



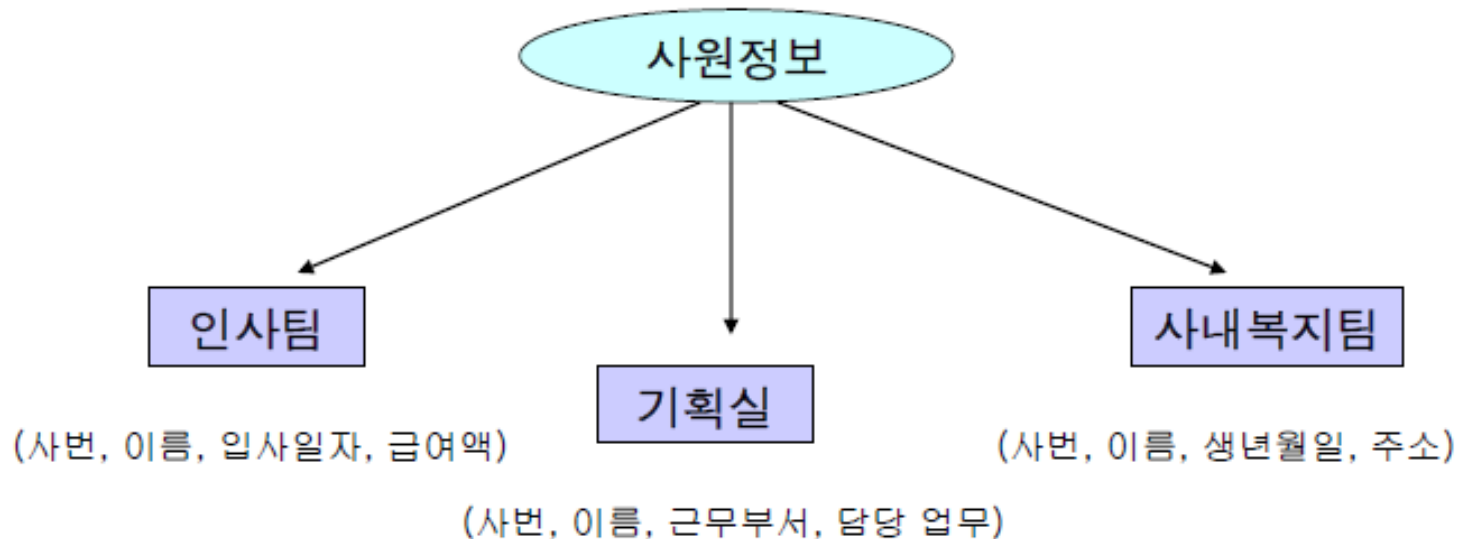
9장. 뷰, 트랜잭션

- ☐ 뷰
- ☐ 트랜잭션

9.1 뷰(view)

□ 뷰의 필요성

- 하나의 테이블, 혹은 여러 테이블에 대하여 특정 사용자나 조직의 관점에서 데이터를 바라볼 수 있도록 해주는 수단



9.1 뷰(view)

□ 뷰의 필요성

EMP

empid	ename	dept	hire_date	birthday	Address	job	salary
1001	홍성길	영업부	2001.2.1	1985.10.12	서울 대림동	특수영업	350
1002	곽희준	영업부	1999.1.1	1984.9.10	안양 용봉동	영업관리	400
1003	김동준	생산부	2000.9.1	1986.5.16	부산 대하동	품질관리	300
1004	성재규	인사부	1997.2.1	1982.4.10	대구 달성동	급여	450
1005	박성범	구매부	2000.2.1	1986.12.4	광주 금남동	수입자재	320

<그림 9.1> 전체 조직 관점에서의 사원 테이블

9.1 뷰(view)

VIEW_EMP1

empid	ename	hire_date	salary
1001	홍성길	2001.2.1	350
1002	곽희준	1999.1.1	400
1003	김동준	2000.9.1	300
1004	성재규	1997.2.1	450
1005	박성범	2000.2.1	320

VIEW_EMP2

empid	ename	dept	job
1001	홍성길	영업부	특수영업
1002	곽희준	영업부	영업관리
1003	김동준	생산부	품질관리
1004	성재규	인사부	급여
1005	박성범	구매부	수입자재

VIEW_EMP3

empid	ename	birthday	Address
1001	홍성길	1985.10.12	서울 대림동
1002	곽희준	1984.9.10	안양 용봉동
1003	김동준	1986.5.16	부산 대하동
1004	성재규	1982.4.10	대구 달성동
1005	박성범	1986.12.4	광주 금남동

<그림 9.2> 사원 테이블에 대한 세가지 뷰

9.1 뷰(view)


□ 뷰의 정의

```
CREATE VIEW high_salary  
  AS SELECT empno, ename, deptno, sal  
     FROM   emp  
     WHERE  sal >= 2500;
```

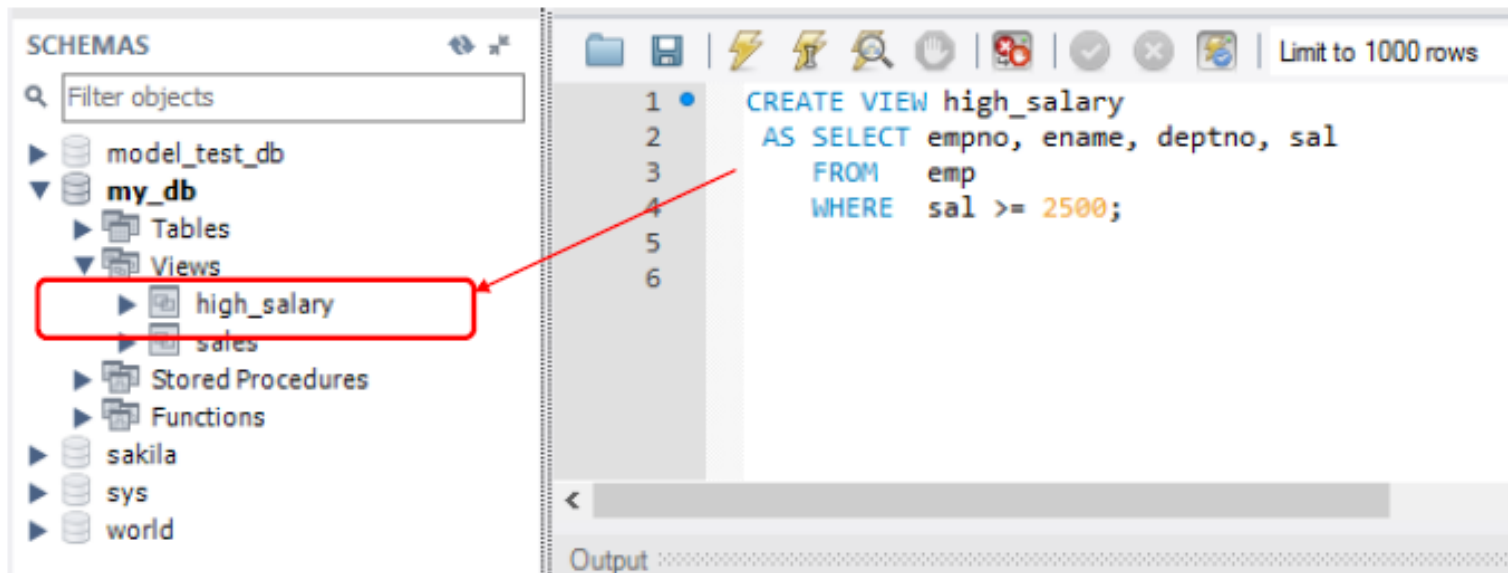
- 뷰에 대한 질의는 뷰가 정의된 기본 테이블에 대한 질의로 바뀌어 실행된다

```
SELECT empno, ename, salary  
FROM   high_salary  
WHERE  dept = '영업부';
```

```
SELECT empid, ename, salary  
FROM   emp  
WHERE  salary >= 350  
AND    dept = '영업부';
```



9.1 뷰(view)



뷰가 만들어지면 사용자 입장에서는 일반 테이블과 동일하게 느껴진다

9.1 뷰(view)

```
SELECT empno, ename, sal  
FROM   high_salary  
WHERE  deptno =10;
```

```
SELECT ename, dname, sal  
FROM   high_salary h, dept d  
WHERE  h.deptno = d.deptno;
```

```
SELECT empno, job, ename, sal  
FROM   high_salary  
WHERE  deptno =10;
```

=> 에러 발생. 이유는?

9.1 뷰(view)

□ 뷰를 사용하는 경우

- <그림 9.2>의 경우와 같이 하나의 테이블에 대하여 여러 부서에서 서로 다른 관점으로 보기를 원할 때
- 테이블에 급여와 같이 일반 사용자에게는 감추어야 할 컬럼이 있을 때 그것을 제외하고 뷰를 만들어 제공함으로써 보안을 유지.
- 자주 사용하는 복잡한 질의문을 미리 뷰로 정의하여 두고 간편하게 쓰고자 할 때

- 뷰는 물리적인 데이터를 갖지 않는다
- 뷰에 대한 갱신 연산은 경우에 따라 실행될수도 있고 실행되지 않을수도 있다

[과제 1]

(1) emp 와 dept 테이블로 부터 다음과 같은 조건을 만족하는 뷰를 생성하시오

뷰이름	empd
컬럼	사원번호, 사원이름, 입사일자, 급여, 직책, 부서명, 부서위치(근무지)

(2) 생성된 뷰를 이용하여 다음의 질의에 대한 sql 문을 작성하시오

- ① 모든 사원의 이름, 부서명을 보이시오
- ② 급여가 2500 보다 많은 사원의 사원번호, 이름, 급여, 부서위치를 보이시오
- ③ 부서위치가 BOSTON 인 사원의 이름과 입사일자를 보이시오
- ④ SMITH 사원의 부서위치를 보이시오

[과제 1]

(3) emp 테이블에서 BOSTON 에 근무하는 직원들만 추려서 emp_boston 뷰를 생성하시오. emp_boston 의 내용을 보이시오.

(4) 다음과 같은 구조를 갖는 뷰 v_emp 를 emp 와 dept 테이블로 부터 생성하고 내용을 보이시오 (컬럼 이름을 표와 동일하게 해야함)

dept_name	tot_emp	tot_sal	avg_sal

부서명

부서의
직원수

부서사원의
급여합계

부서사원의
평균급여

9.2 Transaction

□ 개념

- ATM, 데이터베이스 등의 시스템에서 사용되는 쪼갤 수 없다는 업무처리의 단위
- 여기서 쪼갤 수 없다는 말의 의미는 실제로 쪼갤 수 없다가보다는 만일 쪼개질 경우 시스템에 심각한 오류를 초래할 수 있다는 것

계좌이체의 예

```
update account set balance = balance - 5000  
Where userid = 333377 ;
```

```
update account set balance = balance + 5000  
Where userid = 111122 ;
```

9.2 Transaction

```
start transaction ;
```

```
update account set balance = balance - 5000  
Where userid = 333377 ;
```

```
update account set balance = balance + 5000  
Where userid = 111122 ;
```

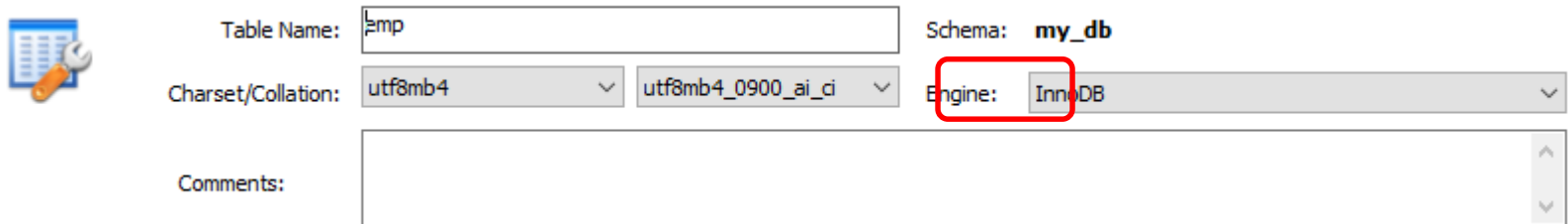
```
commit ;
```

* commit, rollback

9.2 Transaction

❑ MySQL 에서 transaction

- InnoDB 파일시스템을 사용해야 한다.



The image shows the MySQL Workbench Table Creation Wizard. The 'Table Name' is 'emp', 'Schema' is 'my_db', 'Charset/Collation' is 'utf8mb4' and 'utf8mb4_0900_ai_ci', and 'Engine' is 'InnoDB' (highlighted with a red box). There is a 'Comments' text area at the bottom.

- 환경변수중 autocommit 을 off 시킨다.

```
SHOW VARIABLES LIKE 'autocommit';  
SET autocommit = OFF;
```

Result Grid			Filter Rows:	
	Variable_name	Value		
▶	autocommit	ON		



Result Grid			Filter Rows:	
	Variable_name	Value		
▶	autocommit	OFF		

9.2 Transaction

트랜잭션

```
START TRANSACTION;  
delete from emp where empno = 7369;  
select * from emp ;  
rollback;  
select * from emp ;
```

트랜잭션

```
START TRANSACTION;  
delete from emp where empno = 7369;  
select * from emp ;  
commit;  
select * from emp ;
```

9.2 Transaction

- ❑ 다시 autocommit 을 on 으로 한다.

```
SET autocommit = ON;  
SHOW VARIABLES LIKE 'autocommit';
```

Result Grid	Filter Rows:
Variable_name	Value
▶ autocommit	ON