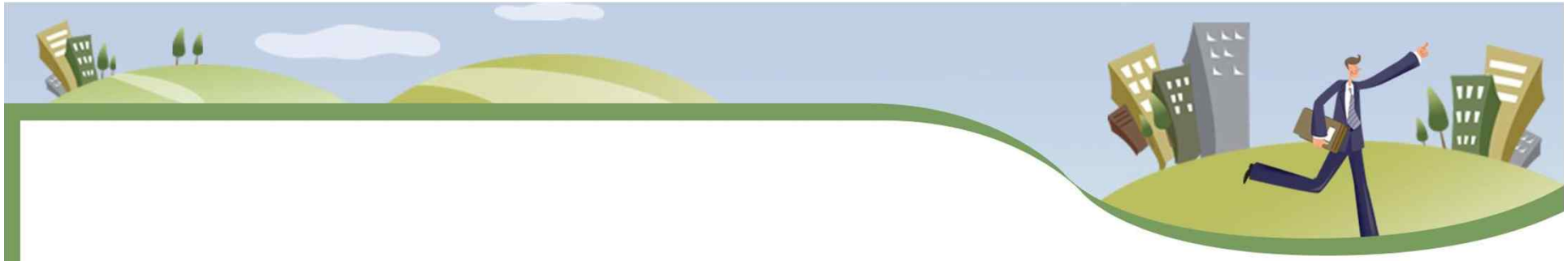
10. 데이터 조작(DML)



학습 목표

학습을 마친 후 여러분은 …

- 테이블에 새로운 행을 추가하고 기존의 데이터를 변경하고 삭제할 수 있다.
- 9i 추가된 MERGE , 다중 테이블 INSERT 구문을 활용할 수 있다.
- COMMIT, SAVEPOINT 및 ROLLBACK을 사용하여 Transaction을 제어할 수 있다

DML

명령어	설 명
INSERT	테이블에 새로운 행을 삽입
UPDATE	테이블에 있는 행을 변경
DELETE	테이블에 있는 행을 삭제
MERGE	테이블에 이미 데이터가 존재하면 **UPDATE,** 새로운 데이터면 **INSERT**

테이블에 새 행 추가

DEPTNO	DNAME	LOC_CODE
10	ACCOUNTING	A1
20	REARCH	B1
30	SALES	C1
40	OPERATION	A1

DEPT테이블에 새 행을
입력한다.

70	MARKETING	B1

INSERT 구문

- INSERT문을 사용하여 테이블에 새 행을 입력한다.
- 한번에 한 행만 삽입할 수 있다.

```
INSERT INTO table [( column [, column...] )]
VALUES (value [, value...] );
```

NULL 값을 갖는 행 삽입

- 암시적 방법: 열 목록에서 열을 생략한다.

```
INSERT  INTO   dept ( deptno , dname)
VALUES                    (70, 'MIS');
```

- 명시적 방법: NULL 키워드를 지정한다.

```
INSERT  INTO   dept
VALUES                    (70, 'MIS' , NULL);
```

특정 값을 갖는 행 삽입

- 현재 날짜/시간 입력:SYSDATE 함수 이용

```
INSERT  INTO   emp ( empno , ename, hiredate)
VALUES                  (7233, 'PAUL', SYSDATE);
```

- 현재 사용자 이름입력: USER 함수 이용

```
INSERT  INTO   emp (empno, ename, sal)
VALUES                  (7234, user  , 3400);
```

INSERT 구문

- 서브쿼리를 이용하여 새 행을 입력한다.
- INSERT 절의 열 수와 서브쿼리의 열수가 일치해야 한다.
- 한번에 여러 행을 입력할 수 있다.

```
INSERT INTO table [( column[, column...] )]
subquery;
```

테이블 데이터 변경

DEPTNO	DNAME	LOC_CODE
10	ACCOUNTING	A1
20	REARCH	B1
30	SALES	C1
40	OPERATION	A1

```
UPDATE dept
SET dname='test' ;
```

DEPTNO	DNAME	LOC_CODE
10	test	A1
20	test	B1
30	test	C1
40	test	A1

UPDATE 문

- UPDATE문을 사용하여 기존의 행을 수정한다.
- 필요한 경우 한 번에 여러 행을 수정 할 수 있다.

```
UPDATE table
SET column = value [, column = value, ... ]
[WHERE condition];
```

테이블 데이터 수정

- WHERE절을 생략하면 테이블의 모든 행이 수정된다

```
UPDATE      emp
SET         deptno= 20 ,  job ='CLERK';
```

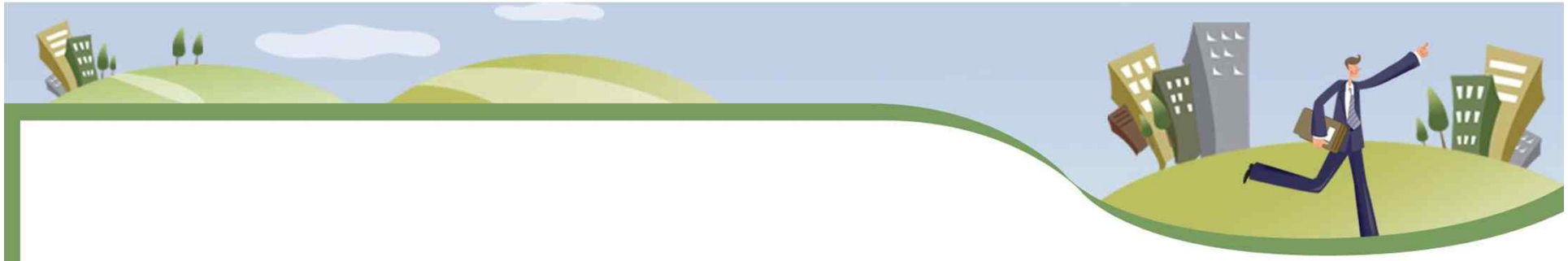
- WHERE절을 사용하여 특정 행을 수정한다.

```
UPDATE      emp
SET         deptno  = 20
WHERE       empno = 7233;
```

데이블 데이터 수정

- 서브 쿼리를 이용한 데이터를 수정

```
UPDATE      emp
SET         deptno = ( SELECT    deptno
                       FROM      emp
                       WHERE     ename = 'SCOTT ')
WHERE       empno  =  7902;
```



행 제거

DEPTNO	DNAME	LOC_CODE
10	ACCOUNTING	A1
20	REARCH	B1
30	SALES	C1
40	OPERATION	A1

```
DELETE  dept
WHERE  deptno=30;
```

DEPTNO	DNAME	LOC_CODE
10	ACCOUNTING	A1
20	REARCH	B1
40	OPERATION	A1

DELETE 문

- DELETE문을 사용하여 기존 행을 삭제한다.
- WHERE절을 지정하여 특정 행을 삭제한다.
- WHERE절을 생략하면 모든 행이 삭제된다.

```
DELETE [FROM] table
[WHERE condition];
```

테이블 데이터 삭제

- WHERE절을 생략하면 테이블의 모든 행이 삭제된다

```
DELETE  FROM      emp  ;
```

- WHERE절을 사용하여 특정 행을 삭제한다.

```
DELETE      emp
WHERE       empno = 7233;
```

데이블 데이터 삭제

- 서브 쿼리를 이용한 데이터를 삭제

```
DELETE      emp
WHERE       deptno = ( SELECT    deptno
                       FROM      emp
                       WHERE     ename = 'SCOTT ') ;
```

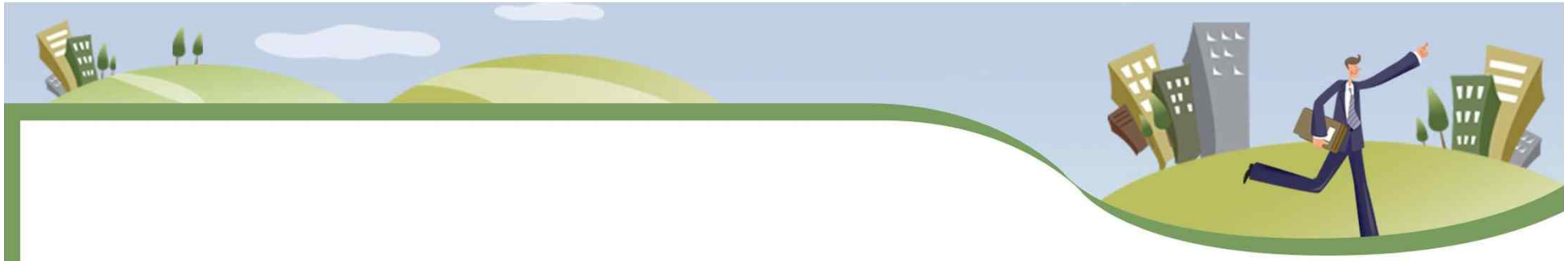
MERGE 문

- 데이터베이스 테이블에서 조건에 따라 데이터를 갱신하거나 삽입하는 기능을 제공한다.

- 해당 행이 존재하는 경우 UPDATE를 수행하고, 새로운 행일 경우 INSERT를 수행한다.
  - 성능이 향상되고 사용하기 편리하다
  - 데이터 웨어하우징 응용프로그램에 유용하다.

MERGE 구문

```
MERGE  INTO table_name table_alias
USING (table|view|subquery) alias
ON (join condition)
WHEN MATCHED THEN
       UPDATE SET col1 = col_val1,
                  col2 = col_val2
WHEN NOT MATCHED THEN
       INSERT (column_list)
       VALUES (column_values);
```



다중 테이블 입력

- 무조건 INSERT
- 조건 ALL INSRET
- 조건 FIRST INSERT
- 피벗 INSERT

```
 INSERT [ALL] [conditional_insert_clause]
[insert_into_clause values_clause] [subquery]
```

```
 [ALL] [FIRST]
 [WHEN condition THEN]
       [insert_into_clause values_clause]
 [ELSE] [insert_into_clause  values_clause]
```

무조건 INSERT ALL

```
INSERT ALL
        INTO  sal_history VALUES (EMPID, HIREDATE, SAL)
        INTO mgr_history VALUES (EMPID, MGR, SAL)
       SELECT employee_id  EMPID,
           hire_date  HIREDATE, salary  SAL,
           manager_id  MGR
       FROM employees
       WHERE employee_id > 200 ;
```

조건 INSERT ALL

```
INSERT ALL
   WHEN SAL > 10000  THEN
      INTO sal_history  VALUES (EMPID, HIREDATE, SAL)
    WHEN MGR > 200    THEN
      INTO mgr_history VALUES (EMPID, MGR, SAL)
    SELECT employee_id  EMPID,
         hire_date  HIREDATE, salary  SAL,
         manager_id  MGR
     FROM employees
    WHERE employee_id > 200 ;
```

조건 FIRST INSERT

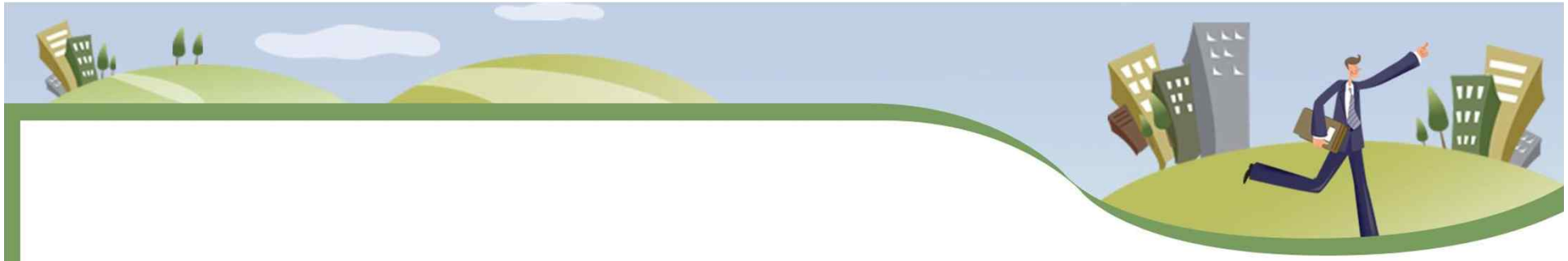
```
INSERT FIRST
      WHEN SAL > 25000          THEN
          INTO special_sal VALUES (DEPTID, SAL)
      WHEN HIREDATE like ('%00%') THEN
          INTO hire_history_00 VALUES (DEPTID, HIREDATE)
      WHEN HIREDATE like ('%99%') THEN
          INTO hire_history_99 VALUES (DEPTID, HIREDATE)
      ELSE
          INTO  hire_history   VALUES (DEPTID, HIREDATE)
    SELECT department_id  DEPTID, SUM(salary)  SAL,
          MAX(hire_date)  HIREDATE
      FROM employees
     GROUP BY department_id ;
```

피벗 INSERT

```
INSERT  ALL
    INTO  sales_info  VALUES (emp_id, week_id, sales_MON)
    INTO  sales_info  VALUES (emp_id, week_id, sales_TUE)
    INTO  sales_info  VALUES (emp_id, week_id, sales_WED)
    INTO  sales_info  VALUES (emp_id, week_id, sales_THU)
    INTO  sales_info  VALUES (emp_id, week_id, sales_FRI)
    SELECT  employee_id, week_id, sales_MON, sales_THE,
                sales_WED, sales_THU, sales_FRI
    FROM     sales_source_data ;
```

트랜잭션이란?

- 데이터베이스 Transaction은 다음 중 하나로 구성됩니다.
  - 데이터를 변경하는 하나 이상의 DML문
  - DDL문 하나
  - DCL문 하나



트랜잭션

- 명시적(Explicit) Transaction 제어
  - COMMIT
  - SAVEPOINT 이름
  - ROLLBACK
- 암시적(Implicit) Transaction 제어
  - 자동 COMMIT
  - 자동 ROLLBACK

Savepoint

- Savepoint 사용

```
C:\> sqlplus  scott/tiger
SQL>
SQL>  INSERT INTO DEPT
          VALUES (99, '인사과', '서울');
SQL> SAVEPOINT A;
SQL>
SQL> UPDATE  INTO EMP
         SET MGR = 7902
         WHERE EMPNO = 7934;
SQL> ROLLBACK TO  A;
SQL>
SQL> DELETE FROM EMP;
SQL> COMMIT;
```

INSERT문의 실행까지를 표시합니다

UPDATE문 만 실행 취소되며 **INSERT** 문의 트랜잭션은 아직 유효합니다.

SAVEPOINT A 이후에 실행된 **INSERT**와 **DELETE**문의 결과가 테이블에 영구히 반영됩니다.

Read Consistency

사원정보

사원번호	사원명	직급	급여
1	Scott	차장	300
2	Tom	과장	250
3	Jane	과장	200

A

```
UPDATE 사원정보
SET 급여 = 급여 * 1.1;
```

```
1    Scott  차장  330
2    Tom    과장   275
3    Jane   과장  220
     ..................
```

C

```
SELECT * FROM 사원정보;
```

```
1    Scott   차장  300
2    Tom     과장  250
3    Jane    과장  200
     ..................
```

B

```
UPDATE 사원정보
SET 급여 = 급여 * 2;
```

```
1    Scott  차장  600
2    Tom    과장  500
3    Jane   과장  400
     ..................
```

LOCK

오라클 데이터베이스에서의 lock은 다음 기능을 가진다

- 동시에 수행되는 Transaction 간에 파괴적인 상호 작용을 방지한다.
- 사용자 작업이 필요 없다.
- 자동으로 최대한 낮은 레벨의 제한을 사용한다.
- Transaction 수행 기간 동안 LOCK은 유지된다.