**1과목 데이터 모델링의 이해**

**1-1 데이터 모델링의 이해**

**1-1-1 데이터 모델링의 이해**

모델링의 특징

1. 현실세계를 일정한 형식에 맞추어 표현하는 추상화의 의미를 가질 수 있음
2. 복잡한 현실을 제한된 언어나 표기법을 통해 이해하기 쉽게 하는 단순화의 의미를 가지고 있음
3. 애매모함을 배제하고 누구나 이해가 가능하도록 정확하게 현상을 기술하는 정확화의 의미를 가짐
4. 시스템 구현만을 위해 진행하지 않음, 시스템 구현을 포함한 업무분석 및 업무형상화를 하는 목적 있음

데이터 모델링이 필요한 이유

1. 업무정보를 구성하는 기초가 되는 정보들에 대해 일정한 표기법에 의해 표현
2. 분석된 모델을 가지고 데이터베이스를 생성하여 개발 및 데이터관리에 사용하기 위한 것
3. 데이터모델링 자체로서 업무의 흐름을 설명하고 분석하는 부분에 의미를 가짐

데이터 모델링 할 때 유의점

1. 중복성 : 여러 장소의 데이터베이스에 같은 정보를 저장하지 않도록 하여 중복성을 최소
2. 비유연성 : 데이터의 정의를 데이터의 사용 프로세스와 분리하여 유연성을 높인다
3. 비일관성 : 데이터간의 상호 연관관계를 명확하게 정의하여 일관성 있게 데이터가 유지되도록 한다

데이터 모델링 개념

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

데이터베이스 스키마 구조 3단계

1. ~~외부스키마 :~~
2. 개념스키마 :
   1. 모든 사용자 관점을 통합한 조직 전체 관점의 통합적 표현
   2. 모든 응용시스템이나 사용자들이 필요로 하는 데이터를 통합한 조직 전체의 DB를 기술한 것으로 DB에 저장되는 데이터와 그들간의 관계를 표현하는 스키마
3. ~~내부스키마 :~~

ERD 설명

1. 1976년 피터첸에 의해 E-R모델 표기법이 만들어짐
2. 엔터티를 어디에 배치하는가에 대한 문제는 필수사항이 아니다

ERD 작성 순서

1. 엔터티를 그린다
2. 엔터티를 적절하게 배치
3. 엔터티간 관계를 설정
4. 관계명을 기술
5. 관계의 참여도를 기술
6. 관계의 필수여부를 기술

**1-1-2 엔터티**

엔터티의 특징

1. 속성이 없는 엔터티는 있을 수 없다.(엔터티는 반드시 속성을 가져야 한다)
2. 엔터티는 다른 엔터티와 관계가 있을 수 밖에 없다.(단, 통계성 코드성 엔터티의 경우 생략 가능)
3. 데이터로서 존재하지만 업무에서 필요로 하지 않으면 해당 업무의 엔터티로 성립될 수 없다.
4. ~~반드시 해당 업무에서 필요하고 관리하고자 하는 정보이어야 한다.~~
5. 유일한 식별자에 의해 식별이 가능해야 한다
6. ~~영속적으로 존재하는 (2개 이상의)인스턴스의 집합이어야 한다~~
7. 엔터티는 업무 프로세스에 의해 이용되어야 한다
8. 엔터티는 반드시 속성이 있어야 한다
9. ~~엔터티는 다른 엔터티와 최소 한 개 이상의 관계가 있어야 한다.~~

기본 엔터티(키엔터티)

1. 다른 엔터티로부터 주식별자를 상속받지 않고 자신의 고유한 주식별자를 가진다

엔터티의 이름을 부여하는 방법

1. 현업의 업무 용어를 사용하여 업무상의 의미를 분명하게 한다
2. 모든 엔터티에서 유일한 이름을 부여
3. 엔터티가 생성되는 의미대로 자연스럽게 부여
4. ~~가능하면 약어를 사용하지 않는다~~
5. ~~단수명사를 사용~~

**1-1-3 속성**

속성(ATTRIBUTE)

1. 업무에서 필요로 하는 인스턴스에 관리하고자 하는 의미상 더 이상 분리되지 않는 최소의 데이터 단위

속성의 특징

1. 엔터티에 대한 자세하고 구체적인 정보를 나타낸다
2. 하나의 엔터티는 두 개 이상의 속성을 갖는다
3. 속성도 집합이다
4. 하나의 인스턴스에서 각각의 속성은 한 개의 속성값을 가져야 한다

속성의 특성에 따른 분류

1. 기본속성 : ~~기본적인 베이스~~
2. 설계속성 : ~~계산된 속성~~
3. 파생속성 : ~~정의된 속성~~

도메인

1. 가질 수 있는 값의 범위
2. 엔터티 내에서 속성에 대한 데이터타입과 크기, 제약사항을 지정

속성의 명칭 부여

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

**1-1-4 관계**

데이터모델링의 관계 설명

1. 존재적 관계와 행위에 의한 관계를 구분하는 표기법이 없다
2. 존재에 의한 관계와 행위에 대한 관계로 구분될 수 있으나 ERD에서는 관계를 연결할 때, 존재와 행위를 구분하지 않고 단일화된 표기법을 사용
3. UML에서는 연관관계와 의존관계에 대해 다른 표기법을 가지고 표현한다
4. UML에는 클래스다이어그램 관계 중 연관관계와 의존관계가 있고 이것은 실선과 점선의 표기법으로 다르게 표현
5. 관계는 존재적 관계와 행위에 의한 관계로 나누어볼 수 있다.
6. 부서와 사원 엔터티 간의 ‘소속’관계는 존재적 관계의 사례
7. 주문과 배성 엔터티 간의 ‘배송근거’관계는 행위에 의한 관계의 사례
8. 관계 표기법은 관계명, 관계차수, 선택성 3가지 개념으로 표현

관계 표기법

1. 관계명
2. 관계차수 : 1:1, 1:M, M:N
3. 관계선택사항

관계 읽기

1. 두 개의 엔터티 사이에 관심있는 연관규칙이 존재?
2. 두 개의 엔터티 사이에 정보의 조합이 발생?
3. 업무기술서, 장표에 관계연결에 대한 규칙이 서술?
4. 업무기술서, 장표에 관계연결을 가능하게 하는 동사가 있는가?

**1-1-5 식별자**

식별자의 종류

1. ~~엔터티 내에서 대표성을 가지는가에 따라 (주, 보조)~~
2. ~~엔터티 내에서 스스로 생성되었는지 여부에 따라 (내부, 외부)~~
3. ~~단일 속성으로 식별되는가에 따라 (단일, 복합)~~
4. ~~업무적으로 의미가 있던 식별자 속성을 대체하여 새롭게 만든 식별자를 구분 (본질, 인조)~~

주식별자를 지정할 때 고려사항

1. 주식별자에 의해 엔터티 내의 모든 인스턴스들이 유일하게 구분
2. 주식별자를 구성하는 속성의 수는 유일성을 만족하는 최소의 수가 되어야 한다
3. 지정된 주식별자의 값은 자주 변하지 않는 것이어야 한다
4. 주식별자가 지정이 되면 반드시 값이 들어와야 한다.

주식별자의 특징

1. 유일성 : 주식별자에 의해 엔터티내에 모든 인스턴스들을 유일하게 구분
2. 최소성 : 주식별자를 구성하는 속성의 수는 유일성을 만족하는 최소의 수가 되어야 함
3. 불변성 : 주식별자가 한 번 특정 엔터티에 지정되면 그 식별자의 값은 변하지 않아야 함
4. 존재성 : 주식별자가 지정되면 반드시 데이터 값이 존재 (Null 안됨)

주식별자를 도출하기 위한 기준

1. 해당 업무에서 자주 이용되는 속성
2. ~~명칭, 내영 등과 같이 이름으로 기술되는 것들은 가능하면 지정하지 않는다.~~
3. 복합으로 주식별자로 구성할 경우 너무 많은 속성이 포함되지 않도록 한다

비식별자관계를 선택하는 기준

1. 관계의 강약을 분석하여 상호간의 연관성이 약할 경우 비식별관계를 고려
2. 자식테이블에서 독립적인 primary Key의 구조를 가지기 원할 때 비식별자관계를 고려
3. 모든 관계가 식별자 관계로 연결되면 SQL Where절에서 비교하는 항목이 증가되어 조인에 참여한 테이블에 따라 SQL문장이 길어져 SQL문의 복잡성이 증가되는 것을 방지하기 위해 비식별자관계를 고려

비식별자관계로 연결하는 것을 고려

1. 부모엔터티에 참조값이 없어도 자식엔터티의 인스턴스가 생성될 수 있다.
2. 여러 개의 엔터티를 하나로 통합하면서 각각의 엔터티가 갖고 있던 여러 개의 개별 관계가 통합
3. 자식쪽 엔터티의 주식별자를 부모엔터티와는 별도로 생성하는 것이 더 유리하다고 판단하는 경우
4. ~~부모엔터티의 인스턴스가 자식과 관계를 가지고 있지만 자식만 남겨두고 먼저 소멸될 경우~~

**1-2 데이터 모델과 성능**

**1-2-1 성능 데이터 모델링의 개요**

성능 데이터모델링

1. 데이터의 증가가 빠를수록 성능저하에 따른 성능개선비용은 증가한다
2. 데이터모델은 성능을 튜닝 하면서 변경이 될 수 있는 특징이 있다
3. 분석/설계 단계에서 성능을 고려한 데이터모델링을 수행할 경우 성능저하에 따른 Rework 비용을 최소화 할 수 있는 기회를 가지게 된다

데이터 모델링 수행 절차

1. 데이터모델링을 할 때 정규화를 정확하게 수행
2. 데이터베이스 용량산정을 수행
3. 데이터베이스에 발생되는 트랜잭션의 유형을 파악
4. 용량과 트랜잭션의 유형에 따라 반정규화를 수행
5. 이력모델의 조정, PK/FK조정, 슈퍼타입/서브타입 조정 등을 수행

성능 데이터 모델링 할 때 고려사항

1. 용량산정 : 전체적인 데이터베이스에 발생되는 트랜잭션의 유형과 양을 분석하는 자료가 되므로 성능데이터 모델링을 할 때 중요한 작업이 될 수 있다.
2. 물리적인 데이터 모델링 : PK/FK의 칼럼의 순서조정, FK인덱스 생성 등은 성능 향상을 위한 데이터 모델링 작업에 중요한 요소가 된다
3. 이력데이터 : 시간에 따라 반복적으로 발생이 되기 때문에 대량 데이터일 가능성이 높아 특별히 성능을 고려하여 칼럼 등을 추가하도록 설계해야 한다.

**1-2-2 정규화와 성능**

25p : 35번 ~ 26p : 39번

**1-2-3 반정규화와 성능**

반정규화를 고려할 때 판단요소

1. 반정규화 정보에 대한 재현의 적시성으로 판단

디스크 I/O를 줄이기 위한 반정규화 기법

1. 테이블추가 – 부분테이블추가

|  |  |  |
| --- | --- | --- |
| 테이블의 반정규화 | | |
| 기법 분류 | 반정규화 기법 |  |
| 테이블병합 | 1:1 관계 테이블 병합 |  |
| 1:M 관계 테이블 병합 |  |
| 슈퍼/서브타입 테이블 병합 |  |
| 테이블분할 | 수직분할 |  |
| 수평분할 |  |
| 테이블추가 | 중복테이블 추가 |  |
| 통계테이블 추가 |  |
| 이력테이블 추가 |  |
| 부분테이블 추가 |  |
| 칼럼의 반정규화 | | |
|  | 중복칼럼 추가 | 조인감소를 위해 여러 테이블에 동일한 칼럼을 갖는다 |
|  | 파생칼럼 추가 | 조회 성능을 우수하게 하기 위해 미리 계산된 칼럼을 갖는다 |
|  | 이력테이블 칼럼추가 | 최신값을 처리하는 이력의 특성을 고려하여 기능성 칼럼 추가 |
|  | PK에 의한 칼럼 추가 |  |
|  | 응용시스템 오작동을 위한 칼럼 추가 |  |

**1-2-4 대량 데이터에 따른 성능**

28p : 43번 ~ 29p : 44번

**1-2-5 데이터베이스 구조와 성능**

칼럼수가 많은 테이블에 대한 설명

1. 로우체이닝이 발생할 정도로 한 테이블에 많은 칼럼들이 존재할 경우 조회성능저하가 발생할 수 있다.
2. 트랜잭션이 접근하는 칼럼유형을 분석하여 1:1로 테이블을 분리하면 디스크 I/O가 줄어들어 조회 성능을 향상 시킬 수 있다

데이터 액세스 성능을 향상시키기 위한 방법

1. 하나의 테이블에 많은 양의 데이터가 저장되면 인덱스를 추가하고 테이블을 몇 개로 쪼개도 성능이 저하되는 경우가 있다.
2. 이때 논리적으로 하나의 테이블이지만 물리적으로는 여러 개의 테이블로 분리하여 데이터 액세스 성능도 향상시키고, 데이터 관리방법도 개선할 수 있도록 테이블에 적용하는 기법을 파티셔닝이라고 한다

31p : 47번 ~ 34p : 51번

논리데이터모델 > 물리적인 테이블 형식으로 변환

1. 트랜잭션 : 항상 전체를 대상으로 일괄 처리하는데 테이블은 서브타입 별로 개별 유지하는 것으로 변환하면 Union 연산에 의해 성능이 저하 될 수 있다.
2. 트랜잭션 : 항상 서브타입 개별로 처리하는데 테이블은 하나로 통합하여 변환하면 불필요하게 많은 양의 데이터가 집적되어 있어 성능이 저하될 수 있다.
3. 트랜잭션 : 항상 슈퍼+서브 타입을 함께 처리하는데 개별로 유지 하면 조인에 의해 성능이 저하될 수 있다.

**1-2-6 분산 데이터베이스와 성능**

분산 데이터베이스의 분산설계를 하여 효율성을 증대

1. 공통코드, 기준정보 등 마스터 데이터는 분산데이터베이스에 복제분산을 적용
2. 거의 실시간 업무적인 특성을 가지고 있을 때 분산데이터베이스를 사용하여 구성할 수 있다.
3. 백업 사이트를 구성할 때 간단하게 분산기능을 적용하여 구성할 수 있다.

**2과목 SQL 기본 및 활용**

**2-1 SQL 기본**

**2-1-1 관계형 데이터베이스 개요**

DDL / 데이터 조작어의 명령어

1. CREATE, ALTER, DROP, ~~RENAME~~
2. 테이블의 구조를 생성, 변경, 삭제하는 등 데이터 구조를 정의하는데 사용되는 명령어
3. 스키마, 도메인, 테이블, 뷰, 인덱스를 정의하거나 변경 또는 제거할 때 사용

DML의 명령어

1. SELECT, INSERT, DELETE, UPDATE
2. 데이터베이스 사용자가 응용 프로그램이나 질의어를 통하여 저장된 데이터베이스를 실질적으로 접근하는데 사용
3. 호스트 프로그램 속에 삽입되어 사용되는 명령어들을 데이터 부속어 라고 한다

DCL / 데이터 제어어의 명령어

1. GRANT, REVOKE

TCL

1. Commit, Rollback, Savepoint
2. 논리적인 작업의 단위를 묶어 DML에 의해 조작된 결과를 작업단위 별로 제어
3. 일부에서 DCL로 분류하기도 한다

NULL

1. 모르는 값
2. 값의 부재
3. NULL과 모든 비교(IS NULL 제외)는 알 수 없음(Unknown)을 반환

**2-1-2 DDL**( CREATE, ALTER, DROP, ~~RENAME, TRUNCATE )~~

CREATE TABLE 테이블이름 ( 속성이름 자료형 제약조건, 반복 ) ;

CONSTRAINT : 기본키를 지정할 때 사용

* CONSTRAINT 제약조건이름 PRIMARY KEY ( PK할속성이름 )

ALTER TABLE 테이블이름 ALTER COLUMN 속성이름 변경자료형 변경제약조건 ;

ALTER TABLE 테이블이름 ADD 추가속성이름 추가제약조건 ;

ALTER TABLE 테이블이름 ADD CONSTRAINT 그냥이름 PRIMARY KEY ( PK할속성이름 ) ;

DROP TABLE 테이블이름 ;

DROP COLUMN 속성이름 ;

CREATE INDEX 인덱스이름 ON 테이블이름 ( 속성이름 ) ;

RENAME 테이블이름 TO 변경테이블이름 ;

TRUNCATE : DROP과 차이점 = DROP은 테이블을 완전 삭제, TRUNCATE은 테이블을 완전초기상태로 되돌림

TRUNCATE TABLE 테이블이름 ; / 테이블의 모든 데이터를 삭제하는 명령문

**2-1-3 DML**( SELECT, INSERT, DELETE, UPDATE )

SELECT \* FROM 테이블이름 ;

INSERT INTO 테이블이름 ( 속성이름1, 속성이름2, 속성이름3 ) VALUES ( 튜플1, 튜플2, 튜플3 );

DELETE FROM 테이블이름 ;

UPDATE 테이블이름 SET 속성이름 = 튜플 ;

옵션

1. PK 주키 : 테이블당 1개만 생성 가능
2. FK 외래키 : 테이블당 여러 개 생성 가능, 테이블 생성시 설정 가능, 참조 무결성 제약을 받는다
3. UNIQUE : 테이블 내에서 중복 불가, NULL 가능
4. NOT NULL : 명시적으로 NULL 입력을 방지
5. DISTINCT : 데이터의 중복을 제거하는 명렁어
6. DASCADE : 참조된 것도 삭제됨
7. REFERENCES : FK를 설정할 때, 다른 테이블의 PK를 참조하는 제약조건
   1. FORIGN KEY (FK속성이름) REFERENCES 다른테이블이름(PK속성이름) ON DELETE CASCADE ;

테이블명, 속성명(칼럼명)

1. 문자로 시작
2. 특수문자 \_, $, # 만 가능

**2-1-4 TCL**( Commit, Rollback, Savepoint )

트랜잭션의 4가지 특성(목표)

1. 원자성 : 트랜잭션에서 정의된 연산들은 모두 성공적으로 실행 또는 전혀 실행되지 않은 상태 일 것
2. 고립성 : 트랜잭션이 실행되는 도중에 다른 트랜잭션의 영향을 받아 잘못된 결과를 만들어서는 안 된다
3. ~~일관성 : 트랜잭션이 실행 되기 전의 데이터베이스 내용이 잘못 되어 있지 않다면 트랜잭션이 실행된 이후에도 데이터베이스의 내용에 잘못이 있으면 안 된다~~
4. ~~지속성 : 트랜잭션이 성공적으로 수행되면 그 트랜잭션이 갱신한 데이터베이스의 내용은 영구적 저장~~

격리성이 낮은 경우 발생할 수 있는 문제점

1. Non-Repeatable Read : 한 트랜잭션 내에서 같은 쿼리를 두 번 수행했는데, 그 사이에 다른 트랜잭션이 값을 수정 또는 삭제하는 바람에 두 쿼리 결과가 다르게 나타나는 현상
2. Phantom Read : 한 트랜잭션 내에서 같은 쿼리를 두 번 수행했는데, 첫 번째 쿼리에서 없던 유령 레코드가 두 번째 쿼리에서 나타나는 현상을 말한다
3. Dirty Read : 다른 트랜잭션에 의해 수정되었지만 아직 커밋되지 않은 데이터를 읽는 것을 말한다.

Oracle, SQL Server 차이 : DDL사용시 Oracle은 auto commit 된다, SQL Server는 안 된다

트랜잭션 transaction : 데이터베이스의 논리적 연산단위로서 밀접히 관련되어 분리될 수 없는 한 개 이상의 데이터베이스 조작을 가리킨다

트랜잭션의 종료를 위한 대표적 명령어로서는 데이터에 대한 변경사항을 데이터베이스에 영구적으로 반영하는 커밋 commit과 데이터에 대한 변경사항을 모두 폐기하고 변경전의 상태로 되돌리는 롤백 rollback이 있다

BEGIN TRANSACTION ; / 트랜잭션 단위로 시작하며 COMMIT 또는 ROLLBACK 문을 사용하여 트랜잭션을 종료

SAVE TRANSACTION 저장이름 ; / SAVA포인트 만듦

ROLLBACK TRANSACTION 저장이름 ; / 롤백 할 때 SAVE포인트로 롤백

**2-1-5 WHERE절**

WHERE : SQL을 사용하여 데이터베이스에서 데이터를 조회할 때 원하는 데이터만을 검색하기 위해서 SELECT, FROM 절과 함께 이용하여 조회되는 데이터의 조건을 설정하여 데이터를 제한할 수 있다.

IS : 조건절에서 NULL값을 사용하는 경우, 앞에 IS 키워드를 사용해야 한다.

**2-1-6 함수**

함수

1. 내장 함수 : 함수의 입력 행수에 따라 구분
   1. 단일행 함수 : ~~SELECT, WHERE, ORDER BY, UPDATE의 SET 절에 사용 가능~~
   2. 다중행 함수 : ~~다중행 함수와 동일하게단일 값만을 반환~~
      1. 집계함수
      2. 그룹함수
      3. 윈도우 함수
2. 사용자 함수

LENGTH : 문자열의 길이를 반환하는 함수

CHR : 주어진 ASCII 코드에 대한 문자를 반환하는 함수

REPLACE : 문자열을 치환하는 함수

NVL( 표현식, 대체값 ) : 표현식이 NULL인 경우 반환될 대체값

SELECT ISNULL( 표현식, ‘대체값’ ) FROM … ; / 표현식이 NULL인 경우 반환될 대체값

NULLIF( 표현식1, 표현식2 ) : 표현식1 = 표현식2 -> NULL로 반환, 표현식1 != 표현식2 -> 표현식1을 반환

COALESCE( 표현식1, 표현식2, ... ) : 첫 번째 NULL이 아닌 값을 반환

**2-1-7 Group By, Having절**

66p : 55번 ~ 70p : 56번

~~GROUP BY, HAVING 특성~~

1. ~~GROUP BY절을 통해 소그룹별 기준을 정한 후, SELECT 절에 집계 함수를 사용~~
2. ~~집계 함수의 통계 정보는 NULL값을 가진 행을 제외하고 수행~~
3. ~~GROUP BY절에서는 SELECT절과는 달리 ALIAS명을 사용할 수 없다~~
4. ~~집계 함수는 WHERE절에는 올 수 없다.~~
5. ~~WHERE절은 전체 데이터를 GROUP으로 나누기 전에 행들을 미리 제거~~
6. ~~HAVING절은 GROUP BY절의 기준 항목이나 소그룹의 집계함수를 이용한 조건을 표시할 수 있다.~~
7. ~~GROUP BY절에 의한 소그룹별로 만들어진 집계 데이터 중, HAVING절에서 제한 조건을 두어 조건을 만족하는 내용만 출력~~
8. ~~HAVING절은 일반적으로 GROUP BY절 뒤에 위치한다~~

**2-1-8 Order By절**

ORDER BY 특징

1. ~~기본적인 정렬 순서는 오름차순(ASC)~~
2. SQL 문장으로 조회된 데이터들을 다양한 목적에 맞게 특정 컬럼을 기준으로 정렬하는데 사용
   1. ~~ASC의 경우 숫자 > 대문자 > 소문자 순서로 정렬~~
   2. ~~숫자형 데이터 타입 : 오름차순으로 정렬했을 경우 가장 작은 값부터 출력~~
   3. ~~날짜형 데이터 타입 : 오름차순으로 정렬했을 경우 날짜값이 가장 빠른 값이 먼저 출력~~
3. DBMS마다 NULL값에 대한 정렬 순서가 다를 수 있으므로 주의
   1. ~~Oracle : NULL값을 가장 큰 값으로 간주하여 오름차순으로 정렬했을 경우에는 가장 마지막에 출력~~
   2. ~~SQL Server : NULL값을 가장 작은 값으로 간주하여 오름차순으로 정렬했을 경우 가장 먼저 출력~~
4. GROUP BY절을 사용하는 경우 ORDER BY절에 집계 함수를 사용할 수도 있다.
5. ORDER BY절에서 컬럼명 대신 Alias명이나 컬럼 순서를 나타내는 정수를 혼용하여 사용 가능

SELECT 문장의 실행 순서 : FROM – WHERE – GROUP BY – HAVING – SELECT – ORDER BY

SELECT TOP(개수) WITH TIES 속성명 FROM … ORDER BY 속성명 [ASC | DESC] ;

1. ORDER BY에 대한 상위 개수만큼 출력
2. 개수에 해당하는 값이 동일한 경우 함께 출력되도록 하는 WITH TIES 옵션을 사용한다.

**2-1-9 조인(JOIN)**

73p : 61번 ~ 74p : 64번

Join 특징

1. 일반적으로 Join은 PK와FK값의 연관성에 의해 성립된다
2. EQUI Join에 관여하는 테이블 간의 컬럼 값들이 정확하게 일치하는 경우에 사용되는 방법
3. EQUI Join은 ‘=’연산자에 의해서만 수행되며, 그 이외의 비교연산자를 사용하는 경우에는 모두 Non EQUI Join이다
4. 대부분 Non EQUI Join을 수행할 수 있지만, 때로는 설계상의 이유로 수행이 불가능한 경우도 있다.
5. DBMS 옵티마이저는 FROM절에 나열된 테이블이 아무리 많아도 항상 2개의 테이블씩 짝을 지어 Join을 수행

LIKE 연산자

1. % : 0개 이상의 임의의 문자열을 나타냄
2. \_ : 단일 임의의 문자를 나타냄

**2-2 SQL 활용**

**2-2-1 표준 조인**

순수 관계 연산자

1. SELECT, ~~PROJECT~~, JOIN, DIVIDE

FROM 1, 2 = FROM 1 CROSS JOIN 2

~~INNER JOIN : INNER 생략 불가, 우측 테이블에 행이 없다면 결과에 포함되지 않는다~~

OUTER JOIN : OUTER 생략 가능, ~~우측 테이블에 행이 없어도 결과에 포함~~

1. 좌측 LEFT OUTER JOIN 우측 : 좌측을 기준으로 연결되는 테이블을 만든다
2. 좌측 RIGHT OUTER JOIN 우측: 우측을 기준으로 연결되는 테이블을 만든다
3. 좌측 FULL OUTER JOIN 우측: 좌우측 합쳐서 연결되는 테이블을 만든다

WHERE NOT EXISTS (서브쿼리) : 서브쿼리을 제외한 나머지를 출력

**2-2-2 집합 연산자**( UNION, INTERSECT, EXCEPT )

UNION 합집합 : 결과에서 모든 중복된 행은 하나의 행으로 만든다 / 정렬 발생

UNION ALL 합집합 : 중복된 행도 그대로 결과를 표시 / 정렬 안함

INTERSECT 교집합 : 중복된 행은 하나의 행으로 만든다

EXCEPT / MINUS 차집합 : 중복된 행은 하나의 행으로 만든다 / NOT IN, NOT EXISTS 대체 가능

**2-2-3 계층형 질의와 셀프 조인**

Oracle 계층형 질의

1. START WITH절 계층 구저의 시작점을 지정하는 구문
2. ORDER SIBLINGS BY절 : 형제 노드 사이에서 정렬을 지정하는 구문
3. 순방향전개 : 부모 노드 -> 자식 노드 방향으로 전개
4. 루트 노드의 LEVEL값 = 1
5. WHERE절 : 모든 전개를 진행한 이후 필터 조건으로서 조건을 만족하는 데이터만을 추출하는데 활용
6. PRIOR키워드 : SELECT, WHERE절에서도 사용할 수 있다.

SQL Server 계층형 질의

1. CTE을 재귀 호출함으로써 계층 구조를 전개
2. 앵커 맴버를 실행하여 기본 결과집합을 만들고 이후 재귀 맴버를 지속적으로 실행

START WITH절 : 계층 구조의 시작을 지정하며, WHERE절에서는 제외될 내용도 포함이 된다.

CONNECT BY PRIOR절 : 좌측 = 우측 일 때, 우측 참조 -> 좌측 참조한다

SELF JOIN : 한 테이블 내에서 두 칼럼이 연관 관계가 있을 때 사용

**2-2-4 서브쿼리**

Single Row : 서브쿼리의 실행 결과가 항상 1건 이하 / 비교연산자 : =, <, <=, >, >=, <>

Multi Row : 서브쿼리의 실행 결과가 여러 건인 서브쿼리 / 비교연산자 : IN, ALL, ANY, SOME, EXISTS

Multi Column : 서브쿼리의 실행 결과로 여러 칼럼을 반환

서브쿼리 특징

1. 단일 행 또는 복수 행 비교연산자와 함께 사용할 수 있다.
2. SELECT절, FROM절, HAVING절, ORDER BY절 등에서 사용이 가능
3. 연관 서브쿼리 = 서브쿼리가 메인 쿼리 컬럼을 포함하고 있는 형태의 서브쿼리이다
4. 다중 행 서브쿼리 비교 연산자는 단 일행 서브쿼리의 비교연산자로 사용이 가능
5. FROM절에 사용되는 서브쿼리는 Inline View, Dynamin View라고 부른다
6. ~~결과가 복수 행 결과를 반환하는 경우에는 IN, ALL, ANY 등의 복수 행 비교연산자와 사용하여야 한다~~
7. ~~다중 컬럼 서브쿼리 = 서브쿼리의 결과로 여러 개의 컬럼이 반환되어 메인 쿼리의 조건과 비교되는데, SQL Server에서는 현재 지원하지 않는 기능이다~~

뷰의 특징

1. 단지 정의만을 가지고 있으며, 실행 시점에 질의를 재작성하여 수행
2. 보안을 강화하기 위한 목적으로 활용할 수 있다.
3. 실제 데이터를 저장하고 있는 뷰를 생성하는 기능을 지원하는 DBMS도 있다.

**2-2-5 그룹 함수**

GROUP BY ROLLUP (속성) :

1. 집계함수가 있으면 각 그룹별로 집계하여 출력에 사용
2. ~~함수의 인자로 주어진 컬럼의 순서에 따라 다른 결과를 추출하게 되는 그룹 함수~~
3. ~~나열된 컬럼에 대해 계층 구조로 집계를 출력~~

~~GROUP BY CUBE (속성) :~~

1. ~~인자로 주어진 컬럼의 결합 가능한 모든 조합에 대해서 집계를 수행하므로 다른 그룹 함수에 비해 시스템에 대한 부하가 크다~~

~~CUBE, GROUPING SETS, ROLLUP~~

1. ~~일반 그룹함수로 동일한 결과를 추출할 수 있다~~
2. ~~함수들에 의해 집계된 레코드에서 집계 대상 컬럼 이외의 GROUP 대상 컬럼의 값은 NULL을 반환~~

**2-2-6 윈도우 함수**

윈도우 함수 특징

1. Partition과 Group By 구문은 의미적으로 유사
2. Partition 구문이 없으면 전체 집합을 하나의 Partition으로 정의한 것과 동일
3. 윈도우 함수 적용 범위는 Partition을 넘을 수 없다
4. ~~결과에 대한 함수처리이기 때문에 결과 건수는 줄지 않는다~~

RANK() :

1. 특정 컬럼에 대한 순위를 구하는 함수
2. 동일한 값 = 동일한 순위

ROW\_NUMBER() :

1. 동일한 값 != 동일한 순위
2. 동일한 값 = 고유한 순위

**2-2-7 DCL**( GRANT, REVOKE )

DBMS에 생성된 USER의 다양한 권한들 사이에서 중개 역할을 할 수 있도록 DBMS에서는 ROLE을 제공한다

ROLE

1. DBMS 사용자를 생성하면 기본적으로 많은 권한을 부여해야 한다
2. 많은 DBMS에서는 DMBS 관리자가 사용자 별로 권한을 관리해야 하는 부담과 복잡함을 줄이기 위하여 다양한 권한을 그룹으로 묶어 관리할 수 있도록 사용자와 권한 사이에서 중개 역할을 수행

GRANT

1. GRANT 권한1, 권한2 … ON 테이블 TO 사용자 ;
2. GRANT 권한1, 권한2 … TO 사용자 ;
3. ROLE을 DBMS USER에게 부여하기 위해서 사용

REVOKE

1. REVOKE 권한1, 권한2, ... ON 객체 FROM 사용자 ;
2. REVOKE 권한1, 권한2, ... FROM 사용자 ;
3. ROLE을 회수하기 위해서 사용

**2-2-8 절차형 SQL**

PL/SQL 특징

1. 변수와 상수 등을 사용하여 일반 SQL 문장을 실행할 때 WHERE절의 조건 등으로 대입할 수 있다
2. Procedure, User Defined Function, Trigger 객체를 PL/SQL로 작성 가능
3. Procedure 내부에 작성된 절차적 코드는 PL/SQL엔진이 처리
4. 일반적인 SQL 문장은 SQL실행기가 처리

SQL 모듈 특징

1. 저장형 프로시져는 SQL을 로직과 함께 데이터베이스 내에 저장해 놓은 명령문의 집합
2. 저장형 함수(사용자 정의 함수)는 단독적으로 실행되기 보다는 다른 SQL문을 통하여 호출되고 그 결과를 리턴하는 SQL의 보저적인 역할을 한다
3. 트리거는 특정한 테이블에 INSERT, UPDATE, DELETE와 같은 DML문이 수행되었을 때 데이터베이스에서 자동으로 동작하도록 작성된 프로그램이다.

Trigger 트리거

1. 데이터베이스에 의해서 자동으로 호출되고 수행
2. 특정 테이블에 대해서 INSERT, UPDATE, DELETE문이 수행되었을 때 호출되도록 정의
3. 데이터베이스에 로그인하는 작업에도 정의
4. TCL 사용 불가

**2-3 SQL 최적화 기본 원리**

**2-3-1 옵티마이저와 실행계획**

CBO, 비용기반 옵티마이저

1. 테이블 및 인덱스 등의 통계 정보를 활용하여 SQL문을 실행하는데 소요될 처리시간 및 CPU, I/O 자원량 등을 계산하여 가장 효율적일 것으로 예상되는 실행계획을 선택하는 옵티마이저

실행계획 읽는 순서

1. 안에서 밖으로
2. 위에서 아래로

실행계획을 통해 알 수 있는 정보

1. 액세스 기법
2. 질의 처리 예상 비용
3. 조인 순서

실행계획 특징

1. SQL 처리를 위해 실행 절차와 방법을 표현
2. 조인 방법, 조인 순서, 액세스 기법 등이 표현
3. CBO의 실행계획에는 단계별 예상 비용 및 건수 등이 표시
4. 실행방법이 달라진다고 결과가 달라지지 않는다

~~트레이스 정보~~

1. ~~실제 처리 건수~~

SQL 처리 흐름도 특징

1. 인덱스 스캔, 테이블 전체 스캔 등과 같은 액세스 기법이 표현
2. SQL의 내부적인 처리 절차를 시각적으로 표현

옵티마이저와 실행계획 특징

1. 규칙기반 옵티마이저에서 제일 높은 우선순위는 행에 대한 고유 주소를 사용하는 방법
2. SQL 처리 흐름도는 인덱스 스캔 및 전체 테이블 스캔 등의 액세스 기법을 표현
3. Oracle의 규칙기반 옵티마이저에서 가장 우선 순위가 높은 규칙은 Single row by rowed 액세스 기법
4. 비용기반 옵티마이저는 테이블, 인덱스, 컬럼 등 객체의 통계정보를 사용하여 실행계획을 수립하므로 통계정보가 변경되면 SQL의 실행계획이 달라질 수 있다
5. Oracle의 실행계획에 나타나는 기본적인 Join기법으로 NL Join, Hash Join, Sort Merge Join 등이 있다.

**2-3-2 인덱스 기본**

인덱스 특징

1. 기본 인덱스에 널 값들이 나타날 수 없다
2. 테이블의 전체 데이터를 읽는 경우는 인덱스가 거의 불필요
3. B트리는 관계형 데이터베이스의 주요 인덱스 구조이다.
4. 규칙기반 옵티마이저는 적절한 인덱스가 존재하면 항상 인덱스를 사용하려고 한다
5. 인덱스 범위 스캔은 결과가 없으면 한 건도 반환하지 않을 수 있다
6. 인덱스의 목적은 조회 성능을 최적화
7. INSERT, UPDATE, DELETE 등의 DML 처리 성능을 저하시킬 수도 있다(부하를 줄 수 있다)
8. INSERT, DELETE와 다르게 UPDATE는 부하가 없을 수 있다
9. B-트리 인덱스는 일치 및 범위 검색에 적절한 구조

B-TREE 인덱스 :

1. 브랜치 블록과 리프 블록으로 구성
2. 브랜치 블록 = 분기를 목적
3. 리프블록 = 인덱스를 구성하는 컬럼의 값으로 정렬
4. OLTP 시스템에서 가장 많이 사용

CLUSTERED 인덱스 :

1. 인덱스의 리프 페이지 = 데이터 페이지
2. 리프 페이지의 모든 데이터는 인덱스 키 컬럼 순으로 물리적으로 정렬되어 저장

BITMAP 인덱스 :

1. DW 및 AD-HOC 질의 환경을 위해서 설계
2. 하나의 인덱스 키 엔트리가 많은 행에 대한 포인터를 저장하고 있는 구조

**2-3-3 조인 수행 원리**

Nested Loop Join 특징

1. 조인 칼럼에 적당한 인덱스가 있어서 자연조인이 효율적일 때 유용
2. Driving Table의 조인 데이터 양이 큰 영향을 주는 조인 방식이다
3. 유니크 인덱스를 활용하여 수행시간이 적게 걸리는 소량 테이블을 온라인 조회하는 경우 유용
4. 선택도가 낮은 테이블이 선행 테이블로 선택되는 것이 일반적

SMJ(Sort Merge Join) 특징

1. 조인 칼럼에 적당한 인덱스가 없어서 NL조인가 비효율적일 때 사용할 수 있다
2. Driving Table의 개념이 중요하지 않는 조인 방식
3. 조인 조건의 인덱스의 유무에 영향을 받지 않는다
4. ~~비 동등에서도 사용할 수 있다~~

해시조인 Hash Join

1. 조인 컬럼에 적당한 인덱스가 없어서 자연조인이 비효율적일 때
2. 자연 조인 시 드라이빙 집합 쪽으로 조인 액세스 량이 많아 Random 액세스 부하가 심할 때
3. 소트 머치 조인을 하기에는 두 테이블이 너무 커서 소트 부하가 심할 때
4. ~~행의 수가 작은 테이블을 선행 테이블로 선택하는 것이 유리~~
5. ~~SMJ보다 일반적으로 더 우수한 성능을 보임~~
6. ~~Join 대상 테이블이 Join Key 컬럼으로 정렬되어 있을 때는 SMJ가 더 우수한 성능을 낼 수 있다.~~