UNIT 25

객체 관계형 특성

DB2가 지원하는 객체 관계형 (Object-Relational) 특성인 사용자 정의 개별 유형과 사용자 정의 구조화 유형 (UDT), 사용자 정의 함수(UDF), 사용자 정의 메소드(UDM), 사용자 정의 구조화 유형을 이용한 테이블 (Typed Table) 등에 대해 설명합니다.

DB2 9.7 개발자 가이드

Administrator Edition

- 객체 관계형 특성의 개요
- 사용자 정의 개별 유형
- 사용자 정의 구조화 유형

TOPIC 04-01 객체 관계형 특성의 개요

Point

DB2에서의 Object-Relational 의 개념정의를 알아봅니다.

Tip

DB2의 확장된 기능으로 RDBMS로서의 Object Oriented의 개념과 Methodology를 함께 사용할 수 있습니다.

1 DB2 Object Extensions

➲ 컴퓨터 프로그램 언어 기술에 있어, 최근의 중요한 개발 기술의 하나는 Object-Orientation(객체 지향)입니다. 객체지향이라 함은 응용프로그램의 Object가 classification에 의해 서로 연관이 있는 독립된 Object로서 모델로 만들어 질 수 있는 개념입니다.

➲ 내부의 구체적인 구현사항은 숨겨진 채로, 외부의 function과 attribute는 구체화 되어 공개됩니다.

➲ DB2의 Object 기술은 RDBMS기술을 뿐만 아니라 포함하여 많은 Object 기술과 관련하여 많은 장점을 제공합니다.

2 DB2 Object-Relational 특성

➲ Data Type for very large objects
- TEXT, AUDIO, ENGINEERING DATA, VIDEO
- Binary Large Objects (BLOB)
- Character Large Objects (CLOB)
- Double-Byte Character Large Objects (DBCLOB)

➲ User Defined data Type
- Distinct Type
- Structured Type

➲ User Defined behaviors
- User Defined functions (UDF)
- User Defined methods : encapsulated with Structured type.

➲ Index extensions
- External Table function정의를 통해 structure type 및 distinct type에 대한 Index Key값 변경 및 성능최적화를 위한 검색 기능 제공

➲ Constraints
- Unique
- Referential integrity
- Table check
- Triggers

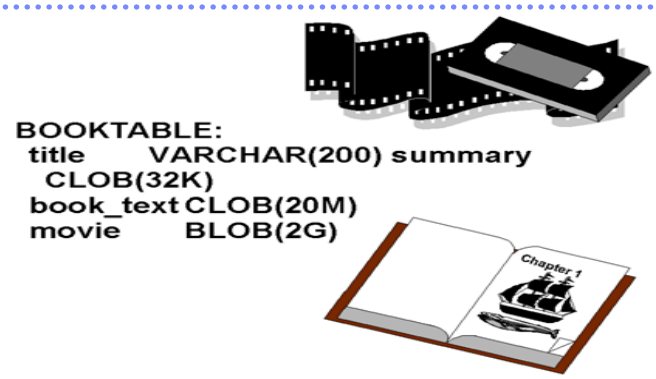


Figure 2501A DB2 Object-Relational 특성

TOPIC 04-02 사용자 정의 개별 유형

Point

User Defined Distinct Type에 대해 알아봅니다.

Tip

User Defined Distinct type을 통해 확장성, 유연성, 일관성 및 성능을 제공합니다.

1 User Defined Distinct Type

➲ **Extensibility**
새로운 Type을 정의 함으로써, Application에서 사용되는 데이터 Type을 다양하게 쓸 수 있습니다.

➲ **Flexibility**
User Defined Function에 사용하여 새로운 데이터 타입을 위한 별도의 Function을 기술함으로써 시스템의 다양성을 제공합니다.

➲ **Consistency**
Distinct Type을 통해 일관된 Process를 합니다. Distinct Type에 정의된 해당 Function을 수행함으로써 일관성을 유지합니다.

➲ **Encapsulation**
Distinct Type에 대한 Function과 연산자를 정의할 수 있습니다. 이를 통해, 수행중인 Application은 distinct type 내의 정의된 Function 내부의 Logic과 독립적으로 수행가능 합니다.

➲ **Performance**
Distinct type은 DBMS에 통합됩니다. 내부적으로는 내장 데이터 타입과 동일한 형태로 표현되며, 내장함수, 비교연산자 및 인덱스 등을 사용함에 있어 동일한 효율성을 제공합니다.

다른 통화(Currency)를 다룰 필요가 있고, 이 통화가 서로 비교되거나 Query에 의해 직접적으로 다뤄지는 것을 허락하지 않는 것을 보증하기 위한 업무를 가정하고 이를 위한 Type을 정의합니다.

즉, 서로 다른 화폐 단위와 가치 비교를 할 경우, 전환이 필요합니다.이를 위해, 필요한 각종 통화를 정의합니다.

```
CREATE DISTINCT TYPE US_DOLLAR
                    AS DECIMAL (9,2) WITH COMPARISONS
CREATE DISTINCT TYPE CANADIAN_DOLLAR
                    AS DECIMAL (9,2) WITH COMPARISONS
CREATE DISTINCT TYPE EURO
                    AS DECIMAL (9,2) WITH COMPARISONS
```

TOPIC 04-02 사용자 정의 개별 유형

Point

User Defined Distinct Type을 통한 Table 정의를 살펴 봅니다.

입사 지원자들의 양식이 Tabled에 저장되었다면, 이 양식으로부터 정보를 얻어내기 위해 function을 사용할 것입니다. 이를 위해 별도의 Type을 정의합니다.

```
CREATE DISTINCT TYPE PERSONAL.APPLICATION_FORM
                                    AS CLOB(32K)
```

LOB Type은 Function을 지원하지 않으므로, 이를 위한 별도의 Type을 통해 Function을 작성합니다.

2 User Defined Distinct Type 을 통한 Table 정의

Sales :

```
➲ CREATE TABLE US_SALES (
    PRODUCT_ITEM INTEGER,
    MONTH INTEGER CHECK (MONTH BETWEEN 1 AND 12),
    YEAR INTEGER CHECK (YEAR > 1985),
    TOTAL US_DOLLAR)

➲ CREATE TABLE CANADIAN_SALES (
    PRODUCT_ITEM INTEGER,
    MONTH INTEGER CHECK (MONTH BETWEEN 1 AND 12),
    YEAR INTEGER CHECK (YEAR > 1985),
    TOTAL CANADIAN_DOLLAR)

➲ CREATE TABLE GERMAN_SALES (
    PRODUCT_ITEM INTEGER,
    MONTH INTEGER CHECK (MONTH BETWEEN 1 AND 12),
    YEAR INTEGER CHECK (YEAR > 1985),
    TOTAL EURO)
```

TOPIC 04-02 사용자 정의 개별 유형

Point

Distinct Type에 대한 예를 통해 살펴 봅니다.

3 Distinct Type 에 대한 사용 예

➲ Distinct Type 과 상수의 비교

```
SELECT PRODUCT_ITEM
  FROM US_SALES
 WHERE TOTAL > US_DOLLAR (100000)
   AND month = 7
   AND year = 1999;

SELECT PRODUCT_ITEM
  FROM US_SALES
 WHERE TOTAL > CAST (100000 AS us_dollar)
   AND MONTH = 7
   AND YEAR = 1999;
```

➲ 다른 Type 간의 Cast

```
CREATE FUNCTION EURO_TO_US_DOUBL ( EURO DOUBLE )
RETURNS DOUBLE SPECIFIC EURO_TO_US_DOUBL_DO
LANGUAGE SQL CONTAINS SQL
NO EXTERNAL ACTION
NOT DETERMINISTIC
RETURN EURO * 1.2;

CREATE FUNCTION CDN_TO_US_DOUBL ( CDN DOUBLE )
RETURNS DOUBLE SPECIFIC CDN_TO_US_DOUBL
LANGUAGE SQL CONTAINS SQL
NO EXTERNAL ACTION
NOT DETERMINISTIC
RETURN CDN * 0.87;

CREATE FUNCTION EURO_TO_US_DEC (DECIMAL(9,2))
RETURNS DECIMAL(9,2)
SOURCE EURO_TO_US_DOUBL (DOUBLE);

CREATE FUNCTION US_DOLLAR (EURO)
RETURNS US_DOLLAR
SOURCE EURO_TO_US_DEC (DECIMAL());

CREATE FUNCTION CDN_TO_US_DEC (DECIMAL(9,2))
RETURNS DECIMAL(9,2)
SOURCE CDN_TO_US_DOUBL (DOUBLE);

CREATE FUNCTION US_DOLLAR (CANADIAN_DOLLAR)
RETURNS US_DOLLAR
SOURCE CDN_TO_US_DEC (DECIMAL());
```

TOPIC 04-02 사용자 정의 개별 유형

Point

Distinct Type에 대한 예를 통해 살펴 봅니다.

➲ 다른 Type 간의 비교

```
SELECT US.PRODUCT_ITEM, US.TOTAL
  FROM US_SALES AS US
     ,CANADIAN_SALES AS CDN
     ,GERMAN_SALES AS GERMAN
 WHERE US.PRODUCT_ITEM = CDN.PRODUCT_ITEM
   AND US.PRODUCT_ITEM = GERMAN.PRODUCT_ITEM
   AND US.TOTAL > US_DOLLAR (CDN.TOTAL)
   AND US.TOTAL > US_DOLLAR (GERMAN.TOTAL)
   AND US.MONTH = 7
   AND US.YEAR = 1999
   AND CDN.MONTH = 7
   AND CDN.YEAR = 1999
   AND GERMAN.MONTH = 7
   AND GERMAN.YEAR = 1999;
```

➲ Sourced UDF

```
CREATE FUNCTION SUM (EURO)
RETURNS EURO
SOURCE SYSIBM.SUM (DECIMAL());

CREATE FUNCTION SUM (CANADIAN_DOLLAR)
RETURNS CANADIAN_DOLLAR
SOURCE SYSIBM.SUM (DECIMAL());

CREATE FUNCTION SUM (US_DOLLAR)
RETURNS US_DOLLAR
SOURCE SYSIBM.SUM (DECIMAL());

SELECT PRODUCT_ITEM, US_DOLLAR (SUM (TOTAL))
FROM CANADIAN_SALES
WHERE YEAR = 1994
GROUP BY PRODUCT_ITEM;
```

TOPIC 04-02 사용자 정의 개별 유형

Point

Distinct Type에 대한 예를 통해 살펴 봅니다.

➲ 다른 Type 간의 Assignment

```
CREATE TABLE US_SALES_2006
(PRODUCT_ITEM INTEGER,
 TOTAL        US_DOLLAR);

CREATE TABLE GERMAN_SALES_2006
(PRODUCT_ITEM INTEGER,
 TOTAL        US_DOLLAR);

CREATE TABLE CANADIAN_SALES_2006
(PRODUCT_ITEM INTEGER,
 TOTAL        US_DOLLAR);
```

```
INSERT INTO US_SALES_2006
SELECT PRODUCT_ITEM, SUM (TOTAL)
  FROM US_SALES
 WHERE YEAR = 1994
 GROUP BY PRODUCT_ITEM;

INSERT INTO GERMAN_SALES_2006
SELECT PRODUCT_ITEM, US_DOLLAR (SUM (TOTAL))
  FROM GERMAN_SALES
 WHERE YEAR = 1994
 GROUP BY PRODUCT_ITEM;

INSERT INTO CANADIAN_SALES_2006
SELECT PRODUCT_ITEM, US_DOLLAR (SUM (TOTAL))
  FROM CANADIAN_SALES
 WHERE YEAR = 1994
 GROUP BY PRODUCT_ITEM;
```

➲ 다른 Type 간의 UNION

```
CREATE VIEW ALL_SALES AS
SELECT PRODUCT_ITEM, MONTH, YEAR, TOTAL
  FROM US_SALES
 UNION ALL
SELECT PRODUCT_ITEM, MONTH, YEAR
     , US_DOLLAR (TOTAL) TOTAL
  FROM CANADIAN_SALES
 UNION ALL
SELECT PRODUCT_ITEM, MONTH, YEAR
     , US_DOLLAR (TOTAL) TOTAL
  FROM GERMAN_SALES;
```

TOPIC 0403 사용자 정의 구조화 유형

Point

User Defined Structured Type에 대해 살펴 봅니다.

Tip

"AS"절 이하에 Type과 관련된 속성을 정의합니다. BusinessUnit_T는 Name과 HeadCount 속성을 가집니다. Structured Type을 생성하기 위해서는 "MODE DB2SQL"을 지정해야 합니다.

1 User Defined Structured Type

➲ Structured Type은 잘 Attribute로 구성된 잘 정의된 구조체 Modeling에 유용합니다. 각 Attribute는 Type의 속성들입니다.

➲ Type을 생성하기 위해, Type의 이름, attribute의 이름 및 Data type를 기술 합니다. 또한 선택적으로 Type에 대한 reference 방법을 정의합니다.

➲ 예제 : BusinessUnit_T 정의

```
CREATE TYPE BusinessUnit_t AS (
      Name       VARCHAR(20)
     ,Headcount  INT
) REF  USING INT MODE DB2SQL;

CREATE TYPE ADDRESS_T AS (
      STREET VARCHAR(30)
     ,NUMBER VARCHAR(15)
     ,CITY   VARCHAR(30)
     ,STATE  VARCHAR(10)
) REF USING INTEGER MODE DB2SQL;
```

➲ Structured Type의 특징

1. Inheritance : Structured Type을 정의 할 때 subtype을 포함할 수 있습니다. 즉, subtype에 포함된 attribute를 재 사용할 수 있습니다.

2. Storing instance of structured type :
   → As a row in a table
   Create table Person of Person_t ...
   → As a value in a column
   Create table Properties
   (ParcelNum Int,
   Photo BLOB(2k),
   Address Address_t)...

TOPIC 04-03 사용자 정의 구조화 유형

Point

User Defined Structured Type에 대한 예를 통해 살펴 봅니다.

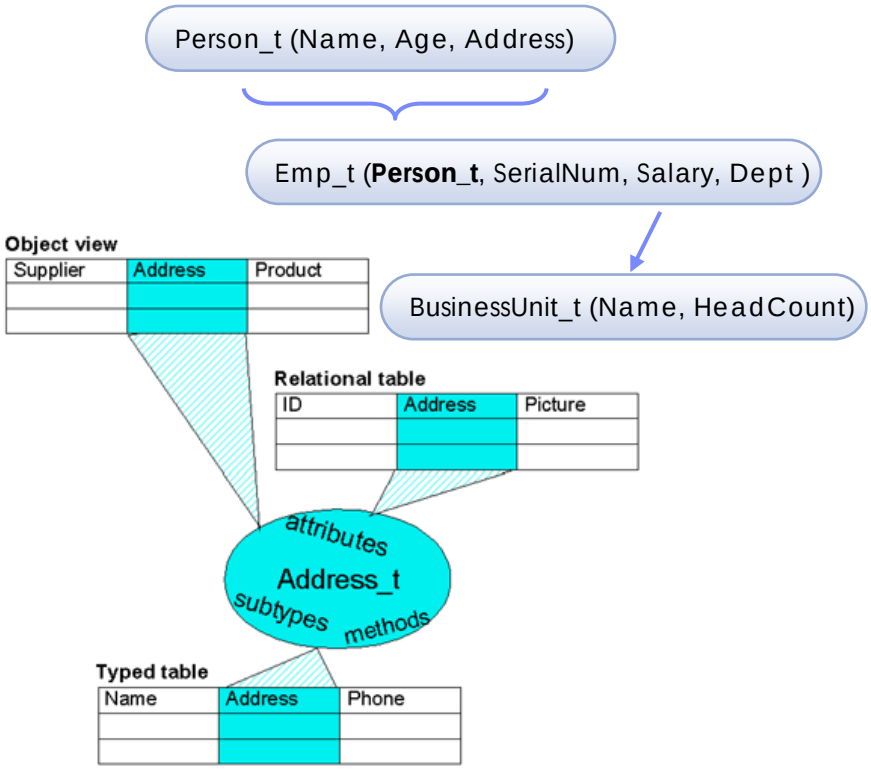


Figure 2503A • • • User Defined Structured Type 예제

2 Structured Type 생성

➲ Structured Type은 다른 Structured Type을 포함하여 생성될 수 있습니다.

Tip

Original Structured Type을 "Super Type"이라 하고, 새로 생성된 Structured Type을 "Sub Type" 이라고 합니다.

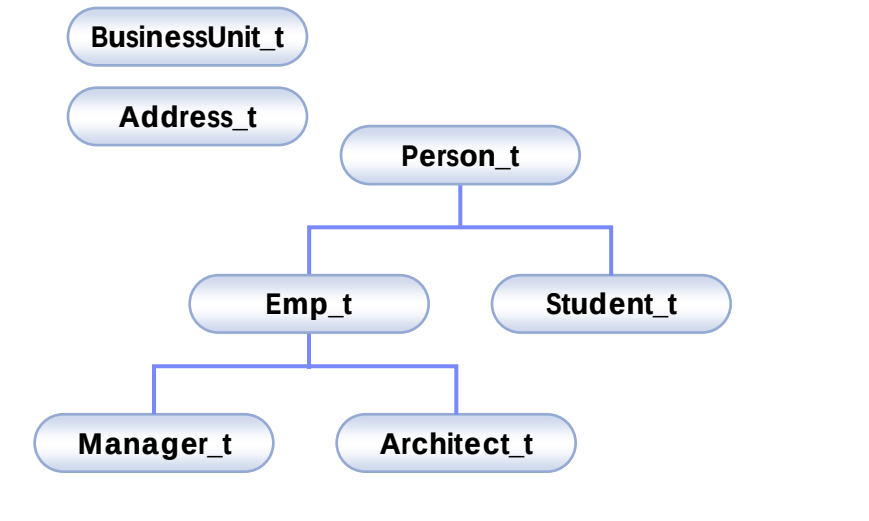


Figure 2503B • • • Structured Type예제

TOPIC 04-03 사용자 정의 구조화 유형

Point

User Defined Structured Type에 대한 예를 통해 살펴 봅니다.

➲ 예제 : BusinessUnit_T 정의

```
CREATE TYPE PERSON_T AS (
      NAME VARCHAR(20)
     ,AGE INT
     ,ADDRESS ADDRESS_T
)
INSTANTIABLE
REF USING VARCHAR(13) FOR BIT DATA MODE DB2SQL;


CREATE TYPE EMP_T UNDER PERSON_T AS (
      SERIALNUM INT
     ,SALARY    DECIMAL (9,2)
     ,DEPT      REF(BUSINESSUNIT_T)
) MODE DB2SQL;


CREATE TYPE STUDENT_T   UNDER PERSON_T AS (
      SERIALNUM    VARCHAR(6)
     ,GPA DOUBLE ) MODE DB2SQL;


CREATE TYPE MANAGER_T   UNDER EMP_T    AS (
      BONUS       DECIMAL(7,2)
) MODE DB2SQL;


CREATE TYPE ARCHITECT_T UNDER EMP_T    AS (
      STOCKOPTION INTEGER
) MODE DB2SQL;
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

Reference type과 Representation Type 에 대해 살펴 봅니다.

Tip

모든 Structured Type에 대해 DB2는 자동적으로 Companion Type을 생성합니다.

Tip

Reference Type을 Typed Table의 Reference를 저장하기 위해 사용합니다. 또한, Table의 각각의 Row를 참조하기 위해 사용합니다.

Tip

REF USING 절로 그 type을 명시하지 않으면, DB2는 VARCHAR FOR BIT DATA (16)의 Default Type을 사용합니다.

3 Reference Type

➲ Companion Type은 Reference Type이라 불리고, 그것이 참조하는 Structured Type은 Referenced Type이라고 불립니다.

➲ SQL 문장에서 Reference Type을 사용하기 위해서는, REF (type-name)을 사용하며, type-name은 Referenced type을 나타냅니다.

➲ DB2는 Type된 Table에서 Reference Type을 Object Identifier Column으로 사용합니다. Object Identifier는 Typed Table Hierarchy에서 Unique Identifier로 사용됩니다.

4 Representation Type

➲ Type Hierarchy의 Root Type을 생성할 때, "Create Type ... REF USING ..."을 사용하여 참조를 위한 Base Type을 명시합니다.
이때, Base Type을Representation Type이라고 합니다.

➲ Root Type의 Representation Type은 모든 Subtype에 상속됩니다.
"REF USING ..." 은 Type Hierarchy에서 Root type에서만 정의할 수 있습니다.

In the examples used throughout this section, the representation type for the BusinessUnit_t type is INTEGER, while the representation type for Person_t is VARCHAR(13).

BusinessUnit_t Representation Type : Integer
Person_t Representation Type : VARCHAR(13)

TOPIC 04-03 사용자 정의 구조화 유형

Point

Casting 과 Reference Type 에 대해 살펴 봅니다.

Tip

기본값은 Structured Type과 representation Type의 이름을 사용합니다.

5 Casting 과 Reference Type 비교

➲ DB2는 Reference Type과 Representation Type간의 Cast를 위해 자동으로 Casting function을 생성합니다.

➲ Create Type 문장의 "CAST WITH ..."절을 사용하여 Cast function의 이름을 지정할 수 있습니다.

➲ Ex : Create Type Person_t ...

다음은 "Create Type Person_t..."에 의해 자동으로 생성되는 Function 입니다.

```
CREATE FUNCTION VARCHAR (REF (Person_t) )
RETURNS VARCHAR;
```

또한, 반대의 Operation을 위한 function을 생성합니다.

```
CREATE FUNCTION Person_t(VARCHAR(13))
RETURNS REF (Person_t);
```

6 기타 System-Generated Routine

➲ DB2는 Structured Type을 생성할 때 마다, 내재적으로 Function과 method를 생성합니다 – Constructor, Observer, Modify

➲ Constructor

```
Constructor function :
CREATE FUNCTION Person_t  ( ) RETURNS Person_t
CREATE FUNCTION Manager_t ( ) RETURNS Manager_t
```

➲ Mutator Method

Mutator Method는 Object의 개별 Attribute별로 존재합니다. Mutator Method가 호출될 때, attribute의 새로운 값을 반환 받습니다.

예로, Person_t는 다음의 각각의 속성에 대해 Mutator Method를 만듭니다. - name, age, address.

```
ALTER TYPE Person_t
    ADD METHOD AGE(int)
  RETURNS Person_t;
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

Type에 대한 Method 정의에 대해 살펴 봅니다.

➲ Observer Method

Observer method는 Object의 개별 Attribute별로 존재합니다.
Observer Method는 Object의 값을 Return합니다.
예로, Person_t는 다음의 Observer method를 만듭니다.

```
ALTER TYPE Person_t
    ADD  METHOD AGE()
 RETURNS INTEGER;
```

7 Type 에 대한 Method 정의

Structured Type에 대해 Method를 정의 할 수 있습니다. Create Method 생성 이전에 Method Specification 정의 되어야 합니다.

➲ Method 생성

```
   ALTER TYPE EMP_T
     ADD METHOD CALC_BONUS (RATE DOUBLE)
  RETURNS DECIMAL(9,2)
 LANGUAGE SQL
 CONTAINS SQL
      NO EXTERNAL ACTION DETERMINISTIC;

   CREATE METHOD CALC_BONUS (RATE DOUBLE)
  RETURNS DECIMAL(9,2)
     FOR EMP_T
   RETURN SELF..SALARY * RATE;
-----
   ALTER TYPE EMP_T
     ADD METHOD CALC_BONUS (RATE INT)
  RETURNS DECIMAL(9,2)
 LANGUAGE SQL
 CONTAINS SQL
      NO EXTERNAL ACTION DETERMINISTIC;

   CREATE METHOD CALC_BONUS (RATE INT)
  RETURNS DECIMAL(9,2)
     FOR EMP_T
   RETURN SELF..SALARY * RATE;
```

TOPIC 0403 사용자 정의 구조화 유형

Point

Typed Table에 대한 Object 저장에 대해 살펴 봅니다.

Tip

Typed Table은 다른 Table이 참조할 수 있는 Identity Attribute를 가지고 있습니다.

8 Typed Table에 대한 Object 저장

➲ Typed Table에 Row 형태로 저장하거나, Structured type의 속성을 가진 Column에 저장 할 수 있습니다.

➲ Pserson_t를 위한 Create Table 문장

```
CREATE TABLE Person OF Person_t
(REF IS Oid USER GENERATED);
```

➲ Person Table에 Data Insert

```
INSERT INTO Person (Oid, Name, Age)
VALUES (Person_t('a'), 'Andrew', 29);
```

OID	NAME	AGE	Address
a	Adnrew	29	NULL

➲ Person Table에 Data Update

```
UPDATE Person SET Age=30 WHERE Name='Andrew';
```

OID	NAME	AGE	Address
a	Adnrew	30	NULL

➲ Person_t Table 의 subtable생성 → Emp_t를 위한 Table

Person table속성을 가지 Emp Table을 subtable 이라고 합니다.

```
CREATE TABLE Emp OF Emp_t UNDER Person
INHERIT SELECT PRIVILEGES
( SerialNum WITH OPTIONS NOT NULL,
  Dept      WITH OPTIONS SCOPE BusinessUnit);
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

Typed Table에 대한 Object 저장에 대해 살펴 봅니다.

➲ Employee Table에 Data Insert

```
INSERT INTO Emp
  (Oid, Name, Age, SerialNum, Salary)
VALUES
  (Emp_t('s'), 'Susan', 39, 24001, 37000.48);
```

OID	NAME	AGE	ADDRESS	SerialNum	Salary	Dept
s	Susan	39		24001	37000.48	

➲ Table 조회

```
SELECT oid, name, age, salary FROM emp;
OID     NAME    AGE         SALARY
------- ------- ----------- -----------
x'73'   Susan           39    37000.48

SELECT oid, name, age FROM person;
OID     NAME    AGE
------- ------- -----------
x'61'   Andrew          30
x'73'   Susan           39
```

TOPIC 0403 사용자 정의 구조화 유형

Point

Typed Table 간의 Relationship 정의에 대해 살펴 봅니다.

9 Typed Table간의 Relationship 정의

Typed Table과 Typed Table간의 관계를 정의할 수 있으며, 동일한 Typed Table에 대해 관계를 정의할 수 있습니다.

➲ Emp Table과 Business Table간의 관계 정의

```
update emp set dept = BusinessUnit_t(1)
  where oid = emp_t('s');
update emp set dept = NULL
  where oid = emp_t('s');
```

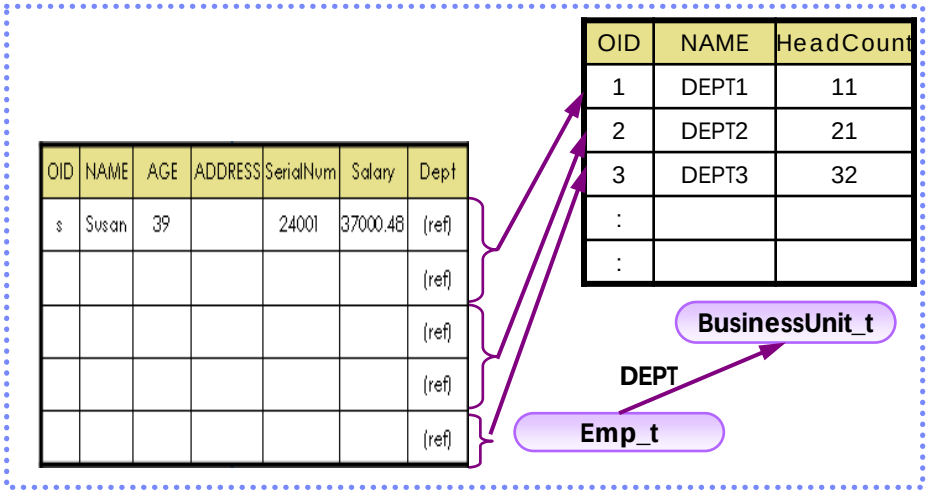


Figure 2503C Emp table과 Business Table간의 관계 정의

➲ Emp Table 조회

```
select oid, name
     ,dept->name
     ,dept->headcount
 from emp
where oid = emp_t('s')"

OID    NAME   NAME   HEADCOUNT
------ ------ ------ -----------
x'73'  Susan  DEPT1         11


INSERT INTO BusinessUnit (Oid, Name, Headcount)
  VALUES (BusinessUnit_t(1), 'DEPT1',11)
       ,(BusinessUnit_t(2), 'DEPT2',21)
       ,(BusinessUnit_t(3), 'DEPT3',32);

select * from businessunit;
OID         NAME    HEADCOUNT
----------- ------- -----------
          1 DEPT1           11
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

Column에 Object를 저장하는 것에 대해 살펴 봅니다.

10 Column에 Object 저장

DBMS의 built-in data type으로 Data를 저장하기에 적절하지 않을 경우, Object를 column에 저장할 수 있다.

➲ Person Table

OID	NAME (VARCHAR)	AGE (INT)	Address (Address_t)			
			Street	Number	City	State

➲ Person table에 address 저장

```
UPDATE EMP
  SET ADDRESS=ADDRESS..NUMBER('4869')..STREET('APPLETREE')
 WHERE NAME='FRANKY'
  AND ADDRESS..STATE='CA';

UPDATE EMP
  SET ADDRESS..NUMBER = '4869',
      ADDRESS..STREET = 'APPLETREE'
 WHERE NAME='FRANKY'
  AND ADDRESS..STATE='CA' ;

INSERT INTO Person (Oid, Name, Age, Address)
VALUES ( Person_t('B')
    , 'Billy'
    , 29
    , ADDRESS_T('Gang Name Dogok','123','Seoul','Korea'));
```

➲ Address_t의 Table function

```
CREATE FUNCTION ADDRESS_T
   (street VARCHAR(30),
    number VARCHAR(15),
    city   VARCHAR(30),
    state  VARCHAR(20) )
  RETURNS ADDRESS_T
 LANGUAGE SQL
  RETURN ADDRESS_T()..street(street)
                ..number(number)
                ..city(city)
                ..state(state);
```

TOPIC 0403 사용자 정의 구조화 유형

Point

Typed table에서 Structured Type 사용하는 것에 대해 살펴봅니다.

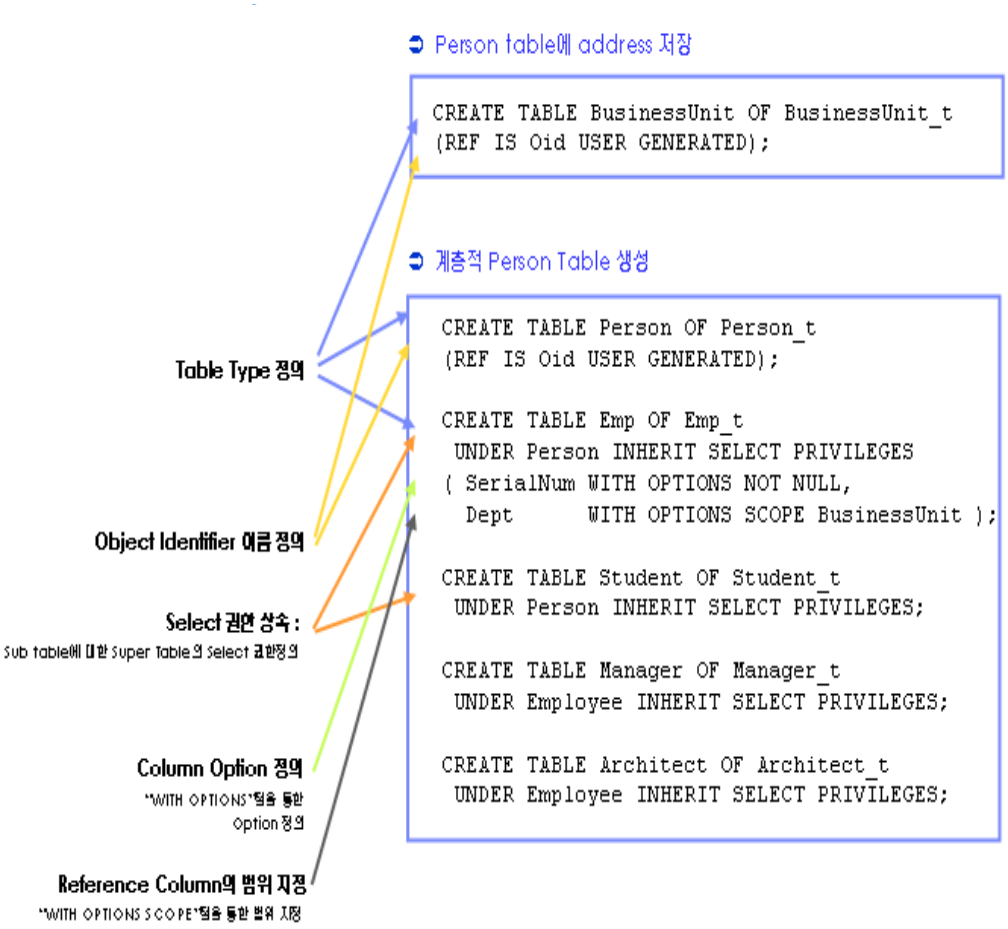
Tip

Typed Table은 "Create Type"문장을 통해 기술된 Object를 실제로 저장하는데 사용할 수 있습니다.

11 Typed Table에서 Structured Type 사용

"Create Table"문장을 사용하여 다양한 Typed Table을 생성 할 수 있습니다. 또한 Structured Type 구조를 가진 Type을 통해 계층적 Table구조를 생성 할 수 있습니다.

➲ Typed Table 생성



TOPIC 04-03 사용자 정의 구조화 유형

Point

Typed table Record 생성에 대해 살펴봅니다.

12 Typed Table Record 생성

➲ BusinessUnit Table

```
INSERT INTO BusinessUnit (Oid, Name, Headcount)
VALUES(BusinessUnit_t(1), 'Toy', 15);

INSERT INTO BusinessUnit (Oid, Name, Headcount)
VALUES(BusinessUnit_t(2), 'Shoe', 10);
```

➲ Person Table

```
INSERT INTO Person (Oid, Name, Age)
VALUES(Person_t('a'), 'Andrew', 20);

INSERT INTO Person (Oid, Name, Age)
VALUES(Person_t('b'), 'Bob', 30);

INSERT INTO Person (Oid, Name, Age)
VALUES(Person_t('c'), 'Cathy', 25);
```

➲ Person Table

```
INSERT INTO Emp (Oid, Name, Age, SerialNum, Salary, Dept)
VALUES(Employee_t('d'), 'Dennis', 26, 105, 30000,
BusinessUnit_t(1));

INSERT INTO Emp (Oid, Name, Age, SerialNum, Salary, Dept)
VALUES(Employee_t('e'), 'Eva', 31, 83, 45000,
BusinessUnit_t(2));

INSERT INTO Emp (Oid, Name, Age, SerialNum, Salary, Dept)
VALUES(Employee_t('f'), 'Franky', 28, 214, 39000,
BusinessUnit_t(2));
```

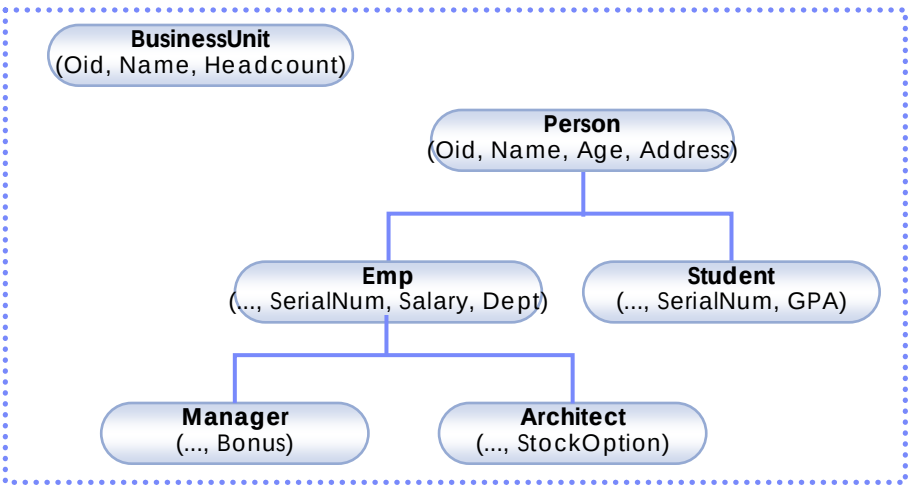


Figure 2503C Typed Table Record 생성 구조

TOPIC 04-03 사용자 정의 구조화 유형

Point

Typed table에서 Structured Type 사용하는 것에 대해 살펴봅니다.

➲ Student Table

```
INSERT INTO Student (Oid, Name, Age, SerialNum, GPA)
VALUES(Student_t('g'), 'Gordon', 19, '10245', 4.7);

INSERT INTO Student (Oid, Name, Age, SerialNum, GPA)
VALUES(Student_t('h'), 'Helen', 20, '10357', 3.5);
```

➲ Manager Table

```
INSERT INTO Manager (Oid, Name, Age, SerialNum,
   Salary, Dept, Bonus)
VALUES(Manager_t('i'), 'Iris', 35, 251, 55000,
   BusinessUnit_t(1), 12000);

INSERT INTO Manager (Oid, Name, Age, SerialNum,
   Salary, Dept, Bonus)
VALUES(Manager_t('j'), 'Christina', 10, 317, 85000,
   BusinessUnit_t(1), 25000);

INSERT INTO Manager (Oid, Name, Age, SerialNum,
   Salary, Dept, Bonus)
VALUES(Manager_t('k'), 'Ken', 55, 482, 105000,
   BusinessUnit_t(2), 48000);
```

➲ Architecture Table

```
INSERT INTO Architect (Oid, Name, Age, SerialNum,
   Salary, Dept, StockOption)
VALUES(Architect_t('l'), 'Leo', 35, 661, 92000,
   BusinessUnit_t(2), 20000);

INSERT INTO Architect (Oid, Name, Age, SerialNum,
   Salary, Dept, StockOption)
VALUES( Architect_t('m'), 'Brian', 7, 882, 112000,
    (SELECT Oid FROM BusinessUnit WHERE name = 'Toy'),
      30000);
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

Reference Type 생성에 대해 살펴봅니다.

13 Reference Type 생성

➲ 각각의 Structured type에 대해, DB2는 이에 상응하는 reference type을 제공합니다. 예를 들어, Person_t type을 생성할 때, REF(Person_t) type을 생성합니다.

➲ Create Table문장의 "With Options"절을 사용하여 Column과 Object간의 연관관계를 정의할 수 있습니다.

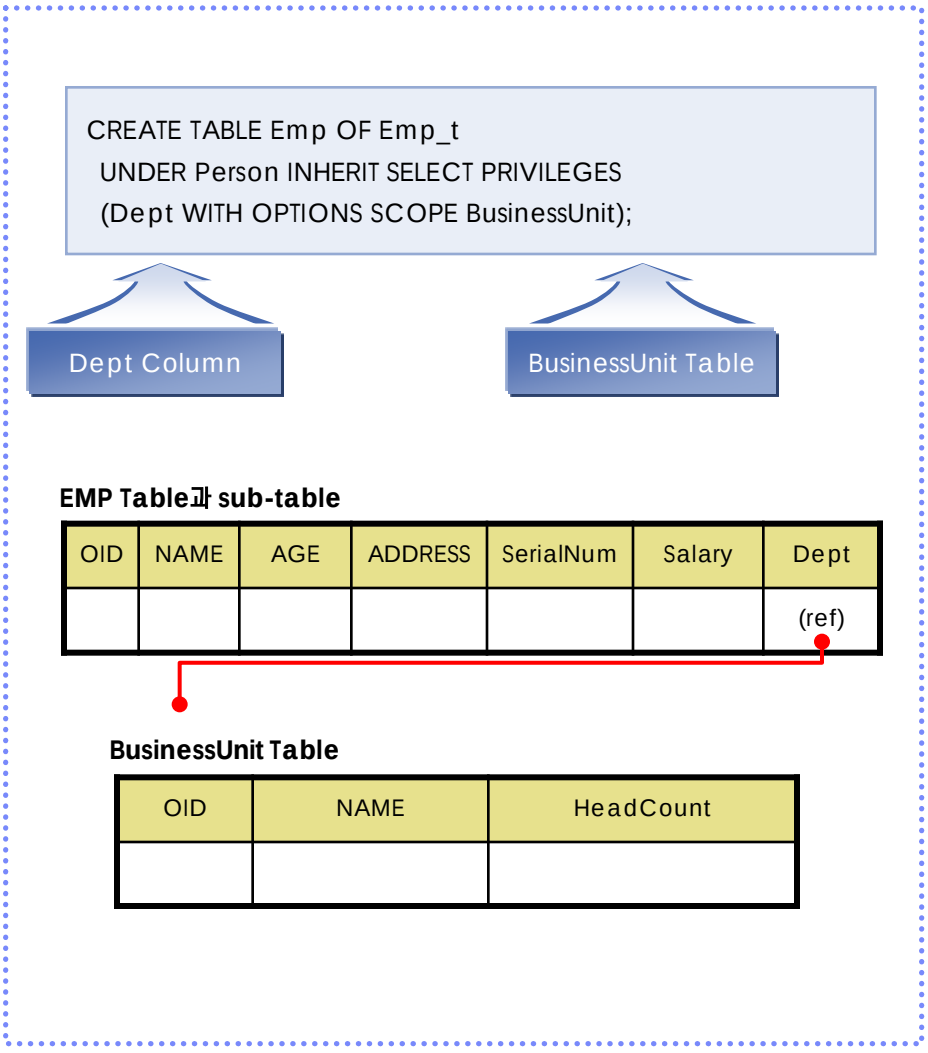


Figure 2503D Reference Type 생성

TOPIC 04-03 사용자 정의 구조화 유형

Point

Self-Reference Relationship 정의에 대해 살펴봅니다.

14 Self-Reference Relationship 정의

➲ Type 생성

```
CREATE TYPE COMPANY_T AS (
  NAME     VARCHAR(30),
  LOCATION VARCHAR(30)
) MODE DB2SQL ;

CREATE TYPE PART_T AS (
  DESCRIPT    VARCHAR(20),
  SUPPLIED_BY REF(COMPANY_T),
  USED_IN     REF(PART_T)
) MODE DB2SQL;
```

➲ Table 생성

```
CREATE TABLE SUPPLIERS OF COMPANY_T (
   REF         IS SUPPNO USER GENERATED);

CREATE TABLE PARTS OF PART_T (
   REF         IS PARTNO USER GENERATED,
   SUPPLIED_BY WITH OPTIONS SCOPE SUPPLIERS,
   USED_IN     WITH OPTIONS SCOPE PARTS);
```

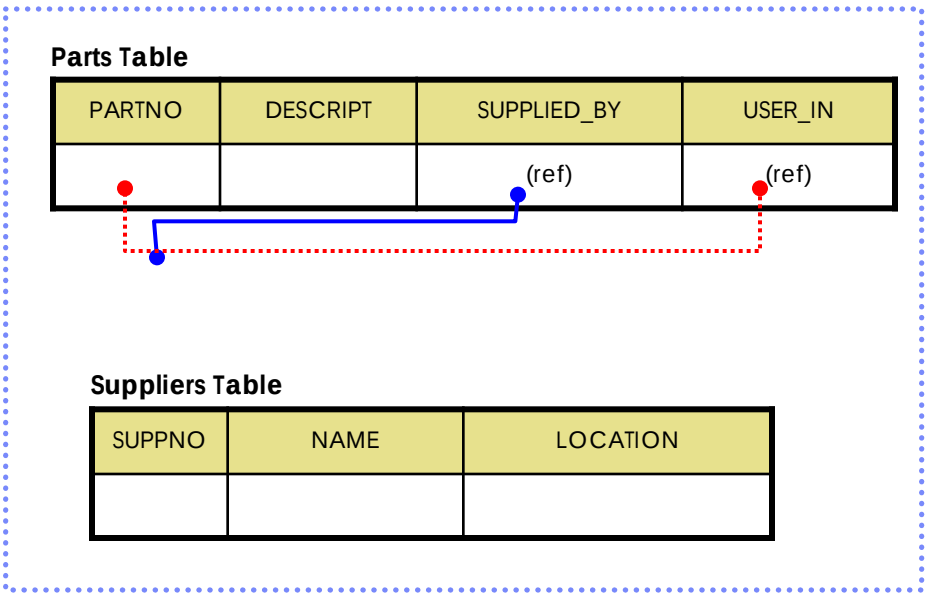


Figure 2503E ••• Self-Reference Relationship 정의

TOPIC 0403 사용자 정의 구조화 유형

Point

RI 와 Scoped Reference 의 차이에 대해 살펴봅니다.

Tip

RI관계를 강제적으로 정의 하려면 Referential Integrity를 정의하여야 한다.

15 RI (Referential Integrity) 와 Scoped Reference 의 차이

➲ Scoped Reference는 Table Object간 관계를 정의하지만 RI와는 다르다. Scope는 Target table에 대한 정보만을 제공한다.

➲ Scoped Reference는 다른 Object에 해당 값이 필요하거나, 강제적으로 제약하지는 않는다.

➲ Type 정의 및 Table 정의

```
CREATE TYPE  Empl_t AS (
  Name VARCHAR(10)
, Mgr REF(Empl_t)
) ref using integer MODE DB2SQL;

CREATE TABLE Empl   OF Empl_t (
  REF IS Oid USER GENERATED);
```

➲ RI 정의

```
ALTER TABLE Empl
  ADD CONSTRAINT pk1 UNIQUE(Oid);

ALTER TABLE Empl
  ADD CONSTRAINT fk1
     FOREIGN KEY(Mgr) REFERENCES Empl (Oid);
```

➲ Record 생성

```
insert into empl (oid,name)
values ( empl_t(1  ), '홍길동'              );

insert into empl (oid,name,mgr)
values ( empl_t(11 ), '김서방'   ,empl_t(1  ) );

insert into empl (oid,name,mgr)
values ( empl_t(111), '김서방111',empl_t(11 ) );

insert into empl (oid,name,mgr)
values ( empl_t(112), '김서방112',empl_t(11 ) );

insert into empl (oid,name,mgr)
values ( empl_t(12 ), '이서방'   ,empl_t(1  ) );

insert into empl (oid,name,mgr)
values ( empl_t(121), '이서방121',empl_t(12 ) );

insert into empl (oid,name,mgr)
values ( empl_t(122), '이서방122',empl_t(12 ) );
```

Empl Table

OID	NAME	MGR
		(ref)

Figure 2503F RI 와 Scoped Reference 차이

TOPIC 0403 사용자 정의 구조화 유형

Point

Typed View에 대해 살펴봅니다.

16 Typed View 정의

"Create View"문장을 사용하여 Typed View를 생성할 수 있습니다. 원하는 Attribute를 만들기 위해서는 Type을 생성할 수 있으며, 이를 바탕으로 View를 정의 할 수 있습니다.

➲ View를 위한 Type정의

```
CREATE TYPE VBUSINESSUNIT_T AS (
      NAME VARCHAR(20)
) REF USING INTEGER MODE DB2SQL;
```

➲ View를 정의

```
CREATE VIEW VBUSINESSUNIT OF VBUSINESSUNIT_T
MODE DB2SQL (REF IS VOID USER GENERATED)
AS SELECT VBUSINESSUNIT_T(INTEGER(OID))
       , NAME
    FROM BUSINESSUNIT;
```

➲ View 조회

```
SELECT * FROM VBUSINESSUNIT;

VOID       NAME
---------- ---------
         1 DEPT1
         2 DEPT2
         3 DEPT3
         4 DEPT4
         5 DEPT5
```

➲ View Hierarchy 생성

```
CREATE TYPE VPERSON_T
AS (NAME VARCHAR(20))
MODE DB2SQL;

CREATE TYPE VEMP_T UNDER VPERSON_T
AS (SALARY INT, DEPT REF(VBUSINESSUNIT_T))
MODE DB2SQL;
```

```
CREATE VIEW VPERSON OF VPERSON_T MODE DB2SQL
     (REF IS VOID USER GENERATED)
AS SELECT VPERSON_T (VARCHAR(OID))
       , NAME
    FROM ONLY(PERSON);

CREATE VIEW VEMP OF VEMP_T MODE DB2SQL
      UNDER VPERSON INHERIT SELECT PRIVILEGES
     (DEPT WITH OPTIONS SCOPE VBUSINESSUNIT)
AS SELECT VEMP_T(VARCHAR(OID))
       , NAME
       , SALARY
       , VBUSINESSUNIT_T(INTEGER(DEPT))
    FROM EMP;
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

조회에 대해 자세히 살펴 봅니다.

Tip

조회 권한이 있으면 해당 Table을 일반 Table조회 하듯이 조회 할 수 있습니다.

17 Typed Table 조회

➲ Person table 조회

```
select name, age from person;
```

18 Dereference Reference 에 대한 조회

Scoped Reference를 조회 하고자 할 때는 "Dereference Operation" 사용합니다.

Dereference Operator : ->
Scoped-reference-expression->column-in-target-typed-table

➲ Table 조회

```
select name
    , salary
    , dept->name
 from emp;
```

```
SELECT P.Descript
    , P.Supplied_by -> Location
 FROM Parts P
 WHERE P.Used_in -> Descript='Wing';
```

19 Built-in Function

```
DEREF           : Scope reference Column명을 통해 Object 확보
TYPE_NAME
TYPE_ID
TYPE_SCHEMA
```

➲ DEREF

```
DEREF ( Scoped-reference-expression )
```

TOPIC 0403 사용자 정의 구조화 유형

Point

조회에 대해 자세히 살펴 봅니다.

➲ TYPE_* function

```
CREATE TYPE PROJECT_T
AS (PROJID INT, RESPONSIBLE REF(EMP_T))
MODE DB2SQL;

CREATE TABLE PROJECT
OF PROJECT_T (REF IS OID USER GENERATED,
RESPONSIBLE WITH OPTIONS SCOPE EMP);


Insert into project
values ( project_t('A'),10, emp_t('s') );

SELECT OID
    , PROJID
    , RESPONSIBLE->NAME
    , TYPE_NAME  (DEREF(RESPONSIBLE))
    , TYPE_ID    (DEREF(RESPONSIBLE))
    , TYPE_SCHEMA(DEREF(RESPONSIBLE))
FROM PROJECT;
```

20 Query Specification

```
ONLY                    : 특정 type의 Object만 Return
IS OF                   : Type에 대한 유형 제어
OUTER                   : Object가 가질 수 있는 모든 Attribute 표시
GENERATE_UNIQUE()       : 유일한 값을 생성하는 변수
```

➲ ONLY

```
SELECT NAME
  FROM ONLY ( EMP );
```

➲ IS OF

```
 <expression> IS OF ( TYPE_NAME[, ... ] )

SELECT NAME
  FROM EMP E
 WHERE E.AGE > 15
   AND DEREF(E.OID) IS OF
     ( EMP_T, MANAGER_T, ARCHITECT_T);
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

조회에 대해 자세히 살펴 봅니다.

➲ OUTER

```
SELECT TYPE_NAME(DEREF(Oid)) TYPE_NAME
    , AGE
    , NAME
    , SALARY
    , STOCKOPTION
 FROM OUTER(EMP);
```

```
TYPE_NAME   AGE  NAME      SALARY STOCKOPTION
----------- ---- -------- --------- -----------
EMP_T         26 Dennis   30000.00         -
EMP_T         36 Charlie  34000.00         -
EMP_T         46 Bill     44000.00         -
EMP_T         36 Charlie  34000.00         -
EMP_T         46 Bill     44000.00         -
EMP_T         39 Susan    37000.48         -
ARCHITECT_T    7 Brian   112000.00     30000
EMP_T         35 Marie    55000.00         -
```

➲ GENERATE_UNIQUE()

```
INSERT INTO EMP
 (Oid, Name, Age, SerialNum, Salary, Dept)
VALUES
 (emp_t(generate_unique())
  , 'Dennis', 26, 105, 30000,BusinessUnit_t(1));
```

```
select oid, name, age from emp where serialnum=105

OID                         NAME   AGE
--------------------------- ------ ---
x'20060404092229377998000000' Dennis  26
```