1과목 데이터 모델링의 이해

1장 데이터 모델링의 이해

1절 데이터 모델의 이해

1. 모델링의 정의 : 복잡한 현실세계를 일정한 표기법에 의해 표현하는 일
   ① 모델링 특징
      - 추상화 : 현실세계, 다양한 현상 일정한 양식인 표기법에 의해 표현(=모형화, 가설적)
      - 단순화 : 복잡한 현실세계를 약속된 규약에 의해 제하된 표기법이나 언어로 표현
      - 명확화 : 누구나 이해하기 쉽게 대상에 대한 애매모호함 제거
   ② 모델링의 세 가지 관점
      - 데이터 관점(What, Data) : 업무가 어떤 데이터와 관련 있는지, 데이터 간 관계는 무엇인지
      - 프로세스 관점(How, Process) : 업무가 실제 하고 있는 일이 무엇인지, 무엇을 해야 하는지
      - 상관 관점(Interaction) : 업무 처리하는 방법에 따라 데이터가 어떻게 영향을 받고 있는지
2. 데이터 모델링 정의
   - 정보시스템을 구축하기 위한 데이터 관점의 업무 분석 기법
   - 현실세계의 데이터에 대해 약속된 표기법에 의해 표현되는 과정
   - 데이터베이스를 구축하기 위한 분석/설계의 과정
   ① 기능 : 명세화, 구조화, 문서화, 다양한 관점, 구체화
   ② 중요성 : 파급효과 큼(Leverage), 복잡한 정보 요구사항의 간결한 표현(Conciseness), 데이터 품질(Data Quality)
   ③ 유의점
      - 중복 : 같은 시간 같은 데이터 제공
      - 비유연성 : 사소한 업무변화에 데이터 모델이 수시로 변경되면 안됨. 데이터 정의를 사용 프로세스와 분리
      - 비일관성 : 데이터 간 상호 연관 관계에 대해 명확히 정의
   ④ 3단계 진행 : 개념 데이터 모델링(추상적) → 논리 데이터 모델링 → 물리 데이터 모델링(구체적)
      ▷ 프로젝트 생명주기(Life Cycle) : 정보전략계획 → 분석 → 설계 → 개발 → 테스트 → 전환/이행 단계
      - 개념적 데이터 모델링(계획/분석단계) : 추상화, 업무중심적, 포괄적, 전사적, EA 수립시 사용
      - 논리적 데이터 모델링(분석단계) : KEY, 속성, 관계 표현, 재사용성 높음(정규화)
      - 물리적 데이터 모델링(설계단계) : 실제 데이터베이스에 이식할 수 있도록 성능, 저장 등 물리적 성격 고려
3. ANSI/SPARC 데이터 독립성 모델
   ① 데이터 독립성의 필요성 : 유지보수 비용 증가, 데이터 중복성, 복잡성 증가, 요구사항 대응 저하
   ② 3단계 구조
      - 외부 스키마(사용자 관점) : 개개 사용자나 응용프로그래머가 접하는 DB
      - 개념 스키마(통합 관점) : DB에 저장되는 데이터와 그들간의 관계를 표현하는 스키마
      - 내부 스키마(물리적 관점) : 물리적 장치에서 데이터가 실제적으로 저장되는 방법을 표현하는 스키마
   ③ 2가지 독립성
      - 논리적 독립성 : 개념 스키마 변경 → 외부 스키마 영향X - 논리적 구조 변경 → 응용 프로그램 영향X
      - 물리적 독립성 : 내부 스키마 변경 → 외부/개념 스키마 영향X - 저장 장치 구조 변경 → 응용 프로그램 & 개념 스키마 영향X
   ④ 사상(Mapping) 2가지
      - 외부적/개념적 사상(논리적 사상) : 사용자가 접근하는 형식에 따라 다른 타입의 필드 가질 수 있음
      - 개념적/내부적 사상(물리적 사상) : 저장된 DB구조 바뀌면 개념적/내부적 사상이 바뀌어야 개념 스키마 남음
4. 데이터 모델링의 세 가지 요소
   ① 어떤 것(Things) : 엔터티(Entity), 인스턴스(Instance)
   ② 관계(Relationships) : 관계(Relationship), 패어링(Pairing)
   ③ 성격(Attributes) : 속성(Attribute), 속성값(Attribute Value)

5. 좋은 데이터 모델의 요소
① 완전성 : 업무에 필요한 모든 데이터가 모델에 정의
② 중복배제 : 하나의 DB 내에 동일한 사실은 한 번만 저장
③ 업무규칙 : 많은 규칙을 사용자가 공유하도록 제공
④ 데이터 재사용 : 데이터 통합성과 독립성 고려
⑤ 의사소통 : 업무규칙은 엔터티, 서브타입, 속성, 관계 등의 형태로 최대한 자세히 표현
⑥ 통합성 : 동일한 데이터는 유일하게 정의해서 다른 영역에서 참조하도록 함

2절 엔터티

1. 개념
- 사람, 장소, 물건, 사건, 개념 등의 명사
- 업무상 관리가 필요한 관심사
- 저장이 되기 위한 어떤 것(Thing)

2. 특징
- 업무에서 필요한 정보
- 식별 가능 : 인스턴스 각각을 구분하기 위한 유일한 식별자 필요
- 인스턴스의 집합 : 하나의 엔터티는 2개 이상의 인스턴스 포함
- 업무 프로세스에 의해 이용
- 속성 포함
- 관계 존재

3. 분류
① 유무형에 따른 분류
- 유형 : 물리적인 형태, 안정적 (사원, 물품, 강사)
- 개념 : 물리적인 형태 존재하지 않는 개념적 정보 (조직, 보험상품)
- 사건 : 업무 수행에 따라 발생, 발생량 많으며 통계자료에 이용 (주문, 청구, 미납)

② 발생시점에 따른 분류
- 기본 : 원래 존재하는 정보, 독립적으로 생성 가능 (사원, 부서, 고객)
- 중심 : 기본 엔터티로부터 발생되고 업무에 중심적 역할 (계약, 접수)
- 행위 : 두 개 이상의 부모엔터티로부터 발생, 자주 바뀌거나 데이터 증가 (주문내역, 계약진행)

4. 명명 : 현업에서 사용하는 용어, 약어X, 단수 명사, 유일한 이름, 생성 의미에 따른 이름

3절 속성

1. 개념 : 업무에서 필요, 의미상 더 이상 분리되지 않음, 엔터티를 설명하는 인스턴스의 구성 요소

2. 엔터티-인스턴스-속성-속성값 관계
- 엔터티는 두 개 이상의 인스턴스의 집합
- 엔터티는 두 개 이상의 속성을 가짐
- 속성은 한 개의 속성값 가짐

3. 분류
① 특성에 따른 분류
- 기본속성 : 업무분석을 통해 바로 정의 (제품번호, 제품명, 제조원가)
- 설계속성 : 업무상으로 존재하지 않지만 설계하면서 도출하는 속성 (제조사코드)
- 파생속성 : 다른 속성으로부터 계산/변형되어 생성 (판매금액)

② 엔터티 구성방식에 따른 분류
- PK(Primary Key) : 엔터티 식별 속성 (부서-부서번호 / 사원-사원번호)
- FK(Foreign Key) : 다른 엔터티와의 관계에서 포함된 속성 (사원-부서번호)
- 일반속성 : 엔터티에 포함되었지만 PK, FK 아닌 속성 (부서-부서명 / 사원-사원명, 전화번호)

4. 도메인 : 각 속성이 가질 수 있는 값의 범위

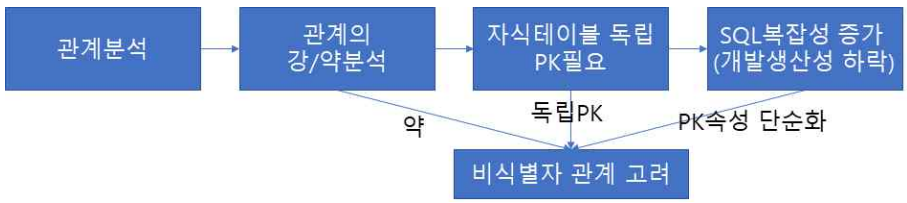
5. 속성의 명명 : 업무에서 사용하는 이름, 서술식x, 약어 사용 제한, 유일성 확보

4절 관계

1. 정의 : 상호 연관성이 있는 상태, 존재의 ㅎ여태로서나 행위로서 서로에게 연관성이 부여된 상태
2. 패어링
   - 엔터티 안에 인스턴스가 개별적으로 관계를 가지는 것
   - 개별 인스턴스가 각각 다른 종류의 관계를 가지고 있다면 두 엔터티 사이에 두 개 이상의 관계 형성 가능
   - 관계 패어링 : 각 엔터티의 인스턴스들이 자신이 관련된 인스턴스들과 관계의 어커런스로 참여하는 형태
3. 관계의 분류 : 존재에 의한 관계, 행위에 의한 관계
4. 관계의 표기법
   ① 관계명 : 관계의 이름
      - 엔터티가 관계에 참여하는 형태
      - 애매한 동사x, 현재형 표현
   ② 관계차수 : 1:1, 1:N, M:N
   ③ 관계선택사양 : 필수관계, 선택관계
5. 체크사항
   - 두 엔터티 사이에 관심있는 연관 규칙 존재?
   - 두 엔터티 사이에 정보 조합 발생?
   - 업무기술서, 장표에 관계연결에 대한 규칙 서술?
   - 업무기술서, 장표에 관계연결 가능하게 하는 동사?

5절 식별자

1. 개념 : 엔터티 내에서 인스턴트들을 구분할 수 있는 구분자
2. 특징
   - 유일성 : 주식별자에 의해 엔터티 내의 모든 인스턴트들 유일하게 구분
   - 최소성 : 주식별자를 구성하는 속성의 수는 유일성을 만족하는 최소의 수
   - 불변성 : 주식별자가 한 번 지정되면 변하지 않아야 함
   - 존재성 : 주식별자가 지정되면 NULL값 허용x
3. 분류
   ① 대표성 여부 : 주식별자-대표, 참조관계연결 / 보조식별자
   ② 스스로 생성 여부 : 내부식별자-엔터티 내 스스로 생성 / 외부식별자-관계를 통해 받아옴
   ③ 속성의 수 : 단일식별자-하나의 속성으로 구성 / 복합식별자-둘 이상의 속성으로 구성
   ④ 대체여부 : 본질식별자-업무에 의해 / 인조식별자-원조식별자가 복잡해 인위적으로 만듦
4. 주식별자 도출 기준
   - 자주 이용되는 속성
   - 이름으로 기술되는 것x (명칭, 내역)
   - 복합 구성의 경우 너무 많이 포함x
5. 식별자 관계
   ① 식별자 관계 : 부모엔터티로부터 받은 외부식별자를 자식엔터티의 주식별자로 이용
      - 반드시 부모엔터티가 생성되어야 함
      - 강한 연결관계, 실선 표기
      - 주식별자 속성이 지속적으로 증가하는 구조 → 개발의 복잡성과 오류 가능성 유발
   ② 비식별자 관계 : 부모엔터티로부터 받은 외부식별자를 부모와 연결이 되는 속성으로서만 이용
      - 부모 없이 자식 생성 가능
      - 약한 연결관계, 점선 표기



2장 데이터 모델과 성능

1절 성능 데이터 모델링의 개요

1. 성능 데이터 모델링
   - 성능향상을 목적으로 설계단계의 데이터 모델링 때부터 여러 가지 성능 향상과 관련된 사항이 데이터 모델링에 반영될 수 있도록 하는 것
     - ex) 정규화, 반정규화, 테이블통합, 테이블분할, 조인구조, PK, FK 등
   - 분석/설계 단계에서 데이터베이스 처리 성능을 향상시킬 수 있는 방법 고려
   - 데이터의 증가가 빠를수록 성능저하에 따른 성능개선 비용 증가
2. 고려사항
   - 데이터 모델링을 할 때 정규화를 정확하게 수행
   - 데이터베이스 용량 산정 수행
   - 데이터베이스에 발생되는 트랜잭션 유형 파악
   - 용량과 트랜잭션의 유형에 따라 반정규화 수행
   - 이력 모델의 조정, PK/FK 조정, 슈퍼/서브타입 조정 등을 수행
   - 성능관점에서 데이터 모델 검증

2절 정규화와 성능

제1정규형 : 모든 속성은 반드시 하나의 값을 가져야 한다. (중복제거)

제2정규형 : 엔터티의 일반 속성은 주식별자 전체에 종속적이어야 한다. (부분적 함수종속성 제거)

- 함수 종속성 : 데이터들이 어떤 기준값에 의해 종속되는 현상 (기준: 결정자, 종속: 종속자)

제3정규형 : 엔터티의 일반속성 간에는 서로 종속적이지 않는다. (이행종속x)

1. 정규화를 통한 성능 향상 전략
   - 데이터를 결정하는 결정자에 의해 함수적 종속을 가지고 있는 일반 속성을 의존자로 하여 입력/수정/삭제 이상 현상을 제거
   - 중복속성을 제거하고 동일한 의미의 일반 속성이 하나의 테이블로 집약되어 한 테이블의 데이터 용량이 최소화
   - 속도가 빨라질 수도 있고 느려질 수도 있음
     - 조회 : 향상 혹은 저하 / 입력, 수정, 삭제 : 무조건 향상
2. 장점 : 유연성 극대화, 재활용 가능성 높아짐, 중복 최소화

3절 반정규화와 성능

1. 정의
   - 정규화된 엔터티, 속성, 관계에 대해 성능향상과 개발/운영 단순화를 위해 중복, 통합, 분리 등을 수행
   - 데이터를 조회할 때 디스크 I/O 양이 많아서 성능 저하, 경로 멀어 조인으로 인한 성능 저하, 칼럼 계산하여 읽을 때 성능 저하 예상 될 경우 반정규화 수행
2. 장점 : 일반적으로 조회 성능 향상
3. 단점
   - 조회 성능을 향상시키지만, 불필요한 입력(input), 갱신(update), 삭제(delete) 로직이 추가
   - 데이터 불일치로 인한 정합성 문제, 불필요한 트랜잭션으로 인한 성능 문제
4. 절차
   - 반정규화 대상 조사 : 범위 처리 빈도수, 대량의 범위 처리, 통계성 프로세스, 테이블 조인 개수
   - 다른 방법 유도 검토 : 뷰 테이블, 클러스터링, 인덱스 조정, 응용 애플리케이션
   - 반정규화 적용 : 테이블, 속성, 관계 반정규화 적용
5. 기법
   ① 테이블 반정규화
     - 테이블 병합 : 1:1 관계 테이블 병합, 1:M 관계 테이블 병합, 슈퍼/서브타입 테이블 병합
     - 테이블 분할 : 수직분할, 수평분할
     - 테이블 추가 : 중복 테이블 추가, 통계 테이블 추가, 이력 테이블 추가, 부분 테이블 추가

② 칼럼 반정규화
- 중복 칼럼 추가(조인감소), 파생 칼럼 추가(계산감소), 이력 테이블 칼럼 추가(조회향상), PK에 의한 칼럼 추가(조회향상), 응용시스템의 오작동을 위한 칼럼 추가

③ 관계 반정규화 : 중복 관계 추가, 테이블/칼럼 반정규화와 달리 데이터 무결성 영향 없음

4절 대량 데이터에 따른 성능

1. 대량 데이터 발생에 따른 테이블 분할
   - 수직분할 : 한 테이블에 많은 수의 칼럼 → 칼럼 단위로 분할하여 I/O 감소
   - 수평분할 : 대용량 테이블 → 로우 단위로 분할하여 I/O 감소
2. 대용량 테이블에서 발생하는 현상
   - 로우 체이닝(Row Chaining) : 로우 길이가 너무 길어서 두 개 이상의 블록에 걸쳐 하나의 로우가 저장
   - 로우 마이그레이션(Row Migration) : 데이터 블록에서 수정이 발새하면 수정된 데이터를 해당 데이터 블록에서 저장하지 못하고 다른 블록의 빈 공간을 찾아 저장하는 방식
   - 위의 현상이 발생하여 많은 블록에 데이터가 저장되면 데이터 조회 시 절대적인 Block I/O 횟수가 많아짐
3. 파티셔닝
   - Range Partition : 범위를 나눔. 대상 테이블이 날짜 또는 숫자 값으로 분리 가능하고 각 영역별로 트랜잭션이 분리 가능할 때 적용
   - List Partition : 특정값 지정. 지역 단위 파티셔닝 등에 적용
   - Hash Partition : Hash 조건에 따라 해싱 알고리즘이 적용되어 테이블 분리
4. 수평/수직 분할 절차
   - 데이터 모델링 완성
   - 데이터베이스 용량 산정
   - 대량 데이터가 처리되는 테이블의 트랜잭션 처리 패턴 분석
   - 집중화된 단위로 테이블 분리 검토

5절 데이터베이스 구조와 성능

1. 슈퍼/서브타입 모델 : 업무를 구성하는 데이터의 공통점과 차이점의 특징을 고려하여 효과적 표현, 논리적 모델
   - 슈퍼 타입 : 공통 부분
   - 서브 타입 : 공통으로부터 상속받아 다른 엔터티와 차이가 있는 속성
2. 변환기술

	One to One (1:1)	Plus (슈퍼+서브)	Single (All in One)
트랜잭션	개별로 발생	슈퍼+서브타입에 발생	전체를 하나로 묶어 발생
특징	개별 테이블 유지	슈퍼+서브 타입 테이블	하나의 테이블
확장성	우수	보통	나쁨
조인성능	나쁨	나쁨	우수
I/O 성능	좋음	좋음	나쁨
관리용이성	좋지않음	좋지않음	좋음

3. 인덱스 특성을 고려한 PK/FK DB 성능향상
   - 트랜잭션 조회 패턴에 따라 PK/FK 칼럼 순서 조정
   - 인덱스 선두 칼럼에 대한 조건 : 가능한 '=' 조건으로, 최소한 BETWEEN

6절 분산 데이터베이스와 성능

1. 분산 데이터베이스
   - 여러 곳으로 분산된 DB를 하나의 가상 시스템으로 사용할 수 있도록 한 DB
   - 논리적으로 동일한 시스템에 속하지만 컴퓨터 네트워크를 통해 물리적으로 분산되어 있는 데이터 집합
2. 분산 데이터의 투명성
   - 분할 투명성(단편화) : 하나의 논리적 Relation이 여러 단편으로 분할되어 각 단편의 사본이 여러 site에 저장
   - 위치 투명성 : 사용하려는 데이터의 저장 장소 명시 불필요. 위치정부가 System Catalog에 유지
   - 지역 사상 투명성 : 지역 DBMS와 물리적 DB 사이의 Mapping 보장
   - 중복 투명성 : DB 객체가 여러 site에 중복되어 있는지 알 필요가 없는 성질
   - 장애 투명성 : 구성요소(DBMS, Computer)의 장애와 무관한 트랜잭션의 원자성 유지
   - 병행 투명성 : 다수 트랜잭션 도잇 수행 시 결과의 일관성 유지, Time Stamp, 분산 2단계 Locking 이용
3. 분산 데이터베이스 장단점
   - 장점 : 지역자치성, 신뢰성, 가용성, 효용성, 융통성, 빠른 응답속도, 통신비용 절감, 각 지역 사용자의 요구 수용
   - 단점 : 소프트웨어 개발 비용, 오류 잠재성 증대, 처리 비용 증대, 설계/관리 복잡성, 불규칙한 응답속도, 통제 어려움, 데이터 무결성 위협
4. 분산 데이터베이스 적용 기법
   ① 테이블 위치 분산 : 테이블 위치 다르게(ex: 설계된 테이블을 본사와 자사 단위로 분산), 위치별 DB 문서 필요
   ② 테이블 분할 분산 : 각각의 테이블을 쪼개어 분산
      - 수평분할 : 특정 칼럼을 기준으로 Row 분리, 중복x
      - 수직분할 : 특정 칼럼 기준으로 Column 분리, 각 테이블에 동일 PK 있어야 함
   ③ 테이블 복제 분산 : 동일한 테이블을 다른 지역이나 서버에 동시 생성하여 관리
      - 부분 복제 : 마스터 DB에서 테이블의 일부 내용만 다른 지역이나 서버에 위치
      - 광역 복제 : 마스터 DB 테이블의 내용을 각 지역이나 서버에 존재
   ④ 테이블 요약 분산 : 지역/서버 간에 데이터가 비슷하지만 다른 유형으로 존재
      - 분석 요약 : 분산되어있는 동일한 테이블을 통합하여 전체에 대한 요약 정보 산출 (판매실적: A사, B사)
      - 통합 요약 : 분산되어있는 다른 테이블을 통합하여 전체에 대한 요약 정보 산출 (판매실적: A사-a, B사-b)
5. 분산 데이터베이스 설계를 고려해야 하는 경우
   - 성능이 중요한 사이트
   - 공통코드, 기준정보, 마스터 데이터의 성능향상
   - 실시간 동기화가 요구되지 않는 경우, Near Real Time 특징을 가진 경우
   - 특정 서버에 부하가 집중되어 부하를 분산
   - 백업 사이트 구성하는 경우

2과목 SQL 기본 및 활용

1장 SQL 기본

1절 관계형 데이터베이스 개요

1. 데이터베이스 : 특정 기업이나 조직 또는 개인이 필요에 의해 데이터를 일정한 형태로 저장해 놓은 것
   - DBMS : 효율적인 데이터 관리, 데이터 손상 방지, 데이터 복구를 위한 강력한 기능의 시스템
2. 관계형 데이터베이스
   - 정규화를 통한 합리적인 테이블 모델링을 통해 이상 현상 제거, 데이터 중복 피함, 동시성 관리, 병행 제어를 통해 많은 사용자들이 동시에 데이터 공유 및 조작 할 수 있는 기능 제공
   - 보안기능 제공, 데이터 무결성 보장
   - 갑작스런 장애로부터 사용자가 입력, 수정, 삭제 하려던 데이터가 제대로 반영될 수 있도록 보장
   - 시스템 다운, 재해 등의 상황에서도 데이터 회복/복구할 수 있는 기능 제공
3. SQL : 관계형 데이터베이스에서 데이터 정의, 조작, 제어를 위해 사용하는 언어
   - DML(Data Manipulation Language, 데이터 조작어) : SELECT, INSERT, UPDATE, DELETE
   - DDL(Data Definition Language, 데이터 정의어) : CREATE, ALTER, DROP, RENAME
   - DCL(Data Control Language, 데이터 제어어) : GRANT, REVOKE
   - TCL(Transaction Control Language, 트랜잭션 제어어) : COMMIT, ROLLBACK
4. 테이블 : 데이터를 저장하는 객체, 데이터베이스의 기본 단위
   - Column : 세로방향으로 이루어진 하나하나의 속성
   - Row : 가로방향으로 이루어진 데이터, 튜플, 인스턴스
   - 정규화 : 데이터의 정합성 확보&데이터 입력, 수정, 삭제시 발생할 수 있는 이상현상을 방지하기 위해 중복 제거
   - 기본키 : 테이블에 존재하는 각 행을 한 가지 의미로 특정할 수 있는 한 개 이상의 칼럼
   - 외부키 : 다른 테이블의 기본 키로 사용되고 있는 관계를 연결하는 칼럼
   - ERD(Entity Relationship Diagram) : 테이블 간 상관관계 도식화, 엔터티, 관계, 속성으로 구성

2절 DDL

1. 주요 데이터 타입
   - CHAR(L) : 고정 길이 문자열, 할당된 변수값이 L보다 작은 경우 공백으로 채움
   - VARCHAR2(L) : 가변 길이 문자열, L만큼의 최대 길이를 가짐, L보다 작은 경우 해당 값 만큼만 공간 차지
   - NUMBER(L, D) : 정수, 실수 저장, L값은 전체 자리 수, D값은 소수점 자리 수
   - DATE : 날짜와 시각 정보 “년월일시분초”
2. CREATE TABLE
   - 테이블명 : 단수형 권고, 중복X
   - 한 테이블 내에서 컬럼명 중복X
   - 칼럼은 ,로 구분 / 테이블 생성은 ;로 끝
   - 데이터 유형 반드시 지정
   - 테이블명과 칼럼명은 반드시 문자로 시작
   - A-Z, a-z, 0-9, _, $, # 문자만 허용
3. 제약조건 : 데이터 무결성 유지를 위해 특정 칼럼에 설정하는 제약
   - 기본키(PRIMARY KEY) : 하나의 테이블에 하나의 기본키, 자동으로 UNIQUE 인덱스 생성, NULL 불가
   - 고유키(UNIQUE KEY) : 행 데이터를 고유하게 식별, NULL 입력 가능
   - NOT NULL : NULL 입력 금지
   - CHECK : 입력할 수 있는 값 종류 및 범위 제한
   - 외래키(FOREIGN KEY) : 다른 테이블의 기본키를 외래키로 지정하는 경우 생성, NULL 가능, 여러 속성 가능

4. CREATE

CREATE TABLE USER_INFO (
USER_ID CHAR(6) **NOT NULL**,
USER_NAME VARCHAR2(10) **NOT NULL**,
POSTCODE CHAR(5));

5. ALTER

① ADD COLUMN

ALTER TABLE USER_INFO **ADD** (ADDRESS VARCHAR2(100));

② DROP COLUMN

ALTER TABLE USER_INFO **DROP** COLUMN ADDRESS;

③ MODIFY COLUMN

ALTER TABLE USER_INFO **MODIFY** (POSTCODE CHAR(5) **DEFAULT** '00000' **NOT NULL**);

④ RENAME COLUMN

ALTER TABLE USER_INFO **RENAME COLUMN** POSTCODE **TO** ZIPCODE;

⑤ DROP CONSTRAINT

ALTER TABLE USER_INFO **DROP CONSTRAINT** 조건명;

⑥ ADD CONSTRAINT

ALTER TABLE USER_INFO **ADD CONSISTRAINT** 조건명 조건(칼럼명);

6. TABLE

① RENAME TABLE

- **RENAME** USER_INFO **TO** USER_ADDRESS;

② TRUNCATE TABLE : 테이블 데이터 비우고, ROLLBACK 불가능

- **TRUNCATE TABLE** USER_ADDRESS;

③ DROP TABLE

- **DROP TABLE** USER_ADDRESS;

3절 DML

- DDL 명령어 AUTO COMMIT / DML 명령어 COMMIT 입력해야 됨

① INSERT

INSERT INTO USER_INFO (USER_ID, USER_NAME, ZIPCODE) **VALUES**('000001', 'MINA', '00000');
COMMIT;

② UPDATE

UPDATE USER_INFO **SET** ZIPCODE = '12345'
WHERE USER_ID = '000001';
COMMIT;

③ DELETE

DELETE FROM USER_INFO
WHERE USER_ID = '000001';
COMMIT;

④ SELECT

SELECT USER_NAME **FROM** USER_INFO;

- SELECT DISTINCT : 중복 제거
- 와일드카드 : *(모두), %(모두), -(한 글자)

⑤ ALIAS

SELECT USER_NAME **AS** NAME **FROM** USER_INFO;

⑥ 합성연산자 : || 이용하여 문자와 문자 연결

4절 TCL

1. 트랜잭션 : 밀접히 관련되어 분리될 수 없는 1개 이상의 DB 조작, 논리적 연산단위
   - 테이블 데이터의 변경 수행 시 변경되는 데이터의 무결성을 보장하는 COMMIT & ROLLBACK
   ① 원자성 : 트랜잭션에서 정의된 연산들은 모두 성공적으로 끝나거나 모두 실패해야 함 (All or Nothing)
   ② 일관성 : 트랜잭션 실행 전 잘못된 내용이 없다면 실행 이후에도 잘못된 내용이 없어야 함
   ③ 고립성 : 트랜잭션이 실행되는 중 다른 트랜잭션의 영향을 받아 잘못된 결과 만들어서는 안됨
   ④ 지속성 : 트랜잭션이 성공적으로 수행되면 갱신한 데이터베이스의 내용은 영구적으로 저장됨
2. TCL 명령
   ① COMMIT : 입력, 수정, 삭제한 자료에 문제가 없을 경우 COMMIT 명령어를 통해 트랜잭션 완료
      - COMMIT 이전 : 복구 가능, 현재 사용자만 SELECT로 결과 확인 가능, 변경된 행은 LOCKING 설정되어 다른 사용자가 변경 불가능
      - COMMIT 이후 : 변경사항 반영, 모든 사용자가 결과 확인 가능, LOCKING 풀려 모든 사용자가 조작 가능
   ② ROLLBACK : COMMIT 이전 변경사항을 취소하기 위한 기능
      - 데이터 변경 사항이 취소되어 데이터가 복구되며, LOCKING이 풀려 다른 사용자가 데이터 변경 가능
   ③ SAVE POINT : SAVE POINT 정의를 통해 트랜잭션의 일부만 ROLLBACK 가능

```
SAVEPOINT SVPT1;
-- DML --
ROLLBACK TO SVPT1;
```

3. COMMIT, ROLLBACK과 상관없이 트랜잭션 처리가 일어나는 상황
   - DDL 문장을 실행하면 전후 시점에 자동으로 커밋
   - 데이터베이스를 정상적으로 접속 종료하면 자동으로 커밋
   - 애플리케이션 이상 종료로 접속이 단절되면 트랜잭션 자동으로 롤백

5절 WHERE절

1. 개요
   - FROM절 다음에 위치
   - 구성 : 칼럼명, 비교 연산자, 문자/숫자/표현식, 비교 칼럼 명(JOIN 사용시)
2. 연산자
   ① 연산자 우선순위 : () → NOT → 비교, SQL 비교연산자 → AND → OR
   ② 비교연산자 : =, 〉, 〉=, 〈, 〈=
   ③ SQL 비교연산자 : BETWEEN A AND B, IN(LIST), LIKE '비교문자열', IS NULL
      - NULL은 =로 비교X
   ④ 논리연산자 : AND, OR, NOT
   ⑤ 와일드카드 : %(0개 이상의 문자), _(1개의 단일 문자)
   ⑥ 부정비교연산자 : !=, 〈〉, ^=(같지 않다) / NOT 칼럼명=(-와 같지 않다) / NOT 칼럼명〉(-보다 크지 않다)
3. 문자유형 비교방법
   ① 양쪽 CHAR
      - 길이가 서로 다르면 작은 쪽에 공백 추가하여 길이 같게 함
      - 서로 다른 문자가 나올 때까지 비교하고 달라진 첫 번째 값에 따라 크기를 결정
      - 공백의 수만 다르다면 같은 값으로 결정
   ② 한쪽 VARCHAR
      - 서로 다른 문자가 나올 때까지 비교 (공백도 문자로 판단)
      - 길이가 다르면 짧은 것이 끝날 때까지 비교하고 긴 것이 크다고 판단
      - 길이가 같고 다른 것이 없다면 같다고 판단
      - TRIM 함수 이용하여 공백 제거하고 비교하면 CHAR과 같은 결과
   ③ 상수값과 비교 : 상수 쪽의 타입을 CHAR / VARCHAR로 맞추어 적용

6절 함수

1. 단일행 함수
   - SELECT, WHERE, ORDER BY 절에서 사용 가능
   - 행에 개별적으로 조작
   - 여러 인자 입력해도 하나의 결과만 리턴
   - 함수 인자에 상수, 변수, 표현식 사용 가능
   - 함수 중첩 가능
2. 문자형 함수 : 문자 → 문자/숫자
   - LOWER(str) : 소문자로
   - UPPER(str) : 대문자로
   - ASCII('a') : 아스키 값 반환
   - CHR('ascii') : ASCII 값에 해당하는 문자 출력
   - CONCAT(A, B) : A, B 연결
   - SUBSTR(str, m, n) : 문자열 슬라이싱
   - LENGHT(str) : 문자열 길이 반환
   - TRIN(str), RTRIM(str), LTRIM(str) : 공백 제거
3. 숫자형 함수 : 숫자 → 숫자
   - ABS() : 절대값
   - SIGN() : 양수 → 1, 0 → 0, 음수 → -1
   - MOD(A, B) : A/B 나머지
   - CEIL() : 올림
   - FLOOR() : 내림
   - ROUND(num, m) : m자리에서 반올림
   - TRUNC(num, m) : m자리에서 자름
4. 날짜형 함수 : DATE
   - SYSDATE : 현재 날짜와 시각 출력
   - EXTRACT : 날짜에서 데이터 출력
     - YEAR FROM d, MONTH FROM d, DAY FROM d
   - TO_NUMBER(TO_CHAR(d, 'YYYY')) : 연도 숫자로 출력
     - YYYY-MM-DD HH:MI:SS
   - 1 = 하루
5. 변환형 함수 : 데이터 타입 변환
   - 암시적 형변환 : DBMS가 자동으로 데이터 유형 변환
   - 명시적 형변환 : 데이터 변환 함수로 데이터 유형을 변환하도록 명시
6. 단일행 CASE 표현
   - CASE WHEN 조건 THEN 값 ELSE 값 END;
   - DECODE(칼럼명, 조건1, 값1, 조건2, 값2, 디폴트값)
7. NULL관련 함수
   - NVL(A, B) : A의 값이 NULL이면 B 출력 → NULL값을 대상으로 하는 것으로 공집합은 X
   - NULLIF(A, B) : A와 B가 같으면 NULL 아니면 A
   - COALESCE(A, B, ...) : NULL이 아닌 최초의 표현식, 모두 NULL이면 NULL 반환

7절 GROUP BY, Having절

1. 다중행 집계 함수
   - 여러 행들의 그룹이 모여서 그룹당 단 하나의 결과를 돌려주는 함수
   - GROUP BY 절은 행들을 소그룹화 함
   - SELECT, HAVING, ORDER BY 절에 사용 가능
   - ALL : DEFAULT 옵션으로 생략 가능 / DISTINCT : 중복제거
2. 집계함수 종류
   - COUNT(*) : NULL 포함 행의 수
   - COUNT(표현식) : 표현식의 값이 NULL이 아닌 행의 수
   - SUM, AVG, MAX, MIN, STDDEV, VARIAN : NULL 제외 합, 평균, 최대, 최소, 표준편차, 분산
3. GROUP BY절
   - 소그룹별 기준을 정한 후 SELECT절에 집계 함수 사용
   - 집계 함수의 통계 정보는 NULL값을 가진 행을 제외하고 수행
   - ALIAS 사용 불가
   - GROUP 나누기 전에 WHERE절이 행을 미리 제거 → WHERE절에서 집계 함수 사용X
   - HAVING절은 일반적으로 GROUP BY절 뒤에 오며, 기준 항목이나 소그룹의 집계함수를 이용한 조건 표시 가능

8절 Order by절

1. ORDER BY 특징
   - SQL 문장으로 조회된 데이터들을 다양한 목적에 맞게 특정한 칼럼을 기준으로 정렬하여 출력
   - ALIAS명, 칼럼 순서를 나타내는 정수 사용 가능
   - DEFAULT값 ASC(오름차순), DESC 옵션으로 내림차순 정렬 가능
   - SQL 문장의 제일 마지막에 위치
   - SELECT절에서 정의하지 않은 칼럼 사용 가능
   - Oracle에서는 NULL이 제일 큰 값, SQL Server에서는 NULL이 가장 작은 값
2. SELECT 문장 실행 순서
   - SELECT ALIAS → FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY
3. SQL Server의 WITH TIES
   : WITH TIES 조건 추가하면 TOP(2)에서 공동 2위 모두 출력

9절 Join

1. JOIN : 두 개 이상의 테이블들을 연결 또는 결합하여 데이터를 출력하는 것
   - 일반적으로 PK/FK값의 연관으로 JOIN되지만, PK/FK 관계 없어도 논리적인 값들의 연관만으로 JOIN 가능
2. EQUI JOIN : 2개의 테이블 간에 칼럼 값들이 서로 정확하게 일치하는 경우에 사용, 대부분 PK/FK 관계 기반
   - **WHERE** PLAYER.TEAM_ID = TEAM.TEAM_ID
   - **FROM** PLAYER **INNER JOIN** TEAM **ON** PLAYER.TEAM_ID = TEAM.TEAM_ID
   - INNER JOIN 참여하는 대상 테이블이 N개일 때 필요한 JOIN 조건은 N-1개
3. NON EQUI JOIN : 칼럼 값이 정확하게 일치하지 않아 BETWEEN, 〈 등 연산자 사용
   - E.SAL의 값을 LOSAL과 HSAL 범위에서 찾는 것

   WHERE E.SAL **BETWEEN** S.LOSAL **AND** S.HSAL;

2장 SQL 활용

1절 표준 조인

1. 일반 집합 연산자와 SQL의 비교
   - UNION 연산 - UNION 기능 : 합집합, 시스템에 부하주는 작업 발생
   - INTERSECTION 연산 - INTERSECT 기능 : 교집합
   - DIFFRENCE 연산 - EXCEPT(Oracle은 MINUS) : 차집합
   - PRODUCT 연산 - CROSS JOIN : 곱집합, JOIN 조건이 없는 경우 생길 수 있는 모든 데이터 조합, M*N
2. 순수 관계 연산자와 SQL의 비교
   - SELECT - WHERE, PROJECT - SELECT, NATURAL JOIN - 다양한 JOIN, DIVIDE - 사용X
3. 조인의 형태
   - ANSI SQL의 JOIN : INNER JOIN(NATURAL JOIN, USING, ON), CROSS JOIN, OUTER JOIN
   - INNER JOIN : JOIN 조건에서 도일한 값이 있는 행만
   - NATURAL JOIN : 두 테이블 간 동일한 이름을 갖는 모든 칼럼에 대해 EQUI JOIN
     - JOIN 칼럼은 ALIAS 사용X
   - USING 조건절 : 같은 이름을 가진 칼럼들 중 원하는 칼럼에 대해서만 선택적으로 EQUI JOIN
     - **FROM** USER_INFO **JOIN** PROJECT **USING (**USER_ID)
   - ON 조건절 : 컬럼명이 다르더라도 JOIN 사용 가능
     - **FROM** USER_INFO U **JOIN** PROJECT P **ON** (U.USER_ID = P.USER_ID)
   - CROSS JOIN : JOIN 조건이 없는 경우 생길 수 있는 모든 데이터 조합
   - OUTER JOIN : JOIN 조건에서 동일한 값이 없는 행도 반환
     - LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN

2절 집합 연산자

: JOIN 사용하지 않고 연관된 데이터 조회하는 법

1. UNION
   - 여러 개의 SQL문의 결과에 대한 합집합, 중복행은 한 개의 행으로 출력
2. UNION ALL
   - 여러 개의 SQL문의 결과에 대한 합집합, 중복행 그대로 출력
3. INTERSECT
   - 여러 개의 SQL문에 대한 교집합, 중복행 한 개의 행으로 출력
   - EXISTS, IN 서브쿼리로 변경 가능
4. EXCEPT/MINUS
   - 위의 SQL문의 집합에서 아래의 SQL문의 집합을 뺀 결과 (차집합)
   - NOT EXIST, NOT IN 서브쿼리로 변경 가능

3절 계층형 질의와 셀프 조인

1. 계층형 질의
   - 테이블에 계층형 데이터 존재하는 경우 조회를 위해 계층형 질의 사용
   - 동일 테이블에 계층적으로 상위와 하위 데이터가 포함된 데이터

   ① SELECT : 조회하고자 하는 칼럼 지정

   ② FROM TABLE : 대상 테이블 지정

   ③ WHERE : 필터링

   ④ START WITH : 루트 데이터 지정

   ⑤ CONECT BY [NOCYCLE] [PRIOR] A AND B : 다음에 전개될 자식 데이터 지정
     - PRIOR 자식=부모 : 자식 → 부모 방향(순방향) / PRIOR 부모=자식 : 부모 → 자식(역방향
     - NOCYCLE 추가하면 사이클 발생한 이후 데이터 전개X

   ⑥ ORDER SIBLINGS BY 칼럼 : 동일 LEVEL 사이에서 정렬 수행

⑦ 가상칼럼
- LEVEL : 루트데이터 1, 하위 데이터 있을 때마다 1씩 증가
- CONNECT_BY_ISLEAF : 전개 과정에서 리프데이터면 1, 아니면 0
- CONNECT_BY_ISCYCLE : 전개과정에서 자식을 갖는데 해당 데이터가 조상이면 1, 아니면 0

2. 셀프 조인 : 동일 테이블 사이의 조인

SELECT WORKER.EMPNO, WORKER.ENAME, MANAGER.ENAME
FROM EMP WORKER, EMP MANAGER
WHERE WORKER.MGR = MANAGER.EMPNO;

4절 서브쿼리

1. 서브쿼리
① 서브쿼리란 : 하나의 SQL문 안에 포함되어 있는 또 다른 SQL문. 서브쿼리는 메인 쿼리의 칼럼 모두 사용가능하지만, 메인 쿼리는 서브 쿼리의 칼럼 사용X
② 주의점
- 서브쿼리를 괄호로 감싸서 사용
- 단일행 또는 복수행 비교 연산자와 함께 사용 가능
- 단일행 비교 연산자는 서브쿼리의 결과가 반드시 1건 이하, 복수행 비교 연산자는 결과 건수와 상관 없음
- 서브쿼리에서는 ORDER BY 사용 불가 / 반드시 메인 쿼리의 마지막 문장에 위치

③ 사용 가능한 위치
- SELECT절 - FROM절 - WHERE절 - HAVING절 - ORDER BY절
- INSERT문의 VALUES절 - UPDATE문의 SET절

2. 동작방식에 따른 분류
① 비 연관 서브 쿼리 : 서브 쿼리가 메인 쿼리의 칼럼을 가지고 있지 않음. 메인 쿼리에 값을 제공하기 위한 목적
② 연관 서브 쿼리 : 서브 쿼리가 메인 쿼리의 값을 가지고 있음. 메인 쿼리가 먼저 수행되어 읽혀진 데이터를 서브 쿼리에서 조건이 맞는지 확인할 때 사용
- EXIST 서브쿼리는 항상 연관 서브쿼리로 사용, 여러 건을 만족하더라도 1건만 찾으면 추가 검색X

3. 반환 형태에 따른 분류
① 단일행 서브 쿼리 : 실행 결과가 1건 이상, 항상 비교 연산자와 함께 사용(=, 〈, 〉, 〈=, 〉=, 〈〉)
② 다중행 서브 쿼리 : 실행 결과가 여러 건, 다중행 비교 연산자와 함께 사용(IN, ALL, ANY, SOME, EXISTS)
③ 다중 칼럼 서브 쿼리 : 실행 결과로 여러 칼럼 반환, 메인 쿼리 조건에 여러 칼럼 동시 비교 가능, 서브 쿼리와 메인 쿼리의 칼럼 수와 순서가 동일해야 함

4. 그 밖의 위치에서 사용하는 서브쿼리
① 스칼라 서브 쿼리 : SELECT절에서 사용, 한 행/한 칼럼만을 반환
② 인라인 뷰 : FROM절에서 사용
③ HAVING절, INSERT문의 VALUES절, UPDATE문의 SET절에서도 사용

5. 뷰(실제 데이터 가지고 있지 않는 가상테이블) 사용의 장점
① 독립성 : 테이블 구조 변경되어도 뷰를 사용하는 응용프로그램은 변경X
② 편리성 : 복잡한 질의를 뷰로 생성하여 관련 질의를 단순하게 작성
③ 보안성 : 숨기고 싶은 칼럼을 빼고 뷰를 생성하면 정보 감출 수 있음

5절 그룹 함수 : 그룹 함수를 이용하여 특정 집합의 소계, 중계, 합계, 총 합계 구할 수 있음

1. ROLLUP : 소 그룹간 소계 계산, 인자 순서 바뀌면 결과 바뀜(계층 구조)
GROUP BY ROLLUP (DNAME, JOB); → DNAME1 - JOB1,2 , DNAME2 - JOB1,2

2. CUBE : 결합 가능한 모든 값에 대하여 다차원적인 소계 계산, 2^N만큼의 SUBTOTAL 생성
GROUP BY CUBE (DNAME, JOB);

3. GROUPING SETS : 특정 항목에 대한 소계 계산, 인자 순서 바뀌어도 결과 그대로(평등 관계)
GROUP BY GROUPING SETS (DNAME, JOB); DNAME1,2 - ALL JOB , ALL DNAME - JOB1,2

6절 윈도우 함수

- 행과 행 간의 관계에서 다양한 연산 처리 가능, 중첩하여 호출X

SELECT 윈도우함수(인자) **OVER** (**PARTITION BY** 칼럼 **ORDER BY** 칼럼)
윈도우절
FROM 테이블명;

1. 순위 관련 함수
   - RANK : 특정 항목(칼럼)에 대한 우선순위, 동일한 값은 동일한 순위
   - DENSE_RANK : 동일한 순위를 하나의 건으로 취급
   - ROW_NUMBER : 동일한 값이라도 고유 순위 부여
2. 집계 관련 함수 : SUM, MAX, MIN, AVG, COUNT
   - SQL Server의 경우 집계함수는 OVER절 내의 ORDER BY 지원X
3. 행 순서 관련 함수 (ORACLE에서만 지원)
   - FIRST_VALUE : 파티션별 윈도우에서 가장 먼저 나온 값, 공동 등수 인정하지 않고 처음 나온 행만 처리
   - LAST_VALUE : 파티션별 윈도우에서 가장 나중에 나온 값, 공동 등수 인정하지 않음
   - LAG : 파티션별 윈도우에서 이전 몇 번째 행의 값
     - 2행 앞의 SALARY 가져오기, 없으면 0으로 처리

     SELECT ENAME, HIREDATE, SAL **LAG**(SAL, 2, 0) **OVER** (**ORDER BY** HIREDATE) **AS** PREV_SAL **FROM** EMP **WHERE** JOB='SALESMAN';
   - LEAD : 파티션별 윈도우에서 이후 몇 번째 행의 값
     - 입사일자 빠른 순 정렬, 바로 다음에 입사한 인력의 입사일자 함께 출력

     SELECT ENAME, HIREDATE **LEAD**(HIREDATE, 1) **OVER** (**ORDER BY** HIREDATE) **AS** "NEXTHIRED" **FROM** EMP;
4. 그룹 내 비율 관련 함수
   - CUME_DIST : 현재 행보다 작거나 같은 건수에 대한 누적 백분율
   - PERCENT_RANK : 파티션별 윈도우에서 처음 값을 0, 마지막 값을 1로 하여 순서별 백분율
   - NTILE : 파티션별 전체 건수를 인수 값으로 N등분 한 결과
   - RATIO_TO_REPORT : 파티션 내 전체 SUM(칼럼) 값에 대한 행별 칼럼 값의 백문율을 소수점으로 구함

7절 DCL

1. DCL : 유저를 생성하고 권한을 제어할 수 있는 명령어
   - SCOTT : 테스트용 샘플 유저
   - SYS : DBA 권한을 부여받은 최상위 유저
   - SYSTEM : SYSTEM 데이터베이스의 모든 시스템 권한을 부여받은 SYS 바로 밑 유저
2. ORACLE과 SQL Server의 사용자 아키텍처 차이
   - ORACLE : 유저를 통해 DB에 접속, ID와 PW 방식으로 인스턴스에 접속하고 그에 해당하는 스키마 오브젝트 생성 등의 권한 부여받음
   - SQL Server : 인스턴스에 접속하기 위해 로그인 생성, 인스턴스 내에 존재하는 다수의 DB에 연결하여 작업하기 위해 유저 생성 후 로그인과 유저 매핑, Windows 인증 방식과 혼합 모드 방식 존재
3. 시스템 권한 : 사용자가 SQL문을 실행하기 위해 필요한 적절한 권한 (
   - GRANT : 권한 부여 - **GRANT CREATE USER TO** SCOTT;
   - REVOKE : 권한 취소 - **REVOKE CREATE TABLE FROM** USR;
4. ROLE : 유저에게 알맞은 권한들을 한 번에 부여하기 위해 사용
   - CONNECT : SESSION
   - RESOURCE : CLUSTER, PROCEDURE, TYPE, SEQUENCE, TRIGGER, OPERATOR, TABLE, INDEXTYPE

DROP ROLE TEST;
CREATE ROLE TEST;
GRANT CREATE TABLE TO TEST;
GRANT TEST **TO** TEST_USR;

8절 절차형 SQL

1. 절차형 SQL : SQL문의 연속적 실행이나 조건에 따른 분기처리를 이용하여 특정 기능을 수행하는 저장모듈인 PROCEDURE, TRIGGER, USER DEFINED FUNCTION 만들 수 있음
   - 저장모듈 : PL/SQL 문장을 DB 서버에 저장하여 사용자와 애플리케이션 사이에서 공유할 수 있도록 만든 일종의 SQL 컴포넌트 프로그램, 독립적으로 실행되거나 다른 프로그램으로부터 실행될 수 있는 완전한 실행 프로그램
   - T-SQL : SQL Server를 제어하기 위한 언어
2. PL/SQL : Block 구조로 되어 있고, SQL문, IF, LOOP 등이 포함
   ① 특징
   - Block 구조로 되어있어 각 기능별로 모듈화 가능
   - 변수/상수 선언 및 IF/LOOP문 등의 사용이 가능
   - DBML 에러나 사용자 에러 정의 가능
   - 오라클에 내장되어 있으므로 호환성 좋음
   - Block단위로 묶어 한번에 서버로 보내기 때문에 네트워크 패킷 수 감소

   ② 구조
   - DECLARE(선언부, 필수) : BEGIN~END에서 사용할 변수나 인수에 대한 정의 및 데이터 타입 선언
   - BEGIN(실행부, 필수) : 개발자가 처리하고자 하는 SQL문과 LOGIC이 정의되는 실행부
   - EXCEPTION(예외처리부, 선택) : SQL문에 발생된 에러를 처리하는 예외처리부
   - END(필수)
3. 사용자 정의 함수 : 프로시저처럼 SQL문을 IF/LOOP 등의 LOGIC과 함께 데이터베이스에 저장한 명령집합
   - SUM, AVG, NVL의 함수처럼 호출사용 가능
   - 프로시저와의 차이점 : 반드시 한 건을 RETURN해야 함
4. TRIGGER : 특정한 테이블에 DML문이 수행되었을 때, DB에서 자동으로 동작하도록 작성한 프로그램
   - 사용자가 호출하지 않고 DBMS가 자동으로 수행
   - 이벤트 발생 대상 : 테이블, 뷰, 데이터베이스
   - 발생 범위 : 전체 트랜잭션 작업, 각 행에 대해서 발생

   CREATE OR REPLACE TRIGGER TRIGER_TEST

   ▷ 프로시저와 트리거 차이점

프로시저	트리거
CREATE PROCEDURE 문법 사용	CREATE TRIGGER 문법 사용
EXECUTE/EXEC 명령어로 실행	생성 후 자동으로 실행
내부에서 COMMIT, ROLLBACK 실행 가능	내부에서 COMMIT, ROLLBACK 실행 안됨

3장 SQL 최적화 기본 원리

1절 옵티마이저와 실행계획

2절 인덱스 기본

3절 조인 수행 원리