

Part1. 데이터베이스 정의

1. 데이터베이스

: 특정 조직이 업무 수행하는 데 필요한 관련성 있는 자료들의 집합체 (통합, 저장, 운영, 공유)

2. 데이터웨어하우스(DataWarehouse)

: 기간 시스템의 데이터베이스에 축적된 데이터를 공통의 형식으로 변환하여 일원적으로 관리하는 데이터베이스.
'웨어하우스'는 창고라는 의미인데 데이터의 격납이나 분석 방법까지 포함하여 조직 내 의사 결정을 지원하는 정보 관리 시스템으로 이용

3. 스키마(Schema) 3계층

*스키마: DB의 구조(개체, 속성, 관계)와 제약조건에 대한 명세(Specification)를 기술한것.

1) 외부스키마(External Schema =서브스키마=사용자뷰)

: 사용자가 보는 관점
(사용자에 따라 다름, 여러 개 존재)

2) 개념 스키마 (Conceptual Schema =스키마= 전체적인, 범기관적, 총괄적 입장): DB 전체적인 논리적 구조

3) 내부 스키마(Internal Schema=물리적 스키마)

: (실제 **Data를 저장**)
- DB 전체적인 물리적 구조
- DBA 관리

4. 데이터베이스 설계(모델링)


: **현실세계**의 업무적인 프로세스를 **컴퓨터세계**로 데이터베이스화 하기 위한 과정.

* 설계순서

요구조건분석→**개념적설계**→**논리적설계**→**물리적설계**→ 구현→ 운영→ 감시 및 개선

1) 개념적 설계

개체 타입과 이들 간의 관계 타입을 이용해 현실 세계를 개념적으로 표현 (**산출물 : 개체관계도 = ER 다이어그램**)
- DBMS에 독립적인 개념 스키마 모델링

구성요소	기호	설명	예시
개체 (Entity)		데이터베이스에 표현하려고	학생, 교수, 학과, 과목

		하는 현실 세계의 대상체	
속성 (Attribute)		개체(Entity)의 성질, 분류, 식별, 수량, 상태 등을 나타내는 세부항목	학생 - 학번, 이름, 전화번호
관계 (Relationship)		두 개체 간에 의미 있는 연결	학생은 과목을 수강한다. 과목은 학생에게 수강되어 진다.

2) 논리적 설계

: 목표 DBMS에 맞추어 논리적 모델로 설계
(관계형, 계층형, 망형 모델)

3)물리적 설계

: 저장레코드 양식의 설계 및 물리적 구조 데이터 표현

5. 관계형 데이터베이스의 릴레이션구조 관련 용어

- 테이블 (**릴레이션**) = 개체 (Entity)
- 기본키, 주기, 주식별자 (Primary key)
- 튜플 (Tuple)= 행
- 속성 (Attribute) = 열, Column
- 릴레이션 스키마(스킴,내연)
 - : 속성 이름들 (릴레이션 틀,구조)
- 릴레이션 인스턴스(**외연**)
 - : 튜플들의 집합 (릴레이션 실제값)
- 도메인 (Domain)
 - : 한 속성에 나타날 수 있는 값들의 범위(집합)
- 차수 (Degree): 속성들의 수
- 카디널리티 (cardinality) : 튜플들의 수
- 널 (Null) : "해당없음" 등의 이유로 정보 부재를 나타내기 위해 사용 하는 특수한 데이터 값
[공백이나 0(zero) X]

Part2. ERD, 키, 무결성

1. ERD 모델 (Entity-Relationship, 개체관계도) 관계 종류

:관계를 맺고 있는 두 개체간에 데이터가 매칭되는 유형을 정의한 것으로 1:1, 1:N, N:M 관계가 있다.
→ 개체들 간에 존재할 수 있는 데이터의 카디널리티(Cardinality)를 파악하여 설정해야 한다.

2. 키(Key)

: 특정 조건에 맞는 튜플을 구분할 수 있는 단일 속성 또는 속성 그룹을 말함.

1) 슈퍼키 : 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키를 말한다. (**유일성**)

2) 후보키 : 한 릴레이션 내에 있는 모든 튜플들을 유일하여 식별할 수 있는 하나 또는 몇 개의 애트리뷰트 집합 (최소 슈퍼키 : **유일성 + 최소성**)

3) 기본키 : 후보키 중에 선택한 키
(**중복되어서는 안되며, Null 값을 가질 수 없다.**)

4) 대체키 : 후보키 중에서 기본키를 제외한 속성들

5) **외래키** : 어떤 R에서 다른 R을 참조할 때 참조 기준이 되는 속성으로서 참조하고자 하는 R의 기본키와 동일

3. 무결성(Integrity)

1)**참조 무결성**: 릴레이션은 참조할 수 없는 **외래키**값을 가질 수 없음을 의미하는 제약 조건

2)**개체 무결성**: 한 릴레이션의 **기본키**를 구성하는 어떠한 속성 값도 널(NULL) 값이나 중복 값을 가질 수 없다.

3)**도메인 무결성**: 각 속성 값은 반드시 정의된 도메인에 속한 값이어야 한다.

Part3. 정보시스템 모델링

1. 정보시스템 모델링

-데이터베이스를 기반으로 정보시스템은 개발됨
-업무분석의 흐름에 따라 **데이터관점**과 **프로세스관점**에서 개발과정이 진행됨

1) 데이터 모델링

현실세계를 데이터의 관점에서 파악하여 개념적인 모델로 표현하는 단계를 말하며 **논리적 데이터베이스 설계**에 해당

2) 프로세스 모델링

-현실세계를 업무의 처리 절차나 흐름의 관점에서 파악하여 **개념적 모델로 표현하는 단계**
-업무가 어떻게 구성되어 있는지, 업무의 처리 절차와 방법이 어떻게 되는지를 파악하는 것으로 **기능분해도와 프로세스 흐름도** 등이 사용

3) 상관 모델링

-데이터 모델링과 프로세스 모델링이 완료되면 **데이터 모델과 프로세스 모델을 비교 검토하여 서로 간에 잘 맞는지를 평가하는 단계**
-**CRUD 매트릭스**가 많이 이용
(CRUD : Create, Read, Update, Delete)

2. 정합성

-무모순성(논리적 모순이 없는 성질이나 상태)
-데이터 무결성 (정확성) + 데이터 통합 → 정합성 → DB 신뢰성 향상 → 신속한 의사결정

3. 속성정의서, 개체정의서, 개체관계정의서

: 모델링과정에서 보다 자세한 내용이 적혀있는 문서로 속성명, 데이터유형, 테이블명 등의 정보가 있음.

4. 데이터 사전 = 자료 사전(data dictionary)

: 데이터베이스 관리 시스템에서 사용되는 모든 파일, 속성, 변수의 목록.
이는 나중에 사용자가 데이터베이스를 사용하거나 그를 이용하여 어떤 프로그램을 코딩할 때 그 데이터베이스 내에 어떤 자료가 있는지 또는 그 자료가 어떻게 정의되어 있는지 등을 쉽게 알 수 있도록 한다.

5. DB관리도구 : DBMS 설치시 사용자와 데이터베이스의 편리하게 조작할 수 있도록 지원하는 여러 도구(프로그램)들이 같이 설치 됨.

6. 응용프로그램(응용시스템)

Part4. 논리적 데이터 모델링

1. Mapping Rule(사상)

: ERD에서 관계형 데이터베이스 이론에 입각해서 릴레이션 스키마로 변환하는 과정
(개체 -> 개체(릴레이션,테이블), 속성 -> 속성(칼럼), 식별자-> 기본키, 관계-> 외래키)

2. 관계유형> 식별(Identifying), 비식별(Non-Identifying)

1)식별관계 : 상위(부모) 개체의 **기본키**가 하위 개체에 기본키로 전이 (상위 개체 PK = 하위 개체 PK, FK)

2)비식별관계 : 상위 개체의 기본키가 하위 개체에 **일반 속성**으로 전이 (상위 개체 PK = 하위 개체 FK)

Part5. 정규화(normalization)와 이상(anomaly)

1. 정규화 (normalization)

-정규화를 하는 이유는 데이터의 **중복을 방지**하고 보다 효율적으로 데이터를 저장하기 위함.

-릴레이션 분리, 삽입, 삭제, 갱신 **이상**의 발생 가능성을 줄이는 것

1) 제1정규화(1NF)

: 반복 되는 속성을 제거한 뒤 **모든 속성이 원자 도메인**만으로 되어 있는 정규형

*제1정규형에 위배되는 테이블

(중복 발생→공간 낭비, 이상 발생→무결성 위배 가능)

2) 제2정규화(2NF)

: 제1정규형이고, **부분함수적 종속**을 제거하여 **완전(충분한) 함수적 종속**을 만족하는 정규형.

3) 제3정규화(3NF)

: 제2정규형이고, **이행적 함수적 종속 관계** 제거하여 **비이행적 함수적 종속 관계**를 만족하는 정규형

4) BCNF (Boyce/Codd Normal Form)

: 제3정규형이고, **결정자가 후보키가 아닌 함수 종속 제거** 모든 **결정자가 후보키**이어야 한다는 것

5)제4정규화(4NF)

: **다치 종속** 제거

-**함수적 종속** : 아이디 -> 주민번호

(1:1로 대응해야 하므로 하나의 아이디 값이 2개 이상의 주민번호 속성을 결정하면 안됨)

-**다치 종속** : 아이디 ->> 수강과목

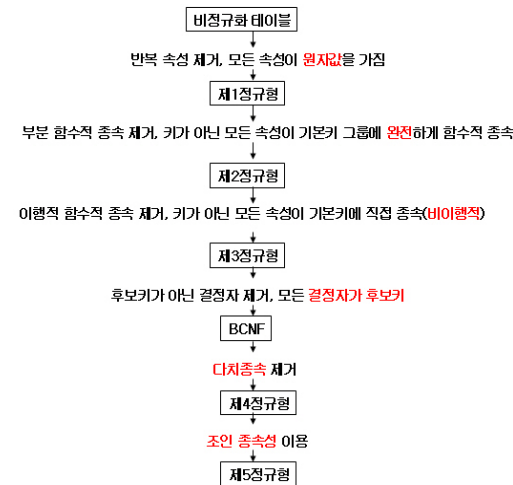
(1:다 대응으로 생각. 릴레이션에서 아이디 속성이 있고, 수강과목 속성이 있다면 하나의 아이디는 여러개 과목을 수강할 수 있으므로 아이디와 수강과목 속성은 함수적 종속에서 다치 종속, 이런 다치 종속은 이상이 발생할 수

있으므로 무순실 분해(정규화)되어야 합니다. 이것을 4정규형이라고 합니다.)

6) 제5정규형 (5NF).

: 후보키를 통하지 않은 **조인종속(JD) 제거**

- 조인 종속 : 릴레이션이 그의 어떤 프로젝트들을 조인한 결과와 똑같아야 한다



2. 이상 (anomaly)

1) **갱신이상** : 반복된 데이터 중에 일부만 수정하면 데이터의 불일치가 발생

2) **삽입이상** : 불필요한 정보를 함께 저장하지 않고는 어떤 정보를 저장하는 것이 불가능

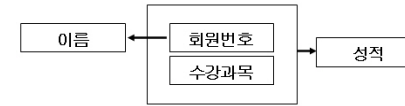
3) **삭제이상** : 유용한 정보를 함께 삭제하지 않고는 어떤 정보를 삭제하는 것이 불가능

3. 함수적 종속

:어떤 릴레이션 R에서 X와 Y를 각각 R의 애트리뷰트 집합의 부분 집합이라고 할 경우, 애트리뷰트 X의 값 각각에 대하여 시간에 관계없이 항상 애트리뷰트 Y의 값이 오직 하나만 연관되어 있을때 Y는 X에 함수 종속이라 하고, **X를 결정자, Y를 종속자** 라고 한다.

- 표기법 : $X \rightarrow Y$

- 예) 회원번호 \rightarrow 이름

1) 완전 함수적 종속: 회원번호,수강과목 \rightarrow 성적2) 부분 함수적 종속: 회원번호 \rightarrow 이름

※정규화 특징

-예를 들어 현재 테이블이 3정규형 상태라면 1,2 정규형은 자동으로 만족한다.

-정규형들은 차수가 높아질수록(제1정규형→제5정규형) 만족시켜야 할 제약조건이 증가된다.

-정규화는 논리적 처리 및 품질에 큰 영향을 미친다.

-정규화의 목적은 논리적 데이터베이스 구조상에 있어 삽입, 수정, 그리고 삭제 결과 생기는 이상현상(anomaly)을 제거하는데 있다.

-레코드들의 관련 속성들 간의 종속성을 최소화하기 위한 구성 기법이다

-정규화가 잘못되면 데이터의 불필요한 중복을 야기하여 릴레이션 조작시 문제를 일으킨다.

-정규화되지 못한 릴레이션의 조작시 발생하는 이상(anomaly) 현상의 근본적인 원인은 여러 가지 종류의 사실들이 하나의 릴레이션에 표현되기 때문이다.

Part6. 관계데이터 연산

1. 관계대수

-**절차적언어(절차중심)**: 원하는 정보를 '어떻게' 유도하는 가를 연산자와 연산규칙을 이용하여 기술

-분류

1) 순수관계 연산자

① SELECT (σ)

- 릴레이션에서 주어진 조건을 만족하는 튜플들을 검색하는 것으로 기호는 그리스 문자의 **시그마(σ)**를 이용.

(**행, 수평적 연산**)

-형식 : σ 조건 (R)

② PROJECT (π)

- 릴레이션에서 주어진 조건을 만족하는 속성들을 검색하는 것으로, 기호는 그리스 문자의 **파이(π)**를 이용.

(**열, 수직적 연산**)

- 형식 : π 속성 (R)

③ JOIN

-두개의 릴레이션A와 B에서 **공통된 속성을 연결**하는 것

-**A*B**(NATURAL JOIN, 자연조인): 공통 속성값 **제거**

-**A \bowtie B**(EQUI JOIN,동등조인): 공통 속성값 **중복**

④ DIVISION (\div)

- 나누어지는 릴레이션인 A는 릴레이션 B의 모든 내용을 포함한 것이 결과 릴레이션이 된다

2) 일반집합 연산자

① **합집합(U)**: 릴레이션 A 또는 B에 속하는 튜플들로 구성된 릴레이션 (**UNION**)

② **교집합(\cap)**: 릴레이션 A 와 B에 공통적으로 속하는 튜플들로 구성된 릴레이션 (**INTERSECTION**)

③ **차집합(-)**: 릴레이션 A에만 있고 B에는 없는 튜플들로 구성된 릴레이션 (**DIFFERENCE**)

④ **카티션 프로덕트(cartesian product) (X)**: A에 속한 각 튜플 a에 대하여 B에 속한 튜플 b를 모두 접속시킨 튜플들(a b)로 구성된 릴레이션

※관계 대수 연산자 중 **합집합, 교집합, 차집합** 연산은 이항 연산으로서 연산에 참가하는 두개의 릴레이션은 차수와 도메인이 같아야 연산을 수행가능.

2. 관계해석

-**비절차적언어(결과중심)**: 원하는 정보가 '무엇'이라는 것만 정의

-분류: 튜플관계해석, 도메인관계해석

-기본적으로 관계해석과 관계대수는 데이터베이스를 처리하는 기능과 능력면에서 동등함.

Part7. SQL

1. SQL(Structured Query Language) 특징

- 1) 관계대수와 관계해석을 기초로 한 고급 데이터 언어
- 2) 이해하기 쉬운 형태
- 3) 대화식 질의어로 사용 가능
- 4) 데이터 정의, 조작, 제어 기능 제공
- 5) COBOL, C, PASCAL 등의 언어에 삽입 -> 내장 SQL
- 6) DBMS에서 사용되는 비 절차적 대화형 Language

2. 시스템 카탈로그(=데이터 사전)

1) 시스템 자신이 필요로 하는 여러 가지 객체에 관한 정보를 포함하고 있는 **시스템 데이터베이스**

2) 특징

- 사용자도 SQL을 이용하여 검색할 수 있다.
- (DBMS만 스스로 갱신 유지, 사용자 갱신 안됨)
- 객체들로서는 기본 테이블, 뷰, 인덱스, 데이터베이스, 접근 권한 등이 있다.

3. SQL 구분

1)정의어:DDL ([CREATE](#), [ALTER](#), [DROP](#))

정의 변경 제거

2)조작어:DML ([SELECT](#), [INSERT](#), [DELETE](#), [UPDATE](#))

검색 삽입 삭제 갱신

3)제어어:DCL ([GRANT](#), [REVOKE](#), [COMMIT](#), [ROLLBACK](#))

권한부여 권한취소 transaction 제어

4. DDL (데이터정의어)

: 도메인, 테이블, 뷰, 인덱스를 정의/변경/제거하는 언어

1) CREATE DOMAIN

```
CREATE DOMAIN SUNG CHAR(2)
DEFAULT '여'
CONSTRAINT SUNG_ck CHECK
(SUNG="남"or SUNG="여");
```

- *CHAR : 문자형, DATE : 날짜형, INTEGER : 정수형
- *반복되는 데이터 타입을 **사용자 정의 데이터 타입**으로 정해서 효율적으로 사용할 수 있다.
- *도메인 무결성과 관련이 있으므로 개념 이해 중요함.
- *DEFAULT : 기본값의 의미로써 입력하지 않았을 경우 자동으로 입력되는 값.

2) CREATE TABLE

```
CREATE TABLE 학생 (
    학번 CHAR(18),
    이름 CHAR(18) NOT NULL,
    성별 SUNG,
    생년월일 DATE,
    PRIMARY KEY(학번),
    UNIQUE(이름),
    FOREIGNKEY(학과코드)REFERENCE학과(학과코드),
    CHECK 생년월일 >= '1974-05-01'
);
```

- *기본키는 NOT NULL, UNIQUE 으로 자동 설정된다.
- *UNIQUE 무결성 : 해당 속성은 유일해야 한다.
(이름은 중복되어서는 안 된다.)
- * NULL 무결성 : 해당 속성은 Null 값을 가질 수 없다.

(이름은 Null 값을 가질 수 없다.)

* FOREIGN KEY ~ REFERENCES 옵션

- ON DELETE[UPDATE] CASCADE
- : 부모 테이블이 삭제/수정되면 관련된 자식 테이블들도 연쇄 삭제/수정된다.
- ON DELETE[UPDATE] SET NULL
- : NULL 이 된다.
- ON DELETE[UPDATE] SET DEFAULT
- : DEFAULT가 된다.
- 무 옵션 : 실행 안됨.

3) CREATE VIEW

```
CREATE VIEW 여학생_view(이름, 성별) AS
SELECT 이름,성별
FROM 학생
WHERE 성별='여';
WITH CHECK OPTION;
```

*뷰에 대한 수정, 삽입 연산이 실행될 경우 WHERE 절 조건에 위배될 경우는 실행을 거부

4) CREATE INDEX

```
CREATE UNIQUE INDEX 이름_idx
ON 학생(이름 ASC)
CLUSTER;
```

- * 자동으로 기본키가 인덱스로 정의된다.
- * UNIQUE : 인덱스로 정의되는 속성 값이 중복되어서는 안 된다. (생략시 중복 허용)
- * 정렬 : ASC (오름차순, 생략시), DESC (내림차순)
- * CLUSTER : 클러스터드 인덱스를 만들게 되면 기본적으로 그 행(Index Key)을 기준으로 물리적으로 데이터를 정렬 (기본 오름차순, 내림차순)시킨다. 기본적으로 **년 클러스터드 인덱스**보다 검색 속도가 빠르다. (물리적으로 인접하므로)
- 한 테이블에 하나의 클러스터드 인덱스만 만들 수 있다.
- * INDEX 를 정의하면 테이블 검색 속도는 빨라지나 수정, 삽입시 수행 속도가 느려질 수 있다.
- 기억공간 낭비

5) ALTER TABLE

```
ALTER TABLE 학과 ADD 연락처 CHAR(18);
ALTER TABLE 학과 ALTER 학과명 SET DEFAULT '정보';
ALTER TABLE 학과 DROP 학과명;
```

- * 테이블의 구조를 변경 → 속성 변경

6) DROP TABLE

DROP TABLE 학과 CASCADE;

- * 참조 무결성 위배를 피하기 위한 2가지 옵션
 - CASCADE : 참조하는 테이블을 연쇄적으로 제거
 - RESTRICT : 참조하는 테이블이 있을 경우 제거 안 됨 (생략 가능)
- DELETE(삭제), UPDATE(갱신) 명령어도 옵션 사용

5. DML(데이터조작어)

1) INSERT(삽입)

- ① INSERT INTO 수강생 VALUES
(‘김길현’, ‘정보’, ‘남구’, 100);
→수강생 테이블에 김길현, 정보, 남구, 100 을 넣어라

- ② INSERT INTO 수강생(이름,수강료) VALUES
(‘이상인’, 120);
→ 수강생 테이블에 이름, 수강료 에 이상인, 120을 넣어라.

- ③ INSERT INTO 정보수강생(이름,과목,수강료)
SELECT 이름,과목,수강료 FROM 수강생 WHERE 주소=‘남구’;
→수강생 테이블에서 주소가 남구인 이름, 과목, 수강료를 SELECT 해서 정보수강생 테이블 속성인 이름, 과목, 수강료에 넣어라.

2) DELETE(삭제)

- * DELETE 는 튜플을 삭제하는 명령어 (테이블은 DROP)
- ① DELETE FROM 수강생 WHERE 과목=‘사무’;
→수강생 테이블에서 과목이 사무인 튜플을 삭제하라.

- ② DELETE FROM 수강생;
→수강생 테이블에 있는 모든 튜플을 삭제하라.

3) UPDATE(갱신)

- ① UPDATE 수강생 SET 과목=‘사무’ WHERE 이름=‘최영희’;
→수강생 테이블에서 이름의 최영희인 튜플의 과목을 사무로 바꾸어라.

- ② UPDATE 수강생 SET 수강료=수강료+10 WHERE 과목=‘워드’;
→수강생 테이블에서 과목이 워드인 튜플의 수강료를 +10 해라.

4) SELECT(검색)

(1) 기본구조

```
SELECT 속성 (or 수식, 그룹함수)
FROM 테이블
[WHERE 조건]; ([ ] 생략 가능)
```

- ① 모든 튜플을 검색하라.
SELECT * FROM 수강생;
SELECT 수강생.* FROM 수강생;
SELECT 이름,과목,주소,수강료 FROM 수강생;
SELECT 수강생.이름,수강생.과목,수강생.주소,수강생.수강료 FROM 수강생;

- ② SELECT DISTINCT 과목 FROM 수강생;
→수강생 테이블에서 중복 제거 된 과목만 검색하시오.

- ③ SELECT * FROM 수강생 WHERE 과목=‘정보’;
SELECT * FROM 수강생 WHERE 과목 IN (‘정보’);
** <>, NOT IN (같은 표현)
→수강생 테이블에서 과목이 정보인 모든 튜플을 검색하시오.

- ④ SELECT * FROM 수강생 WHERE 과목=‘컴활’ OR 과목=‘워드’;
→ 수강생 테이블에서 과목이 컴활 이거나 워드인 모든 튜플을 검색하시오.

- ⑤ SELECT * FROM 수강생 WHERE 이름 LIKE ‘김%’;
→ 수강생 테이블에서 이름이 ‘김’으로 시작하는 모든 튜플을 검색하시오.

- ⑥ SELECT * FROM 수강생 WHERE 수강료 BETWEEN 30 AND 90;
→ 수강생 테이블에서 수강료가 30이상 90이하인 모든 튜플을 검색하시오.

- ⑦ SELECT * FROM 수강생 WHERE 과목 IS NULL;
→ 수강생 테이블에서 과목이 NULL인 모든 튜플을 검색

하시오. (주의 : CREATE 문은 IS 를 생략)

⑧ SELECT **DISTINCT** name FROM Shop WHERE id IN (SELECT Shopid FROM Staff WHERE id = '10');
→직원 코드가 '10'인 직원이 담당하는 **상점이름을 중복**은 배제하고 검색하여 주시오.

(2) 확장구조

```
SELECT [DISTINCT] 속성
FROM 테이블
[WHERE 조건]
[GROUP BY 속성 [HAVING 조건]]
[ORDER BY 속성 [ASC | DESC]];
```

- * **DISTINCT** : 검색 결과에서 중복 배제
(주의 : CREATE 문의 UNIQUE 과 구분)
- * **GROUP BY** : 그룹별 검색
(예, 과목별 수강생 수를 구하라.)
- * **ORDER BY** : 정렬 검색 (오름차순 : A-Z, ↗-↘)

*그룹함수 종류

COUNT(속성) : 그룹별 튜플 수
AVG(속성) : 그룹별 평균
SUM(속성) : 그룹별 합계
MAX(속성) : 그룹별 최대값
MIN(속성) : 그룹별 최소값

-COUNT(*) : 튜플의 건수 → Null 포함
-COUNT(속성명) : 튜플의 건수 → Null 제외
-COUNT(DISTINCT 속성명) : Null, 중복 제외된 건수 반환
-UNION : 중복 없이 병합
-UNION ALL : 중복 허용 병합

* **SQL 수행 순서** : FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY

6. DCL(데이터 제어어)

: DB가 데이터 관리를 목적으로 사용하는 언어로서 보안,회복,사용자 권한 등을 정의 한다.

- GRANT: 권한부여
- REVOKE: 권한취소
- COMMIT

:트랜잭션의 성공했을 경우 그 결과를 DB에 적용하여 완

료 시킴.(트랜잭션 완료 → DB 적용),

-ROLLBACK

트랜잭션의 실패로 작업을 취소하고, 이전 상태로 되돌림. (트랜잭션 취소 → DB 적용 안됨)

Part8. 내장SQL과 물리적 데이터베이스 모델링

1. 내장 SQL

:: 호스트 언어(C, C++, 비주얼베이직 등)에 삽입된 SQL

- 1) 단일 튜플 검색
- 2) 복수 튜플 검색 (커서 이용)

→ 목적 : 일괄처리, 동일 업무 반복 시

2. 물리적 데이터베이스 모델링

: 논리 데이터베이스 구조로부터 효율적인 물리적 구조의 데이터베이스 구조를 설계하는 과정

* 효율적인 물리적 구조란?

:성능↑, 저장공간↓, 보안, 백업, 회복이 용이한 구조

* 물리적 데이터베이스 모델링 단계

- 1) 개발 DBMS 선정
- 2) 칼럼의 데이터 타입과 사이즈 정의
- 3) 데이터 사용량 분석
- 4) 역 정규화
- 5) 인덱스, View, Stored Procedure, Trigger 등 정의
- 6) 데이터베이스 생성

3. 인덱스 (Index)

: 데이터베이스에서 원하는 데이터를 좀더 빨리 찾아줄 수 있도록 데이터의 위치정보를 모아 둠

* 항상 정렬된 상태를 유지한다.

(Table 검색 속도 ↑ → 전체적인 시스템 성능 ↑)

* 책에서 색인을 이용해서 해당 내용을 찾는 것과 동일한 개념

* Data Page : 데이터가 저장되는 물리적인 공간 (논리적 공간:테이블)

4. 트리거(Trigger)

: 특정 테이블의 데이터에 변경이 가해졌을 때 연쇄적으로 수행되는 저장 프로시저(=함수)라고 할 수 있다.

- INSERT, UPDATE, DELETE문이 TABLE에 대해 행해질 때 묵시적으로 수행되는 PROCEDURE

5. 커서 (Cursor)

: 질의 결과인 복수 개의 튜플에서 다음에 처리될 튜플을

지시하는 지시자를 의미. (효율적인 처리 가능)

* 트리거를 이용해서 발생하는 오버헤드를 해결해서 안정적으로 시스템 운영이 가능하다.

6. 뷰 (View)

: 가상 테이블로서 물리적으로 존재하지 않으면서 마치 데이터를 가지고 있는 테이블로 보여짐

* 목적 : 보안상 이용, 조회용으로 이용

7. 스토어드 프로시저 (Stored Procedure)

: 연속된 SQL 문들을 하나로 모아 SQL 서버에 미리 컴파일해서 저장해 놓은 개체로서 프로그램에서 함수와 같은 역할을 한다. (재사용성 ↑)

8. 역정규화 (DeNormalization)

: 시스템의 성능 향상을 위해서 정규화에 위배되는 행위를 하는 것.

→ 정규화 : 테이블 분리, 중복 제거

→ 역정규화 : 테이블 통합 및 분리, 중복 허용

*역정규화 유형

① 테이블 역정규화 → 테이블 병합

-두 개의 테이블이 동일한 프로세스에 의해 처리되며, 조인이 빈번히 발생할 때 수행

-역정규화된 테이블은 정규화에 위배될 수 있으며 데이터의 중복이 발생할 수 있다.

-효과 : 조인, 참조 무결성 등의 문제 해결 →성능 향상

② 테이블 역정규화 → 테이블 분리 (칼럼 기준)

- 칼럼의 개수가 많을 경우 Size가 커서 조회를 할 경우에 부하가 발생하게 된다.

- 업무적 활용도에 따라 분리

③ 테이블 역정규화 → 테이블 분리 (튜플 기준)

- 업무적 활용도에 따라 자주 사용되는 튜플과 나머지 튜플로 분리

④칼럼 역정규화 → 칼럼 중복

- 조인 프로세스를 줄이기 위해 해당 칼럼을 중복함으로 해서 조회를 할 경우 조인을 수행하지 않도록 하기 위한 방법 (제2정규형 위배 : 부분함수적 종속 발생)

⑤ 칼럼 역정규화 → 파생 칼럼

- 계산을 통해서 얻어질 수 있는 결과 값을 테이블의 새로운 칼럼으로 만들어서 값을 저장

- 파생 칼럼 시킬 경우 데이터는 사용자가 직접 입력하는 것이 아니고 데이터가 입력, 수정, 삭제되어질 때 트리거를 이용하여 자동으로 관리 해야 한다.

9. DB튜닝

데이터베이스가 일정한 성능을 유지할 수 있도록 비효율적인 요소를 제거하고 성능 개선을 위하여 SQL 문장을 포함, 데이터베이스의 여러 요소를 **조정(조율,최적화)**하는 작업을 말한다. 데이터베이스 내에 데이터의 양이 증가하고 사용자의 수가 증가하면 자연히 데이터베이스의 응답 속도 및 처리 속도가 저하되기 마련이다. 따라서 데이터베이스 튜닝을 통하여 일정한 성능을 유지시키는 것이 중요하다.

10. 유지보수

시스템이 한번 개발되면 영원히 사용할 수 있는 것이 아니다. 시스템의 개발이 완료되는 시점부터 시스템의 변경이 시작된다는 말이 있다. 시스템에 변경이 일어나는 이유는 시스템의 이용에 따라 **사용자의 요구도 변화하고 비즈니스 환경이 변화됨에 따라 업무 절차도 지속적으로 변화하기** 때문이다. 이에 따라 프로그램도 변경이 일어나지만 데이터베이스에도 지속적인 변경이 일어난다. 새로운 테이블이 생성되기도 하고 기존의 테이블에 새로운 컬럼이 추가되기도 한다. 이러한 일련의 작업을 유지보수라고 한다.

11. 로킹 (Locking)

1) **병행제어** : 동시에 여러 개 수행할 때, 데이터베이스 일관성 유지를 위해 트랜잭션 간의 상호 작용을 제어

2) 병행제어 목적

- ① 데이터베이스 공유 최대화
- ② 시스템 활용도 최대화
- ③데이터베이스 일관성 유지
- ④사용자에 대한 응답시간 최소화

3) 병행제어 기법

①로킹 (Locking)

- 하나의 트랜잭션이 데이터를 액세스하는 동안 다른 트랜잭션이 그 데이터 항목을 액세스할 수 없도록 하는 방법

②로킹 단위

- 병행제어에서 한꺼번에 로킹 할 수 있는 단위
- 로킹 단위가 크면 → 로크수가 작다. → 관리가 수월해

지고 병행성 수준↓

- 로킹 단위가 작으면 → 로크수가 커진다. → 관리가 복잡해지고 병행성 수준↑

Part9. 데이터베이스 주요 기술 용어

1. 크기에 따른 데이터베이스 분류

: 사용자의 수(number of users)에 따라 분류

1) 개인용 DB

-개인용 컴퓨터에 탑재되어 한명의 사용자가 이용

-공유(share)하여 사용하지 않음

-모든 책임을 사용자 자신에게 있음

2) 워크그룹(Workgroup) DB

-동일한 프로젝트를 수행하는 집단으로 25명 내외 구성

-워크그룹에 속한 사용자들은 LAN(Local Area Network)을 통하여 데이터베이스를 이용

3) 부서(Department) DB

-한 기관 내에서 동일한 업무를 수행하는 조직

-워크그룹보다는 큰 조직 규모

-구성 형태는 워크그룹 데이터베이스와 동일

4) 전사적(Enterprise) DB

-특정 기관에 속한 부서와 워크그룹이 포함하는 대용량 DB

-구축 시 가장 많이 이용하는 방법론

: 데이터 웨어하우스(DW : Data Warehouse)

2. 데이터 웨어하우스(DW : Data Warehouse)

-사용자의 효율적인 의사 결정에 도움을 주기 위하여, 시스템에서 추출, 변환, 통합되고 요약된 주제 중심적인 DB

-주제 중심적(subject-oriented) 구성

-통합된(integrated) 내용

-시간에 따라 변화되는(time-variant) 값의 유지

: 운용 데이터베이스는 필요할 때마다 갱신되지만 데이터 웨어하우스는 일단 일련의 스냅샷(snapshot)으로 올바르게 기록되면 갱신되지 않는다.

-비광선성(non-volatile): 갱신 이상(update anomaly)을 염려할 필요 X

* 출현배경

① 현 전산시스템의 공로

OLTP(On-Line Transaction Processing)이 DW의 기반구조: 주로 신용카드 조회 업무나 자동 현금 지급 등 금융 전산 관련 부문에서 이용

② 현 전산시스템의 한계

OLAP(On-Line Analytical Processing, 이용자 직접 데이터베이스를 검색, 분석해서 문제점이나 해결책을 찾는 분석형 애플리케이션 개념)에 따른 효율적인 의사결정지원 필요-DW 개념 필요

③ 의사 결정 지원 시스템(DDS : Decision Support System)과 최고경영자 정보 시스템(EIS : Executive Information System)의 기대감 증가

* 데이터웨어 하우스 특성 분석

기존의 OLTP 업무 중심의 전산 시스템으로부터 생성되는 데이터를 데이터 웨어하우스 용도에 맞게 변환, 추출하여 저장소 데이터베이스로 적재한 후 관리하는 것으로, 주로 경영자나 분석가들이 전략 정보 획득을 목적으로 이용

* 기존OLTP 전산 시스템과 차이점

①사용목적의 차이점

-OLTP:거래(업무)처리 용도

-DW: 의사결정지원 용도

②사용대상의 차이점

-OLTP:담당(창구)직원이 주 사용

-DW: 의사결정을 시행하는 경영자, 분석가들이 주 사용

③처리 데이터 특성상의 차이점

-OLTP

:응용 중심(application-oriented)의 데이터

현재 시점을 중시하며 데이터가 동적(dynamic)으로 변경되는 특성

-DW

:주제 중심(subject-oriented)인 데이터

과거 지향적이며 데이터의 변경이 발생하지 않는 정적(static)인 특성

3. 데이터 마이닝(DM : Data Mining)

엄청난 데이터 양을 유용한 정보(가치있는 정보)로 가공

하고자하는 연구 중 한 분야.

* IT발전 단계에 따른 출현배경

1960년대: 데이터 수집(collection)이 주요기술

1980년대: 데이터 접근(access)하는것이 주요기술

1990년대: 데이터에 대한 질의(data queries)가 주요기술

→“데이터 과잉 문제(data glut problem)”가 발생

그 결과 현재 데이터간에 의미 있는 정보를 추출하는 데이터 마이닝 기술이 주요 기술로 부상

(의사결정자이 효과적으로 지원할 수 있는 귀중한 정보를 찾아내는 방법에 대한 기술)

* 데이터마이닝 개발절차

①적용업무 정의

②원천 데이터를 선택

③데이터정제(cleaning), 코딩(coding), 보강(enrichment)

④마이닝 도구 선정

⑤데이터 마이닝 작업을 구현

⑥결과분석

4. 3단계 데이터베이스 구조

-데이터베이스를 관점(view)에 따라 3계층으로 분리

1) 외부단계(external level)

=외부 스키마(external schema)=뷰 단계(view level)

-각각의 데이터베이스 사용자 관점 또는 사용자 뷰(user view)를 표현하는 단계

-동일한 하나의 데이터베이스라도 그것을 이용하는 사용자에 따라 관심 분야가 다르기 때문에 필요한 내용도 차이가 남

2) 개념단계(conceptual level)

=개념스키마(conceptualschema)=논리단계(logical level)

-데이터베이스에 저장되는 데이터와 그것들간의 관계(relationship)를 표현하는 단계

-이용하는 모든 사용자들의 총체적관점 (community user view)

3) 내부단계(internal level)

=내부스키마(internalschema)=물리스키마(physical schema)=물리 단계(physical level)

-물리적인 저장장치(일반적으로 하드디스크)에서 데이터가 실제로 저장되는 방법을 표현하는 단계

-저장장치의 관점(storage view)을 제공

5. 스키마 vs 인스턴스

1) 스키마

-데이터베이스의 전체적인 정의

-데이터베이스 스키마=스키마=논리 스키마

2) 인스턴스:DB에 저장되는 값들

6. 데이터 독립성

1) 물리적데이터 독립성(Physical data independency)

:데이터베이스 저장장치의 구조가 변경되어도 응용 프로그램이나 개념 스키마에는 영향을 미치지 않는 것

2) 논리적데이터 독립성(Logical data independency)

:데이터베이스의 논리적 구조가 변경되어도 응용 프로그램에는 영향을 미치지 않는 것

7. 속성(Attribute)

-파일 시스템의 필드(field)

1) 단순속성(Simple attribute)

: 일반적인 속성 의미

2) 복합속성(Composite attribute)

: 속성 값이 여러의미를 포함하는 것

-복합 속성을 구성하는 년, 월, 일 속성을 **단위속성(component attribute)**라고 함

-복합속성의 필요성

① SQL 질의어의 “문자열 연산식을 이용한 검색(like)기능을 이용가능

② 어떤 속성을 ‘단순 속성’ 또는 ‘복합 속성’으로 지정하기 이전에 데이터베이스 응용 서비스에 대한 충분한 분석 작업이 선행되어야 하는 것

3) 단일 값 속성(Single-valued attribute)

:속성 값이 원자값

4) 다중 값 속성(Multi-valued attribute)

:속성 값이 여러 개 존재할 수 있는 것

-특성인 “모든 속성 값은 원자 값(atomicvalue)이다”라는 규칙을 위반하는 것(관계데이터 모델에서는 지원안함)

5) 유도 속성(Derived attribute)

:기존 릴레이션의 속성 값을 이용하여 새롭게 유도해 낸

속성

-생일 속성이용→나이=현재날짜-생일

-유도 속성을 계산하는 데 이용된 생일(S_BIRTH) 속성을 기본속성(baseattribute) 또는 저장 속성(stored attribute)

8. 데이터 정화(Data Cleansing)

:데이터베이스의 불완전 데이터에 대한 검출·이동·정정 등의 작업

9. 데이터 적재(Data Import)

: 대량의 데이터 파일을 데이터베이스 데이터로 읽어오는 DBMS 기능

10. 데이터 반출(Data Export)

: 데이터베이스 데이터를 외부 파일로 기록하는 DBMS 기능

11. 데이터 모델 구성요소

1) 구조 (Structure) : 개체들 간의 관계

2) 연산 (Operation) : 데이터 처리하는 방법

3) 제약조건 (Constraint) : 실제 데이터의 논리적인 제약 조건

12. 데이터베이스 관리 시스템(DBMS)

1) 질의 처리기 (Query processor)

: 데이터베이스 사용자의 요구를 받아 해석하는 역할을 수행

(1) 비절차적 DML 컴파일러

or DML 컴파일러(DML compiler)

- 최종 사용자들의 질의어로서 최종 사용자가 질의를 하면 DML 컴파일러는 DBMS가 이해할 수 있도록 번역한다.

(2) 절차적 DML 예비컴파일러

or 내장DML 예비컴파일러

(Embedded DML precompiler)

- 응용 프로그램에서 사용한 DML 문장들을 프로그래밍 언어의 프로시저로 변환한다. 예를 들어, 절차적 DML을 이용하여 작성한 응용 프로그램에서 사용한 SQL 문장은 호스트 언어가 이해할 수 있는 프로시저로 번역한다.

(3) DDL 인터프리터 (DDL interpreter)

- DBA 또는 데이터베이스 설계자가 작성한 DDL을 해석

하여 실행한다.

(4) 질의 실행기 (Query processing engine)

- DML 컴파일러와 응용 프로그램에서 요청하는 질의를 실행하는 역할을 담당한다.

2. 저장 관리자 (Storage manager)

- 디스크에 저장되어 있는 데이터를 접근하고 관리하는 역할을 수행한다. 이 때 실질적으로 디스크에 저장되어 있는 데이터를 접근하는 기능을 수행하는 것은 운영체제(OS)이므로 저장 관리자는 운영 체제와 협력하여 데이터를 접근하고 관리한다.

데이터베이스 시스템이 관리하는 데이터베이스는 데이터 파일, 데이터 사전, 인덱스 등으로 구성된다.

13. 테이블 기술서

: 여러 가지 이유로 테이블 하나하나에 대한 출력된 문서를 필요로 한다. 이와 같은 이유로 만들어진 문서

14. varchar : 가변길이 문자열

-50byte까지 넣을 수 있는 가변길이 문자열을 의미함.
(영문자 기준, 한글은 2바이트를 차지하므로 한글로만 넣을 경우 25자까지 넣을 수 있음)

15. char : 고정길이 문자열

-50byte까지 넣을 수 있는 고정길이 문자열을 의미함.

16. ROW MIGRATION

-한 row 에 update 가 발생할 경우 변경되어 저장될 데이터의 사이즈는 큰데 현재 그 row 가 저장되어 있는 block 의 공간이 변경 저장될 데이터의 사이즈만큼 남아 있지 않다면 해당 block 엔 저장할 수 없으므로 해당 row 를 온전히 저장할 수 있도록 다른 block 에 해당 row 전체를 옮겨(이주) 저장하는데
-주로 VARCHAR 타입을 가진 컬럼에서 발생

17. ROW CHAINING

- row chaining 은 한 컬럼의 데이터 사이즈가 너무 큰 경우에 주로 발생하는데 예를 들어, block size가 2KB 인데 특정 컬럼의 데이터 사이즈가 CHAR(3000) 이라고 가정하자. 결국, 한 row 가 한 block 에 모두 저장될 수 없는 상황이다. 따라서, 해당 row 는 여러 block 에 걸쳐 저장되는데 이를 row chaining 이라고 하는 것.

Part10. JOIN, SQL[심화]

1. 조인(JOIN)

-하나의 SQL 명령문으로 여러 테이블에 저장된 데이터를 한 번에 검색할 수 있는 강력한 기능
-두 개 테이블을 연결하는 조인 속성

-잘못 사용하면 오히려 질의 처리 성능을 저하시키는 경우도 발생

-서로 다른 테이블에 있는 동일한 열 이름을 사용할 경우, 열 이름이 중복되어 구별이 어려운 문제가 발생

-조인의 종류

1) CROSS JOIN : 카르테시안 조인

→ 카티션 프로덕트(cartesian product)

-조인된 테이블 사이에 조건이 걸리지 않을 경우에 발생하며, 테이블간의 모든 경우의 수에 대해서 로우가 생성되는 조인 방식

-CROSS JOIN은 실제 많이 사용하지는 않지만, SQL 성능 튜닝에 사용

2) INNER JOIN : 내부 조인 → JOIN

-일반적으로 부르는 조인은 내부 조인을 의미

-종류

① 세타조인(THETA JOIN) ⊕

② 이퀄조인(equal) =

③ NATURAL JOIN

④ Non-EQUI JOIN

⑤ 셀프 조인(SELF JOIN)

3) OUTER JOIN : 외부 조인

-외부 조인은 조인되는 A 테이블에서 B 테이블에 연결되는 컬럼값이 존재하지 않더라도(즉, NULL 값) A 테이블의 데이터를 가져올 수 있는 조인 방법

-종류

① LEFT OUTER JOIN

② RIGHT OUTER JOIN

③ FULL OUTER JOIN

4) SEMI JOIN : 세미 조인 ×

-조인 대상 릴레이션 중 하나를 프로젝트(PROJECT) 연산을 수행한 후 조인을 하는 것

2. IN 키워드 사용법

1) 일반 질의문

-일반적으로 IN 은 다중 값의 의미

-단일 값일 경우에도 IN 을 사용할 수 있으나

일반적으로 = 를 사용

[예시]

* 수강생 테이블에서 과목이 컴활 이거나 워드인 모든 튜플을 검색시오.

→SELECT * FROM 수강생 WHERE 과목 = '컴활' OR 과목 = '워드';

→SELECT * FROM 수강생 WHERE 과목 IN ('컴활', '워드');

2)부속(subquery) 질의문

-부속 질의에는 select 절과 from 절이 꼭 포함

-상위 질의보다 먼저 실행되어야 하기 때문에 괄호()로 묶어야함

-부속질의 결과가 다중 값이므로 IN 을 사용

-만일 부속질의 결과가 단일 값일 경우는 = 를 사용가능

[예시]

*수강과목 테이블에서 과목이 정보인 주소를 검색시오.

→SELECT 주소 FROM 수강생 WHERE 이름 IN (SELECT 수강생이름 FROM 수강과목 WHERE 과목 = '정보');

→SELECT 주소 FROM 수강생, 수강과목 WHERE 과목 = '정보 AND 이름 = 수강생이름;

-부속질의 결과가 단일 값이므로 = 를 사용.

물론 IN 도 사용 가능하다.

(IN 은 단일, 다중 값 모두 사용)

[예시]

*수강과목 테이블에서 과목이 컴활인 주소를 검색시오.

→ SELECT 주소 FROM 수강생 WHERE 이름 = (SELECT 수강생이름 FROM 수강과목 WHERE 과목 = '컴활');

3. EXISTS 키워드 사용법

-부속질의 결과가 TRUE(검색 결과 존재) 일 경우만 검색됨

[예시]

수강과목 테이블에서 과목이 정보인 수강생이름이 있을 경우 수강생 테이블을 검색시오.

→ SELECT * FROM 수강생 WHERE EXISTS (SELECT 수강생이름 FROM 수강과목 WHERE 과목 = '정보');

-부속질의 결과가 FALSE(검색 결과 존재 하지 않음) 이므로 빈 테이블 검색됨

[예시]

수강과목 테이블에서 과목이 사무인 수강생이름이 있을 경우 수강생 테이블을 검색하시오.

→ SELECT * FROM 수강생 WHERE EXISTS (SELECT 수강생이름 FROM 수강과목 WHERE 과목 = '사무');

4. ALL 키워드 사용법

- 검색된 모든 결과를 만족할 경우 참(TRUE)

- 형식 :⊙ ALL (부속질의)

- ⊙는 {<, <=, =, <>, >, >=} 중 하나 사용

[예시]

SELECT * FROM 수강과목 WHERE 수강료 > ALL (100,200,300);

→ 100,200,300 조건값보다 모두(all) 커야 검색됨.

→ SELECT * FROM 수강과목 WHERE 수강료 > 300; 와 동일함

5. ANY 키워드 사용법

- 검색된 모든 결과 중 최소한 하나를 만족할 경우 참(TRUE)

- 형식 :⊙ ANY (부속질의)

- ⊙는 {<, <=, =, <>, >, >=} 중 하나 사용

[예시]

SELECT * FROM 수강과목 WHERE 수강료 > ANY (100,200,300);

→ 100,200,300 조건값 중 아무거나(any) 하나만 있어도 검색됨.

6. 집합 연산을 이용한 검색

1) INTERSECTION 명령을 이용한 검색

: 두 개의 릴레이션에 모두 있는 행들을 포함하는 테이블 생성

2) DIFFERENCE 명령을 이용한 검색

: 첫 번째 릴레이션에는 있지만 두 번째 집합에는 없는 행들을 포함하는 테이블 생성

7. UNIQUE 제약조건

- UNIQUE 제약조건은 기본 키 제약조건에 NULL 값 허용이 추가된 제약조건

-NULL 값도 유일해야 하기 때문에 한번만 입력가능

-만약 중복된 NULL 값이 입력 될 때는 에러가 발생

-기본 키 제약조건은 NOT NULL 과 UNIQUE를 함께 선언한 것이라 할 수 있음

8. IDENTITY 속성

-테이블을 생성할 때 행 정의에 IDENTITY 속성을 추가하여 열을 IDENTITY 열로 지정가능

-열이 IDENTITY 속성으로 생성된다면 초기값과 증가값을 기초로 하여 자동적으로 그 열의 행 값을 생성

9. CHECK

-제약조건은 열에서 허용 가능한 데이터의 범위나 조건을 지정하는 제약조건

-실수 등으로 부정확한 값이 입력되는 것을 예방

-한 열에는 CHECK 제약조건을 원하는 만큼 지정할 수 있으며 조건에 AND 및 OR로 결합된 여러 논리식을 포함 가능

10. LIKE 연산자

-사용자가 데이터 값을 정확하게 모르는 경우에 유용하게 사용할 수 있는 연산자

[예시]

LIKE '정%' : '정' 으로 시작하는 값

결과> 정보처리기사, 정보처리기능사

LIKE '%동%': 문자열에 '동' 이 들어 가는 값

결과> 홍길동, 동대문.

LIKE '김_ _': '김'으로 시작하고 뒤에 두글자인 값

결과> 김길현, 김수로

LIKE '_상%': 두번째 위치에 '상' 이 들어 가는 값

결과> 상상플러스

11. IS NULL 연산자

*학생 테이블에서 별명이 없는 학생의 이름, 별명, 학년, 주소를 검색하라.

→ SELECT 이름, 별명, 학년, 주소 FROM 학생 WHERE 별명 IS NULL;

*학생 테이블에서 별명이 'NULL' 인 학생의 이름, 별명, 학년, 주소를 검색하라.

→ SELECT 이름, 별명, 학년, 주소 FROM 학생 WHERE 별명 = 'NULL';

12. 집단합수, 그룹화

*수강생 테이블에서 권씨 성을 가진 학생 중에서 '전화번호' 정보가 있는 학생의 수'를 검색하라.

→ SELECT COUNT(전화번호) AS '전화번호 정보 있는

학생의 수' FROM 수강생 WHERE 이름 LIKE '권%';

* COUNT(전화번호)는 NULL인 행을 제외하고 계산하므로 다음과 같이 WHERE 절에서 전화번호 IS NOT NULL 을 추가한 COUNT(*)와 동일하다.

→ SELECT COUNT(*) AS '전화번호 정보 있는 학생의 수' FROM 수강생 WHERE 이름 LIKE '권%' AND 전화번호 IS NOT NULL;

(2) 수강생 테이블에서 과목번호 1에 대한 성적의 평균은?

→ SELECT AVG(성적) AS 성적의평균 FROM 수강생 WHERE 과목번호 = 1;

(3) 수강생 테이블에서 과목번호별 성적의 평균은 얼마인가?

→ SELECT 과목번호, AVG(성적) AS 성적의평균 FROM 수강생 GROUP BY 과목번호;

* GROUP BY 절에 명시하지 않은 열을 SELECT 절에서 사용하면 에러가 발생한다.→ SELECT 과목번호, 이름, AVG(성적) AS 성적의평균 FROM 수강생 GROUP BY 과목번호;

라는 SQL 명령문은 잘못된 것이다. 그 이유는 GROUP BY 절에 명시하지 않은 이름을 SELECT 절에서 사용했기 때문이다. 그러나 GROUP BY 절에 기술한 열 이름은 SELECT 절에서 명시하지 않아도 된다.

Part11. 장애, 회복, 인덱스

1. 장애

-시스템의 내적 또는 외적인 문제점으로 인하여 시스템이 정상적으로 동작할 수 없는 상태

1) 트랜잭션 장애

-잘못된 입력, 데이터를 찾을 수 없는 경우, 오버플로우(overflow), 가용 자원(resource)의 초과 요청

-논리적 오류 (logical error)

2) 시스템 장애

-교착상태(deadlock)와 같은 예기치 못한 상황이 발생

-하드웨어적인 오동작으로 인하여 휘발성(volatile) 기억장치인 주기억장치에 저장된 데이터가 손실

3) 미디어 장애

-비휘발성(nonvolatile) 기억장치인 하드디스크의 고장(예, 디스크 헤드 고장이나 하드디스크로 데이터를 전송 시 오류 발생 등)으로 디스크 블록(disk block)이 붕괴되는 것

2. 회복

데이터베이스 작업 도중 발생하는 정보를 한 곳에만 저장하는 것이 아니라, 별도로 저장 관리하여, 장애 발생시에 활용

1) 중복저장기법

① 덤프(Dump): 데이터베이스 내용 전체를 일정 기간(예, 시간별, 일별, 주별 등)마다 다른 저장 장치(예, 하드디스크 또는 자기테이프)에 저장하는 것

② 로그(Log)

-변경될 때마다 변경 내용을 로그 파일에 저장(장애 발생 시에 변경 전 내용으로 복구가 가능하도록)

-실행 중인 트랜잭션에 대한 정보를 저장하는 로그(대부분 하드디스크에 저장)를 온라인 로그(on-line log)

-테이프 저장장치에 저장하는 로그를 보관 로그(archival log)

2) 회복의 기본전략

1) Redo

-장애 발생으로 인하여 데이터베이스의 내용 자체가 손상된 경우에 사용

-복제본 이후에 발생한 데이터베이스 변경 내용들을 로그에 기록된 내용을 이용하여 재실행(redo, →)함으로써 데이터베이스 내용을 복원

2) Undo

-장애 발생으로 인하여 데이터베이스의 내용 자체는 손상되지 않았지만 갱신하였거나 갱신 중인 내용에 대한 신뢰성을 상실한 경우에 사용

-장애 발생시에 데이터베이스에 대한 모든 변경 사항들을 취소(undo, ←)시킴으로써 데이터베이스 내용을 복원

3. 회복기법

1) 로그 기반 회복 기법

변경(갱신) 내용을 언제 데이터베이스에 반영하느냐에 구

분

① 자연 갱신 회복 기법

-트랜잭션이 부분 완료(partially committed) 상태에 이르기 전까지 발생한 모든 변경 내용을 로그 파일에만 저장하고, 데이터베이스에 저장하는 것을 지연시키는 방법W
-트랜잭션이 수행도중 실패(failed) 상태에 도달하여 트랜잭션을 철회(abort)할 경우에는 로그 파일에 저장된 내용만 폐기 처리
-자연 갱신 회복 기법은 데이터베이스의 회복 과정에서 undo 연산실행 불필요

② 즉시 갱신 회복 기법

-미완료 갱신(uncommitted update)이라고도 함
-트랜잭션 수행도중 발생하는 변경 내용을 즉시 데이터베이스에 반영하는 방법
-로그 파일에 저장된 내용을 참조하여 undo 연산을 수행

2) 검사점 회복 기법

트랜잭션 수행 중 발생하는 변경 내용을 로그 파일에 기록하되 일정 기간 단위(검사시점 : checkpoint)로 검사시점을 로그 파일에 기록

3) 그림자 페이지 회복 기법

-현재 페이지 테이블(current page table)과 그림자 페이지 테이블(shadow page table)을 이용
(로그파일 이용 x)
-현재 페이지 테이블은 주기억장치에 저장하고, 그림자 페이지 테이블은 하드디스크(비휘발성 저장장치)에 저장
-장애가 발생할 경우 트랜잭션 시작 시 최초의 내용을 저장하고 있는 그림자 페이지를 이용하여 회복 처리가 가능

4. 트랜잭션(TR : TTransaction)**1) 트랜잭션의 특성(ACID특성)****① 원자성(Atomicity): 가장 중요**

- 모두 반영되거나 아니면 전혀 반영되지 아니어야 된다.
(부분 실행 안됨, All or Nothing)

② 일관성(Consistency)

- 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있게 DB 상태로 변환

③ 독립성,격리성(Isolation)

- 둘 이상의 트랜잭션이 동시에 병행 실행되고 있을 때

또 다른 하나의 트랜잭션의 연산이 끼어들 수 없다.

④ 영속성,지속성(Durability)

- 트랜잭션의 결과는 영구적으로 반영

2) 트랜잭션 상태**① 활동(Active)**

: 트랜잭션의 실행을 시작하였거나 실행 중인 상태

② 부분적 완료(Partially committed)

:트랜잭션의 마지막 명령문을 성공적으로 실행한 직후의 상태

③ 실패(Failed)

:트랜잭션이 정상적인 실행을 더 이상 수행할 수 없는 상태

④ 철회(Aborted)

:트랜잭션이 실행에 실패하여 rollback 연산을 수행한 상태

⑤ 완료(Committed)

:상대는 트랜잭션이 실행을 성공적으로 완료하여 commit 연산을 수행한 상태

*** 철회 트랜잭션 처리방법****① 트랜잭션 재시작**

트랜잭션의 철회 원인이 트랜잭션의 내부적인 논리 오류가 아니라, 하드웨어 또는 소프트웨어적인 오류로 인하여 발생하였을 경우에 사용하는 것

② 트랜잭션 폐기

트랜잭션의 내부적인 논리 오류에 의하여 발생하였을 경우에 사용하는 것
(철회 트랜잭션을 재시작하여도 동일한 결과를 초래할 것이기 때문에 트랜잭션을 폐기)

5. 동시성 제어의 필요성**1) 갱신내용손실**

동일한 필드의 내용을 두 개의 트랜잭션이 동시에 이용함으로써 갱신 내용이 손실되는 결과를 초래

2) 모순성

트랜잭션이 성공적으로 수행한 결과값으로 가져올 경우 두 트랜잭션이 동시에 수행되면서 데이터의 일관성이 없어지게 됨

3) 연쇄적인 복귀

트랜잭션의 연산결과를 복귀시키지 못함으로써 이전 값으

로 복귀할 수 없게됨

6. 동시성 제어 기법**1) 로킹 기법**

-트랜잭션들이 사용하는 자원(데이터 항목)에 대하여 상호 배제(mutual exclusive) 기능을 제공하는 것
-잠금(lock)을 설정하면 잠금을 설정한 트랜잭션은 잠금을 해제(unlock)할 때까지 데이터 항목을 독점적으로 사용
-lock(x) 연산을 먼저 실행, 트랜잭션 T의 실행 종료 이전에 unlock(x) 연산을 실행하여 잠금을 해제해야 함

로킹 연산의 종류*① 공유잠금(Shared-lock)**

: 판독(read(x)) 연산은 실행할 수 있으나 기록(write(x)) 연산은 실행 불가능

② 전용잠금(Exclusive-lock)

:판독(read(x)) 연산은 물론 기록(write(x)) 연산도 실행 가능

2단계 로킹 프로토콜: 동시성제어기법 중 가장 많이 사용 (2PLP : Two Phase Locking Protocol)*① 확장단계(Growing phase)**

: 트랜잭션들은 잠금(lock) 연산만을 수행할 수 있고 해제(unlock) 연산은 수행할 수 없다.

② 축소단계(Shrinking phase)

: 트랜잭션들은 해제(unlock) 연산만을 수행할 수 있고 잠금(lock) 연산은 수행할 수 없다.

2) 시간스탬프순서 기법

고유 번호인 시간 스탬프(timestamp)를 트랜잭션에 부여하는 것

7. 불필요한 컬럼이 포함된 기본키

최소한의 컬럼 혹은 컬럼들의 집합으로 후보키를 구성해야 함

8. 자기 참조(self reference)

한 테이블이 자신을 참조하는 경우

9. 업무규칙

참조무결성 제약조건을 유지하기 위한 규칙

10. 인덱스 설계

-테이블에 대한 검색 속도를 향상시킬 수 있는 수단
-검색의 기준이 되는 컬럼만 뽑아 정렬한 상태를 유지
-인덱스의 각 튜플은 원래 데이터가 저장되어 있는 테이블에 대응하는 튜플의 주소값을 가짐
-순차접근검색(full scan search) 단점 보완
**순차접근에 의한 검색은 튜플수가 많아지면 시간이 매우 오래걸리기 때문에 데이터베이스와 같이 대량의 데이터에 대한 검색 방법으로는 적당하지 않음
-이진검색(binary search) 단점 보완
**알고리즘에 의해 적은 횟수의 검색만으로 원하는 튜플을 찾을 수 있으나 수시로 입력되거나 삭제되는 대량의 데이터들을 정렬한 상태로 유지하는것은 많은 비용이 들어 현실성이 어려움

***c.f> 검색방법 (검색시간이 짧은 순으로 정렬)**

- ① 찾고자 하는 튜플의 주소값(보통 ROW-ID라고 부른다)을 알면 한 번의 검색으로 찾을 수 있다.
- ② 해쉬(hash) 함수를 적용할 수 있도록 데이터가 저장된 경우
- ③ 인덱스를 가지고 검색
- ④ 순차 접근에 의한 검색

11. 트리거(trigger)

-데이터베이스의 무결성을 유지하기 위한 일반적이고 강력한 도구

-이벤트-조건-동작(ECA) 규칙

CREATE TRIGGER <트리거이름>

AFTER <트리거를 유발하는 이벤트들이 OR로 연결된 리스트> ON <릴레이션> ← 이벤트
[WHEN<조건>] ← 조건
BEGIN<SQL문(들)>END ← 동작

-트리거가 제약조건과 유사하지만 어떤 이벤트가 발생했을 때 조건이 참이 되면 트리거와 연관된 동작이 수행되고, 그렇지 않으면 아무 동작도 수행되지 않음.
-트리거는 능동 데이터베이스(active database)의 중요한 특징

-트리거를 과도하게 사용한 경우:복잡한 상호 의존성 야기(한 트리거 내의 SQL문이 다른 트리거의 활성화를 유발하면 트리거들이 연쇄됨)

-트리거 작성시 신중히 작성해야 함
(조건만 맞으면 자동적으로 수행되기에 사용자가 트리거의 영향을 인식하지 못할 수도 있음)

Part12. 기출문제 용어

1. MARC

[**Machine** Readable Cataloging, 기계가독목록]

: 컴퓨터를 통해 목록 데이터를 효율적으로 처리하기 위해서는 모든 데이터를 컴퓨터가 인식할 수 있는 형식으로 변환시켜, 이를 정형화된 형식으로 배열해야 하는데, MARC는 이와 같이 컴퓨터가 목록 데이터(저자, 서명, 형태, 출판사항 등)를 식별하여 축적 · 유통할 수 있도록 코드화한 일련의 메타데이터 표준 형식이다.

MARC는 도서관의 자동화된 목록 작성에 사용되는 대표적인 메타데이터 형식 표준으로 도서관 간에 목록 레코드를 상호 교환하기 위해 미국 의회 도서관(Library of Congress, LC)이 개발하였다.

2. 메타데이터 [metadata]

: 데이터에 관한 구조화된 데이터로, 다른 데이터를 설명해 주는 데이터.

속성정보라고도 한다. 대량의 정보 가운데에서 찾고 있는 정보를 효율적으로 찾아내서 이용하기 위해 일정한 규칙에 따라 콘텐츠에 대하여 부여되는 데이터이다. 여기에는 콘텐츠의 위치와 내용, 작성자에 관한 정보, 권리 조건, 이용 조건, 이용 내역 등이 기록되어 있다. 컴퓨터에서는 보통 메타데이터를 데이터를 표현하기 위한 목적과 데이터를 빨리 찾기 위한 목적으로 사용하고 있다.

3. 상호 운용성 [interoperability]

같은 기종 또는 다른 기종의 기기끼리 상호간에 통신할 수 있고, 정보 교환이나 일련의 처리를 정확하게 실행할 수 있는 것.

4. MODS (Metadata Object Description Schema)

: MODS는 디지털 객체의 서지정보를 위한 표준 메타데이터
세계적으로 통용되는 메타데이터들과의 매핑테이블이 작성되어 있어 메타데이터 상호운용성이 입증된 메타데이터

5. MDR (MetaData Registry, 메타데이터 등록소)

: 메타데이터의 등록과 인증을 통하여 메타데이터를 유지 · 관리하며, 메타데이터의 명세를 공유하는 레지스트리. 메타데이터를 사용하여 데이터에 대한 접근과 사용을 촉진하고 메타데이터가 설명하는 특징에 따른 데이터의 조작을 가능하게 한다.

6. SMS (System Management Service)

: 시스템의 CPU, Memory, Disk 등의 자원 사용률을 모니터링 하여 로그로 남기고, 임계치 이상의 사용률을 보일 경우 알람 메시지나 메일 등으로 사용자에게 알려주는 서비스

7. APM (Application Performance Management)

: 최종 사용자에게 향상된 서비스를 제공하기 위해 애플리케이션의 흐름 모니터링과 성능 예측을 통해 최적의 애플리케이션 상태를 보장하고 관리하는 것
- Application : 전용 프로그램 (게임, 워드프로세서 등)

8. CDP (Continuous Data Protection, 지속 데이터보호)

: 저장한 데이터가 변할 때마다 자동으로 복사 저장하는 백업
- 어느 때나 데이터 복원이 가능하며, 기존 백업이 행해진 시점에서만 데이터 복원이 가능한 것과 다르다.

9. Data De-duplication (데이터 중복 제거 기술)

: 중복된 데이터를 제거하여 물리적인 스토리지 용량을 줄일 수 있는 비용 절감 솔루션

10. 접근통제 (Access Control)

: 사용자(주체)의 신원을 식별하고 인증하여 대상 정보(객체)의 접근, 사용 수준을 인가하는 절차

11. 강제적 접근통제 (규칙기반)

(MAC, Mandatory Access Control)

: 어떤 주체(사용자)가 특정 객체(데이터베이스)에 접근시 양자의 보안 레벨에 기초하여 낮은 수준의 주체가 높은 수준의 객체의 정보에 접근하는 것을 제한하는 방법. 컴퓨터의 모든 자원(중앙 처리 장치, 메모리, 프린터, 모니터, 저장 장치 등)을 객체로 추상화하고, 그 객체를 사용하고자 하는 것을 주체(사용자 및 모든 프로세스)로 설정하여, 각 객체 파일의 비밀 등급과 각 주체의 허가 등급을 보여준다. 이후, 주체가 객체를 읽거나 기록하거나 실행시키고자 할 때마다, 그 주체가 그 객체에 대한 권한을 가지고 있는지를 확인하는 방식이다. (예: 1급 비밀, 2급 비밀 등)

12. 임의적 접근통제 (신분기반)

(DAC, Discretionary Access Control)

: 시스템 객체에 대한 접근을 사용자 개인 또는 그룹의 식별자를 기반으로 하는 방법. 임의라는 말은 어떤 종류의 접근 권한을 갖는 사용자는 다른 사용자에게 자신의

판단에 따라 권한을 줄 수 있다는 뜻이다.

(예: 아이디/패스워드 이용해서 데이터베이스 접근)

- 접근 제어 목록 (ACL)

: 객체에 대한 접근이 허가된 주체들과 이들 주체가 허가 받은 접근 종류들이 기록된 목록

13. 역할기반 접근통제

(RBAC, Role Based Access Control)

: 주체에 대한 행위나 역할에 의해 접근 권한이 결정되는 접근 제어 방식. (예: 팀장, 팀원 등 직급별)

14. ETT (Extraction, Transformation, Transportation)

: 기업의 내부 및 외부 데이터를 추출하여 정제 후에 데이터 웨어하우스에 적재하는 과정
- ETT 가 중요한 이유 : 데이터의 중요성

15. DQM3 (Data Quality Management Model)

: 조직의 현재 데이터 품질관리 수준을 진단하고 데이터 품질 향상을 위해 도입해야 할 개선 과제 및 방안을 단계적, 체계적으로 제시하기 위해 국내에서 개발된 데이터 품질관리 프로세스의 수준을 평가하는 모형
- 데이터 품질 관리를 위한 가이드라인 제공

16. 올랩 (Online Analytical Processing, 온라인 분석 처리, OLAP)

: 데이터 웨어하우스를 다차원적으로 분석하고 시각화하는 시스템
(예. 고객의 구매 내역을 분석하고 매출을 지역별로 또는 제품별, 월별 등 다양한 차원에서 즉시 분석)

17. DOLAP (Desktop OLAP)

: 데이터를 PC로 내려서 PC 내에서 OLAP 수행

18. ROLAP (Relational OLAP)

: 서버측에서 관계형 DB 를 사용하고 클라이언트에서 다차원 분석하므로 속도는 느림
- MOLAP 에 비해 이미 표준화된 관계형 DB 를 사용하기 때문에 업체 간 호환성 우수

19. MOLAP (Multidimensional OLAP)

: 다차원 데이터베이스(배열)에 기반한 OLAP 아키텍처.

20. HOLAP (Hybrid OLAP)

: ROLAP 장점 + MOLAP 장점

21. MMDB (Main Memory DB)

: 데이터베이스 전체를 주 기억장치에 상주시켜서 처리하는 데이터베이스
- 디스크 기반 데이터베이스보다 10배 이상 빠른 트랜잭션 처리

22. ASSERTION (주장)

: 트리거와 반대로 제약조건을 위반하는 연산이 수행되지 않도록 하는 것이다.
- DB 무결성을 시행하는 메커니즘

CREATE ASSERTION 이름

CHECK 조건;

ex) 사원의 급여가 자신이 근무하는 부서의 관리자의 급여보다 많으면 안된다.

23. 인덱스 종류

1) 논리적 인덱스

- **Single 인덱스** : 하나의 컬럼으로 생성되는 인덱스
- **Concatenated 인덱스** : 여러 개의 컬럼으로 생성되는 인덱스
- **Unique 인덱스** : 컬럼의 속성이 유일한 값으로 저장되어 있는 컬럼에 대한 인덱스
- **NonUnique 인덱스** : 컬럼의 속성이 유일하지 않는 값으로 저장되어 있는 컬럼에 대한 인덱스

2) 물리적 인덱스

- **Balance-Tree 인덱스** : 물리적 구조가 좌우 대칭 구조 (일반적인 인덱스)
- **Reverse 인덱스** : 밸런스트리 인덱스의 구조와 동일한 메커니즘을 가지고 있으나 거꾸로 인덱스를 만듦
→ 인덱스 블록 여러 개에 분산되어 저장 → 해당 값들에 대한 연속적인 삭제 작업이 발생하더라도 어느 한쪽 블록에서의 데이터가 삭제되는 현상을 막을 수 있는 인덱스 (불균형 문제 해결)
- **합수 기반 인덱스** : 오라클 8i 버전부터 새롭게 추가된 인덱스 기법 → WHERE 절에 산술 표현 또는 함수를 자주 사용할 때 빠른 검색 속도 보장
- **IOT 인덱스** : 인덱스는 테이블과 독립적인 저장 구조를 가지지만 IOT를 생성하면 인덱스와 테이블이 같은 저장 구조에 생성되어 빠른 검색이 가능하고 저장 공간이 적게 사용됨
- **비트맵 인덱스** : 인덱스된 컬럼의 정보를 0과 1의 값으로 표현 → 저장 공간의 사용이 감소되고 아주 큰 테이블에서 검색할 때 효과적

24. 시퀀스 (SEQUENCE)

- : 연속적인 숫자 값을 자동적으로 증가시켜야 하는 경우 사용하는 객체 (ex. 001부터 100까지 번호 생성)
- 생성 : CREATE SEQUENCE [시퀀스명]...
 - 삭제 : DROP SEQUENCE [시퀀스명];

25. 시노님 (SYNONYM)

- : 시노님은 오라클 객체(테이블, 뷰, 시퀀스, 프로시저)에 대한 대체이름(Alias)를 말하며, 실질적으로 그 자체가 객체가 아니라 객체에 대한 직접적인 참조 이다.
- 실제 이름과 소유자 그리고 위치를 감춤으로써 데이터베이스 보안을 개선하는데 사용
 - 짧은 이름 사용 → SQL 단순화

26. ODBC (Open DataBase Connectivity, 개방형 데이터베이스 접속성)

- : 윈도즈 응용 프로그램에서 다양한 데이터베이스 관리 시스템(DBMS)에 접근하여 사용할 수 있도록 개발한 표준 개방형 응용 프로그램 인터페이스(API) 규격.

27. JDBC (Java DataBase Connectivity, 자바 데이터베이스 접속성)

- : 자바(Java) 프로그램이 데이터베이스에 연결하고 구조화 조회 언어(SQL)로 질의를 하며, 자료를 갱신할 수 있도록 제공하는 응용 프로그램 인터페이스(API).