**70. 데이터베이스 설계**

사용자의 요구를 분석 후 DBMS로 데이터베이스를 구현하여 일반 사용자들이 사용하게 하는 것

2. 데이터베이스 설계 시 고려사항 – 무결성, 일관성, 회복, 보안, 효율성, 데이터베이스 확장

3. 데이터베이스 설계 순서

요구 조건 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현

4. 요구 조건 분석 – 용도 파악, 데이터의 종류/용도/처리 형태/흐름/제약조건 수집, 명세 작성

5. 개념적 설계(정보 모델링, 개념화) – 정보 추상화, 스키마/트랜잭션 모델링, E-RD 작성

6. 논리적 설계(데이터 모델링) – 자료를 자료구조로 변환, 논리적 스키마/인터페이스/테이블 설계

7. 물리적 설계(데이터 구조화)- 물리적 데이터로 변환, 파일 정보 결정, 저장되는 방법 묘사

8. 데이터베이스 구현

스키마 기술→빈 데이터베이스 파일 생성→데이터 입력→트랜잭션 작성→응용 프로그램 작성

**71. 데이터 모델의 개념**

정보들을 단순화, 추상화하여 체계적으로 표현한 개념적 모형, 도구들의 모임, 지능적 도구

2. 데이터 모델의 구성 요소 – 개체(Entity, 정보), 속성(Attribute, 항목), 관계(Relationship)

3. 개념적 데이터 모델 – 정보를 추상적 개념으로 표현, 정보 모델, E-R 모델

4. 논리적 데이터 모델 – 개념적 구조를 데이터로 변환하는 과정, 관계/계층/네트워크 모델

5. 논리적 데이터 모델의 품질 검증 – 기업 적합성 판단, 개체/속성/관계/식별자/모델 리스트 작성

개체/속성/관계/식별자/전반적인 품질 검증 항목

6. 데이터 모델에 표시할 요소 – 구조(Structure), 연산(Operation), 제약 조건(Constraint)

**72. 데이터 모델의 구성 요소 – 개체(Entity)**

정보, 속성으로 구성, 정보 제공, 독립적 존재, 식별자에 의해 식별, 하나이상의 개체와 관계

2. 개체 선정 방법 – 업무 기술서 이용, 담당자 인터뷰, 실제 업무 확인, 장부와 전표 이용

3. 개체명 지정 방법 – 실제 업무 사용 용어 지정, 약어x, 단수 명사 사용, 유일해야함

**73. 데이터 모델의 구성 요소 – 속성(Attribute)**

가장 작은 논리적 단위, 데이터항목/필드에 해당, 개체를 구성하는 항목, 개체의 특성 기술, 차수

2. 속성의 종류 – 기본(Basic)/설계(Designed)/파생(Derived) 속성, 기본키/외래키/일반 속성

3. 속성 후보 선정 원칭 – 후보는 최대한 많이, 원시 속성 버리지 않기, 소그룹 별로 후보군 생성

4. 속성명 지정 원칙 – 정확하고 혼란이 없도록 지정, 서술형/약어 x, 속성명x, 업무 사용 언어

**74. 데이터 모델의 구성 요소 – 관계(Relationship)**

개체와 개체 사이의 논리적인 연결을 의미, 개체 간의 관계와 속성 간의 관계가 있음

2. 관계의 형태 – 1:1, A1-B1/1:N, A1-B1, A1-B2, A2-B3/N:M. A1-B1, A1-B2, A2-B1

3. 관계의 종류

종속(Dependent)/중복(Redundant)/재귀(Recursive)/배타(Exclusive) 관계

식별 관계 - A개체의 기본키 → B개체의 외래키&기본키, 실선 표시 ex)회원의 회원번호→프로필

비 식별 관계 - A개체의 기본키 → B개체의 외래키, 점선 표시

**75. 식별자(Identifier)**

하나의 개체 내에서 각각의 인스턴스를 유일하게 구분할 수 있는 구분자, 모든 개체가 가짐

대표성 여부, 스스로 생성 여부, 단일 속성 여부, 대체 여부

2. 주 식별자/보조 식별자 – 개체를 대표, 한개/대신해 개체 식별, 한개↑

특성 – 유일성(구분), 최소성(속성 수 최소화), 불변성, 존재성(데이터 값)

3. 내부 식별자/외부 식별자 – 개체 내 자체 생성 식별자/외부 개체 식별자 사용, 연결자 역할

4. 단일 식별자/복합 식별자 – 한가지 속성/두 개 이상의 속성

5. 원조 식별자/대리 식별자 – 본질 식별자, 가공x/속성 2개 이상인 경우, 하나의 속성으로 묶음

6. 대리 식별자의 조건 – 범용 값, 내부적으로만 사용 가능, 편의성과 단순성, 의미의 체계화

7. 후보 식별자 – 객체에서 각 인스턴스를 유일하게 식별할 수 있는 속성 또는 속성 집합

Null값 x, 개념적으로 유일, 데이터 변경↓, 각 인스턴스를 유일하게 식별, 속성 직접 식별

**76. E-R(개체-관계) 모델**

현실 정보를 개념적인 논리 데이터로 표현하기 위한 방법, 다이어그램으로 표현

2. E-R 다이어그램(E-RD) – 이해하기 쉽게 기호를 사용하여 시각적으로 표현한 그림, 뷰를 단일화

3. 피터 첸 표기법

4. 정보 공학 표기법(Information Engineering Notation)

5. 바커 표기법(Barker Notation)

**77. 관계형 데이터 모델**

2차원적인 표(Table)를 이용해 데이터 상호 관계를 정의하는 DB 구조, SQL

2. 관계형 테이터 모델의 구성

**78. 관계형 데이터베이스의 구조**

개체나 관계를 릴레이션이라는 표로 표현, 간결하고 보기 편함, 변환 용이, 성능↓

2. 관계형 데이터베이스의 Relation 구조

튜플(Tuple), 속성의 모임/속성(Attribute), 개체의 특성/도메인(Domain), 같은 타입의 값들의 집합

3. 릴레이션의 특징

튜플, 스키마 속성 순서x, 속성값 = 원자값, 튜플 구분을 위한 키, 속성의 명칭은 유일해야함

**79. 관계형 데이터베이스의 제약 조건 – 키(Key)**

데이터의 정확성을 보장하기 위해 키를 이용하여 입력되는 데이터에 제한을 주는 것

1. 키의 개념 및 종류 – Tuple들을 서로 구분할 수 있는 기준이 되는 Attribute(속성)

2. 후보키(Candidate Key) – 튜플을 식별 가능한 기본키로 사용할 수 있는 속성들, 유일성, 최소성

3. 기본키(Primer Key) – Main Key, 중복된 값x, 튜플을 유일하게 구별할 수 있는 속성, NULLx

4. 대체키(Alternate Key) – 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키를 의미

5. 슈퍼키(Super Key) – 속성들의 집합으로 구성된 키, 유일성o, 최소성x

6. 외래키(Foreign Key) – 다른 테이블의 기본키를 참조하는 속성 또는 속성들의 집합

**80. 관계형 데이터베이스의 제약 조건 – 무결성**

데이터 값과 실제 값이 일치하는 정확성을 의미, 무결성 제약 조건 - 부정확한 자료 저장 방지

2. 개체 무결성(Entity Integrity, 실체 무결성) – 기본키를 구성하는 속성은 Null, 중복값 x

3. 도메인 무결성(Domain Integrity, 영역 무결성) – 주어진 속성 값은 정의된 도메인에 속한 값

4. 참조 무결성(Referential Integrity) – 외래키 값은 null or 참조 릴레이션의 기본키 값과 동일

5. 사용자 정의 무결성 – 속성 값들이 사용자가 정의한 제약 조건을 만족

6. 데이터 무결성 강화 – 데이터 특성에 맞는 적절한 무결성을 정의하고 강화

애플리케이션 – 무결성 조건을 검증하는 코드를 데이터를 조작하는 프로그램 내에 추가

장점 : 복잡한 무결성 조건 구현 가능, 단점 : 관리가 힘듬, 적정성 검토의 어려움

데이터베이스 트리거 – 절차형 SQL 추가/통합관리 가능/운영 중 변경 힘듬, 주의 필요

제약 조건 – 제약조건을 설정해 무결성 유지/관리가 쉬움/복잡한 구현과 예외 처리 불가능

**81. 관계 대수 및** **관계 해석**

정보 검색을 유도하는 절차적인 언어, 연산의 순서 명시, 순수 관계 연산자/일반 집합 연산자

2. Seclect

[σ](javascript:answer_check(54,%204)) ㅁ

3. Project

[π](javascript:answer_check(54,%203)) ㅁ

4. Join

[⋈](javascript:answer_check(54,%201))

5. Division

⊃

6. 일반 집합 연산자

7. 관계 해석

**82. 정규화(Normalization)**

관계형 스키마를 바람직한 스키마로 설계해 가는 과정, 데이터의 중복 최소화, 일관성 보장

2. 정규화의 목적

안정성, 무결성 유지, 저장 공간의 최소화, 단순화, 효율성↑, 안정성/무결성 유지, 누락 여부 확인

3. 이상(Anomaly)의 개념 및 종류 – 데이터의 중복으로 인한 이상 현상 발생

삽입 이상(Insertion Anomaly), 삭제 이상(Deletion Anomaly), 갱신 이상(Update Anomaly)

4. 정규화의 원칙 – 정보의 손실x(무손실 표현), 분리의 원칙, 중복성↓

5. 정규화 과정:1NF-원자 값, 2NF-함수적 종속, 3NF-이행적 종속 만족x, BCNF, 4NF-다치 종속, 5NF

두부이걸다줘? →도(메인 원자값)부(분적 함수 종속 제거)이(행적)결(정자)다(치 종속)조(인 종속성)

**83. 반정규화(Denormalization)**

정규화된 데이터 모델을 통합, 중복, 분리하는 과정, 의도적으로 정규화 원칙을 위배하는 행위

2. 테이블 통합 – 잦은 Join으로 인해 하나로 합쳐 사용하는 것이 성능 향상에 도움이 될 경우

3. 테이블 분할 – 수평 분할, 수직 분할(갱신, 조회, 크기), 보안을 적용해야 하는 속성 분할

4. 중복 테이블 추가 – 여러 테이블에서 데이터를 추출해서 사용 or 다른 서버 테이블을 이용

중복 테이블을 추가해야 하는 경우 – 많은/특정 범위의 데이터를 자주 처리, 수행 속도↓

추가 방법 : 집계/진행/특정부분 테이블의 추가, 트리거 설정/기본키 설정/해당부분으로 신규 생성

5. 중복 속성 추가 – 데이터를 조회하는 경로를 단축하기 위해 사용 빈도↑ 속성을 하나 더 추가

추가 경우 – 조인 발생↑, 전근 경로 복잡, 액세스의 조건으로 자주 사용, 기본키의 형태 적절x

고려 사항 – 테이블/속성 중복 고려, 일관성/무결성 유의, SQL로 처리 가능, 저장 공간 낭비 고려

**84. 시스템 카탈로그**

시스템 그 자체에 관련이 있는 다양한 객체에 관한 정보를 포함하는 시스템 베이스

2. 시스템 카탈로그 저장 정보

메타 데이터 : 데이터베이스 객체 정보, 사용자 정보, 테이블의 무결성 제약 조건 정보

3. 카탈로그의 특징

갱신 : 사용자 sql문의 실행 후 변화에 따라 시스템이 자동 갱신, 분산 시스템에서의 카탈로그

4. 카탈로그/데이터 사전을 참조하기 위한 DBMS 내의 모듈 시스템

데이터 정의어 번역기(DDL Compiler) – 메타 데이터를 갖는 테이블로 변환, 데이터 사전에 저장

데이터 조작어 번역기(DML Compiler) – DML → 프로시저 호출 변환, 질의 처리기와 상호 통신

Data Diredctory – 데이터를 실제로 접근하는 데 필요한 정보를 관리 유지하는 시스템

질의 최적화기 – 사용자의 요구를 효율적인 형태로 변환, 질의를 처리

트랜잭션 처리기 – 동시 발생 트랜잭션 문제를 해결, 사용자의 자원 배타적 이용 가능

**85. 사전 조사 분석**

1. 물리 데이터베이스 설계

논리적 데이터베이스를 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정

고려 사항 : 인덱스 구조, 레코드 크기/개수, 스키마 변경 여부, 수행속도↑, 크기 변화 가능성

2. 데이터 명명 규칙 파악 – 물리 데이터 모델에 적용, 업무코드, 리소스 그룹, 거래 코드

3. 시스템 자원 파악 – 데이터 베이스 설치에 영향을 미칠 수 있는 물리적인 요소들

하드웨어 자원(중앙처리장치, 메모리, 디스크, I/O컨트롤러, 네트워크), 운영체제 및 DBMS 버전

4. 데이터베이스 관리 요소 파악 – DB 구조, 이중화 구성, 분산 DB, 접근 제어/통제, 암호화

**86. 데이터베이스 저장 공간 설계**

1. 테이블(Table) – Row(행)과 Coumn(열)로 이루어진 기본 객체, 일반 테이블 – 표준 테이블 형태

3. 클러스터드 인덱스 테이블(Clustered) – 기본/인덱스키의 순서에 따라 데이터 저장, 접근 경로↓

4. 파티셔닝 테이블(Partitioning) – 대용량 데이터 효과적 관리, 범위/해시/조합 분할

5. 외부 테이블 – DB에서 일반 테이블처럼 이용할 수 있는 외부 파일, DB내에 객체로 존재

6. 임시 테이블 – 트랜잭션, 세션별로 데이터를 저장하고 처리할 수 있는 테이블, 종료 시 삭제

7. 컬럼(Colunm) – 테이블의 열을 구성하는 요소, 데이터 타입 : 데이터의 일관성 유지 위해 사용

8. 테이블스페이스(Tablespace) – 테이블이 저장되는 논리적인 영역

**87. 트랜잭션 분석 / CRUD 분석**

DB의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위, 일련의 연산

2. 트랜잭션의 특성 – 원자성, 일관성, 독립/격리/순차성(Isolation), 영속/지속성(Durability)

3. CRUD 분석 – 생성/읽기/갱신/삭제 를 뜻함, CRUD 매트릭스를 작성하여 분석하는 것

4. CRUD 매트릭스 - 2차원 표, 업무 프로세스와 데이터 간 상관 분석표, C>D>U>R 우선순위

5. 트랜잭션 분석 – 트랜잭션 양 분석→데이터의 양 유추→DB용량 산정→DB구조 최적화

6. 트랜잭션 분석서 – 단위 프로세스, CRUD 연산, 테이블/컬럼명, 테이블 참조 횟수, 발생 주기,수

**88. 인덱스 설계**

데이터 레코드를 빠르게 접근하기 위해 <키 값, 포인터> 쌍으로 구성되는 데이터 구조

포인터 = 물리적 주소, 빠른 접근 가능, 삽입삭제↑ 인덱스 개수↓, 인덱스x 테이블 스캔 발생

클러스터드 인덱스 – 키의 순서에 따라 데이터가 정렬, 삽입 삭제 발생 시 데이터 재정렬

2. 트리 기반 인덱스 – 트리 구조를 이루고 있는 인덱스 블록, B/B+ 트리 인덱스

3. 비트맵 인덱스 – 인덱스 컬럼의 데이터를 0 or 1로 변환하여 인덱스 키로 사용

4. 함수 기반 인덱스 – 컬럼에 특정 함수나 수식을 적용하여 산출된 값을 사용

5. 비트맵 조인 인덱스 – 다수의 조인된 객체로 구성된 인덱스, 비트맵 인덱스와 물리 구조 동일

6. 도메인 인덱스 – 개발자가 직접 설계, 확장형 인덱스(Extensible Index)

7. 인덱스 설계 – 대상 선정 → 인덱스 효율성 검토 → 최적화 수행 → 인덱스 정의서 작성

8. 인덱스 대상 테이블 선정 기준 – 순차적 조인 발생, 랜덤 액세스, 특정 범위/순서 데이터 조회

9. 인덱스 대상 컬럼 선정 기준 – 분포도 15%이내(컬럼값 평균 Row수/테이블 총 Row수)x100

부분 처리 목적, 조회 및 출력 조건, 수정↓, 정렬/그룹 지정/통합 지정 빈번 컬럼

분포도가 좁은 컬럼은 단독 인덱스 생성, 인덱스 조합 빈번할 시에 결합 인덱스로 생성

10. 인덱스 설계 시 고려 사항 - 인덱스↑ 오버헤드 발생, 추가 저장 공간, 경로 영향, 공간 분리

**89. 뷰(View) 설계**

접근이 허용된 자료만을 제한적으로 보여주기 위한 가상 테이블, 조인문↓ 편의성 최대화

2. 뷰의 특징 – 논리적 독립성 제공, 관리 용이, 명령문 간단, 데이터 보호

3. 뷰의 장/단점 – 데이터 관리 간편, 자동 보안/독립적 인덱스x, 정의 변경x, I, U, D 제약 따름

4. 뷰 설계 순서 – 테이블 선정 → 컬럼 선정 → 정의서 작성(뷰 명/설명, 테이블, 컬럼, Data타입)

5. 뷰 설계 시 고려 사항

구조 단순화를 위해 반복적으로 조인을 설정하여 사용하거나 동일한 조건절을 사용하는 테이블을 뷰로 생성, 사용 데이터를 다양한 관점에서 제시, 보안 유지를 고려하여 설계

**90. 클러스터 설계**

데이터 저장 시 데이터 액세스 효율을 향상시키기 위해 동일한 성격의 테이블을 동일한 데이터 블록에 저장하는 물리적 저장방법

2. 클러스터의 특징 – 조회 속도↑, IUD성능↓, 분포도↑ 유리+저장 공간 절약, 디스크I/O↓

3. 클러스터 대상 테이블 – 분포도↑, 대량의 범위 자주 조회, IUD 자주 발생x, 자주 조인

**91. 파티션 설계**

대용량의 테이블이나 인덱스를 작은 논리적 단위인 파티션으로 나누는 것

2. 파티션의 장 ∙ 단점 – 쿼리/디스크 성능↑, 속도↑, 데이터 손상 최소화, 가용성↑, 입/출력 분산

세심한 관리 필요, 조인 비용↑, 용량 적은 테이블에 파티셔닝 수행 시 성능 저하

3. 파티션의 종류 – 범위/해시/조합 분할 – 열 값 기준/해시 함수 적용, 고르게 분할/범위+해시

4. 파티션키 선정 시 고려 사항 – 이력성 데이터는 생성/소멸주기 일치, 특정 컬럼 파티션키

5. 인덱스 파티션 – 파티션 된 테이블의 데이터를 관리하기 위해 인덱스를 나눈 것

Local/Global Partitioned Index – 테이블 종속 여부, 테이블, 인덱션 파티션의 1:1 대응/독립적 구성

Prefixed/Non-Prefixed Partitioned Index – 컬럼의 위치, index 파티션키와 첫 번째 컬럼 같음/다름

**92. 데이터베이스 용량 설계**

데이터가 저장될 공간 정의, 테이블에 저장될 데이터/인덱스/클러스터 공간 등을 예측하여 반영

2. 데이터베이스 용량 설계의 목적 – 확장성/가용성/접근성↑, 병목 현상, 데이터 접근성 향상

설계 방법 –테이블 스페이스를 각각 분리하여 구성, extend 발생 최소화, 적합 저장 옵션 지정

3. 데이터베이스 용량 분석 절차 – 기초 자료 수집 후 용량 분석→오브젝트별 용량 산정→테이블, 인덱스의 테이블 스페이스 용량 산정→모든 데이터 용량과 시스템 용량을 합쳐 디스크 용량 산정

**93. 분산 데이터베이스 설계**

논리적으로는 하나의 시스템, 물리적으로는 여러 개의 컴퓨터 사이트에 분산되어 있는 DB

2. 분산 데이터베이스의 구성 요소 – 분산 처리기, 분산 데이터베이스, 통신 네트워크

3. 설계 시 고려 사항 – 노드 별 분산/지역 자치성 보장/일관성 정책, 회복 기능, 원격 접근 기능

4. 목표 – 위치(Locateion)/중복(Replication)/병행(Concurrency)/장애 투명성(Transparency)

5. 장/단점 – 자치성, 공유성, 효용성, 신뢰성↑, 분산 제어, 성능 향상/기능 복잡, 비용, 오류↑

6. 분산 데이터베이스 설계 – 전역 관계망 소규모 단위로 분할 후 결과를 복수의 노드에 할당

7. 테이블 위치 분산 – 테이블을 각기 다른 서버에 분산시켜 배치, 테이블 구조 변경/중복x

8. 분할(Fragment) – 규칙 : 완전성, 재구성, 상호 중첩 배제, 방법 : 수평/수직 분할

9. 할당(Allocation) – 동일한 분할을 여러 서버에 생성하는 분산 방법, 비중복/중복 할당 방식

**94. 데이터베이스 이중화 / 서버 클러스터링**

시스템 오류로 인한 DB 서비스 중단, 물리적 손상 발생 시 복구를 위해 동일 DB를 복제 후 관리

2. DB 이중화의 분류 – Eager : 변경 내용 즉시 적용, Lazy : 새 트랜잭션에 작성 후 각 DB에 전달

3. DB 이중화 구성방법

활동-대기 방법 : 활성 DB에 장애 발생 시 대기 상태의 DB가 자동으로 모든 서비스를 대신 수행

활동-활동 방법 : 2개의 DB가 서로 다른 서비스를 제공, 장애 발생 시 한쪽 DB가 서비스 제공

4. 클러스터링(Clustering) – 두개 이상의 서버를 하나의 서버처럼 운영하는 기술

고가용성 클러스터링 : 하나의 서버에 장애 발생 시 다른 서버가 받아 처리, 일반적.

병렬처리 클러스터링 : 전체 처리율을 높이기 위해 하나의 작업을 여러 서버에서 분산해 처리

**95. 데이터베이스 보안 / 암호화**

권한이 없는 사용자가 액세스하는 것을 금지하기 위해 사용되는 기술

2. 암호화(Encryption) – 수신자 이외 내용x, 암호화 과정(평문→암호문)/복호화 과정(암호문→)

3. 개인키 암호 방식(비밀키 암호 방식) – 동일 키로 암호/복호화, 비밀키-사용자만 나누어가짐

4. 공개키 암호 방식 – 서로 다른 키로 암호/복호화, 공개키-사용자, 비밀키-관리자, RSA

**96. 데이터베이스 보안 – 접근통제**

데이터가 저장된 객체와 이를 사용하려는 주체 사이의 정보 흐름을 제한하는 것

임의 접근통제(DAC) : 객체 생성자 권한 부여, 신원/강제 접근통제(MAC) : 임의의 관리자, 등급

2. 접근통제 정책 – 허용 여부 정의, 신분/규칙/역할 기반 정책 – IBP, GBP/MLP, CBP/주체 역할

3. 접근통제 매커니즘 – 정의된 접근통제 정책을 구현하는 기술적인 방법

접근통제 목록, 능력 리스트(권한 목록), 보안 등급(보안 속성의 집합), 패스워드, 암호화

4. 접근통제 보안 모델 – 기밀성 모델/무결성(정보변경 방지)/접근통제 모델(행, 사용자/열, 객체)

5. 접근통제 조건 – 값 종속/다중 사용자/컨텍스트 기반(조건에 따라 접근 제어) 통제

6. 감사 추적 – DB에 접근하여 수행한 모든 활동을 기록하는 기능(DB복구, 데이터 조작 파악)

**97. 데이터베이스 백업**

DB에 저장된 데이터를 보호하고 복구하기 위한 작업, 치명적인 데이터 손실 방지

2. DB 장애 유형 – 사용자 실수, 미디어/구문/사용자 프로세스/인스턴스 장애

3. 로그 파일 – 상태 변화를 시간의 흐름에 따라 모두 기록한 파일/트랜잭션 내용, 식별, 레코드

4. 데이터베이스 복구 알고리즘

동기적 갱신 : 트랜잭션 완료 전 DB 버퍼 내용을 동시적으로 저장매체에 기록

비동기적 갱신 : 트랜잭션이 완료된 내용을 일정 주기나 작업량에 따라 시간 차이를 두고 기록

UNDO : 기록 내용 취소, REDO : 트랜잭션 내용을 다시 실행함

NO-UNDO/REDO : 비동기적 갱신, UNDO/NO-REDO : 동기적 갱신

UNDO/REDO : 동기/비동기적 갱신, NO-UNDO/NO-REDO : DB 버퍼의 내용을 타 저장매체에 저장

5. 백업 종류

물리 백업 : DB파일을 백업, 속도↑ 작업 단순, 문제 발생 시 원인 파악 및 문제 해결이 어려움

논리 백업 : DB 내의 논리적 객체들을 백업, 데이터 손상을 막고 문제 발생 시 원인 파악 및 해결이 수월, 백업/복원 시 시간이 많이 소요

**98. 스토리지**

단일 디스크로 처리할 수 없는 대용량의 데이터를 저장하기 위해 서버와 저장장치를 연결하는 것

2. DAS(Direct Attached Storage) – 서버와 저장장치를 직접 연결 ex)컴퓨터-외장하드

속도↑, 유지보수 비용↓, 설치 및 운영 쉬움, 타 서버 접근 및 파일 공유x, 확장/유연성↓

3. NAS(Network Attached Storage) – 네트워크를 통해 서버와 저장장치를 연결, 접속↑성능↓

4. SAN(Storage Area Network) – DAS의 빠른 처리, NAS 공유 장점 혼합, 전용 네트워크 별도 구성

파이버 채널 스위치, 확장/유연/가용성↑, 비용↑, 높은 트랜잭션의 처리에 효과적

**99. 논리 데이터 모델의 물리 데이터 모델 변환**

1. 테이블(Table) – DB의 기본 오브젝트, 로우(튜플)/컬럼/기본키/외래키

2. 엔티티(Entity)를 테이블로 변환 – 변환 후 테이블 목록 정의서 작성

변환 규칙 : 엔티티-테이블, 속성-컬럼, 주 식별자-기본키, 외부 식별자-외래키

고려사항 : 테이블 명칭 = 엔티티 명칭, 가독성을 위해 테이블 명 영문, 표준 용어 존재 시 사용

3. 슈퍼타입/서브타입을 테이블로 변환

슈퍼타입 기준 테이블 변환 – 서브타입을 슈퍼타입에 통합

장점 : 데이터 액세스 용이, 수행속도 빨라짐, 문장 구성 단순, 임의 집합에 대한 처리 용이

단점 : 컬럼 증가로 인한 저장 공간 증가, 구분 필요 경우 다수 발생, 인덱스 효율 감소

서브타입 기준 테이블 변환 – 슈퍼타입 속성들을 각각의 서브타입에 추가 후 개별 테이블 작성

장점 : 유형 구분 필요x, 테이블당 크기 감소로 전체 테이블 스캔 시 유리

단점 : 수행 속도 감소 우려, 복잡한/부분 범위 처리 힘듬, 통합 뷰는 조회만 가능, UID 관리 힘듬

개별타입 기준 테이블 변환 – 슈퍼/서브타입들을 각각의 개별적인 테이블로 변환, 1:1관계

적용 경우 : 서브타입의 컬럼 수↑, 트랜잭션이 주로 슈퍼타입에서 발생, 서브타입의 처리 독립적

저장공간↓, 개별 정보 조회시 문장 작성 용이, 슈퍼/서브타입 정보 동시 처리시 성능 저하

4. 속성을 컬럼으로 변환 – 표준 약어 사용, 예약어 사용 회피, 가능한 짧게 지정, 샘플데이터작성

5. 관계를 외래키로 변환

1:1 관계 : 개체 A의 기본키를 개체 B의 외래키로 추가하거나 반대로도 추가하여 표현

1:M 관계 : 개체 A의 기본키를 개체 B의 외래키로 추가하여 표현 혹은 별도의 테이블로 표현

N:M 관계 ; A와 B의 기본키를 모두 포함한 별로의 릴레이션으로 표현

1:M 순환 관계 : 개체 A에 개체 A의 기본기를 참조하는 외래키 컬럼을 추가하여 표현

6. 관리 목적의 테이블/컬럼 추가 – DB 관리 혹은 프로그래밍의 수행 속도 향상을 위해 추가

7. 데이터 타입 선택 – 최적의 데이터타입 및 길이 선택, CHAR, VARCHAR2, NUMBER, DATE

**100. 물리 데이터 모델 품질 검토**

물리 데이터 모델 설계, DB 객체 생성 후 개발 단계로 넘어가기 전에 관리자들이 모여 수행

2. 물리 데이터 모델 품질 기준 – 정확성/완전성/준거성/최신성/일관성/활용성

3. 물리 데이터 모델 품질 검토 항목 – 품질 기준 작성 후 이를 기반하여 작성

물리 데이터 모델의 전반적인 것을 검토 항목으로 작성 ex) 명명, 설명, 정의, 권한

4. 물리 데이터 모델의 품질 검토 순서

데이터 품질 정책 및 기준 확인 → 품질 기준 작성 → 체크리스트 작성 → 논리/물리 데이터 모델 비교 → 모델러와 이해관계자가 품질 검토 수행 → 품질 검토 보고서 작성

**101. SQL의 개념**

국제 표준 DB 언어, 관계대수와 관계해석을 기초로 한 혼합 데이터 언어, 질의어

2. SQL의 분류 – DDL, DML, DCL

**102. DDL(Data Define Language, 데이터 정의어)**

DB를 구축하거나 수정할 목적으로 사용하는 언어, 번역 결과는 데이터 사전에 테이블로 저장

2. CREATE SCHEMA – CREATE SCHEMA 스키마명 AUTHORIZATION 사용자\_ID;

3. CREATE DOMAIN – CREATE DOMAIN 도메인명 [AS] 데이터\_타입 [DEFAULT 기본값]

[CONSTRAINT 제악조건명 CHECK (범위값)]; 속성값의 범위가 일부분일 때 사용

4. CREATE TABLE – 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 UPDATE옵션]

[, CONSTRAINT 제약조건명] [CHECK (조건식)];

5. CREATE VIEW – CREATE VIEW 뷰명[(속성명[, 속성명, …])] AS SELECT문;

6. CREATE INDEX – CREATE [UNIQUE] INDEX 인덱스명 ON 테이블명(속성명 [ASC | DESC])

[, 속성명 [ASC \ DESC]]) [CLUSTER]; / ASC-오름차순, DESC-내림차순

7. ALTER TABLE – 테이블에 대한 정의를 변경하는 명령문

ALTER TABLE 테이블명 ADD 속성명 데이터\_타입 [DAFAULT ‘기본값’];

ALTER TABLE 테이블명 ALTER 속성명 [SET DEFAULT ‘기본값’];

ALTER TABLE 테이블명 DROP COLUMN 속성명 [CASCADE];

8. DROP – 스키마, 도메인, 기본/뷰 테이블, 인덱스, 제약 조건 등을 제거하는 명령문

DROP SCHEMA/DOMAIN/TABLE/VIEW/INDEX 이름 [CASCADE | RESTRICTED]; / DROP CONSTRAINT

CASCADE : 제거할 요소를 참조하는 다른 모든 개체를 함께 제거

RESTRICTED : 다른 개체가 제거할 요소를 참조중일 때는 제거를 취소한다.

**103. DCL(Data Control Language, 데이터 제어어)**

데이터의 보안, 무결성, 회복, 병행 제어 등을 정의하는 데 사용하는 언어

2. GRANT / REVOKE – 권한 부여를 위한 명령어 / 권한 취소를 위한 명령어

사용자등급 지정 및 해제

GRANT 사용자등급 TO 사용자\_ID\_리스트 [IDENTIFIED BY 암호];

REVOKE 사용자등급 FROM 사용자\_ID\_리스트;

테이블 및 속성에 대한 권한 부여 및 취소

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, ALTER 등

WITH GRANT OPTION/FOR : 받은 권한을 다른 사용자에게 부여할 수 있는 권한을 부여/취소

3. COMMIT – 트랜잭션 성공 시 변경된 모든 내용을 유에 반영할 때 사용하는 명령어

4. ROLLBACK – 아직 COMMIT되지 않은 변경된 모든 내용들을 취소하고 DB를 이전상태로 되돌림

5. SAVEPOINT – 트랜잭션 내에 ROLLBACK 할 위치인 저장점을 지정하는 명령어, 이름을 부여

**104. DML(Data Manipulation Language, 데이터 조작어)**

DB 사용자가 응용 프로그램이나 질의어를 통해 저장된 데이터를 관리하는데 사용되는 언어

2. 삽입문(INSERT INTO~)

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

3. 삭제문(DELETE FROM~) – DELETE FROM 테이블명 [WHERE 조건]; 테이블의 구조는 남아있음

4. 갱신문(UPDATE~SET~)

UPDATE 테이블명 SET 속성명 = 데이터[, 속성명=데이터, …][WHERE 조건];

**105. DML-SELECT-1**

SELECCT [PREDICATE] [테이블명.]속성명 [AS별칭][.[테이블명.]속성명, …] [, 그룹함수(속성명 [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 : 불러올 튜플 수를 제한할 명령어를 기술, ALL/DISTINCT/DISTINCTROW

**106. DML-SELECT-2**

그룹함수 : GROUP BY절에 지정된 그룹별로 속성의 값을 집계할 함수를 기술한다. / SUM, AVG 등

WINDOW 함수 : GROUP BY절을 이용하지 않고 속성의 값을 집계할 함수를 기술한다.

-PARTITION BY : WINDOW 함수가 적용될 범위로 사용할 속성 지정

-ORDER BY : PARTITION 안에서 정렬 기준으로 사용할 속성 지정

GROUP BY절 : 특정 속성을 기준으로 그룹화하여 검색할 때 사용, 보통 그룹함수와 함께 쓰임

HAVING절 : GROUP BY와 함께 사용되며, 그룹에 대한 조건을 지정한다.

4. 집합 연산자를 이용한 통합 질의

FROM 테이블명 UNION | UNION ALL | INTERSECT | EXCEPT SELECT FROM 테이블명 …

UNION/UNION ALL : 두 SELECT문 조회 결과 통합하여 모두 출력, 중복행 1번 출력/그대로 출력

INTERSECT : 두 SELECT문의 조회 결과 중 공통된 행만 출력한다.

EXCEPT : 첫 번째 SELECT문의 조회 결과에서 두 번째 SELECT문의 조회 결과를 제외한 행 출력

**107. DML-JOIN**

2개의 테이블에 대해 연관된 튜플들을 결합하여, 하나의 새로운 릴레이션을 반환

2. INNER JOIN

EQUI JOIN – 대상 테이블에서 공통 속성을 기준으로 ‘=’ 비교에 의해 같은 값을 가지는 행을 연결하여 결과를 생성하는 JOIN 방법, 같은 속성을 한번만 표시할 시 NATURAL JOIN

Ex) WHERE 테이블명1.속성명 = 테이블명2.속성명; / FROM 테이블명1 NATURAL JOIN 테이블명2;

FROM 테이블명1 JOIN 테이블명2 USING(속성 명); / NON EQUI JOIN – 비교 연산자 사용 JOIN

3. OUTER JOIN – JOIN 조건을 만족하지 않는 튜플도 결과로 출력하기 위한 JOIN 방법

LEFT/RIGHT OUTER JOIN – FROM 테이블명 1 LEFT/RIGHT OUTER JOIN 테이블명2 ON …

INNER JOIN의 결과를 구한 후 우측/좌측 항 릴레이션의 맞지않는 튜플에 NULL값 부여 후 추가

FULL OUTER JOIN – 둘을 합쳐놓음. 양쪽 맞지않는 튜플에 NULL값 부여 후 추가

4. SELF JOIN – 같은 테이블에서 2개의 속성을 연결하여 EQUI JOIN을 하는 방법

SELECT [별칭1].속성명, [별칭1.]속성명…FROM 테이블명1[AS] 별칭1 JOIN 테이블명1 [AS]별칭2 ON..

**108. 프로시저(Procedure)**

절차형 SQL을 활용하여 특정 기능을 수행하는 일종의 트랜잭션 언어, 미리 저장한 작업 수행

DECLARE : 프로시저의 명칭, 변수, 인수, 데이터 타입을 정의하는 선언부이다.

BEGIN / END : 시작과 종료, CONTROL : 조건문 or 반복문이 삽입되어 순차적으로 처리

SQL : DML, DCL이 삽입되어 데이터 관리를 위한 조회, 추가, 수정, 삭제 작업을 수행

EXCEPTION : BEGIN ~ END 안의 구문 실행 시 예외가 발생하면 이를 처리하는 방법을 정의

TRANSACTION : 수행된 데이터 작업을 DB에 적용할지 취소할지를 결정하는 처리부

2. 프로시저 생성

CREATE [OR REPLACE] PROCEDURE 프로시저명(파라미터) [지역변수 선언] BEGIN BODY; END;

OR REPLACE : 선택적인 예약어, 동일한 프로시저 이름이 이미 존재하는 경우 기존 프로시저 대체

파라미터 : IN/OUT/INOUT : 호출 프로그램이 프로시저에게 값을 전달/반환/전달, 실행 후 반환

매개변수명 : 호출 프로그램으로부터 전달받은 값을 저장할 변수의 이름을 지정

프로시저 BODY : 코드 기록 부분, BEGIN에서 시작 END에서 종료, 사이에는 하나이상의 SQL 존재

3. 프로시저 실행 – EXECUTE(EXEC), CALL사용, EXECUTE/EXEC/CALL 프로시저 명;

4. 프로시저 제거 – DROP PROCEDURE 프로시저명;

**109. 트리거(Trigger)**

데이터의 삽입, 갱신, 삭제 등의 이벤트가 발생할 때 관련 작업이 자동으로 수행되는 절차형SQL

2. 트리거의 구성

DECLARE→EVENT(트리거가 실행되는 조건을 명시)→BEGIN→CONTROL→SQL→EXCEPTION→END

3. 트리거의 생성

CREAT [OR REPLACE TRIGGER] 트리거명 [동작시기 옵션][동작 옵션] ON 테이블명

REFFERENCING [NEW | OLD] AS 테이블명 FOR EACH ROW [WHEN 조건식] BEGIN BODY; END;

동작시기 옵션 : 트리거가 실행될 때를 지정, AFTER/BEFORE – 테이블이 변경된 후/되기 전 실행

동작 옵션 : 트리거가 실행되게 할 작업의 종류를 지정

INSERT/DELETE/UPDATE : 새로운 튜플을 삽입/삭제/수정할 때 트리거가 실행

NEW/OLD : 추가되거나 수정에 참여할/수정되거나 삭제전 대상이되는 튜플들의 집합을 의미

FOR EACH ROW : 각 튜플마다 트리거를 적용한다는 의미

WHEN 조건식 : 선택적인(Optional)예약어이다. 트리거를 적용할 튜플의 조건을 지정한다.

**110. 사용자 정의 함수**

SQL을 사용하여 일련의 작업을 연속적으로 처리, 종료 시 처리 결과를 단일값으로 반환

2, 사용자 정의 함수의 구성도

DECLARE→BEGIN→CONTROL→SQL→EXCEPTION→RETURN(필, 반환할 값, 변수 정의)→END

3. 사용자 정의 함수 생성 - CREATE [OR REPLACE] FUNCTION 사용자 정의 함수명(파라미터) [지역변수 선언] BIGIN 사용자 정의 함수 BODY; RETURN 반환값; END;

파라미터 : IN(값 전달), 매개변수명, 자료형 / RETURN : 반환 값 or 변수 호출 프로그램으로 반환

4. 사용자 정의 함수 실행

SELECT 함수 FROM 테이블명 / INSERT INTO 테이블명(속성명) VALUES 함수명;

DELETE FROM 테이블명 WHERE 속성명 = 함수명; / UPDATE 테이블명 SET 속성명 = 함수명;

**111. DBMS 접속 기술**

1. DBMS 접속 – 사용자가 데이터를 사용하기 위해 응용 시스템을 이용하여 DBMS에 접근하는 것

2. DBMS 접속 기술 – DBMS에 접근하기 위해 사용하는 API 또는 사용을 도와주는 프레임워크

JDBC(Java DataBase Connectivity) - DB접속 및 SQL문 수행 시 사용되는 표준 API, SUN MS개발

ODBC(Open DataBase Connectivity) – 표준 개방형 API, MS개발, DBMS의 종류를 몰라도 이용가능

My Batis – SQL Mapping기반 프레임워크, 접속 기능 강화, SQL 문장을 분리해 XML 파일 생성

3. 동적 SQL(Dynamic SQL) – 개발 언어에 삽입되는 SQL 코드를 문자열 변수에 넣어 처리

사용자가 응용 프로그램을 실행시킨 후 입력란에 SQL을 직접 입력하여 결과를 확인할 수 있음

**112. SQL 테스트**

SQL이 작성 의도에 맞게 원하는 기능을 수행하는지 검증하는 과정

2. 단문 SQL 테스트 DDL, DML, DCL이 포함되어 있는 SQL과 TCL을 테스트하는 것, 직접 실행

DESCRIBE명령어 사용 시 테이블, 뷰의 속성, 자료형, 옵션 확인 가능 / DESC [개체명];

사용자 권한 정보 : 정보가 저장된 테이블을 SELECT로 조회 또는 SHOW 명령어로 확인 가능

3. 절차형 SQL 테스트 – 디버깅을 통해 기능의 적합성 여부 검증, 실행으로 결과 학인

명령어를 통해 오류 내용 확인 및 문제 수정, SHOW ERRORS; 주석 #(MySQL),--(Oracle), /\*\*/

DBMS\_OUTPUT.ENABLE; : 화면에 출력하기 위해 DBMS\_OUTPUT 패키지를 불러온다.

DBMS\_OUTPUT.PUT\_LINE(데이터); / SELECT 데이터; : 데이터에 넣은 변수나 값을 화면에 출력

**113. ORM(Object-Relational Mapping)**

객체지향 프로그래밍의 객체와 관계형 DB의 데이터를 연결(Mapping)하는 기술을 의미

가상DB를 만들어 코드와 데이터 연결, 코드 또는 DB와 독립적이어서 재사용, 유지보수 용이

SQL 코드를 직접 입력x, 부수 코드가 생략되어 직관적이고 간단하게 데이터 조작이 가능

2. ORM 프레임 워크 – ORM을 구현하기 위한 구조와 구현을 위해 필요한 기능들을 제공하는 SW

3. ORM의 한계

프레임워크의 자동 SQL이 의도대로 작성되었는지 확인 필요, 객체지향적인 사용을 고려하고 설계되지 않을 경우 프로젝트가 크고 복잡해질수록 ORM 기술 적용이 어려움

기존 기업들은 ORM 을 고려하지 않은 DB를 사용해 적합하게 변환하려면 많은 시간과 노력 필요

**114. 쿼리 성능 최적화**

데이터 입∙출력 App의 성능 향상을 위해 SQL 코드를 최적화

성능 측정 도구인 APM을 사용하여 최적화할 쿼리 선정 → 옵티마이저가 수립한 실행 계획 검토 → SQL 코드, 인덱스 재구성

옵티마이저(Optimizer) – SQL이 가장 효율적으로 수행되도록 최적의 경로를 찾아 주는 모듈

사용자가 쿼리 실행 시 SQL문을 파싱하여 실행 계획 작성, 계획에 따라 데이터를 조작한다.

2. 실행 계획(Execution Plan) – DBMS의 옵티마이저가 수립한 SQL 코드의 실행 절차와 방법

EXPLAIN 명령어를 통해 확인 가능, 그래픽이나 텍스트로 표현, 연산(조인, 필터 등) 순서 기입

3. 쿼리 성능 최적화 – 실행 계획을 참고하여 SQL 코드와 인덱스를 재구성하는 것, 속도, 효율↑

SQL코드 재구성 – WHERE 절을 추가, 일부 레코드만 조회해 비용↓, 연산자 사용 자제

인덱스 재구성 – SQL 코드에서 조회되는 속성과 조건들을 고려하여 인덱스를 구성

인덱스 추가or기존 인덱스 열 순서 변경, 신중한 결정, 단인 인덱스/읽기 테이블 IOT 구성 고려

**115. 데이터 전환**

운영 중인 기존 정보 시스템에 축척되어 있는 데이터를 추출(Extraction)하여 새로 개발할 정보 시스템에서 운영 가능하도록 변환(Transformation)한 후, 적재(Loading)하는 일련의 과정

2. 데이터 전환 계획서 – 데이터 전환 개요, 데이터 전환 대상 및 범위, 데이터 전환 환경 구성

데이터 전환 조직 및 역할, 데이터 전환 일정, 전환 방안, 정비방안, 비상 계획, 데이터 복구 대책

**116. 데이터 전환 계획서 작성**

1. 데이터 전환 개요 – 데이터 전환 목표/주요 요인/작업을 위한 전제 조건 및 제약 사항을 기술

2. 데이터 전환 대상 및 범위 – 단위 업무별로 데이터 전환 대상 정보, 테이블의 수, 데이터 크기

3. 데이터 전환 환경 구성 – 원천 시스템/목적 시스템의 구성도, 전환 단계 별 DISK 사용량 기술

4. 데이터 전환 조직 및 역할 작성 – 데이터 전환 수행, 결과 검증할 작업자와 역할 상세 기술

5. 데이터 전환 일정 작성 – 데이터 전환 및 검증 작업별 상세하게 일정을 수립하여 작성, 도식화

**117. 데이터 전환 방안**

2. 데이터 전환 규칙 – 본 전환 시 동시 수행 프로세스 수, DDL 수행 성능 강화, DW 초기 적재 시 분산 처리 성능 강화, 선 전환 시 운영 서버 부하 최소화, 운영 서버 여유 공간

3. 데이터 전환 절차 – 데이터 전환 절차 체계적이고 상세하게 기술, 데이터 흐름도 작성

전환 준비 → 전환 설계/개발 → 전환 테스트 → 실데이터 전환 → 최종 전환 및 검증

4. 데이터 전환 방법 – 단위 업무별로 데이터 전환 방법/업무별 요구 전제 조건 기술

5. 데이터 전환 설계 – 업무별 전환 대상/제외 대상 기술, 원천/목적 시스템 테이블 매핑 정의서

6. 전환 프로그램 개발 및 테스트 계획 – 체크리스트 기반 테스트 실시, 전환 시나리오 반영

7. 데이터 전환 계획 – 선/본/후 전환 분리 후 계획 수립, 내용 기술, 전환 시나리오 작성

8. 데이터 검증 방안 – 전환 데이터 정합성 검증, 문제 대응을 위해 단계별 검증 방안 수립

**118. 데이터 검증**

원천 시스템의 데이터를 목적 시스템의 데이터로 전환하는 과정이 정상적으로 수행됐는지 확인

2. 검증 방법에 따른 분류 – 로그/기본 항목/응용 프로그램/응용 데이터/값 검증

3. 검증 단계에 따른 분류 – 추출→전환→DB적재→적재 후 정합성 확인→추가 검증(응용 검증)

**119. 오류 데이터 측정 및 정제**

고품질의 데이터를 운영 및 관리하기 위해 수행, 품질 분석→오류 데이터 측정→오류 데이터 정제

2. 데이터 품질 붐석 – 오류 데이터를 찾기 위해 원천, 목적 시스템 데이터의 정합성 여부 확인

3. 오류 데이터 측정 – 데이터 품질 분석 기반, 정산/오류 데이터 구분 후 수량 측정 및 기재

4. 오류 데이터 정제 – 오류 관리 목록의 각 항목 분석, 원천 데이터 정제, 전환 프로그램 수정

**120. 데이터 정제요청서 및 정제보고서**

원천 데이터의 정제와 전환 프로그램의 수정을 위해 데이터 정제와 관련된 내용을 문서로 작성

2. 데이터 정제 요건 목록 작성 – 오류 관리 목록의 각 항목에 대해 정제 유형 분류, 상태 정의

정제 유형 : 완전성, 유효성, 일치성, 유일성/정제 방법 : 원천, 전환, 모두/상태 : 진행 상태

3. 데이터 정제 요청서 작성 – 데이터 전환 시 발생한 오류의 수정을 위한 내용, 해결방안 포함

4. 데이터 정제보고서의 개요 및 작성 – 요청서를 통해 정제된 데이터 확인 결과 작성

육안으로 직접 비교하여 확인, 정제 ID별로 작성, 데이터 전환 프로그램을 이용하여 전환한 후 오류 발생 여부 확인→목적DB에서 전환된 데이터 확인