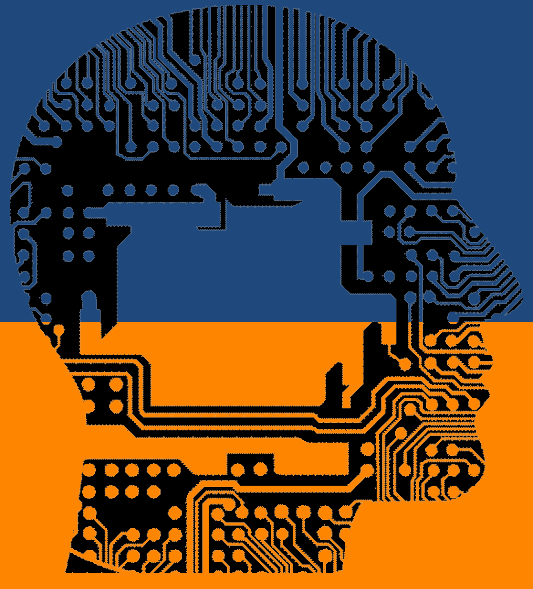
한국IT진흥부설 정보보호교육학원 아이섹

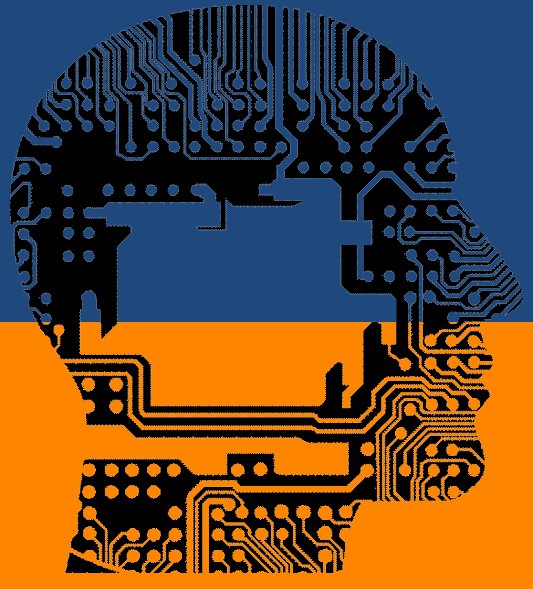


JAVA 웹 개발자 양성과정
DataBase - SQLD

2강 - 데이터 모델과 성능

By SoonGu Hong

한국IT진흥부설
정보보호교육학원 아이섹



JAVA 웹 개발자 양성과정
DataBase

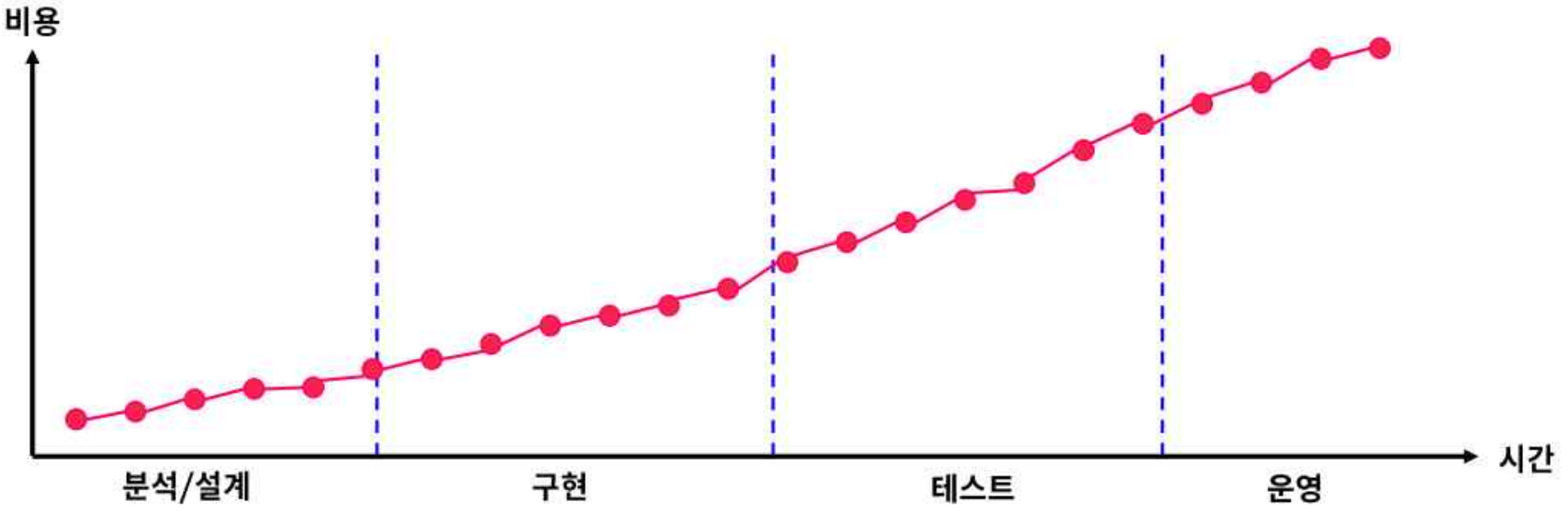
1. 성능 데이터 모델링 개요

➢ 성능 데이터 모델링의 정의

① 성능 데이터 모델링이란 데이터베이스 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 것으로 정의할 수 있다.

➢ 성능 데이터 모델링 수행시점

① 성능 향상을 위한 비용은 프로젝트 수행 중에 있어서 사전에 할수록 비용이 들지 않는다.
② 분석/설계 단계에서 데이터 모델에 성능을 고려한 데이터 모델링을 수행할 경우 성능 저하에 따른 재 업무(Rework) 비용을 최소화 할 수 있는 기회를 가지게 된다.
③ 분석/설계 단계에서 데이터베이스 처리 성능을 향상시킬 수 있는 방법을 주도면밀하게 고려해야 한다.

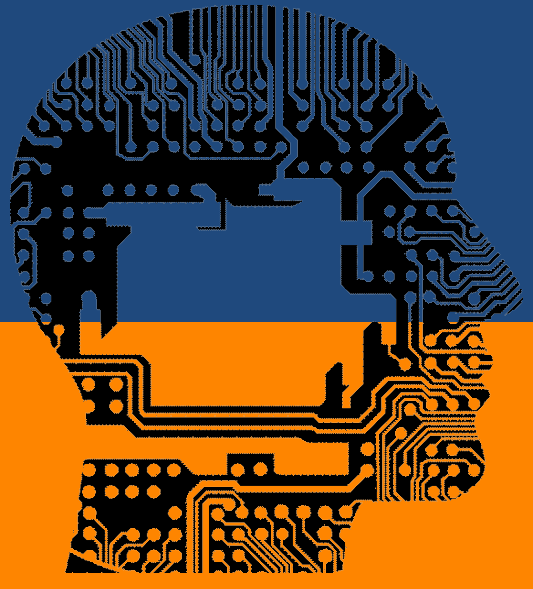


➢ 성능 데이터 모델링 고려사항

① 데이터 모델링을 할 때 **정규화**를 정확하게 수행한다.
② 데이터베이스 **용량 산정**을 수행한다.
③ 데이터베이스에 발생되는 **트랜잭션의 유형**을 파악한다.
④ 용량과 트랜잭션의 유형에 따라 **반 정규화**를 수행한다.
⑤ **이력 모델의 조정, PK/FK조정, 슈퍼 타입/서브타입 조정** 등을 수행한다.
⑥ 성능관점에서 **데이터 모델을 검증**한다.

- 데이터 모델링을 할 때 기본적으로 정규화를 완벽하게 수행해야 한다. 정규화된 모델이 데이터를 주요 관심사별로 분산시키는 효과가 있기 때문에 그 자체로 성능을 향상시키는 효과가 있다.
- 데이터 모델에 발생되는 트랜잭션의 유형을 파악할 필요가 있다. 트랜잭션의 유형에 대한 파악은 CRUD 매트릭스를 보고 파악하는 것도 좋은 방법이 될 수 있고 객체지향 모델링을 적용한다면 시퀀스 다이어그램을 보면 트랜잭션의 유형을 파악하기에 용이하다.
- 파악된 용량 산정과 트랜잭션의 유형데이터를 근거로 정확하게 테이블에 대해 반 정규화를 적용하도록 한다. 반 정규화는 테이블, 속성, 관계에 대해 포괄적인 반 정규화의 방법을 적용해야 한다.
- 대량 데이터가 처리되는 이력 모델에 대해 성능 고려를 하고 PK/FK의 순서가 인덱스 특성에 따라 성능에 영향을 미치는 영향도가 크기 때문에 반드시 PK/FK를 성능이 우수한 순서대로 칼럼의 순서를 조정해야 한다.

한국IT진흥부설
정보보호교육학원 아이섹



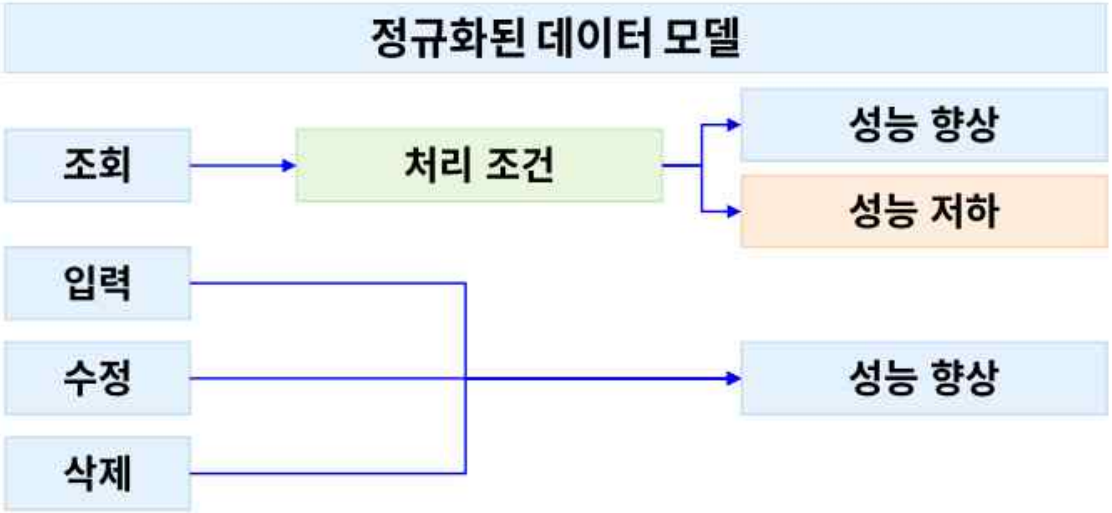
JAVA 웹 개발자 양성과정

DataBase

2. 정규화와 성능

➢ 정규화를 통한 성능 향상 전략

① 정규화를 수행한다는 것은 데이터를 결정하는 **결정자에 의해 함수적 종속을 가지고 있는 일반 속성을 의존자로 하여 입력/수정/삭제 이상현상을 제거**하는 것이다.
② 데이터의 중복 속성을 제거하고 결정자에 의해 동일한 의미의 일반 속성이 하나의 테이블로 집약되므로 **한 테이블의 데이터 용량이 최소화되는 효과**가 있다.
③ 정규화된 테이블은 데이터를 처리할 때 **속도가 빨라질 수도 있고 느려질 수도 있는 특성**이 있다.



- 정규화된 데이터 모델은 조회 시에는 처리 조건에 따라 성능이 향상 혹은 저하 된다.
- 정규화된 데이터 모델은 입력/수정/삭제 시 무조건 성능이 향상된다.

➢ 정규화 용어

용어	설명
정규화 (Normalization)	- 함수적 종속성(FD: Functional Dependency) 등과 같은 이론에 근거하여 관계형 데이터베이스 테이블의 삽입·삭제·갱신 이상(Anomaly) 현상 발생을 최소화하기 위해 좀 더 작은 단위의 테이블로 설계하는 과정. 즉, 데이터 모델을 정규형에 맞도록 고치는 과정
정규형 (NF: Normal Form)	- 정규화 규정. 정규화 결과에 의해 도출된 데이터 모델이 갖춰야 할 특성
함수적 종속성 (FD: Functional Dependency)	- 테이블의 특정 컬럼 A의 값을 알면 다른 컬럼 B 값을 알 수 있을 때, 컬럼 B는 컬럼 A에 함수적 종속성이 있다고 함(ex. 고객명은 고객주민등록번호에 함수적 종속성이 있음)
결정자(Determinant)	- 함수적 종속성 설명에서, 컬럼 A를 결정자라고 함
다치종속 (MVD: MultiValued Dependency)	- 결정자 컬럼 A에 의해 컬럼 B의 값을 다수 개 알 수 있을 때, 컬럼 B는 컬럼 A에 다치종속 되었다고 함 - (ex. 학번을 알면 해당 학생의 다수 개 수강과목을 알 수 있을 때, 수강과목은 학번에 다치종속관계임)

➢ 정규화 효과 및 장점

① 상호 종속성이 강한 데이터 요소들을 분리, 독립된 개념(엔티티, 테이블)으로 정의하게 됨에 따라 High Cohesion & Loose Coupling 원칙에 충실해지며 이로 인해 **유연성**이 극대화 됨

② 개념이 좀 더 세분화됨에 따라 해당 개념에 대한 **재활용** 가능성이 높아짐(일반적으로 각종 참조 모델은 정규형을 만족하고 있음)

③ Non-key 데이터 요소가 한번 만 표현됨에 따라 **중복이 최소화** 됨(데이터 품질확보, 저장공간 절약, DML 성능)

➢ 정규화 이론

① 1차, 2차, 3차, 보이스코드정규화는 함수종속성에 근거하여 정규화를 수행하고,
② 4차정규화는 속성의 값이 여러 개 발생하는 다치종속,
③ 5차 정규화는 조인에 의해 발생하는 이상현상 제거로 정규화 수행한다.

정규화 유형	설명
1차 정규화	- 함수종속, 복수의 속성값을 갖는 속성을 분리, 속성의 원자성 확보
2차 정규화	- 함수종속, 주식별자에 완전종속적이지 않은 속성의 분리 - 부분종속 속성(PARTIAL DEPENDENCY ATTRIBUTE) 을 분리
3차 정규화	- 함수종속, 일반속성에 종속적인 속성의 분리 - 이전종속 속성(TRANSITIVE DEPENDENCY)의 분리
보이스-코드 정규화	- 함수종속, 결정자안에 함수종속을 가진 주식별자 분리
4차 정규화	- 다가 종속(Multi-Valued Dependency) 속성분리
5차 정규화	- 결합종속(Join Dependency)일 경우는 두개 이상의 N개로 분리

➢ 제 1정규형

① 모든 속성은 원자 값을 가져야 함
② 다중 값을 가질 수 있는 속성은 분리되어야 함

▪ 제 1정규형 위반

- 회원 엔터티

회원아이디	나이	성별	회원구분	연락처
sujilee	3	여	프리미엄	010-1234-1235, sujilee@naver.com
kolee	36	남	프리미엄	010-1234-1236, kolee@naver.com
yhyoon	34	여	일반	010-1234-1237, yhyoon@naver.com
boralee	38	여	일반	010-1234-1238, boralee@naver.com
ijlee	42	남	프리미엄	010-1234-1239, ijlee@naver.com

- 연락처 속성에 두 가지 값이 들어가 있음
- 이럴 경우 제 1 정규형 위반임

➢ 제 1정규형 - 계속

▪ 제 1정규형 위반 해소

- 회원 엔터티

회원아이디	나이	성별	회원구분
sujilee	3	여	프리미엄
kolee	36	남	프리미엄
yhyoon	34	여	일반
boralee	38	여	일반
ijlee	42	남	프리미엄

- 연락처 속성을 삭제

- 회원연락처 엔터티

회원아이디	연락처구분	연락처
sujilee	휴대폰	010-1234-1235
sujilee	이메일	sujilee@naver.com
kolee	휴대폰	010-1234-1236
kolee	이메일	kolee@naver.com
yhyoon	휴대폰	010-1234-1237
yhyoon	이메일	yhyoon@naver.com
boralee	휴대폰	010-1234-1238
boralee	이메일	boralee@naver.com
ijlee	휴대폰	010-1234-1239
ijlee	이메일	ijlee@naver.com

- 기존의 회원 엔터티에서 **회원연락처 엔터티를 추가하여 제 1정규형을 만족**하게 함

- 회원연락처 엔터티 추가

➢ 제 2정규형

① 제 1정규형을 만족하고 모든 Non-key 컬럼은 기본 키 전체에 종속되어야 함
② 기본 키에 종속적이지 않거나 기본 키 일부 컬럼(들)에만 종속적인 컬럼은 분리되어야 함

▪ 제 2정규형 위반

- 주문 테이블

고객아이디	주문번호 PK	주문일자	고객명	고객등급
C00001	20200505000001	20200505	이경오	프리미엄
C00001	20200505000002	20200505	이경오	프리미엄
C00001	20200505000003	20200505	이경오	프리미엄
C00002	20200505000004	20200505	이수지	일반
C00002	20200505000005	20200505	이수지	일반

- 올바른 집합 단위에 기초하고 있지 않음
- 갱신 시에 갱신 이상이 발생할 가능성 존재
- 주문 시마다 고객정보를 저장해야 함
- 고객 정보의 중복이 발생할 수 있음
- 고객 정보를 모르면 주문이 불가능 함

➢ 제 2정규형 - 계속

▪ 제 2정규형 위반 해소

- 고객주문 엔터티

고객아이디	주문번호	주문일자
C00001	20200505000001	20200505
C00001	20200505000002	20200505
C00001	20200505000003	20200505
C00002	20200505000004	20200505
C00002	20200505000005	20200505

- 고객 엔터티

고객아이디	고객명	고객등급
C0001	이경오	프리미엄
C0002	이수지	일반

- 모든 속성이 식별자만으로 함수 종속 가짐
- 식별자의 일부에만 종속하는 속성 없음

➢ 제 3정규형

① 제 2정규형을 만족하고 일반속성들간에도 종속관계가 존재하지 않아야 함
② 일반속성들 간 종속관계가 존재하는 것들은 분리되어야 함

▪ 제 3정규형 위반

- 고객 엔터티

고객아이디PK	고객명	나이	직업코드	직업명
C00001	이경오	36	J001	SQL개발자
C00002	이수지	27	J002	변호사
C00003	이나라	25	J003	미용사
C00004	이지수	24	J004	건축사
C00005	이효성	27	J005	용접공

- 식별자를 제외한 일반 속성 끼리 함수 종속이 발생
- 식별자 이외의 키 간 발생하는 함수의 종속

➢ 제 3정규형 - 계속

▪ 제 3정규형 위반 해소

- 고객 엔터티

고객아이디 PK	고객명	나이	직업코드
C00001	이경오	36	J001
C00002	이수지	27	J002
C00003	이나라	25	J003
C00004	이지수	24	J004
C00005	이효성	27	J005

- 직업 엔터티

직업코드	직업명
J001	SQL개발자
J002	변호사
J003	미용사
J004	건축사
J005	용접공

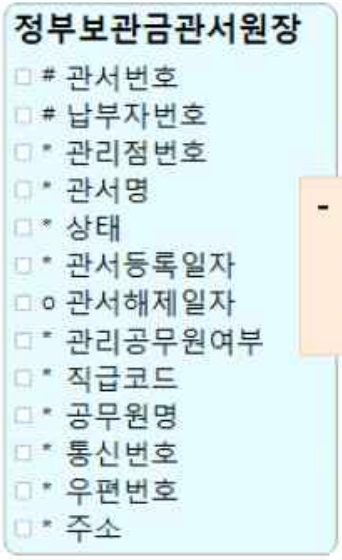
- 직업 엔터티을 추가하여 일반 속성 끼리의 함수 종속을 제거함

➢ 정규화와 성능

① 정규화를 수행해서 조인이 발생하게 되더라도 효율적인 **인덱스 사용을 통해 조인 연산을 수행**하면 성능 상 단점은 거의 없다.
② 정규화를 수행하여 소량의 테이블이 생성된다면 **소량의 테이블을 먼저 읽어 조인 연산을 수행하면 되므로 성능 상 유리**할 수 있다.
③ 정규화가 제대로 되지 않으면 동일한 종류의 속성을 여러 개 가지고 있어서 과다한 인덱스가 만들어 질 수 있는데 **정규화를 한다면 하나의 인덱스만 만들어도 된다.**

➢ 반정규화된 테이블의 성능저하 사례1

▪ 정규화 되지 않은 모델



- 2차 정규화가 안된 엔터티의 모습이다. 관서번호에만 함수종속되는 관서에 대한 속성들이 있는 것을 알 수 있다.

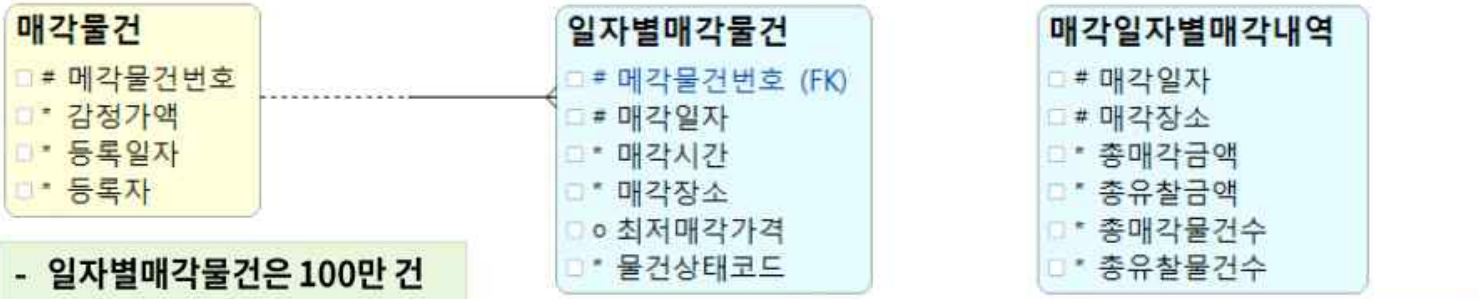
▪ 정규화가 된 모델

- 2차 정규화된 엔터티는 관서번호, 관서명이 관서테이블에만 존재하기 때문에 두 개의 테이블을 결합하여 처리해야 한다.

- 정부보관금관서원장에서 데이터를 조회하는 것이나, 관서와 정부보관금관서원장을 조인하여 데이터를 조회하나 처리 성능은 사용자가 느끼기에는 거의 차이가 나지 않는다. PK가 걸려있는 방향으로 조인이 걸려 Unique Index를 곧바로 찾아서 데이터를 조회하기 때문에, 하나의 테이블에서 조회하는 작업과 비교했을 때 미미하게 성능 차이가 날 뿐 사용자에게 크게 영향을 줄 만큼 성능이 저하되는 일은 없는 것이다.
- '관서등록일자가 2010년 이후 관서를 모두 조회하라'는 SQL 구문을 처리하는 것으로 바꾸면, 2차 정규화된 테이블이 훨씬 빠르다. 정규화 되지 않은 모델에서는 불필요하게 납부자번호만큼 누적된 데이터를 읽어서 결과를 구분하여 보여주어야 하지만 정규화된 모델에서는 관서수만큼만 존재하는 데이터를 읽어 곧바로 결과를 보여주기 때문이다.

➢ 반정규화된 테이블의 성능저하 사례2

▪ 정규화 되지 않은 모델



- 일자별매각물건은 100만 건
- 매각일자는 5천 건

- 매각일자별매각내역 2만건

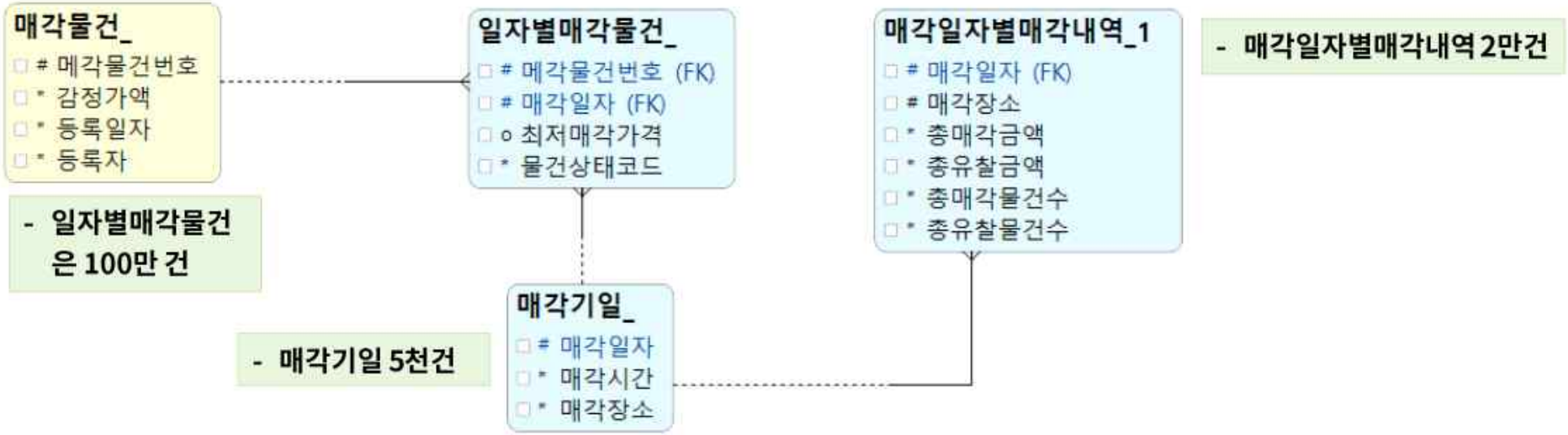
```
SELECT B.총매각금액 , B.총유찰금액
  FROM (SELECT DISTINCT 매각일자
          FROM 일자별매각물건
          WHERE 매각장소 = '서울 7호'
       ) A
     , 매각일자별매각내역 B
 WHERE A.매각일자 = B.매각일자
   AND A.매각장소 = B.매각장소;
```

- 특정 매각 장소에 대해 매각일자를 찾아 매각내역을 조회하려면
- 100만건의 데이터를 읽어 매각일자를 distinct하여
- 매각일자별매각내역과 조인되어야 한다.

- 100만건의 데이터를 읽어 DISTINCT함

➢ 반정규화된 테이블의 성능저하 사례2

▪ 정규화가 된 모델



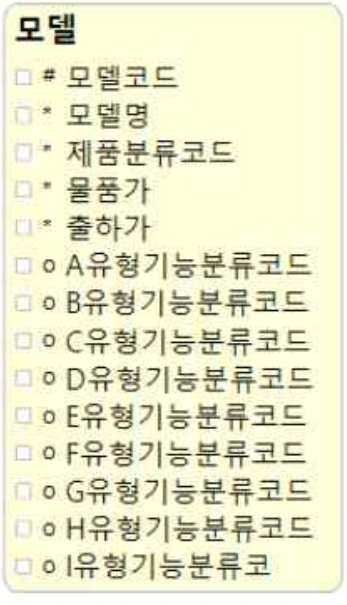

```
SELECT B.총매각금액
     , B.총유찰금액
  FROM 매각기일 A
     , 매각일자별매각내역 B
 WHERE A.매각장소 = '서울 7호'
   AND A.매각일자 = B.매각일자
   AND A.매각장소 = B.매각장소
;
```

- 5천건의 데이터를 읽음

- 매각일자를 PK로 하고 매각시간과 매각장소는 일반속성이 되었다. 정규화를 적용했기 때문에 매각일자를 PK로 사용하는 매각일자별매각내역과도 관계가 연결된다.
- 따라서 업무흐름에 따른 정확한 데이터 모델링 표기도 가능해지고, 데이터 건수가 약 5천건인 매각기일테이블이 드라이빙 테이블이 되므로 성능 상 유리하다.

➢ 반정규화된 테이블의 성능저하 사례3

▪ 정규화가 안된 모델



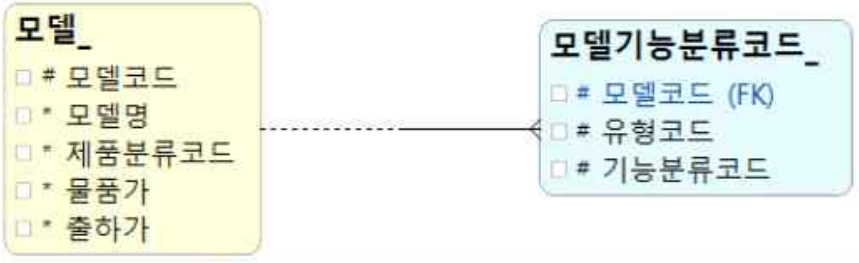
- 동일한 속성 형식을 두 개 이상의 속성으로 나열하여 반정규화한 경우에 해당

- 유형기능분류코드에 따라 데이터를 조회하는 경우가 많이 나타나 **인덱스를 생성하려면 유형기능분류코드 각각에 대해 인덱스를 생성해야 하므로 9개나 되는 인덱스를 추가 생성**해야 한다.

- 30만 건

```
CREATE IDX_모델_01 ON 모델(A유형기능분류코드);
CREATE IDX_모델_02 ON 모델(B유형기능분류코드);
CREATE IDX_모델_03 ON 모델(C유형기능분류코드);
CREATE IDX_모델_04 ON 모델(D유형기능분류코드);
CREATE IDX_모델_05 ON 모델(E유형기능분류코드);
CREATE IDX_모델_06 ON 모델(F유형기능분류코드);
CREATE IDX_모델_07 ON 모델(G유형기능분류코드);
CREATE IDX_모델_08 ON 모델(H유형기능분류코드);
CREATE IDX_모델_09 ON 모델(I유형기능분류코드);
```

▪ 정규화가 된 모델



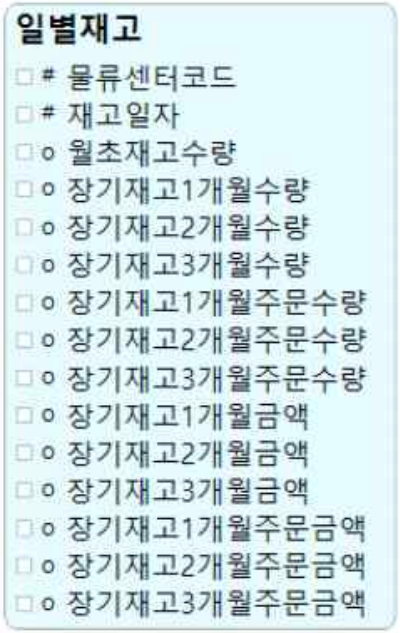
- 30만 건

- 하나의 테이블에 9개가 반복적으로 나열이 되어 있을 때는 인덱스 생성이 어려웠지만 **정규화되어 분리된 이후에는 인덱스 추가 생성이 0개가** 되었다. 또한 분리된 테이블 모델기능분류코드에서 PK인덱스를 생성하여 이용함으로써 성능이 향상될 수 있다.

```
CREATE IDX_모델기능분류코드__01 ON 모델기능분류코드_
(유형코드, 기능분류코드, 모델코드);
```

➢ 반정규화된 테이블의 성능저하 사례 4

▪ 정규화가 안된 모델



- 일별재고 엔터티 조회 시 과도한 부하가 발생할 수 있고
- 과도한 인덱스를 생성해야 한다.

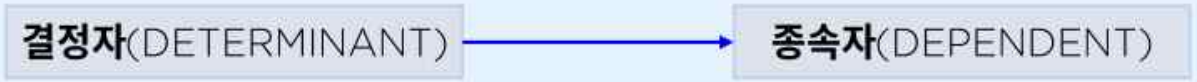
▪ 정규화가 된 모델

- 일별재고와 일별재고상세를 구분함으로써 일별재고에 발생되는 트랜잭션의 성능저하를 예방할 수 있게 되었다.

➢ 함수적 종속성(Functional Dependency)에 근거한 정규화 수행 필요

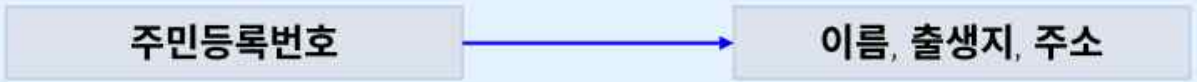
① 함수의 종속성(Functional Dependency)은 데이터들이 어떤 기준값에 의해 종속되는 현상을 지칭하는 것이다.
② 이 때 기준값을 결정자(Determinant)라 하고 종속되는 값을 종속자(Dependent)라고 한다.

함수종속성



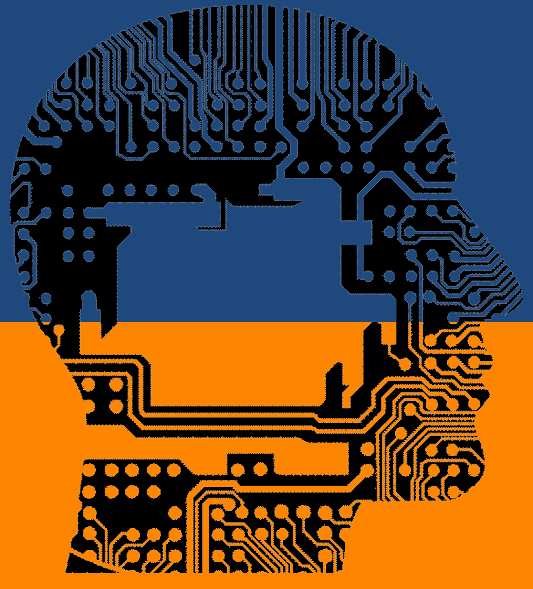
- 종속자는 근본적으로 결정자에 함수적으로 종속성을 가지고 있음

함수종속성



- 이름, 출생지, 주소는 주민등록번호에 함수 종속성을 가지고 있음
- 즉 "주민등록번호가 이름, 출생지, 주소를 함수적으로 결정한다."라고 말할 수 있다.

한국IT진흥부설
정보보호교육학원 아이섹



JAVA 웹 개발자 양성과정
DataBase

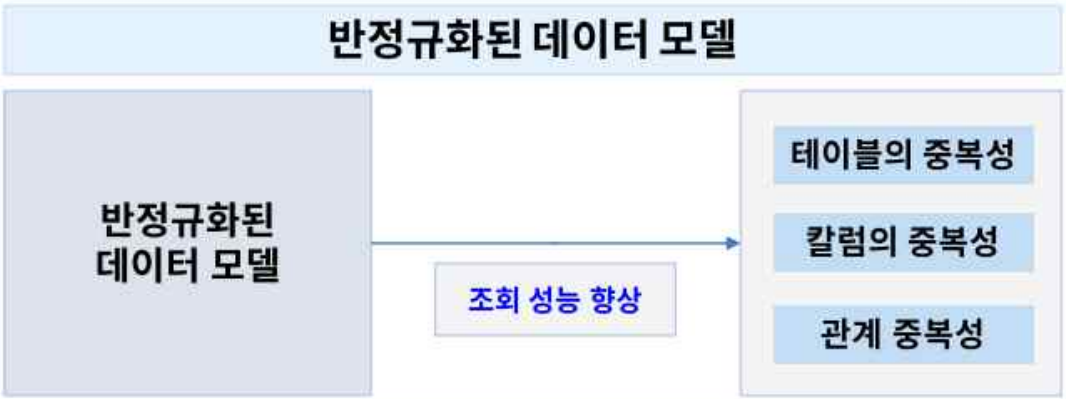
3. 반정규화와 성능

➢ 반정규화의 정의

① 정규화된 엔터티, 속성, 관계에 대해 시스템의 성능향상과 개발(Development)과 운영(Maintenance)의 단순화를 위해 **중복, 통합, 분리** 등을 수행하는 데이터 모델링의 기법을 의미

② 협의의 반정규화는 데이터를 중복하여 **성능을 향상시키기 위한 기법**이라고 정의할 수 있고 좀 더 넓은 의미의 반정규화는 성능을 향상시키기 위해 정규화된 데이터 모델에서 중복, 통합, 분리 등을 수행하는 모든 과정을 의미

③ 데이터 무결성이 깨질 수 있는 위험을 무릅쓰고 데이터를 중복하여 반정규화를 적용하는 이유는 데이터를 조회할 때 디스크 I/O량이 많아서 성능이 저하되거나 경로가 너무 멀어 조인으로 인한 성능저하가 예상되거나 칼럼을 계산하여 읽을 때 성능이 저하될 것이 예상되는 경우 반정규화를 수행하게 된다.



- 중복성의 원리를 활용하여 데이터 조회 시 성능을 향상시키는 역할을 할 수 있음

➢ 반정규화 절차

① 반정규화도 하나의 난이도 높은 데이터 모델링의 실무기술이다.
② 반정규화에 대한 필요성이 결정이 되면 칼럼의 반정규화 뿐만 아니라 테이블의 반정규화와 관계의 반정규화를 종합적으로 고려하여 적용
③ 반정규화를 막연하게 중복을 유도하는 것만을 수행하기 보다는 성능을 향상시킬 수 있는 **다른 방법들을 고려하고 그 이후에 반정규화를 적용하도록** 해야 한다.

1. 반정규화 대상 조사	2. 다른 방법 유도 검토	3. 반정규화 적용
• 범위 처리 빈도수 조사 • 대량의 범위 처리 조사 • 통계성 프로세스 조사 • 테이블 조인 개수	• 뷰 테이블 • 클러스터링 적용 • 인덱스의 조정 • 응용 애플리케이션	• 테이블 반정규화 • 속성의 반정규화 • 관계의 반정규화
- 반정규화의 대상을 조사한다.	- 반정규화의 대상에 대해 다른 방법으로 처리할 수 있는지 검토한다.	- 반정규화를 적용한다.

- 정규화의 대상을 조사하고 **다른 방법을 적용할 수 있는지 검토하고 그 이후에 반정규화를 적용**하도록 한다.

➢ 반정규화의 기법 – 테이블 반정규화

기법분류	기법	내용
테이블 병합	1:1관계 테이블 병합	- 1:1 관계를 통합하여 성능향상
	1:M 관계 테이블 병합	- 1:M 관계 통합하여 성능향상
	슈퍼/서브타입 테이블 병합	- 슈퍼/서브관계를 통합하여 성능향상
테이블 분할	수직분할	- 칼럼단위의 테이블을 디스크I/O 분산처리를 하기 위해 테이블을 1:1로 분리하여 성능 향상(트랜잭션의 처리되는 유형 파악이 선행되어야 함)
	수평분할	- 로우 단위로 집중 발생되는 트랜잭션을 분석하여 디스크 I/O 및 데이터 접근 효율을 높여 성능을 향상하기 위해 로우단위로 테이블을 쪼갬(관계가 없음)
테이블 추가	중복 테이블 추가	- 다른 업무이거나 서버가 다른 경우 동일한 테이블 구조를 중복하여 원격조인을 제거하여 성능을 향상
	통계 테이블 추가	- SUM, AVG 등을 미리 수행하여 계산해둠으로써 조회 시 성능을 향상
	이력 테이블 추가	- 이력 테이블 중에서 마스터 테이블에 존재하는 레코드를 중복하여 이력테이블에 존재하는 방법은 반정규화 유형
	부분 테이블 추가	- 하나의 테이블의 전체 칼럼 중 자주 이용하는데 자주 이용하는 집중화된 칼럼들이 있을 때 디스크I/O를 줄이기 위해 해당 칼럼들을 모아놓은 별도의 반정규화된 테이블을 생성

➢ 반정규화의 기법 – 칼럼의 반정규화

기법	내용
중복 칼럼 추가	조인에 의해 처리할 때 성능 저하를 예방하기 위해 즉, 조인을 감소시키기 위해 중복된 칼럼을 위치 시킴
파생 칼럼 추가	트랜잭션이 처리되는 시점에 계산에 의해 발생되는 값을 성능저하를 예방하기 위해 미리 값을 계산하여 칼럼에 보관함
이력 테이블 칼럼 추가	대량의 이력 데이터를 처리할 때 불특정날 조회나 최근값을 조회할 때 나타날수 있는 성능저하를 예방하기 위해 이력테이블에 칼럼 추가(최근값 여부, 시작과 종료일자 등)
PK에 의한 칼럼 추가	복합의미를 갖는 PK를 단일 속성으로 구성하였을 경우 발생됨, 단일 PK안에서 특정 값을 별도로 조회하는 경우 성능 저하가 발생함, 이때 이미 PK안에 데이터가 존재하지만 성능향상을 위해 일반속성으로 생성하는 방법이 PK에 의한 칼럼 추가 반정규화임
응용시스템의 오작동을 위한 칼럼 추가	업무적으로 의미가 없지만 사용자가 데이터 처리를 하다가 잘못 처리하여 원래의 값으로 복구를 원하는 경우 이전 데이터를 임시적으로 중복하여 보관하는 기법임 칼럼으로 이것을 보관하는 방법은 오작동 처리를 위한 임시적인 기법이지만 이것을 이력데이터 모델로 풀어내면 정상적인 데이터 모델의 기법이 될 수 있음

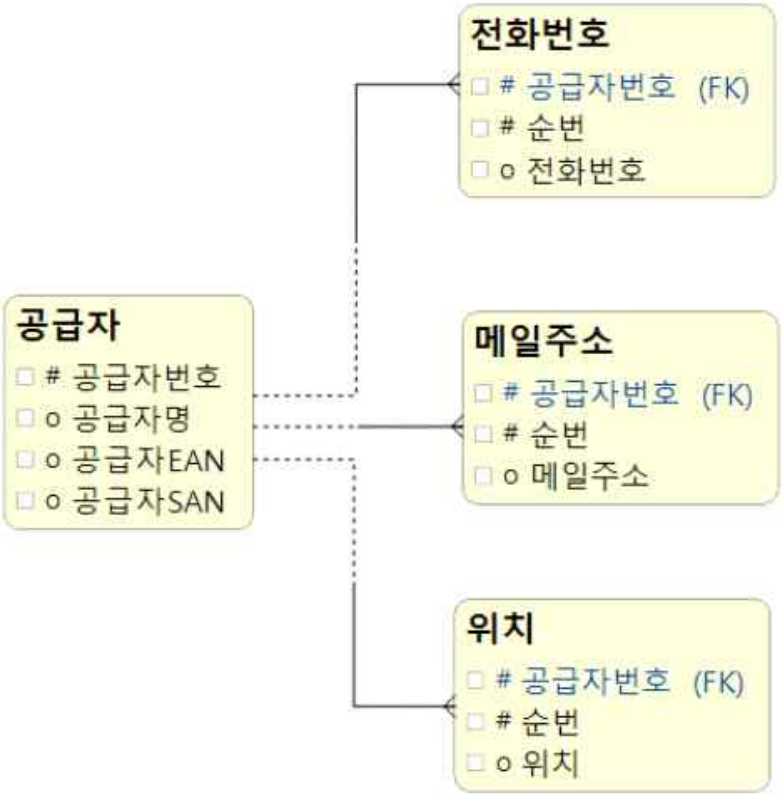
➢ 반정규화의 기법 – 관계 반정규화

기법	내용
중복 관계 추가	데이터를 처리하기 위한 여러 경로를 거쳐 조인이 가능하지만 이때 발생할 수 있는 성능저하를 예방하기 위해 추가적인 관계를 맺는 방법이 관계의 반정규화임

- 테이블과 칼럼의 반정규화는 데이터 무결성에 영향을 미치게 되나
- 관계의 반정규화는 데이터 무결성을 깨뜨릴 위험을 갖지 않고서도 데이터처리의 성능을 향상시킬 수 있는 반정규화의 기법이 된다.

➢ 정규화가 잘 정의된 데이터 모델에서 성능이 저하될 수 있는 경우

▪ 정규화가 된 모델



- 공급자와 전화번호, 메일주소, 위치는 1:M 관계이므로 한 명의 공급자당 여러 개의 전화번호, 메일주소, 위치가 존재한다.
- 따라서 가장 최근에 변경된 값을 가져오기 위해서는 조금 복잡한 조인이 발생될 수 밖에 없다.

▪ 반정규화가 된 모델


```
SELECT 공급자명
     , 전화번호
     , 메일주소
     , 위치
  FROM 공급자
 WHERE 공급자번호 BETWEEN '1001' AND '1005'
 ;
```

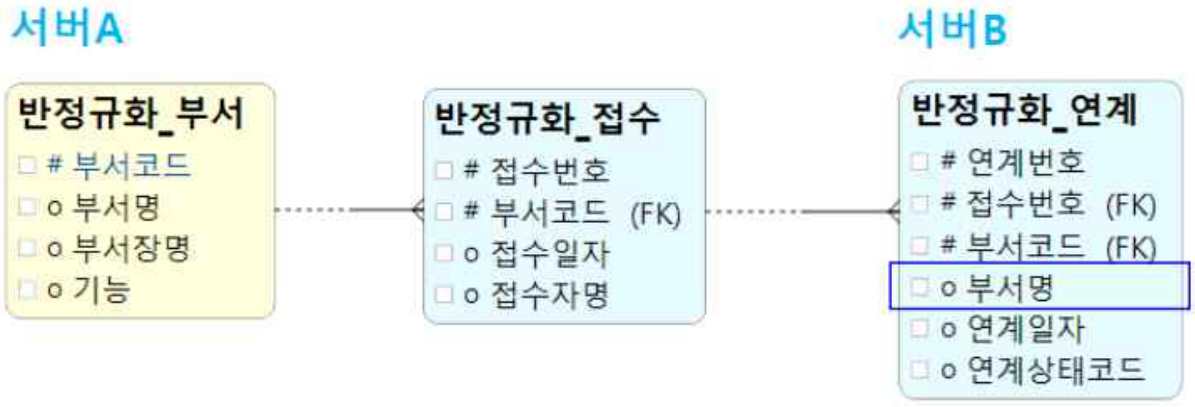
- 반정규화를 적용하면 즉, 가장 최근에 변경된 값을 공급자 엔터티에 위치시키면 아주 간단한 SQL 구문이 작성 된다.

➢ 정규화가 잘 정의된 데이터 모델에서 성능이 저하된 경우 2

▪ 정규화가 된 모델

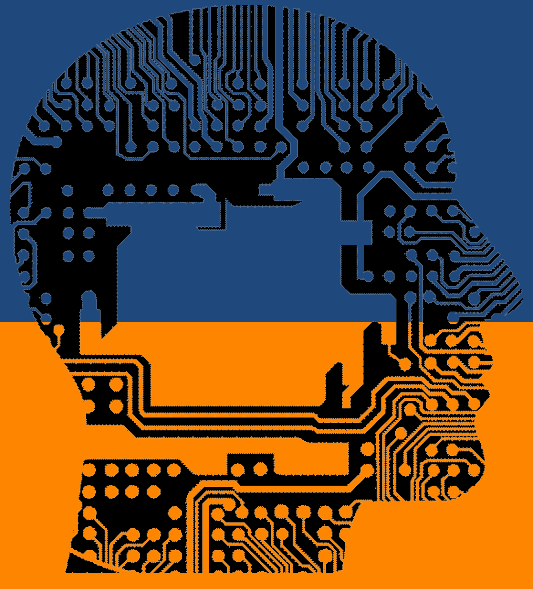
- 서버A에 부서와 접수 테이블이 있고 서버B에 연계라는 테이블이 있는데 서버B에서 데이터를 조회할 때 빈번하게 조회되는 **부서명**이 서버A에 존재하기 때문에 **연계, 접수, 부서 테이블이 모두 조인**이 걸리게 된다. **게다가 분산데이터베이스 환경이기 때문에 다른 서버간에도 조인이 걸리게 되어 성능이 저하**되는 것이다.

▪ 반정규화가 된 모델



- 연계 테이블에서 바로 **부서명**을 조회하면 된다.
- SQL구문도 간단해지고 분산되어 있는 서버 간에도 **DB LINK 조인이 발생하지 않아 성능이 개선**되었다.
- 반정규화를 적용할 때 기억해야 할 내용은 데이터를 입력, 수정, 삭제할 때는 성능이 떨어지는 점을 기억해야 하고 데이터의 무결성 유지에 주의를 해야 한다.

한국IT진흥부설
정보보호교육학원 아이섹

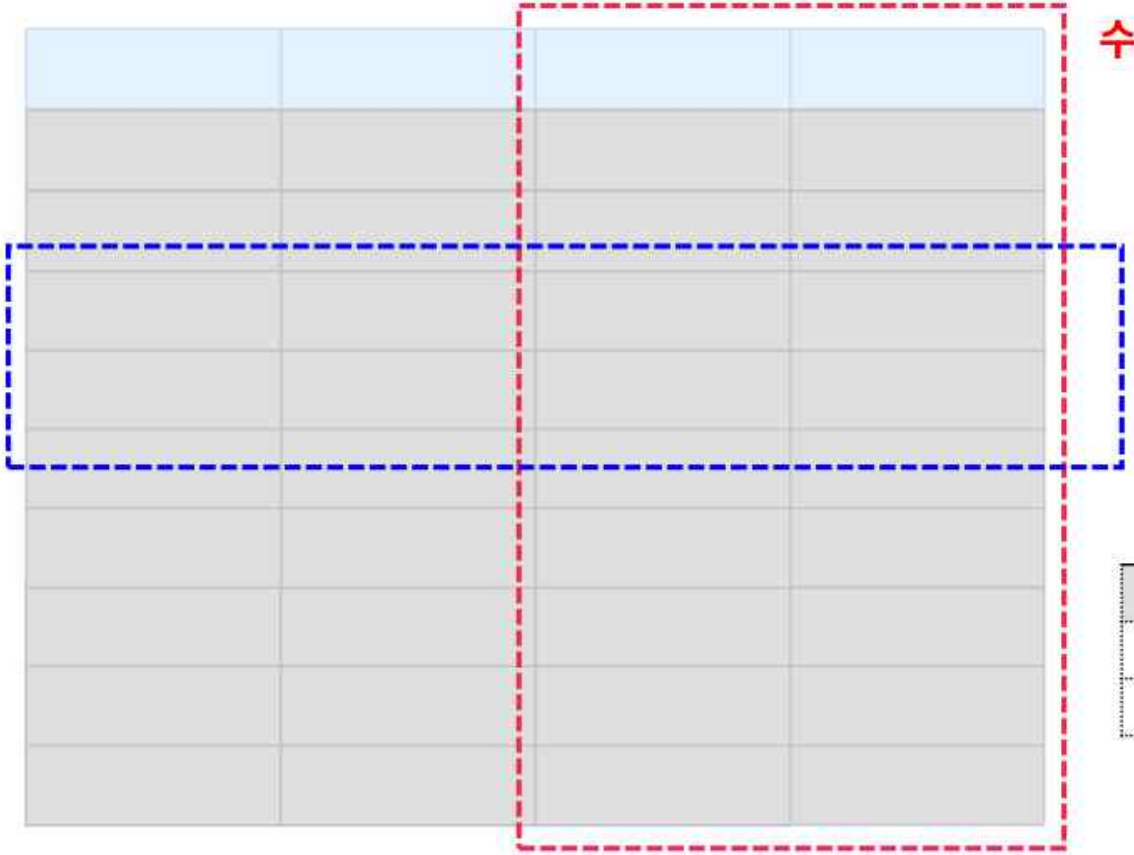


JAVA 웹 개발자 양성과정
DataBase

4. 대량데이터에 따른 성능

➢ 대량 데이터발생에 따른 테이블 분할 개요

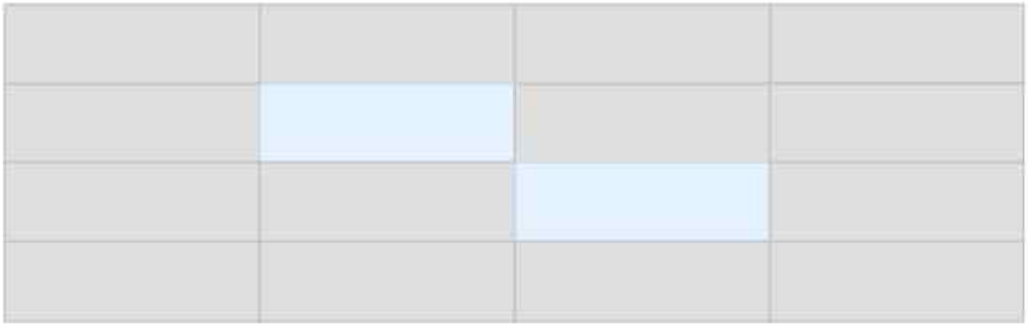
① 대량의 데이터가 존재하는 테이블에 많은 트랜잭션이 발생하여 성능이 저하되는 테이블 구조에 대해 수평/수직 분할 설계를 통해 성능 저하를 예방할 수 있음



분할방색	설명
수직분할	칼럼 단위로 분할하여 I/O를 감소 시킴
수평분할	로우 단위로 분할하여 I/O를 감소 시킴

➢ 대량 데이터발생에 따른 테이블 분할 개요 - 계속

- 테이블의 데이터는 Block 단위로 디스크에 저장된다.



- 칼럼이 많아지게 되면 하나의 로우를 저장 시 물리적인 디스크에 여러 블록에 데이터가 저장될 가능성이 높아짐
- 즉 하나의 행을 읽더라도 여러 개의 블록을 읽어야함
- 자연스레 해당 SQL문의 Block I/O가 많아짐

- 대용량 테이블에서 발생할 수 있는 현상

현상명	설명
로우 체이닝 (Row Chaining)	로우 길이가 너무 길어서 데이터 블록 하나에 데이터가 모두 저장되지 않고 두 개 이상의 블록에 걸쳐 하나의 로우가 저장되어 있는 형태
로우 마이그레이션 (Row Migration)	데이터 블록에서 수정이 발생하면 수정된 데이터를 해당 데이터 블록에서 저장하지 못하고 다른 블록의 빈 공간을 찾아 저장하는 방식

- 로우체이닝과 로우마이그레이션이 발생하여 많은 블록에 데이터가 저장되면 데이터 조회 시 절대적인 Block I/O의 횟수가 많아지게 된다.
- Block I/O의 횟수가 많아지면 Disk I/O를 할 가능성도 높아진다.
- Disk I/O를 하게 되는 경우 성능이 급격히 저하 된다.

➢ 한 테이블에 많은 수의 칼럼을 가지고 있는 경우

도서정보

- # 도서번호
- o 발행기관명
- o 제품번호
- o 출판사제품번호
- o 국제표준번호
- o 제품형태
- o 수량
- o 전자출판유형코드
- o 전차출판포맷코드
- o 전자출판소스파일버전
- o 전자출판소스파일주기
- o 회의기술
- o 회의날짜
- o 회의명
- o 회의장소
- o 대체제품도서번호
- o 대체제품제품번호
- o 대체포맷유형코드
- o 대체제품형태
- o 대체제품수량
- o 공고일
- o 발행일
- o 초판년
- o 판권년
- o 전체높이
- o 전체넓이

```
SELECT
        발행기관명
      , 수량
      , 공고일
      , 발행일
  FROM 도서정보
 WHERE 초판년 = '2002'
;
```

- 칼럼수가 많은 테이블은 SELECT시 Block I/O의 수가 많아짐
- Block I/O가 많아지면 자연스레 Disk I/O의 양이 증가될 가능성이 높아짐
- Disk I/O가 많아지면 성능이 저하됨
- 이럴 경우 **지나치게 많은 칼럼이 존재하는 테이블을 수직분할하면 성능이 향상될 수 있음**

➢ 한 테이블에 많은 수의 칼럼을 가지고 있는 경우 - 수직분할


```
SELECT
        발행기관명
      , 수량
      , 공고일
      , 발행일
  FROM 도서정보
 WHERE 초판년 = '2002'
;
```

- 전자출판유형에 대한 트랜잭션이 독립적으로 발생이 되는 경우가 많아 1:1 관계로 수직분할
- 대체제품에 대한 유형의 트랜잭션이 독립적으로 발생되는 경우가 많이 있어 1:1 관계로 수직분할
- 분리된 테이블은 칼럼의 수가 적으므로 로우마이그레이션과 로우체이닝이 많이 줄어들 수 있다
- 도서정보 테이블 조회시에도 디스크 I/O가 줄어들어 성능이 좋아짐

➢ 대용량 테이블 성능 향상 방안 - 수평분할

▪ RANGE PARTITION 적용

- 요금테이블에 PK가 요금일자+요금번호로 구성되어 있고 데이터건수가 1억2천만 건인 대용량 테이블의 경우
- 하나의 테이블로는 너무 많은 데이터가 존재하므로 인해 성능이 느린 경우에 해당
- 요금의 특성상 항상 **월단위로 데이터 처리를 하는 경우가 많으므로 PK인 요금일자의 년+월을 이용하여 12개의 파티션 테이블을 생성함**
- 데이터보관주기에 따라 테이블에 데이터를 쉽게 지우는 것이 가능하므로(파티션 테이블을 DROP하면 되므로) **데이터보관주기에 다른 테이블관리가 용이**

➢ 대용량 테이블 성능 향상 방안 – 수평분할 - 계속

▪ LIST PARTITION 적용

- 고객 테이블에 데이터가 1억 건이 있는데 하나의 테이블에서 데이터를 처리하기에는 SQL문장의 성능이 저하되어 **지역을 나타내는 사업소코드별로 LIST PARTITION**을 적용
- **LIST PARTITION**은 대용량 데이터를 **특정 값에 따라 분리 저장**할 수는 있으나 RANGE PARTITION과 같이 데이터 보관 주기에 따라 쉽게 삭제하는 기능은 제공될 수 없음

➢ 대용량 테이블 성능 향상 방안 – 수평분할 - 계속

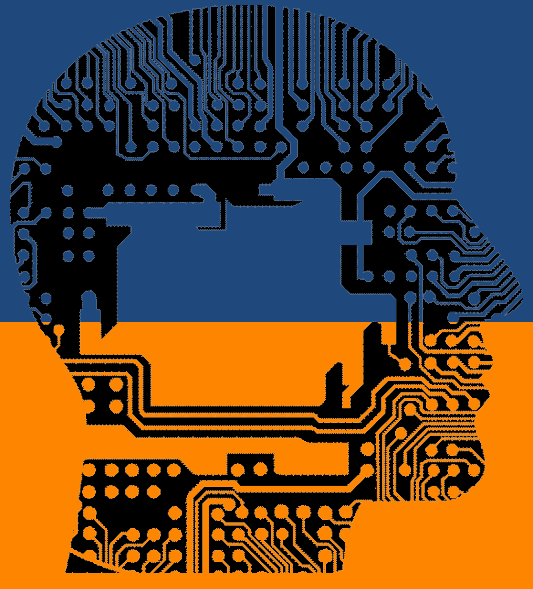
▪ HASH PARTITION 적용

- HASH PARTITION은 지정된 HASH 조건에 따라 해싱 알고리즘이 적용되어 테이블이 분리됨
- 설계자는 테이블에 데이터가 정확하게 어떻게 들어갔는지 알 수 없음
- 성능향상을 위해 사용하며 데이터 보관 주기에 따라 쉽게 삭제하는 기능은 제공될 수 없다.

➢ 테이블에 대한 수평 분할/수직 분할의 절차

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

한국IT진흥부설
정보보호교육학원 아이섹



JAVA 웹 개발자 양성과정

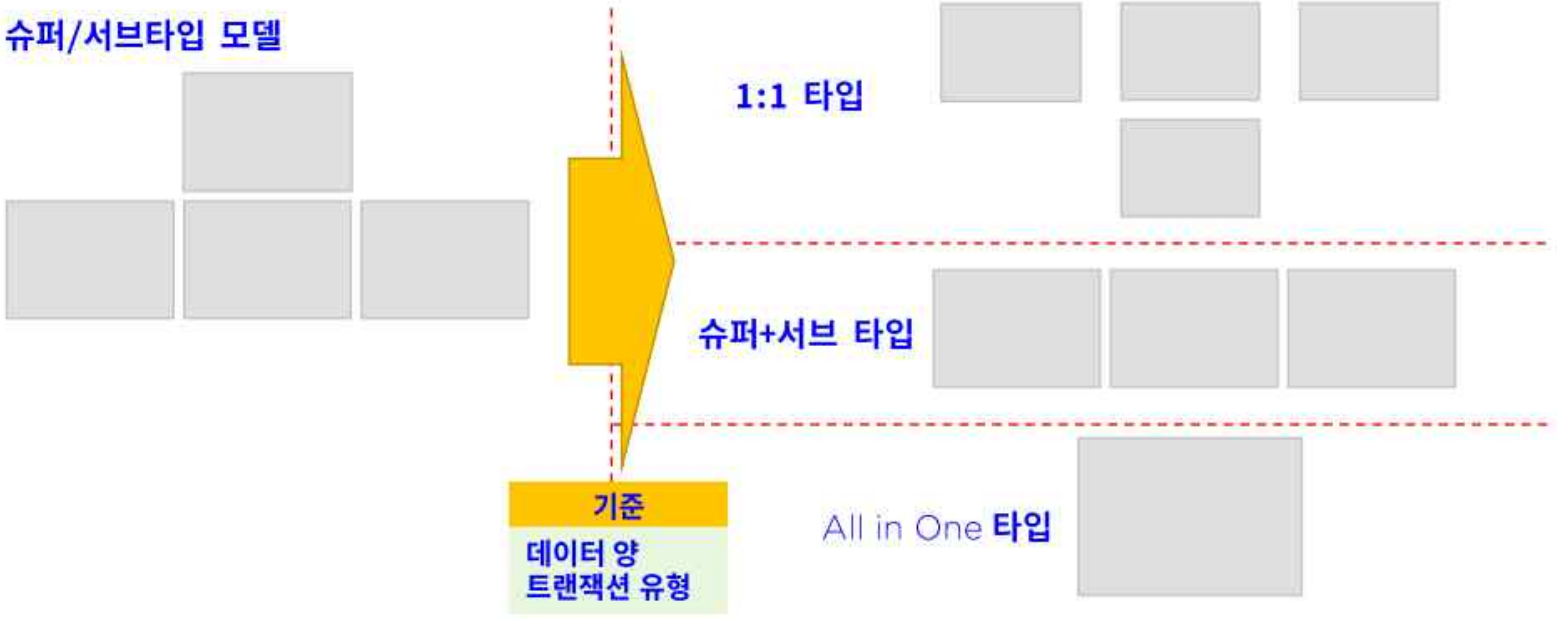
DataBase

5. 데이터베이스 구조와 성능

➢ 슈퍼 타입/서브타입 모델

① 업무를 구성하는 데이터의 특징을 분석하여 공통점/차이점을 고려하여 효과적으로 표현할 수 있음
② 공통의 부분을 슈퍼타입으로 모델링하고 공통으로 부터 상속받아 다른 엔터티와 차이가 있는 속성에 대해서는 별도의 서브엔터티로 구분
③ 업무의 모습을 정확하게 표현하면서 물리적인 데이터 모델로 변환을 할 때 선택의 폭을 넓힐 수 있는 장점이 있음

▪ 슈퍼/서브타입 데이터 모델의 변환



➢ 슈퍼 타입/서브타입 모델

▪ 슈퍼/서브타입 데이터 모델의 변환의 중요성

- 트랜잭션은 항상 일괄로 처리하는데 테이블은 개별로 유지되어 Union연산에 의해 성능이 저하될 수 있다.
- 트랜잭션은 항상 서브타입 개별로 처리하는데 테이블은 하나로 통합되어 있어 불필요하게 많은 양의 데이터가 집약되어 있어 성능이 저하되는 경우가 있다.
- 트랜잭션은 항상 슈퍼+서브 타입을 공통으로 처리하는데 개별로 유지되어 있거나 하나의 테이블로 집약되어 있어 성능이 저하되는 경우가 있다.

➢ 슈퍼/서브 타입 데이터 모델의 변환 기술

▪ 개별로 발생되는 트랜잭션에 대해서는 개별 테이블로 구성

- 슈퍼타입과 서브타입각각에 대해 독립적으로 트랜잭션이 발생이 되면 슈퍼타입에도 꼭 필요한 속성만을 가지게 하고 서브타입에도 꼭 필요한 속성 및 자신이 타입에 맞는 데이터만 가지게 하기 위해서 모두 분리하여 1:1 관계를 갖도록 한다.

➢ 슈퍼/서브 타입 데이터 모델의 변환 기술 - 계속

▪ 슈퍼타입+서브타입에 대해 발생되는 트랜잭션에 대해서는 슈퍼타입+서브타입 테이블로 구성

- 대리인이 10만 건, 매수인 500만 건, 이해관계인 500만 건의 데이터가 존재한다고 가정하고 슈퍼타입과 서브타입이 모두 하나의 테이블로 통합되어 있다고 가정
- 매수인, 이해관계인에 대한 정보는 배제하고 **10만 건뿐인 대리인에 대한 데이터만 처리할 경우 다른 테이블과 같이 데이터가 1010만 건이 저장되어 있는 곳에서 처리해야 하므로 비효율이 발생됨**

➢ 슈퍼/서브 타입 데이터 모델의 변환 기술

▪ 전체를 하나로 묶어 트랜잭션이 발생할 때는 하나의 테이블로 구성

- 대리인 10만 건, 매수인 500만 건, 이해관계인 500만 건의 데이터가 존재한다고 하더라고 데이터를 처리할 때 대리인, 매수인, 이해관계인을 항상 통합하여 처리한다고 하면
- 테이블을 개별로 분리하면 불필요한 조인을 유발하거나 불필요한 UNION ALL과 같은 SQL구문이 작성되어 성능이 저하

▪ 슈퍼/서브타입 데이터 모델 변환 타입 비교

구분	1:1 타입	슈퍼/서브 타입	All in One 타입
특징	개별 테이블 유지	슈퍼+서브 타입 테이블	하나의 테이블
확장성	우수함	보통	나쁨
조인성능	나쁨	나쁨	우수함
I/O 성능	좋음	좋음	나쁨
관리용이성	좋지않음	좋지않음	좋음
트랜잭션 유형에 다른 선택 방법	개별 테이블로 접근이 많은 경우 선택	슈퍼+서브 형식으로 데이터를 처리하는 경우 선택	전체를 일괄적으로 처리하는 경우 선택

➢ PK/FK 칼럼 순서와 성능

▪ PK/FK 칼럼 순서와 성능개요

- 테이블에 발생되는 트랜잭션 조회 패턴에 따라 PK/FK 칼럼의 순서를 조정해야 함
- 성능저하 현상이 많은 부분이 PK가 여러 개의 속성으로 구성된 복합식별자 일 때 PK순서에 대해 별로 고려하지 않고 데이터 모델링을 한 경우에 해당
- 물리적인 데이터 모델링 단계에서는 스스로 생성된 PK순서 이외에 다른 엔터티로부터 상속받아 발생되는 PK순서까지 항상 주의하여 표시

▪ PK가 복합키일 경우 칼럼순서가 성능에 영향을 미치는 이유

- 인덱스 선두 칼럼에 대한 조건이 들어와야 한다. (가능한한 '=' 조건으로)
- 인덱스 선두 칼럼에 대한 조건이 들어오지 않을 경우 인덱스 전체를 읽거나 테이블 전체를 읽게 됨

➢ PK/FK 칼럼 순서와 성능 - 계속

▪ PK 순서의 중요성

- 입시마스터라는 테이블의 PK는 수험번호+년도+학기로 구성되어 있고 전형과목실적 테이블은 입시마스터 테이블에서 상속받은 수험번호+년도+학기에 전형과목코드로 PK가 구성되어 있는 복합식별자 구조의 테이블
- 입시마스터에는 200만 건의 데이터가 있고 학사는 4학기로 구성되어 있고 데이터는 5년간 보관되어 있다. 그러므로 한 학기당 평균 2만 건의 데이터가 있다고 가정

입시마스터
- # 수험번호
- # 년도
- # 학기
- o 대학원구분코드
- o 학위구분코드
- o 등기부상주소

```
SELECT COUNT(수험번호)
  FROM 입시마스터
 WHERE 년도 = '2008'
   AND 학기 = '1'
;
```

❖ 수험번호+년도+학기 중 수험번호에 대한 값이 WHERE절에 들어오지 않으므로 FULL TABLE SCAN이 발생, 200만 건의 데이터를 모두 읽게 되어 성능이 저하

입시마스터_
- # 년도
- # 학기
- # 수험번호
- o 대학원구분코드
- o 학위구분코드
- o 등기부상주소

```
SELECT COUNT(수험번호)
  FROM 입시마스터
 WHERE 년도 = '2008'
   AND 학기 = '1'
;
```

❖ 년도+학기+수험번호 순으로 되어 있으므로 해당 조건이 '=' 조건으로 들어오게 되어 성능이 향상됨

➢ PK/FK 칼럼 순서와 성능 - 계속

▪ PK순서를 잘못 지정하여 성능이 저하된 경우 - 복잡한 오류

- 현금출급기실적의 PK는 거래일자+사무소코드+출급기번호+명세표번호로 되어 있는데 대부분의 SQL문장에서는 조회를 할 때 사무소코드가 '='로 들어오고 거래일자에 대해서는 'BETWEEN' 조회
- SQL은 정상적으로 인덱스를 이용할 수 있지만 인덱스 효율이 떨어져 성능이 저하되는 경우에 해당

현금출급기실적
- # 거래일자
- # 사무소코드
- # 출급기번호
- # 명세표번호
- o 건수
- o 금액

```
SELECT
       건수
     , 금액
  FROM 현금출급기실적
  WHERE 거래일자 BETWEEN '20040701'
                     AND '20040702'
    AND 사무소코드 = '000368'
;
```

❖ 인덱스 스캔은 가능하나 최적화된 인덱스 사용은 되지 않음

현금출급기실적_
- # 사무소코드
- # 거래일자
- # 출급기번호
- # 명세표번호
- o 건수
- o 금액

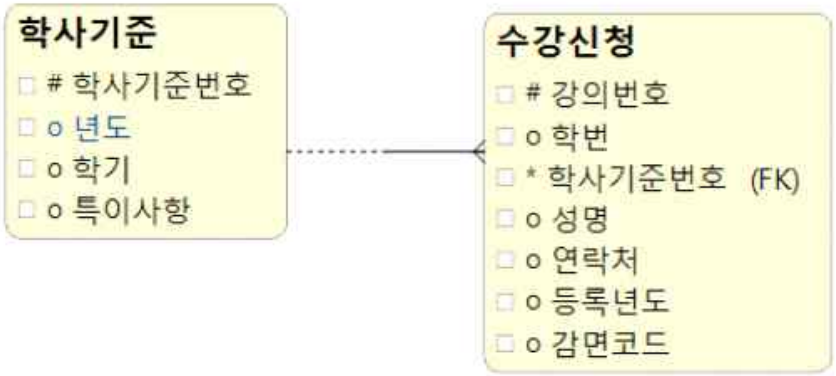
```
SELECT
       건수
     , 금액
  FROM 현금출급기실적
  WHERE 거래일자 BETWEEN '20040701'
                     AND '20040702'
    AND 사무소코드 = '000368'
;
```

❖ 사무소코드+거래일자 순으로 스캔하게 되므로 최적화된 인덱스 스캔이 가능하게 됨

➢ 인덱스 특성을 고려한 PK/FK 데이터베이스 성능향상

▪ 물리적인 테이블에 FK제약이 걸려있지 않을 경우 인덱스 미생성으로 성능저하

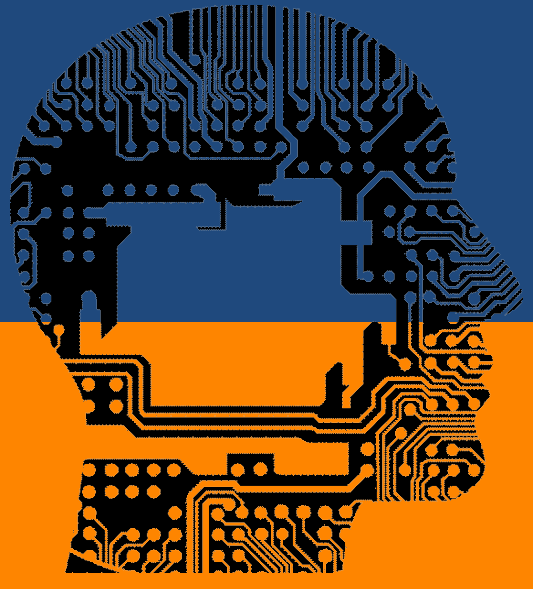
- 물리적인 테이블에 FK를 사용하지 않아도 데이터 모델 관계에 의해 상속받은 FK속성들은 SQL WHERE 절에서 조인으로 이용되는 경우가 많이 있으므로 FK 인덱스를 생성해야 성능이 좋은 경우가 빈번



❖ 수강신청 테이블의 학사기준번호에 인덱스가 존재하지 않는 경우 조인 시 성능이 매우 안좋을 수 있음

❖ 학사기준번호 테이블에 인덱스를 생성하여 성능을 향상 시킬 수 있음

한국IT진흥부설 정보보호교육학원 아이섹



JAVA 웹 개발자 양성과정

DataBase

6. 분산 데이터베이스와 성능

➢ 분산 데이터베이스의 개요

① 여러 곳으로 분산되어 있는 데이터베이스를 하나의 가상 시스템으로 사용할 수 있도록 한 데이터베이스
② 논리적으로 동일한 시스템에 속하지만, 컴퓨터 네트워크를 통해 물리적으로 분산되어 있는 데이터들의 모임.
물리적 Site 분산, 논리적으로 사용자 통합·공유

➢ 분산 데이터베이스의 투명성(Transparency)

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

➢ 분산 데이터베이스의 적용 방법 및 장단점

▪ 분산 데이터베이스 적용방법

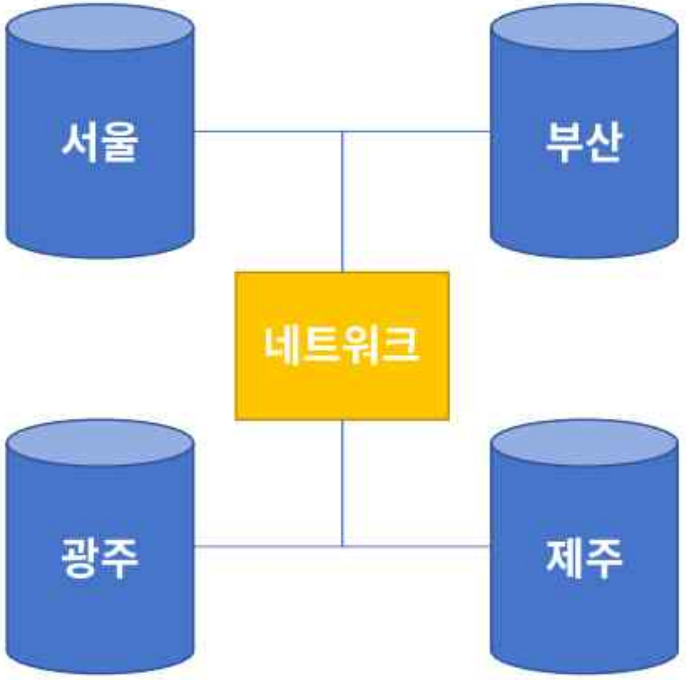
- 분산 환경의 데이터베이스를 성능이 우수하게 현장에서 가치 있게 사용하는 방법은 업무의 흐름을 보고 업무구성에 따른 아키텍처 특징에 따라 데이터베이스를 구성하는 것
- 단순히 분산 환경에서 데이터베이스를 구축하는 것이 목적이 아니라, 업무의 특징에 따라 데이터베이스 분산구조를 선택적으로 설계하는 능력이 필요

➢ 분산 데이터베이스 장단점

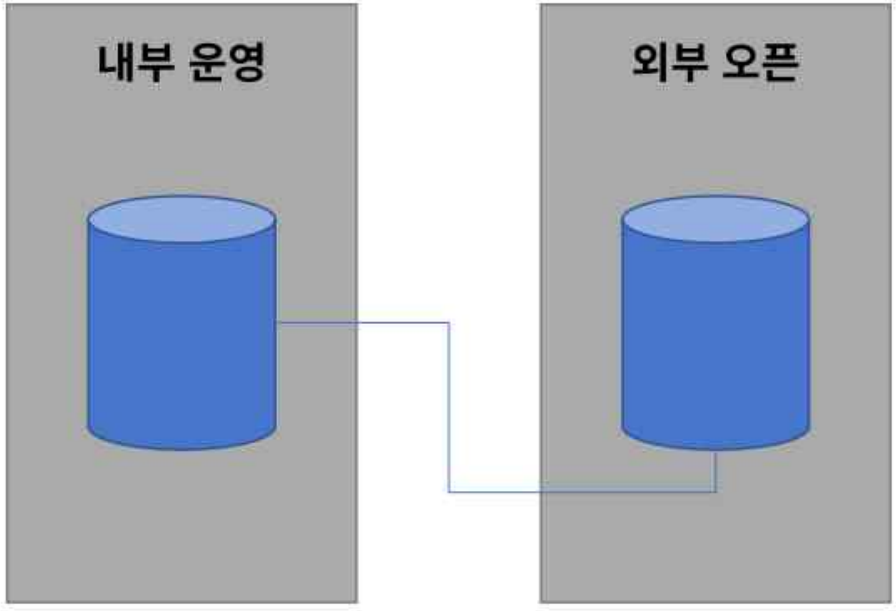
장점	단점
- 지역자치성, 점증적 시스템 용량 확장 - 신뢰성과 가용성 - 효용성과 융통성 - 빠른 응답 속도와 통신비용 절감 - 데이터의 가용성과 신뢰성 증가 - 시스템 규모의 적절한 조절 - 각 지역 사용자의 요구 수용 증대	- 소프트웨어 개발 비용 - 오류의 잠재성 증대 - 처리 비용의 증대 - 설계, 관리의 복잡성과 비용 - 불규칙한 응답 속도 - 통제의 어려움 - 데이터 무결성에 대한 위협

➢ 분산 데이터베이스의 활용 방향성

▪ 위치 중심의 분산 설계(과거 방식)

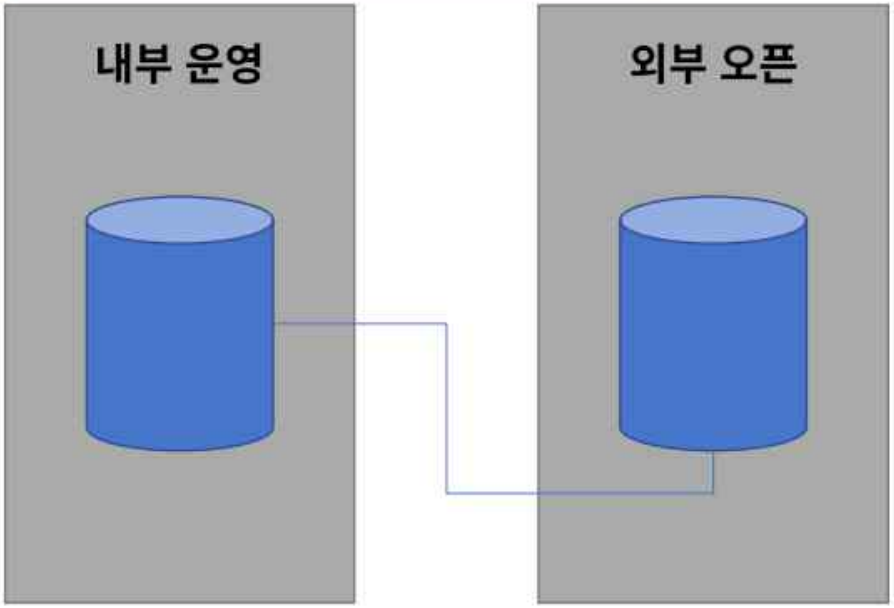


▪ 업무 필요에 의한 분산 설계(현재 방식)



➢ 데이터베이스 분산 구성의 가치

▪ 업무 필요에 의한 분산 설계(현재 방식)



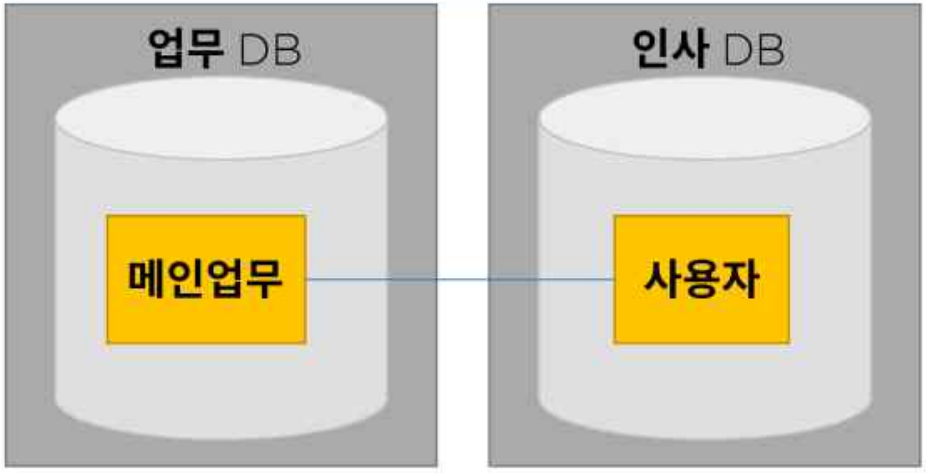
- 통합된 데이터베이스에서 제공할 수 없는 빠른 성능을 제공
- 원거리 또는 다른 서버에 접속하여 처리함으로 인해 발생되는 **네트워크 부하 및 트랜잭션 집중에 따른 성능 저하의 원인**을 분석하여
- 분산 데이터베이스 환경 구축을 함으로써 성능 상 문제 발생 원인을 제거할 수 있음

➢ 분산 데이터베이스의 적용 기법

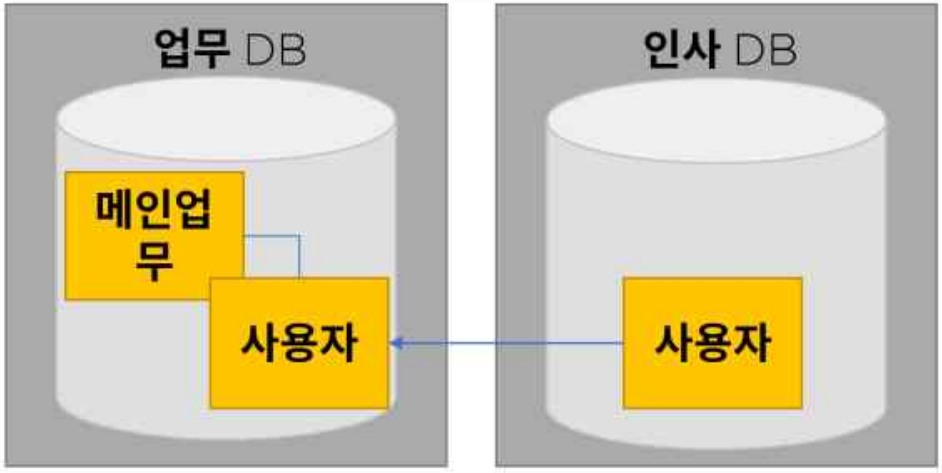
기법	설명
테이블 위치 분산	- 설계된 테이블의 위치를 각각 다르게 위치시키는 것(자재품목은 본사DB, 생상제품은 지사 DB)
테이블 분할(Fragmentation) 분산	- 각각의 테이블을 쪼개어 분산하는 방법 - 수평분할 : 지사(Node)에 따라 테이블을 특정 칼럼의 값을 기준으로 로우(Row)를 분리 - 수직분할 : 지사(Node)에 따라 테이블 칼럼을 기준으로 칼럼(Row)을 분리한다. 로우(Row) 단위로는 분리되지 않는다.
테이블 복제(Replication) 분산	- 동일한 테이블을 다른 지역이나 서버에서 동시에 생성하여 관리하는 유형 - 부분복제 : 통합된 테이블을 한군데(본사)에 가지고 있으면서 각 지사별로는 지사에 해당된 로우(Row)를 가지고 있는 형태 - 광역복제 : 통합된 테이블을 한군데(본사)에 가지고 있으면서 각 지사에도 본사와 동일한 데이터를 모두 가지고 있는 형태
테이블 요약(Summarization) 분산	- 지역간에 또는 서버 간에 데이터가 비슷하지만 서로 다른 유형으로 존재하는 경우 - 분석요약 : 분석요약(Rollup Replication)은 각 지사별로 존재하는 요약정보를 본사에 통합하여 다시 전체에 대해서 요약정보를 산출하는 분산방법 - 통합요약 : 각 지사별로 존재하는 다른 내용의 정보를 본사에 통합하여 다시 전체에 대해서 요약정보를 산출하는 분산방법

➢ 분산 데이터베이스를 적용하여 성능이 향상된 사례

▪ 트랜잭션 마다 개별적으로 원격지 조인



▪ 트랜잭션 마다 내부적으로 조인



- 개인정보를 관리하는 데이터베이스가 인사 데이터베이스일 때 분산이 안된 경우의 각 서버에 독립적으로 테이블이 있을 때 성능이 저하될 수 있음 (원격지 조인으로 인한 성능 저하)
- 인사DB의 사용자 정보를 복제분산하여 업무DB에도 위치 시키면 성능이 향상됨 (로컬 조인으로 성능이 향상됨)

감사합니다
THANK YOU