

제04장

관계형 DBMS

MySQL

학습목표

1. 관계형 DBMS에 대해서 이해할 수 있다.
2. 키(Key)의 종류와 역할에 대해서 이해할 수 있다.
3. 관계에 대해서 이해할 수 있다.
4. 관계형 데이터 모델링에 따라 데이터베이스 오브젝트를 생성할 수 있다.

목차

1. 관계형 DBMS 이해
2. 키(Key)의 종류와 역할
3. 관계
4. 데이터베이스 오브젝트 생성

```

each: function(e, t, n) {
  var r, i = 0,
      o = e.length,
      a = M(e);
  if (n) {
    if (a) {
      for (; o > i; i++)
        if (r = t.apply(e[i], n), r === !1) break;
    } else
      for (i in e)
        if (r = t.apply(e[i], n), r === !1) break;
  } else if (a) {
    for (; o > i; i++)
      if (r = t.call(e[i], e[i]), r === !1) break;
  } else
    for (i in e)
      if (r = t.call(e[i], e[i]), r === !1) break;
  return e;
},
trim: b && !b.call("\uffff\u00a0") ? function(e) {
  return null == e ? "" : b.call(e);
} : function(e) {
  return null == e ? "" : (e + "").replace(C, "");
},
makeArray: function(e, t) {
  var n = t || [];
  return null != e && (M(Object(e)) ? x.merge(n, "string"
),
isArray: function(e, t, n) {
  var r;
  if (t) {
    if (n) return m.call(t, e, n);
    for (r = t.length, r = r ? 0 > n ? Math.max(0, r + n) : n; r--;)
      if (n in t && t[r] === e) return n;
  }
}

```

1. 관계형 DBMS 이해

관계형 DBMS

- RDBMS : Relational DataBase Management System
- 데이터를 구성하기 위해서 테이블을 사용하는 데이터베이스
- 직관적 구조인 테이블을 사용하기 때문에 학습 및 유지 보수가 수월함
- 대용량의 데이터를 손쉽게 저장하고 관리할 수 있음

관계형 DBMS 특징

■ 관계형 DBMS 장점

- 편리한 유지 보수
- 대용량 데이터의 관리 가능
- 데이터의 무결성 보장

■ 관계형 DBMS 단점

- 시스템 자원을 많이 차지
- 구조가 복잡해질수록 성능 최적화가 반드시 필요함

관계형 DBMS 종류

■ ORACLE

- Oracle, 미국 오라클(Oracle)사의 RDBMS
- 현재 가장 널리 사용되는 RDBMS

■ MySQL

- MySQL, 미국 오라클(Oracle)사의 RDBMS
- 오픈 소스(open source) 기반이지만 무료 버전과 상용 버전이 있음

■ MariaDB

- MariaDB, MySQL의 수정 버전으로 오픈 소스(open source) 코드 기반
- 오라클 소유의 불확실한 MySQL 라이선스 문제를 해결하기 위해 만들어짐

■ Microsoft

- MS SQL Server, 미국 마이크로소프트(Microsoft)사의 RDBMS

■

- PostgreSQL, 오픈 소스(open source) 기반의 객체-관계형 데이터베이스 관리 시스템
- 테이블 상속과 같은 객체 지향 언어의 특성을 추가로 가지는 ORDBMS

테이블

■ 테이블 (Table)

- 데이터를 표의 형식으로 체계화하여 행과 열의 집합으로 구성한 것
- RDBMS에서는 테이블을 릴레이션(Relation)이라고도 함

■ 테이블의 구성 요소

- 행 (Row)
 - 실제로 저장된 데이터를 의미함
 - 동의어 : Record, Tuple
- 열 (Column)
 - 행(데이터)을 구성하는 개별적인 속성을 의미함
 - 동의어 : Field, Attribute

상품코드	상품명	상품가격
P0001	태블릿	50
P0002	스피커	20
P0003	USB 허브	10
P0004	노트북	100

Row

Column

테이블 용어

■ 도메인 (Domain)

- 어떤 한 열이 가질 수 있는 값의 범위(집합)

■ 차수 (Degree)

- 어떤 테이블에 있는 열의 갯수

■ 기수 (Cardinality)

- 어떤 테이블에 있는 행의 갯수

Cardinality : 4

아이디	비밀번호	성명	나이	주소	핸드폰	성별
ruby	1***	김철수	22	서울	010-1111-1111	남
pearl	2***	이영숙	23	경기	010-2222-2222	여
sapphire	1***	정승철	51	인천	010-3333-3333	남
diamond	d***	한소정	60	제주	010-4444-4444	여

Degree : 7

성별 도메인 : ('남', '여')

테이블 작성 순서

■ 테이블 작성 순서

- 열 → 행 순으로 작성

1. 열 (Column)

- 각 칼럼의 이름, 데이터 타입, 제약 조건 등을 설정
- 테이블을 만든다는 것은 실제로 칼럼을 만드는 것을 의미함

2. 행 (Row)

- 완성된 각 칼럼에 저장할 데이터를 입력함

```

each: function(e, t, n) {
    var r, i = 0,
        o = e.length,
        a = M(e);
    if (n) {
        if (a) {
            for (; o > i; i++)
                if (r = t.apply(e[i], n), r === !1) break;
        } else
            for (i in e)
                if (r = t.apply(e[i], n), r === !1) break;
    } else if (a) {
        for (; o > i; i++)
            if (r = t.call(e[i], e[i]), r === !1) break;
    } else
        for (i in e)
            if (r = t.call(e[i], e[i]), r === !1) break;
    return e;
},
trim: b && !b.call("\uffff\u00a0") ? function(e) {
    return null == e ? "" : b.call(e);
} : function(e) {
    return null == e ? "" : (e + "").replace(C, "");
},
makeArray: function(e, t) {
    var n = t || [];
    return null != e && (M(Object(e)) ? x.merge(n, "string"
),
isArray: function(e, t, n) {
    var r;
    if (t) {
        if (n) return m.call(t, e, n);
        for (r = t.length, r = r ? 0 > n ? Math.max(0, r + n) : n; n in t && t[n] === e) return n;
    }
}

```

2. 키(Key)의 종류와 역할

키

- 테이블의 데이터(행, Row)를 고유하게(Unique) 식별할 수 있는(Identify) 칼럼 또는 칼럼의 조합을 의미함

- 키의 종류

- 슈퍼키 (Super Key)
- 후보키 (Candidate Key)
- 대체키 (Alternate Key)
- 기본키 (Primary Key)
- 외래키 (Foreign Key)

가장 중요한 키는 기본키와 외래키이다.

슈퍼키

- 테이블의 각 데이터를 식별할 수 있는 칼럼 또는 칼럼의 집합

고객 테이블

고객번호	아이디	고객명	직업	휴대폰번호	주민등록번호
1	aaa	제임스	자영업	010-1234-1234	851122-111111
2	bbb	에밀리	개발자	010-8282-8282	951017-222222
3	ccc	사만다	인턴	010-2580-2580	980608-111111
4	ddd	제임스	개발자	010-5678-5678	771024-222222
5	eee	브라운	공무원	010-9876-9876	771024-222222

- 고객 테이블을 분석하여 슈퍼키가 될 수 있는 칼럼이나 칼럼의 집합을 선정

➤ 판단 기준 : 모든 데이터가 서로 다른 값을 가지고 있는 칼럼인가? 유일성(Unique)을 가지는지 체크함

- 선정된 슈퍼키

- 고객번호
- 아이디
- 휴대폰번호
- 주민등록번호
- 고객명 + 직업

후보키

- 최소한의 칼럼으로 구성된 슈퍼키만 선정할 것을 후보키라고 함

고객 테이블

고객번호	아이디	고객명	직업	휴대폰번호	주민등록번호
1	aaa	제임스	자영업	010-1234-1234	851122-111111
2	bbb	에밀리	개발자	010-8282-8282	951017-222222
3	ccc	사만다	인턴	010-2580-2580	980608-111111
4	ddd	제임스	개발자	010-5678-5678	771024-222222
5	eee	브라운	공무원	010-9876-9876	771024-222222

- 슈퍼키 중에서 가장 적은 수의 칼럼으로 구성된 키를 선정

➤ 판단 기준 : 최소한의 칼럼으로 구성되었는가? 최소성을 가지는지 체크함

- 선정된 후보키

➤ 고객번호

➤ 아이디

➤ 휴대폰번호

➤ 주민등록번호

➤ 고객명 + 직업 (다른 슈퍼키는 단일 칼럼이지만 이 슈퍼키는 2개의 칼럼을 가지고 있어 최소성을 가지지 못함)

기본키

- 후보키 중에서 데이터베이스 관리자(DBA)가 선택한 키를 의미함
- 기본키로 선택한 칼럼은 데이터를 식별하기 위한 식별자의 역할을 수행
- 기본키(Primary Key, 주키)는 개체 무결성을 가짐
 - NOT NULL : 기본키는 NULL 값을 가질 수 없다.
 - UNIQUE : 기본키는 중복 값을 가질 수 없다.

기본키 선정

- 값 또는 의미가 변경될 가능성이 있는 칼럼은 기본키로 부적절함

고객 테이블

고객번호	아이디	고객명	직업	휴대폰번호	주민등록번호
1	aaa	제임스	자영업	010-1234-1234	851122-111111
2	bbb	에밀리	개발자	010-8282-8282	951017-222222
3	ccc	사만다	인턴	010-2580-2580	980608-111111
4	ddd	제임스	개발자	010-5678-5678	771024-222222
5	eee	브라운	공무원	010-9876-9876	771024-222222

- 후보키 중에서 선정된 기본키

- 고객번호 (PK로 선정)
- 아이디 (접속 방식이 이메일과 같은 다른 수단으로 변경될 경우 향후 아이디가 NULL일 가능성 있음)
- 휴대폰번호 (휴대폰이 없는 고객이 있음)
- 주민등록번호 (개인정보보호법에 의해서 수집할 수 없는 데이터)

기본키의 종류

■ 자연키

- Natural Key
- 비즈니스 모델을 통해서 추출한 키
- 이미 존재하고 있던 데이터 중 하나
- 아이디, 이메일 등

■ 인공키

- Artificial Key
- 실제로 존재하지 않는 데이터를 인위적으로 추가한 키
- 자동으로 증가하는 순번을 주로 사용함 (Auto Increment)
- 회원번호, 고객번호 등

대체키

■ 후보키 중에서 기본키를 제외한 나머지를 의미함

고객 테이블

고객번호	아이디	고객명	직업	휴대폰번호	주민등록번호
1	aaa	제임스	자영업	010-1234-1234	851122-111111
2	bbb	에밀리	개발자	010-8282-8282	951017-222222
3	ccc	사만다	인턴	010-2580-2580	980608-111111
4	ddd	제임스	개발자	010-5678-5678	771024-222222
5	eee	브라운	공무원	010-9876-9876	771024-222222

■ 후보키 중에서 선정된 대체키

- 아이디
- 휴대폰번호
- 주민등록번호

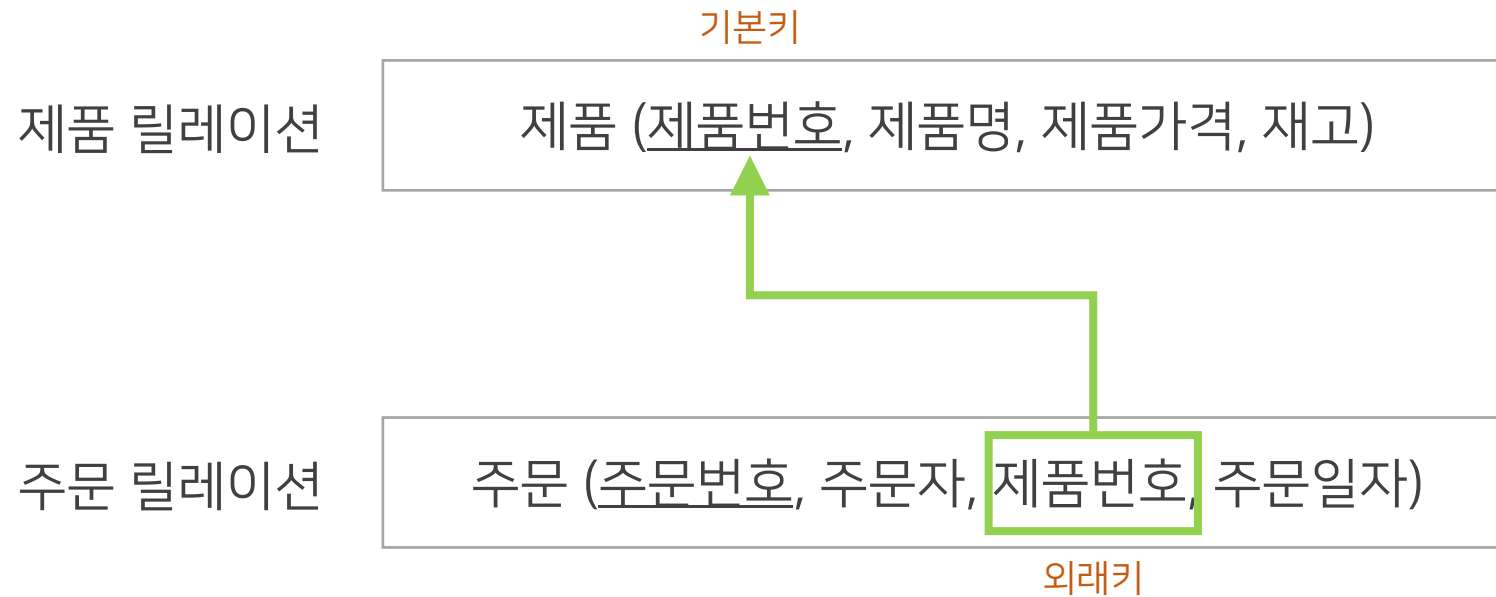
대체키는 보조키라고도 한다.

외래키

- 다른 테이블과 관계(Relationship)를 맺기 위해서 설정하는 키
- 외래키는 다른 테이블의 기본키(Primary Key) 또는 중복 없는 칼럼(Unique Column)과 관계를 맺을 수 있음
- 외래키는 Null 값이나 중복 값을 가질 수 있음
- 외래키는 참조 무결성을 가짐
 - 관계를 맺은 테이블의 키가 가진 도메인을 벗어난 값을 가질 수 없다.

외래키와 기본키의 표시 형식

- 외래키가 기본키를 가리키는 방식으로 표시함



부모 테이블과 자식 테이블

■ 부모 테이블과 자식 테이블

- 부모 테이블 : 외래키를 가진 자식 테이블이 참조하는 테이블
- 자식 테이블 : 외래키를 가진 테이블

부모 테이블

제품번호	제품명	제품가격	재고
1	새우깡	1000	15
2	감자깡	1000	10
3	양파링	2000	20
4	맛동산	3000	15

기본키(PK)

주문번호	주문자	제품번호	주문일자
1000	ab***	1	21/03/03
1001	gt***	4	21/03/03
1002	le***	4	21/03/05
1003	ko***	3	21/03/05
1004	db***	3	21/03/06
1005	as***	1	21/03/06
1006	zw***	2	21/03/07

자식 테이블

외래키(FK)

참조 무결성

- 외래키로 지정된 칼럼은 오직 참조 중인 부모 테이블의 데이터만 가질 수 있음

외래키로 설정된 제품번호는 오직 1, 2, 3, 4 중 하나의 값만 가질 수 있음



참조 무결성 위배 상황

- 부모 테이블의 데이터가 삭제되거나 수정되는 경우
자식 테이블의 데이터는 참조 무결성을 위배할 수 있음

제품번호 4 인 제품을 삭제한다면,

기존에 제품번호 4인 제품의 주문내역은 참조 무결성을 위배하게 됨

부모 테이블

제품번호	제품명	제품가격	재고
1	새우깡	1000	15
2	감자깡	1000	10
3	양파링	2000	20
4	맛동산	3000	15

기본키(PK)

자식 테이블

주문번호	주문자	제품번호	주문일자
1000	ab***	1	21/03/03
1001	gt***	4	21/03/03
1002	le***	4	21/03/05
1003	ko***	3	21/03/05
1004	db***	3	21/03/06
1005	as***	1	21/03/06
1006	zw***	2	21/03/07

외래키(FK)

RESTRICT

- 부모 테이블의 데이터를 삭제/변경할 때 이를 참조하는 자식 테이블의 데이터가 존재하면 데이터 삭제/변경을 취소함

제품번호 4 인 제품을 삭제하려고 시도하면, 기존에 제품번호 4인 제품의 주문내역이 존재하므로 삭제가 취소됨



CASCADE

- 부모 테이블의 데이터가 삭제/변경되면 이를 참조하는 자식 테이블의 데이터를 함께 삭제/변경함

제품번호 4 인 제품을 삭제한다면,

기존에 제품번호 4인 제품의 주문내역도 함께 삭제함

부모 테이블

제품번호	제품명	제품가격	재고
1	새우깡	1000	15
2	감자깡	1000	10
3	양파링	2000	20
4	맛동산	3000	15

기본키(PK)

자식 테이블

주문번호	주문자	제품번호	주문일자
1000	ab***	1	21/03/03
1001	gt***	4	21/03/03
1002	le***	4	21/03/05
1003	ko***	3	21/03/05
1004	db***	3	21/03/06
1005	as***	1	21/03/06
1006	zw***	2	21/03/07

외래키(FK)

SET NULL

- 부모 테이블의 데이터 삭제/변경으로 인해 더 이상 자식 테이블이 부모 테이블을 참조하지 못하는 경우 해당 칼럼 값을 Null 값으로 바꿈

제품번호 4 인 제품을 삭제한다면,

기존에 제품번호 4인 제품의 전체 주문내역은 남기고 제품번호만 삭제함

부모 테이블

제품번호	제품명	제품가격	재고
1	새우깡	1000	15
2	감자깡	1000	10
3	양파링	2000	20
4	맛동산	3000	15

기본키(PK)

자식 테이블

주문번호	주문자	제품번호	주문일자
1000	ab***	1	21/03/03
1001	gt***	Null	21/03/03
1002	le***	Null	21/03/05
1003	ko***	3	21/03/05
1004	db***	3	21/03/06
1005	as***	1	21/03/06
1006	zw***	2	21/03/07

외래키(FK)

```

each: function(e, t, n) {
    var r, i = 0,
        o = e.length,
        a = M(e);
    if (n) {
        if (a) {
            for (; o > i; i++)
                if (r = t.apply(e[i], n), r === !1) break;
        } else
            for (i in e)
                if (r = t.apply(e[i], n), r === !1) break;
    } else if (a) {
        for (; o > i; i++)
            if (r = t.call(e[i], e[i]), r === !1) break;
    } else
        for (i in e)
            if (r = t.call(e[i], e[i]), r === !1) break;
    return e;
},
trim: b && !b.call("\uffff\u00a0") ? function(e) {
    return null == e ? "" : b.call(e);
} : function(e) {
    return null == e ? "" : (e + "").replace(C, "");
},
makeArray: function(e, t) {
    var n = t || [];
    return null != e && (M(Object(e)) ? x.merge(n, "string"
),
isArray: function(e, t, n) {
    var r;
    if (t) {
        if (n) return m.call(t, e, n);
        for (r = t.length, r = r ? 0 > n ? Math.max(0, r + n) : n : 0; r < n; r++)
            if (n in t && t[r] === e) return r;
    }
}

```

3. 관계

관계

■ 엔티티 (Entity)

- 정의할 수 있는 사물이나 개념을 의미함
- 유형 또는 무형의 정보도 가능함
- 관계형 데이터베이스에서는 테이블이 엔티티로 표현됨

■ 관계 (Relationship)

- 엔티티 간 논리적인 연관성을 의미함
- 즉 테이블 간 상호 연관성을 의미함

관계 표기법

- 관계 표기법은 "관계명", "관계차수", "관계선택사양"으로 구성됨

- 관계명

- 엔티티가 관계에 참여하는 형태를 지칭함
- 현재형으로 표현하며 명확한 동사를 사용함 ('강의한다', '수강신청한다' ...)

- 관계차수

- 두 개의 엔티티 간 관계에서 참여하는 데이터의 수를 표현함
- 일대일(1:1), 일대다(1:M), 다대다(M:N) 관계가 존재함

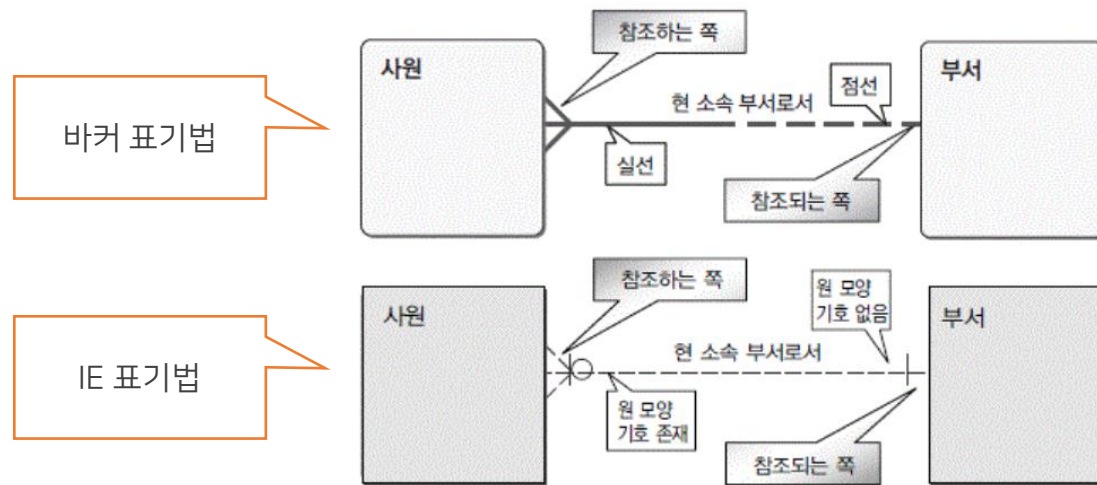
- 관계선택사양

- 참여하는 엔티티가 항상 참여하는지 여부를 나타냄
- 필수참여, 선택참여로 나뉨

일대다(1:M) 관계

■ 일대다(1:M) 관계

- 한 쪽의 엔티티가 관계를 맺은 다른 엔티티 쪽의 여러 객체를 가질 수 있다.
- 현실 세계에서 가장 흔한 관계이다.
- 예) 어머니가 자식 3명을 낳으셨다. 어머니(1) : 자식(3)



M : 1

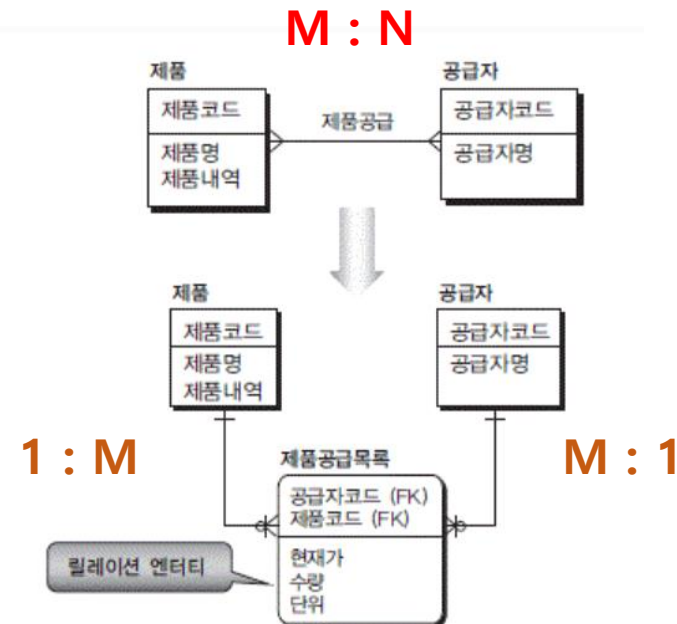
일대다(1:M) 관계 예시



다대다(M:N) 관계

■ 다대다(M:N) 관계

- 관계를 가진 양쪽 모두에게 1:M 관계가 존재할 때 나타난다.
- 현실 세계에서 의외로 빈번하게 발생하는 관계이다.
- 예 : 학생들이 수강신청을 한다. 학생도 여러 명(M) : 과목도 여러 과목(N)
- 1:M 관계 2개를 이용해서 구현한다.



다대다(M:N) 관계 예시



학생 테이블

수강신청 테이블

과목 테이블

```

each: function(e, t, n) {
  var r, i = 0,
      o = e.length,
      a = M(e);
  if (n) {
    if (a) {
      for (; o > i; i++)
        if (r = t.apply(e[i], n), r === !1) break;
    } else
      for (i in e)
        if (r = t.apply(e[i], n), r === !1) break;
  } else if (a) {
    for (; o > i; i++)
      if (r = t.call(e[i], e[i]), r === !1) break;
  } else
    for (i in e)
      if (r = t.call(e[i], e[i]), r === !1) break;
  return e;
},
trim: b && !b.call("\uffff\u00a0") ? function(e) {
  return null == e ? "" : b.call(e);
} : function(e) {
  return null == e ? "" : (e + "").replace(C, "");
},
makeArray: function(e, t) {
  var n = t || [];
  return null != e && (M(Object(e)) ? x.merge(n, "string"
),
isArray: function(e, t, n) {
  var r;
  if (t) {
    if (n) return m.call(t, e, n);
    for (r = t.length, r = r ? 0 > n ? Math.max(0, r + n) : n : 0; r < n; r++)
      if (n in t && t[r] === e) return r;
  }
}

```

4. 데이터베이스 오브젝트 생성

데이터베이스 생성 계획

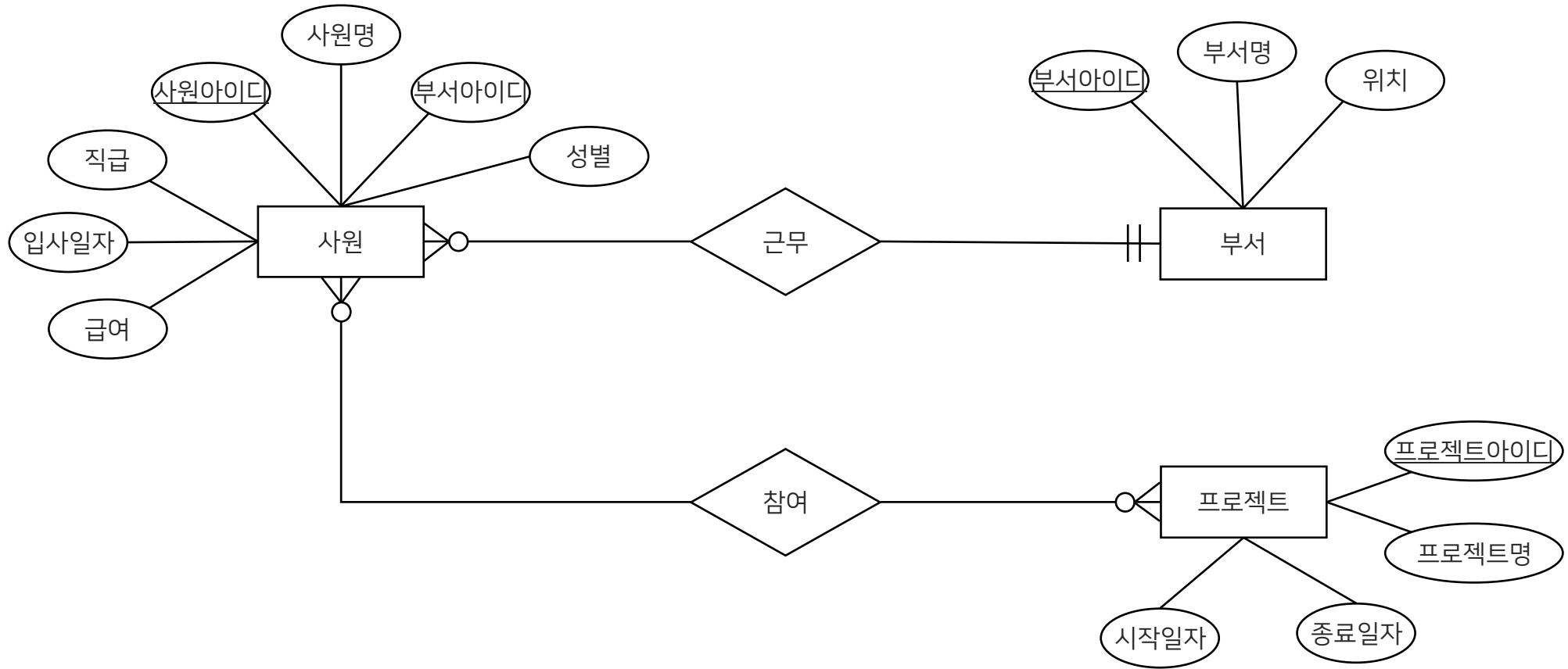


개념적 모델링

- 데이터 간의 관계를 정의
- 관계 표현을 위해서 ERD 다이어그램을 사용 (Entity Relation Diagram)
- 피터 첸 표기법(Peter Chen Notation)에 따라 ERD 다이어그램을 구성함

기호	의미
	개체 (Entity)
	속성 (Attribute)
	주식별자 (Primary Identifier)
	관계 (Relation)
	연결선

ERD



ERD 분석

사원 사원아이디, 부서아이디, 사원명, 직급, 성별, 입사일자, 급여

부서 부서아이디, 부서명, 위치

프로젝트 프로젝트아이디, 프로젝트명, 시작일자, 종료일자


■ 설계 분석

- 사원 엔티티의 사원아이디는 **주식별자(primary identifier)**이다.
- 사원 엔티티의 부서아이디는 부서 엔티티의 부서아이디를 참조하는 **외래식별자(foreign identifier)**이다.
- 부서 엔티티의 부서아이디는 **주식별자(primary identifier)**이다.
- 프로젝트 엔티티의 프로젝트아이디는 **주식별자(primary identifier)**이다.
- 사원 엔티티와 프로젝트 엔티티는 다대다 관계이므로 프로젝트에 참여한 사원 명단을 별도로 관리하는 프로젝트사원 엔티티가 추가로 필요하다.

ERD 모델링




■ 논리적 모델링

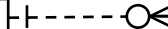
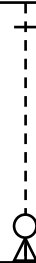
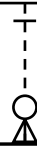
- 테이블 / 컬럼명을 한글로 작성

부서			
	부서아이디	INT	NOT NULL
	부서명	VARCHAR(30)	NULL
	위치	VARCHAR(50)	NULL

프로젝트			
	프로젝트아이디	INT	NOT NULL
	프로젝트명	VARCHAR(30)	NULL
	시작일자	DATE	NULL
	종료일자	DATE	NULL

사원			
	사원아이디	INT	NOT NULL
	부서아이디	INT	NULL
	사원명	VARCHAR(15)	NULL
	직급	CHAR(10)	NULL
	성별	CHAR(1)	NULL
	입사일자	DATE	NULL
	급여	INT	NULL


프로젝트사원			
	등록아이디	INT	NOT NULL
	사원아이디	INT	NULL
	프로젝트아이디	INT	NULL
	진행상태	INT	NOT NULL






ERD 모델링




■ 물리적 모델링

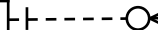
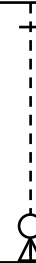
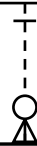
- 테이블 / 컬럼명을 영문으로 작성

tbl_department			
	dept_id	INT	NOT NULL
	dept_name	VARCHAR(30)	NULL
	location	VARCHAR(50)	NULL

tbl_project			
	proj_id	INT	NOT NULL
	proj_name	VARCHAR(30)	NULL
	begin_date	DATE	NULL
	end_date	DATE	NULL

tbl_employee			
	emp_id	INT	NOT NULL
	dept_id	INT	NULL
	emp_name	VARCHAR(15)	NULL
	position	CHAR(10)	NULL
	gender	CHAR(1)	NULL
	hire_date	DATE	NULL
	salary	INT	NULL

tbl_proj_emp			
	reg_id	INT	NOT NULL
	emp_id	INT	NULL
	proj_id	INT	NULL
	state	INT	NOT NULL



데이터베이스 오브젝트 설계

■ 테이블 정의서 : 부서

객체	테이블		명	tbl_department	작성일			작성자	
데이터베이스			db_company		설명		부서 테이블		
번호	논리칼럼명	물리칼럼명	데이터타입	크기	NN	UQ	PK	FK	COMMENT
1	부서아이디	dept_id	INT		Y		Y		
2	부서명	dept_name	VARCHAR	30					
3	위치	location	VARCHAR	50					
4									
5									
6									
7									

데이터베이스 오브젝트 설계

■ 테이블 정의서 : 사원

객체	테이블		명	tbl_employee	작성일			작성자	
데이터베이스			db_company		설명		사원 테이블		
번호	논리칼럼명	물리칼럼명	데이터타입	크기	NN	UQ	PK	FK	COMMENT
1	사원아이디	emp_id	INT		Y		Y		
2	부서아이디	dept_id	INT					Y	tbl_department(dept_id)
3	사원명	emp_name	VARCHAR	15					
4	직급	position	CHAR	10					
5	성별	gender	CHAR	1					
6	입사일자	hire_date	DATE						
7	급여	salary	INT						

데이터베이스 오브젝트 설계

■ 테이블 정의서 : 프로젝트

객체	테이블		명	tbl_project	작성일			작성자	
데이터베이스			db_company		설명		프로젝트 테이블		
번호	논리칼럼명	물리칼럼명	데이터타입	크기	NN	UQ	PK	FK	COMMENT
1	프로젝트아이디	proj_id	INT		Y		Y		
2	프로젝트명	proj_name	VARCHAR	30					
3	시작일자	begin_date	DATE						
4	종료일자	end_date	DATE						
5									
6									
7									

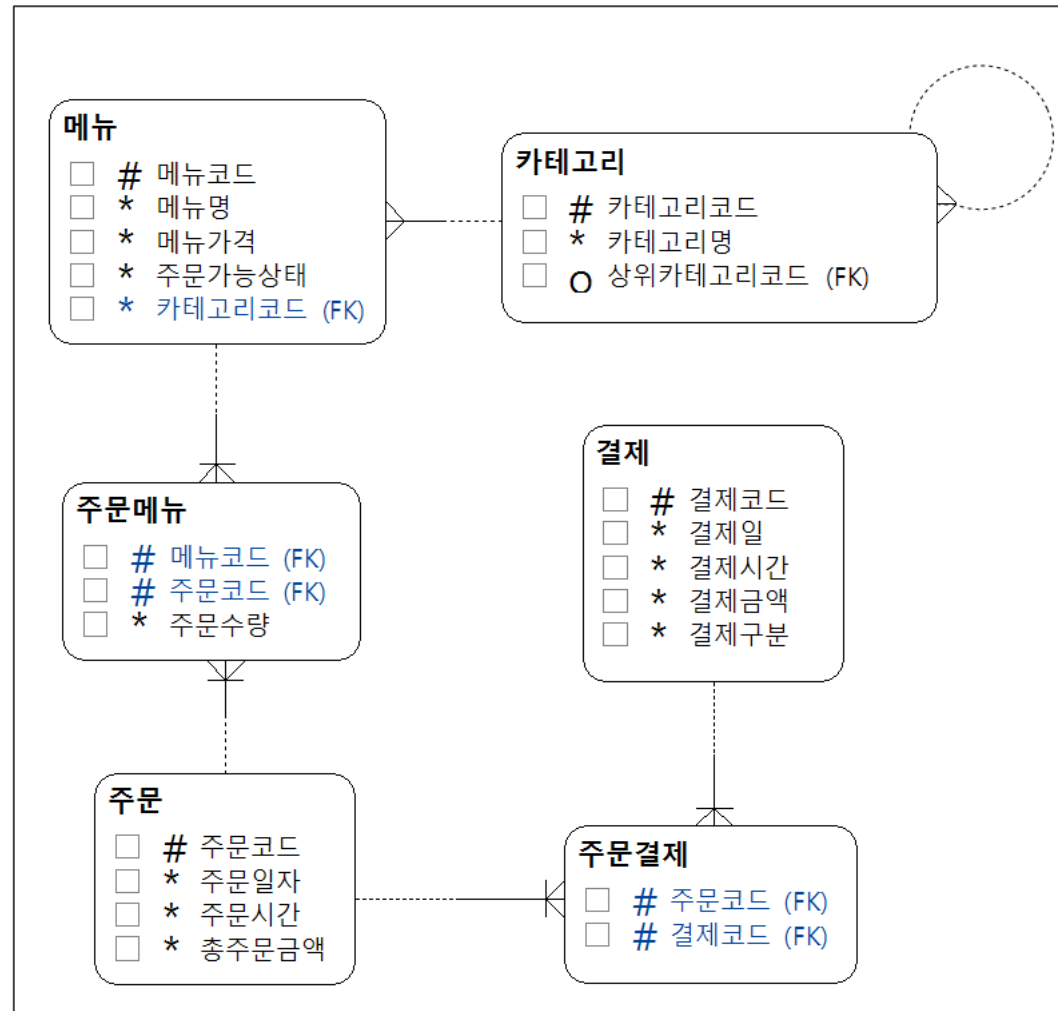
데이터베이스 오브젝트 설계

■ 테이블 정의서 : 프로젝트사원

객체	테이블		명	tbl_proj_emp	작성일			작성자	
데이터베이스			db_company		설명		프로젝트사원 테이블		
번호	논리칼럼명	물리칼럼명	데이터타입	크기	NN	UQ	PK	FK	COMMENT
1	등록아이디	reg_id	INT		Y		Y		
2	사원아이디	emp_id	INT					Y	tbl_employee(emp_id)
3	프로젝트아이디	proj_id	INT					Y	tbl_project(proj_id)
4	진행상태	state	INT		Y				
5									
6									
7									

실습. db_menu

■ ERD 모델링 - 논리 모델링



실습. db_menu

■ ERD 모델링 - 물리 모델링

