1장. 데이터 모델링의 이해

01. 데이터 모델의 이해

1. 데이터 모델의 기본 개념과 이해

1-1. 데이터 모델링 정의
1) 정보시스템을 구축하기위한 데이터 관점의 업무 분석 기법 -> 업무 내용을 정확하게 분석할 수 있음
2) 현실세계의 데이터에 대해 약속된 표기법에 의해 표현하는 과정
3) 데이터베이스를 구축하기 위한 분석/설계의 과정

따라서, 데이터베이스만을 구축하기 용도로 쓰이는 것이 아니라 데이터모델링 자체로서 업무를 설명하고 분석하는 부분에서도 매우 중요한 의미를 가짐

1-2. 모델링의 특징
1) 추상화
: 현실세계를 일정한 형식에 맞추어 표현함
다양한 현상을 일정한 양식인 표기법에 의해 표현
2) 단순화
: 복잡한 현실세계를 약속된 규약에 의해 제한된 표기법이나 언어로 표현하여 쉽게 이해 할 수 있도록 함
3) 명확화
: 누구나 이해하기 쉽게 하기 위해 대상에 대한 애매모호함을 제거하고 정확하게 현상을 기술

*모델링 = 현실세계를 추상화, 단순화, 명확화하기 위해 일정한 표기법에 의해 표현하는 기법

1-3. 데이터 모델이 제공하는 기능
1) 현재 시스템의 모습 또는 원하는 모습으로 가시화
2) 시스템의 구조와 행동을 명세화
3) 시스템을 구축하는 과정에서의 문서화
4) 데이터를 추상화하는 과정에서 다양한 관점에서 볼 수 있음
5) 특정 목표에 따라 구체화된 상세 수준의 표현방법 제공

2. 데이터 모델링 유의점

데이터 모델은 구축할 시스템의 정보 요구사항이 정확하고 간결하게 표현해야 한다.

1) 중복(Duplication)
: 데이터베이스가 여러 장소에 같은 정보를 저장하지 않도록 함
2) 비유연성(Inflexibility)
: 데이터모델이 수시로 변경될 경우 유지보수가 어려워 질 수 있다. 이를 위해 데이터의 정의를 데이터 사용 프로세스와 분리해야한다.
그러므로 데이터 혹은 프로세스의 작은 변화가 애플리케이션과 데이터베이스에 중대한 변화를 일으킬 수 있는 가능성을 줄이도록 한다.
3) 비일관성(Inconsistency)
: 데이터의 중복이 없더라도 비일관성이 발생할 수 있다. 그러므로 데이터간의 상호 연관 관계에 대한 명확한 정의해야 한다.
사용자가 처리하는 프로세스 혹은 이와 관련된 프로그램과 테이블의 연계성을 높이는 것은 데이터모델이 업무 변경에 대해 취약하게 만드는 단점에 해당

3. 데이터 모델링 개념

1) 개념적 데이터 모델링
: 추상화 수준이 높고 업무중심적이고 포괄적인 수준의 모델링 진행
전사적 데이터 모델링
EA수립시 많이 이용
2) 논리적 데이터 모델링 : 시스템으로 구축하고자 하는 업무에 대해 Key, 속성, 관계 등을 정확하게 표현
재사용성이 높음
3) 물리적 데이터 모델링 : 실제로 데이터베이스에 이식할 수 있도록 성능, 저장 등 물리적인 성격을 고려하여 설계

4. 데이터베이스 독립성

데이터베이스 스키마 구조는 3단계로 구분되고 각각은 상호 독립적인 의미를 가지고 고유한 기능을 가짐

4-1. 데이터 독립성 요소
1) 외부스키마(External Schema)
: 사용자 관점으로 구성, 개개 사용자가 보는 개인적 DB스키마
2) 개념스키마(Conceptual Schema)
: 통합관점의 스키마구조 표현, 전체 입장에서 DB정의
3) 내부스키마(internal Schema)
: 물리적 장치에서 데이터가 실제적으로 저장되는 방법을 표현

4-2. 데이터 독립성
1) 논리적 독립성
: 개념 스키마가 변경되어도 외부 스키마 변하지 않음
2) 물리적 독립성
: 내부 스키마가 변경되어도 외부/개념 스키마 영향 받지 않음

4-3. 사상(Mapping)
외부 화면이나 사용자에게 인터페이스 하기 위한 스키마 구조는 전체가 통합된 개념적 스키마와 연결되는 것이 논리적인 사상
통합된 개념적 스키마 구조와 물리적으로 저장된 구조의 물리적인 테이블 스페이스와 연결되는 구조가 물리적 사상

5. 데이터 모델링의 3요소

1) 업무가 관여하는, 사물이나 사건들 어떤 것(Things)
2) 어떤 것이 가지는 성격(Attributes)
3) 업무가 관여하는 어떤 것 간의 관계(Relationships)

이 세가지는 데이터 모델링을 완성해가는 핵심 개념으로서 결국 엔티티, 속성, 관계로 인식됨

6. 데이터 모델의 표기법 ERD

- 1976년 피터첸이 Entity-Relationship Model 만듬
- ERD 작성순서
: 엔티티 도출 -> 앤티티 배치 -> 엔티티 관계 설정 -> 관계명 기술 -> 관계의 필수여부 기술

7. 좋은 데이터 모델의 요소

1) 완전성
2) 중복배제
3) 업무규칙
4) 데이터 재사용
5) 의사소통
6) 통합성

02. 엔티티

1. 엔티티의 개념

- 업무에 필요하고 유용한 정보를 저장하고 관리하기 위한 집합적인 것(Thing)
- 최소의 데이터 단위인 속성(Attribute)를 갖고, 인스턴스의 집합이라고 말할 수 있음
- 눈에 보이지 않는 개념들도 엔티티라고 할 수 있음

2. 엔티티와 인스턴스

과목	강사
수학	홍길동
영어	이순신

과목 & 강사 : 엔티티
수학,홍길동 & 영어&이순신 : 인스턴스

3. 엔티티의 특징

1) 반드시 해당 업무에서 필요하고 관리하고자 하는 정보 (환자, 토익의 응시횟수..)
2. 유일한 식별자에 의해 식별 가능
3. 두 개 이상의 인스턴스의 집합
4. 업무 프로세스에 의해 이용되어야 함
5. 반드시 속성이 있어야 함
6. 다른 엔터티와 최소 1개 이상의 관계가 있어야 함

4. 엔티티의 분류

4-1. 유무형에 따른 분류
1) 유형엔티티
: 물리적인 형태가 있고, 안정적임, 지속적으로 활용
2) 개념엔티티
: 관리해야 할 개념적 정보 (보험상품..)
3) 사건엔티티
: 업부 수행시 발생, 발생량 많음 (주문, 청구..)

4-2. 발생시점에 따른 분류
1) 기본엔티티
: 업무에 원래 존재하는 정보, 다른 엔티티로부터 상속받지 않고 독립적으로 생성가능 (사원, 부서..)
2) 중심엔티티
: 기벤엔티티로부터 발생, 업무에 중심적인 역할
3) 행위엔티티
: 두 개 이상의 부모엔티티로부터 발생
자주 내용이 바뀌거나 데이터 양 증가

스스로 생성될 수 있는지에 따라 독립엔티티, 의존엔티티 구분

5. 엔티티 이름 부여하는 방법

1) 현업업무에서 사용하는 용어를 사용
2) 약어를 사용하지 않음
3) 단수명사를 사용
4) 모든 엔티티에서 유일하게 이름이 부여
5) 엔티티 생성의미대로 이름을 부여

03. 속성

1. 속성 개념

업무에서 필요로 하는 인스턴스에서 관리하고자 하는 의미상 더 이상 분리되지 않는 최소의 데이터 단위

2. 엔티티, 인스턴스, 속성, 속성값의 관계

- 한 개의 엔티티는 두 개 이상의 인스턴스의 집합이어야 함
- 한 개의 엔티티는 두 개 이상의 속성을 갖음
- 한 개의 속성은 한 개의 속성값을 갖음

3. 속성의 분류

3-1. 속성 특성에 따른 분류
1) 기본속성
  : 업무로부터 추출한 모든 속성, 엔티티에 가장 일반적이고 많은 속성을 차지함 (이자율..)
2) 설계속성
  : 업무를 규칙화 하기 위해 속성을 새로 만들거나 변형하여 정의하는 속성 (일련번호..)
3) 파생속성
  : 다른 속성에 영향을 받아 발생하는 속성
  빠른 성능을 할 수 있도록 계산된 값들 (이자..)

3-2. 엔티티구성방식에 따른 분류
1) PK속성(Primary Key)
  : 엔티티를 식별할 수 있는 속성
2) FK속성(Foreign Key)
  : 다른 엔티티와의 관계에서 포함된 속성
3) 일반속성
  : PK, FK에 포함되지 않는 속성

4. 속성의 명칭 부여 방법

1) 해당업무에서 사용하는 이름을 부여
2) 서술식 속성명은 사용하지 않음
3) 약어 사용은 가급적 제한
4) 전체 데이터모델에서 유일성을 확보하는 것이 좋음

5. 도메인

각 엔티티의 속성에 대해서 어떤 유형의 값이 들어가는지 정의하는 개념

04. 관계

1. 관계 개념

엔터티의 인스턴스 사이의 논리적인 연관성으로서 존재의 형태로서나 행위로서 서로에게 연관성이 부여된 상태

관계는 존재에 의한 관계와 행위에 대한 관계로 구분되나 ERD에서는 관계를 연결할 때, 존재와 행위를 구분하지 않고 단일화된 표기법을 사용함
- 존재에 의한 관계
  : 사원은 부서에 항상 속해있음
- 행위에 의한 관계
  : 주문은 고객이 주문할 때 발생함

2. 관계의 패어링

각 엔티티의 인스턴스가 개별적으로 관계를 가지는 것

UML(Unified Modeling Language)에는 클래스다이어그램의 관계 중 연관관계(Association)와
의존관계(Dependency)가 있고 이것은 실선과 점선의 표기법으로 다르게 표현

3. 관계의 표기법

- 관계명
  : 관계의 이름
- 관계차수
  : 1:1, 1:M, M:N
- 관계선택사항
  : 필수관계, 선택관계

4. 관계 체크사항 및 관계 읽기

4-1. 관계 체크사항
- 두 개의 엔티티 사이에 관심있는 연관규칙이 존재하는가?

- 두 개의 엔티티 사이에 정보의 조합이 발생되는가?
- 업무기술서, 장표에 관계연결에 대한 규칙이 서술되어 있는가?
- 업무기술서, 장표에 관계연결을 가능하게 하는 동사가 있는가?

4-2. 관계 읽기
- 기준(Source) 엔티티를 한 개 또는 각으로 읽는다.
- 대상 엔티티의 관계참여도 즉 개수를 읽는다.
- 관계선택사양과 관계명을 읽는다.

05. 식별자

1. 식별자 개념

하나의 엔티티에 구성되어 있는 여러 개의 속성 중에서 엔티티를 대표할 수 있는 속성을 의미하며, 하나의 엔티티는 반드시 하나의 유일한 식별자가 존재해야 함

2. 식별자 분류

분류	식별자	설명
대표성여부	주식별자	엔티티 내에서 각 래퍼런스를 구분할 수 있는 구분자이며, 타 엔티티와 참조관계를 연결할 수 있는 식별자
	보조식별자	엔티티 내에서 각 어커런스를 구분할 수 있는 구분자이나 대표성을 가지지 못해 참조관계 연결을 못함
스스로생성여부	내부식별자	엔티티 내부에서 스스로 만들어지는 식별자
	외부식별자	타 엔티티와의 관계를 통해 타 엔티티로부터 받아오는 식별자
속성의 수	단일식별자	하나의 속성으로 구성된 식별자
	복합식별자	둘 이상의 속성으로 구성된 식별자
대체여부	본질식별자	업무에 의해 만들어지는 식별자
	인조식별자	업무적으로 만들어지지는 않지만 원조식별자가 복잡한 구성을 가지고 있기 때문에 인위적으로 만든 식별자

3. 주식별자 특징

- 유일성
 : 엔티티 내에 모든 인스턴스들 유일하고 구분함
  예) 사원번호가 주식별자가 모든 직원들에 대해 개인별로 고유하게 부여됨
- 최소성
 : 주식별자를 구성하는 속성의 수는 유일성을 만족하는 최소의 수가 되어야 함
  예) 사원번호만으로도 고유한 구조인데 사원분류코드 + 사원번호로 식별자가 구성될 경우 부적절한 주식별자 구조임
- 불변성
 : 주식별자가 한 번 특정 엔티티에 지정되면 그 식별자의 값은 변하지 않아야 함
  예) 사원번호의 값이 변한다는 의미는 이전기록이 말소되고 새로운 기록이 발생되는 개념임
- 존재성
 : 반드시 데이터 값이 존재해야함 (Null 값 안됨)
  예) 사원번호 없는 회사직원은 있을 수 없음

4. 식별자와 비식별자관계 비교

항목	식별자관계	비식별자관계
목적	강한 연결관계 표현	약한 연결관계 표현
자식 주식별자 영향	자식 주식별자의 구성에 포함됨	자식 일반 속성에 포함됨
표기법	실선 표현	점선 표현
연결 고려사항	- 반드시 부모엔티티 종속 - 자식 주식별자구성에 부모	- 약한 종속관계 -자식 주식별자구성을 독립적으로 구성 - 자식

	주식별자포함 필요 - 상속받은 주식별자속성을 타엔티티에 이전 필요	주식별자구성에 부모 주식별자 부분 필요 - 상속받은 주식별자속성을 타 엔티티에 차단 필요 - 부보쪽의 관계참여가 선택관계

2장. 데이터 모델과 성능

01. 성능 데이터 모델리의 개요

1. 성능데이터모델링

- DB 성능향상을 목적으로 설계단계의 데이터 모델링 때부터 정규화, 반정규화, 테이블통합, 테이블분할, 조인구조, PK, FK 등 여러 가지 성능과 관련된 사항이 데이터 모델링에 반영될 수 있도록 하는 것
- 분석/설계 단계에서 데이터 모델에 성능을 고려한 데이터 모델링을 수행할 경우 성능저하에 따른 재업무 비용을 최소화 할 수 있는 기회를 가지게 됨
- 데이터의 증가가 빠를수록 성능저하에 따른 성능개선 비용은 기하급수적으로 증가

2. 성능 데이터 모델링 고려사항 순서

1. 데이터 모델링을 할 때 정규화를 정확하게 수행
2. DB 용량산정을 수행한다.
3. DB에 발생되는 트랜잭션의 유형을 파악한다.
4. 용량과 트랜잭션의 유형에 따라 반정규화를 수행
5. 이력모델의 조정, PK/FK조정, 슈퍼/서브타입 조정
6. 성능관점에서 데이터 모델을 검증한다.

기본적으로 데이터는 속성간의 함수종속성에 근거하여 정규화되어야 함. 정규화는 선택이 아니라 필수사항

[정규화 후 용량,트랜잭션 파악 후 반정규화 시행]

3. 함수적종속성

데이터들이 어떤 기준 값에 의해 종속되는 현상

02. 정규화와 성능

1. 정규화

정해진 규칙에 의거 규칙화 하는 작업
반복적인 데이터를 분리하고 각 데이터가 종속된 테이블에 적절하게 배치되도록 하는 것

-1차 정규화
: 중복된 속성을 다른 테이블로 뺌, 1:M관계 형성
-2차 정규화
: 기본키가 복합키일 때, 의존컬럼이 있으면 제거
-3차 정규화
: 기본키에 의존하지않고 일반컬럼에 의존하는컬럼 제거

03. 반정규화와 성능

1. 반정규화

정규화된 엔터티, 속성, 관계에 대해 시스템의 성능향상과 개발과 운영의 단순화를 위해 중복, 통합, 분리 등을 수행하는 데이터 모델링의 기법
정규화시 입력/수정/삭제 성능이 향상
반정규화시 조인 성능이 향상

2. 반정규화 절차

1. 반정규화 대상조사(범위처리빈도수, 범위, 통계성)
2. 다른 방법유도 검토(뷰, 클러스터링, 인덱스 조정)
3. 반정규화 적용(테이블, 속성, 관계 반정규화)

3. 반정규화 대상조사

1. 자주 사용되는 테이블에 접근하는 프로세스의 수가 많고 항상 일정한 범위만을 조회하는 경우
2. 테이블에 대량의 데이터가 있고 대량의 데이터 범위를 자주 처리하는 경우에 처리범위를 일정하게 줄이지 않으면 성능을 보장할 수 없는 경우
3. 통계성 프로세스에 의해 통계 정보를 필요로 할 때 별도의 통계테이블을 생성
4. 테이블에 지나치게 많은 조인이 걸려 데이터를 조회하는 작업이 기술적으로 어려울 경우

4. 다른 방법유도 검토

1. 지나치게 많은 조인이 걸려 데이터를 조회하는 작업이 기술적으로 어려울 경우 VIEW를 사용
2. 대량의 데이터처리나 부분처리에 의해 성능이 저하되는 경우 클러스터링을 적용하거나 인덱스를 조정함
3. 대량의 데이터는 PK의 성격에 따라 부분적인 테이블로 분리할 수 있다. (파티셔닝 기법)
4. 응용 애플리케이션에서 로직을 구사하는 방법을 변경함으로써 성능을 향상시킴

5. 테이블 반정규화

5-1. 테이블 병합 (1:1관계, 1:M관계, 슈퍼/서브타입)
1. 1:1관계를 통합하여 성능향상
2. 1:M관계를 통합하여 성능향상
3. 슈퍼/서브 관계를 통합하여 성능향상

5-2. 테이블분할 (수직분할, 수평분할)
1. 칼럼단위 테이블을 디스크 I/O를 분산처리하기 위해 테이블을 1:1로 분리하여 성능향상
2. 로우단위로 집중 발생되는 트랜잭션을 분석하여 디스크 I/O 및 데이터 접근의 효율성을 높여 성능을 향상하기 위해 로우단위로 테이블을 쪼갬

5-3. 테이블추가 (중복, 통계, 이력, 부분테이블 추가)
1. 다른 업무이거나 서버가 다른 경우 동일한 테이블구조를 중복하여 원격조인을 제거하여 성능 향상
2. SUM, AVG 등을 미리 수행하여 계산해 둠으로써 조회 시 성능을 향상
3. 이력테이블 중에서 마스터 테이블에 존재하는 레코드를 중복하여 이력테이블에 존재시켜 성능 향상
4. 하나의 테이블의 전체 칼럼 중 자주 이용하는 집중화된 칼럼들이 있을 때 디스크 I/O를 줄이기 위해 해당 칼럼들을 모아놓은 별도의 반정규화된 테이블을 생성

6. 칼럼 반정규화

- 중복칼럼 추가
: 조인에 의해 처리할 때 성능저하를 예방하기 위해 중복된 칼럼을 위치시킴
- 파생칼럼 추가
: 트랜잭션이 처리되는 시점에 계산에의해 발생되는 성능저하를 예방하기위해 미리 값을 계산하여 칼럼에 보관
- 이력테이블 칼럼추가
: 대량의 이력데이터를 처리할 때 불특정 날 조회나 최근 값을 조회할 때 나타날 수 있는 성능저하를 예방하기 위해 이력테이블에 기능성 칼럼
(최근값 여부, 시작과 종료일자 등)을 추가함
- 응용시스템 오작동을 위한 칼럼 추가
: 업무적으로는 의미가 없지만 원래 값으로 복구하기 원하는 경우 이전 데이터를 임시적으로 중복하여 보관하는 기법

7. 관계 반정규화

- 중복관계 추가
: 데이터를 처리하기 위한 여러 경로를 거쳐 조인이 가능하지만, 이 때 발생할 수 있는 성능저하를 예방하기 위해 추가적인 관계를 맺는 방법

8. 반정규화의 대상에 대해 다른 방법으로 처리

- 지나치게 많은 조인이 걸려 데이터를 조회하는 작업이 기술적으로 어려울 경우 뷰(View)를 사용하여 해결
- 대량의 데이터처리나 부분처리에 의해 성능이 저하되는 경우에 클러스터링을 적용하거나 인덱스를 조정함으로써 성능을 향상
- 대량의 데이터는 PK의 성격에 따라 부분적인 테이블로 분리 할 수 있다. 파티셔닝으로 성능저하 방지
 * 파리셔닝: 논리적으로는 하나의 테이블이지만 물리적으로 여러개 테이블로 분리하여 데이터 관리하는 기법
- 응용 애플리케이션에서 로직을 구사하는 방법을 변경함으로써 성능 향상

9.
- 로우 체이닝
: 로우의 길이가 너무 길어서 데이터 블록 하나에 데이터가 모두 저장되지 않고 두 개 이상의 블록에 걸쳐 하나의 로우가 저장되어 있는 형태
- 로우 마이그레이션
: 데이터블록에서 수정이 발생하면 수정된 데이터를 해당 데이터 블록에서 저장하지 못하고 다른 블록의 빈 공간을 찾아 저장하는 방식

로우 체이닝과 로우 마이그레이션이 발생하여 많은 블록에 데이터가 저장되면 DB 메모리에서 디스크 I/O가 발생할 때 많은 I/O가 발생하여 성능저하 발생
트랜잭션을 분석하여 적절하게 1:1관계로 분리함으로써 성능향상이 가능하도록 해야함

10. PK에 의해 테이블을 분할하는 방법(파티셔닝)
- RANGE PARTITION
: 대상 테이블이 날짜 또는 숫자값으로 분리가 가능하고 각 영역별로 트랜잭션이 분리되는 경우

- LIST PARTITION
: 지점, 사업소 등 핵심적인 코드값으로 PK가 구성되어 있고 대량의 데이터가 있는 테이블의 경우
- HASH PARTITION
: 지정된 HASH 조건에 따라 해시 알고리즘이 적용되어 테이블이 분리

11. 테이블에 대한 수평/수직분할의 절차
1. 데이터 모델링을 완성한다.
2. DB 용량산정을 한다.
3. 대량 데이터가 처리되는 테이블에 대해 트랜잭션 처리 패턴을 분석한다.
4. 칼럼 단위로 집중화된 처리가 발생하는지, 로우 단위로 집중화된 처리가 발생하는지 분석하여 집중화된 단위로 테이블을 분리하는 것을 검토한다.

12. 슈퍼/서브 타입
- 슈퍼/서브 타입 모델
: 업무를 구성하는 데이터의 특징을 공통과 차이점의 특징을 고려하여 효과적 표현

- 슈퍼/서브 타입 데이터 모델의 변환기술
1. 개별로 발생되는 트랜잭션에 대해서는 개별 테이블로 구성(OneToOne Type)
2. 슈퍼타입+서브타입에 대해 발생되는 트랜잭션에 대해서는 슈퍼+서브타입 테이블로 구성(Plus Type)
3. 전체를 하나로 묶어 트랜잭션이 발생할 때는 하나의 테이블로 구성(Single Type, All in One Type)

13. 인덱스 특성을 고려한 PK/FK DB 성능향상
- 인덱스의 특징은 여러 개의 속성이 하나의 인덱스로 구성되어 있을 때 앞쪽에 '=' 다음에 범위조회 'BETWEEN' '<>' 가 오면 효율이 좋아진다.
- FK생성여부와 상관 없이 조인기능을 향상시키기위한 인덱스를 생성해주면 좋다.

04. 분산 데이터베이스와 성능

1. 분산 DB

여러 곳으로 분산되어있는 DB를 하나의 가상 시스템으로 사용할 수 있도록 한 DB
논리적으로 동일한 시스템에 속하지만, 컴퓨터 네트워크를 통해 물리적으로 분산되어 있는 데이터집합

2. 분산 DB를 만족하기 위한 6가지 투명성
- 분할 투명성(단편화)
: 하나의 논리적 Relation이 여러 단편으로 분할되어 각 사본이 여러 site에 저장
- 위치 투명성
: 사용하려는 데이터의 저장 장소 명시 불필요, 위치정보가 시스템 카탈로그에 유지
- 지역사상 투명성
: 지역 DBMS와 물리적 DB 사이의 Mapping 보장
- 중복 투명성
: DB 객체가 여러 stie에 중복 되어 있는지 알 필요가 없는 성질
- 장애 투명성
: 구성요소의 장애에 무관한 트랜잭션의 원자성 유지
- 병행 투명성
: 다수 트랜잭션 동시 수행시 결과의 일관성 유지, TimeStamp, 분산 2단계 Locking 이용

3. 분산 DB 장-단점
- 장점
: 지역 자치성, 신뢰성 가용성, 효용성 융통성, 빠른 응답속도, 비용절감, 각 지역 사용자 요구 수용
- 단점
: 비용증가, 오류의 잠재성 증대, 설계 관리의 복잡성, 불규칙한 응답속도, 통제의 어려움, 데이터 무결성 위협

4. 분산 DB 적용 기법

4-1. 테이블 위치 분산
: 설계된 테이블을 본사와 지사단위로 분산

4-2. 테이블 분할 분산 : 각각의 테이블을 쪼개어 분산
- 수평분할 : 로우 단위로 분리
- 수직분할 : 칼럼 단위로 분리

4-3. 테이블 복제 분산
: 동일한 테이블을 다른 지역이나 서버에서 동시에 생성하여 관리하는 유형

-부분복제
: 마스터 DB에서 테이블의 일부의 내용만 다른 지역이나 서버에 위치
-광역복제
: 마스터 DB 테이블의 내용을 각 지역이나 서버에 존재

4-4. 테이블 요약 분산
: 지역 간에 또는 서버 간에 데이터가 비슷하지만 서로 다른 유형으로 존재하는 경우
-분석요약
: 동일한 테이블 구조를 가지고 있으면서 분산되어 있는

동일한 내용의 데이터를 이용하여 통합된 데이터를 산출하는 방식
-통합요약
: 분산되어 있는 다른 내용의 데이터를 이용하여 통합된 데이터를 산출하는 방식

5. 분산 DB 설계를 고려해야 하는 경우

1. 성능이 중요한 사이트
2. 공통코드, 기준정보, 마스터 데이터의 성능향상
3. 실시간 동기화가 요구되지 않는 경우, 거의 실시간의 업무적인 특징을 가지고 있는 경우(?)
4. 특정 서버에 부하가 집중되어 부하를 분산
5. 백업 사이트 구성하는 경우

*** * * * * * * * * * 2과목 * * * * * * * * * * ***

1장. SQL 기본

01. 관계형 데이터베이스 개요

1. DB

특정 기업이나 조직 또는 개인이 필요에 의해
데이터를 일정한 형태로 저장해 놓은 것

2. DBMS

DB의 양이 많아지고 여러 사람들이 동시에 사용하게 되면서 효율적인 데이터 관리가 필요해짐
중복성을 없애고 동시처리를 할 수 있는 소프트웨어

3. 관계형 데이터베이스

데이터를 행과 열로 이루어진 테이블 형태로 저장하는 것으로 테이블이 다른 테이블과 관계를 맺고 있는 것
데이터의 중복 저장을 피하고, 동시 관리를 할 수 있고, 데이터 공유 및 조작을 할 수 있는 것
(ex, 엑셀 표 데이터 저장)

4. SQL

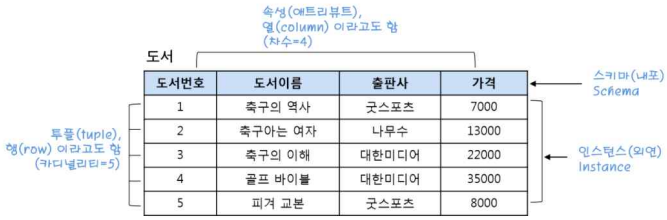
4-1. SQL 개념
관계형 DB에서 데이터 정의, 조작, 제어를 위해 사용하는 언어

- DDL : CREATE, ALTER, DROP, RENAME
- DCL : GRANT, REVOKE
- DML : SELETE, INSERT, DELETE, UPDATE
- TCL : COMMIT, ROLLBACK

4-2. SQL문 실행순서
1) 파싱(Parsing) : SQL의 문법을 확인하고 구문분석. 구문분석한 SQL문은 Libray Cache에 저장
2) 실행(Exceution) : 옵티마이저가 수립한 실행 계획에 따라 SQL 실행
3) 인출(Fetch) : 데이터를 읽어서 전송

5. TABLE

5-1. TABLE 개념
데이터를 저장하는 객체(Object)로서 관계형 데이터베이스의 기본단위
- 세로 방향: 칼럼(Column), 열



- 가로 방향: 로우(ROW), 행, 튜플, 스키마, 인스턴스
- 칼럼과 행이 겹치는 하나의 공간: 필드

5-2. TABLE 관계 용어들
- 정규화
  : 데이터의 정합성 확보와 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 기본키로 사용되고 있는 관계를 연결하는 칼럼

02. DDL

1. 데이터 유형

- CHAR(s)
  : 고정 길이 문자열 정보 최대 길이 만큼 공간 채움 'AA' = 'AA '
- VARCHAR2(s)

: 가변 길이 문자열 정보 할당된
변수값의 바이트만 적용 'AA' != 'AA '
- NUMBER
: 정수, 실수 등 숫자 정보
- DATE
: 날짜와 시각 정보

2. CREATE TABLE

2-1. 테이블 생성시 주의사항
1) 테이블명은 객체를 의미할 수 있는 적절한 이름을 사용한다. 가능한 단수형을 권고한다.
2) 테이블명은 다른 테이블과 중복되지 않아야 한다.
3) 테이블내에서 컬럼명이 중복되게 지정될 수 없다.
4) 테이블 이름을 지정하고 각 컬럼들은 괄호 "()" 로 묶어 지정한다.
5) 각 컬럼들은 콤마 "," 로 구분되고, 테이블 생성문의 끝은 항상 세미콜론 ";" 으로 끝난다.
6) 칼럼에 대해서는 다른 테이블까지 고려하여 테이터베이스내에서는 일관성있게 사용하는 것이 좋다. (데이터 표준화 관점)
7) 컬럼 뒤에 테이터 유형은 꼭 지정되어야한다.
8) 테이블명과 칼럼명은 반드시 문자로 시작해야 하고, 벤더별로 길이에 대한 한계가 있다.
9) 벤더에서 사전에 정의한 예약어는 쓸 수 없다.
10) A-Z, a-z, 0-9, _, $, # 문자만 허용된다.

2-2. 제약조건
1. PRIMARY KEY(기본키) : 기본키 정의
2. UNIQUE KEY(고유키) : 고유키 정의
3. NOT NULL : NULL 값 입력금지
4. CHECK : 입력 값 범위 제한
5. FOREIGN KEY(외래키) : 외래키 정의

2-3. DDL 문법
[테이블 생성]
CREATE TABLE 테이블명 (
칼럼명 데이터타입 [NULL여부 옵션])

[테이블 구조 확인]
: Oracle [describe 테이블명] & [desc 테이블명]
예) describe test01;, desc test01;
: SQL ['sp_help "dbo.테이블명"']
예) 'sp_help "dbo.test01"'

[컬럼 추가]
ALTER TABLE 테이블명
ADD 칼럼명 데이터 유형;

[컬럼 삭제]
ALTER TABLE 테이블명
DROP COLUMN 칼럼명;

[테이블 정의변경]
문장 하나당 하나의 정의 변경 가능
ALTER TABLE 테이블명
MODIFY (칼럼명 데이터유형 DEFAULT식 NOT NULL); 오라클
ALTER (칼럼명 데이터유형 DEFAULT식 NOT NULL); SQL Server

[제약조건 삭제]
ALTER TABLE 테이블명
DROP CONSTRAINT 조건명;

[제약조건 추가]
ALTER TABLE 테이블명
ADD CONSTRAINT 조건명 조건 (칼럼명); 조건 추가

[테이블 이름 변경]
RENAME 테이블명 TO 변경후테이블명;

[테이블 삭제]
DROP TABLE 테이블명

3. 부모와 자식테이블 관계

두 테이블을 연결해서 pk를 가지고 있는 쪽의 값을 삭제하면 연결된 값이 동시에 삭제되게 하는 옵션

- references 테이블명(컬럼명) on delete cascade
: 연결된 자식의 컬럼도 함께 삭제한다
B REFERENCES A(a) on delete cascade

- references 테이블명(컬럼명) on delete set null
: 부모 컬럼은 지워지고 연결된 컬럼은 null로 셋팅
B REFERENCES A(a) on delete set null

- references 테이블명(컬럼명) on delete set default
: 부모 컬럼은 지워지고 연결된 컬럼은 default로 셋팅

- references 테이블명(컬럼명) on delete restrict
: 자식 테이블에 PK값이 없는 경우에만 부모 컬럼 삭제

automatic	부모 테이블에 pk가 없는 경우, 부모 테이블에 pk 생성 후 자식 입력
set null	부모 테이블에 pk가 없는 경우, 자식 fk를 null값으로 처리
set default	부모 테이블에 pk가 없는 경우, 자식 fk를 기본값으로 처리
dependent	부모 테이블에 pk가 존재할때만 자식 입력 허용
no action	참조무결성을 위반하는 입력 액션을 취하지 않음

-delete from 테이블명 : 로그는 남겨두고 데이터 삭제
-truncate table 테이블명 : 데이터 삭제후 저장 공간을 재사용 하도록 해제
-drop table 테이블명 : 테이블구조 완적히 삭제

drop	truncate	delete
ddl	ddl (일부 dml성격)	dml
rollback 불가능	rollback 불가능	commit 이전 rollback 가능
auto commit	auto commit	사용자 commit
테이블의 정의 자체를 완전히 삭제함	테이블을 최초 생성된 초기상태로 만듬	데이터만 삭제

03. DML

1. DML

DDL AUTO COMMIT ,DML의 경우 COMMIT 입력

[[데이터 입력]
INSERT INTO PLAYER (PLAYER) VALUES ('PJS');

[데이터 수정]
UPDATE PLAYER SET BACK_NO = 60;

[데이터 삭제]
DELETE FROM PLAYER;

[데이터 읽기]]
SELECT PLAYER_ID FROM PLAYER;
SELECT DISTINCT POSITION 시 구분값만 출력 ex)GK, FW, DF, MF
SELECT PLAYER AS "선수명" FROM PLAYER;

와일드카드
* : 모든
% : 모든
- : 한 글자

연결 연산자
문자와 문자 연결 : ||(Oracle), +(SQL Server)
SELECT PLAYER_NAME + '선수' "정보"

04. TCL

1. 트랜잭션

논리적인 집합단위로, 그 단위별로 작업을 진행 하는 것 밀접히 관련되어 분리될 수 없는 1개 이상의 DB 조작

COMMIT : 올바르게 반영된 데이터를 DB에 반영
ROLLBACK : 트랜잭션 시작 이전의 상태로 되돌림
SAVEPOINT : 저장 지점

2. 트랜잭션의 특성

특성	설명
원자성	트랜잭션에서 정의된 연산자들은 모두 성공적으로 실행되던지 아니면 전혀 실행되지 않은 상태로 남아 있어야 한다.
일관성	트랜잭션이 실행 되기 전의 DB내용이 잘못 되어 있지 않다면 트랙잭션이 실행된 이후에도 DB내용에 잘못이 있으면 안된다.
고립성	트랜잭션이 실행되는 도중에 다른 트랜잭션의 영향을 받아 잘못된 결과를 만들어서는 안된다.
지속성	트랜잭션이 성공적으로 수행되면 그 트랜잭션이 갱신한 DB의 내용은 영구적으로 저장된다.

- Dirty Read
: 다른 트랜잭션에 의해 수정되었지만 아직 커밋되지 않은 데이터를 읽는 것을 말한다.
- isolation
: 트랜잭션이 실행되는 도중에 다른 트랜잭션의 영향을

받아 잘못된 결과를 만들어서는 안된다.

- Non-Repeatable Read
: 한 트랜잭션 내에서 같은 쿼리를 두 번 수행 했는데, 그 사이에 다른 트랜잭션이 값을 수정 또는 삭제하는 바람에 두 쿼리 결과가 다르게 나타는 현상
- Phantom Read
: 한 트랜잭션 내에서 같은 쿼리를 두 번 수행했는데 첫 번째 쿼리에서 없던 유령 레코드가 두 번째 쿼리에서 나타나는 현상

('' -> sql server에선 문자''로 인식
'' -> oracle에선 null로 인식)

04. 연산자

1. 연산자의 종류

-BETWEEN a AND b : a와 b 값 사이에 있으면 됨
-IN (list) : 리스트에 있는 값중 어느 하나라도 일치
-LIKE '비교문자열' : 비교문자열과 형태가 일치
-IS NULL : NULL 값인 경우
-NOT IN (list) : list의 값과 일치하지 않는다
-IS NOT NULL : NULL 값을 갖지 않는다.

2. 연산자 우선순위
()->NOT->비교연산자->AND->OR

NULL 값과의 수치연산은 NULL 값을 리턴한다.
NULL 값과의 비교연산은 거짓(FALSE)를 리턴한다.

ROWNUM
: 원하는 만큼의 행을 가져올 때 사용(Or)
TOP()
: 상위 랭크와 상위 랭크 동일한 숫자를 뽑을 때 사용
ex) 급여가 높은 2명 내림차순 출력, 같은 급여 받는사원 출력
```
SELECT TOP(2) WITH TIES ENAME,SAL
FROM EMP
ORDER BY SAL DESC;
```

05. 함수

1. 단일행 함수

- SELECT, WHERE, ORDER BY 절에서 사용 가능
- 행에 개별적 조작
- 여러 인자가 있어도 결과는 1개만 출력
- 함수 인자에 상수 변수 표현식 , 사용 가능
- 함수 중첩 가능

5-1. 문자형 함수
LOWER(문자열) : 문자열을 소문자로
UPPER(문자열) : 문자열을 대문자로
ASCII(문자) : ASCII 문자의 값 반환
CHR(ASCII번호) : ASCII 값에 해당하는 문자 반환
CONCAT(1,2) : 1, 2 문자열 를 연결
CONCAT('RDBMS',' SQL') -> 'RDBMS SQL'
SUBSTR : m n 문자열 중 위치에서 개의 문자 반환
SUBSTR('SQL Expert',5,3) -> 'Exp'
LENGTH(문자열) : 문자열 길이를 숫자 값으로 반환

5-2. 숫자형함수
SIGN(n) : 숫자가 양수면1 음수면-1 0이면 0 반환
MOD : 숫자1을 숫자2로 나누어 나머지 반환
CEIL/CEILING(n) : 크거나 같은 최소 정수 반환
FLOOR(n) : 작거나 같은 최대 정수 리턴

ROUND(38.5235,3) -> 38.524
ROUND(38.5235) -> 39
TRUNC(38.5235,3) -> 38.523
TRUNC(38.5235) -> 38

5-3. 날짜형함수
SYSDATE/GETDATE()
: 현재날짜와 시각 출력
EXTRACT/DATEPART
: 날짜에서 데이터 출력
TO_NUMBER(TO_CHAR(d,'YYYY'))
: 연도를 숫자로 출력

sysDate +1 : 1일
sysDate + 1/24 : 1시간
sysDate + 1/24/6 : 10분
sysDate + 1/24/60 : 1분
Turncate(sysDate) + 23/24 : 매일 23시

5-4. NULL 관련 함수
NVL(식1,식2) / ISNULL(식1,식2)
: 식1의 값이 NULL 이면 식2 출력
NULLIF(식1,식2)
: 식1=식2 : NULL / 식1!=식2 식1을 출력
COALESCE(식1,식2)
: 임의의 개수표현식에서 NULL이 아닌 최초의 표현식,

모두 NULL이면 NULL 반환
ex)COALESCE(NULL,NULL,'abc') -> 'abc'

2. 다중행 함수

집계 함수
- 여러 행들의 그룹이 모여서 그룹당 단 하나의 결과를 돌려주는 함수
- GROUP BY 절은 행들을 소그룹화 한다.
- SELECT, HAVING, ORDER BY 절에 사용 가능
- ALL : Default 옵션
- DISTINCT : 같은 값을 하나의 데이터로 간주 옵션

COUNT(*) : NULL 포함 행의 수
COUNT(표현식) : NULL 제외 행의 수
SUM, AVG : NULL 제외 합계, 평균 연산
STDDEV : 표준 편차
VARIAN : 분산
MAX, MIN : 최대값, 최소값

05. GROUP BY, HAVING 절의 특징

1. GROUP BY 절을 통해 소그룹별 기준을 정한 후, SELECT 절에 집계 함수를 사용
2. 집계 함수의 통계 정보는 NULL 값을 가진 행을 제외하고 수행
3. GROUP BY 절에서는 ALIAS 사용 불가
4. 집계 함수는 WHERE 절에 올 수 없음
5. HAVING 절에는 집계함수를 이용하여 조건 표시o
6. HAVING 절은 일반적으로 GROUP BY 뒤에 위치
7. group by로 집계하지 않으면 select 된 컬럼이 아니여도 테이블이 가지고 있는 컬럼으로 order by가 가능
8. GROUP BY 및 집계함수가 없어도 HAVING절 사용 가능

1. 추가 문법

3-1. REPLACE ('A', 'B', 'C')
A문장에 B단어를 찾아 C단어로 바꿔라!
REPLACE(C1,CHR(10))
C1에 CHR(10)을 찾아 공백으로 바꿔라(=삭제해라)

3-2. CASE WHEN THAN ELSE
CASE WHEN '조건' THAN '조건에 맞을 때 출력데이터'
ELSE '조건에 맞지않을 때 출력데이터' END
CASE WHEN RANK="사원" THAN 2000
WHEN RANK="대리" THAN 3000
ELSE 4000 END

CASE WHEN LOC = 'a' THEN 'b'
= CASE LOC WHEN 'a' THEN 'b'

3-3.
EXISTS ()
()안의 조건에 맞는 값이 있어야지만 문장 실행

NOT EXISTS ()
()안의 조건에 맞는 값이 없는 경우 문장 실행

05. ORDER BY

1. 특징

- SQL 문장으로 조회된 데이터들을 다양한 목적에 맞게 특정한 칼럼을 기준으로 정렬하여 출력하는데 사용
- ORDER BY 절에 칼럼명 대신 ALIAS 명이나 칼럼 순서를 나타내는 정수도 사용 가능
- DEFAULT 값으로 오름차순(ASC)이 적용되며 DESC 옵션을 통해 내림차순으로 정렬이 가능
- SQL 문장의 제일 마지막에 위치함
- SELECT 절에서 정의하지 않은 칼럼 사용 가능

Oracle에서는 NULL을 가장 큰 값으로 취급하며 SQL Server에서는 NULL을 가장 작은 값으로 취급

2. SELECT 문장 실행 순서

FROM -> WHERE -> GROUP BY -> HAVING -> SELECT -> ORDER BY

06. JOIN

1. JOIN

- 두 개 이상의 테이블들을 연결 또는 결합하여 데이터를 출력하는 것
- 일반적으로 행들은 PK나 FK 값의 연관에 의해 JOIN이 성립됨
- 어떤 경우에는 PK, FK 관계가 없어도 논리적인 값들의 연관만으로 JOIN이 성립가능
-5가지 테이블을 JOIN 하기 위해서는 최소 4번의 JOIN 과정이 필요함

2. EQUI JOIN

- 2개의 테이블 간에 칼럼 값들이 서로 정확하게 일치하는 경우에 사용,
대부분 PK, FK의 관계를 기반으로 함
(where에 key값 연결, and에도)

3.NON EQUI JOIN

- 2개의 테이블 간에 칼럼 값들이 서로 정확하게 일치하지 않는 경우에 사용
'=' 연산자가 아닌 BETWEEN, >, <= 등 연산자 사용

2장. SQL 활용

01. JOIN 연산

1. 집합 연산자

두 개 이상의 테이블에서 조인을 사용하지 않고 연관된 데이터를 조회할 때 사용
SELECT 절의 칼럼 수가 동일하고, SELECT 절의 동일 위치에 존재하는 칼럼의 데이터 타입이 상호 호환할 때 사용 가능

2. 일반 집한 연산자

1. UNION : 합집합(중복 행은 1개로 처리)
2. UNION ALL : 합집합(중복 행도 표시)
3. INTERSECT : 교집합(INTERSECTION)
4. EXCEPT,MINUS : 차집합(DIFFERENCE)
5. CROSS JOIN : 곱집합(PRODUCT)

3. 순수 관계 연산자

관계형 DB를 새롭게 구현

1. SELECT -> WHERE
2. PROJECT -> SELECT
3. NATRUAL JOIN -> 다양한 JOIN
4. DIVIDE -> 사용x
{a,x}{a,y}{a,z} divdie {x,z} = {a}

4. FROM JOIN 절 형태

4-1. INNER JOIN
JOIN 조건에서 동일한 값이 있는 행만 반환,
USING이나 ON 절을 필수적으로 사용

4-2. NATURAL JOIN
두 테이블 간의 동일한 이름을 갖는 모든 칼럼들에 대해 EQUI JOIN 수행
NATURAL JOIN이 명시되면 추가로 USING, ON, WHERE 절에서 JOIN 조건을 정의할 수 없음
SQL Sever는 지원x

3. USING (칼럼명)
같은 이름을 가진 칼럼들 중에서 원하는 칼럼에 대해서만 선택적으로 EQUI JOIN을 할 수 있음
JOIN 칼럼에 대해서 ALIAS나 테이블 이름과 같은 접두사를 붙일 수 없음. SQL Server 지원x

5. CROSS JOIN
= 카다시안 곱
테이블 간의 조인조건이 없는 경우
생길 수 있는 모든 데이터 조합 (where문이 없는 경우)

6. OUTER JOIN
조인 조건에 만족하지 않는 행들도 반환

- LEFT OUTER JOIN
 : 조인 수행시 먼저 표기된 좌측 테이블에 해당하는 데이터를 읽은 후, 나중 표기된 우측 테이블에서 JOIN 대상 데이터를 읽음
- RIGHT OUTER JOIN
 : LEFT OUTER JOIN의 반대
- FULL OUTER JOIN
 : 좌측,우측 모든 데이터 읽은 후 중복데이터 삭제

02. 계층형 질의와 셀프조인

1. 계층형 질의

상하 수직관계의 구조

[START WITH … CONNECT BY (PRIOR)…]

- START WITH
 : 시작 조건
  시작 조건은 Root Node이며 Level은 1
- CONNECT BY : 연결 조건

- PRIOR
 : CONNECT BY, SELECT, WHERE 절에서 사용 가능
  *PRIOR가 붙은 쪽이 화살표를 받음
   PRIOR A = B 이면 A <- B 구조
  *부모,자식 테이블 정리 후 부모->자식이면 순방향
                자식->부모면 역방향
- NOCYCLE : 동일한 데이터가 전개되지 않음
- ORDER SIBLINGS BY
 : 계층형에 있는 가나다 정렬(들여쓰기로)
- CONNECT_BY_ISLEAF
 : 해당 데이터가 리프 데이터면 1, 그렇지 않으면 0
- CONNECT_BY_ISCYCLE
 : 해당 데이터가 조상이면 1, 아니면 0
   (CYCLE 옵션 사용했을 시만 사용 가능)
- SYS_CONNECT_BY_PATH
 : 루트 데이터부터 현재 전개할 데이터까지의 경로를 표시
- CONNECT_BY_ROOT
: 현재 전개할 데이터의 루트 데이터를 표시
  단항 연산자

2. 셀프조인

한 테이블 내에서 두 컬럼이 연관관계
반드시 테이블 별칭을 사용해야 함

3. 서브쿼리

하나의 SQL문안에 포함되어 있는 또 다른 SQL문
알려지지 않은 기준을 이용한 검색에 사용

3-1. 서브쿼리 사용시 주의 사항
 1. 서브쿼리를 괄호로 감싸서 사용
 2. 서브쿼리는 단일 행 또는 복수 행 비교 연산자와
  함께 사용 가능. 단일 행 비교 연산자는 서브쿼리의 결과가 반드시 1건 이하여야 하고 복수 행 비교 연산자는 결과 건수와 상관없다.
3. 서브쿼리에서는 ORDER BY를 사용하지 못함
4. SELECT, FROM, WHERE, HAVING, ORDER BY, INSERT-VALUES, UPDATE-SET 절에 사용 가능

3-2. 서브쿼리 종류
- 단일행 서브쿼리
 : 서브쿼리 실행 결과가 항상 1건 이하
   단일행 비교연산자와 함께 사용
   * 단일행 비교연산자 : =,<,>,<>, <=, >=
- 다중행 서브쿼리
 : 서브쿼리의 실행 결과가 여러 건의 서브쿼리 의미
   다중행 비교연산자와 함께 사용
   * 다중행 비교연산자 : IN,ALL,ANY,SOME,EXISTS
- 다중컬럼 서브쿼리
 : 서브쿼리의 실행 결과가 여러컬럼을 반환
   메인쿼리의 조건절에 여러 컬럼을 동시 비교 가능
- 스칼라 서브쿼리
 : 한 행, 한 칼럼만을 반환하는 서브쿼리

4. VIEW

테이블은 실제로 데이터를 가지고 있는 반면
뷰는 실제 데이터를 가지고 있지 않다. 가상 테이블이라고도 함

4-1. VIEW 사용 장점
 - 테이블 구조변경에 따른 영향으로부터 응용프로그램의 분리가 가능
 - 테이터 구조 단순화
 - User에게 단순한 인터페이스 제공
 - 테이블의 자료에 대한 접근 제한(보안좋음)
 - 복잡한 SQL 저장

5. 그룹 함수

5-1. ROLLUP
 : GROUP BY ROLLUP(C1,C2)
   (C1,C2), (C1,NULL), (NULL,NULL)
  GROUP BY ROLLUP((C1,C2),C3)
   (C1,C2,C3), (C1,C2,NULL), (NULL,NULL)
  GROUP BY ROLLUP((C1,C2,C3)
   (C1,C2,C3), (C1,C2,NULL), (C1,NULL,NULL)
   (NULL,NULL,NULL)

5-2. GROUPING
 : GROUP BY 할땐 없는 로우였지만
  ROLLUP이 들어가면서 생긴 로우 값 = 1
  원래 있었던 로우 = 0

5-3. GROUPING SETS
- 다양한 GROUP BY 처리 가능
- ROLLUP 및 CUBE로 구현하는 것 모두 가능
- GROUPING SETS절에 지정된 모든 그룹을 계산하고 각 그룹의 결과를 UNION ALL로 조작하는 것과 유사

GROUPING SETS (C1,C2)
(C1,NULL), (C2,NULL)
GROUPING SETS (C1,C2) GROUPING SETS(C3,C4)
(C1,C3), (C1,C4), (C2,C3), (C2,C4)
GROUPING SETS (NO,JOB) GROUPING SETS(YEAR)
(NO,YEAR), (JOB,TEAR)
GROUPING SETS((상품ID,월)) = GROUP BY(상품ID,월)

5-4. CUBE
: GROUP BY CUBE(C1,C2)
(C1,C2), (C1,NULL), (C2,NULL), (NULL,NULL)
GROUP BY CUBE((C1,C2),C3)
(C1,C2,C3), (C1,C2,NULL), (C3,NULL,NULL),
(NULL,NULL,NULL)
GROUP BY CUBE(C1,C2,C3)
(C1,C2,C3), (C1,C2,NULL), (C1,C3,NULL),
(C1,NULL,NULL), (C2,C3,NULL), (C2,NULL,NULL)
(C3,NULL,NULL), (NULL,NULL,NULL)

03. 윈도우 함수

1. 윈도우 함수

행과 행간의 관계를 정의하거나 행과 행간을 비교, 연산하는 함수

6-1. 함수1
- ROW_NUMBER () OVER

1
2
3
4

- RANK () OVER = TOP(4) WITH TIES

1
1
3
4

- DENSE_RANK() OVER

1
1
2
3

6-2. 함수2
- RANGE
: 원하는 조건에 맞는 데이터에 대한 통계 값
ex)같은 매니저 내에서 앞의 사번과 뒤의 사번
RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING
(현재 행을 기준으로 파티션 내에서 앞의 1건, 현재행, 뒤의 1건을 범위로 지정)

UNMOUNDED PRECEDING
PRECEDING
CURRENT ROW
FOLLOWING
UNMOUNDED FOLLOWING

04. DCL

1. DCL

유저 생성하고 권한을 제어할 수 있는 명령어

2. Oracle과 SQL Server의 사용자 아키텍처 차이

- Oracle
: 유저를 통해 DB에 접속을 하는 형태, ID와 PW 방식으로 인스턴스에 접속을 하고 그에 해당하는 스키마에 오브젝트 생성 등의 권한을 부여받게 됨
- SQL Server
: 인스턴스에 접속하기 위해 로그인이라는 것을 생성하게 되며, 인스턴스 내에 존재하는 다수의 DB에 연결하여 작업하기 위해 유저를 생성한 후 로그인과 유저를 매핑해 주어야 한다. Windows 인증 방식과 혼합 모드 방식이 존재함

3. 시스템 권한

사용자가 SQL 문을 실행하기 위해 필요한 적절한 권한

- GRANT : 권한 부여
[GRANT .. (ON OBJECT) TO USER]
GRANT CREATE USER TO SCOTT;
- REVOKE : 권한 취소
[REVOKE .. (ON OBJECT) FROM USER]
REVOKE CREATE TABLE FROM PJS;

- ROLE
: 유저에게 권한들을 한번에 부여하기 위해 사용하는 것
CREATE ROLE LOGIN_TABLE;
GRANT CREATE TABLE TO LOGIN_TABLE;

- WIHT GRANT OPTION
: 권한을 부여해주면 다른사람에게 권한을 줄 수 있음

- DROP USER PJS CASCADE;
CASCADE : 하위 오브젝트까지 삭제

05. 절차형 SQL

1. 절차형 SQL

SQL문의 연속적인 실행이나 조건에 따른 분기처리를 이용하여 특정 기능을 수행하는 저장 모듈을 생성
Procedure,User Defined Function,Trigger 등이 있음

2. 저장 모듈

: 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

- PL/SQL 특징
 1. Block 구조로 되어있어 각 기능별로 모듈화 가능
 2. 변수, 상수 등을 선언하여 SQL 문장 간 값을 교환
 3. IF, LOOP 등의 절차형 언어를 사용하여 절차적인 프로그램이 가능하도록 한다.
 4. DBMS 정의 에러나 사용자 정의 에러를 정의하여 사용할 수 있다.
 5. PL/SQL은 Oracle에 내장되어 있으므로 호환성 굳
 6. 응용 프로그램의 성능을 향상시킨다.
 7. Block 단위로 처리 -> 통신량을 줄일 수 있다.
 8. DCL을 실행하기 위해선
  EXECUTE IMMEDIATE '--'로 실행

- Trigger
: 특정한 테이블에 INSERT, UPDATE, DELETE와 같은 DML문이 수행되었을 때, DB에서 자동으로 동작하도록 작성된 프로그램, 사용자 호출이 아닌 DB 자동 수행

CREATE Trigger Trigger_name

- 프로시저와 트리거의 차이점
프로시저는 BEGIN~END 절 내에 COMMIT, ROLLBACK과 같은 트랜잭션 종료 명령어 사용가능, DB 트리거는 BEGIN~END 절 내에 사용 불가

3장. SQL 최적화 기본 원리

01. 옵티마이저와 실행계획

1. 옵티마이저

사용자가 질의한 SQL문에 대한 최적의 실행방법을 결정

- 규칙 옵티마이저
 : 우선순위를 가지고 실행계획을 생성
  적절한 인덱스가 존재하면 인덱스 사용하려고 함
  제일 낮은 순위 : 전체 테이블 스캔
  제일 높은 순위 : Row(행)를 활용하여 테이블 액세스

- 비용 옵티마이저
 : 가장 효율적인 계획을 선택
  풀 테이블 스캔이 효율적이라고 생각하면 풀 테이블 스캔 실행함

2. 실행계획

= 예상정보
SQL에서 요구한 사항을 처리하기 위한 절차와 방법

- 실행계획을 읽는 순서
 : 위에서 아래로, 안에서 밖으로
- 실행계획이 달라진다고해서 결과가 달라지지 않음

3. SQL 처리 흐름도

- SQL 실행계획을 시각화해서 표현한 것
- 인덱스 스캔, 전체 테이블 스캔 등 액세스 기법 표현
 성능적인 측면도 표현
- 인덱스 범위 스캔은 결과 건수만큼 반환되지만,
 결과가 없으면 한건도 반환하지 않을 수 있다.

02. 인덱스

1. 인덱스

- 데이터를 찾을 때 좀더 빠르게 찾을 수 있는 하는 기능
- 컬럼의 값을 주어진 순서로 정렬해서 보관
- 조회 성능은 최적화 시키지만 데이터의 insert, delete, update 등의 DML 처리 성능을 저하
- 인덱스가 존재하는 상황에서 데이터를 입력하면 매번 인덱스 정렬이 일어나므로 대량의 데이터 삽입시엔 모든 인덱스를 제거한 후, 데이터 삽입이 끝난 후 인덱스 다시 생성
- 컬럼들의 순서는 데이터 조회 시 성능적인 관점에서 매우 중요

- B-TREE 인덱스
 : 브랜치 블록과 리프 블록으로 구성되며, 브랜치 블록은 분기를 목적으로 하고 리프블록은 인덱스를 구성하는 컬럼의 값으로 정렬된다. 일반적으로 OLTP 시스템 환경에서 가장 많이 사용한다.
  테이블 내 데이터 중 10%이하 데이터 검색시 유리
  일치 및 범위 검색에 적절한 구조

- CLUSTERED 인덱스
 : 인덱스의 리프페이지가 곧 데이터페이지이며, 리프페이지의 모든 데이터는 인덱스 키 컬럼순으로 물리적으로 정렬되어 저장된다.

- BITMAP 인덱스
 : 시스템에서 사용될 질의를 시스템 구현 시에 모두 알 수 없는 경우인 DW 및 AD-HOC 질의 환경을 위해서 설계되었으며, 하나의 인덱스 키 엔트리가 많은 행에 대한 포인터를 저장하고 있는 구조이다.

2. 조인

2-1. NL Join(Nested Loop)
- 랜덤 액세스 방식으로 데이터 읽음
- 목록처리 업무에 많이 사용 됨
- 조인 컬럼에 적당한 인덱스가 있어서 자연조인이 효율적일 때 유용
- Driving Table의 조인 데이터 양이 큰 영향을 줌
- Non-EQUI JOIN조건 사용

2-2. Sort Merge Join
- NL에서 부담되던 넓은 범위 데이터 처리할 때 이용
- 조인 컬럼을 기준으로 데이터를 정렬하여 조인 수행
- DW등의 데이터 집계 업무에 많이 사용됨
- 정렬한 데이터 많으면 임시 디스크를 사용하기 때문에 성능이 떨어질 수 있다.
- Non-EQUI JOIN조건 사용

2-3. Hash Join
- DW등의 데이터 집계 업무에 많이 사용됨
- CPU 작업 위주로 데이터 처리함
- 소트머지조인을 하기에 두 테이블이 너무 커서 소트 부하가 심할 때 유용
- EQUI JOIN에서만 동작
- Hash Table을 메모리에 생성 후 조인 작업 수행 ->Hash Table의 크기가 메모리에 적재할 수 있는 크기보다 더 커지면 임시 디스크에 저장함 -> 행의 수가 작은 테이블을 선행 테이블로 사용