제 1장 데이터 모델링의 이해

제 1절 데이터 모델의 이해

모델링의 특징

1) 추상화: 현실세계를 일정한 형식에 맞추어 표현을 한다는 의미로 정의할 수 있다. 즉, 다양한 현상을 일정한 양식인 표기법에 의해 표현한다는 것이다.
2) 단순화: 복잡한 현실세계를 약속된 규약에 의해 제한된 표기법이나 언어로 표현하여 쉽게 이해할 수 있도록 하는 개념을 의미한다.
3) 명확화: 누구나 이해하기 쉽게 하기 위해 대상에 대한 애매모호함을 제거하고 정확하게 현상을 기술하는 것을 의미한다.

데이터 모델링의 중요성 및 유의점

1) 중복: 데이터베이스가 여러 장소에 같은 정보를 저장하는 잘못을 하지 않도록 한다.
2) 비유연성: 데이터 모델링은 데이터 혹은 프로세스의 작은 변화가 애플리케이션과 데이터베이스에 중대한 변화를 일으킬 수 있는 가능성을 줄인다.
3) 비일관성: 데이터의 중복이 없더라도 비일관성은 존재한다. 데이터 모델링을 할 때 데이터와 데이터간 상호 연관 관계에 대한 명확한 정의는 이러한 위험을 사전에 예방할 수 있도록 해준다.

데이터 모델링의 3단계 진행

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

데이터독립성 요소

- 외부스키마: 개개 사용자 단계로서 개개 사용자가 보는 개인적 DB스키마(사용자 관점)
- 개념스키마: 개념단계 하나의 개념적 스키마로 구성 모든 사용자 관점을 통합한 조직 전체의 DB를 기술하는 것, DB에 저장되는 데이터와 그들간의 관계를 표현하는 스키마(통합관점)
- 내부스키마: DB가 물리적으로 저장된 형식, 물리적 장치에서 데이터가 실제적으로 저장되는 방법을 표현하는 스키마(물리적 저장구조)

두 영역의 데이터 독립성

- 논리적 독립성: 개념 스키마가 변경되어도 외부 스키마에는 영향을 미치지 않도록 지원하는 것.
논리적 구조가 변경되어도 응용 프로그램에 영향없음.
- 물리적 독립성: 내부스키마가 변경되어도 외부/개념스키마는 영향을 받지 않도록 지원하는 것.
저장장치의 구조변경은 응용프로그램과 개념스키마에 영향 없음.

사상

외부적/개념적 사상(논리적 사상): 사용자가 접근하는 형식에 따라 다른 타입의 필드를 가질 수 있음. 개념적 뷰의 필드 타입은 변화가 없음.
개념적/내부적 사상(물리적 사상): 저장된 데이터베이스 구조가 바뀐다면 개념적/내부적 사상이 바뀌어야 함. 그래야 개념적 스키마가 그대로 남아있게 됨.

데이터 모델링의 세 가지 요소

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

ERD 작업순서

1. 엔터티를 그린다.
2. 엔터티를 적절하게 배치한다.
3. 엔터티간 관계를 설정한다.
4. 관계명을 기술한다.
5. 관계의 참여도를 기술한다.
6. 관계의 필수여부를 기술한다.

좋은 데이터 모델의 요소

완전성
중복배제
업무규칙
데이터 재사용
의사소통
통합성

제 2절 엔터티(Entity)

엔터티의 특징
가. 업무에서 필요로 하는 정보
나. 식별이 가능해야 함
다. 인스턴스의 집합(두 개 이상)
라. 업무프로세스에 의해 이용
마. 속성을 포함
바. 관계의 존재(최소 한 개 이상의 관계)

엔터티의 분류
가. 유무형에 따른 분류
유형엔터티: 물리적인 형태가 있고 안정적이며 지속적으로 활용되는 엔터티(ex. 사원, 물품, 강사 등)
개념엔터티: 물리적인 형태는 존재하지 않고 관리해야 할 개념적 정보로 구분이 되는 엔터티(ex. 조직, 보험상품 등)

나. 발생시점에 따른 분류
기본엔터티: 업무에 원래 존재하는 정보로서 다른 엔터티와 관계에 의해 생성되지 않고 독립적으로 생성이 가능하고 자신은 타 엔터티의 부모의 역할을 하게 된다.
(ex. 사원, 부서, 고객, 상품, 자재 등)
중심엔터티: 기본엔터티로부터 발생되고 그 업무에 있어서 중심적인 역할을 한다.
(ex. 계약, 사고, 예금원장, 청구, 주문, 매출 등)
행위엔터티: 두 개 이상의 부모엔터티로부터 발생되고 자주 내용이 바뀌거나 데이터양이 증가된다.
(ex. 주문목록, 사원변경이력)

엔터티의 명명
1. 가능하면 현업업무에서 사용하는 용어를 사용한다.
2. 가능하면 약어를 사용하지 않는다.
3. 단수명사를 사용한다
4. 모든 엔터티에서 유일하게 이름이 부여되어야 한다.
5. 엔터티 생성의미대로 이름을 부여한다.

제 3절 속성

속성의 개념
- 업무에서 필요로 한다.
- 의미상 더 이상 분리되지 않는다.
- 엔터티를 설명하고 인스턴스의 구성요소가 된다.

엔터티, 인스턴스, 속성, 속성값의 관계
- 한 개의 엔터티는 두 개 이상의 인스턴스 집합이어야 한다.
- 한 개의 엔터티는 두 개 이상의 속성을 갖는다.
- 한 개의 속성은 한 개의 속성값을 갖는다.

속성의 분류
가. 속성의 특성에 따른 분류
1) 기본속성: 업무로부터 추출한 모든 속성이 여기에 해당하며 엔터티에 가장 일반적이고 많은 속성을 차지한다. 다른 속성을 계산하거나 영향을 받아 생성된 속성을 제외한 모든 속성은 기본속성이다.
2) 설계속성: 업무상 필요한 데이터 이외에 데이터 모델링을 위해, 업무를 규칙화 하기 위해 속성을 새로 만들거나 변형하여 정의하는 속성이다. (코드성 속성, 일변번호)
3) 파생속성: 다른 속성에 영향을 받아 발생하는 속성으로서 보통 계산된 값들이 이에 해당한다.

나. 엔터티 구성방식에 따른 분류
PK속성: 엔터티를 식별할 수 있는 속성
FK속성: 다른 엔터티와의 관계에서 포함된 속성
일반속성: 엔터티에 포함되어 있고 PK, FK에 포함되지 않은 속성

속성의 명명
1. 해당업무에서 사용하는 이름을 부여 한다.
2. 서술식 속성명은 사용하지 않는다.
3. 약어사용은 가급적 제한한다.
4. 전체 데이터모델에서 유일성 확보하는 것이 좋다.

제 4절 관계

관계의 표기법
관계명: 관계의 이름
관계차수: 1:1, 1:M, M:N
관계선택사양: 필수관계, 선택관계

관계 체크사항
- 두 개의 엔터티 사이에 관심있는 연관규칙이 존재하는가?
- 두 개의 엔터티 사이에 정보의 조합이 발생하는가?
- 업무기술서, 장표에 관계연결에 대한 규칙이 서술되어 있는가?
- 업무기술서, 장표에 관계연결을 가능하게 하는 동사가 있는가?

제 5절 식별자

식별자 특징

- 주식별자에 의해 엔터티 내에 모든 인스턴스들이 유일하게 구분되어야 한다.
- 주식별자를 구성하는 속성의 수는 유일성을 만족하는 최소의 수가 되어야 한다.
- 지정된 주식별자의 값은 자주 변하지 않는 것이어야 한다.
- 주식별자가 지정이 되면 반드시 값이 들어와야 한다.

주식별자의 특징

유일성: 주식별자에 의해 엔터티 내에 모든 인스턴스들을 유일하게 구분함.
최소성: 주식별자를 구성하는 속성의 수는 유일성을 만족하는 최소의 수가 되어야 함.
불변성: 주식별자가 한 번 특정 엔터티에 지정되면 그 식별자의 값은 변하지 않아야 함.
존재성: 주식별자가 지정되면 반드시 데이터 값이 존재(NULL 안됨)

식별자 분류

[표 Ⅰ-1-8] 식별자의 분류체계

분류	식별자	설명
대표성 여부	주식별자	엔터티 내에서 각 어커런스를 구분할 수 있는 구분자이며, 타 엔터티와 참조관계를 연결할 수 있는 식별자
	보조식별자	엔터티 내에서 각 어커런스를 구분할 수 있는 구분자이나 대표성을 가지지 못해 참조관계 연결을 못함
스스로 생성여부	내부식별자	엔터티 내부에서 스스로 만들어지는 식별자
	외부식별자	타 엔터티와의 관계를 통해 타 엔터티로부터 받아오는 식별자
속성의 수	단일식별자	하나의 속성으로 구성된 식별자
	복합식별자	둘 이상의 속성으로 구성된 식별자
대체 여부	본질식별자	업무에 의해 만들어지는 식별자
	인조식별자	업무적으로 만들어지지는 않지만 원조식별자가 복잡한 구성을 가지고 있기 때문에 인위적으로 만든 식별자

주식별자 도출기준

- 해당 업무에서 자주 이용되는 속성을 주식별자로 지정한다.
- 명칭, 내역 등과 같이 이름으로 기술되는 것들은 가능하면 주식별자로 지정하지 않는다.
- 복합으로 주식별자를 구성할 경우 너무 많은 속성이 포함되지 않도록 한다.

식별자관계

부모로부터 받은 식별자를 자식엔터티의 주식별자로 이용하는 경우는 NULL값이 오면 안되므로 반드시 부모엔터티가 생성되어야 자기 자신의 엔터티가 생성되는 경우이다.
자식엔터티의 주식별자로 부모의 주식별자가 상속이 되는 경우를 식별자 관계라고 지징힌다.

비식별자관계

1) 자식엔터티에서 받은 속성이 반드시 필수가 아니어도 무방하기 때문에 부모 없는 자식이 생성될 수 있는 경우이다.
2) 엔터티별로 데이터의 생명주기(Life Cycle)를 다르게 관리할 경우이다. 예를 들어 부모엔터티에 인스턴스가 자식의 엔터티와 관계를 가지고 있었지만 자식만 남겨두고 먼저 소멸될 수 있는 경우가 이에 해당된다. 이에 대한 방안으로 물리데이터베이스 생성 시 Foreign Key를 연결하지 않는 임시적인 방법을 사용하기도 하지만 데이터 모델상에서 관계를 비식별자관계로 조정하는 것이 가장 좋은 방법이다.
3) 여러 개의 엔터티가 하나의 엔터티로 통합되어 표현되었는데 각각의 엔터티가 별도의 관계를 가질 때이며 이에 해당된다.
4) 자식엔터티에 주식별자로 사용하여도 되지만 자식엔터티에서 별도의 주식별자를 생성하는 것이 더 유리하다고 판단될 때 비식별자 관계에 의한 외부식별자로 표현한다.

[표 Ⅰ-1-10] 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 부모엔터티 종속 - 자식 주식별자구성에 부모 주식별자포함 필요 - 상속받은 주식별자속성을 타 엔터티에 이전 필요	- 약한 종속관계 - 자식 주식별자구성을 독립적으로 구성 - 자식 주식별자구성에 부모 주식별자 부분 필요 - 상속받은 주식별자속성을 타 엔터티에 차단 필요 - 부모쪽의 관계참여가 선택관계

제 2장 데이터 모델과 성능

제 1절 성능 데이터 모델링의 개요

성능 데이터 모델링 고려사항

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.
⑥ 성능관점에서 데이터 모델을 검증한다.

제 2절 정규화와 성능

정규화를 통한 성능 향상 전략

데이터의 중복속성을 제거하고 결정자에 의해 동일한 의미의 일반속성이 하나의 테이블로 집약되므로 한 테이블의 데이터 용량이 최소화되는 효과가 있다. 따라서 정규화된 테이블은 데이터를 처리할 때 속도가 빨라질 수도 있고 느려질 수도 있는 특성이 있다.

1차 정규화: 복수의 속성 값을 갖는 속성을 분리
2차 정규화: 주식별자에 종속적이지 않은 속성의 분리
부분 종속 속성을 분리

함수적 종속성에 근거한 정규화 수행 필요

함수의 종속성은 데이터들이 어떤 기준값에 의해 종속되는 현상을 지칭하는 것이다.
ex) 주민등록번호 -> (이름, 출생지, 호주)

제 3절 반정규화와 성능

반정규화의 정의

정규화된 엔터티, 속성, 관계에 대해 시스템의 성능향상과 개발(Development)과 운영(Maintenance)의 단순화를 위해 중복, 통합, 분리 등을 수행하는 데이터 모델링의 기법을 의미한다.

반정규화 절차

1. 반정규화 대상조사
2. 다른 방법유도 검토
3. 반정규화 적용

반정규화의 기법

가. 데이블 반정규화

[표 Ⅰ-2-1] 테이블의 반정규화

기법분류	기법	내용
테이블병합	1:1 관계 테이블병합	1:1 관계를 통합하여 성능향상
	1:M 관계 테이블병합	1:M 관계 통합하여 성능향상
	슈퍼/서브타입 테이블병합	슈퍼/서브 관계를 통합하여 성능향상
테이블분할	수직분할	칼럼단위의 테이블을 디스크 I/O를 분산처리 하기 위해 테이블을 1:1로 분리하여 성능향상(트랜잭션의 처리되는 유형을 파악이 선행되어야 함)
	수평분할	로우 단위로 집중 발생되는 트랜잭션을 분석하여 디스크 I/O및 데이터접근의 효율성을 높여 성능을 향상하기 위해 로우단위로 테이블을 쪼갬(관계가 없음)
테이블추가	중복테이블 추가	다른 업무이거나 서버가 다른 경우 동일한 테이블구조를 중복하여 원격조인을 제거하여 성능을 향상
	통계테이블 추가	SUM, AVG 등을 미리 수행하여 계산해 둠으로써 조회 시 성능을 향상
	이력테이블 추가	이력테이블 중에서 마스터 테이블에 존재하는 레코드를 중복하여 이력테이블에 존재하는 방법은 반정규화의 유형
	부분테이블 추가	하나의 테이블의 전체 칼럼 중 자주 이용하는데 자주 이용하는 집중화된 칼럼들이 있을 때 디스크 I/O를 줄이기 위해 해당 칼럼들을 모아놓은 별도의 반정규화된 테이블을 생성

나. 칼럼 반정규화

[표 Ⅰ-2-2] 칼럼의 반정규화

반정규화 기법	내용
중복칼럼 추가	조인에 의해 처리할 때 성능저하를 예방하기 위해 즉, 조인을 감소시키기 위해 중복된 칼럼을 위치시킴
파생칼럼 추가	트랜잭션이 처리되는 시점에 계산에 의해 발생되는 성능저하를 예방하기 위해 미리 값을 계산하여 칼럼에 보관함, Derived Column이라고 함
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
PK에 의한 칼럼 추가	복합의미를 갖는 PK를 단일 속성으로 구성하였을 경우 발생됨, 단일 PK안에서 특정 값을 별도로 조회하는 경우 성능저하가 발생될 수 있음, 이 때 이미 PK안에 데이터가 존재하지만 성능향상을 위해 일반속성으로 포함하는 방법이 PK의한 칼럼추가 반정규화임
응용시스템 오작동을 위한 칼럼 추가	업무적으로는 의미가 없지만 사용자가 데이터처리를 하다가 잘못 처리하여 원래 값으로 복구하기를 원하는 경우 이전 데이터를 임시적으로 중복하여 보관하는 기법, 칼럼으로 이것을 보관하는 방법은 오작동 처리를 위한 임시적인 기법이지만 이것을 이력데이터 모델로 풀어내면 정상적인 데이터 모델의 기법이 될 수 있음

다. 관계 반정규화

[표 Ⅰ-2-3] 관계의 반정규화

반정규화 기법	내용
중복관계 추가	데이터를 처리하기 위한 여러 경로를 거쳐 조인이 가능하지만 이 때 발생할 수 있는 성능저하를 예방하기 위해 추가적인 관계를 맺는 방법이 관계의 반정규화임

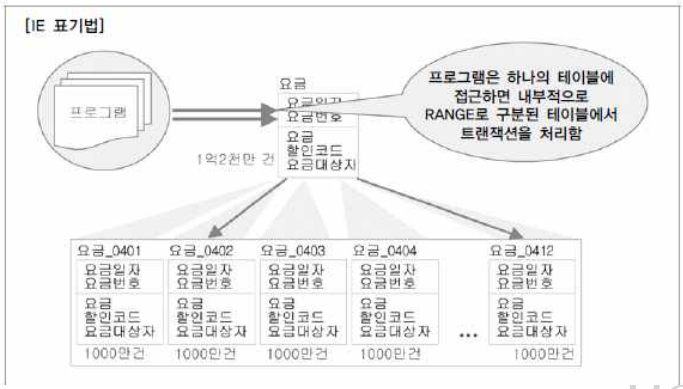
제 4절 대량 데이터에 따른 성능

대량 데이터발생에 따른 테이블 분할 개요

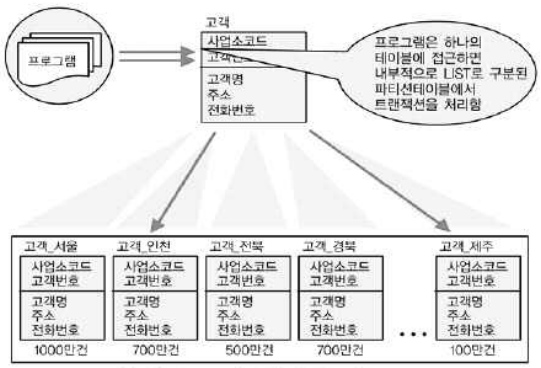
로우체이닝: 로우 길이가 너무 길어서 데이터 블록 하나에 데이터가 모두 저장되지 않고 두 개 이상의 블록에 걸쳐 하나의 로우가 저장되어 있는 형태.
로우마이그레이션: 데이터 블록에서 수정이 발생하면 수정된 데이터를 해당 데이터 블록에서 저장하지 못하고 다른 블록의 빈 공간을 찾아 저장하는 방식.

대량 데이터 저장 및 처리로 인해 성능

가. RANGE PARTITION: 대상 테이블이 날자 또는 숫자 값으로 분리가 가능하고 각 영역별로 트랜잭션이 분리되는 경우.



나. LIST PARTITION: 지점, 사업소, 사업장, 핵심적인 코드값 등으로 PK가 구성되어 있고 대량의 데이터가 있는 테이블이라면 값 각각에 의해 파티셔닝이 되는 경우.



[그림 Ⅰ-2-24] 파티셔닝의 적용-LIST

다. HASH PARTITION: 지정된 HASH 조건에 따라 해쉥 알고리즘이 적용되어 테이블이 분리되며 설계자는 테이블에 데이터가 정확하게 어떻게 들어갔는지 알 수 없다.

테이블에 대한 수평분할/수직분할의 절차

1) 데이터 모델링을 완성한다.
2) 데이터베이스 용량산정을 한다.
3) 대량 데이터가 처리되는 테이블에 대해서 트랜잭션 처리 패턴을 분석한다.
4) 칼럼 단위로 집중화된 처리가 발생하는지, 로우단위로 집중화된 처리가 발생되는지 분석하여 집중화된 단위로 테이블을 분리하는 것을 검토한다.

제 5절 데이터베이스 구조와 성능

슈퍼/서브타입 데이터 모델의 개요

공통의 부분을 슈퍼타입으로 모델링하고 공통으로부터 상속받아 다른 엔터티와 차이가 있는 속성에 대해서는 별도의 서브엔터티로 구분하여 업무의 모습을 정확하게 표현하면서 물리적인 데이터 모델로 변환을 할 때 선택의 폭을 넓힐 수 있는 장점이 있다.

슈퍼/서브 타입 데이터 모델의 변환기술

1) 개별로 발생되는 트랜잭션에 대해서는 개별 테이블로 구성(OneToOne Type)
2) 슈퍼타입+서브타입에 대해 발생되는 트랜잭션에 대해서는 슈퍼타입+서브타입 테이블로 구성(Plus Type)
3) 전체를 하나로 묶어 트랜잭션이 발생할 때는 하나의 테이블로 구성(Single Type)

[표 Ⅰ-2-4] 슈퍼/서브타입 데이터 모델 변환타입 비교

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

인덱스 특성을 고려한 PK/FK 데이터베이스 성능향상

인덱스의 특징은 여러 개의 속성이 하나의 인덱스로 구성되어 있을 때 앞쪽에 위치한 속성의 값이 비교자로 있어야 인덱스가 좋은 효율을 나타낼 수 있다. 앞쪽에 위치한 속성 값이 가급적 '=' 아니면 최소한 범위 'BETWEEN' '< >'가 들어와야 인덱스를 이용할 수 있는 것이다.

제 6절 분산 데이터베이스와 성능

분산 데이터베이스의 개요

- 여러 곳으로 분산되어있는 데이터베이스를 하나의 가상 시스템으로 사용할 수 있도록 한 데이터베이스
- 논리적으로 동일한 시스템에 속하지만, 컴퓨터 네트워크를 통해 물리적으로 분산되어 있는 데이터들의 모임. 물리적 Site 분산, 논리적으로 사용자 통합·공유

분산 데이터베이스의 투명성

1) 분할 투명성 (단편화) : 하나의 논리적 Relation이 여러 단편으로 분할되어 각 단편의 사본이 여러 site에 저장
2) 위치 투명성 : 사용하려는 데이터의 저장 장소 명시 불필요. 위치정보가 System Catalog에 유지되어야 함
3) 지역사상 투명성 : 지역DBMS와 물리적 DB사이의 Mapping 보장. 각 지역시스템 이름과 무관한 이름 사용 가능
4) 중복 투명성 : DB 객체가 여러 site에 중복 되어 있는지 알 필요가 없는 성질
5) 장애 투명성 : 구성요소(DBMS, Computer)의 장애에 무관한 Transaction의 원자성 유지
6) 병행 투명성 : 다수 Transaction 동시 수행시 결과의 일관성 유지, Time Stamp, 분산 2단계 Locking을 이용 구현

분산 데이터베이스 장단점

장점	단점
- 지역 자치성, 점증적 시스템 용량 확장	- 소프트웨어 개발 비용
- 신뢰성과 가용성	- 오류의 잠재성 증대
- 효용성과 융통성	- 처리 비용의 증대
- 빠른 응답 속도와 통신비용 절감	- 설계, 관리의 복잡성과 비용
- 데이터의 가용성과 신뢰성 증가	- 불규칙한 응답 속도
- 시스템 규모의 적절한 조절	- 통제의 어려움
- 각 지역 사용자의 요구 수용 증대	- 데이터 무결성에 대한 위협

분산 데이터베이스의 적용 기법

가. 테이블 위치 분산
나. 테이블 분할 분산
- 수평분할
- 수직분할

다. 테이블 복제 분산
- 부분복제
- 광역복제

라. 테이블 요약 분산
- 분산요약
- 통합요약

데이터베이스 분산 설계가 효과적인 경우

- 성능이 중요한 사이트
- 공통코드, 기준정보, 마스터 데이터 등에 대해 분산환경을 구성하면 성능이 좋아진다.
- 실시간 동기화가 요구되지 않을 때 좋다. 거의 실시간의 업무적인 특징을 가지고 있을 때도 분산 환경을 구성할 수 있다
- 특정 서버에 부하가 집중이 될 때 부하를 분산할 때도 좋다.
- 백업 사이트를 구성할 때 간단하게 분산기능을 적용하여 구성할 수 있다.

제 1장 SQL 기본

제 1절 관계형 데이터베이스 개요

SQL 문장들의 종류
데이터 조작어(DML): SELECT, INSERT, UPDATE, DELETE
데이터 정의어(DDL): CREATE, ALTER, DROP, RENAME
데이터 제어어(DCL): GRANT, REVOKE
트랜잭션 제어어(TCL): COMMIT, ROLLBACK

제 2절 DDL(DATA DEFINITION LANGUAGE)

자주 쓰이는 데이터 유형
CHARACTER(s): 고정 길이 문자열 정보
VARCHAR(s): 가변 길이 문자열 정보
NUMBERIC: 정수, 실수 등 숫자 정보
DATA: 날짜와 시각 정보
● CHAR 유형
'AA' = 'AA '
● VARCHAR 유형
'AA' ≠ 'AA '

CREATE TABLE 테이블 이름(
);

테이블 생성 시에 주의해야 할 사항
- 테이블명은 객체를 의미할 수 있는 적절한 이름을 사용한다. 가능한 단수형을 권고한다.
- 테이블 명은 다른 테이블의 이름과 중복되지 않아야 한다.
- 한 테이블 내에서는 칼럼명이 중복되게 지정될 수 없다.
- 테이블 이름을 지정하고 각 칼럼들은 괄호 "( )" 로 묶어 지정한다.
- 각 칼럼들은 콤마 ","로 구분되고, 테이블 생성문의 끝은 항상 세미콜론 ";"으로 끝난다.
- 칼럼에 대해서는 다른 테이블까지 고려하여 데이터베이스 내에서는 일관성 있게 사용하는 것이 좋다.(데이터 표준화 관점)
- 칼럼 뒤에 데이터 유형은 꼭 지정되어야 한다.
- 테이블명과 칼럼명은 반드시 문자로 시작해야 하고, 벤더별로 길이에 대한 한계가 있다.
- 벤더에서 사전에 정의한 예약어(Reserved word)는 쓸 수 없다.
- A-Z, a-z, 0-9, _, $, # 문자만 허용된다.

제약조건의 종류
PRIMARY KEY(기본키): 테이블에 저장된 행 데이터를 고유하게 식별하기 위한 기본키를 정의한다. 하나의 테이블에 하나의 기본키 제약만 정의할 수 있다.
UNIQUE KEY(고유키): 테이블에 저장된 행 데이터를 고유하게 식별하기 위한 고유키를 정의한다.
NOT NULL: NULL값의 입력을 금지한다.
CHECK: 입력할 수 있는 값의 범위 등을 제한한다.
FOREIGN KEY(외래키): 관계형 데이터베이스에서 테이블 간의 관계를 정의하기 위해 기본키를 다른 테이블의 외래키로 복사하는 경우 외래키가 생성된다.
외래키 지정시 참조 무결성 제약 옵션을 선택할 수 있다.

ALTER TABLE 테이블명
ADD 추가할 칼럼명 데이터 유형;
새롭게 추가된 칼럼은 테이블의 마지막 칼럼이 되며 칼럼의 위치를 지정할 수는 없다.

ALTER TABLE 테이블명
DROP COLUMN 삭제할 컬럼명;
한 번에 하나의 칼럼만 삭제 가능하며, 삭제 후 최소 하나 이상의 칼럼이 테이블에 존재해야 한다. 복구 불가능

ALTER TABLE 테이블명
MODIFY (칼럼명1 데이터 유형 [DEFAULT 식] [NOT NULL]);
ALTER TABLE 테이블명
ALTER (칼럼명1 데이터 유형 [DEFAULT 식] [NOT NULL]);
테이블의 칼럼에 대한 정의를 변경하는 명령이다.

테이블 변경 시에 주의해야 할 사항
- 해당 칼럼의 크기를 늘릴 수는 있지만 줄이지는 못한다. 이는 기존의 데이터가 훼손될 수 있기 때문이다.
- 해당 칼럼이 NULL 값만 가지고 있거나 테이블에 아무 행도 없으면 칼럼의 폭을 줄일 수 있다.
- 해당 칼럼이 NULL 값만을 가지고 있으면 데이터 유형을 변경할 수 있다.
- 해당 칼럼의 DEFAULT 값을 바꾸면 변경 작업 이후 발생하는 행 삽입에만 영향을 미치게 된다.
- 해당 칼럼에 NULL 값이 없을 경우에만 NOT NULL 제약조건을 추가할 수 있다.

ALTER TABLE 테이블명
RENAME COLUMN 변경해야 할 칼럼명 TO 새로운 칼럼명;
sp_rename 변경해야 할 칼럼명, 새로운 칼럼명, 'COLUMN';

ALTER TABLE 테이블명
DROP CONSTRAINT 제약조건명;
제약조건을 삭제하는 명령어

ALTER TABLE 테이블명
ADD CONSTRATINT 제약조건명 제약조건 (컬럼명);
제약조건을 추가하는 명령어

DROP TABLE 테이블명 [CASCADE CONSTRAINT];
테이블의 모든 데이터 및 구조를 삭제.
CASCADE CONSTRAINT 옵션은 해당 테이블과 관계가 있었던 참조되는 계약조건에 대해서도 삭제.

TRUNCATE TABLE PLAYER;
모든 행들이 제거되고 저장 공간을 재사용 가능하도록 해제한다. 테이블 구조를 완전히 삭제하기 위해서는 DROP TABLE을 실행하면 된다.

제 3절 DML(DATA MANIPULATION LANGUAGE)

DML
만들어진 테이블에 관리하기를 원하는 자료들을 입력, 수정, 삭제, 조회함.

INSERT INTO 테이블명 (COLUMN_LIST)
VALUES (COLUMN_LIST에 넣을 VALUE_LIST);
INSERT INTO 테이블명
VALUES (전체 COLUMN에 넣을 VALUE_LIST);

UPDATE 테이블명
SET 수정되어야 할 칼럼명 = 수정되기를 원하는 새로운 값;

DELETE [FROM] 삭제를 원하는 정보가 들어있는 테이블명;
테이블의 전체 데이터 삭제

SELECT [ALL/DISTINCT] 보고 싶은 칼럼명
FROM 해당 칼럼이 있는 테이블명;
DISTINCT: 중복된 데이터가 있는 경우 1건으로 처리해서 출력한다.

SELECT *
FROM 테이블명;
모든 칼럼 정보 조회(WILDCARD)

ALIAS 부여하기
- 칼럼명 바로 뒤에 온다.
- 칼럼명과 ALIAS 사이에 AS, as 키워드를 사용할 수도 있다.
- 이중 인용부호는 ALIAS가 공백, 특수문자를 포함할 경우와 대소문자 구분이 필요할 경우 사용된다.

ex)
SELECT PLAYER_NAME AS 선수명, POSITION AS 위치, HEIGHT AS 키, WEIGHT AS 몸무게
FROM PLAYER;

산술 연산자
우선순위
(), *, /, +, -

합성 연산자
- 문자와 문자를 연결하는 경우 2개의 수직 바(||)에 의해 이루어진다. (Oracle)
- 문자와 문자를 연결하는 경우 + 표시에 의해 이루어진다. (SQL Server)
- 두 벤더 모두 공통적으로 CONCAT (string1, string2) 함수를 사용할 수 있다.
- 칼럼과 문자 또는 다른 칼럼과 연결시킨다.
- 문자 표현식의 결과에 의해 새로운 칼럼을 생성한다.

제 4절 TCL(TRANSACTION CONTROL LANGUAGE)

트랜잭션
데이터베이스의 논리적 연산단위. 분리될 수 없는 한 개 이상의 데이터베이스 조작을 가리킨다.
ALL OR NOTHING 개념

트랜잭션의 특성
원자성: ALL OR NOTHING
일관성: 트랜잭션이 실행되기 전의 데이터베이스 내용이 잘못 되어 있지 않다면 트랜잭션이 실행된 이후에도 데이터베이스의 내용에 잘못이 있으면 안 된다.
고립성: 트랜잭션이 실행되는 도중에 다른 트랜잭션의 영향을 받아 잘못된 결과를 만들어서는 안된다.
지속성: 트랜잭션이 성공적으로 수행되면 그 트랜잭션이 갱신한 데이터베이스의 내용은 영구적으로 저장된다.

COMMIT
입력한 자료나 수정한 자료에 대해서 또는 삭제한 자료에 대해서 전혀 문제가 없다고 판단되었을 경우 COMMIT 명령어를 통해서 트랜잭션을 완료할 수 있다.

AUTO COMMIT
SQL Server의 기본 방식이며, DML, DDL을 수행할 때마다 DBMS가 트랜잭션을 컨트롤하는 방식이다. 명령어가 성공적으로 수행되면 자동으로 COMMIT을 수행하고 오류가 발생하면 자동으로 ROLLBACK을 수행한다.
Oracle DDL 명령어의 경우 AUTO COMMIT, DML의 경우 COMMIT 입력 필요.

ROLLBACK
테이블 내 입력한 데이터나, 수정한 데이터, 삭제한 데이터에 대해서 COMMIT 이전에는 변경 사항을 취소할 수 있는데 데이터베이스에서는 롤백 기능을 사용한다.

COMMIT과 ROLLBACK 효과
- 데이터 무결성 보장
- 영구적인 변경을 하기 전에 데이터의 변경사항 확인 가능
- 논리적으로 연관된 작업을 그룹핑하여 처리 가능

SAVEPOINT
현 시점에서 SAVEPOINT까지 트랙잭션의 일부만 롤백할 수 있다.

SAVEPOINT SVPT1;
ROLLBACK TO SVPT1;

SAVE TRANSACTION SVTR1;
ROLLBACK TRANSACTION SVTR;

제 5절 WHERE 절

연산자 우선순위

[표 Ⅱ-1-16] 연산자의 우선순위

연산 우선순위	설 명
1	괄호 ()
2	NOT 연산자
3	비교 연산자, SQL 비교 연산자
4	AND
5	OR

연산자의 종류

[표 Ⅱ-1-15] 연산자의 종류

구분	연산자	연산자의 의미
비교 연산자	=	같다.
	〉	보다 크다.
	〉=	보다 크거나 같다.
	〈	보다 작다.
	〈=	보다 작거나 같다.
SQL 연산자	BETWEEN a AND b	a와 b의 값 사이에 있으면 된다.(a와 b 값이 포함됨)
	IN (list)	리스트에 있는 값 중에서 어느 하나라도 일치하면 된다.
	LIKE '비교문자열'	비교문자열과 형태가 일치하면 된다.(%, _ 사용)
	IS NULL	NULL 값인 경우
논리 연산자	AND	앞에 있는 조건과 뒤에 오는 조건이 참(TRUE)이 되면 결과도 참(TRUE)이 된다. 즉, 앞의 조건과 뒤의 조건을 동시에 만족해야 한다.
	OR	앞의 조건이 참(TRUE)이거나 뒤의 조건이 참(TRUE)이 되어야 결과도 참(TRUE)이 된다. 즉, 앞뒤의 조건 중 하나만 참(TRUE)이면 된다.
	NOT	뒤에 오는 조건에 반대되는 결과를 되돌려 준다.
부정 비교 연산자	!=	같지 않다.
	^=	같지 않다.
	◇	같지 않다.(ISO 표준, 모든 운영체제에서 사용 가능)
	NOT 칼럼명 =	~와 같지 않다.
	NOT 칼럼명 〉	~보다 크지 않다.
부정 SQL 연산자	NOT BETWEEN a AND b	a와 b의 값 사이에 있지 않다. (a, b 값을 포함하지 않는다)
	NOT IN (list)	list 값과 일치하지 않는다.
	IS NOT NULL	NULL 값을 갖지 않는다.

SQL 연산자

[표 Ⅱ-1-19] SQL 연산자의 종류

연산자	연산자의 의미
BETWEEN a AND b	a와 b의 값 사이에 있으면 된다.(a와 b의 값이 포함됨)
IN (list)	리스트에 있는 값 중에서 어느 하나라도 일치하면 된다.
LIKE '비교문자열'	비교 문자열과 형태가 일치하면 된다.
IS NULL	NULL 값인 경우

와일드 카드
%: 0개 이상의 어떤 문자를 의미.
_: 1개인 단일 문자를 의미.

ROWNUM
Oracle 원하는 만큼의 행만 가져오고 싶을 때 WHERE 절에서 행의 개수를 제한하는 목적으로 사용한다.

TOP
SQL Server는 TOP 절을 사용하여 결과 집합으로 출력되는 행의 수를 제한할 수 있다.
TOP (Expression) [PERCENT] [WITH TIES]
WITH TIES: ORDER BY 절이 지정된 경우에만 사용할 수 있으며, TOP N(PERCENT)의 마지막 행과 같은 값이 있는 경우 추가 행이 출력되도록 지정할 수 있다.

제 6절 함수

단일행 문자형 함수 종류

[표 Ⅱ-1-26] 단일행 문자형 함수 사례

문자형 함수 사용	결과 값 및 설명
LOWER('SQL Expert')	'sql expert'
UPPER('SQL Expert')	'SQL EXPERT'
ASCII('A')	65
CHR(65) / CHAR(65)	'A'
CONCAT('RDBMS',' SQL') 'RDBMS' || ' SQL' / 'RDBMS' + ' SQL'	'RDBMS SQL'
SUBSTR('SQL Expert', 5, 3) SUBSTRING('SQL Expert', 5, 3)	'Exp'
LENGTH('SQL Expert') / LEN('SQL Expert')	10
LTRIM('xxxYYZZxYZ','x') RTRIM('XXYYzzXYzz','z') TRIM('x' FROM 'xxYYZZxYZxx')	'YYZZxYZ' 'XXYYzzXY' 'YYZZxYZ'
RTRIM('XXYYZZXYZ ') → 공백 제거 및 CHAR와 VARCHAR 데이터 유형을 비교할 때 용이하게 사용된다.	'XXYYZZXYZ'

단일행 숫자형 함수 종류

[표 Ⅱ-1-28] 단일행 숫자형 함수 사례

숫자형 함수 사용	결과 값 및 설명
ABS(-15)	15
SIGN(-20) SIGN(0) SIGN(+20)	-1 0 1
MOD(7,3) / 7%3	1
CEIL(38.123) / CEILING(38.123) CEILING(-38.123)	39 39 -38
FLOOR(38.123) FLOOR(-38.123)	38 -39
ROUND(38.5235, 3) ROUND(38.5235, 1) ROUND(38.5235, 0) ROUND(38.5235)	38.524 38.5 39 39 (인수 0이 Default)
TRUNC(38.5235, 3) TRUNC(38.5235, 1) TRUNC(38.5235, 0) TRUNC(38.5235)	38.523 38.5 38 38 (인수 0이 Default)

단일행 날짜형 함수 종류

[표 Ⅱ-1-29] 단일행 날짜형 함수 종류

날짜형 함수	함수 설명
SYSDATE / GETDATE()	현재 날짜와 시각을 출력한다.
EXTRACT('YEAR'|'MONTH'|'DAY' from d) / DATEPART('YEAR'|'MONTH'|'DAY',*d*)	날짜 데이터에서 년/월/일 데이터를 출력할 수 있다. 시간/분/초도 가능함
TO_NUMBER(TO_CHAR(*d*,'YYYY')) / YEAR(*d*), TO_NUMBER(TO_CHAR(*d*,'MM')) / MONTH(*d*), TO_NUMBER(TO_CHAR(*d*,'DD')) / DAY(*d*)	날짜 데이터에서 년/월/일 데이터를 출력할 수 있다. Oracle EXTRACT YEAR/MONTH/DAY 옵션이나 SQL Server DEPART YEAR/MONTH/DAY 옵션과 같은 기능이다. TO_NUMBER 함수 제외시 문자형으로 출력됨

※ 주: Oracle함수/SQL Server함수 표시, '/' 없는 것은 공통 함수

단일행 CASE 표현 종류

[표 Ⅱ-1-33] 단일행 CASE 표현의 종류

CASE 표현	함수 설명
CASE SIMPLE_CASE_EXPRESSION 조건 ELSE 표현절 END	SIMPLE_CASE_EXPRESSION 조건이 맞으면 SIMPLE_CASE_EXPRESSION 조건내의 THEN 절을 수행하고, 조건이 맞지 않으면 ELSE 절을 수행한다.
CASE SEARCHED_CASE_EXPRESSION 조건 ELSE 표현절 END	SEARCHED_CASE_EXPRESSION 조건이 맞으면 SEARCHED_CASE_EXPRESSION 조건내의 THEN 절을 수행하고, 조건이 맞지 않으면 ELSE 절을 수행한다.
DECODE(표현식, 기준값1, 값1 [, 기준값2, 값2, … , 디폴트값)	Oracle에서만 사용되는 함수로, 표현식의 값이 기준값1이면 값1을 출력하고, 기준값2이면 값2를 출력한다. 그리고 기준값이 없으면 디폴트 값을 출력한다. CASE 표현의 SIMPLE_CASE_EXPRESSION 조건과 동일하다.

단일행 NULL 관련 함수 종류

[표 Ⅱ-1-35] 단일행 NULL 관련 함수의 종류

일반형 함수	함수 설명
NVL(표현식1, 표현식2) / ISNULL(표현식1, 표현식2)	표현식1의 결과값이 NULL이면 표현식2의 값을 출력한다. 단, 표현식1과 표현식2의 결과 데이터 타입이 같아야 한다. NULL 관련 가장 많이 사용되는 함수이므로 상당히 중요하다.
NULLIF(표현식1, 표현식2)	표현식1이 표현식2와 같으면 NULL을, 같지 않으면 표현식1을 리턴한다.
COALESCE(표현식1, 표현식2, ……)	임의의 개수 표현식에서 NULL이 아닌 최초의 표현식을 나타낸다. 모든 표현식이 NULL이라면 NULL을 리턴한다.

※ 주: Oracle함수/SQL Server함수 표시, '/' 없는 것은 공통 함수

COALESCE

인수의 숫자가 한정되어 있지 않으며, 임의의 개수 EXPR에서 NULL이 아닌 최초의 EXPR을 나타낸다. 만일 모든 EXPR이 NULL이라면 NULL을 리턴한다.

COALESCE (EXPR1, EXPR2, ...)

제 7절 GROUP BY, HAVING 절

집계 함수

- 여러 행들의 그룹이 모여서 그룹당 단 하나의 결과를 돌려주는 함수이다.
- GROUP BY 절은 행들을 소그룹화 한다.
- SELECT 절, HAVING 절, ORDER BY 절에 사용할 수 있다.

집계 함수의 종류

[표 Ⅱ-1-36] 집계 함수의 종류

집계 함수	사용 목적
COUNT(*)	NULL 값을 포함한 행의 수를 출력한다.
COUNT(표현식)	표현식의 값이 NULL 값인 것을 제외한 행의 수를 출력한다.
SUM([DISTINCT | ALL] 표현식)	표현식의 NULL 값을 제외한 합계를 출력한다.
AVG([DISTINCT | ALL] 표현식)	표현식의 NULL 값을 제외한 평균을 출력한다.
MAX([DISTINCT | ALL] 표현식)	표현식의 최대값을 출력한다. (문자,날짜 데이터 타입도 사용가능)
MIN([DISTINCT | ALL] 표현식)	표현식의 최소값을 출력한다. (문자,날짜 데이터 타입도 사용가능)
STDDEV([DISTINCT | ALL] 표현식)	표현식의 표준 편차를 출력한다.
VARIAN([DISTINCT | ALL] 표현식)	표현식의 분산을 출력한다.
기타 통계 함수	벤더별로 다양한 통계식을 제공한다.

GROUP BY 절과 HAVING 절의 특성

- GROUP BY 절을 통해 소그룹별 기준을 정한 후,

SELECT 절에 집계 함수를 사용한다.
- 집계 함수의 통계 정보는 NULL 값을 가진 행을 제외하고 수행한다.
- GROUP BY 절에서는 SELECT 절과는 달리 ALIAS 명을 사용할 수 없다.
- 집계 함수는 WHERE 절에는 올 수 없다. (집계 함수를 사용할 수 있는 GROUP BY 절보다 WHERE 절이 먼저 수행된다)
- WHERE 절은 전체 데이터를 GROUP으로 나누기 전에 행들을 미리 제거시킨다.
- HAVING 절은 GROUP BY 절의 기준 항목이나 소그룹의 집계 함수를 이용한 조건을 표시할 수 있다.
- GROUP BY 절에 의한 소그룹별로 만들어진 집계 데이터 중, HAVING 절에서 제한 조건을 두어 조건을 만족하는 내용만 출력한다.
- HAVING 절은 일반적으로 GROUP BY 절 뒤에 위치한다.

제 8절 ORDER BY 절

ORDER BY 절 사용 특징
- 기본적인 정렬 순서는 오름차순(ASC)이다.
- 숫자형 데이터 타입은 오름차순으로 정렬했을 경우에 가장 작은 값부터 출력된다.
- 날짜형 데이터 타입은 오름차순으로 정렬했을 경우 날짜 값이 가장 빠른 값이 먼저 출력된다. 예를 들어 '01-JAN-2012'는 '01-SEP-2012'보다 먼저 출력된다.
- Oracle에서는 NULL 값을 가장 큰 값으로 간주하여 오름차순으로 정렬했을 경우에는 가장 마지막에, 내림차순으로 정렬했을 경우에는 가장 먼저 위치한다.
- 반면, SQL Server에서는 NULL 값을 가장 작은 값으로 간주하여 오름차순으로 정렬했을 경우에는 가장 먼저, 내림차순으로 정렬했을 경우에는 가장 마지막에 위치한다.

SELECT 문장 실행 순서
5. SELECT 칼럼명 [ALIAS명]
1. FROM 테이블명
2. WHERE 조건식
3. GROUP BY 칼럼(Column)이나 표현식
4. HAVING 그룹조건식
6. ORDER BY 칼럼(Column)이나 표현식;

1. 발췌 대상 테이블을 참조한다. (FROM)
2. 발췌 대상 데이터가 아닌 것은 제거한다. (WHERE)
3. 행들을 소그룹화 한다. (GROUP BY)
4. 그룹핑된 값의 조건에 맞는 것만을 출력한다. (HAVING)
5. 데이터 값을 출력/계산한다. (SELECT)
6. 데이터를 정렬한다. (ORDER BY)

제 9절 조인

EQUI JOIN
두 개의 테이블 간에 칼럼 값들이 서로 정확하게 일치하는 경우에 사용되는 방법으로 PK - FK의 관계를 기반으로 한다.

SELECT 테이블1.칼럼명, 테이블2.칼럼명, ...
FROM 테이블1, 테이블2
WHERE 테이블1.칼럼명1 = 테이블2.칼럼명2;
→ WHERE 절에 JOIN 조건을 넣는다.

[ANSI/ISO]
SELECT 테이블1.칼럼명, 테이블2.칼럼명, ...
FROM 테이블1 INNER JOIN 테이블2
ON 테이블1.칼럼명1 = 테이블2.칼럼명2;
→ ON 절에 JOIN 조건을 넣는다.

Non EQUI JOIN
두 개의 테이블 간에 컬럼 값들이 서로 정확하게 일치하지 않는 경우에는 EQUI JOIN을 사용할 수 없다. 이런 경우 Non EQUI JOIN을 시도할 수 있으나 데이터 모델에 따라서 불가능한 경우도 있다.

SELECT 테이블1.칼럼명, 테이블2.칼럼명, ...
FROM 테이블1, 테이블2
WHERE 테이블1.칼럼명1 BETWEEN 테이블2.칼럼명1 AND 테이블2.칼럼명2;

제 2장 SQL 활용

제 1절 표준 조인

INNER JOIN
JOIN 조건에서 동일한 값이 있는 행만 반환한다.
USING 조건절이나 ON 조건절을 필수적으로 사용해야 한다.

NATURAL JOIN
두 테이블 간의 동일한 이름을 갖는 모든 칼럼들에 대해 EQUI JOIN을 수행한다.
NATURAL JOIN이 명시되면, 추가로 USING 조건절, ON 조건절, WHERE 절에서 JOIN 조건을 정의할 수 없다.

INNER, NATURAL JOIN 차이점은 NATURAL JOIN은 같은 이름의 칼럼을 1번만 표시.

USING 조건절
FROM 절의 USING 조건절을 이용하면 같은 이름을 가진 칼럼들 중에서 원하는 칼럼에 대해서만 선택적으로 EQUI JOIN을 할 수가 있다.

ON 조건절
JOIN 서술부(ON 조건절)와 비 JOIN 서술부(WHERE 조건절)를 분리하여 이해가 쉬우며, 칼럼명이 다르더라도 JOIN 조건을 사용할 수 있는 장점이 있다.

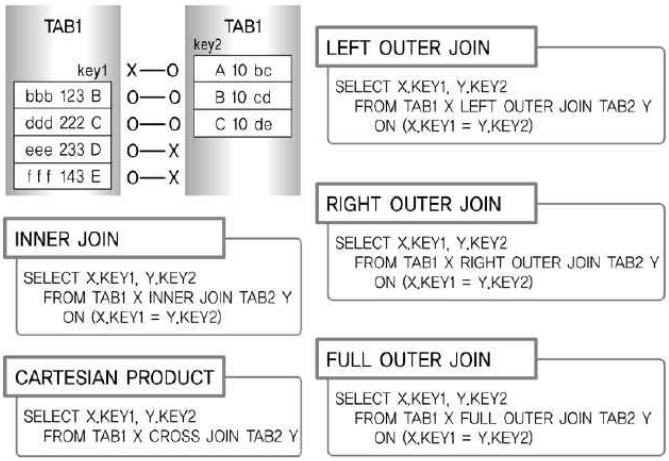
CROSS JOIN
결과는 양쪽 집합의 M*N 건의 데이터 조합이 발생한다.

LEFT OUTER JOIN
먼저 표기된 좌측 테이블에 해당하는 데이터를 먼저 읽은 후, 나중 표기된 우측 테이블에서 JOIN 대상 데이터를 읽어 온다.

RIGHT OUTER JOIN
우측 테이블이 기준이 되어 결과를 생성한다.

FULL OUTER JOIN
좌측, 우측 데이터의 모든 데이터를 읽어 JOIN하여 결과를 생성한다.

INNER vs OUTER vs CROSS JOIN 비교

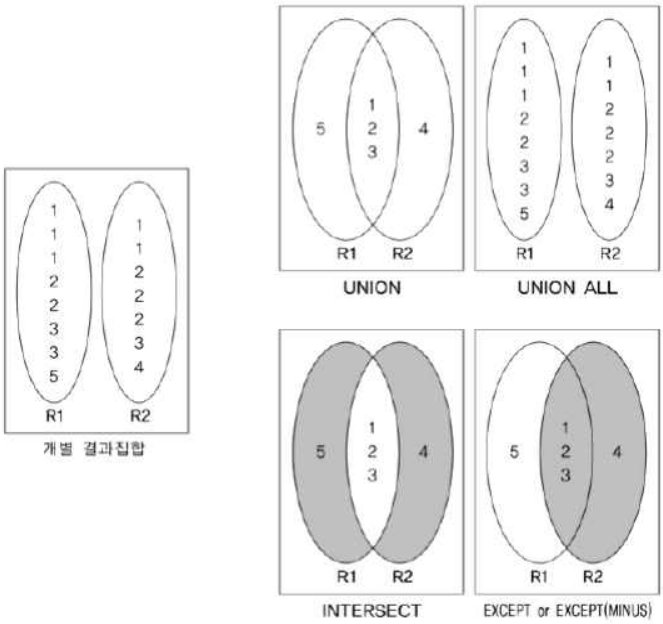


[그림 Ⅱ-2-4] INNER vs OUTER vs CROSS JOIN 문장 비교

첫 번째, INNER JOIN의 결과는 다음과 같다. 양쪽 테이블에 모두 존재하는 키 값이 B-B, C-C 인 2건이 출력된다.
두 번째, LEFT OUTER JOIN의 결과는 다음과 같다. TAB1을 기준으로 키 값 조합이 B-B, C-C, D-NULL, E-NULL 인 4건이 출력된다.
세 번째, RIGHT OUTER JOIN의 결과는 다음과 같다. TAB2를 기준으로 키 값 조합이 NULL-A, B-B, C-C 인 3건이 출력된다.
네 번째, FULL OUTER JOIN의 결과는 다음과 같다. 양쪽 테이블을 기준으로 키 값 조합이 NULL-A, B-B, C-C, D-NULL, E-NULL 인 5건이 출력된다.
다섯 번째, CROSS JOIN의 결과는 다음과 같다. JOIN 가능한 모든 경우의 수를 표시하지만 단, OUTER JOIN은 제외한다. 양쪽 테이블 TAB1과 TAB2의 데이터를 곱한 개수인 4 * 3 = 12건이 추출됨 키 값 조합이 B-A, B-B, B-C, C-A, C-B, C-C, D-A, D-B, D-C, E-A, E-B, E-C 인 12건이 출력된다.

제 2절 집합 연산자

집합 연산자의 종류



[그림 Ⅱ-2-5] 집합 연산자의 연산

제 3절 계층형 질의와 셀프 조인

Oracle 계층형 질의

- START WITH절은 계층 구조 전개의 시작 위치를 지정하는 구문이다.
- CONNECT BY절은 다음에 전개될 자식 데이터를 지정하는 구문이다.
- PRIOR : CONNECT BY절에 사용되며, 현재 읽은 칼럼을 지정한다. PRIOR 자식 = 부모 형태를 사용하면 계층구조에서 자식 데이터에서 부모 데이터(자식 → 부모) 방향으로 전개하는 순방향 전개를 한다. 그리고 PRIOR 부모 = 자식 형태를 사용하면 반대로 부모 데이터에서 자식 데이터(부모 → 자식) 방향으로 전개하는 역방향 전개를 한다.
- NOCYCLE : 데이터를 전개하면서 이미 나타났던 동일한 데이터가 전개 중에 다시 나타난다면 이것을 가리켜 사이클(Cycle)이 형성되었다라고 말한다. 사이클이 발생한 데이터는 런타임 오류가 발생한다. 그렇지만 NOCYCLE를 추가하면 사이클이 발생한 이후의 데이터는 전개하지 않는다.
- ORDER SIBLINGS BY : 형제 노드(동일 LEVEL) 사이에서 정렬을 수행한다.
- WHERE : 모든 전개를 수행한 후에 지정된 조건을 만족하는 데이터만 추출한다.(필터링)

계층형 질의에서 사용되는 가상 칼럼

[표 Ⅱ-2-2] 계층형 질의에서 사용되는 가상 칼럼

가상 칼럼	설명
LEVEL	루트 데이터이면 1, 그 하위 데이터이면 2이다. 리프(Leaf) 데이터까지 1씩 증가한다.
CONNECT_BY_ISLEAF	전개 과정에서 해당 데이터가 리프 데이터이면 1, 그렇지 않으면 0이다.
CONNECT_BY_ISCYCLE	전개 과정에서 자식을 갖는데, 해당 데이터가 조상으로서 존재하면 1, 그렇지 않으면 0이다. 여기서 조상이란 자신으로부터 루트까지의 경로에 존재하는 데이터를 말한다. CYCLE 옵션을 사용했을 때만 사용할 수 있다.

계층형 질의에서 사용되는 함수

[표 Ⅱ-2-3] 계층형 질의에서 사용되는 함수

함수	설명
SYS_CONNECT_BY_PATH	루트 데이터부터 현재 전개할 데이터까지의 경로를 표시한다. 사용법 : SYS_CONNECT_BY_PATH(칼럼, 경로분리자)
CONNECT_BY_ROOT	현재 전개할 데이터의 루트 데이터를 표시한다. 단항 연산자이다. 사용법 : CONNECT_BY_ROOT 칼럼

셀프 조인

동일 테이블 사이의 조인을 말한다. 따라서 FROM 절에 동일 테이블이 두 번 이상 나타난다. 식별을 위해 반드시 테이블 별칭(Alias)를 사용해야 한다.

제 4절 서브쿼리

주의사항

① 서브쿼리를 괄호로 감싸서 사용한다.
② 서브쿼리는 단일 행(Single Row) 또는 복수 행(Multiple Row) 비교 연산자와 함께 사용 가능하다. 단일 행 비교 연산자는 서브쿼리의 결과가 반드시 1건 이하이어야 하고 복수 행 비교 연산자는 서브쿼리의 결과 건수와 상관 없다.
③ 서브쿼리에서는 ORDER BY를 사용하지 못한다. ORDER BY절은 SELECT절에서 오직 한 개만 올 수 있기 때문에 ORDER BY절은 메인쿼리의 마지막 문장에 위치해야 한다.

서브쿼리가 SQL이문에서 사용 가능한 곳

- SELECT 절
- FROM 절
- WHERE 절
- HAVING 절
- ORDER BY 절
- INSERT문의 VALUES 절
- UPDATE문의 SET 절

SELECT 절에 서브쿼리 사용하기

스칼라 서브쿼리. 한 행, 한 칼럼만을 반환하는 서브쿼리를 말한다.

FROM절에서 서브쿼리 사용하기
인라인 뷰. 서브쿼리의 결과가 마치 실행 시에 동적으로 생성된 테이블인 것처럼 사용할 수 있다.(동적인 뷰)
서브쿼리의 칼럼은 메인쿼리에서 사용할 수 없지만 인라인 뷰는 동적으로 생성된 테이블이기 때문에 자유롭게 참조 가능.

뷰
실제 데이터를 가지고 있지 않다.

[표 Ⅱ-2-7] 뷰 사용의 장점

뷰의 장점	설명
독립성	테이블 구조가 변경되어도 뷰를 사용하는 응용 프로그램은 변경하지 않아도 된다.
편리성	복잡한 질의를 뷰로 생성함으로써 관련 질의를 단순하게 작성할 수 있다. 또한 해당 형태의 SQL문을 자주 사용할 때 뷰를 이용하면 편리하게 사용할 수 있다.
보안성	직원의 급여정보와 같이 숨기고 싶은 정보가 존재한다면, 뷰를 생성할 때 해당 칼럼을 빼고 생성함으로써 사용자에게 정보를 감출 수 있다.

제 5절 그룹 함수

ROLLUP 함수
L1 - GROUP BY 수행시 생성되는 표준 집계
L2 - DNAME 별 모든 JOB의 SUBTOTAL
L3 - GRAND TOTAL

GROUPING 함수
- ROLLUP이나 CUBE에 의한 소계가 계산된 결과에는 GROUPING(EXPR) = 1 이 표시되고,
- 그 외의 결과에는 GROUPING(EXPR) = 0 이 표시된다.

CUBE 함수
결합 가능한 모든 값에 대하여 다차원 집계를 생성.

GROUPING SET 함수
인수들에 대한 개별 집계를 구할 수 있음.

제 6절 윈도우 함수

WINDOW FUNCTION
행과 행간의 관계를 쉽게 정의하기 위해 만든 함수.

RANK 함수
특정 항목(칼럼)에 대한 순위를 구하는 함수. 동일한 값에 대해서는 동일한 순위를 부여하게 된다.

DENSE_RANK 함수
동일한 순위를 하나의 건수로 취급하는 함.

ROW_NUMBER 함수
동일한 값이라도 고유한 순위를 부여함.

SUM MAX MIN AVG COUNT

FIRST_VALUE 함수
파티션별 윈도우에서 가장 먼저 나온 값을 구한다.(SQL Server X) = MIN

LAST_VALUE 함수
파티션별 윈도우에서 가장 나중에 나온 값을 구한다.(SQL Server X) = MAX

LAG 함수
파티션별 윈도우에서 이전 몇 번째 행의 값을 가져올 수 있다.(SQL Server X)

LEAD 함수
파티션별 윈도우에서 이후 몇 번째 행의 값을 가져올 수 있다.(SQL Server X)

RATIO_TO_REPORT 함수
파티션 내 전체 SUM값에 대한 행별 칼럼 값의 백분율을 소수점으로 구할 수 있다.(SQL Server X)

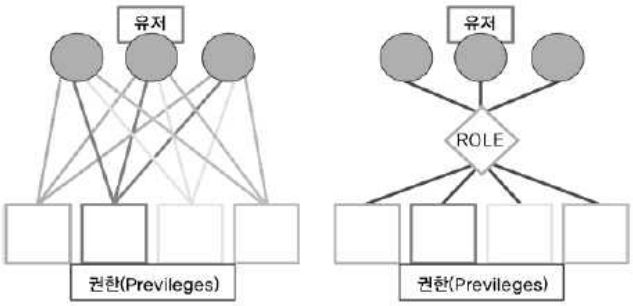
PERCENT_RANK 함수
파티션별 윈도우에서 제일 먼저 나오는 것을 0으로, 제일 늦게 나오는 것을 1로 하여, 값이 아닌 행의 순서별 백분율을 구한다.(SQL Server X)

CUME_DIST 함수
파티션별 윈도우의 전체건수에서 현재 행보다 작거나 같은 건수에 대해 누적백분율을 구한다.(SQL Server X)

제 7절 DCL(DATA CONTROL LANGUAGE)

GRANT: 권한 부여
REVOKE: 권한 취소

ROLE



[그림 Ⅱ-2-17] ROLE 의 개념

제 8절 절차형 SQL

SQL문의 연속적인 실행이나 조건에 따른 분기처리를 이용하여 특정 기능을 수행하는 저장 모듈을 생성할 수 있다.

PL/SQL 특징

- PL/SQL은 Block 구조로 되어있어 각 기능별로 모듈화가 가능하다.
- 변수, 상수 등을 선언하여 SQL 문장 간 값을 교환한다.
- IF, LOOP 등의 절차형 언어를 사용하여 절차적인 프로그램이 가능하도록 한다.
- DBMS 정의 에러나 사용자 정의 에러를 정의하여 사용할 수 있다.
- PL/SQL은 Oracle에 내장되어 있으므로 Oracle과 PL/SQL을 지원하는 어떤 서버로도 프로그램을 옮길 수 있다.
- PL/SQL은 응용 프로그램의 성능을 향상시킨다.
- PL/SQL은 여러 SQL 문장을 Block으로 묶고 한 번에 Block 전부를 서버로 보내기 때문에 통신량을 줄일 수 있다.

T-SQL 특징

- 변수 선언 기능 @@이라는 전역변수(시스템 함수)와 @이라는 지역변수가 있다.
- 지역변수는 사용자가 자신의 연결 시간 동안만 사용하기 위해 만들어지는 변수이며 전역변수는 이미 SQL서버에 내장된 값이다.
- 데이터 유형(Data Type)을 제공한다. 즉 int, float, varchar 등의 자료형을 의미한다.
- 연산자(Operator) 산술연산자(+, -, *, /)와 비교연산자(=, <, >, <>) 논리연산자(and, or, not) 사용이 가능하다.
- 흐름 제어 기능 IF-ELSE와 WHILE, CASE-THEN 사용이 가능하다.
- 주석 기능한줄 주석 : -- 뒤의 내용은 주석범위 주석 : /* 내용 */ 형태를 사용하며, 여러 줄도 가능함

Trigger

특정한 테이블에 INSERT, UPDATE, DELETE와 같은 DML문이 수행되었을 때, 데이터베이스에서 자동으로 동작하도록 작성된 프로그램이다.

프로시저와 트리거의 차이점

[표 Ⅱ-2-19] 프로시저와 트리거의 차이점

프로시저	트리거
CREATE Procedure 문법사용	CREATE Trigger 문법사용
EXECUTE 명령어로 실행	생성 후 자동으로 실행
COMMIT, ROLLBACK 실행 가능	COMMIT, ROLLBACK 실행 안됨

제 3장 SQL 최적화 기본 원리

제 1절 옵티마이져와 실행 계획

옵티마이저

가. 규칙기반 옵티마이저

우선 순위가 높은 규칙이 적은 일량으로 해당 작업을 해당 작업을 수행하는 방법이라고 판단하는 것이다.

나. 비용기반 옵티마이저

SQL을 처리하는데 필요한 비용(소요시간 또는 자원 사용량)이 가장 적은 실행계획을 선택하는 방식이다.

정확한 통계정보를 유지하는 것은 비용기반 최적화에서 중요한 요소이다.

실행계획

조인 순서: 조인작업을 수행할 때 참조하는 테이블의 순서

조인 기법: 두 개의 테이블을 조인할 때 사용할 수 있는 방법. NL Join, Hash Join, Sort Merge Join 등이 있다.

액세스 기법: 하나의 테이블을 액세스할 때 사용할 수 있는 방법. 인덱스 스캔, 전체 테이블 스캔 등이 있다.

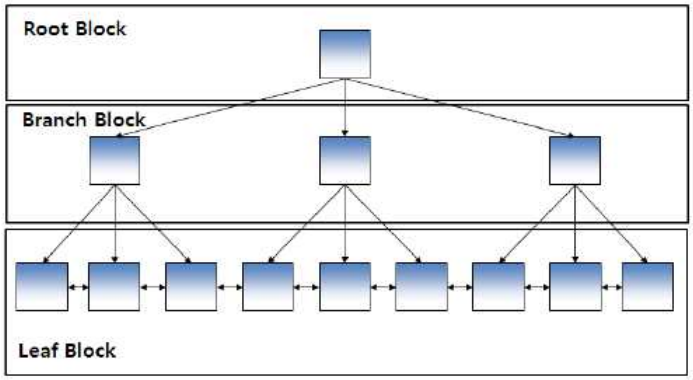
최적화 정보: 옵티마이저가 실행계획의 각 단계마다 예상되는 비용 사항을 표시한 것이다. Cost는 상대적인 비용 정보이고 Card는 주어진 조건을 만족한 결과 집합 혹은 조인 조건을 만족한 결과 집합의 건수를 의미한다. Bytes는 결과 집합이 차지하는 메모리양을 바이트로 표시한 것이다.

연산: 여러 가지 조작을 통해서 원하는 결과를 얻어내는 일련의 작업이다.

제 2절 인덱스 기본

인덱스 특징과 종류

가. 트리 기반 인덱스



[그림 Ⅱ-3-6] B-트리 인덱스 구조

1단계. 브랜치 블록의 가장 왼쪽 값이 찾고자 하는 값보다 작거나 같으면 왼쪽 포인터로 이동

2단계. 찾고자 하는 값이 브랜치 블록의 값 사이에 존재하면 가운데 포인터로 이동

3단계. 오른쪽에 있는 값보다 크면 오른쪽 포인터로 이동

나. SQL Server의 클러스터형 인덱스

첫째, 인덱스의 리프 페이지가 곧 데이터 페이지다. 따라서 테이블 탐색에 필요한 레코드 식별자가 리프 페이지에 없다.

둘째, 리프 페이지의 모든 로우(=데이터)는 인덱스 키 칼럼 순으로 물리적으로 정렬되어 저장된다. 테이블 로우는 물리적으로 한 가지 순서로만 정렬될 수 있다. 그러므로 클러스터형 인덱스는 테이블당 한 개만 생성할 수 있다.

전체 테이블 스캔과 인덱스 스캔

가. 전체 테이블 스캔

전체 테이블 스캔 방식으로 데이터를 검색한다는 것은 테이블에 존재하는 모든 데이터를 읽어 가면서 조건에 맞으면 결과로서 추출하고 조건에 맞지 않으면 버리는 형식으로 검색한다.

1) SQL문에 조건이 존재하지 않는 경우

2) SQL문의 주어진 조건에 사용 가능한 인덱스가 존재하지 않는 경우

3) 옵티마이저의 취사 선택

4) 그 밖의 경우(병렬처리 방식으로 처리하는 경우 등)

나. 인덱스 스캔

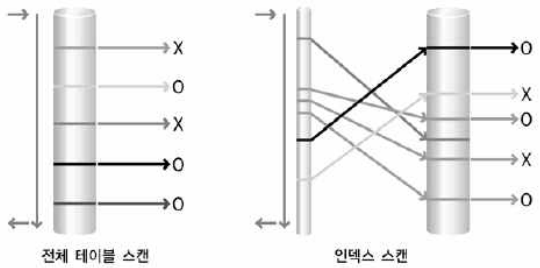
인덱스를 구성하는 칼럼의 값을 기반으로 데이터를 추출하는 액세스 기법.

1) 인덱스 유일 스캔: 유일 인덱스를 사용하여 단 하나의 데이터를 추출하는 방식. 유일 인덱스 구성 칼럼에 대해 모두 '='로 값이 주어진 경우에만 가능한 인덱스 스캔 방식이다.

2) 인덱스 범위 스캔: 인덱스를 이용하여 한 건 이상의 데이터를 추출하는 방식이다. 유일 인덱스의 구성 칼럼 모두에 대해 '='로 값이 주어지지 않은 경우와 비유일 인덱스를 이용하든 모든 액세스 방식은 인덱스 범위 스캔 방식으로 데이터를 액세스하는 것이다.

3) 인덱스 역순 범위 스캔: 인덱스의 리프 블록의 양방향 링크를 이용하여 내림 차순으로 데이터를 읽는 방식이다. 이 바식을 이용하여 최대값을 쉽게 찾을 수 있다.

다. 전체 테이블 스캔과 인덱스 스캔 방식의 비교



[그림 Ⅱ-3-11] 전체 테이블 스캔과 인덱스 스캔에 대한 SQL 처리 흐름도 표현 예시

제 3절 조인 수행 원리

1. NL Join

프로그래밍에서 사용하는 중첩된 반복문과 유사한 방식으로 조인을 수행한다.

결과를 가능한 빨리 화면에 보여줘야 하는 온라인 프로그램에 적당한 조인 기법이다.

① 선행 테이블에서 주어진 조건을 만족하는 행을 찾음

② 선행 테이블의 조인 키 값을 가지고 후행 테이블에서 조인 수행

③ 선행 테이블의 조건을 만족하는 모든 행에 대해 1번 작업 반복 수행

2. Sort Merge Join

조인 칼럼을 기준으로 데이터를 정렬하여 조인을 수행한다. NL Join은 주로 랜덤 액세스 방식으로 데이터를 읽는 반면 Sort Merge Join은 주로 스캔 방식으로 데이터를 읽는다.

Hash Join과 달리 비동등 조인에 대해서도 조인 작업이 가능하다는 장점이 있다.

① 선행 테이블에서 주어진 조건을 만족하는 행을 찾음

② 선행 테이블의 조인 키를 기준으로 정렬 작업을 수행

① ~ ②번 작업을 선행 테이블의 조건을 만족하는 모든 행에 대해 반복 수행

③ 후행 테이블에서 주어진 조건을 만족하는 행을 찾음

④ 후행 테이블의 조인 키를 기준으로 정렬 작업을 수행

③ ~ ④번 작업을 후행 테이블의 조건을 만족하는 모든 행에 대해 반복 수행

⑤ 정렬된 결과를 이용하여 조인을 수행하며 조인에 성공하면 추출버퍼에 넣음

3, Hash Join

해쉥 기법을 이용하여 조인을 수행한다. 조인을 수행할 테이블의 조인 칼럼을 기준으로 해쉬 함수를 수행하여 서로 동일한 해쉬 값을 갖는 것들 사이에서 실제 값이 같은지를 비교하면서 조인을 수행한다.

① 선행 테이블에서 주어진 조건을 만족하는 행을 찾음

② 선행 테이블의 조인 키를 기준으로 해쉬 함수를 적용하여 해쉬 테이블을 생성

→ 조인 칼럼과 SELECT 절에서 필요로 하는 칼럼도 함께 저장됨

① ~ ②번 작업을 선행 테이블의 조건을 만족하는 모든 행에 대해 반복 수행

③ 후행 테이블에서 주어진 조건을 만족하는 행을 찾음

④ 후행 테이블의 조인 키를 기준으로 해쉬 함수를 적용하여 해당 버킷을 찾음

→ 조인 키를 이용해서 실제 조인될 데이터를 찾음

⑤ 조인에 성공하면 추출버퍼에 넣음

③ ~ ⑤번 작업을 후행 테이블의 조건을 만족하는 모든 행에 대해서 반복 수행

동등 조인에서만 사용할 수 있다.