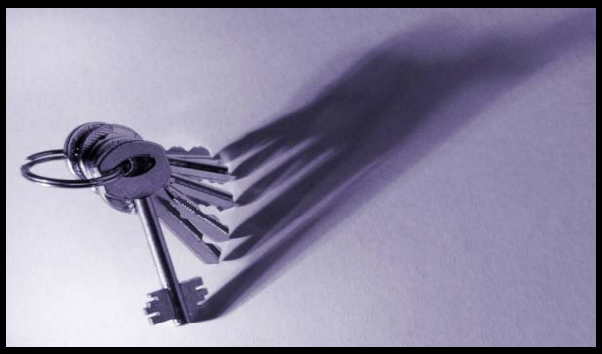
Oracle Data Type



Getting the most out of MetaLink

김재연

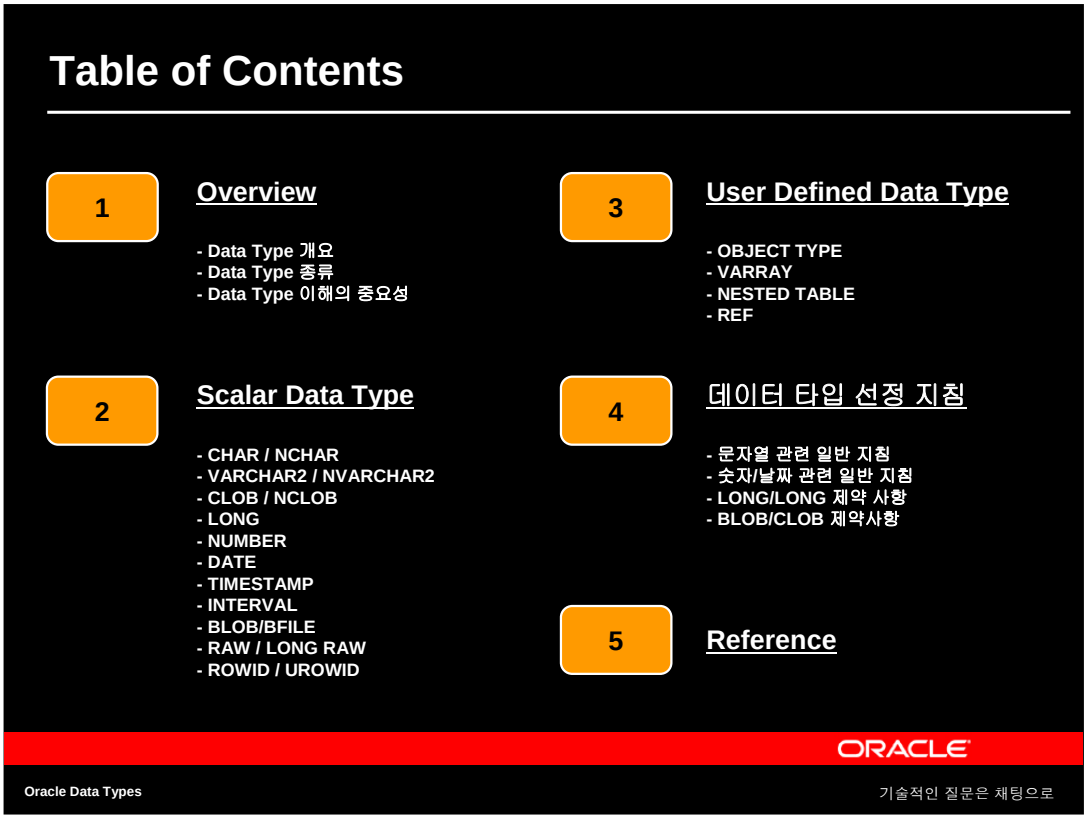
한국 오라클 (주)　제품지원실

ORACLE

Oracle Data Types　　기술적인 질문은 채팅으로

이번 세미나에선 Oracle9i기준으로 Oracle Data Type에 대해 알아 보도록 하겠습니다.

본 세미나에서는 Oracle에서 제공되는 내장 데이터 타입 (Built-In Data Type)과 사용자 정의 데이터 타입(User defined data type) 에 대한 개요 및 특징 , 내부 저장 방식에 대해서 소개해 드림으로써, Oracle Data Type선정에 도움을 드리고자 합니다.



본 세미나에서 다루는 내용은 다음과 같습니다.

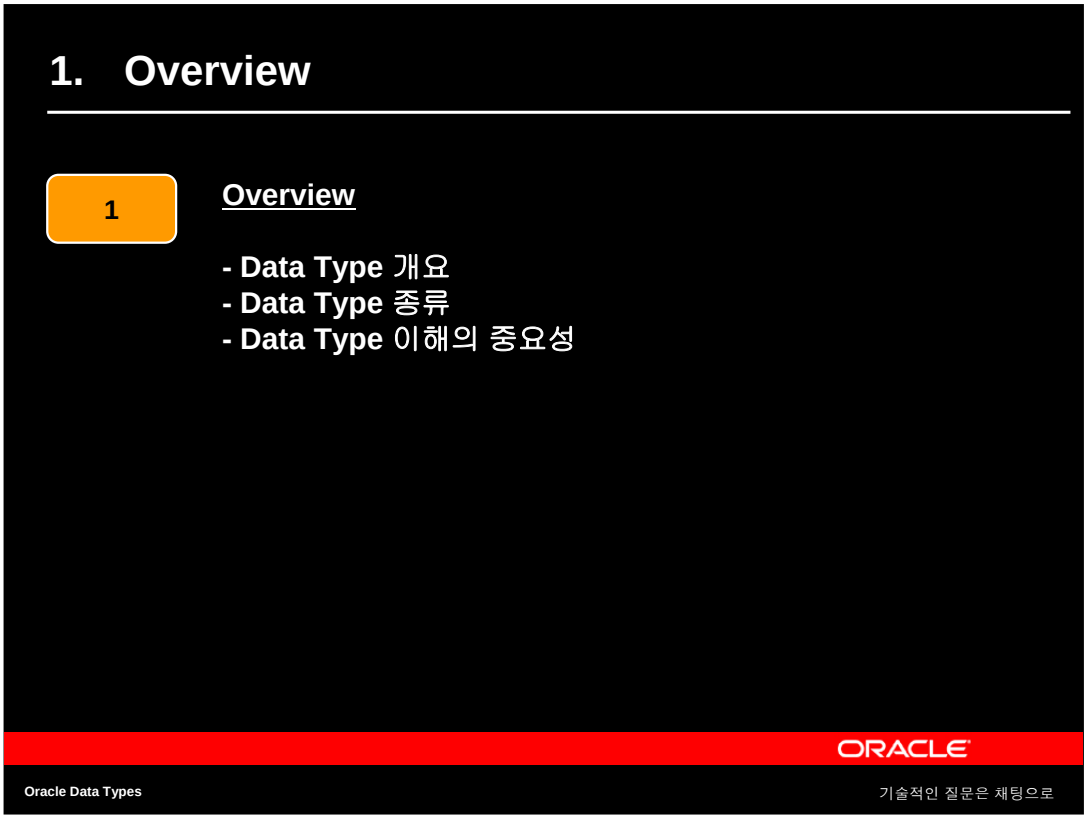
1장의 Overview를 통해 Data Type 개요, Data Type 종류, Data Type 이해의 중요성에 대해서 소개 드리겠습니다.
Oracle Data Type은 크게 내장 데이터 타입과 사용자 정의 데이터 타입으로 나누어 지는데,

2장에서는 스칼라 형태의 Oracle 내장 데이터 타입에 대한 설명 및 특징과 내부 저장방식에 대해 알아 보겠습니다.

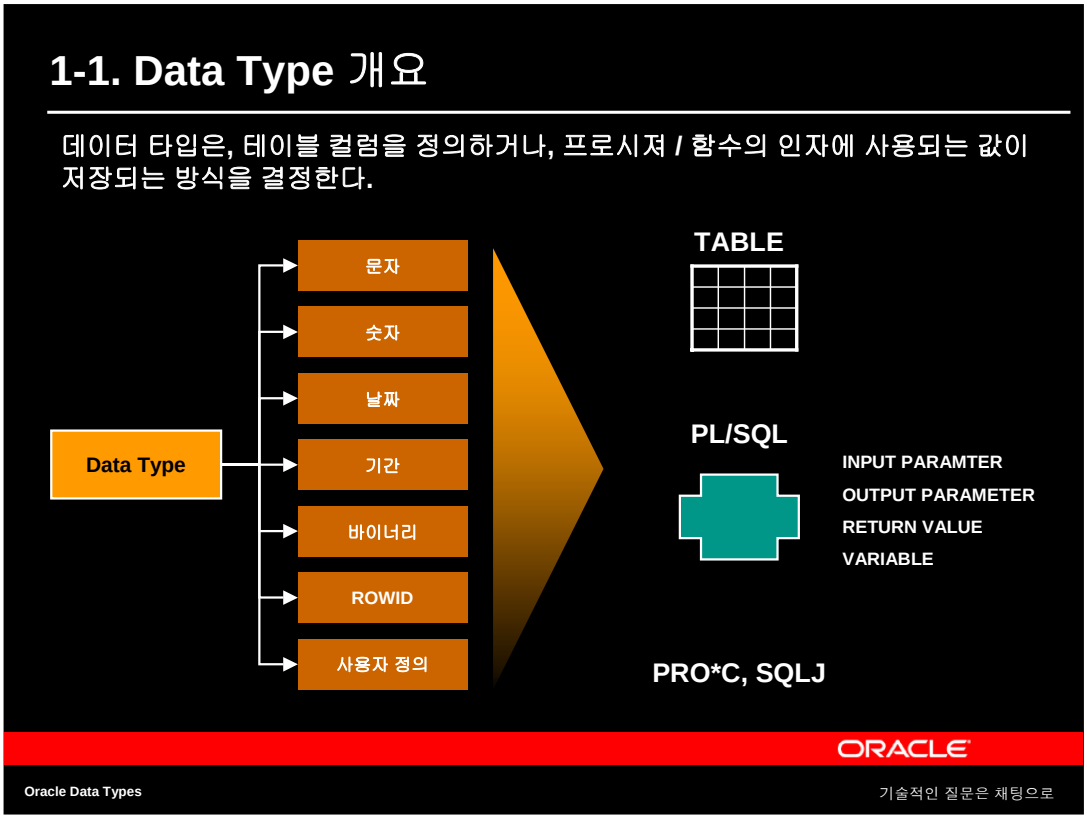
3장에서는 사용자 정의 데이터 타입에 대해 살펴보겠습니다. 사용자 정의 데이터 타입에는 Object Type, VARRAY, NESTED TABLE 및 REF가 있습니다.

4장에서는 Data Type선정에 도움이 될 수 있는 일반적인 지침에 대해서 말씀을 드리겠습니다.

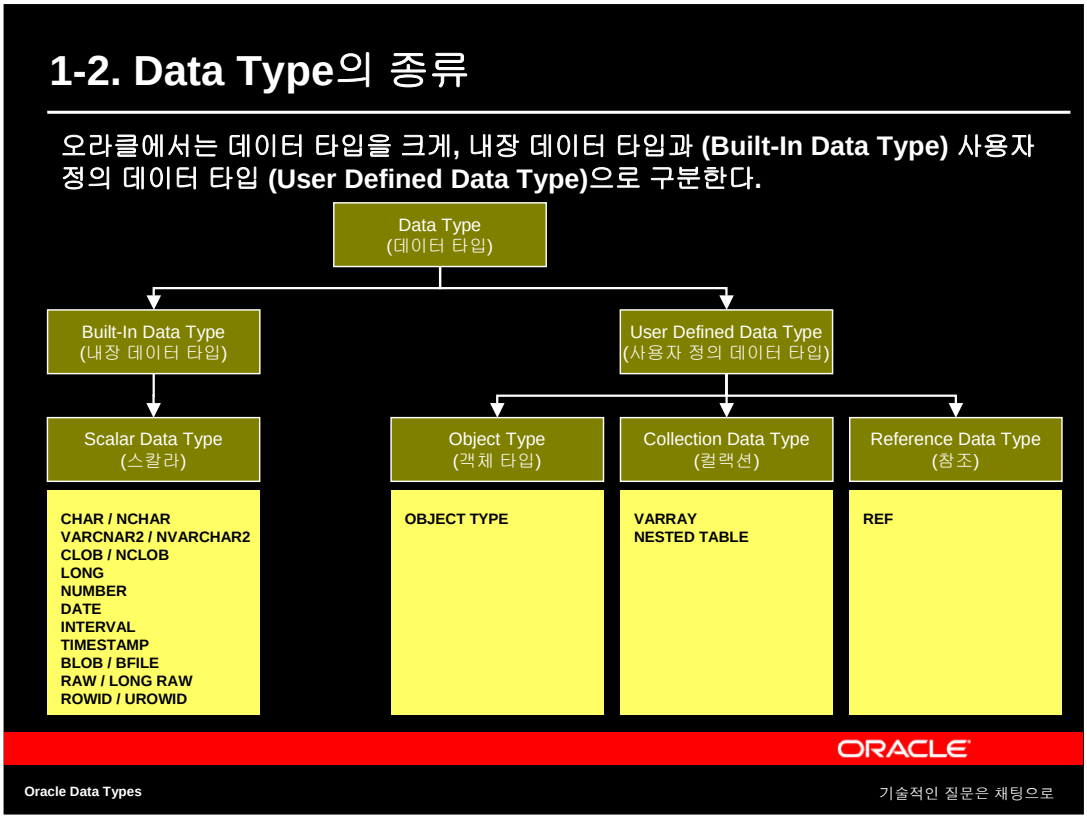
5장에서는 본 세미나에서 참조한 자료에 대해서 소개 드리겠습니다.



1장의 Overview에서는 Data Type 개요, Data Type 종류, Data Type 이해의 중요성에 대해 알아 보겠습니다.



데이터 타입은, 테이블 컬럼을 정의하거나, 프로시져 / 함수의 인자에 사용되는 값이 저장되는 방식을 결정합니다. 오라클에서는, 데이터 타입에 따라 의미상 동일한 값이 다르게 처리 됩니다. 예를 들어, NUMBER 데이터 타입에 저장된 숫자에는 다른 숫자를 더하거나 뺄 수가 있지만, RAW 데이터 타입에 저장된 숫자 표현에 대해서는 더하거나 뺄 수가 없습니다.



오라클에서는 데이터 타입의 종류를 도식화 시켜보면 다음과 같습니다.

오라클의 데이터 타입은 크게 내장 데이터 타입 (Built-In Data Type)과 사용자 정의 데이터 타입 (User Defined Data Type)으로 구분하며, 내장 데이터 타입은 스칼라 Data Type 성격을 갖습니다.

내장 데이터 타입은 저장되는 데이터의 성격에 따라서 CHARACTER Data Type, NUMBER Data Type, DATE Data Type 및 BINARY Data Type으로 나누어 볼 수 있습니다.

CHARACTER Data Type에는 CHAR, NCHAR, VARCHAR2, NVARCHAR2,CLOB, NCLOB및 LONG Data Type이 해당합니다.

Oracle9i기준으로 숫자는 NUMBER Data Type만 제공됩니다.

날짜 및 시간과 관련된 Data Type으로 DATE 외에 INTERVAL DAY TO SECOND, INTERVAL YEAR TO MONTH 타입과 TIMESTAMP 및 TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE타입이 제공됩니다.

Binary Data Type은 BLOB, BFILE, RAW, LONG RAW Data Type이 있습니다.

한편 특정 Row를 나타내기 위해 ROWID및 UROWID Data Type을 사용할 수 있습니다.

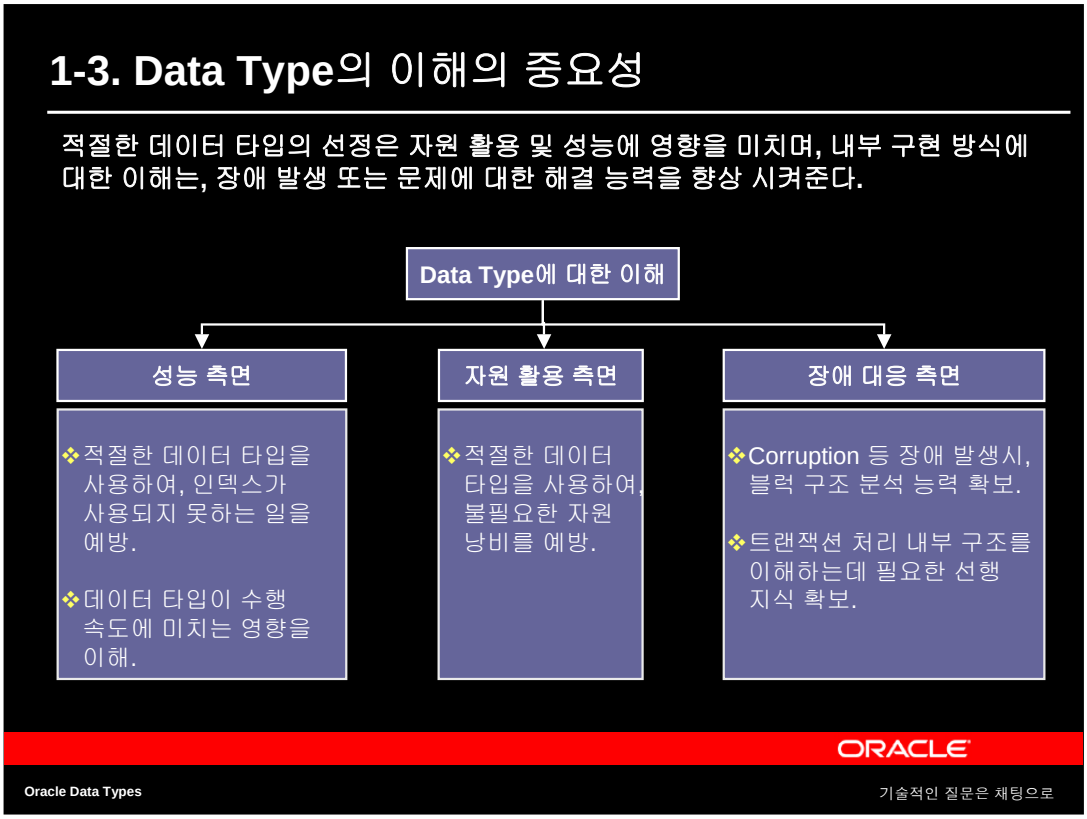
사용자 정의 데이터 타입은 내장 데이터 타입을 기반으로 하거나 다른 사용자 정의 데이터 타입을 기반으로 하여 사용자가 정의하는 Data Type입니다.

사용자 정의 데이터 타입은 OBJECT type, COLLECTION Data Type, REFERENCE Data Type으로 구분합니다.

Object Type은 데이터 타입과 이 데이터들을 조작하기 위한 함수 혹은 프로시져를 묶어서 사용자가 정의한 새로운 데이터 타입입니다.

Collection Data Type은 VARRAY와 NESTED TABLE이 해당하며, 다른 Object를 참조하는 Reference Data Type도 사용자 정의 데이터 타입에 해당합니다.

각 Data Type에 대해서는 하나 하나 말씀해 드리겠습니다.

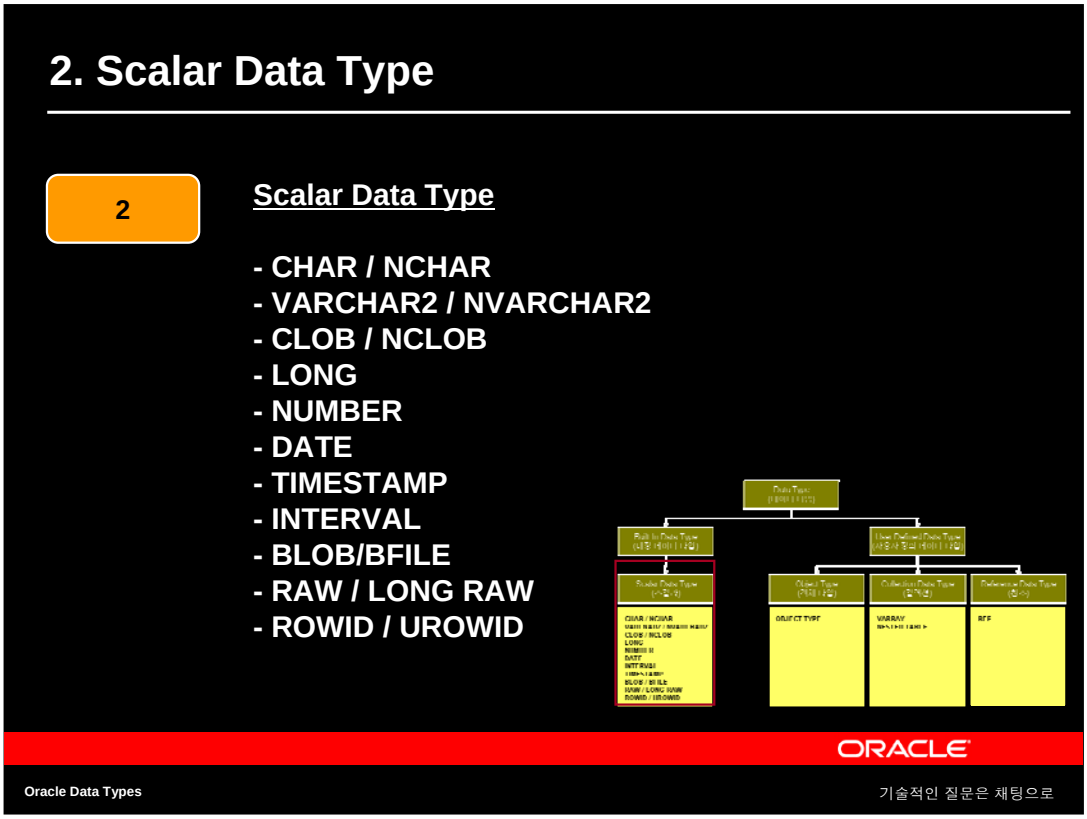


Oracle에서 제공하는 Data Type에 대한 이해는 적절한 Data Type의 선정에 도움이 되며,

성능측면에선, 적절한 Data Type을 사용하여, index를 사용할 수 없게 되는 것을 방지하고, 정확한 비교를 하지 못하는 경우를 예방할 수 있습니다. 즉 적절한 Data Type선정은 수행 속도 저하를 예방할 수 있습니다.

또한 자원 활용측면에선 , 적절한 Data Type을 사용하여 불필요한 자원 낭비를 예방할 수 있습니다.

끝으로 장애대응측면에선, Oracle Data Type에 따른 내부 저장방식을 이해하여 block corruption등 장애 발생시 block 구조에 대한 분석 능력을 확보하고, 트랜잭션 처리 내부 구조를 이해하는데 필요한 선행지식을 확보할 수 있습니다.

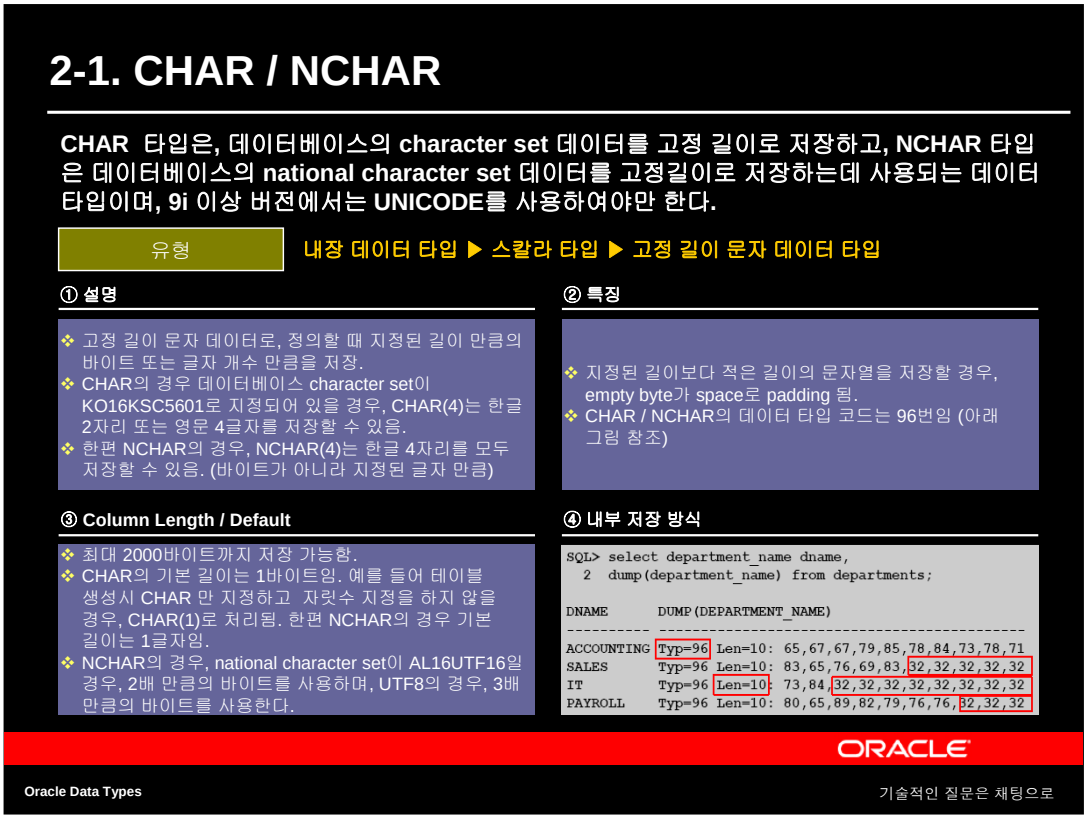


그럼 이제부터 각 Data Type에 대해 알아 보겠습니다.

먼저 내장 데이터 타입인 스칼라 데이터 타입을 화면의 순서대로 말씀을 드리겠습니다.

문자 데이터 타입에 속하는 CHAR/NCHAR, VARCHAR2/NVARCHAR2,CLOB/NCLOB, LONG 의 순으로 설명을 드리고, 숫자데이터 타입인 NUMBER 데이터 타입을 , 날짜 데이터 타입인 DATE, TIMESTAMP 및 INTERVAL에 대해서 말씀을 드리고 , 바이너리 데이터 타입인 BLOB, BFILE, RAW , LONG RAW순으로 설명해 드리겠습니다.

마지막으로 ROWID및 UROWID에 대해서 말씀을 드리겠습니다.



CHAR 타입은, 데이터베이스의 Character Set 데이터를 고정 길이로 저장하고, NCHAR 타입은 데이터베이스의 national character set 데이터를 고정길이로 저장하는데 사용되는 데이터 타입입니다.

National Character Set 데이터는 Oracle 9i 이상 버전에서는 UNICODE를 사용하여야만 합니다. 그래서 UTF80이나 AL16UTF16만 가능합니다.

CHAR나 NCHAR의 주된 특징은, 컬럼 정의 시 지정된 max length보다 적은 길이의 문자열을 저장하는 경우, Empty Byte가 SPACE로 추가(padding)된다는 것입니다. 이 Empty Byte는 경우에 따라서는 불필요한 자원 낭비 및 정확한 비교를 하는데 방해가 될 수 있으므로 Data Type의 특성을 정확히 알고 사용해야 합니다.

Char Data Type은 최대 2000byte까지 저장이 가능하며 , 자릿수를 지정하지 않을 경우 기본 길이는 1 바이트입니다.

NCHAR Data Type도 최대 2000byte까지 저장이 가능한데, 자릿수를 지정하지 않을 경우 한글자가 저장됩니다. 사용되는 national characterset에 따라 AL16UTF16의 경우엔 2 byte 를 , UTF8의 경우는 3 byte가 기본입니다.

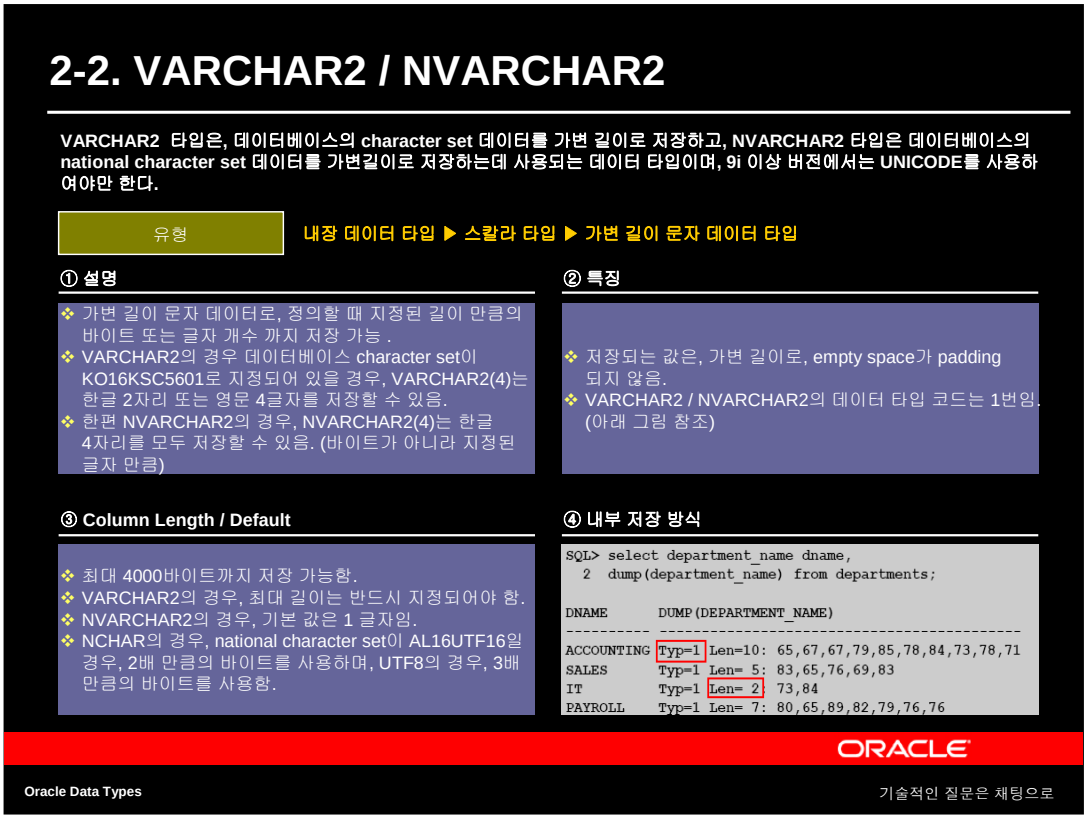
내부 저장방식은 화면의 4번과 같이 dump() function을 통해 확인이 가능합니다.

CHAR및 NCHAR의 내부 데이터 코드값은 96이며, 이는 dump결과 TYPE 에서 보여지는 값으로도 확인이 가능합니다.

Length는 INSERT 되는 데이터의 길이와 무관하게, 컬럼을 선언할 때 지정한 10byte를 모두 사용하고 있습니다.

실제로 저장된 문자 data는 각 문자의 ASCII CODE값으로 저장됩니다.

Dump결과 실 data를 제외한 나머지 부분은 ASCII CODE 32번인 SPACE가 들어 간 것을 확인하실 수 있습니다.



VARCHAR2 타입은, 데이터베이스의 character set 데이터를 가변 길이로 저장하고, NVARCHAR2 타입은 데이터베이스의 national character set 데이터를 가변 길이로 저장하는데 사용되는 데이터 타입입니다.

NVARCHAR2 타입도 NCHAR 타입과 같이 Oracle 9i 이상 버전에서는 UNICODE를 사용하여야만 합니다. 따라서 UTF80이나 AL16UTF16만 가능합니다.

VARCHAR2 나 NVARCHAR2 의 주된 특징은, 컬럼 정의 시 지정된 max length보다 적은 길이의 문자열을 저장하는 경우, Empty Byte가 SPACE로 추가 (padding) 되지 않고, 실제 data만 저장되는 가변 길이라는 것입니다. 그래서 CHAR/NCHAR에 비해 불필요한 자원 낭비를 막고, 테이블에 대한 Full Scan시에도 작은 블록을 읽어 성능상 장점이 있으며, Empty Block이 PADDING되지 않아 정확한 비교를 하는데 유의해야 할 사항이 적습니다.

VARCHAR2 Data Type은 최대 4000byte까지 저장이 가능하며 , 자릿수는 반드시 지정해야 합니다.

한편 NVARCHAR2 Data Type도 최대 4000byte까지 저장이 가능한데, 자릿수를 지정하지 않을 경우 한글자가 지정됩니다. 저장되는 National Character Set에 따라 AL16UTF16의 경우엔 2 BYTE, UTF8의 경우는 3BYTE가 기본입니다.

내부 저장방식을 dump() function으로 확인해 보면,

VARCHAR2 및 NVARCHAR2 의 내부 데이터 코드값은 1 입니다. 이 내부코드 값으로 Data Type에 대한 식별이 가능합니다.

Length는 실제 data를 insert한 양에 따라 유동적입니다. CHAR/NCHAR에 비해 불필요한 space가 추가되지 않았음을 확인할 수 있습니다.

실제로 저장된 문자data는 각 문자의 ASCII CODE값만 저장되었음을 확인이 하실 수 있습니다.

참고>

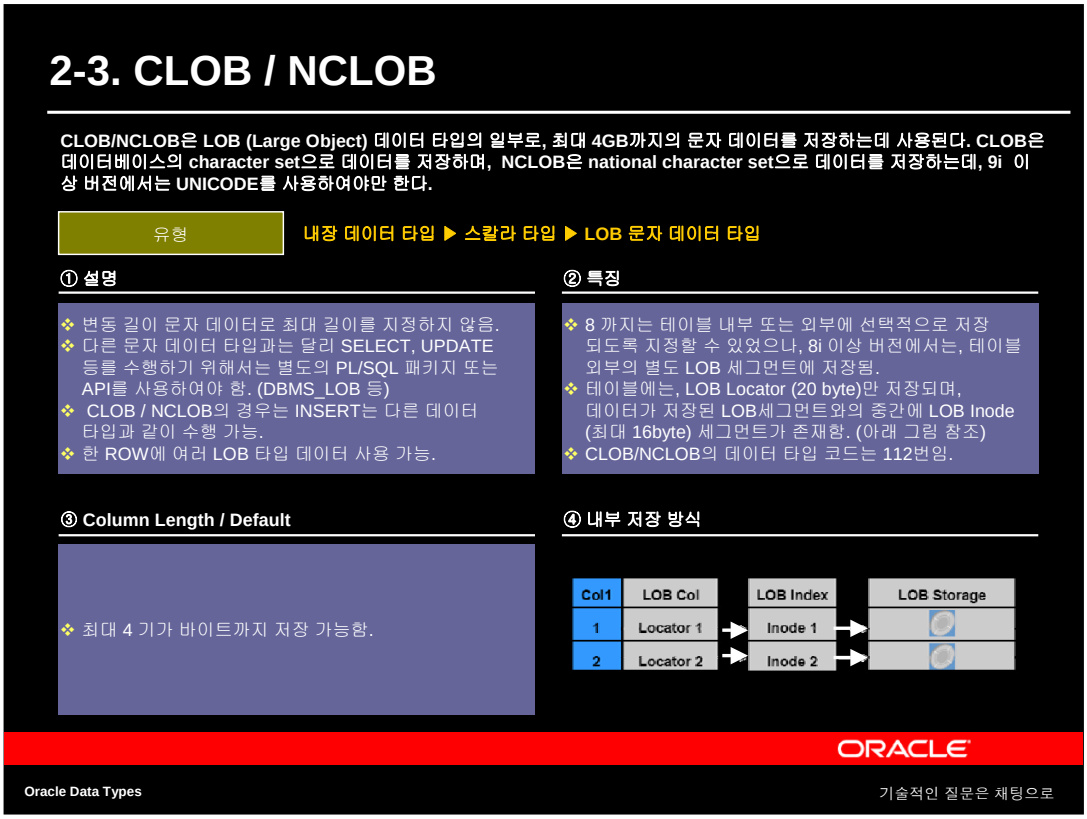
참고로 VARCHAR2와 NVARCHAR2의 dump결과 내부코드가 같으나, 구별은 table description으로 column에 정의된 Data Type

에 대한 확인이 가능하며, 또한 한글의 경우 저장된 byte로도 확인이 가능합니다.

예를 들어 DB characterset이 KO16KSC5601이고 national characterset이 UTF8인 경우,

“가”를 저장한다고 보면, varchar2 Data Type의 length는 2byte이나 nvarchar2 Data Type은 3byte로 보입니다.

물론 DB characterset과 national characterset이 같은 UTF8인 경우엔 column에 정의된 Data Type으로만 확인이 가능합니다.



CLOB/NCLOB은 LOB (Large Object) 데이터 타입의 일부로, 최대 4GB까지의 문자 데이터를 저장하는데 사용됩니다.

CLOB은 데이터베이스의 Character Set으로 데이터를 저장하는 반면, NCLOB은 National Character Set으로 데이터를 저장하는데, 9i 이상 버전에서는 UNICODE를 사용하여야만 합니다.

CLOB과 NCLOB는 charsetform에 의해 차이가 발생합니다.

CLOB/NCLOB는 변경길이 문자 데이터를 저장하며, 최대 길이를 지정하지 않습니다.

SQLPLUS에서는 4000 BYTE이하인 경우 SELECT, INSERT, UPDATE, DELETE작업이 가능하며, PL/SQL에서는 32k까지 가능합니다. 그래서 일반적으로는 Pro*C 또는 JAVA 언어를 사용하여 Oracle에서 제공하는 API를 이용한 프로그램을 작성합니다.

다른 문자 데이터 타입과는 달리 DBMS_LOB과 같은 전용 PL/SQL 패키지 및 API를 제공합니다.

한 ROW에 여러 CLOB/NCLOB Data Type을 사용할 수 있습니다.

Oracle8까지는 LOB INDEX가 TABLE 이 저장되는 곳과 동일한 내부 저장영역 (세그먼트)에 또는 외부에 선택적으로 저장 되도록 지정할 수 있었으나, Oracle8i이상 부터는 DML보다는 SELECT 위주로 인접한 블럭으로부터 데이터를 읽어올 수 있도록, LOB INDEX는 LOB 세그먼트가 같이 저장됩니다.

Internal LOB은 LOB locator, LOB Inode 즉 LOB index와 data인 lob segment로 구성됩니다.

CLOB/NCLOB의 내부 저장 코드는 112번입니다.

LONG 데이터 타입은, 하위 버전에 대한 호환성을 위해 존재하며, 역할은 CLOB / NCLOB과 유사하게 문자 데이터를 저장합니다. 차이점은, 별도의 세그먼트에 데이터가 존재하는 것이 아니라 테이블 내 다른 데이터와 함께 저장된다는 점입니다.

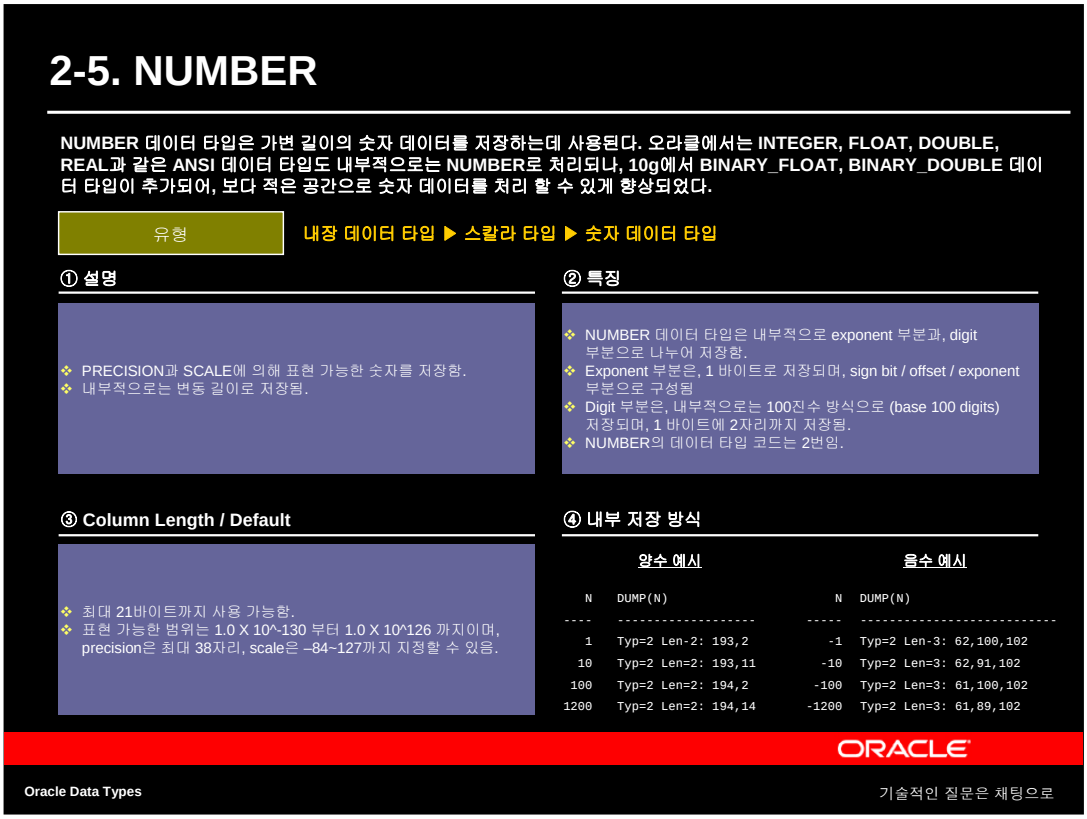
LONG Data Type도 변동 길이 문자 데이터 타입으로 최대 길이를 지정하지 않습니다.

한 ROW에는 한 개의 LONG Data Type만 사용이 가능합니다. 앞에서 설명 드린 CLOB에 비해 많은 제약 사항을 가집니다. 예를 들어 순차적 액세스만 허용하며, 사용자 정의 데이터 타입에 사용되지 못하고, replication에서도 사용할 수 없습니다. 자세한 사항은 데이터 타입 선정 부분에서 다시 소개해 드리겠습니다.

Table생성시 long column은 어느 위치에나 지정은 가능하나, 실제 Oracle database에선 성능상의 issue로 long column의 위치는 row의 맨 뒤에 저장합니다.

Long Data Type은 최대 2GB까지 저장이 가능하며, 내부 저장 방식은 다른 문자 데이터타입과 같이 데이터베이스의 character set에 따라 저장됩니다. Block dump 결과를 보면 각 문자에 대한 ascii code값으로 저장된 값에 대한 확인이 가능합니다. 내부 저장 코드는 8번입니다.

참고로 화면에 보여지는 결과는 dump() function을 사용한 것이 아니라 , alter system command로 block dump를 뜬 결과입니다.



NUMBER 데이터 타입은 가변 길이의 숫자 데이터를 저장하는데 사용됩니다.

오라클에서는 INTEGER, FLOAT, DOUBLE, REAL과 같은 ANSI 데이터 타입도 내부적으로는 NUMBER로 처리됩니다. (참고 : 10g에서 BINARY_FLOAT, BINARY_DOUBLE 데이터 타입이 추가되어, 보다 적은 공간으로 숫자 데이터를 처리 할 수 있게 향상되었습니다.)

Number Data Type은 PRECISION과 SCALE에 의해 표현 가능한 숫자를 정의할 수 있습니다.

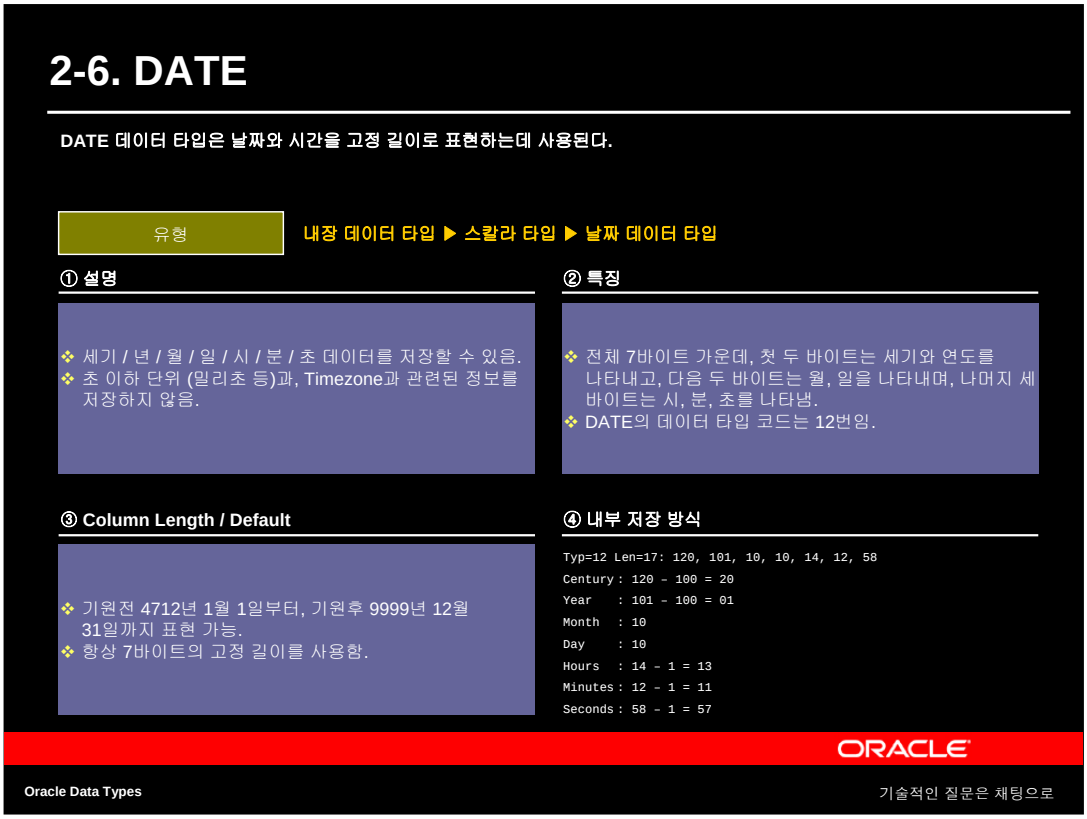
Column정의 시 number(8,2) 인 경우 전체 자리수인 precision은 8자리, 소수점 이하 몇자리 인지 정의하는 scale은 2자리가 됩니다.

최대 21bye까지 사용 가능하며, number Data Type을 통해 표현 가능한 범위는 1.0 X 10^-130 부터 1.0 X 10^126 까지이며, precision은 최대 38자리까지 scale은 –84에서 127까지 지정할 수 있습니다.

내부 저장되는 방식을 보면, Data Type은 내부적으로 exponent부분과 digit으로 나뉘어 집니다.

Dump결과를 보면 NUMBER의 데이터 타입 코드는 2번이고 , length와 숫자 data가 저장된 형태를 보여줍니다.

Number Data Type은 단순히 ASCII 또는 UNICODE로 저장하는 문자열 데이터 타입과는 달리 Oracle의 내부 알고리즘에 따라 계산하여 저장되며, 상세한 내용은 오라클 교육 센타의 별도의 DSI강의를 통해서 접하실 수 있습니다.



DATE 데이터 타입은 날짜와 시간을 고정 길이로 표현하는데 사용됩니다.

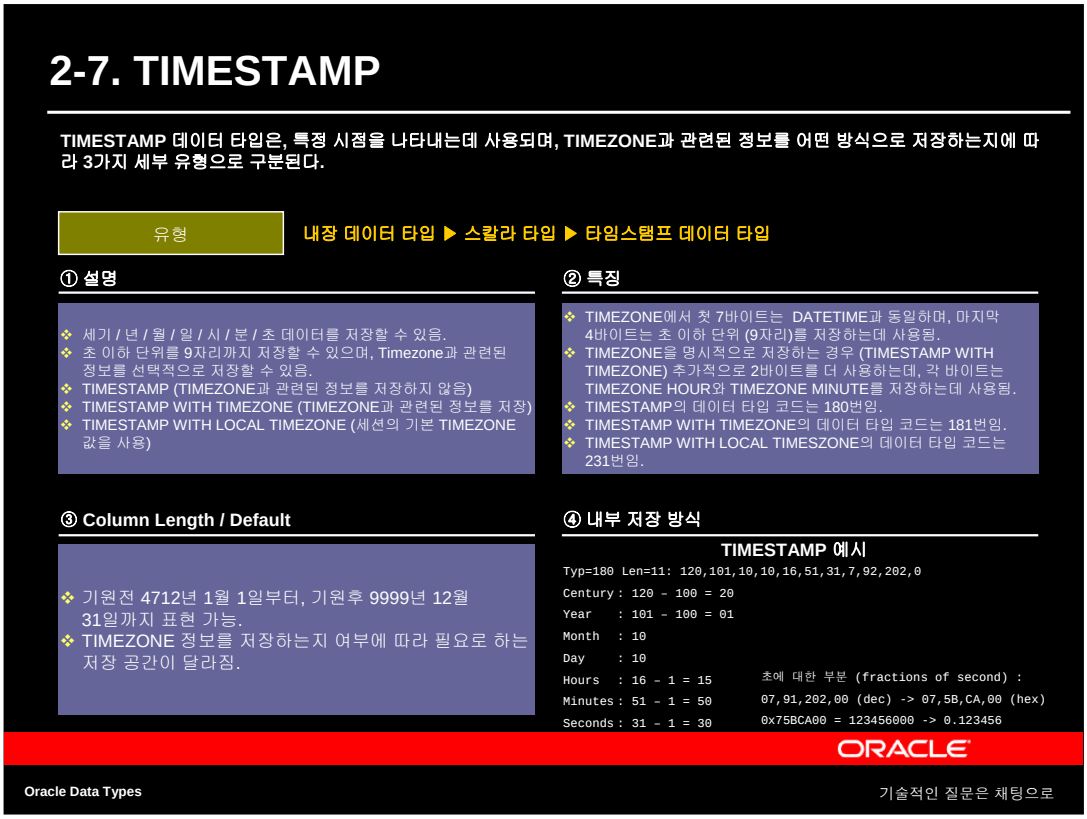
세기 / 년 / 월 / 일 / 시 / 분 / 초 데이터를 저장할 수 있으나 초 이하 단위 (밀리초 등)와 Timezone과 관련된 정보를 저장하지 않습니다.

Date DATA TYPE은 7byte 고정 길이로 저장되는데, 첫 두 바이트는 세기와 연도를 나타내고, 다음 두 바이트는 월, 일을 나타내며, 나머지 세 바이트는 시, 분, 초를 나타냅니다.

표현 가능한 범위는 기원전 4712년 1월 1일부터, 기원후 9999년 12월 31일까지입니다.

내부 저장방식을 보면 화면과 같습니다. 내부 type code는 12번이며, length는 언제나 7 byte입니다. 데이터는 표현하고자 하는 세기, 연도, 월,일,시,분,초로 구성되어 있습니다.

각각의 BYTE의 내용은 마찬가지로 Oracle의 내부 알고리즘에 의해 계산되어 저장됩니다.



TIMESTAMP 데이터 타입은, 특정 시점을 나타내는데 사용되며, TIMEZONE과 관련된 정보를 어떤 방식으로 저장하는지에 따라 3가지 세부 유형으로 구분됩니다.

먼저 timestamp Data Type을 보면 세기 / 년 / 월 / 일 / 시 / 분 / 초 데이터를 저장할 수 있습니다.

초 이하 단위를 9자리까지 저장할 수 있으며 (나노초), Timezone과 관련된 정보를 선택적으로 저장할 수 있습니다.

TIMESTAMP는 TIMEZONE과 관련된 정보를 저장하지 않는 반면. TIMESTAMP WITH TIMEZONE 은 database timezone과 관련된 정보를 저장합니다.한편 TIMESTAMP WITH LOCAL TIMEZONE 은 세션의 기본 TIMEZONE 값을 사용합니다. 즉 client timezone에 의해서 보여집니다.

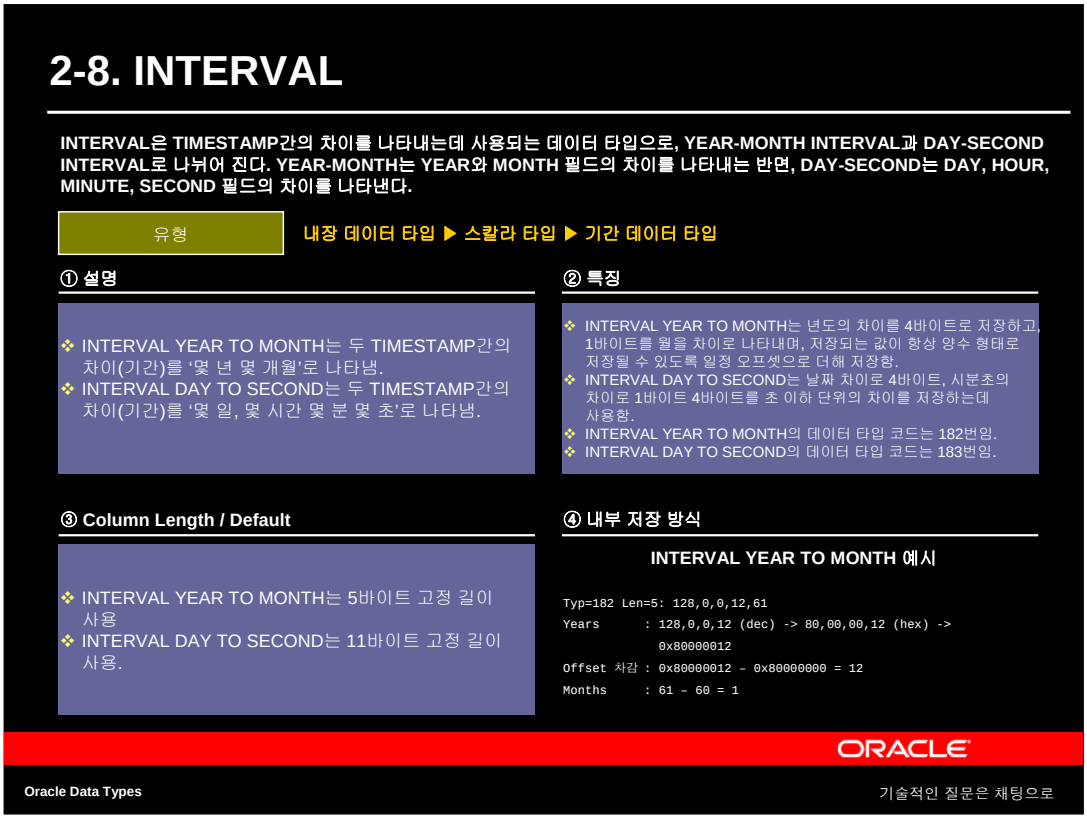
TIMEZONE 정보를 저장하는지 여부에 따라 필요로 하는 저장 공간이 달라집니다.

우선 TIMEZONE은 11byte의 고정길이로, 첫 7바이트는 DATE 타입과 동일하고, 마지막 4바이트는 초 이하 단위 (9자리)를 저장하는데 사용됩니다.

한편 TIMESTAMP WITH TIMEZONE은 TIMEZONE을 명시적으로 저장하는 경우 사용되는데, 필요한 공간은 13byte의 고정길이입니다. 첫 7바이트는 date type과 동일하고, 4 byte는 초 이하 단위를 마지막 2byte는 TIMEZONE HOUR와 TIMEZONE MINUTE를 저장하는데 사용됩니다.

TIMESTAMP WITH LOCAL TIMEZONE 은 11byte 고정길이이며, 첫 7바이트는 date type과 동일하며, 나머지 4 byte는 fractional second를 저장합니다.

TIMESTAMP의 데이터 타입 코드는 180번, TIMESTAMP WITH TIMEZONE의 데이터 타입 코드는 181번, TIMESTAMP WITH LOCAL TIMESZONE의 데이터 타입 코드는 231번이며, Oracle에서 내부적으로 사용하는 알고리즘에 의해 계산되어 값이 저장됩니다.



INTERVAL은 TIMESTAMP간의 차이를 나타내는데 사용되는 데이터 타입으로, YEAR-MONTH INTERVAL과 DAY-SECOND INTERVAL로 나뉘어 집니다.

YEAR-MONTH는 YEAR와 MONTH 필드의 차이를 나타내는데, 두 timestamp간의 차이(기간)를 '몇 년 몇 개월'로 나타냅니다.

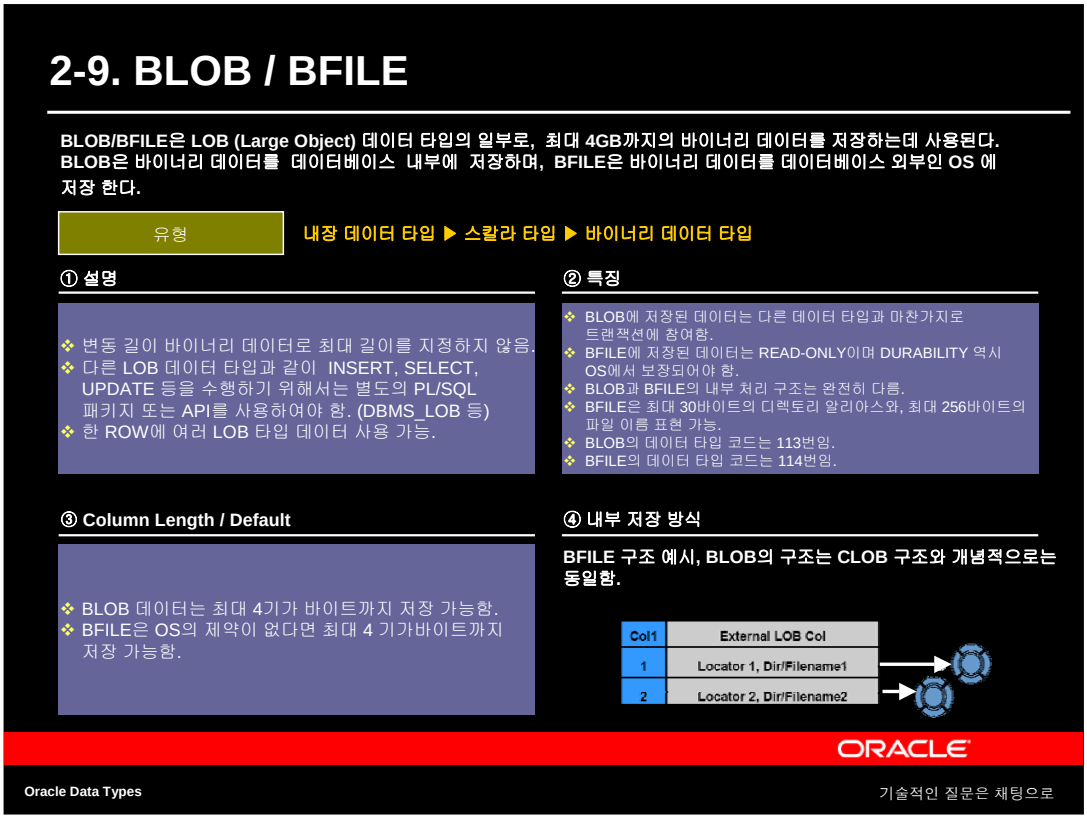
반면, DAY-SECOND는 DAY, HOUR, MINUTE, SECOND 필드의 차이를 나타내는데, 두 TIMESTAMP간의 차이(기간)를 '몇 일, 몇 시간 몇 분 몇 초'로 나타냅니다.

INTERVAL YEAR TO MONTH는 5byte 고정 길이로, 년도의 차이를 4바이트로 저장하고, 1바이트를 월을 차이로 나타내며, 저장되는 값이 항상 양수 형태로 저장될 수 있도록 일정 오프셋을 더해 저장합니다.

INTERVAL DAY TO SECOND는 11byte고정 길이로 , 날짜 차이로 4바이트, 시분초의 차이로 1바이트,나머지 4바이트를 초 이하 단위의 차이를 저장하는데 사용합니다.

Dump를 떠 보면 INTERVAL YEAR TO MONTH의 데이터 타입 코드는 182번이며,

INTERVAL DAY TO SECOND는 183번입니다.



BLOB/BFILE은 LOB (Large Object) 데이터 타입의 일부로, 최대 4GB까지의 바이너리 데이터를 저장하는데 사용됩니다.

BLOB은 바이너리 데이터를 데이터베이스 내부에 저장하며, BFILE은 바이너리 데이터를 데이터베이스 외부인 OS상의 파일 시스템에 저장 합니다. OS 의 제약이 없다면 최대 4GB까지 저장이 가능합니다.

BLOB/BFILE type은 변동 길이 바이너리 데이터로 최대 길이를 지정하지 않으며,

다른 LOB 데이터 타입과 같이 INSERT, SELECT, UPDATE 등을 수행하기 위해서는 별도의 PL/SQL 패키지 또는 API를 사용하여야 합니다.

한 ROW에 여러 BLOB/BFILE 타입을 사용이 가능합니다.

BLOB에 저장된 데이터는 다른 데이터 타입과 마찬가지로 트랜잭션에 의해 변경이 가능합니다.

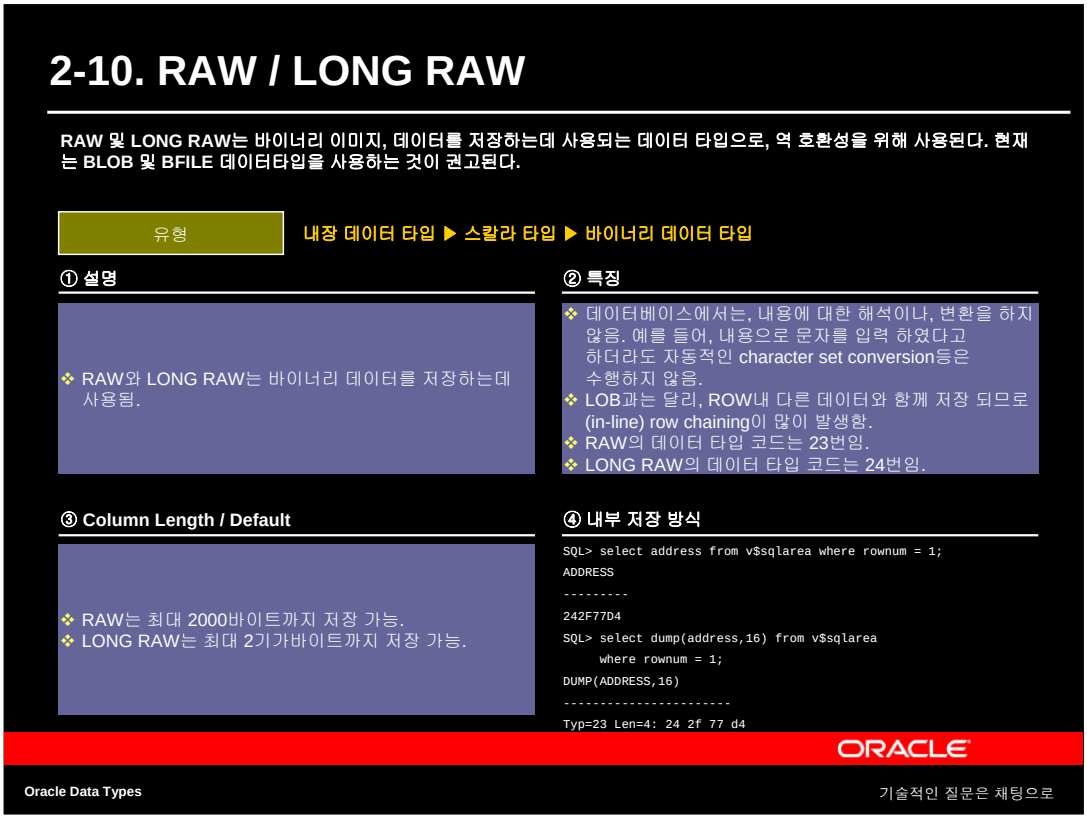
그러나 BFILE에 저장된 데이터는 READ-ONLY이며, 변경이 불가합니다. Bfile에 대한 무결성(integrity) , 보안(security), 지속성(DURABILITY)은 OS에서 보장되어야 합니다.

그래서 BLOB과 BFILE의 내부 처리 구조는 완전히 다릅니다.

BLOB Data Type의 내부 저장 방식은 CLOB Data Type과 유사하며, 저장되는 data성격만 바이너리 데이터를 저장합니다.

BFILE은 최대 30바이트의 디렉토리 알리아스와, 최대 256바이트의 파일 이름 표현이 가능합니다.

Dump를 떠 보면 BLOB의 데이터 타입 코드는 113번이며, BFILE의 데이터 타입 코드는 114번입니다.



RAW 및 LONG RAW는 바이너리 이미지 데이터를 저장하는데 사용되는 데이터 타입으로, 현재는 역 호환성을 위해 사용됩니다.

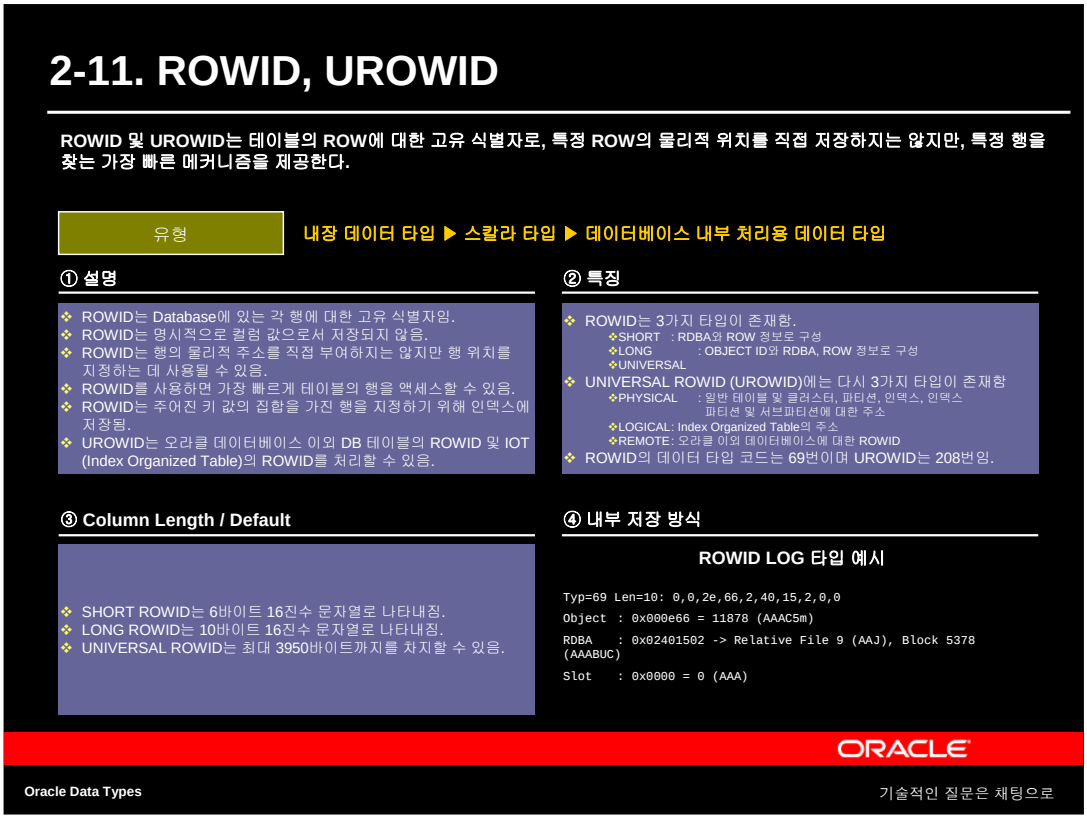
현재는 RAW 나 LONG RAW 대신 BLOB 및 BFILE 데이터타입을 사용하는 것을 권고 합니다.

RAW는 최대 2000바이트까지 저장 가능하며, LONG RAW는 최대 2기가바이트까지 저장 가능합니다.

데이터베이스에서는 SQL에 의해 내용에 대한 해석이나 변환을 하지 않습니다. 예를 들어, 내용으로 문자를 입력 하였다고 하더라도 자동적인 character set conversion등은 수행하지 않습니다.

LOB과는 달리, ROW내 다른 데이터와 함께 저장 되므로 (in-line) row chaining이 많이 발생합니다.

Dump 결과 RAW의 데이터 타입 코드는 23번임을 알 수 있으며, LONG RAW의 데이터 타입 코드는 24번입니다.



ROWID 및 UROWID는 테이블의 ROW에 대한 고유 식별자로, 특정 ROW의 물리적 위치를 직접 저장하지는 않지만, 특정 행을 찾는 가장 빠른 메커니즘을 제공합니다.

ROWID는 Database에 있는 각 행에 대한 고유 식별자이며, 명시적으로 컬럼 값으로서 저장되지 않습니다.

ROWID는 행의 물리적 주소를 직접 부여하지는 않지만 행 위치를 지정하는 데 사용될 수 있으며,

ROWID를 사용하면 가장 빠르게 테이블의 행을 액세스할 수 있습니다.

ROWID는 주어진 키 값의 집합을 가진 행을 지정하기 위해 인덱스에 저장됩니다.

UROWID는 오라클 데이터베이스 이외 DB 테이블의 ROWID 및 IOT (Index Organized Table)의 ROWID를 처리할 수 있습니다.

ROWID는 3가지 타입이 존재하는데, SHORT, LONG, UNIVERSAL로 구분할 수 있습니다.

SHORT ROWID는 Restricted Rowid라고도 하며, Oracle7 이하와의 역 호환성을 위해 존재합니다. 4 byte의 RDBA와 2byte의 ROW 정보로 구성됩니다.

LONG rowid는 extended rowid라고도 불리며,4byte의 OBJECT ID와 4byte의 RDBA와 2byte의 ROW 정보로 구성됩니다.

UNIVERSAL rowid는 다시 3가지 type으로 구분됩니다.

PHYSICAL : 일반 테이블 및 클러스터, 파티션, 인덱스, 인덱스파티션 및 서브파티션에 대한 주소

LOGICAL(primary key based) : Index Organized Table의 주소

REMOTE(foreign) : 오라클 이외 데이터베이스에 대한 ROWID

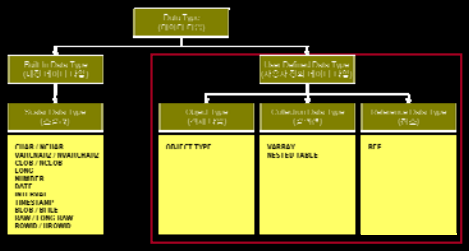
ROWID의 데이터 타입 코드는 69번이며 UROWID는 208번입니다.

3. User Defined Data Type

3

User Defined Data Type

- OBJECT TYPE
- VARRAY
- NESTED TABLE
- REF



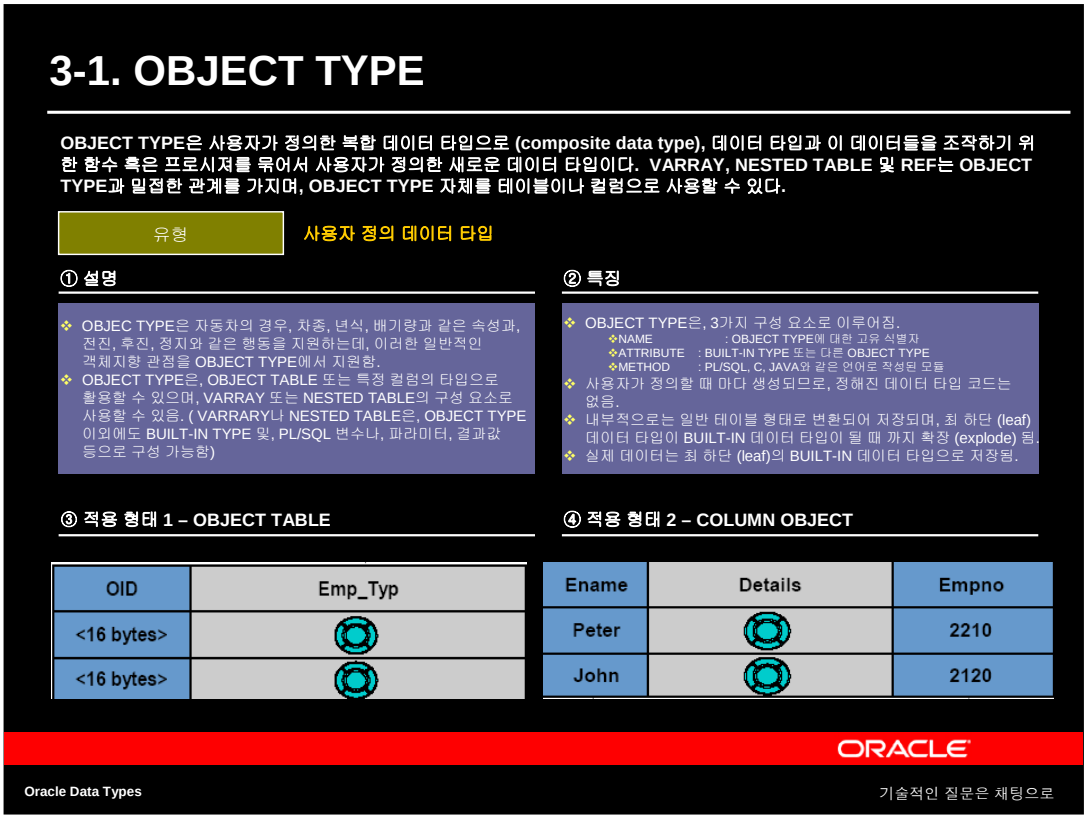
ORACLE

Oracle Data Types

기술적인 질문은 채팅으로

3 장에서는 사용자정의 데이터 타입에 대해서 알아 보겠습니다.

사용자정의 데이터 타입에는 object type, varray, nested table및 reference type이 있으며 각각 순서대로 소개 드리겠습니다.



OBJECT TYPE은 사용자가 정의한 복합 데이터 타입으로 (composite data type), 데이터 타입과 이 데이터들을 조작하기 위한 함수 혹은 프로시져를 묶어서 사용자가 정의한 새로운 데이터 타입입니다.

VARRAY, NESTED TABLE 및 REF는 OBJECT TYPE과 밀접한 관계를 가지며, OBJECT TYPE 자체를 테이블이나 컬럼으로 사용할 수 있습니다.

OBJEC TYPE은 자동차의 경우, 차종, 년식, 배기량과 같은 속성과, 전진, 후진, 정지와 같은 행동을 지원하는데, 이러한 일반적인 객체지향 관점을 OBJECT TYPE에서 지원합니다.

OBJECT TYPE은, OBJECT TABLE 또는 특정 컬럼의 타입(object column)으로 활용할 수 있으며, VARRAY 또는 NESTED TABLE의 구성 요소로 사용할 수 있습니다.

OBJECT TYPE은, 3가지 구성 요소로 이루어 집니다.

NAME : OBJECT TYPE에 대한 고유 식별자

ATTRIBUTE : BUILT-IN TYPE 또는 다른 OBJECT TYPE

METHOD : PL/SQL, C, JAVA와 같은 언어로 작성된 모듈

내부적으로는 일반 테이블 형태로 변환되어 저장되며, 최 하단 (leaf) 데이터 타입이 BUILT-IN DATA TYPE이 될 때 까지 확장 (explode) 됩니다.

Object type은 사용자가 정의할 때 마다 생성되므로, 정해진 데이터 타입 코드는 없으며, 단 object type을 펼쳐보면 최하단은 내장 데이터 타입으로 되어 있으며 , 이에 대한 dump결과는 확인이 가능합니다.

Object type의 선언 및 사용방법에 대한 보다 상세한 내용은 Oracle9i Concept및 Application Developer's Guide를 참조하여 주시기 바랍니다

참고>

Object table을 예로 보면, 맨 끝 가지 레벨의 scalar는 (leaf-level scalar) 테이블의 컬럼에 대응합니다. 그리고 시스템에서 자동으로 생성한 2개 컬럼이 테이블에 추가됩니다.

SYS_NC_OID$: 테이블의 row와 연관된 object ID

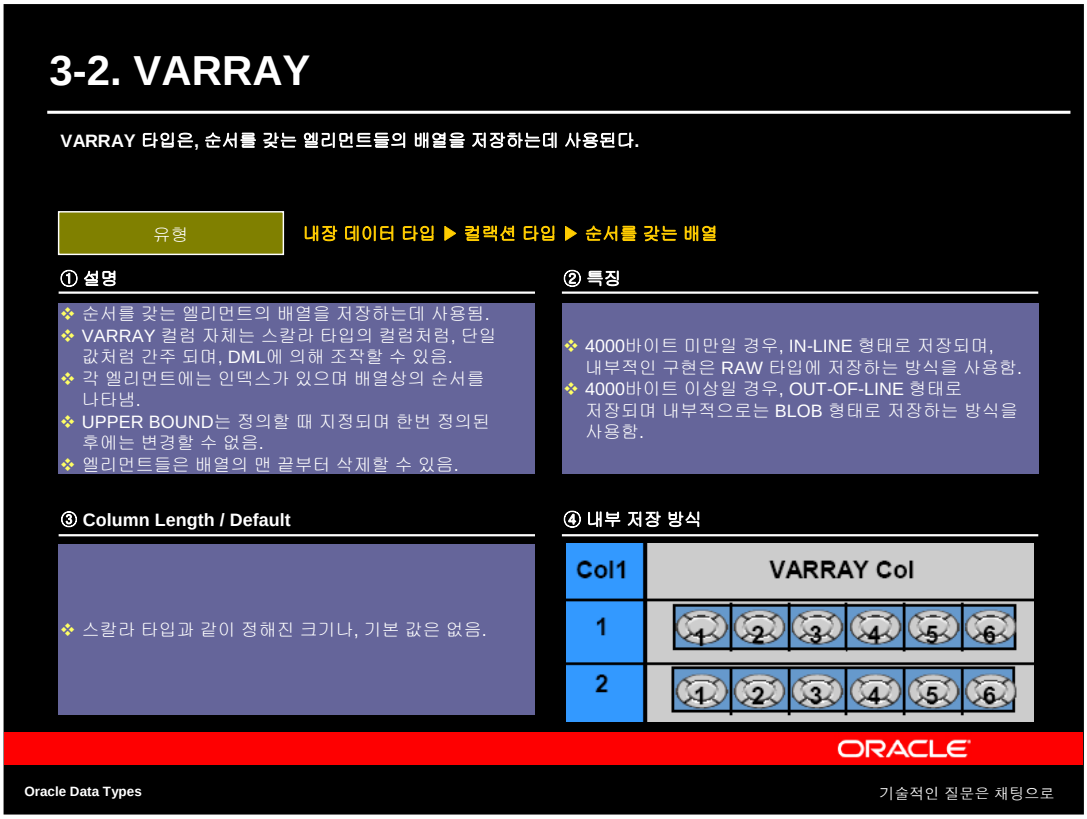
SYS_NC_OWINFO$: 각 row의 constructor를 보여 주기 위한 pseudo column

이 pseudo column이 테이블의 저장에 공간을 차지 하지는 않으나 (아래 블럭 덤프 참조) object에 대한 상세한 정보를 얻기 위해 이 컬럼을 쿼리에 사용할 수 있습니다.

```
SQL> select sys_nc_oid$ from employees1;
SYS_NC_OID$
----------------------------------------
5DCC862136C2484CE0340800207ECAE5
SQL> select sys_nc_rowinfo$ from employees1;
SYS_NC_ROWINFO$ (EMPNO, ENAME)
----------------------------------------
EMP_TYP(1, 'peter')
```

덤프 결과

```
--------
col 0 : [16] 5d cc 86 21 36 c2 48 4c e0 34 08 00 20 7e ca e5   <- OID
col 1 : [ 2] c1 02                                            <- Empno
col 2 : [ 5] 70 65 74 65 72                                   <- Ename
```



VARRAY 타입은, 순서를 갖는 엘리먼트들의 1차원 배열을 저장하는데 사용됩니다.

VARRAY 컬럼 자체는 스칼라 타입의 컬럼처럼, 단일 값처럼 간주 되며, DML에 의해 조작할 수 있습니다. 각 엘리먼트에는 인덱스가 있으며 배열상의 순서를 나타냅니다. UPPER BOUND는 정의할 때 지정되며 한번 정의된 후에는 변경할 수 없습니다. 엘리먼트들은 배열의 맨 끝부터 삭제할 수 있습니다.

4000바이트 미만일 경우, IN-LINE 형태로 저장되며, 내부적인 구현은 RAW 타입에 저장하는 방식을 사용합니다. 4000바이트 이상일 경우, OUT-OF-LINE 형태로 저장되며 내부적으로는 BLOB 형태로 저장하는 방식을 사용합니다.

스칼라 타입과 같이 정해진 크기나, 기본 값은 없습니다, 단, 가장 하단의 내장 데이터 타입에 대한 제약만 있으며, 생성시 array의 최대 개수에 대한 제약은 있습니다.

Varray에 대해 Dump를 떠서 보면, 총길이, 저장된 element 개수 및 size와 각 값에 대한 확인이 가능합니다.

참고>

```
Create or replace type person_typ as object
(a number, b varchar2(10));

Create or replace type v_person_typ as varray(5);

Create table person
(col1 number,
  col2 varchar2(10).
  col3 v_person_typ)
 varry col3 store as lob person_tab;
```

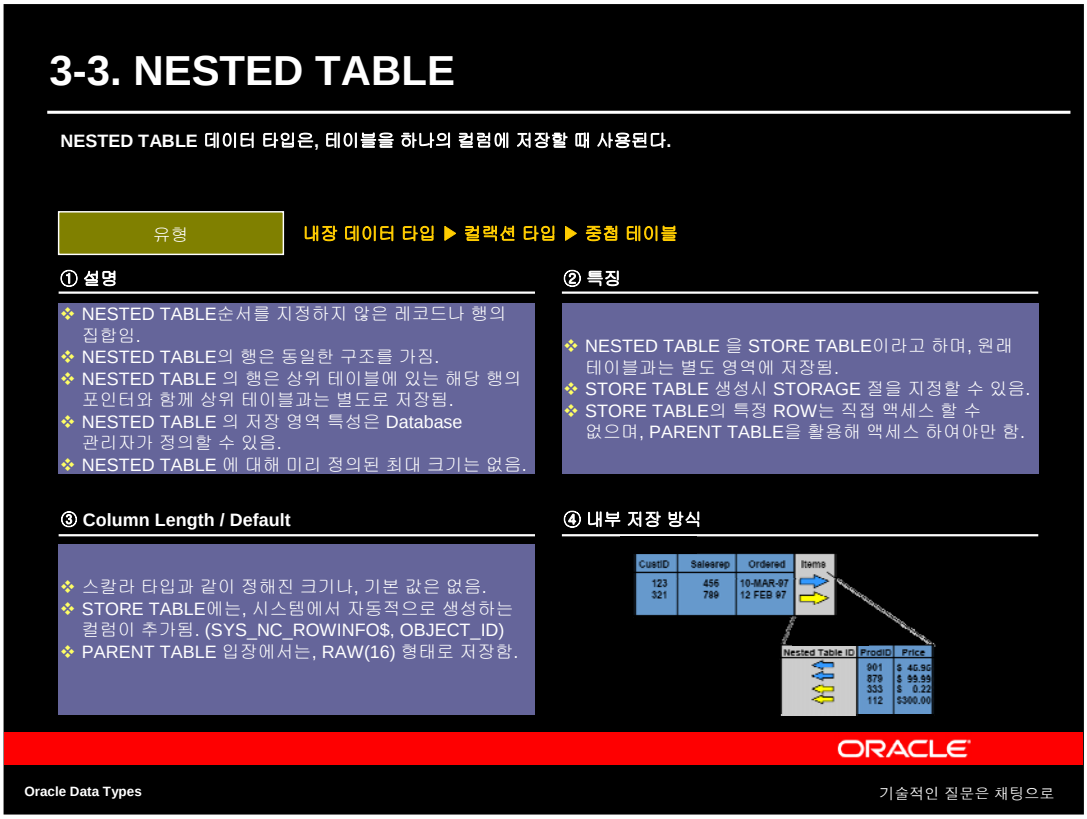
이 덤프는, person_typ에 대한 VARRAY(5)에 대한 내용으로, person_typ은 두개 attribute로 정의 되어 있다: a NUMBER, b VARCHAR2(10)

col 3 : [32] 88 01 20 01 01 00 02

0b 84 01 0b 02 c1 02 04 70 61 75 6c

0c 84 01 0c 02 c1 03 05 70 65 74 65 72

Byte 1 – 6	VARRAY overhead (byte 3(20) -> 전체 VARRAY column(32 bytes))
Byte 7 (02)	VARRAY의 element의 갯수(2 person_typ elements)
Byte 8 (0b)	첫번째 array element의 길이 (11 bytes)
Byte 9 – 11	Object overhead(84 01 0b)
Byte 12 – 14	Length (02) + data (c1,02) -> 1
Byte 15 – 19	Length (04) + data (70,61,75,6c) -> paul
Byte 20 (0c)	두번째 element의 길이 (12 bytes)
Byte 21 – 23	Object overhead(84 01 0c)
Byte 24 – 26	Length (02) + data (c1,03) -> 2
Byte 27 – 32	Length (05) + data (70,65,74,65,72) -> peter



NESTED TABLE 데이터 타입은, 테이블을 하나의 컬럼에 저장할 때 사용됩니다. Varray와 같이 upper bound값을 미리 지정하지 않으며 element최대 개수에 제한이 없는 가변 길이 array입니다.

NESTED TABLE은 순서를 지정하지 않은 레코드나 행의 집합이며, 행은 동일한 구조를 가집니다. NESTED TABLE 의 행은 상위 테이블을 가리키는포인터와 함께 저장되며,상위 테이블과는 별도 세그먼트에 저장됩니다. 또한 저장 영역 특성은 Database 관리자가 정의할 수 있으며 미리 정의된 최대 크기는 없습니다.

NESTED TABLE 을 STORE TABLE이라고 하며, 원래 테이블과는 별도 영역에 저장됩니다., STORE TABLE 생성시 STORAGE 절을 지정할 수 있습니다. STORE TABLE의 특정 ROW는 직접 액세스 할 수 없으며, PARENT TABLE을 활용해 액세스 하여야만 합니다.

STORE TABLE에는, 시스템에서 자동적으로 생성하는 컬럼이 추가됩니다.

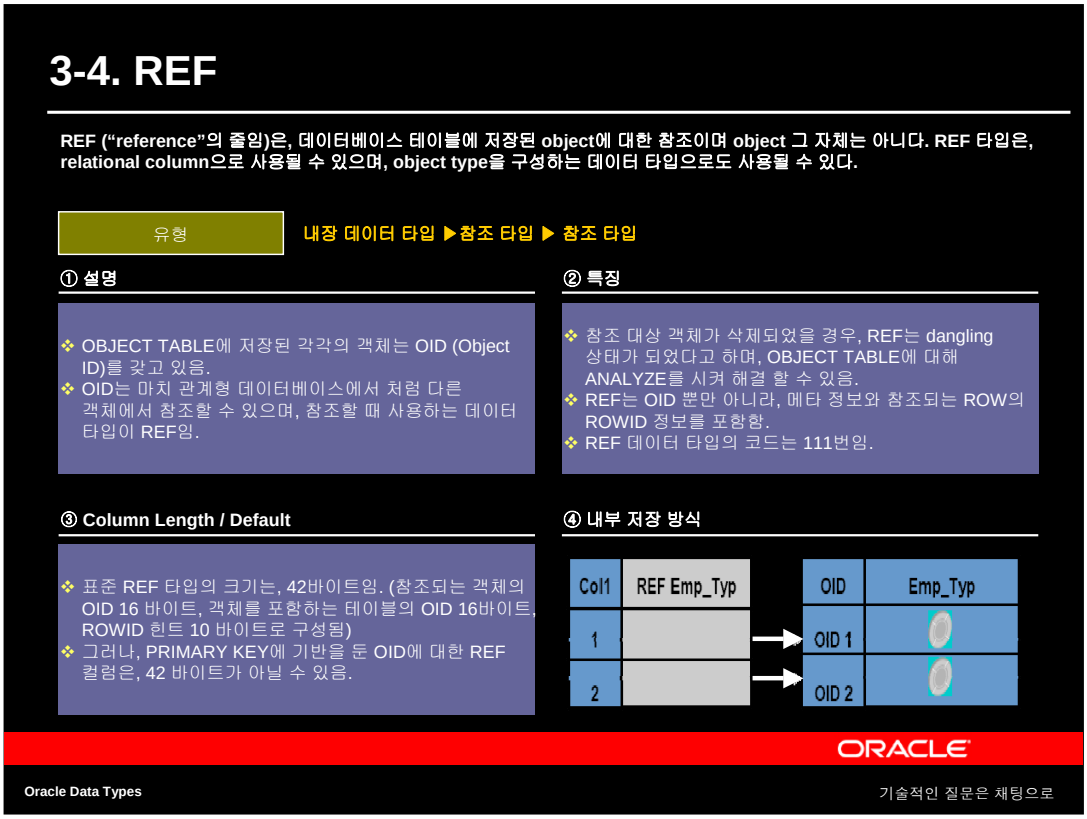
SYS_NC_ROWINFO$는 nested table의 row를 구성하는 object type정보를 저장하며, OBJECT_ID는 nested table의 id를 가집니다.

PARENT TABLE 입장에서는, RAW(16) 형태로 nested table의 id를 저장합니다.

참고 >

store table 내의 row는 직접 엑세스 불가능합니다. 다음은 그 예입니다.

```
SQL> create table storage (
  2  salesman number (4),
  3  elem_id number (6),
  4  ordered date,
  5  items elements_tab)
  6  nested table items store as items_tab;
Table created
SQL> insert into storage
  2  values(100,123456,SYSDATE,
  3  ELEMENTS_TAB (ELEMENTS(175692,120.12),
  4                ELEMENTS(167295,130.45),
  5                ELEMENTS(127569,99.99)));
1 row created
SQL> select * from items_tab;
select * from items_tab
              *
ERROR at line 1 :
ORA-22812: cannot reference nested table column's storage table
```



REF (“reference”의 줄임)은, 데이터베이스 테이블에 저장된 object에 대한 참조이며 object 그 자체는 아닙니다.

REF 타입은, relational column으로 사용될 수 있으며, object type을 구성하는 데이터 타입으로도 사용될 수 있습니다.

OBJECT TABLE에 저장된 각각의 객체는 OID (Object ID)를 갖고 있습니다.

OID는 마치 관계형 데이터베이스에서처럼 다른 객체에서 참조할 수 있는데, 참조할 때 사용하는 데이터 타입이 REF입니다.

참조 대상 객체가 삭제되었을 경우, REF는 dangling 상태가 되었다고 하며, OBJECT TABLE에 대해 ANALYZE를 시켜 해결 할 수 있습니다.

REF는 OID 뿐만 아니라, 메타 정보와 참조되는 ROW의 ROWID 정보를 포함합니다.

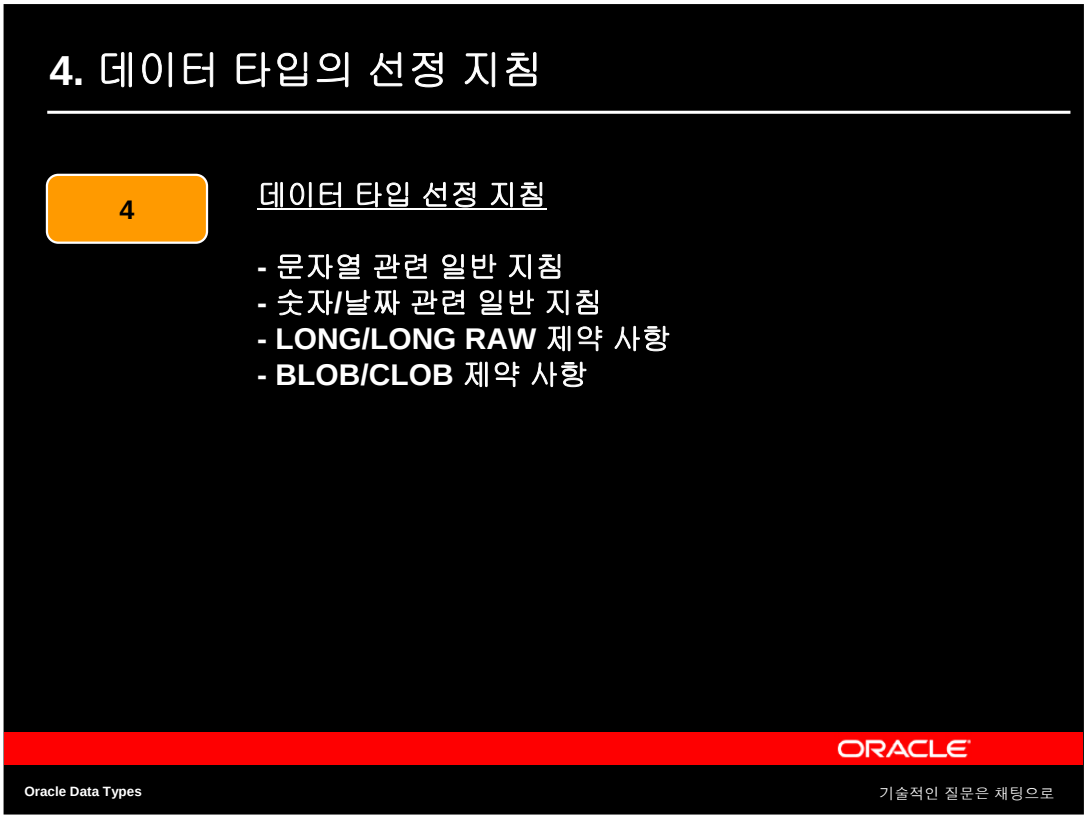
표준 REF 타입의 크기는, 42바이트로 참조되는 객체의 OID 16 바이트, 객체를 포함하는 테이블의 OID 16바이트, ROWID 힌트 10 바이트로 구성됩니다. 그러나, PRIMARY KEY에 기반을 둔 OID에 대한 REF 컬럼은, 42 바이트가 아닐 수 있습니다.

참고>

예를 들어 department table이 두개 컬럼으로 구성되어 있을 경우 : dname(VARCHAR2), mgr(REF EMP_TYP) 테이블에 한 개 row를 추가한 후에는 다음과 같은 저장 특성을 볼 수 있습니다.

```
SQL> select dump (mgr, 16) from department;
DUMP(MGR,16)
------------------------------------
Typ=111 Len=36: 0,22,2,8,5d,cc,86,21,36,c2,48,4c,e0,34,8
0,20,7e,ca,e5,5d,cc,86,21,36,c1,48,4c,e0,34,8,0,20,7e,ca,e5
SQL> select mgr from department;
MGR
------------------------------------00002202085DCC862136C2484CE0340800207
ECAE55DCC862136C1484CE0340800207ECAE5
SQL> selec deref(mgr) from department;
DEREF(MGR) (EMPNO,ENAME)
------------------------------------
EMP_TYP(1,'peter')
```

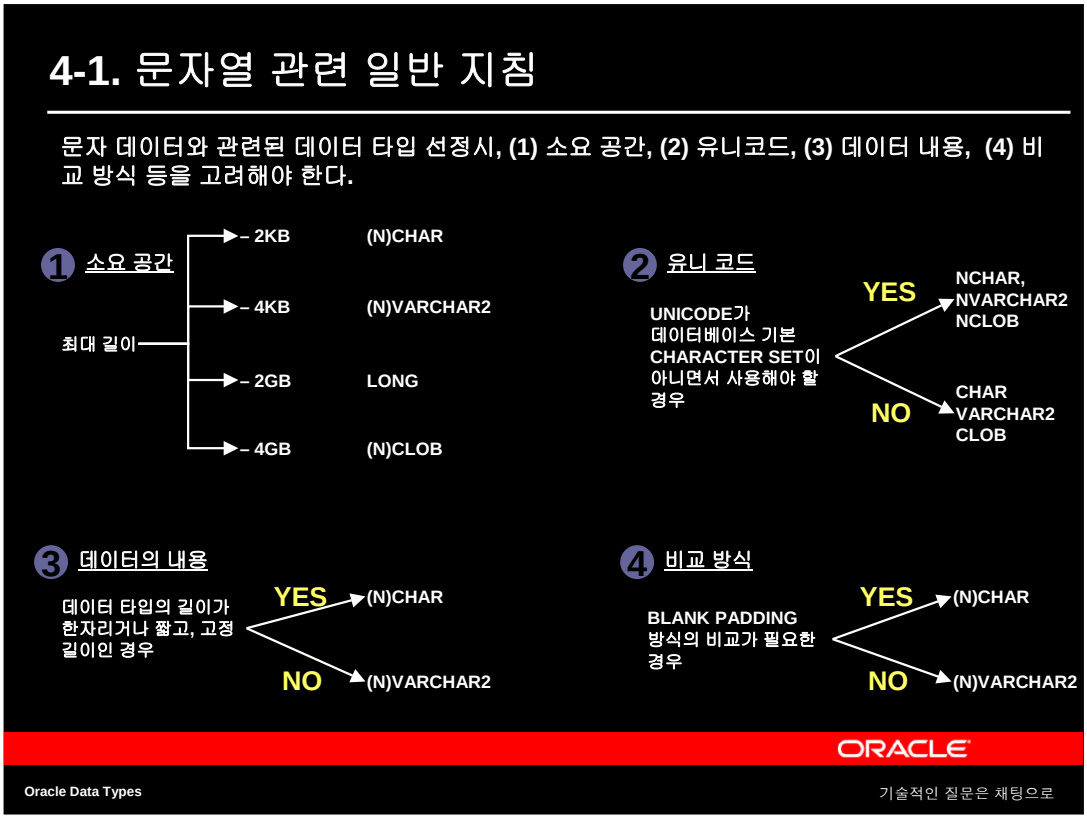
REF 데이터 타입의 코드는 111번으로 확인이 가능합니다.



4 장에서는 앞에서 알아본 Oracle에서 제공되는 각 Data Type의 특성을 근간으로 데이터 타입 선정 지침에 대해서 알아 보겠습니다..

Oracle에서 제공하는 내장 데이터 타입 중 문자, 숫자 및 날짜 관련 일반 지침에 대해서 말씀 드리고,

Long/long raw의 제약 사항 및 blob/clob의 제약 사항에 대해서 살펴 본 후, 관련 Data Type선정에 대한 기준을 소개해 드리겠습니다.



문자 데이터와 관련된 데이터 타입 선정 시에는 (1) 소요 공간, (2) 유니코드, (3) 데이터 내용, (4) 비교 방식 등을 고려해야 합니다.

소요공간이란 저장되어야 하는 user data의 size를 고려하여 각 문자 Data Type별 최대 size를 알고 선정하는 것을 의미합니다.

Oracle9i기준 Char Data Type은 최대 2KB저장이 가능하며, VARCHAR2는 4KB까지,

Long Data Type은 2GB까지, clob Data Type은 4GB까지 가능합니다.

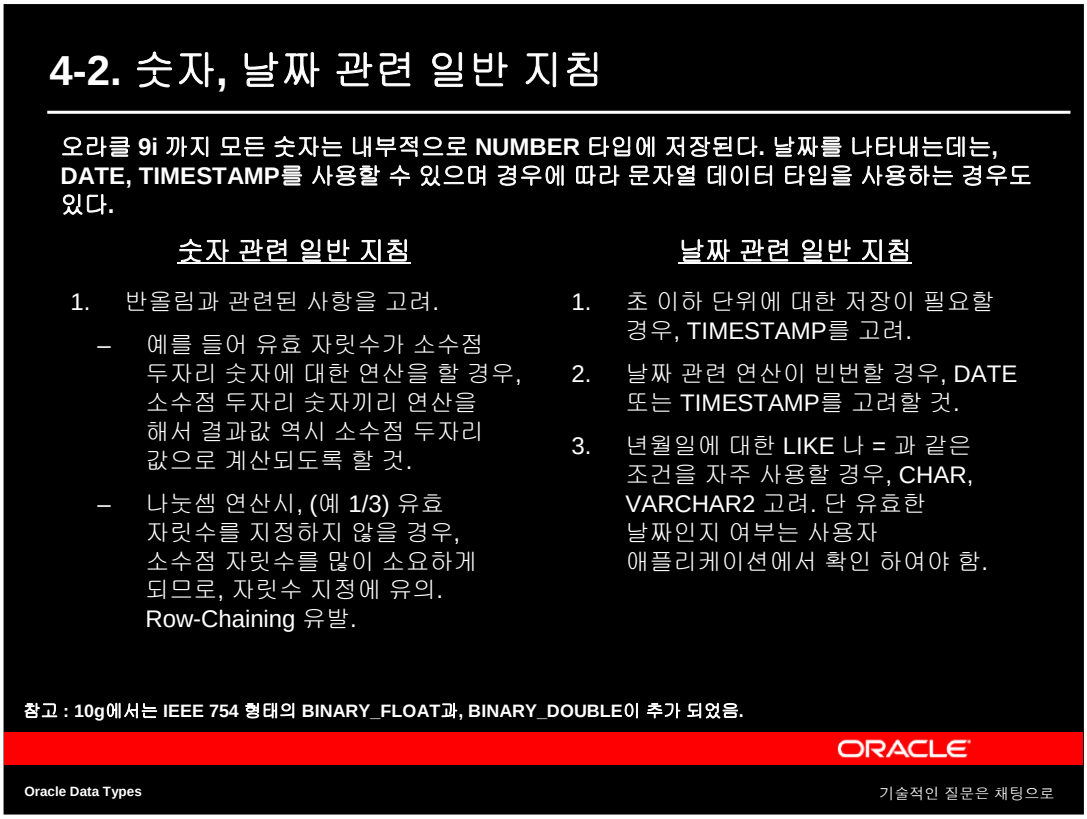
기본 database characterset에 해당하는 값이 아닌, 유니코드 national characterset 형태로 데이터를 저장할 필요가 있는 경우에는 NCHAR, NVARCHAR2, NCLOB 데이터 타입에 대한 고려가 필요합니다.

문자 데이터의 내용에 따라서 저장되는 문자 데이터가 고정길이이면서 한자리거나 짧은 경우엔 CHAR Data Type을 고려 할 필요가 있습니다.

그러나 가변길이인 경우엔 char에 비해 varchar2 Data Type을 사용하시는 것이 자원의 낭비를 줄이고, 테이블에 대한 full scan시 읽어야 할 블럭 갯수를 줄여 성능저하를 막을 수 있습니다.

비교방식을 고려해보면, Char Data Type은 비교시 실제 데이터만 가지고 비교하는 것이 아니라 padding된 blank를 넣어야 올바른 비교가 되므로 , 사용자 어플리케이션 작성시 유의해야 합니다.

테이블 설계 시, NULL column들이 많은 Column들을 물리적으로 뒤로 두는 것이 Space절약을 위해 바람직합니다. 또한 초기에 Null이었다가 Update시 Row길이가 늘어난다면 PCTFREE를 크게 주어야 합니다.



오라클 9i 까지 모든 숫자는 내부적으로 NUMBER 타입에 저장됩니다.

Number Data Type사용 시 일반 지침을 보면 다음과 같습니다.

Number Column의 값이 나누어 지거나 곱해지거나 해서 저장되는 경우 단순히 데이터 타입을 NUMBER로 지정하면 소숫점 자리가 커질 수 있으므로 Row Chaining발생율이 높아 집니다. 이러한 경우를 대비하여 가능하다면 NUMBER(precision,scale)로 precision와 scale도 지정해 둘 것을 권장합니다.

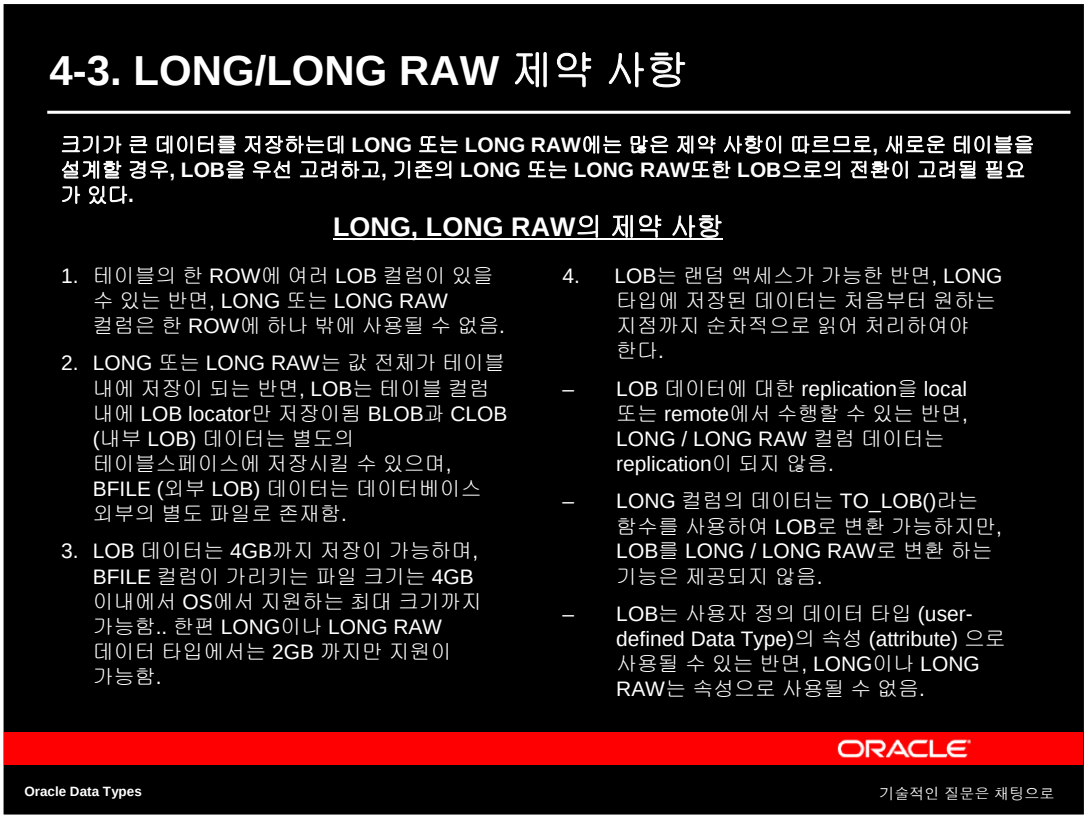
날짜를 표현 하는 데는, DATE, TIMESTAMP를 사용할 수 있으며 경우에 따라 문자열 데이터 타입을 사용하는 경우도 있습니다.

초 이하 단위에 대한 저장이 필요할 경우, TIMESTAMP를 고려가 필요하며, 날짜 관련 연산이 빈번할 경우, 일반 문자 데이터 타입보다는 DATE 또는 TIMESTAMP를 고려해야 합니다.

그러나 년월일에 대한 LIKE 나 = 과 같은 조건을 자주 사용할 경우, CHAR, VARCHAR2 고려해 볼 수 있습니다.

(단순히 같은 날짜인지 여부만 비교하는 경우, = 연산자만으로는 제대로 된 비교를 할 수 없으며, TRUNC 함수를 사용할 경우 기본적으로는 인덱스를 타지 않습니다.)

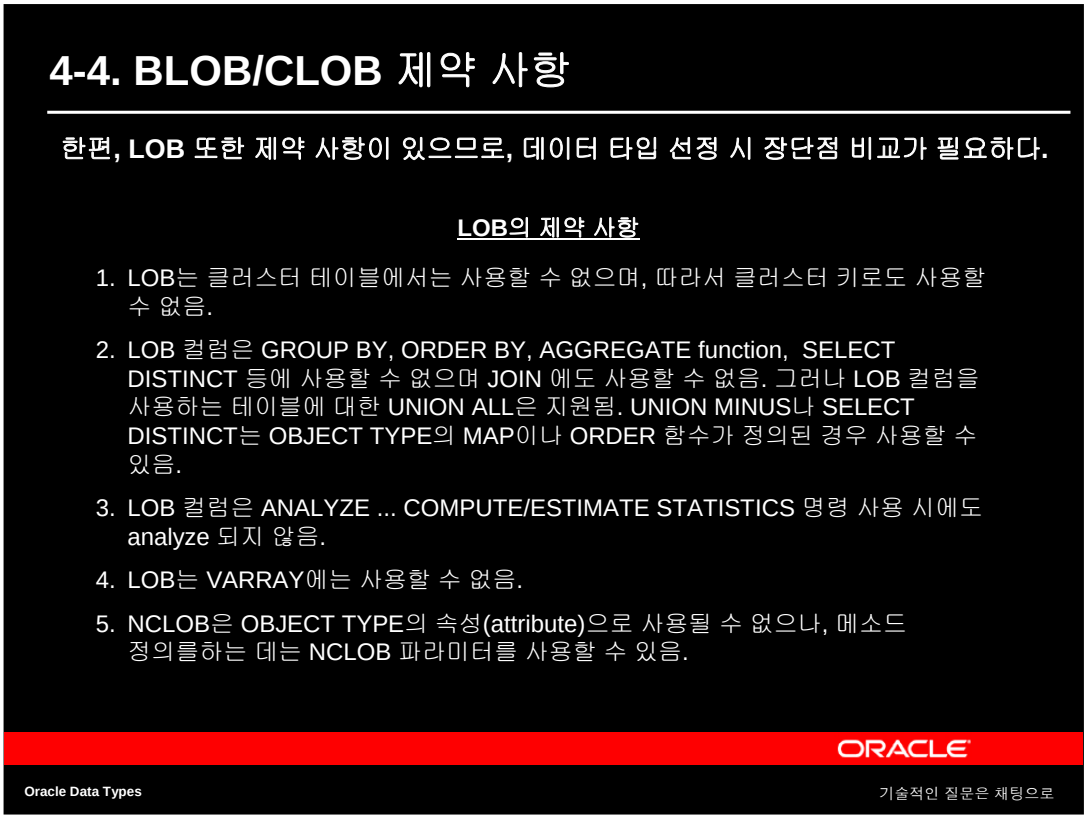
하지만 유효한 날짜인지 여부는 사용자 애플리케이션에서 확인 하여야 합니다.



크기가 큰 데이터를 저장하는데 LONG 또는 LONG RAW에는 많은 제약 사항이 따르므로, 새로운 테이블을 설계할 경우, LOB을 우선 고려하고, 기존의 LONG 또는 LONG RAW또한 LOB으로의 전환이 고려될 필요가 있습니다.

LONG, LONG RAW Data Type은 다음과 같은 제약 사항을 가지고 있습니다.

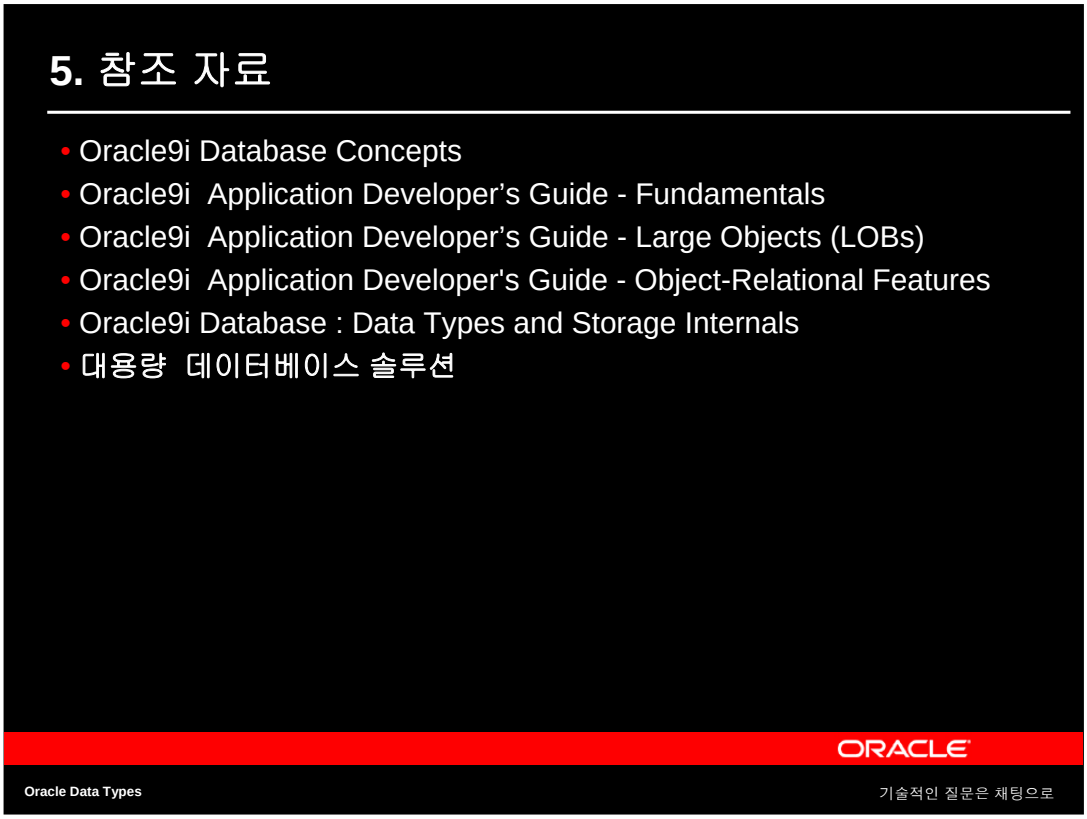
1. 테이블의 한 ROW에 여러 LOB 컬럼이 있을 수 있는 반면, LONG 또는 LONG RAW 컬럼은 한 ROW에 하나 밖에 사용될 수 없습니다.
2. LONG 또는 LONG RAW는 값 전체가 테이블 내에 저장이 되므로 row chaining이 발생할 가능성이 큰 반면, LOB는 테이블 컬럼 내에 LOB locator만 저장 되기 때문에 그와 같은 문제를 피할 수 있습니다.. BLOB과 CLOB (내부 LOB) 데이터는 별도의 테이블스페이스에 저장시킬 수 있으며, BFILE (외부 LOB) 데이터는 데이터베이스 외부의 별도 파일로 존재합니다.
3. LOB 데이터는 4GB까지 저장이 가능하며, BFILE 컬럼이 가리키는 파일 크기는 4GB 이내에서 OS에서 지원하는 최대 크기까지 가능합니다.
4. 한편 LONG이나 LONG RAW 데이터 타입에서는 2GB 까지만 지원이 가능합니다.
5. LOB는 랜덤 액세스가 가능한 반면, LONG 타입에 저장된 데이터는 처음부터 원하는 지점까지 순차적으로 읽어 처리하여야 합니다.
6. LOB 데이터에 대한 replication을 local 또는 remote에서 수행할 수 있는 반면, LONG / LONG RAW 컬럼 데이터는 replication이 되지 않습니다.
7. LONG 컬럼의 데이터는 TO_LOB()라는 함수를 사용하여 LOB로 변환 가능하지만, LOB를 LONG / LONG RAW로 변환 하는 기능은 제공되지 않습니다.
8. LOB는 사용자 정의 데이터 타입 (user-defined Data Type)의 속성 (attribute) 으로 사용될 수 있는 반면, LONG이나 LONG RAW는 속성으로 사용될 수 없습니다.



한편, LOB 또한 제약 사항이 있으므로, 데이터 타입 선정 시 장단점 비교가 필요합니다.

1. LOB는 클러스터 테이블에서는 사용할 수 없으며, 따라서 클러스터 키로도 사용할 수 없습니다.
2. LOB 컬럼은 GROUP BY, ORDER BY, AGGREGATE function, SELECT DISTINCT 등에 사용할 수 없으며 JOIN 에도 사용할 수 없습니다.

   그러나 LOB 컬럼을 사용하는 테이블에 대한 UNION ALL은 지원됩니다. UNION MINUS나 SELECT DISTINCT는 OBJECT TYPE의 MAP이나 ORDER 함수가 정의된 경우 사용할 수 있습니다.
3. LOB 컬럼은 ANALYZE ... COMPUTE/ESTIMATE STATISTICS 명령 사용 시에도 analyze 되지 않습니다.
4. LOB는 VARRAY에는 사용할 수 없습니다.
5. NCLOB은 OBJECT TYPE의 속성(attribute)으로 사용될 수 없으나, 메소드 정의를 하는 데는 NCLOB 파라미터를 사용할 수 있습니다.



본 세미나 자료는 다음의 자료들을 참조하여 작성되었습니다.
기타 상세한 설명이 필요한 경우엔 참조해 확인하여 보실 수 있습니다.