 계획을 생성할 수 있습니다. 따라서 설계상의 이유로 non equi join을 허용하지 않을 수 있습니다.
3. 쿼리 최적화의 어려움: non equi join은 일반적인 조인 조건보다 훨씬 복잡할 수 있습니다. 쿼리 옵티마이저는 조인 조건을 기반으로 최적의 실행 계획을 수립하는데, non equi join의 경우 이 최적화 작업이 어려울 수 있습니다. 따라서 데이터베이스 시스템이 non equi join을 지원하지 않는 경우가 있을 수 있습니다.

SQL 활용

문제 70) 그룹핑 함수 집계된 컬럼 이외의 대상 컬럼값

USING조건, NATURAL JOIN 사용시 JOIN컬럼으로 Alias 혹은 테이블 이름과 같은 접두사 사용 불가
ex1) SELECT * FROM TAB1 NATURAL JOIN TAB2
ex2) SELECT * FROM TAB1 JOIN TAB2 USING(COLA)

문제 76) OUTER JOIN의 ON조건

OUTER JOIN에서 ON 조건절은 조인할 대상을 결정한다. 이때 기준 테이블은 항상 모두 표시한다.

문제 82) 집합연산자

UNION ALL 사용시 중복을 포함하며 컬럼 Alias는 첫 번째 모듈(SELECT절) 기준으로 표시된다.
UNION, INTERSECT, MINUS(EXCEPT) 각각의 연산자들은 모두 중복을 제거하여 연산한다.

문제 88) Oracle 계층형 질의

start with	계층 구조의 시작점
order siblings by	형제노드 사이에서의 정렬
순방향 / 역방향 (prior 위치)	부모-> (자식) / (자식) -> 부모
루트노드 LEVEL	1

connect by 절에 작성된 조건절은 where절에 작성되는 조건절과는 다르다.
start with절에서 필터링 된 시작 데이터는 결과목록에 무조건 포함되어 지고
이후 connect by절에 의해 필터링된다.

문제 95) 서브쿼리의 결과가 복수행 결과를 반환하는 경우

(다) '=', '<=', '>=' 등의 연산자와 함께 사용이 가능하다.
복수행 비교연산자는 in all any가 있지만 다중행 비교연산자가 단일행 비교연산자도 사용이 가능하다.
지문이 이 틀린 이유는 복수행 비교연산자의 예시를 들고, 단일행 연산자 '도' 함께 사용이 가능하다. 라고 적혀있어 맞다.

문제 95) 다중컬럼 서브쿼리

(마) 다중컬럼 서브쿼리란 서브쿼리의 결과가 여러개의 컬럼값을 반환하는 형태이다.
ex) SELECT * FROM T1 WHERE (a, b, c) IN(SELECT a, b, c FROM T2)
(SQL Server에서는 지원하지 않는다.)

문제 98) GROUP BY 생략된 HAVING절

전체 테이블의 데이터가 하나의 그룹으로 간주된다.

문제 108) 그룹핑 함수 집계된 컬럼 이외의 대상 컬럼값

CUBE, ROLL UP, GROUPING SETS 함수들의 대상 컬럼 중 집계된 컬럼 이외의 대상 컬럼 값은 모두 NULL 값을 반환한다.

문제 106) GROUPING(컬럼명)

소계에 참여한 컬럼일 경우 0 반환
소계에 참여하지 않은 컬럼일 경우 1 반환

연관/ 비연관 서브쿼리

연관 서브쿼리	메인쿼리로부터 컬럼값을 제공받아 (비교등의 연산 수행) 사용
비연관 서브쿼리	메인쿼리에 값을 제공하기 위한 목적

GROUPING 함수

ROLLUP(a, b, c)	n+1레벨 구성 abc, ab, a, 총합계 4가지 그룹핑
CUBE(a, b, c)	2의 n제곱 레벨 구성 abc, ab, ac, bc, a, b, c, 총합계 8가지 그룹핑
GROUPING SETS(a, b, c, ( )) GROUPING SETS(a, b, ROLLUP(c)) GROUPING SETS(a, ROLLUP(b, c))	a, b, c, 총합계 ( )공백 괄호 혹은 C를 ROLLUP으로 중첩 그룹핑시 총합계 계산 a, (b,c), b, 총합계

ROLLUP(a, (b, c)) → a, (b, c) / a / 총합계
ROLLUP((a, b), c) → (a, b), c / (a, b) / 총합계

CUBE(a, (b, c)) → a, (b, c) / a / (b,c) / 총합계
CUBE((a, b), c) → (a, b), c / (a, b) / c / 총합계

GROUPING SETS(a, (b, c)) → a / (b,c)

문제 109) CUBE함수

CUBE (a, b, (a,b))
→ a, b, (a, b) 즉 [a, b]로 그룹핑
→ [a, b]로 그룹핑
→ a, (a, b) 즉 [a, b]로 그룹핑
→ b, (a, b) 즉 [a, b]로 그룹핑
→ [a]
→ [b]
→ [a, b]
→ 총합계

→[a, b]로 총 5번 그룹핑 되었다.

윈도우 함수

RANK	동일값에 대해 동일 순위를 부여하고 중간 순위는 비워둠 ex) 1 1 3 4 4 6 혹은 1 1 1 4 5 5 7
DENSE_RANK	동일 순서를 부여하되 중간 순위를 비우지 않는다 ex) 1 1 2 3 3 4 혹은 1 1 1 2 3 3 4
ROW_NUMBER	동일 값에 대해 유일한 순위를 부여한다 → 오라클의 경우 낮은 rowid 우선순위 (DBMS 벤더별로 다른 결과)

FIRST_VALUE	파티션별 가장 선두에 위치한 데이터를 구하는 함수 파티션별 윈도우에서 가장 먼저 나온 값을 구한다. MIN함수를 활용하여 같은 결과를 얻을 수도 있다.	
LAST_VALUE	파티션별 가장 끝에 위치한 데이터를 구하는 함수 파티션별 윈도우에서 가장 나중에 나온 값을 구한다. MAX함수를 활용하여 같은 결과를 얻을 수도 있다.	
LAG	파티션별 윈도우에서 이전 몇 번째 행의 값 파티션별 윈도우에서 이전 몇 번째 행의 값을 가져올 수 있다.	
LEAD	파티션별 윈도우에서 이후 몇 번째 행의 값 파티션별 윈도우에서 이후 몇 번째 행의 값을 가져올 수 있다.	
RATIO_TO_REPORT	파티션 별 합계에서 차지하는 비율을 구하는 함수 SUM(컬럼)값에 대한 행별 컬럼 값의 백분율을 소수점으로 구한다. (0 < 결과값 <= 1) 개별 RATIO의 합을 구하면 1이 된다.	
PERCENT_RANK	해당 파티션이 맨 위 끝 행을 0, 맨 아래 끝 행을 1로 놓고 현재 행이 위치하는 백분위 순위 값 값이 아닌 행의 순서별 백분율을 구한다. (0 <= 결과값 <= 1)	
CUME_DIST	해당 파티션에서의 누적 백분율을 구하는 함수이다. 현재 행보다 작거나 같은 건수에 대한 누적 백분율을 구한다. (0 < 결과값 <= 1)을 가진다.	
NTILE	주어진 수만큼 행들을 N등분 한 후 현재 행에 해당하는 등급을 구하는 함수이다. 파티션별 전체 건수를 ARGUMENT값으로 N등분한 결과를 구한다.	

윈도우 함수 옵션 RANGE, ROW등

범위	의미
UNBOUNDED PRECENDING	위쪽 끝 행
UNBOUNDED FOLLOWING	아래쪽 끝 행
CURRENT ROW	현재 행
n PRECENDING	현재 행에서 위로 n만큼 이동
n FOLLOWING	현재 행에서 아래로 n만큼 이동

기준	의미
ROWS	행 자체가 기준이 된다.
RANGE	행이 가지고 있는 데이터 값이 기준이 된다.

- **RANGE BETWEEN UNBOUNDED PRECENDING AND CURRENT ROW**
  처음부터 현재 행까지, RANGE UNBOUNDED PRECENDING과 같은 의미
- **RANGE BETWEEN 10 PRECENDING AND CURRENT ROW**
  현재 행이 가지고 있는 값보다 10만큼 적은 행부터 현재 행까지, RANGE 10 PRECENDING과 같은 의미
- **ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING**
  현재 행부터 끝까지
- **ROWS BETWEEN CURRENT ROW AND 5 FOLLOWING**
  현재 행부터 아래로 5만큼 이동한 행까지

기출 30회

문제 3) 발생 시점에 따른 엔터티 분류에 의한 중심 엔터티가 아닌 것은?

[사원]은 기본 엔터티이며 발생 시점에 따른 엔터티는 [매출, 계약, 주문] 이다.

문제 5) 논리적 모델링

데이터 모델링이 최종적으로 완료된 상태라고 정의, 물리적 스키마 설계 전 단계

문제 7) 반정규화 대상이 아닌 것

테이블에 지나치게 많은 조인과 SORTING, ORDER BY 프로세스가 많은 경우

문제 11)

COL1의 데이터 타입은 VARCHAR인데 정수로 비교하면 자동으로 형변환이 되지 않는다.
조건절 비교시 서로 다른 데이터 타입일 경우 자동 형변환은 기준컬럼이 정수이고, 비교 컬럼이 문자일 경우에만 해당된다. (컬럼에 정수로 변경되지 않는 문자가 들어있기 때문에 비교되지 않은 것 같다.)

문제 15) IN(NULL)

IN연산자에서 NULL은 연산에 포함하지 않는다. (일반적 집계 함수라고 생각.)

문제 16) NL조인 수행 순서

[테이블 FULL SCAN] - [인덱스 검색] – [조인키 기준 인덱스 테이블 스캔] - [NL JOIN]
[인덱스 검색] – [조인키 기준 인덱스 테이블 스캔] - [NL JOIN] - [SELECT]

문제 17) WHERE 1=2 일 경우 COUNT(*)

집계 함수 COUNT(*)을 출력할 때 WHERE 조건절이 거짓일 경우 0을 반환한다.

문제 25) CHARACTER 데이터 타입 특징

할당된 변수 값의 길이가 고정 길이보다 작을 경우 차이 길이만큼 공백으로 채워짐(VARCHAR는 가변)

문제 30) INNER JOIN

INNER JOIN시 상품을 주문한 고객만 해당 (모든 회원 = 상품주문을 하지 않은 고객도 포함한다)

문제 36) LIST PARTITION

데이터가 매우 많은 대용량테이블, 데이터생성일자를 구분짓는 특정컬럼이 없는 형태 (PK : 지점, 코드)

문제 44) VIEW

복잡한 질의를 단순하게 작성, 사용자에게 정보를 감출수 있다. **실제 데이터를 가지고 있지 않다.**

문제 45) 계층구조 역방향 전개

상위 계층이 IS NULL인 행을 기준으로 역방향 전개시 기준이 최상위 계층이므로 해당 로우만 출력

문제 46) 테이블 컬럼 변경

오라클 : MODIFY / SQL Server : ALTER COLUMN

문제 47) DEFAULT 제약조건

DEFAULT 제약조건시 INSERT문을 통해 인위적으로 NULL을 입력하면 DEFAULT값이 반영이 되지 않으며, INSERT시 해당 컬럼을 제외하고 값을 추가하였을 때 DEFAULT값이 삽입된다.

문제 49) WINDOW 옵션

주문 금액을 기준으로 내림차순하여 DENSE_RANK를 산정 **정답 : 주문금액 DESC**

기출 33회

문제 1) RANGE PARTITION

대상 테이블이 날짜 또는 숫자 값으로 분리가 가능, 각 영역별 트랜잭션 분리,
데이터보관주기에 따라 테이블 데이터 삭제가 쉽다.

문제 10) 반정규화 중복관계 추가

조인이 복잡한 여러 테이블 존재할때 중복관계를 추가한다.

문제 20) HASH JOIN 순서

1. 선행테이블 조건 필터링 – 2. 조인키 기준 해시함수 적용 해시테이블 생성
3. 선행테이블 작업 - 조건만족 모든 행 수행
4. 후행테이블 조건 필터링 - 5. 조인키 기준 해쉬함수 적용 선행테이블 해시함수 반환값과 매핑 - 검색

문제 23) ROWS BETWEEN 1 PRECENDING AND FOLLOWING

PARTITION BY에 의해서 3개의 행씩 계산한다.
1(1,2) 2(1,2,3) (2,3) 3(3,4) 4(3,4,5) 5(4,5) 6(6,7) 7(,7,8) 8(7,8,9) 9(8,9) 괄호 합산

문제 26) HASH JOIN

HASH JOIN은 등가 조인만 가능하다.
등가, 비등가 모두 사용 가능한 조인은 SORT MERGE JOIN이다.

문제 27) ROLE

다양한 유저에게 다양한 권한을 효율적으로 관리하기 위한 권한들의 집합(GRANT, REVOKE)

문제 30) UNION ALL과 DISTINCT

집합연산 수행시 각각의 모듈단위별로 DISTINCT를 먼저 진행한 뒤 집합연산을 수행한다.

문제 32) CROSS JOIN

A-(NULL,B) (1-2개) / NULL-(A,B) (2-2개) / B-(A,NULL) (3-2개) / C-(A,B,NULL) (4-3개)
이때 NULL은 제외한다. A 1*1개 / NULL 제외(0개) / B 3*1개 / C 4*2개
1+0+3+8 = 12이다.

문제 35) >= ANY

문제 44) LAG와 LEAD

LAG(A,N,D) : 컬럼 A를 대상으로 현재행 기준 이전 N번째 행 반환, 반환값이 NULL이면 D 반환
LEAD(A,N,D) : 컬럼 A를 대상으로 현재행 기준 이후 N번째 행 반환, 반환값이 NULL이면 D 반환
PARTITION BY에 의해 A-3개 B-3개 D-2개로 A, B, C의 경우 1 번째 행은 값을 제대로 가져오고
2 번째 행과 3번째 행은 이후 2번째행이 존재하지 않기 때문에 값이 9로 반환되었다. (답 2, 9)

문제 47) LPAD함수 2,'0'

LPAD(A, B, C) 문자데이터 A값을 B바이트로 변경하고 빈 공백을 앞부분부터 C로 채운다.
LPAD('1', 2, '0')을 가정
길이가 1인 문자 '1'의 길이를 2바이트로 변경하고 빈 공백을 앞부분부터 '0'으로 채운다.

문제 50) GROUPING SETS함수

결과값은 1레벨과 2레벨까지만 적용이 되었고 3레벨인 총합계는 생략되었다.
GROUPING SETS를 유추해볼 수 있다.
A컬럼 B컬럼 C컬럼이 존재한다고 가정할 때,
GROUPING SETS를 적용한다면 1레벨의 경우 A,B의 조합이고 2레벨의 경우는 A의 조합이다.

기출 34회

문제 13) PROCEDURE와 TRIGGER

PROCEDURE는 EXECUTE명령어로 수행 / TRIGGER는 DML이벤트 발생시 자동 수행
PROCEDURE는 TCL가능 / TRIGGER는 TCL불가능

문제 16) A, B 테이블 조인 실행계획 순서

TABLE ACCESS FULL A → TABLE ACCESS FULL B → HASH JOIN → SELECT STATEMENT
(선행테이블 FULLSCAN → 후행테이블 FULLSCAN → JOIN → SELECT)
HASH JOIN에서는 ROW가 적은 테이블을 선행으로 둔다.

NESTED LOOP JOIN	SORT MERGE JOIN	HASH JOIN
랜덤 액세스	등가, 비등가 조인 가능	등가조인만 가능
대용량 SORT작업에 유리	조인키 기준 정렬	대량작업 유리, 함수처리

문제 36) DELETE

DELETE절은 FROM키워드를 생략할 수 있다.
EX) DELETE FROM TAB WHERE A = B; → DELETE TAB WHERE A = B;

문제 39) HASH JOIN

HASH JOIN은 정렬작업이 없어 정렬이 부담되는 대량 배치작업에 유리
정렬작업이 있는 JOIN은 SORT MERGE JOIN이다.

문제 40) SELF JOIN 계층쿼리

셀프 조인을 통해 계층형 질의를 작성하여 최상위레벨을 반환하려면 LEFT OUTER JOIN을 걸어야 한다.
(최상위 레벨의 상위는 NULL이므로)

문제 48) 조건절 순서 SALARY > 200 OR MGR_ID IS NULL AND CODE = 'B'

NOT AND OR 순으로 계산된다.
틀린 이유는 SALARY > 200 AND CODE = 'B' OR MGR_ID IS NULL (2개)
CODE = 'B' 그리고 SALARY > 200 OR MGR_ID (4개)로 계산했음.
단순히 괄호만 씌우면 된다.
SALARY > 200 OR (MGR_ID IS NULL AND CODE = 'B')
SALARY > 200 (3개) , MGR_ID IS NULL AND CODE = 'B'(0개)

문제 49) 10개 행 NTILE(4)

기존 테이블 COL1 개수는 10개 VAL컬럼 기준 그룹핑집계 COUNT(*)결과가 각각 3 3 2 2로 되어있다.
10 / 4 몫2 나머지2 로 각 행별 몫 값 만큼 삽입 (2 2 2 2) 후 나머지 2개 데이터를 첫 번째 행부터 순차적으로 삽입

문제 50) LAG(A , N)

LAG(A, N, D)	A컬럼 대상 현재 행 기준 이전 N번째 행 반환 가져올 값이 NULL인 경우 D값 반환	N 생략시 DEFAULT 1 D 생략시 NULL
LEAD(A, N, D)	A컬럼 대상 현재 행 기준 이후 N번째 행 반환 가져올 값이 NULL인 경우 D값 반환	

기출 35회

문제 16) 트랜잭션 아직 COMMIT되지 않은 데이터

ROLLBACK으로 COMMIT한지점까지 데이터 복구 가능
다른 사용자와 나 자신이 볼 수 없다
다른 사용자가 COMMIT되지 않은 변경된 데이터를 고칠 수 없다
→ COMMIT 되지 않았기 때문에 DB상에는 이전 데이터로 존재 즉, 이전 데이터의 값을 고치게 된다.

문제 23) 인덱스의 특징

B-TREE 인덱스는 Equal뿐만아니라 BETWEEN, 비교연산으로도 사용이 가능하다.

문제 26) 데이터 무결성을 보장하기 위한 방법

애플리케이션(개발자 코딩 로직), TRIGGER, 제약조건
(Lock/Unlock은 병행성 제어(동시성) 기법이다.)

문제 37) 다중행 입력 쿼리문

CASE문과 동일하게 수행되며 WHEN을 만족하면 WHNE절은 종료된다.
3의 C >= 3은 이미 C1 >= 2 에서 종료되었다.
따라서 T3에는 아무런 작업이 발생하지 않았으므로 데이터의 수가 0개이다.

문제 40) GROUP BY ROLLUP(A), A

A를 기준으로 그룹핑을 2번했다.
(A), A & ( ), A 로 집계된다.

문제 50)

(1 – 3 – 7) (1 – 4 – 8)
1 – (3 – 7) (4 – 8) 다섯 개가 조회되고 WHERE COL1 <> 4에 의해 COUNT (1, 3, 7, 8) 4건 조회됨

기출 37회

문제 3) 슈퍼/서브타입

ONE TO ONE	테이블을 개별로 구성 다수의 테이블로 조인 많이 발생
PLUS TYPE	슈퍼 / 서브타입 형식으로 처리 두 개의 테이블로 조인 발생
SINGLE TYPE	슈퍼 + 서브 = 하나의 테이블로 통합 조인성능이 좋으나 IO성능이 나쁨

문제 11, 14) 온라인 트랜잭션 처리(OLTP) / NESTED LOOP

적은 테이블 구조를 조인할 때 유리한 **NESTED LOOP**방식이 적합

문제 17) NVL2(A, B, C)

NVL과 DECODE를 하나로 만든 형태로 A가 NULL이 아니면 B NULL이면 C를 반환

문제 22) 윈도우함수 옵션

ROWS BETWEEN 1(위치) PRECENDING AND 1(위치) FOLLOWING
현재 행을 기준으로 현재행을 포함하여 앞의 1건 뒤의 1건의 범위
해설 : 가 MGR 파티션 내에서 날짜를 기준으로 정렬을 수행한 뒤 급여의 평균을 계산

문제 24) PL/SQL

PL/SQL에서 DECLRAE와 BEGIN~END는 필수, EXCEPTION은 선택사항

문제 25) DECODE()

DECODE(A, B, C, D) : A가 B와 같으면 C출력 다르면 D출력 D ARGUMENTS 생략시 NULL출력

문제 29, 30) IN(NULL)

IN연산자에 NULL이 오면 NULL값은 비교에서 제외된다.
29) NOT IN(SELECT COL1 FROM TAB) COL1컬럼에 (2, NULL) 두 행의 값이 존재할 경우 2만 출력
30) WHERE COL1 IN('A', 'B') OR COL1 IS NULL로 수정해야한다.

문제 31) LIKE조건의 ESCAPE옵션

LIKE의 와일드카드는 %와_ 이므로 해당 문자가 포함된 문자열에 LIKE연산 수행시 ESCAPE 옵션 사용
'_ _ A' 문자열을 검색하기 위해서는 컬럼명 **LIKE '%@_%' ESCAPE '@';** 와 같이 ESCAPE '@' 를 사용.
(추측해보면 @뒤의 _를 ESCAPE처리, @대신 주로 역슬래시를 사용하며 다른 특수문자 사용)

문제 35) [오류] WHERE 회원번호 = (SELECT DISTINCT 회원번호 FROM 고객);

고객테이블에는 회원번호가 여러개 존재하기 때문에 동등비교연산을 할 경우 오류가 발생. (IN 사용)

문제 45) 오라클 정렬 NULLS LAST

NULLS LAST : 오라클 ORDER BY 정렬할 때 정렬 기준의 값중 NULL을 가장 마지막으로 정렬

문제 49) NTILE(3) OVER(ORDER BY COL3)

COL3컬럼을 오름차순 정렬하여 전체행을 3등분한다.
(8 / 3 몫 2 나머지 2 → 2 2 2 나머지 2개의 데이터는 첫 번째 행부터 순차적으로 삽입 → 3 3 2)

문제 50) 윈도우함수 옵션 (RANGE / ROWS)

RANGE는 데이터값을 기준으로 앞뒤 데이터의 범위. (ROWS는 행을 기준으로 앞뒤 위치의 범위이다)

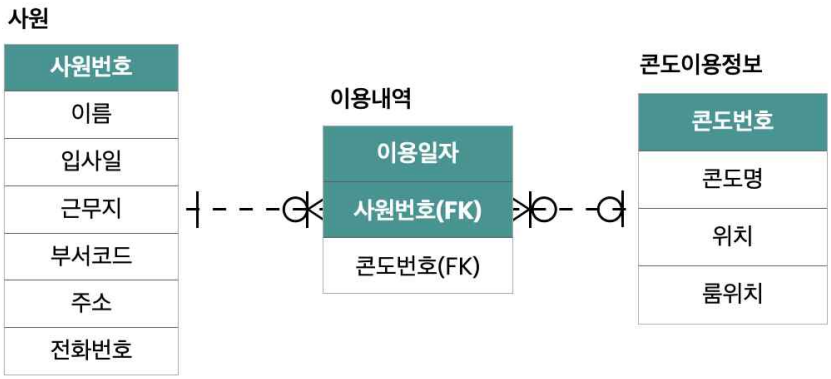
기출 38회

문제 4) 관계 선택사양

| : 필수 (1개)
○ : 선택 (0개 이상)
∈ : 1개 이상

[사원]	1명의 사원은 일자가 다른 여러개의 이용내역을 갖는다. → 이용내역이 존재할수도있고 존재하지않을수도 있다.
[콘도이용정보]	1개의 콘도이용정보는 일자가 다른 여러개의 이용내역을 갖는다. → 이용내역이 있을수도 있고 없을수도 있다.
[이용내역]	1개의 이용내역은 하나의 사원이 존재한다. 1개의 이용내역은 하나의 콘도이용정보가 존재한다. → 콘도이용정보가 없을수도 있다.

[ERD]



문제 20) COUNT(*) / WHERE 1 = 2

집계함수 COUNT(*)는 조건절이 거짓이면 0을 반환한다.

문제 27) ANY

다중행연산자 ANY 하나라도 만족하면 TRUE <>ANY는 일치하지 않는 하나라도 만족하는 행을 구한다.

문제 28) '인덱스' WHERE A – B = C

인덱스 생성에 참여한 컬럼으로 연산을 하면 인덱스가 변형되므로 인덱스를 사용할 수가 없다.

문제 31) 파티션 인덱스

파티션 키에 대한 인덱스 생성 가능 → GLOBAL인덱스

문제 33) ROWNUM

ROWNUM은 논리적인 숫자이다.
사용 가능한 조건은 EQUAL의 경우 1만 가능하고 1을 제외한 나머지는 비교연산만 가능

문제 27) WHERE절 연산자 우선순위

산술 → 연결(||) → 비교 → IS NULL, LIKE, IN → BETWEEN → NOT → AND → OR

문제 45) 테이블 생성과 동시에 테이블 참조 데이터 입력

ORACLE : CREATE TABLE 테이블명 **AS SELECT** * FROM 참조테이블
SQL Server : SELECT * **INTO 생성테이블명** FROM 참조테이블

문제 49) cross join 답 16

1 (233) 3개 / 1 (233) 3개 / 2 (1133) 4개 / 3 (112) 3개 / 3 (112) 3개

기출 39회

문제 1) DISK I/O를 경감할 수 있는 반정규화 방법

'수직분할'은 컬럼단위로 테이블을 분할하여 저장되는 로우의 길이가 줄어들기 때문에 DISK I/O를 경감 개별 테이블의 크기가 작아집니다. 작은 테이블은 더 적은 디스크 공간을 차지하므로 디스크 I/O 용량이 감소합니다. 이로 인해 디스크에서 데이터를 읽고 쓰는 속도가 향상될 수 있습니다.

문제 5) 주식별자와 관련성이 가장 낮은 것은?

제 3 정규화 : 주식별자를 제외한 컬럼간 종속성을 확인해서 종속성이 있으면 분할하는 과정이다.

문제 10) 식별자 중 비즈니스 프로세스에 의해 만들어지는 식별자로 대체여부로 분리되는 식별자는?

대체여부에 따라서 본질식별자와 인조 식별자로 분류
비즈니스 프로세스에 의해 만들어 지는 식별자는 '본질 식별자'이다.

문제 33) 컬럼 변경 DDL문

오라클 : ALTER TABLE 테이블명 'MODIFY' 컬럼명 타입
SQL Server : ALTER TABLE 테이블명 'ALTER COLUMN' 컬럼명 타입

문제 39) PL/SQL에서 데이터베이스 CURSOR를 사용할 때 FETCH 전에 해야하는 것은?

CURSOR OPEN
FETCH(읽어오기) 전에 해야할 것은 CURSOR OPEN
FETCH(읽어오기) 후에 해야할 것은 CURSOR CLOSE
CORSOR순서 : 선언 → OPEN → FETCH → CLOSE

문제 42) HASH JOIN

1. 해시 함수를 이용하여 조인을 수행하기 때문에 '='로 수행하는 조인 동등조건에만 사용가능
2. 두 테이블 중 작은 테이블을 HASH메모리에 로딩하고 두 테이블을 조인키를 사용해서 테이블 생성
3. HASH함수를 사용해서 주소를 계산하고 해당 주소를 사용해서 테이블을 조인하기 때문에 CPU연산이 많이 수행됨
4. HASH조인시에는 선행 테이블의 크기가 작아서 충분히 메모리에 로딩되어야 한다.

[NESTED LOOP]
조인시 RANDOM ACCESS로 인한 부하로 성능지연이 발생할 수 있다.

문제 43) 3차정규화

문제) ROUND, ABS, FLOOR, TRUNC, CEIL

ROUND(7.45, 1)	ABS(7.45)	FLOOR(7.45)	TRUNC(7.45)	CEIL(7.45)
7.5	7.45	7	7	8
소수 1까지 반올림	절대값	정수로 내림	소수점 버림	정수로 올림

문제 46)

TRUNCATE는 AUTOCOMMIT된다.
문제상에서 새로운 테이블을 생성하면서 이전 테이블의 INSERT는 COMMIT되었고, FROM절의 테이블은 이전테이블 이므로 1 + 2 = 3

문제 49) 답 : 9

1 : (1,1) 2개 / 1 : (1,1) 2개 / 2 : (2) 1개 / 3 : (3,3) 2개 / 3 : (3,3) 2개