Advanced Data Bases

Course II - SQL 3

Hanen Ochi

part I

-

Introduction

Design, Development, Use, Administration

1. Conceptual step: Design and Database Modeling
2. logical step: Establishment of a database
3. Physical Stage: Software (DBMS, interfaces, ...) & equipment

Database Modeling

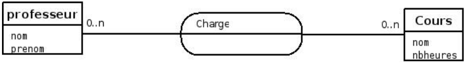
- Analysis phase: definition of a conceptual schema
- Conceptual diagram of data (SCD): according to the formalism,

  set of entities and Associations

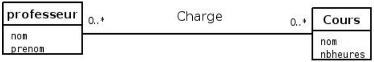
  or

  set of Classes

- Formalism EA, ER



- UML formalism:



Database Modeling

Different modeling formalism of conceptual schemes of DB:

- Formalism EA, ER, EER
  - Model Entity-Relationship (Entity-Relationship Model)
  - Model Entity-Relationship Extended (Extended Entity-Relationship Model)
- Formalism UML (Unified Modeling Language)

Database Modeling

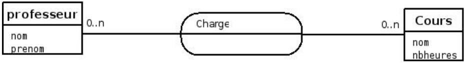
Both formalisms E / R and UML are very close / "equivalent"

Entity / Association	→	UML
Entity		Object
Type of entity		Class
Relationship		Object
Type of association		Class
Attribute / Property		Property
Role / Label		Role
		Method
Field		domain constraint
Key		Key Constraint
Constraint		Constraint
cardinality		Multiplicity / cardinality
0.1 1.1 0, n 1, n, b, a		1 0..1 0 .. * 1 .. * a..ba
Diagram E / A		UML Class Diagram

Entity-Relationship Model

Recall

- Entity: Any concrete or abstract concept that can be individualized
- Class or type of entities: entities grouping similar (generic level)
- Association relation between multiple entities
- Class or type of association: group of associations with the same characteristics



Extended entity-relationship model

- Entity-relationship model: reduced concept but sufficient for modeling simple problems (or less complex)
- Extended Entity-relationship Model: more precise and expressive modeling of complex problems.

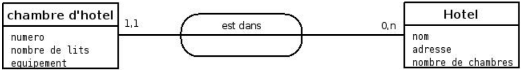
  Introduction of abstraction mechanisms:

  - Weak types
  - classification
  - aggregation
  - inheritance
  - ...

Weak Types

Type of entities or weak associations:

- existence of a body subject to the existence of another type of entity or association



Here the identifier "number" of a single room is unique only for a given hotel!

Classification

- Grouping entities into classes based on common properties
- Possibility of classifying an object in several classes

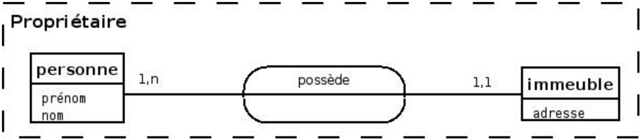
example:

- eBook: electronic file, and book
- Bus: Public transport vehicle, motor vehicle explosion

Aggregation

Description of complex entities' types.

One type of associations between entities' types is considered as a new type of entities

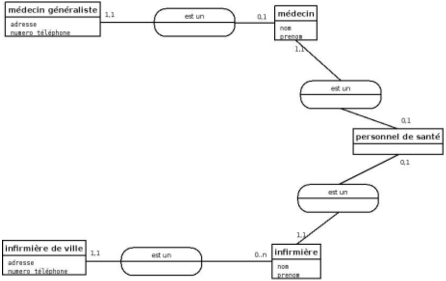


Inheritance

Specialization - Generalization

One type of entity A is a specialization of another type of entity B if

- each entity A is a B entity
- One entity (at most) of B is associated with a unit of A



Translation of functional constraints of integrity

Constraints: Score, Exclusion, Totality, Concurrency, Inclusion ...

All constraints can be set or programmed via:

- the declaration of constraints (constraints)
- programming
  - functions (functions)
  - procedures (procedures)
  - packages (packages)
  - triggers (triggers)

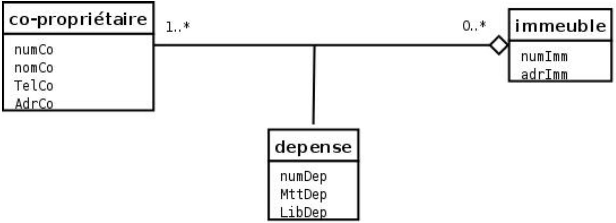
PL / SQL or with a host language such as C, C ++, Java

part II

-

Abstraction in SQL2

Translation aggregation relationships



Each building "includes" a number of owners.

Each building has at least 1 owner!

Translation aggregation relationships

```
REM ___ Un co−propriétaire peut posséder plusieurs immeubles
create table COPROPRIETAIRE
(
          NUMCO number ( 7 ) ,
          NOMCO varchar ( 1 0 ) ,
          TELCO varchar ( 1 5 ) ,
          ADRCO varchar ( 5 0 ) ,
          constraint PK_COPROPRIETAIRE primary key (NUMCO)
) ;
REM ___ Un immeuble doit être possédé par un ou
REM plusieurs copropriétaires
create table IMMEUBLE
(
          NUMIMM number ( 7 ) ,
          ADRIMM varchar ( 5 0 ) ,
          constraint PK_IMMEUBLE primary key (NUMIMM)
) ;
```

Translation aggregation associations

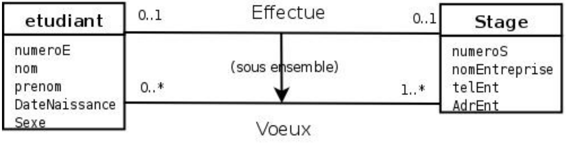
```
create table DEPENSE
(
        NUMCO number ( 7 ) ,
        NUMIMM number ( 7 ) ,
        DATEDEP date ,
        MTTDEP number ( 10 , 2 ) ,
        LIBDEP varchar ( 50 ) ,
        constraint PK_DEPENSE primary key (NUMCO,NUMIMM) ,
        constraint FK_DEPENSE_NUMCO_COPROPR foreign key (NUMCO)
                references COPROPRIETAIRE(NUMCO) on delete cascade ,
        constraint FK_DEPENSE_NUMIMM_IMMEUBLE foreign key (NUMIMM)
                references IMMEUBLE(NUMIMM ) on delete cascade

) ;
```

Translation aggregation associations

```
create table DEPENSE
(
          NUMCO number ( 7 ) ,
          NUMIMM number ( 7 ) ,
          DATEDEP date ,
          MTTDEP number ( 1 0 , 2 ) ,
          LIBDEP varchar ( 5 0 ) ,
          constraint PK_DEPENSE primary key (NUMCO,NUMIMM) ,
          constraint FK_DEPENSE_NUMCO_COPROPR foreign key (NUMCO)
                    references COPROPRIETAIRE(NUMCO) on delete cascade ,
          constraint FK_DEPENSE_NUMIMM_IMMEUBLE foreign key (NUMIMM)
                    references IMMEUBLE(NUMIMM ) on delete cascade

) ;
```

The minimum cardinality of the association will be tested through a PL / SQL procedure

Inclusion constraint



```
create    table  STAGE
(
  NUMEROS number (7),
   NOMENTREPRISE varchar (4 0),
   TELENT varchar (1 5)
   ADRENT varchar (5 0),
    constraint  PK_STAGE primary key (NUMEROS)
);
```

Inclusion constraint

Table per class

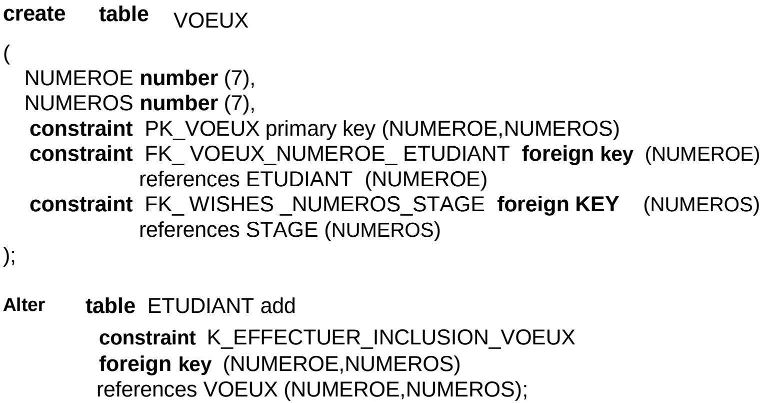
```
Create table ETUDIANT
(
          NUMEROE number ( 7 ) ,
          NOM varchar ( 1 0 ) ,
          PRENOM varchar ( 1 0 ) ,
          DATENAISSANCE date ,
          SEXE char ( 1 ) ,
          NUMEROS number ( 7 ) ,
          constraint PK_ETUDIANT primary key (NUMEROE) ,
          constrain t FK_ETUDIANT_NUMEROS_STAGE foreign key (NUMEROS)
                    references STAGE(NUMEROS),
          constrain t CK_ETUDIANT_SEXE check (SEXE i n ( ’M’ , ’F ’ ) )
) ;
```

Inclusion constraint

```
create    table   VOEUX
(
  NUMEROE number (7),
  NUMEROS number (7),
   constraint  PK_VOEUX primary key (NUMEROE,NUMEROS)
   constraint  FK_ VOEUX_NUMEROE_ ETUDIANT  foreign key  (NUMEROE)
               references ETUDIANT  (NUMEROE)
   constraint  FK_ WISHES _NUMEROS_STAGE  foreign KEY    (NUMEROS)
               references STAGE (NUMEROS)
);
```

inclusion Constraint?

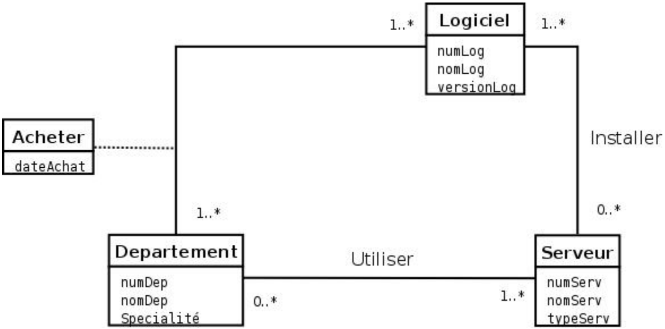
Inclusion constraint



```
create    table  VOEUX
(
  NUMEROE number (7),
  NUMEROS number (7),
   constraint  PK_VOEUX primary key (NUMEROE,NUMEROS)
   constraint  FK_ VOEUX_NUMEROE_ ETUDIANT  foreign key  (NUMEROE)
               references ETUDIANT  (NUMEROE)
   constraint  FK_ WISHES _NUMEROS_STAGE  foreign KEY   (NUMEROS)
               references STAGE (NUMEROS)
);

Alter     table  ETUDIANT add
            constraint  K_EFFECTUER_INCLUSION_VOEUX
            foreign key  (NUMEROE,NUMEROS)
           references VOEUX (NUMEROE,NUMEROS);
```

Inclusion constraint

Inclusion constraint: The software must be installed on a server of the department that purchased the program.



The software L purchased by D department is installed on a server S, for among other things, this department

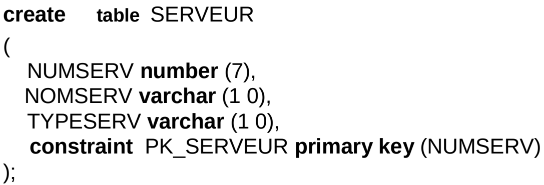
Inclusion constraint

Table per Class

```
create      table    DEPARTMENT
(
   NUMDEP number (7),
   NOMDEP varchar (1 0),
    SPECIALTE varchar (2 0),
    constraint PK_DEPARTEMENT primary key                (NUMDEP)

);

create       table    LOGICIEL
(
   NUMLOG number (7),
   NOMLOG varchar (1 0),
    VERSIONLOG varchar (1 0)          ,
    constraint  PK_LOGICIEL          primary key      (NUMLOG)
);
```

Inclusion constraint

Table per Class



```
create     table  SERVEUR
(
   NUMSERV number (7),
   NOMSERV varchar (1 0),
   TYPESERV varchar (1 0),
    constraint  PK_SERVEUR primary key (NUMSERV)
);
```

Inclusion constraint

A table by Association or Class-Association

```
create     table  ACHETER (
   NUMDEP number (7), NUMLOG number (7), DATEACHAT date ,
    constraint PK_ACHETER primary key (NUMDEP, NUMLOG),
    constraint FK_ACHETER_NUMDEP_DEPARTEMENT
                Foreign KEY (NUMDEP) references DEPARTMENT (NUMDEP)
    constraint  FK_ACHETER_NUMLOG_LOGICIEL
                foreign KEY                    (NUMLOG)      references LOGICIEL (NUMLOG)
);

create table UTILISER(
   NUMDEP number (7), NUMSERV number (7),
    constraint PK_UTILISER primary key (NUMDEP, NUMSERV),
    constraint FK_UTILISER_NUMDEP_DEPARTEMENT
         foreign key (NUMDEP) references DEPARTMENT (NUMDEP),
   constraint FK_UTILISER_NUMSERV_SERVEUR
         foreig NKEY    (NUMSERV) references SERVER (NUMSERV)
);
```

Inclusion constraint

A table by Association or Class-Association

```
    Create Table INSTALLER (
NUMLOG number (7), NUMSERV number (7),
     constraint PK_INSTALLER primary key (NUMLOG, NUMSERV)
     constraint FK_INSTALLER_NUMLOG_LOGICIEL
          foreign KEY(NUMLOG)      references      LOGICIEL (NUMLOG)
 constraint FK_INSTALLER_NUMSERV_SERVEUR
          foreign KEY(NUMSERV) references SERVEUR (NUMSERV)        );
```

Inclusion Constraint

trigger

The software purchased by D department is installed on a server S dedicated to this department

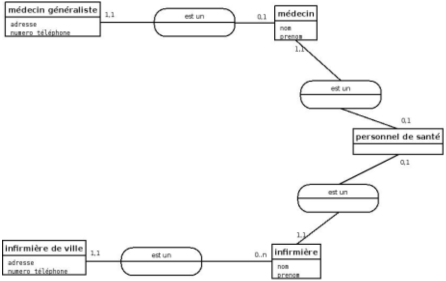
```
Create or replace trigger trig_contrainte_inclusion
  before    insert on INSTALLER
  For each row
  declared
 LOGIC number (7);
 SERV number (7);
  begin
        select  ACHETER.NUMLOG,     UTILISER .NUMSERV into LOGIC SERV
        from   ACHETER, UTILISER
        Where   ACHETER.NUMDEP = UTILISER .NUMDEP and
                ACHETER.NUMLOG =  : new .NUMLOG and
                UTILISER .NUMSERV = : new .NUMSERV;
  exception
        When  NO_DATA_FOUND       Then
        raise_application_error (-20,100,
         Le logiciel doit être installé sur
           un serveur du département acheteur');

end ;
/
```

Inheritance

Specialization - Generalization

One type of entity A is a specialization of another type of entity B if

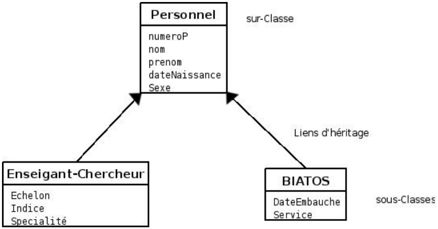
- each entity A is a B entity
- One entity (at most) of B is associated with a unit of A



Translation of inheritance associations

Translation of inheritance constraints

Personnel management in a university



Inheritance associations in UML (1)

- Presenting the different cases of inheritance based on instances
- Modeling of different inheritance in the UML formalism with constraints
  - partition
  - exclusion
  - totality

Inheritance associations in UML (2)

Expression of inheritance cases using

- coverage
- disjunction

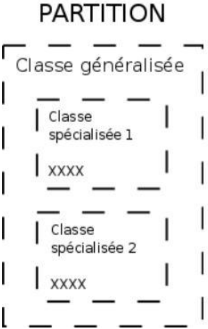
of instances in a given population

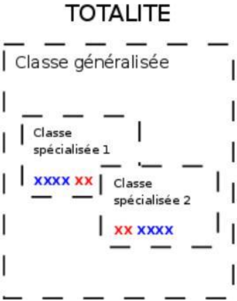
Four types of constraints are identified:

- partition
- totality
- exclusion
- the lack of constraints

Inheritance constraints PARTITION and TOTALITY

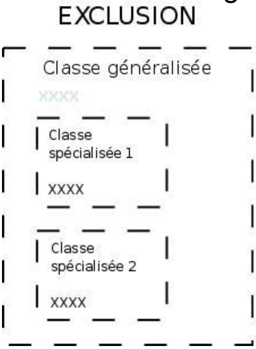
- Disjunction & coverage → Partition
- Non-disjunction & Coverage →Totality

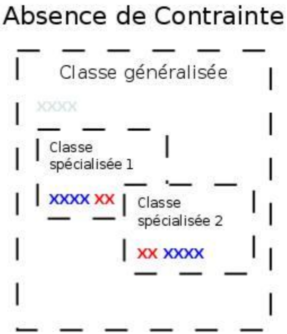




Inheritance Constraints EXCLUSION and NO CONSTRAINTS

- Disjunction & Non-Coverage → Exclusion
- Non-disjunction & Non-Coverage → No Constraints





Example (1)

Personnel management in a university

- Cover + disjunction → Partition

  Personal (P) equals to the union of Teacher (EC) and BIATOS (B) and the EC and B Intersection is Empty

- Cover + Non-disjunction → Totality

  Personal (P) equals to the union of Teacher (EC) and BIATOS (B) and the EC and B Intersection is not Empty

Example (2)

Personnel management in a university

- Non-Coverage + disjunction → Exclusion

  The Union of Teacher (EC) and BIATOS (B) is included in P and EC and B Intersection is Empty

- Non-Coverage + non-disjunction → No constraints

  The Union of Teacher (EC) and BIATOS (B) is included in P and EC and B Intersection is not Empty

Transformation of Inheritance associations

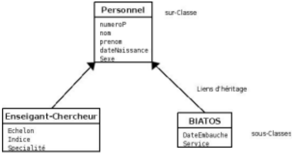
Translation of inheritance relationship depends on its constraints

→ 3 decomposition families:

- Decomposition by distinction
- Decomposition top-down (push-down)
- Decomposition bottom-up (push-up)

Decomposition by distinction

- Transformation of each subclass in a relationship
- Migration of the primary key of the super class to relationships or issues of the subclasses
- The primary key of the super class becomes both primary and foreign key



Distinction

PERSONNEL (Numéro , Nom, Prénom ,
DateNaissance , Sexe)

ENSEIGNANT(Numéro * ,
Echelon,
Indice , Spécialité)

BIATOS(Numéro _ , DateEmbauche ,
Service)

Top-down decomposition

Two possible cases according to the inheritance constraint:

- Constraint of *partition* or *totality* on the association:

  Possibility of no translation of the relationship resulting of the super class

  → Migration of all attributes on the relationships from the subclasses
- Otherwise: Migration of all attributes on relationships from the subclasses

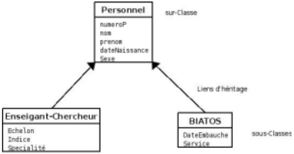
  → Duplicate data

Top-down decomposition

Example

Partition Constraint:

- No staff can be both teacher and BIATOS
- It also no personal being neither teacher nor BIATOS.



ENSEIGNANT(Numéro , Nom, Prénom , DateNaissance , Sexe , Echelon , Indice , Spécialité)

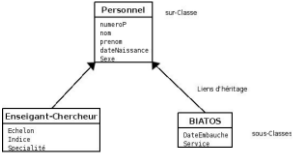
BIATOS(Numéro , Nom, Prénom , DateNaissance , Sexe

DateEmbauche , Service)

Bottom-up decomposition

- Removed all relationships resulting from or subclasses
- Migration of attributes on the relationship of the super class

Example: (No constraints)



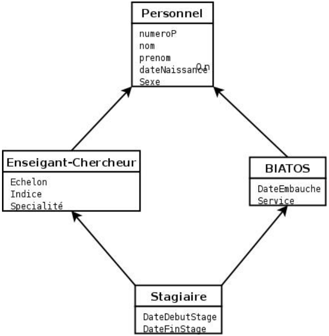
ascending

PERSONNEL(Numéro , Nom, Prénom ,
DateNaissance , Sexe ,
Echelon , Indice ,
Specialité,
DateEmbauche , Service)

Transformation of multiple inheritance associations

Same rules; several possibilities

Example: (Bottom up decomposition) teacher on exclusion Constraint on *Enseignant* and *BIATOS*



PERSONNEL(Numéro , Nom, Prénom ,
DateNaissance , Sexe)

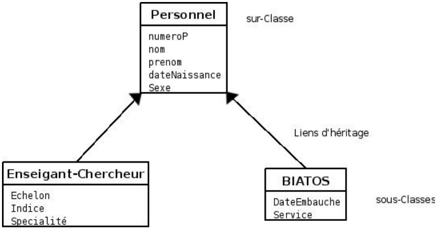
ENSEIGNANT (Numéro * ,
Echelon , Indice , Spécialité ,
DateDébutStage , DateFinStage)

BIATOS(Numéro* , DateEmbauche ,
Service ,
DateDébutStage , Dat eFinStage)

Transformation of inheritance associations on SQL 2

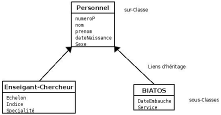
Example

Personnel management in a university



Transformation of inheritance associations

Decomposition by distinction



Distinction

PERSONNEL(Numéro , Nom, Prénom , DateNaissance , Sexe)

ENSEIGNANT(Nu ENSEIGNANT(Numéro_ ,
méro_ ,
Echelon , Echelon ,
Indice , Spécialité)

BIATOS(Numéro_ , DateEmbauche ,

```
REM***     A staff at University ed
 create    Table PERSONNEL
 (NUMBER number (7),
    NOM varchar (1 0),
    PRENOM varchar (1 0),
     DATENAISSANCE date,
     SEXE   char (1),
      constraint        PK_PERSONNEL primarykey        (NUMERO)
      constraint        CK_SEXE_PERSONNEL check  (In SEXE ( 'M', 'F'))


 );
```

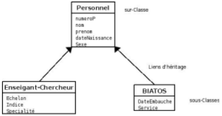
Transformation of inheritance associations

Decomposition by distinction

```
REM***      Personnel enseignant
 create     table  ENSEIGNANT
 (NUMERO   number (7), ECHELON number (2),
    INDEX    number (5) SPECIALTY varchar (2 0),
    constraint        PK_ENSEIGNANT primary key (NUMERO)
    constraint        FK_ENS_PERS foreign key (NUMERO)
                                 References PERSONNEL
 );

REM           Personnel BIATOS (I ng, Adm, Tech, Ouv, S erv)
 create     table  BIATOS
 (   NUMERO number (7), DATEEMBAUCHE dates
    SERVICE varchar (2 0),
    constraint  PK_BIATOS primary key               (NUMERO)
    constraint  FK_BIATOS_PERS         foreign       key  (NUMERO)
                                 references        PERSONNEL
 );
```

Transformation of inheritance associations

Down up decomposition



down

ENSEIGNANT(Numéro , Nom, Prénom , DateNaissance , Sexe , Echelon , Indice , Spécialité)

BIATOS(Numéro , Nom, Prénom , DateNaissance , Sexe , DateEmbauche , Service)

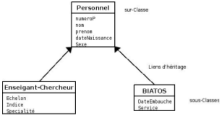
REM **** Personnel enseignant

create **table** ENSEIGNANT
(NUMERO number (7) NOM varchar (1 0),
PRENOM varchar (1 0), DATENAISSANCE **date** ,
SEX char (1), STEP number (2),
INDEX number (5) SPECIALTY varchar (2 0),
constraint CK_SEXE_ TEACHER check
(SEX in ('M', 'F'))
key
(NUMERO));

constraint PK_ENSEIGNANT primary

Transformation of inheritance associations

Down up decomposition



down

ENSEIGNANT(Numéro , Nom, Prénom , DateNaissance , Sexe , Echelon , Indice , Spécialité)

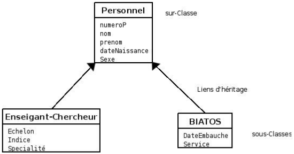
BIATOS(Numéro , Nom, Prénom , DateNaissance , Sexe , DateEmbauche , Service)

REM P ersonnel BIATOS

```
create   table  BIATOS
(NUMEROnumber (7) NAME varchar (1 0) NOM varchar (1 0),
   BirthDate Date, SEX char (1), DATEEMBAUCHE date
   SERVICE varchar (0 2),
   constraint  CK_SEXE_ BIATOS check (in SEX          ( 'M',     'F'))
   constraint  PK_BIATOS primary key (NUMERO));
```

Transformation of inheritance associations

Bottom updecomposition



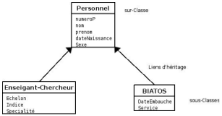
ascending

PERSONNEL (Numéro , Nom, Prénom ,
DateNaissance , Sexe ,
Echelon , Indice ,
Spécialité ,
DateEmbauche , Service)

```
                    Create table PERSONNEL
(NUMERO number (7) NOM varchar (1 0) PRENOM varchar (1 0),
   DateNaissance Date, SEXE char (1), Echelon number (2),
   INDICE number (5) SPECIALTE varchar (2 0),
  DATEEMBAUCHE dates SERVICE varchar (2 0),
   constraint  CK_SEXE_ STAFF check (in SEXE              ( 'M',    'F'))
   constraint  PK_ PERSONNEL PRIMARY KEY (NUMERO)
);
```

Translation of inheritance constraints

Decomposition distinction



● → score Constraint ● Stress screen ● exclusion constraint ● Without restraint

Inheritance Constraints:

- (Strain A) There is no staff both teaching and BIATOS
- (Strain B) There is no staff or teacher or BIATOS

Translation of inheritance constraints

Decomposition distinction

Implementing the constraint A: 2 triggers

```
REM          D e Trigger            TEACHER on

creat r eo                  puts        trigger  TRIG_ENSEIGNANT
before    inser update to r             NUMBER of one TEACHER
for   EACH rOW
declared
    number num;
begin
     select NUMBER INTO num
     from Where BIATOS NUMBER =: new .Reference;
     _ _ raise Application error (-20001, 'Lep erson the | |
                     _ to char (num) | | 'Es td e j BIATOS! ! ! ');
exception
    When                         Then  null ;
    NO_DATA_FOUND
end ;
/
```

Translation of inheritance constraints

Decomposition distinction

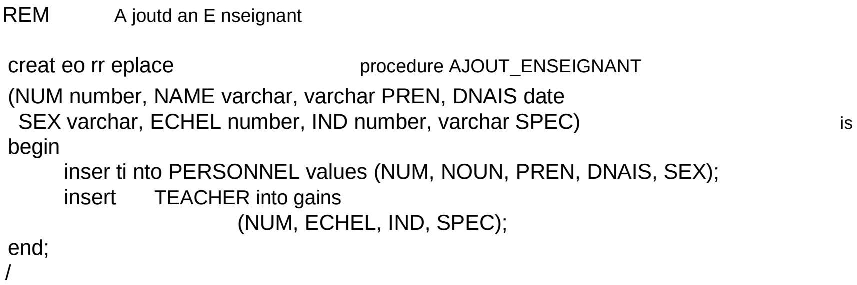
```
REM           D e Trigger               on BIATOS

 creat r eo                  puts        trigger  TRIG_BIATOS                    /
 before     inser update to r            NUMBER one of BIATOS
 for    EACH rOW
 declared
     number num;
 begin
      select NUMBER INTO num                                           '| |
     from TEACHER Where NUMBER =: new .Reference;
      _ _ raise Application error (-20001, 'The              staff
          _ to char (num) | | 'Es td e j? E nseignant! ! ! ');

 exception
     When  Then NO_DATA_FOUND  null ;
end ;
```

Translation of inheritance constraints

Decomposition distinction

Implementing the constraint B:

- stored procedures (Insert, Delete)
- triggers (Change)



```
REM         A joutd an E nseignant

 creat eo rr eplace                              procedure AJOUT_ENSEIGNANT
 (NUM number, NAME varchar, varchar PREN, DNAIS date
   SEX varchar, ECHEL number, IND number, varchar SPEC)                                                       is
 begin
        inser ti nto PERSONNEL values (NUM, NOUN, PREN, DNAIS, SEX);
        insert     TEACHER into gains
                                  (NUM, ECHEL, IND, SPEC);
 end;
/
```

Translation of inheritance constraints

Decomposition distinction

```
REM          A joutd a BIATOS

 creat eo rr eplace                       procedure AJOUT_BIATOS
 (NUM number, NAME varchar, varchar PREN, DNAIS date
   SEX varchar, demb date SERV varchar)                    is
 begin

        inser ti nto BIATOS values (NUM demb, SERV);
 end;
/
```

Translation of inheritance constraints

Decomposition distinction

```
REM          S uppressiond an E nseignant
            gol
 create     d    replaceprocedure              SUPPR_ENSEIGNANT
 (NUM number) is
 begin
       delete       from   TEACHER       Where NUMBER = num     ;
       delete       from   Where PERSONNEL NUMBER = num        ;
 end;
/

REM          S uppressiond a BIATOS
            gol
 create     d    replaceprocedure              SUPPR_BIATOS
 (NUM number) is
 begin
       delete       from   Where BIATOS NUMBER = num;
       delete       from   Where PERSONNEL NUMBER = num        ;
end ;
/
```

Translation of inheritance constraints

Decomposition distinction

```
REM       D e Trigger    for    l ar ed percussio has ndel              change    of   Scan

 creat r eo     puts       trigger  TRIG_ENSBIATOS
 before    update  NUMBER of it STAFF
 for    EACH rOW
 begin
     begin
         update TEACHER
         set NUMBER =: new .Reference
         Where NUMBER =: old .Reference;
     exception
         When    Then NO_DATA_FOUND null;
     end ;
     update     BIATOS
         set NUMBER =        : New .Reference
     Where NUMBER =        : Old .Reference;
 exception
    When  Then NO_DATA_FOUND          null ;
 end ;
/
```

Translation of inheritance constraints

use

```
REM                 I nsertions     of the    ed given Essous        SQLPLUS

REM             launching           of the          proc hard é
 select                          'I nsertion _ of _ data é es'    from  dual;
 PAUSE

 executed           AJOUT_ENSEIGNANT          (1,  'TRAIFOR'          Clément '
                                          '17-09-1958'       'M'  6      7 8 0,      BD ');
 executed           AJOUT_ENSEIGNANT          (2,  'TRAIFOR'          'C l e mentine'

                                          '22-11-1969'       'F',  6     7 8 0,      'IA');
 executed            AJOUT_BIATOS (3,         'COOKING POT ' ,  'A lex'

      '16-10-1960', 'M'                           '01-01-2002'  'C ommercial');

 PAUSE

 select        from      STAFF ;
 select        from      TEACHER ;
 select        from      BIATOS;
 PAUSE
```

Translation of inheritance constraints

Decomposition distinction

→ *Stress screen*

Inheritance Constraints:

- (Strain B) There is no staff or teacher or BIATOS
- (Strain C) There may be personal to both teacher and BIATOS

implementation:

- Constraint B: see above
- Constraint C: equivalent to not program the previous constraint A

  ⇒ No implementation of the triggers and tables TEACHER BIATOS: TRIG_ENSEIGNANT, TRIG_BIATOS

Translation of inheritance constraints

Decomposition distinction

→ *exclusion constraint*

Inheritance Constraints:

- (Strain A) There is no staff both teaching and BIATOS
- (Strain D) There may be a teacher or staff or BIATOS

implementation:

- Constraint A: see above
- Constraint D: equivalent to not program the previous constraint B
  ⇒not to implement the four procedures (add, delete) and trigger TRIG_ENSBIATOS

Translation of inheritance constraints

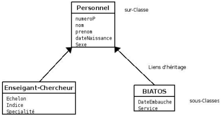
Decomposition distinction

→ *Without restraint*

No compulsion is to be programmed!

Translation of inheritance constraints

down decomposition



down

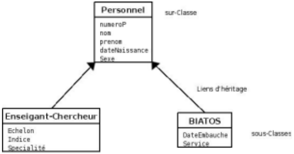
TEACHER (number,
Last name First Name , Date of birth ,
Gender, E chelon, Index ,
S p e cialit e)

BIATOS (Number, Name,
Date of birth , gender,
HireDate , S ervice)

- Constraint score? → no staff can be both teacher and IATOS and it does not is a personal being neither teacher nor IATOS
- Forced to all?
- exclusion constraint? → should the staff table for non-teaching personnel and non BIATOS
- Without restraint !

Translation of inheritance constraints

decomposition upward



● → score Constraint ● Stress screen ● exclusion constraint ● Without restraint

Inheritance Constraints:

- (Strain A) There is no staff both teaching and BIATOS
- (Strain B) There is no staff or teacher or BIATOS

Translation of inheritance constraints

decomposition upward

Implementation constraints A and B:

- staff at the table
- with the type of CHECK constraints

- Constraint A:

  Check the ECHELON columns, INDEX, SPECIALTY, DATEEMBAUCHE SERVICE and are not initialized all
- Constraint B:

  Check the ECHELON columns, INDEX, SPECIALTY, DATEEMBAUCHE SERVICE and are not all zero

Translation of inheritance constraints

decomposition upward

```
          A REMCONTRAINTE
alter     table          STAFF
  add     constraint CK_CONTRAINTE_A
          check    (
              (ECHELON is      null and INDEX      is   null and
                   SPECIALTY is   null )
            gold (HIREDATE          is   null   and   SERVICE    is   null )
          );


          REMCONTRAINTE B
alter     table          STAFF
  add     constraint CK_CONTRAINTE_B
          check    (
                   (ECHELON is   not   null    gold INDEX    is   not   null    gold
                  SPECIALTY        is   not null )
            gold  (HIREDATE           is   not   null    gold   SERVICE is       not   null )
          );
```

Translation of inheritance constraints

decomposition upward

→ *Stress screen*

Inheritance Constraints:

- (Strain B) There is no staff or teacher or BIATOS
- (Strain C) There may be personal to both teacher and BIATOS

Translation of inheritance constraints

decomposition upward

→ *Stress screen*

- Constraint B: see above
- Constraint C: Remove or disable the constraint A previous (**DROP CONSTRAINT** or DISABLE **CONSTRAINT**)
  - **DROP CONSTRAINT** : If reactivation of stress, you need to recreate (ADD CONSTRAINT)
  - DISABLE CONSTRAINT: if reactivation of the constraint, simply reactivate the request with ENABLE CONSTRAINT

Translation of inheritance constraints

decomposition upward

→ *Stress screen*

```
REM    The constraint    C      returns to fairela
REM    Qu deactivation    of   At the STRESS

     altertable  STAFF
   disableconstraint               CK_CONTRAINTE_A;
```

Translation of inheritance constraints

decomposition upward

→ *exclusion constraint*

inheritance constraint:

- A Constraint → Reactivating the constraint A deleting tuples previously not responding to this constraint
- No-stress B → Disable strain B

Translation of inheritance constraints

decomposition upward

→ *exclusion constraint*

```
REM  R e activation                of    At the STRESS
 altertable           STAFF
        enable      constraint  CK_CONTRAINTE_A;

REM  Qu deactivation                 of    the B STRAIN
 altertable           STAFF
        disable       constraint  CK_CONTRAINTE_B;
```

Translation of inheritance constraints

decomposition upward

→ *Without restraint*

No type of CHECK constraint is to program!

Conclusion / Summary

None of the solutions a panacea.

It is necessary to measure the performance of queries.

See also the kind of queries

part III

-

Inheritance on SQL 3

Transformation of Inheritance associations SQL 3

- Inheritance of types:
  - since version 9.1 of Oracle (November 2001)
  - No multiple inheritance:
- Inheritance of tables
  - Only on the Oracle's latest version

Inheritance of types

Definition of a staff at the University

```
                    Creation ot the super class
---                 type
 create  type       PERSONNEL_TYPE AS OBJECT
 (

        NUMERO number ( 7 ) ,
        NOM varchar ( 1 0 ) ,
        PRENOM varchar ( 1 0 ) ,
        DATENAISSANCE date ,
        SEXE char ( 1 )
 )
NOT FINAL     / *can include sub classes          * /
 /
```

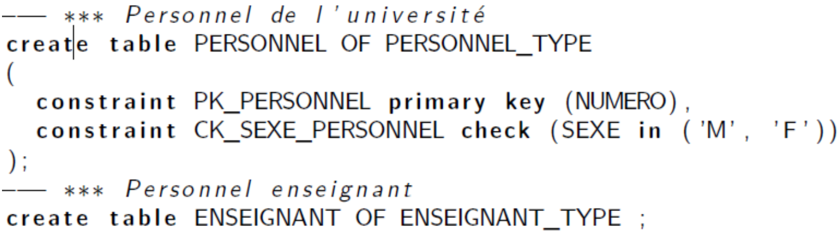
Inheritance of types

Definition of a teacher

```
---            Creation of the sub class
 Create type ENSEIGNANT_TYPE UNDER PERSONNEL_TYPE
 (
        ECHELON number ( 2 ) ,
        INDICE number ( 5 ) ,
        SPECIALITE varchar (20 )
 )
FINAL
/
```

Creating object tables and constraints

- Creating object tables depending in the types previously defined
- No guidelines specifying the inheritance; it is induced by the existing type hierarchy



```
-- *** Personnel de l'université
create table PERSONNEL OF PERSONNEL_TYPE
(
  constraint PK_PERSONNEL primary key (NUMERO),
  constraint CK_SEXE_PERSONNEL check (SEXE in ('M', 'F'))
);
-- *** Personnel enseignant
create table ENSEIGNANT OF ENSEIGNANT_TYPE ;
```

IMPORTANT: the constraints are defined only in the “Personnel” table

Creating object tables and constraints

Illustration

NB: the constraints are defined only in the “Personnel” table → It inherits a type

Inserting data into the “Personnel” table:

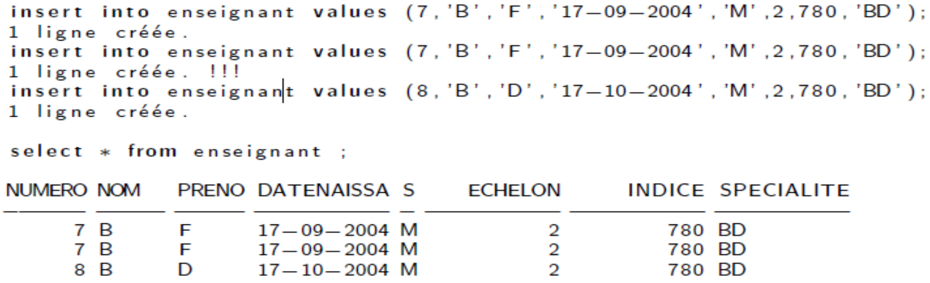
```
insert into personnel values (1, 'B', 'F', '17-09-2004', 'M');
insert into personnel values (1, 'B', 'F', '17-09-2004', 'M');
*
ERREUR à la ligne 1 :

ORA-00001: violation de contrainte unique (FB.PK_PERSONNEL)

select * from personnel;
 NUMERO       NOM          PRENOM       DATENAISSA S
---------- ------------ ------------ ---------- -
          1 B            F            17-09-2004 M
```

Creating object tables and constraints

Illustration

NB: the constraints are defined only in the “Personnel” table → It inherits a type

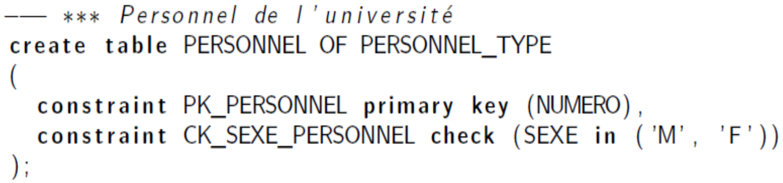
Inserting data into the “Enseignant” table:



```
insert into enseignant values (7,'B','F','17-09-2004','M',2,780,'BD');
1 ligne créée.
insert into enseignant values (7,'B','F','17-09-2004','M',2,780,'BD');
1 ligne créée. !!!
insert into enseignant values (8,'B','D','17-10-2004','M',2,780,'BD');
1 ligne créée.

select * from enseignant ;

NUMERO NOM    PRENO DATENAISSA S    ECHELON     INDICE SPECIALITE
------ ------ ----- ---------- - ---------- ---------- ----------
     7 B      F     17-09-2004 M          2        780 BD
     7 B      F     17-09-2004 M          2        780 BD
     8 B      D     17-10-2004 M          2        780 BD
```

Creating object tables and constraints (2)

- Creating object tables depending on types previously defined
- Defining constraints on tables



```
-- *** Personnel de l'université
create table PERSONNEL OF PERSONNEL_TYPE
(
  constraint PK_PERSONNEL primary key (NUMERO),
  constraint CK_SEXE_PERSONNEL check (SEXE in ('M', 'F'))
);
```

Transformation of inheritance associations SQL 3

```
-- *** Personnel enseignant
create table ENSEIGNANT OF ENSEIGNANT_TYPE
(
  constraint PK_ENSEIGNANT primary key (NUMERO),
  constraint CK_SEXE_ENSEIGNANT check (SEXE in ('M', 'F'))
);
```

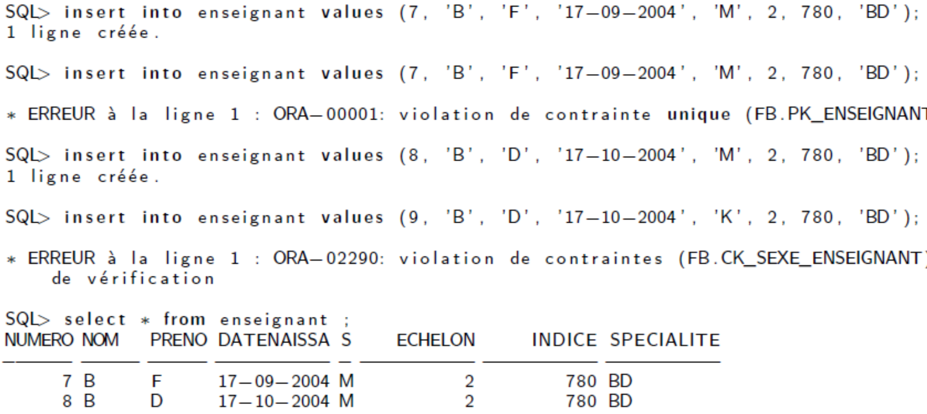
WARNING :

- define constraints, in the “Enseignant” table
- Inheritance of a type

Creating tables and constraints object (2)

illustrations

The constraints should be defined in the “Enseignant” table too:



```
SQL> insert into enseignant values (7, 'B', 'F', '17-09-2004', 'M', 2, 780, 'BD');
1 ligne créée.

SQL> insert into enseignant values (7, 'B', 'F', '17-09-2004', 'M', 2, 780, 'BD');

* ERREUR à la ligne 1 : ORA-00001: violation de contrainte unique (FB.PK_ENSEIGNANT

SQL> insert into enseignant values (8, 'B', 'D', '17-10-2004', 'M', 2, 780, 'BD');
1 ligne créée.

SQL> insert into enseignant values (9, 'B', 'D', '17-10-2004', 'K', 2, 780, 'BD');

* ERREUR à la ligne 1 : ORA-02290: violation de contraintes (FB.CK_SEXE_ENSEIGNANT)
   de vérification

SQL> select * from enseignant ;
NUMERO NOM    PRENO  DATENAISSA S    ECHELON      INDICE SPECIALITE
------ ------ ------ ---------- - ---------- ---------- ----------
     7 B      F      17-09-2004 M          2         780 BD
     8 B      D      17-10-2004 M          2         780 BD
```

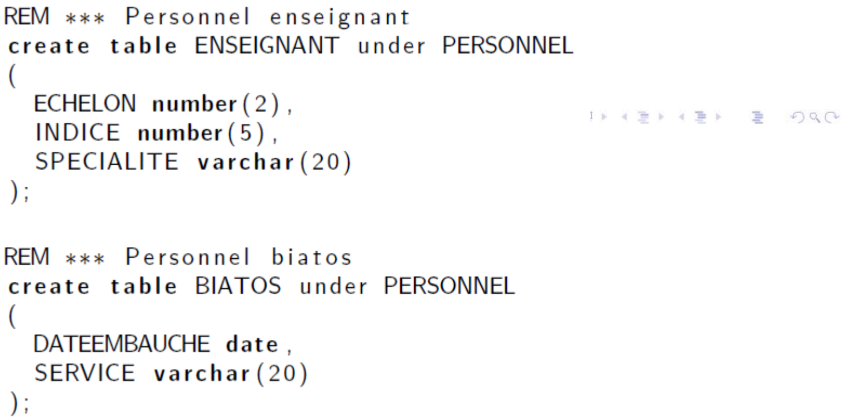
Transformation of Inheritance associations SQL 3

Inheritance of tables

```
REM *** Un personnel à l'Université
create table PERSONNEL
(
  NUMERO number(7), NOM varchar(10),
  PRENOM varchar(10), DATENAISSANCE date,
  SEXE char(1),
  constraint PK_PERSONNEL primary key (NUMERO),
  constraint CK_SEXE_PERSONNEL check (SEXE in ('M', 'F'))
);
```

Transformation of Inheritance associations SQL 3

Inheritance of tables



```
REM *** Personnel enseignant
create table ENSEIGNANT under PERSONNEL
(
  ECHELON number(2),
  INDICE number(5),
  SPECIALITE varchar(20)
);

REM *** Personnel biatos
create table BIATOS under PERSONNEL
(
  DATEEMBAUCHE date,
  SERVICE varchar(20)
);
```

part IV

-

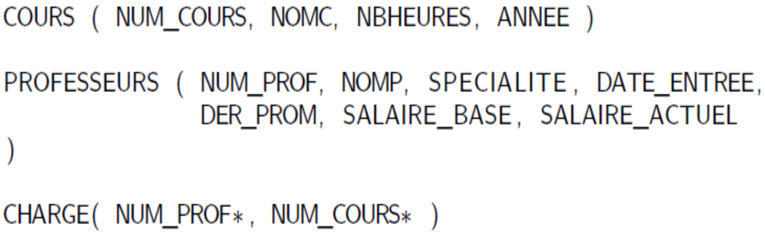
Objects in SQL3

Object Programming - SQL 3

- Object-Relational - Object
- Translation UML → Object / Relational Object

relational schema / SQL2

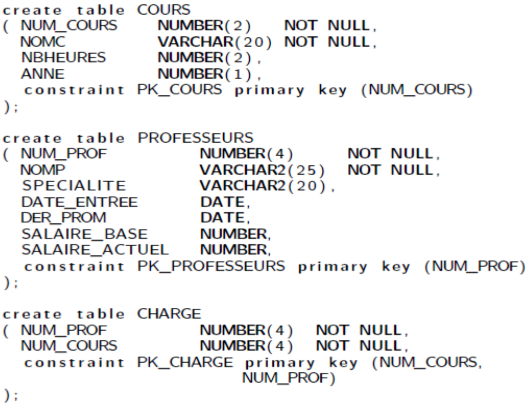
- Relational schema:



```
COURS ( NUM_COURS, NOMC, NBHEURES, ANNEE )

PROFESSEURS ( NUM_PROF, NOMP, SPECIALITE, DATE_ENTREE,
              DER_PROM, SALAIRE_BASE, SALAIRE_ACTUEL
)

CHARGE( NUM_PROF*, NUM_COURS* )
```

relational schema / SQL2

- SQL2:



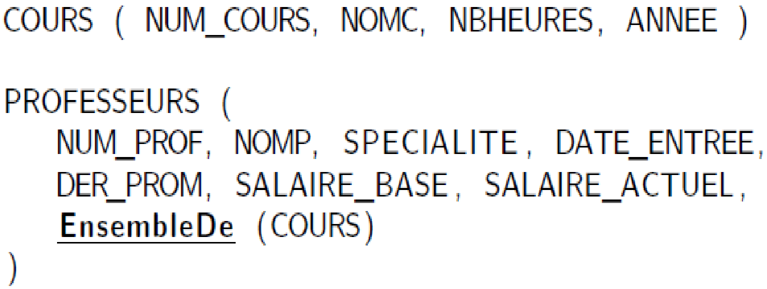
```
create table COURS
( NUM_COURS     NUMBER(2)    NOT NULL,
  NOMC          VARCHAR(20)  NOT NULL,
  NBHEURES      NUMBER(2),
  ANNE          NUMBER(1),
  constraint PK_COURS primary key (NUM_COURS)
);

create table PROFESSEURS
( NUM_PROF          NUMBER(4)      NOT NULL,
  NOMP              VARCHAR2(25)   NOT NULL,
  SPECIALITE        VARCHAR2(20),
  DATE_ENTREE       DATE,
  DER_PROM          DATE,
  SALAIRE_BASE      NUMBER,
  SALAIRE_ACTUEL    NUMBER,
  constraint PK_PROFESSEURS primary key (NUM_PROF)
);

create table CHARGE
( NUM_PROF          NUMBER(4)  NOT NULL,
  NUM_COURS         NUMBER(4)  NOT NULL,
  constraint PK_CHARGE primary key (NUM_COURS,
                              NUM_PROF)
);
```

relational schema / SQL2

```
alter table CHARGE
    add constraint FK_CHARGE_COURS
        foreign key  (NUM_COURS)
            references COURS (NUM_COURS);

alter table CHARGE
    add constraint FK_CHARGE_PROFESSEUR
        foreign key  (NUM_PROF)
            references PROFESSEURS (NUM_PROF);
```

Schema object-relational / SQL3

- object-relational schema

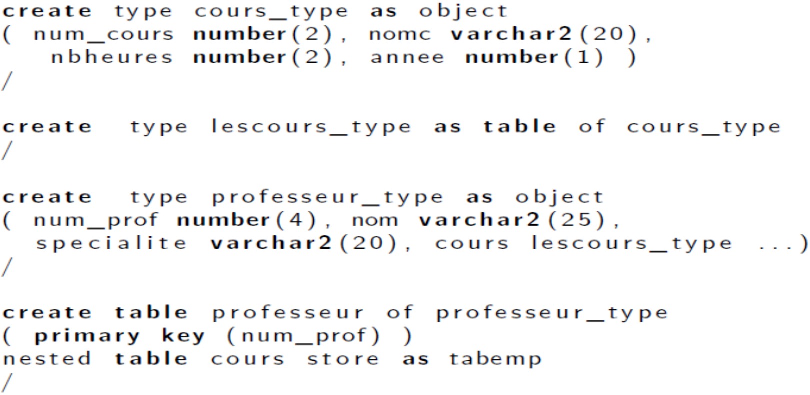


COURS (NUM_COURS, NOMC, NBHEURES, ANNEE)

PROFESSEURS (
    NUM_PROF, NOMP, SPECIALITE, DATE_ENTREE,
    DER_PROM, SALAIRE_BASE, SALAIRE_ACTUEL,
    EnsembleDe (COURS)
)

Schema object-relational / SQL3

- SQL3:

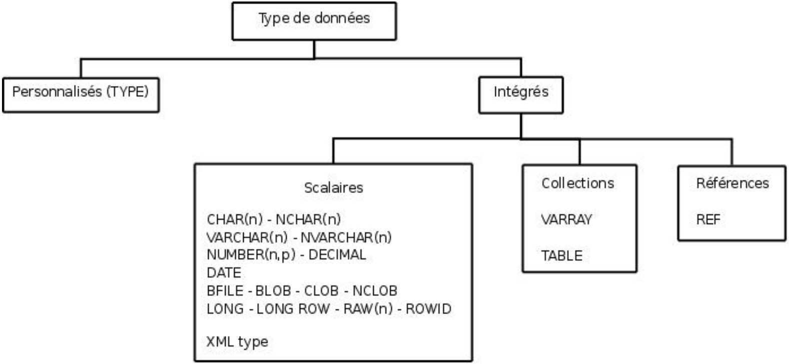


```
create type cours_type as object
( num_cours number(2), nomc varchar2(20),
   nbheures number(2), annee number(1) )
/

create  type lescours_type as table of cours_type
/

create  type professeur_type as object
( num_prof number(4), nom varchar2(25),
  specialite varchar2(20), cours lescours_type ...)
/

create table professeur of professeur_type
( primary key (num_prof) )
nested table cours store as tabemp
/
```

Object Types

Data type

Main type of Oracle data:



Type de données

Personnalisés (TYPE)

Intégrés

Scalaires

CHAR(n) - NCHAR(n)
VARCHAR(n) - NVARCHAR(n)
NUMBER(n,p) - DECIMAL
DATE
BFILE - BLOB - CLOB - NCLOB
LONG - LONG ROW - RAW(n) - ROWID

XML type

Collections

VARRAY

TABLE

Références

REF

Object Types

Persistence

Under Oracle, three categories of objects:

- Objects column (column objects) stored as structured column in a relational table;
- Online Items (row objects) stored as a line item table.
  - possess a unique identifier known as OID (Object Identifier)
  - can be indexed and partitioned
- non-persistent objects: not stored
  - nor in a column of a relational table
  - neither in a table object Online

  These objects exist only during the execution of a PL / SQL program

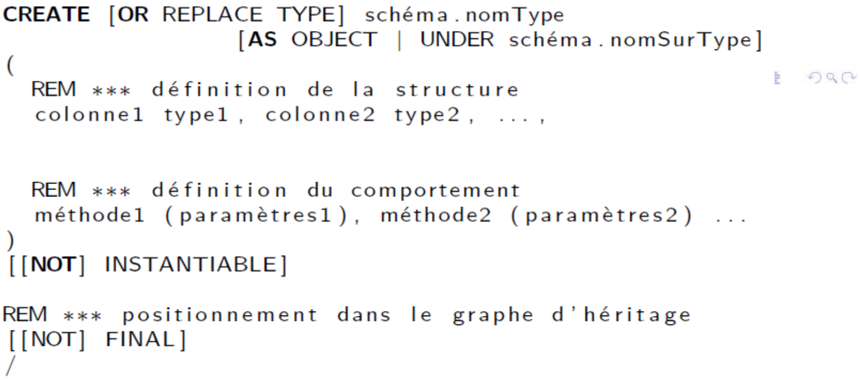
Object Types

- Definition of each object from a type describing
  - a data structure positioning in an inheritance hierarchy
  - methods
- Using a Type:
  - Build More types
  - Define one or more object tables
  - Define a column in a relational table
  - Building object views

Object Types

Creating a type

- Creation



```
CREATE [OR REPLACE TYPE] schéma.nomType
                    [AS OBJECT | UNDER schéma.nomSurType]
(
  REM *** définition de la structure
  colonne1 type1, colonne2 type2, ...,


  REM *** définition du comportement
  méthode1 (paramètres1), méthode2 (paramètres2) ...
)
[[NOT] INSTANTIABLE]

REM *** positionnement dans le graphe d'héritage
[[NOT] FINAL]
/
```

Creating a type

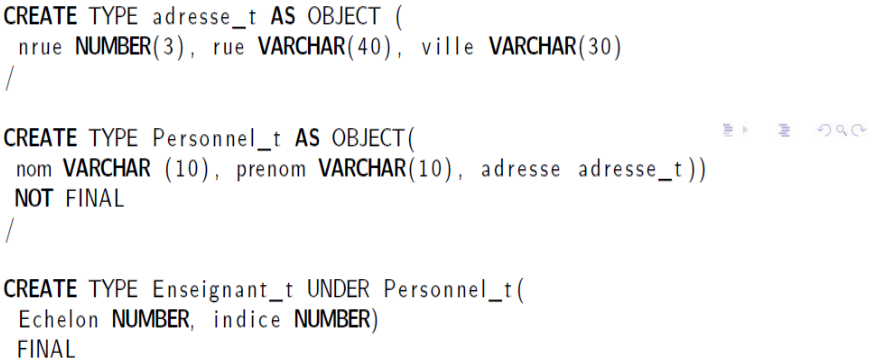
FINAL directive

- FINAL and NOT FINAL: positioning of a type in the inheritance graph
- NOT Final: to be applied to generic types
- By default, any type is FINAL

  A FINAL type can be used to define subtypes

Creating a type

FINAL Directive - Examples



```
CREATE TYPE adresse_t AS OBJECT (
 nrue NUMBER(3), rue VARCHAR(40), ville VARCHAR(30)
/

CREATE TYPE Personnel_t AS OBJECT(
 nom VARCHAR (10), prenom VARCHAR(10), adresse adresse_t))
 NOT FINAL
/

CREATE TYPE Enseignant_t UNDER Personnel_t(
 Echelon NUMBER, indice NUMBER)
 FINAL
```

Creating a type

INSTANTIABLE directive

- INSTANTIABLE and NOT INSTANTIABLE: to instantiate a type
  All types are created by default INSTANTIABLE
- NOT INSTANTIABLE: similar to the concept of abstract class
- Each type has
  - a constructor to create objects (persistent or not) with the NEW command or in an INSERT clause
  - a constructor (default) and several in the case of overload
- One type NOT INSTANTIABLE can not be FINAL
- A subtype NOT INSTANTIABLE can inherit a type INSTANTIABLE

Creating a type

INSTANTIABLE Directive - Examples

```
CREATE TYPE Personnel_t AS OBJECT(
 nom VARCHAR (10), prenom VARCHAR(10), adresse adresse_t))
 NOT INSTANTIABLE NOT FINAL
/

CREATE TYPE Enseignant_t UNDER Personnel_t(
 Echelon NUMBER, indice NUMBER)
 INSTANTIABLE FINAL
/
```

Object Types

Deleting a type

DROP TYPE nomType [FORCE | VALIDATE] ;

- FORCE: remove a type even if there exist objects belong to this type in a database
  Oracle : columns of this type, are labeled “UNUSED”, and they become inaccessible (not recommended)
- VALIDATE: Check if instances of the type to be deleted can be substituted by a superclass.

example:

DROP TYPE Personnel_t FORCE

Object Types

Creating a type

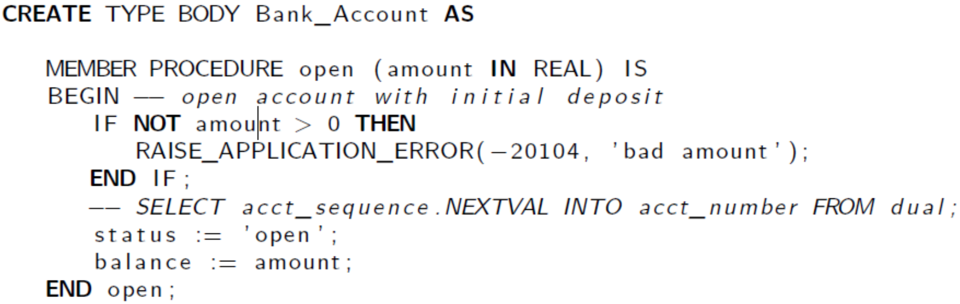
Specifying the Object

```
CREATE TYPE Bank_Account AS OBJECT (
    acct_number INTEGER(5),
    balance           REAL,
    status            VARCHAR2(10),

    MEMBER PROCEDURE open
          (amount IN REAL),

    MEMBER PROCEDURE verify_acct
          (num IN INTEGER),

    MEMBER PROCEDURE close
          (num IN INTEGER, amount OUT REAL)
);

CREATE TYPE BODY Bank_Account AS
    ...
END;
```

Object Types

Creating a type

Definition of methods associated with the object



```
CREATE TYPE BODY Bank_Account AS

   MEMBER PROCEDURE open (amount IN REAL) IS
   BEGIN -- open account with initial deposit
      IF NOT amount > 0 THEN
         RAISE_APPLICATION_ERROR(-20104, 'bad amount');
      END IF;
      -- SELECT acct_sequence.NEXTVAL INTO acct_number FROM dual;
      status := 'open';
      balance := amount;
   END open;
```

Object Types

Creating a type

```
    MEMBER PROCEDURE verify_acct (num IN INTEGER) IS
    BEGIN -- check for wrong account number or closed account
        IF (num <> acct_number) THEN
            RAISE_APPLICATION_ERROR(-20105, 'wrong number');
        ELSIF (status = 'closed') THEN
                RAISE_APPLICATION_ERROR(-20106, 'account closed');
        END IF;
    END verify_acct;

    MEMBER PROCEDURE close (num IN INTEGER, amount OUT REAL) IS
    BEGIN -- close account and return balance
        verify_acct(num);
        status := 'closed';
        amount := balance;
    END close;
END;
```

Object Types

Extraction of the description of a type

Defining new views to account types

example:

```
create type emp_type as object ( ninsee varchar2 ( 1 3 ) ,
                                 age number , nom varchar2 ( 3 0 ) ) har 2 (3 0))
/
```

Description of the structure of the 1st level of a type:

```
SQL> DESC emp_type
```

Object Types

Extraction of the description of a type

Examples of views: (USER _..., DBA _..., ALL _...)

Description:

- collections: USER_COLL_TYPES
- indexes on the types: USER_INDEXTYPES
- types in general: USER_TYPES
- Attribute types: USER_TYPE_ATTRS
- Methods types: USER_TYPE_METHODS
- Revision types: USER_TYPE_VERSIONS

To the object

relational tables

```
-- Table : MAGASINS2 SQL2
create table MAGASINS2
(   NUMMAG          INTEGER       ,
    NOMMAG          CHAR(30)      ,
    TELMAG          CHAR(15)      ,
    ADRNUMMAG       VARCHAR2(10),
    ADRRUEMAG       VARCHAR2(50),
    ADRCPMAG        VARCHAR2(10),
    ADRVILLEMAG     VARCHAR2(50),
    ADRPAYSMAG      VARCHAR2(50),
    constraint PK_MAGASINS2
        primary key (NUMMAG) );
```

```
-- Table : CLIENTS2 SQL2
create table CLIENTS2
(   NUMCLI          INTEGER       ,
    NOMCLI          CHAR(20)      ,
    TELCLI          CHAR(15)      ,
    ADRNUMCLI       VARCHAR2(10),
    ADRRUECLI       VARCHAR2(50),
    ADRCPCLI        VARCHAR2(10),
    ADRVILLECLI     VARCHAR2(50),
    ADRPAYSCLI      VARCHAR2(50),
    constraint PK_CLIENTS2
        primary key (NUMCLI));
```

```
insert into MAGASINS2 values (1, 'FB', '0145454545', '13', 'Avenue de la paix',
                              '75015', 'Paris', 'France');
```

```
NUMMAG NOMMAG    TELMAG      ADRNU ADRRUEMAG             ADRCP ADRVILLEMA  ADRPAYSMAG
 1        FB     0145454545   13   Avenue de la paix     75015   Paris       France
 2        FB     0155555555   20   Avenue de la liberté  06100   Nice        France
 3        FB     0155555555   10   Avenue des Amis        6050   Bruxelles   Belgique
 4        FB     71226002     10   Avenue du soleil       1001   Tunis       Tunisie
```

```
NUMCLI NOMCLI   TELCLI      ADRNU ADRRUECLI               ADRCP ADRVILLECL  ADRPAYSCLI
 1     TRAIFOR  0645454545  13    Avenue de la paix       75015   Paris       France
 2     CLEMENT  0607080910  17    Avenue de la paix       75015   Paris       France
 3     SOUCY    98980307    77    Route de la corniche     4001   Sousse      Tunisie
```

Object Tables

Creating a type - TAD

First extension of the relational model: Abstract Data Types (ADT)

TAD (BD context):

- New attribute type defined by the user
- shared data structure
  - Use of the type in one or more tables
  - Participation in the composition of one or more other types

Remarks:

- A TAD includes methods that are procedures or functions
- They allow you to manipulate objects of the abstract type

Object Tables

Creating a type – example of TAD

```
create  type ADRESSE_TYPE as object
  ( ADRNUM      VARCHAR2(10),
    ADRRUE      VARCHAR2(50),
    ADRCP       VARCHAR2(10),
    ADRVILLE    VARCHAR2(50),
    ADRPAYS     VARCHAR2(50)  )
/
```

```
create type MAG_TYPE as object
(     NUMMAG    INTEGER  ,
      NOMMAG    CHAR(30),
      TELMAG    CHAR(15),
      ADRMAG    ADRESSE_TYPE )
/
```

```
create type CLI_TYPE as object
(     NUMCLI    INTEGER  ,
      NOMCLI    CHAR(30),
      TELCLI    CHAR(15),
      ADRCLI    ADRESSE_TYPE )
/
```

Object Tables

Creating a table - Examples

```
create table MAGASINS3 OF MAG_TYPE
( constraint PK_MAGASINS3 primary key (NUMMAG) );

create table CLIENTS3 OF CLI_TYPE
( constraint PK_CLIENTS3  primary key (NUMCLI) );
```

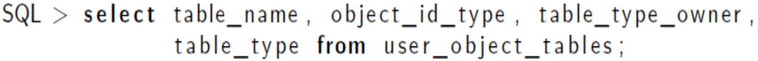
Object Tables

Creating a type

Remarks:

- One type can not contain constraints (NOT NULL, CHECK, UNIQUE, DEFAULT, PRIMARY KEY, FOREIGN KEY, etc.).
- The constraints must be declared at the table object

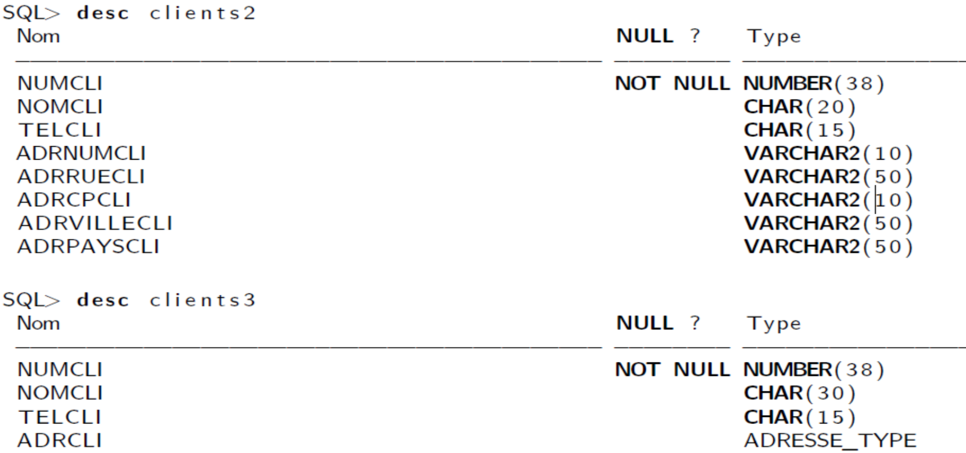
Access to the description of the types from the Data Dictionary:



```
SQL > select table_name, object_id_type, table_type_owner,
             table_type from user_object_tables;
```

Object Tables

Creation / description of a table



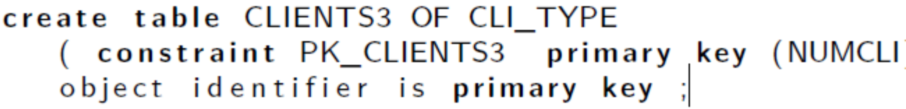
```
SQL> desc clients2
 Nom                                                 NULL ?     Type
 --------------------------------------------------- ---------- ---------------
 NUMCLI                                              NOT NULL   NUMBER(38)
 NOMCLI                                                         CHAR(20)
 TELCLI                                                         CHAR(15)
 ADRNUMCLI                                                      VARCHAR2(10)
 ADRRUECLI                                                      VARCHAR2(50)
 ADRCPCLI                                                       VARCHAR2(10)
 ADRVILLECLI                                                    VARCHAR2(50)
 ADRPAYSCLI                                                     VARCHAR2(50)

SQL> desc clients3
 Nom                                                 NULL ?     Type
 --------------------------------------------------- ---------- ---------------
 NUMCLI                                              NOT NULL   NUMBER(38)
 NOMCLI                                                         CHAR(30)
 TELCLI                                                         CHAR(15)
 ADRCLI                                                         ADRESSE_TYPE
```

Object Tables

Object identifier (OID)

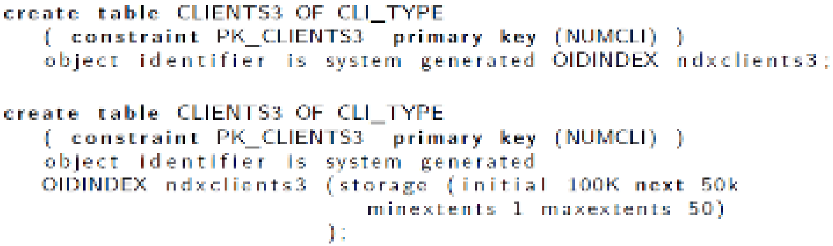
- OID based on the primary key: Using the primary key option

  example:



```
create table CLIENTS3 OF CLI_TYPE
   ( constraint PK_CLIENTS3  primary key (NUMCLI
   object identifier is primary key ;
```

- OID Indexes:

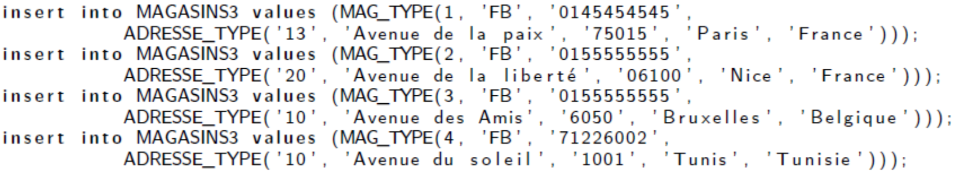


```
create table CLIENTS3 OF CLI_TYPE
   ( constraint PK_CLIENTS3  primary key (NUMCLI) )
   object identifier is system generated OIDINDEX ndxclients3;

create table CLIENTS3 OF CLI_TYPE
   ( constraint PK_CLIENTS3  primary key (NUMCLI) )
   object identifier is system generated
   OIDINDEX ndxclients3 (storage (initial 100K next 50k
                           minextents 1 maxextents 50)
                        );
```

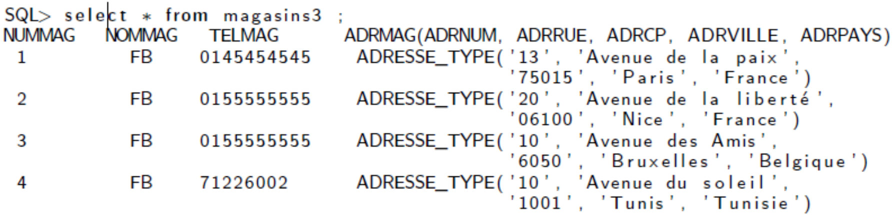
Object Tables

Instantiation - examples

Inserting a "line" (or rather an object):



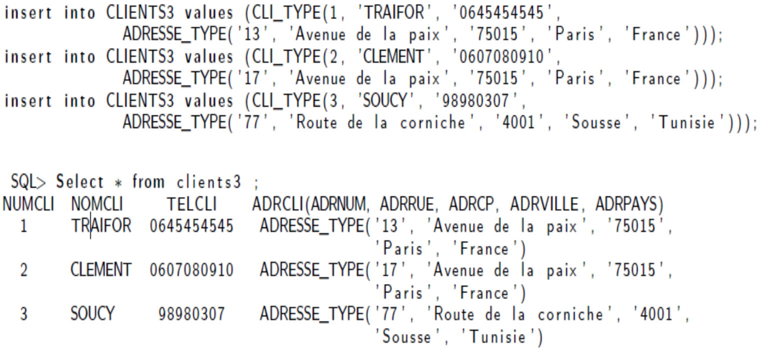
```
insert into MAGASINS3 values (MAG_TYPE(1, 'FB', '0145454545',
        ADRESSE_TYPE('13', 'Avenue de la paix', '75015', 'Paris', 'France')));
insert into MAGASINS3 values (MAG_TYPE(2, 'FB', '0155555555',
        ADRESSE_TYPE('20', 'Avenue de la liberté', '06100', 'Nice', 'France')));
insert into MAGASINS3 values (MAG_TYPE(3, 'FB', '0155555555',
        ADRESSE_TYPE('10', 'Avenue des Amis', '6050', 'Bruxelles', 'Belgique')));
insert into MAGASINS3 values (MAG_TYPE(4, 'FB', '71226002',
        ADRESSE_TYPE('10', 'Avenue du soleil', '1001', 'Tunis', 'Tunisie')));
```



```
SQL> select * from magasins3 ;
NUMMAG   NOMMAG   TELMAG       ADRMAG(ADRNUM, ADRUE, ADRCP, ADRVILLE, ADRPAYS)
 1         FB     0145454545    ADRESSE_TYPE('13', 'Avenue de la paix',
                                             '75015', 'Paris', 'France')
 2         FB     0155555555    ADRESSE_TYPE('20', 'Avenue de la liberté',
                                             '06100', 'Nice', 'France')
 3         FB     0155555555    ADRESSE_TYPE('10', 'Avenue des Amis',
                                             '6050', 'Bruxelles', 'Belgique')
 4         FB     71226002      ADRESSE_TYPE('10', 'Avenue du soleil',
                                             '1001', 'Tunis', 'Tunisie')
```

Object Tables

Instantiation - examples



```
insert into CLIENTS3 values (CLI_TYPE(1, 'TRAIFOR', '0645454545',
            ADRESSE_TYPE('13', 'Avenue de la paix', '75015', 'Paris', 'France')));
insert into CLIENTS3 values (CLI_TYPE(2, 'CLEMENT', '0607080910',
            ADRESSE_TYPE('17', 'Avenue de la paix', '75015', 'Paris', 'France')));
insert into CLIENTS3 values (CLI_TYPE(3, 'SOUCY', '98980307',
            ADRESSE_TYPE('77', 'Route de la corniche', '4001', 'Sousse', 'Tunisie')));
```

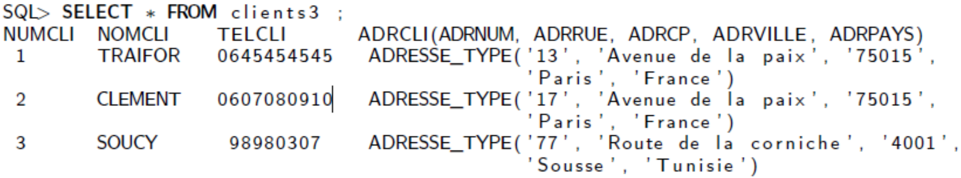
```
 SQL> Select * from clients3 ;
NUMCLI  NOMCLI    TELCLI     ADRCLI(ADRNUM, ADRRUE, ADRCP, ADRVILLE, ADRPAYS)
  1     TRAIFOR  0645454545  ADRESSE_TYPE('13', 'Avenue de la paix', '75015',
                                          'Paris', 'France')
  2     CLEMENT  0607080910  ADRESSE_TYPE('17', 'Avenue de la paix', '75015',
                                          'Paris', 'France')
  3     SOUCY     98980307   ADRESSE_TYPE('77', 'Route de la corniche', '4001',
                                          'Sousse', 'Tunisie')
```

Object Tables

instantiation

object-relational table:

- dependent table of a type
- Records (rows) in this table seen as objects because they all have a single OID (Object Identifier)

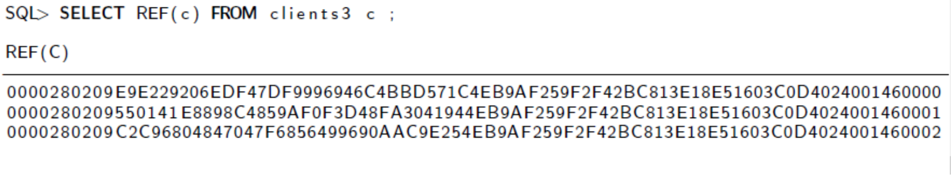


```
SQL> SELECT * FROM clients3 ;
NUMCLI  NOMCLI    TELCLI      ADRCLI(ADRNUM, ADRRUE, ADRCP, ADRVILLE, ADRPAYS)
 1      TRAIFOR   0645454545   ADRESSE_TYPE('13', 'Avenue de la paix', '75015',
                                            'Paris', 'France')
 2      CLEMENT   0607080910   ADRESSE_TYPE('17', 'Avenue de la paix', '75015',
                                            'Paris', 'France')
 3      SOUCY      98980307    ADRESSE_TYPE('77', 'Route de la corniche', '4001',
                                            'Sousse', 'Tunisie')
```

Object Tables

instantiation

- Reference OID objects of the table:



```
SQL> SELECT REF(c) FROM clients3 c ;

REF(C)
--------------------------------------------------------------------------------
0000280209E9E229206EDF47DF9996946C4BBD571C4EB9AF259F2F42BC813E18E51603C0D4024001460000
0000280209550141E8898C4859AF0F3D48FA3041944EB9AF259F2F42BC813E18E51603C0D4024001460001
0000280209C2C96804847047F6856499690AAC9E254EB9AF259F2F42BC813E18E51603C0D4024001460002
```

Object Tables

Updates

Changes / Deletions of "lines" or objects

- Updating a standard column

```
update clients3
set NOMCLI = ’CBON’ where NUMCLI=2;
```

- Changing a column belonging to a nested type

```
update clients3 c
set c . ADRCLI . ADRVILLE = ’MAVILLE ’ where c .NUMCLI=2;
```

- Deleting object

```
delete from clients3
where numcli = 3 ;
delete from clients3 c
where upper ( c . ADRCLI .ADRPAYS) = ’FRANCE ’ ;
```

Object Tables

interrogations

- Use of standard columns

```
select numcli , nomcli from clients3 ;
     NUMCLI NOMCLI
         ------ ----------
              1   TRAIFOR
              2 CLEMENT
              3 SOUCY
```

- Use of a column belonging to a nested type

```
select numcli , nomcli , c. ADRCLI. ADRPAYS
from clients3 c ;
     NUMCLI CUSTNAME      ADRCLI. ADRPAYS
        ---------- ---------- -          -----------------
               1  TRAIFOR         La France
               2 CLEMENT          La France
               3 SOUCY             T unisia
```

Subject Tables

interrogations

- with formatting

```
A10 name collar size
He co oc                 A15 size
select    numcli    ace cli, CUSTNAME            ace  name,
   c. ADRCLI. ADRVILLE | | '' | |                c. ADRCLI.        ac     loc
   ADRPAYS                                                         e
from   3 customers c;

---------- ---------- ---------------
                1 TRAIFOR        P aris    La France
                2 CLEMENT        P aris    La France
                3 SOUCY         S ousse    T unisia
```

Subject Tables

interrogations

- constrained

```
SQL>  COLC. ADRCLI. ADRPAYS     format    A10
SQL>  COLC. ADRCLI. ADRVILLE    format      A10
SQL>  select  numcli,           CUSTNAME, c. ADRCLI. ADRPAYS,
   2  c. ADRCLI. ADRVILLE from customers 3      c
   3  WHERE upper (C. ADRCLI. ADRVILLE)      like  'P%';

    NUMCLI CUSTNAME     ADRCLI. ADRPAYS      ADRCLI. ADRVILLE
---------- ---------- --------------- ---------------
           1 TRAIFOR        La France                 P aris
           2 CLEMENT        La France                 P aris
```

part V

-

nested tables in SQL3

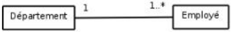
nested Tables

(NESTED TABLE)

nested table (NESTED TABLE): unordered collection rather limited of the same type elements

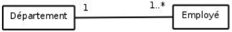
Example: table Department

NumDep	Budget	employees		
		NInsee	Name	Age



One table containing a column (table):

Association of Type 1-N



? ? ? 1 or more tables NN type Association

Nested Tables (NESTED TABLE)

Creation

```
create        type    emp_type asobject
              (Ninseevarchar 2 (1 3), age           number , Name varchar 2 (3 0))
/

create        type    emps_type astableof            emp_type
/
create        type    _ Department typeasobject
              (Varchar numdep 2 (1 1), the budget number,
                employed emps_type)
/
CreateTable           departementofdepartem ent types _
              ( primary key  (Numdep))
                nested   table    employee ss torus          ace  tabemp
/
```

- NESTED TABLE clause: definition of a nested table
- STORE AS clause: Naming the internal structure that stores the "records" of this nested table

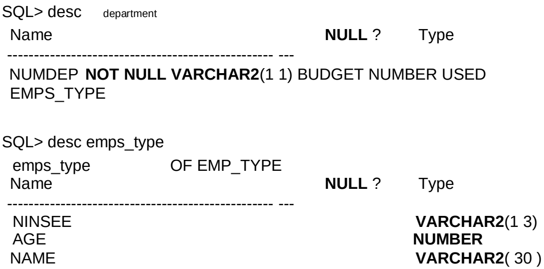
Nested Tables (NESTED TABLE)

Example

```
 create    type   emp_type      ace  object  (Ninseevarchar 2 (1 3),
                                       age number, name varchar 2 (3 0))
 /
 create    type   emps_type       astable  emp_type of
 /
 create    type   _ Department typeas         object       (Numdep varchar 2 (1 1),
                                        budget  number ,      employed emps_type)
 /
 CreateTable      departementofdepartem ent types _
                                      (primary  key  (Numdep))
nested    table    employee ss torus              ace  tabemp
 /
```

Nested Tables (NESTED TABLE)

Example

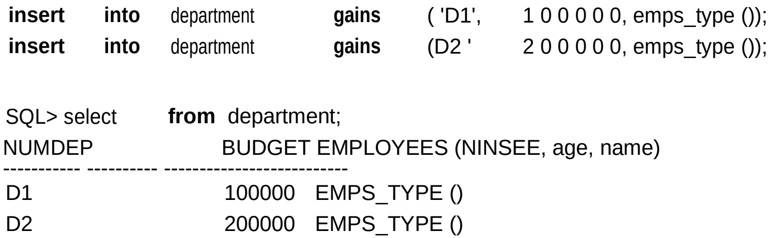


```
SQL> desc    department
 Name                                              NULL ?      Type
 -------------------------------------------------- ---
 NUMDEP  NOT NULL VARCHAR2(1 1) BUDGET NUMBER USED
 EMPS_TYPE

SQL> desc emps_type
  emps_type            OF EMP_TYPE
 Name                                              NULL ?      Type
 -------------------------------------------------- ---
  NINSEE                                                      VARCHAR2(1 3)
  AGE                                                        NUMBER
 NAME                                                         VARCHAR2( 30 )
```

Nested Tables (NESTED TABLE)

Insertion

- Inserting data into a nested table



```
insert    into    department    gains    ( 'D1',    1 0 0 0 0 0, emps_type ());
insert    into    department    gains    (D2 '      2 0 0 0 0 0, emps_type ());

SQL> select      from  department;
NUMDEP                  BUDGET EMPLOYEES (NINSEE, age, name)
----------- ---------- --------------------------
D1                      100000   EMPS_TYPE ()
D2                      200000   EMPS_TYPE ()
```

Nested Tables (NESTED TABLE)

Insertion

- NB: In the following example, the vacuum table is uninitialized

```
insert      into   department (numdep,          budget)
                   gains    (D3 '      3 0 0 0 0 0);

SQL> select         from  department          ;
NUMDEP                   BUDGET EMPLOYEES (NINSEE, age, name)
----------- ---------- ---------------------------- ----
D1                         100000 EMPS_TYPE ()
D2                         200000 EMPS_TYPE ()
D3                         300000
```

Nested Tables (NESTED TABLE)

Insertion

- Inserting data into a nested table

```
insert nto                 department        gains  ( 'D4',      400000,
                 emps_type (emp_type ( 'N5'       25  'B ibi),
                            emp_type ( 'N6'       26  'C here),
                             emp_type (N7'        27  'D idi),
                            emp_type ( 'N8'      28,  'F ifi')));
```

Nested Tables (NESTED TABLE)

Insertion

```
SQL> select                  from  department;
NUMDEP                 BUDGET EMPLOYEES (NINSEE, age, name)
----------- ---------- -----------------------------
D1                     100000   EMPS_TYPE ()
D2                     200000   EMPS_TYPE ()
D3                     300000
D4                     400000    EMPS_TYPE (EMP_TYPE ( 'N5'   2 5  'B ibi),
                                            EMP_TYPE ( 'N6'   2 6  'C here),
                                            EMP_TYPE ( 'N7'   2 7  'D idi),
                                            EMP_TYPE ( 'N8'  2 8,  'F ifi'))
```

Note: INSERT command with the builders of the types of NESTED TABLE

- stores an object in the table
- initializes the nested table associated with records

Nested Tables (NESTED TABLE)

Insertion

- Inserting data into a nested table

```
InsertInto  departementvalues (D5 ', 4 0 0 0 0 0,
                    emps_type (emp_type ( 'N5',            2 5  'B ibi),
                                   emp_type ( 'N8'          2 8,  'F ifi')));

SQL> select        from  department;
NUMDEP                 BUDGET EMPLOYEES (NINSEE, age, name)
----------- ---------- ---------------------------- --------
D1                      100000   EMPS_TYPE ()
D2                      200000   EMPS_TYPE ()
D3                      300000
D4                      400000    EMPS_TYPE (EMP_TYPE ( 'N5'    2 5    'B ibi),
                                             EMP_TYPE ( 'N6'    2 6    'C here),
                                             EMP_TYPE ( 'N7'    2 7    'D idi),
                                             EMP_TYPE ( 'N8'   2 8,    'F ifi'))
D5                      400000    EMPS_TYPE (EMP_TYPE ( 'N5'    2 5    'B ibi),
                                             EMP_TYPE ( 'N8'   2 8,    'F ifi'))
```

Nested Tables (NESTED TABLE)

Integration with operator TABLE

- TABLE insertion with the operator in a nested table (D1 and D2 were initialized to empty)

```
insert     into    TABLE (Selectd. Employees from          department        d
                                     Where  d. numdep =     'D1')
                         gains          ( 'N1'  2 1    'CLEMENT');
insert     into    TABLE (Selectd. Employees from          department        d
                                     Where  d. numdep =     'D2')
                         gains           (N2 '  2 2,   'CLEMENTINE');
```

NB: the THE operator is obsolete and has been replaced by the TABLE operator

Nested Tables (NESTED TABLE)

Integration with operator TABLE

```
SQL> select          from  department           ;
NUMDEP          BUDGET EMPLOYEES (NINSEE, age, name)
-------- ---------- ------------------------------- -----------
D1                    100000       EMPS_TYPE (EMP_TYPE ( 'N1'    2 1  'CLEMENT'))
D2                    200000     EMPS_TYPE (EMP_TYPE (N2 '   2 2,  'CLEMENTINE'))
D3                    300000
D4                    400000 EMPS_TYPE (EMP_TYPE ( 'N5     '    2 5  'B ibi),
                                             EMP_TYPE ( 'N6     '    2 6  'C here),
                                              EMP_TYPE (N7      '    2 7  'D idi),
                                             EMP_TYPE ( 'N8     '   2 8,  'F ifi'))
```

Remarks:

- INSERT INTO TABLE (SELECT ...): storage of a record in the nested table designated TABLE
- SELECT after TABLE: Returns a single object, which selects the associated nested table

Nested Tables (NESTED TABLE)

Integration with operator TABLE

- Integration with the operator in a nested table TABLE

(D3 was not initialized to empty)
Inserting an employee in the department D3
while it did not initialize

```
insert      into TABLE (Selectd. Employees from departementd
              Where  d. numdep = 'D3')
              gains      ( 'N3', 2 3, 'DOES NOT');
```

Nested Tables (NESTED TABLE)

Integration with operator TABLE

```
SQL> insert      into     table          (Selectd. Employees  from      departementd
   2 Where       d. numdep  =  'D3')    gains            ( 'N3',  2 3,    'DO NOT WORK ' ) ;
insert      into   table            (Selectd. Employees from departementd
                                                   Where  d. numdep = 'D3')

FAULT     to the   line        1   :

ORA-22908:    reference     at    a     value     of table       NULL
```

explanations:

1. The D3 department is an object of the table Department
2. but it does not have nested table
3. because it was not created during insertion. We

must destroy the D3 object and recreate it!

Nested Tables (NESTED TABLE)

change

- update the main table

```
update    departementd
          set d. Budget = d. budge t1. 5
                    Where  d. Budget <=      200000;

SQL>         selec tfrom  department          ;
NUMDEP        BUDGET EMPLOYEES (NINSEE, age, name)
-------- -------- -------------------------------- ---------
D1             150000     EMPS_TYPE (EMP_TYPE ( 'N1'    2 1   'CLEMENT'))
D2             300000      EMPS_TYPE (EMP_TYPE (N2 '   2 2,   'CLEMENTINE'))
D3             300000
D4             400000     EMPS_TYPE (EMP_TYPE ( 'N5'    2 5   'B ibi),
                                    EMP_TYPE ( 'N6'    2 6   'C here),
                                    EMP_TYPE ( 'N7'    2 7   'D idi),
                                    EMP_TYPE ( 'N8'   2 8,   'F ifi'))
D5             400000     EMPS_TYPE (EMP_TYPE ( 'N5'    2 5   'B ibi),
                                    EMP_TYPE ( 'N8'   2 8,   'F ifi'))
```

Nested Tables (NESTED TABLE)

Editing (continued)

- Update from the main table as a predicate in the nested table

```
update    departementdsetd. Budget = d. Budget + 777
            wheree xists                    (select       from
            table    (   select    dt. employees from department           dt
                         Where  dt. numdep =
                         d. numdep)      nt
          Where  nt. age <25)             ;
```

Description:

- Query that returns the employees in each department

```
select      dt. employees      from  departementdt
                         Wher    dt. numdep = d. numdep
                         e
```

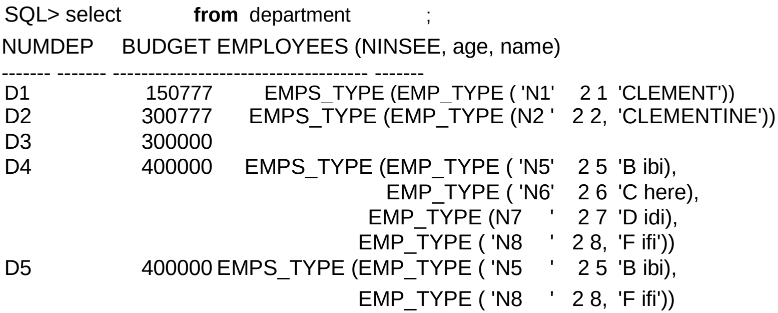
- Condition on an attribute of the nested table:

```
              Where  nt. age <25
```

- Alias of the nested table: nt

Nested Tables (NESTED TABLE)

Editing (continued)

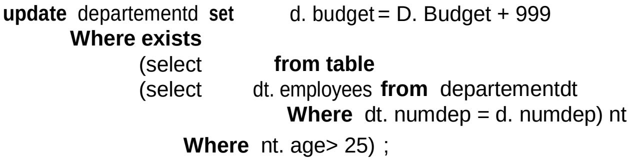


```
SQL> select         from  department          ;
NUMDEP    BUDGET EMPLOYEES (NINSEE, age, name)
------- ------- ------------------------------------ -------
D1              150777       EMPS_TYPE (EMP_TYPE ( 'N1'    2 1  'CLEMENT'))
D2              300777     EMPS_TYPE (EMP_TYPE (N2 '   2 2,  'CLEMENTINE'))
D3              300000
D4              400000    EMPS_TYPE (EMP_TYPE ( 'N5'    2 5  'B ibi),
                                                EMP_TYPE ( 'N6'    2 6  'C here),
                                              EMP_TYPE (N7     '   2 7  'D idi),
                                             EMP_TYPE ( 'N8     '  2 8,  'F ifi'))
D5              400000 EMPS_TYPE (EMP_TYPE ( 'N5     '   2 5  'B ibi),
                                             EMP_TYPE ( 'N8     '  2 8,  'F ifi'))
```

Nested Tables (NESTED TABLE)

change

- Update from the main table as a predicate in the nested table



```
update  departementd  set        d. budget = D. Budget + 999
        Where exists
                 (select         from table
                 (select      dt. employees  from  departementdt
                                   Where  dt. numdep = d. numdep) nt
                         Where  nt. age> 25)  ;
```

Nested Tables (NESTED TABLE)

change

```
SQL> select          from  department          ;
NUMDEP    BUDGET EMPLOYEES (NINSEE, age, name)
------- ------- ------------------------------------ -------
D1             150777       EMPS_TYPE (EMP_TYPE ( 'N1'    2 1  'CLEMENT'))
D2             300777     EMPS_TYPE (EMP_TYPE (N2 '   2 2,  'CLEMENTINE'))
D3             300000
D4             400999    EMPS_TYPE (EMP_TYPE ( 'N5'    2 5  'B ibi),
                                       EMP_TYPE ( 'N6'    2 6  'C here),
                                     EMP_TYPE (N7     '    2 7  'D idi),
                                    EMP_TYPE ( 'N8     '   2 8,  'F ifi'))
D5             400999 EMPS_TYPE (EMP_TYPE ( 'N5     '    2 5  'B ibi),
                                    EMP_TYPE ( 'N8     '   2 8,  'F ifi'))
```

Note: the same employees are in two departments

Nested Tables (NESTED TABLE)

change

- Update in the nested table

```
update
          table        (select      d. employees from           departementd
                                              Where   d. numdep = D2 ')          nt
          set
                   Where  nt. ninsee = 'N2'             ;
```

Nested Tables (NESTED TABLE)

change

```
SQL> select          from  department          ;
NUMDEP     BUDGET EMPLOYEES (NINSEE, age, name)
------- ------- ------------------------------------ -------
D1            150777        EMPS_TYPE (EMP_TYPE ( 'N1'    2 1  'CLEMENT'))
D2            300777     EMPS_TYPE (EMP_TYPE (N2 '    4 4  'CLEMENTINE'))
D3            300000
D4            400999     EMPS_TYPE (EMP_TYPE ( 'N5'    2 5  'B ibi),
                                      EMP_TYPE ( 'N6'    2 6  'C here),
                                    EMP_TYPE (N7     '    2 7  'D idi),
                                  EMP_TYPE ( 'N8     '   2 8,  'F ifi'))
D5            400999 EMPS_TYPE (EMP_TYPE ( 'N5     '    2 5  'B ibi),

                                  EMP_TYPE ( 'N8     '   2 8,  'F ifi'))
```

Note: It is impossible to change several diff erent records nested tables

with a single UPDATE!

Nested Tables (NESTED TABLE)

suppression

- Delete in the main table

```
delete      from   department
            Wher    numdep = 'D3';
            e
SQL> select         from  department           ;
NUMDEP     BUDGET EMPLOYEES (NINSEE, age, name)
                  -------- -------- ----------------------------------- -------
D1              150777      EMPS_TYPE (EMP_TYPE ( 'N1'    2 1  'CLEMENT'))
D2              300777    EMPS_TYPE (EMP_TYPE (N2 '    4 4  'CLEMENTINE'))
D4              400999    EMPS_TYPE (EMP_TYPE ( 'N5'    2 5  'B ibi),
                                   EMP_TYPE ( 'N6'    2 6  'C here),
                                  EMP_TYPE (N7     '    2 7  'D idi),
                                 EMP_TYPE ( 'N8     '   2 8,  'F ifi'))
D5              400999 EMPS_TYPE (EMP_TYPE ( 'N5     '    2 5  'B ibi),
                                 EMP_TYPE ( 'N8     '   2 8,  'F ifi'))
```

Nested Tables (NESTED TABLE)

suppression

- Deleting from a value of the nested table Elimination of departments that employ a person whose name is FIFI

```
delete from  departementd
          Where   exists     (select        from
          table   (select              dt. employees from department         dt
                                   Where  dt. numdep = d. numdep)          nt
          Where   upper (Nt. Name)         like  '%% FIFI');
```

Nested Tables (NESTED TABLE)

suppression

```
SQL> select          **from** department          ;
NUMDEP    BUDGET EMPLOYEES (NINSEE, age, name)
------- ------- ----------------------------------- -------
D1               150777 EMPS_TYPE (EMP_TYPE ( 'N1'         2 1  'CLEMENT'))
D2               300777 EMPS_TYPE (EMP_TYPE (N2 '          4 4  'CLEMENTINE'))
```

Nested Tables (NESTED TABLE)

suppression

- Deleting a nested table

  Elimination of departments that employ a person whose name CLEMENT

```
delet and reliable            (select    dt. employees
                   from departementdt where where    dt. numdep = D1 ')     nt
                   nt. name = 'CLEMENT';
```

```
SQL> select        from  department        ;
NUMDEP     BUDGET EMPLOYEES (NINSEE, age, name)
------- ------- ----------------------------------- -------
D1              150777
D2              300777 EMPS_TYPE (EMP_TYPE (N2 ', 4 4' CLEMENTINE '))
```

Nested Tables (NESTED TABLE)

interrogation

- *What are the numbers and employee names D4 department?*

```
SQL> select          from  department          ;
NUMDEP    BUDGET EMPLOYEES (NINSEE, age, name)
                  ------- ------- ---------------------------------- -------
D1             150777      EMPS_TYPE (EMP_TYPE ( 'N1'    2 1  'CLEMENT'))
D2             300777    EMPS_TYPE (EMP_TYPE (N2 '    4 4  'CLEMENTINE'))
D4             400999    EMPS_TYPE (EMP_TYPE ( 'N5'    2 5  'B ibi),
                                   EMP_TYPE ( 'N6'    2 6  'C here),
                                  EMP_TYPE (N7     '    2 7  'D idi),
                                 EMP_TYPE ( 'N8     '   2 8,  'F ifi'))
D5             400999 EMPS_TYPE (EMP_TYPE ( 'N5     '    2 5  'B ibi),
                                 EMP_TYPE ( 'N8     '   2 8,  'F ifi'))
```

Nested Tables (NESTED TABLE)

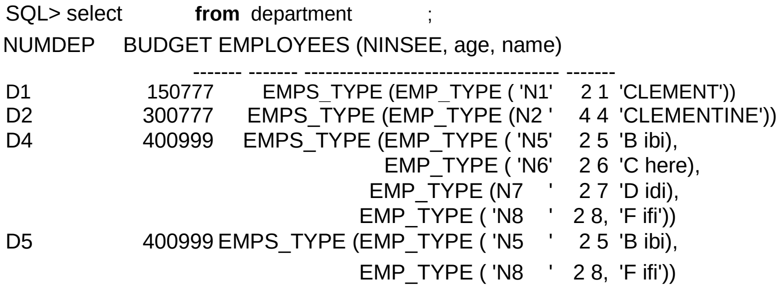
interrogation

```
select    nt. ninsee,      nt. name
          from table     (select    dt. employees from         department        dt
          Where  dt. numdep = 'D4') nt;
NINSEE              NAME
----------------------------- -------------
N5                   B ibi
N6                   C here
N7                   D idi
N8                    F ifi
```

Nested Tables (NESTED TABLE)

interrogation

- *What are the numbers and employee names D4 department with less than 26 years?*



```
SQL> select          from  department          ;
NUMDEP    BUDGET EMPLOYEES (NINSEE, age, name)
                    ------- ------- ----------------------------------- -------
D1             150777     EMPS_TYPE (EMP_TYPE ( 'N1'   2 1  'CLEMENT'))
D2             300777    EMPS_TYPE (EMP_TYPE (N2 '   4 4  'CLEMENTINE'))
D4             400999    EMPS_TYPE (EMP_TYPE ( 'N5'   2 5  'B ibi),
                                     EMP_TYPE ( 'N6'   2 6  'C here),
                                    EMP_TYPE (N7    '   2 7  'D idi),
                                   EMP_TYPE ( 'N8    '  2 8,  'F ifi'))
D5             400999 EMPS_TYPE (EMP_TYPE ( 'N5    '   2 5  'B ibi),
                                   EMP_TYPE ( 'N8    '  2 8,  'F ifi'))
```

Nested Tables (NESTED TABLE)

interrogation

```
select     nt. ninsee,        nt. name
           from table      (select       dt. employees from department                 dt
           Where  dt. numdep = 'D4') Where nt nt. age <2 6;
NINSEE                NAME
----------------------------- -------------
N5                     B ibi
```

Nested Tables (NESTED TABLE)

interrogation

- Query: What is the number of employees D4 department?

```
SQL> select          from  department          ;
NUMDEP     BUDGET EMPLOYEES (NINSEE, age, name)
                          ------- ------- ---------------------------------- -------
D1             150777      EMPS_TYPE (EMP_TYPE ( 'N1'    2 1  'CLEMENT'))
D2             300777     EMPS_TYPE (EMP_TYPE (N2 '    4 4  'CLEMENTINE'))
D4             400999    EMPS_TYPE (EMP_TYPE ( 'N5'    2 5  'B ibi),
                                   EMP_TYPE ( 'N6'    2 6  'C here),
                                  EMP_TYPE (N7    '    2 7  'D idi),
                                 EMP_TYPE ( 'N8    '   2 8,  'F ifi'))
D5             400999 EMPS_TYPE (EMP_TYPE ( 'N5    '    2 5  'B ibi),
                                 EMP_TYPE ( 'N8    '   2 8,  'F ifi'))
```

Nested Tables (NESTED TABLE)

interrogation

```
select COUNT( ) " Number of employees "
        from table    (select   dt. employees     from    department    dt
        Where  dt. numdep = 'D4') nt           ;
Number of employees
        -----------------
                       4
```

Nested Tables (NESTED TABLE)

interrogation

- *What are the numbers and names of employees of departments D1 and D2?*

```
SQL> select          from  department          ;
NUMDEP    BUDGET EMPLOYEES (NINSEE, age, name)
                    ------- ------- ---------------------------------- -------
D1          150777     EMPS_TYPE (EMP_TYPE ( 'N1'    2 1  'CLEMENT'))
D2          300777    EMPS_TYPE (EMP_TYPE (N2 '    4 4  'CLEMENTINE'))
D4          400999    EMPS_TYPE (EMP_TYPE ( 'N5'    2 5  'B ibi),
                                     EMP_TYPE ( 'N6'    2 6  'C here),
                                    EMP_TYPE (N7     '    2 7  'D idi),
                                   EMP_TYPE ( 'N8    '   2 8,  'F ifi'))
D5          400999 EMPS_TYPE (EMP_TYPE ( 'N5    '    2 5  'B ibi),
                                   EMP_TYPE ( 'N8    '   2 8,  'F ifi'))
```

Nested Tables (NESTED TABLE)

interrogation

```
select          select  nt. ninsee,        nt. name from     table
               (Selectdt. Employees          from  department          dt
                                   Where  dt. numdep =       'D1')      nt
union
select         nt. ninsee, nt. name       from table
               (Selectdt. Employees          from  department          dt
                                   Where  dt. numdep =       'D2')      nt;
```

```
NINSEE                NAME
------------------------------ -------------
N1                     CLEMENT
N2                     CLEMENTINE
```

Several nested tables

Creation

Grouping nested tables and Teachers Training Course in the table

NUMC	title	professors		formations	
		Name	Specialty	spinner et	schedule

```
create      typeprof _ typeasobject
           (Name varchar 2 (3 0), specialty                    varchar 2 ( 30 ) )
/
create      typeprofs _ _ kind typeastableofprof
/
create      typeformation _ typeasobject
           (Filierevarchar 2 (3 0), time                       number (5))
/
create      typeformations typeastableofformatio _ n _ Type
/
```

Several nested tables

Creation

```
create     typeprof _ typeasobject
         (name      varchar 2 ( 30 ) ,   specialty          varchar 2 ( 30 ) )
/
create     typeprofs _ _ kind typeastableofprof
/
create     typeformation _ typeasobject
         (Filierevarchar 2 (3 0),              schedule      number (5))
/
create     typeformations typeastableofformatio _ n _ Type
/
create     typecours _ typeasobject
         (NUM
         C              varchar 2 (5)   titrevarchar 2 (1 5)
           professeursprofs _ type formationsformations _ type)
/
CreateTable  coursofcours _ kind
 (constraint          pk _ coursprimary          key  (NUMC))
   nestedtableprofesseur sstoreastabprofs,
   nested      table  formationsstoreastabf ormation;
```

Several nested tables

Insertion

Inserting an object in the Course table, without linking it to teachers or training

```
 InsertInto  coursvalues (BD 'Teacher Type   ' Data base ' ,
                                        _ (),   training _ the type ());
```

```
select        from  course;

NUMC TITLE                      FACULTY (NAME, SPECIALTY) FORMATIONS (DIE, SCHEDULE)
------------------------------------------------ ----------------------------
BD     B ases of PROFS_TYPE Data ()                          FORMATIONS_TYPE ()
```

Several nested tables

Insertion

- Insert, with VALUES in 2 nested tables:

```
insert    into     course     gains      ( 'DW'    'Data    WareHouse '

Teacher Type _ (_ prof template ( 'Clemence' BD '),

                  prof _ deviation (Adam 'BD')),

         _-type formations (training _ template ( 'M aster 1', 1 0 0),
                      _ training template ( 'Master          SIA ' 2 0 0),
                      training _ deviation (DEA          AIOC ' 2 0 0)     ));
```

Several nested tables

Insertion

```
select          from     course;

NUMC TITLE                    FACULTY (NAME, SPECIALTY)                        FORMATIONS (DIE, SCHEDULE)
------------------------------------------------- -----------------------------------------
 BD      B ases of PROFS_TYPE Data ()                                         FORMATIONS_TYPE ()
dwData WareHouse                                                              PROFS_TYPE (
       FORMATIONS_TYPE (
                              PROF_TYPE ( 'Clemence' BD '),                              FORMATION_TYPE ( 'M aster
                              1',
                              PROF_TYPE (Adam 'BD'))                                     FORMATION_TYPE ( 'M aster
                              2P'
                                                                                 FORMATION_TYPE ( 'M aster 2R
```

Several nested tables

Insertion

Insert, with VALUES in 2 nested tables

insert **into** course **gains** 'B ('BDA'
ases of data A VANC ed es'

Teachers _ kind (Prof. _ template ('Mercy', 'BD'),
Prof. _ kind ('T raifor' BD '), Prof. _ the type ('
The Good ',' BD '))

training _ kind (
_ training template ('M aster 2P' 2 0 0),
_ training template ('M aster 2R', 2 0 0)));

Several nested tables

Insertion

course

NUMC	title	professors		formations	
		Name	Specialty	spinneret	schedule
BD	Data base				
DW	DataWareHouse	Clemency	BD	master 1	100
		Adam	BD	Master 2P	200
				Master 2R	200
BDA	Advanced Data Bases	Clemency	BD	Master 2P	200
		Traifor	BD	Master 2R	200
		Good	BD		

The SQL chage a ffi + is very bad ...

```
 select      from  course;
NUMC   TITLE                    FACULTY (NAME, SPECIALTY)            FORMATIONS (DIE, SCHEDULE)
 BD    Data base                PROFS_TYPE ()                        FORMATIONS_TYPE ()
DW     Data WareHouse           PROFS_TYPE (PROF_TYPE ( 'Clemence' BD ') PROF_TYPE (Adam' BD '))
                                                                     FORMATIONS_TYPE (FORMATION_TYPE ( 'M aster
 FORMATION_TYPE ( 'M aster     2P ', 2 0 0), FORMATION_TYPE (' M aster 2R ', 2 0 0))
 BDA   Data base               A VANC ed es PROFS_TYPE (PROF_TYPE ( 'Mercy', 'BD'), PROF_TYPE (T raifor '
 PROF_TYPE ( 'The Good', 'BD'))
                                                                     FORMATIONS_TYPE (FORMATION_TYPE ( 'M aster
 FORMATION_TYPE ( 'M aster     2R ', 2 0 0))
```

Several nested tables

Insertion

Insert, and with TABLE VALUES in 2 nested tables Data recording: the Traifor Parisi and teachers teach BD

```
insert    into    table        (Selectc. Teachers          from     coursc
                           Where
                           NUMC      =  BD ')    gains      (T raifor '          ' IF ' ) ;
insert    into    table        (Selectc. Teachers          from     coursc
                           Where
                           NUMC      =  BD ')    gains      ( 'P Arisi'          'DM');


select          from   course;
```

NUMC	title	professors		formations	
		Name	Specialty	spinneret	schedule
BD	Data base	Traifor	IF		
		Parisi	DM		
DW	DataWareHouse	Clemency	BD	master 1	100
		Adam	BD	Master 2P	200
				Master 2R	200
BDA	Advanced Data Bases	Clemency	BD	Master 2P	200
		Traifor	BD	Master 2R	200
		Good	BD		

Several nested tables

Insertion

Insert, and with TABLE VALUES in 2 nested tables

The price BD:

- belongs to INFO1 curriculum
- requires an hourly volume of 70 hours

```
inser ti nt ot ble                    (select     c.          from  coursc
training                                                     ( 'INFO1'  7 0);
                Where  NUMC = 'BD')       gains
```

```
select        from  course;
```

NUMC	title	professors		formations	
		Name	Specialty	spinneret	schedule
BD	Data base	Traifor	IF	INFO1	70
		Parisi	DM		
DW	DataWareHouse	Clemency	BD	master 1	100
		Adam	BD	Master 2P	200
				Master 2R	200
BDA	Advanced Data Bases	Clemency	BD	Master 2P	200
		Traifor	BD	Master 2R	200
		Good	BD		

Several nested tables

Insertion

Insert, with TABLE and SELECT in 2 nested tables

The BD course must now be taught in all sectors concerned by DW material provided that it had a volume of less than 150 hours

```
inser ti nt ot ble                     (select            c. training   from  coursc
                                   Where  c. NUMC = 'BD')
           select    nestedf. filiere,           nestedf. schedule
                     from table        (select      c. training          from  course      c
                                        Where  c. NUMC = 'DW')      nestedf
          Where  nestedf. Zone <1 5 0;
```

Several nested tables

Insertion

selec tfrom course;

NUMC	title	professors		formations	
		Name	Specialty	spinneret	schedule
BD	Data base	Traifor	IF	INFO1	70
		Parisi	DM	master 1	
DW	DataWareHouse	Clemency	BD	master 1	100
		Adam	BD	Master 2P	200
				Master 2R	200
BDA	Advanced Data Bases	Clemency	BD	Master 2P	200
		Traifor	BD	Master 2R	200
		Good	BD		

Several nested tables

Modification - Example

In the Data WareHouse subject, Professor Adam is replaced by the Saitout professor and schedules for 2P Master increase of 30%

update table (Selectc. Where teachers from coursc v. Title = 'Data
WareHouse') nestedprfnestedprf. name = 'S aitou' where nestedprf.
name = set Adam ';

updatet reliable (select c. training **from** coursc
Where c. title = 'Data WareHouse') nestedfrm
set nestedfrm. time = time 1. 3
Where nestedfrm. filiere **like** 'M aster 2P%';

Several nested tables

change

NUMC	title	professors		formations	
		Name	Specialty	spinneret	schedule
BD	Data base	Traifor	IF	INFO1	70
		Parisi	DM	master 1	
DW	DataWareHouse	Clemency	BD	master 1	100
		Saitou	BD	Master 2P	**260**
				Master 2R	200
BDA	Advanced Data Bases	Clemency	BD	Master 2P	200
		Traifor	BD	Master 2R	200
		Good	BD		

Several nested tables

Modification - Example

explanations:

- Editing using the UPDATE of one or more attributes in one of two nested tables in the current table
- Changing a teachers and training as part of a given material: two distinct UPDATE requests (Because the two nested tables are involved)
- Requires the use of an alias to identify the object in the nested table

Several nested tables

change

For DW material, replacing the Master1 die through the die MASTER 2

and recording an hourly volume of 150 hours

```
        update table  (Select  c. training          from  coursc
                   Where c. Ti be = 'Data WareHouse')            nestedfrm

set     nestedfrm. time = 1 5 0,               nestedfrm. filiere = 'MASTER 2'
                Where  nestedfrm. filiere = 'M aster 1';
```

Several nested tables

change

NUMC	title	professors		formations	
		Name	Specialty	spinneret	schedule
BD	Data base	Traifor	IF	INFO1	70
DW	DataWareHouse	Parisi	DM	master 1	
		Clemency	BD	**MASTER 2**	**150**
		Saitou	BD	Master 2P	260
				Master 2R	200
BDA	Advanced Data Bases	Clemency	BD	Master 2P	200
		Traifor	BD	Master 2R	200
		Good	BD		

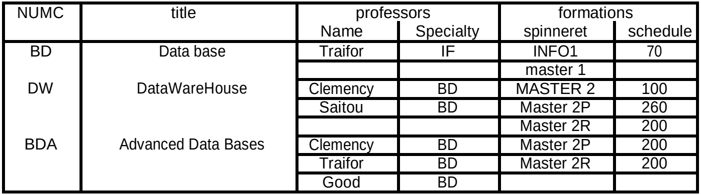
Several nested tables

suppression

Deleting in 2 nested tables

Professor Parisi no longer teaches comics material. Recording this information

```
DeleteTable  (Selectc. Teachers
                                        c.
               from   coursc      Where  NUMC    = 'BD')     nt
               Where  nt. name  = 'P Arisi';
```



NUMC	title	professors		formations	
		Name	Specialty	spinneret	schedule
BD	Data base	Traifor	IF	INFO1	70
				master 1	
DW	DataWareHouse	Clemency	BD	MASTER 2	100
		Saitou	BD	Master 2P	260
				Master 2R	200
BDA	Advanced Data Bases	Clemency	BD	Master 2P	200
		Traifor	BD	Master 2R	200
		Good	BD		

Several nested tables

suppression

Deleting in 2 nested tables

The sector includes not Master1 comics material in its curriculum. Recording this information

DeleteTable (Selectc. Training
from Where coursc c. NUMC = BD ') nt
Where nt. filiere = ' Master 1 ' ;

NUMC	title	professors		formations	
		Name	Specialty	spinneret	schedule
BD	Data base	Traifor	IF	INFO1	70
DW	DataWareHouse	Clemency	BD	MASTER 2	100
		Saitou	BD	Master 2P	260
				Master 2R	200
BDA	Advanced Data Bases	Clemency	BD	Master 2P	200
		Traifor	BD	Master 2R	200
		Good	BD		

Several levels of nesting

NUMC	title	professors		formations		
		Name	Specialty	spinneret	schedule	dates
						Day
BD	Data base	Traifor	IF	INFO1	70	
DW	DataWareHouse	Clemency	BD	MASTER 2	100	
		Saitou	BD	Master 2P	260	
				Master 2R	200	
BDA	Advanced Data Bases	Clemency	BD	Master 2P	200	
		Traifor	BD	Master 2R	200	
		Good	BD			

Oracle 8 does not allow to install several nesting levels in an object-relational table

? ? in Oracle 9i and / or 10g? ?

Pre-sized arrays (VARRAY)

- VARRAY (varrying ARRAY) ordered collection and limited items of its type
- If the maximum number of items contained in a nested table is known a priori
  possibility of using a VARRAY type of table instead of a nested table
- Example: storage of up to 3 telephone numbers per teacher

professors:

Nump	pname	Address				phones
		AdrNum	AdrRue	AdrVille	AdrCP	TEL NUMB R

Pre-sized size arrays (VARRAY)

Example

Storage of up to 3 telephone numbers per teacher

- Creation

```
create      AA stands _ typeasobject
            (AdrNum varchar 2 (1 0), AdrNom              varchar 2 ( 30 ) ,
              A V dr illevarchar 2 (2 0),        AdrCP       varchar 2 (5))
/
create      typetel _ typeasobject               (TEL NUMBR varchar 2 (2 0))
/

create      typetels typeasvarray _ (3) _ OFTEL Type
/

create      typeprofesseur _ typeasobject
(NUMP  varchar 2 (5)        pname varchar 2 (2 0),
               A stands AA stands _ type _ elephonestels T type)
/
CreateTable  professeursofprofesse kind heart _
            (constraint                       pk _ professeursprimary key (NUMP));
```

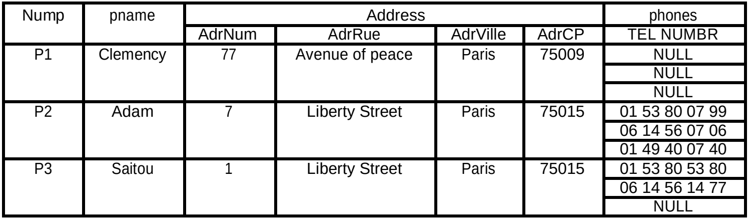
Pre-sized size arrays (VARRAY)

- Insert: INSERT with VALUES

  Storage Professor 3 type with no object, respectively, three and two telephone numbers (VARRAY of records)

```
insert      into    professors          gains     ( 'P1',       'Mercy'
                AA stands _ kind (7,        'Avenue          the   Peace ' ,  'P aris'        '75009'),
                such kind _ ());

insert      into    professors          gains     ( 'P2',       Adam '
                                               'Stre          th
                AA stands _ kind (7 7              et          of  e     freedom ' ,      'P aris'          '75015'),
                such _ type (such kind _ ( '01 53          80 07          99 '),
                                    such kind _ ( '06 14  56 07          06 '),
                                    such kind _ ( '01 49  40 07     40 ')));

inser t nto   professeursvalues AA stands _    (P3 ',       S aitou '
              Type (1, '_ Rue such type (such _ del al ibert é ',' P aris', '75015'),
              template (' _ 01 such deviation (
              '06 14 56 14
                                                  53   80 53 80 '),
                                                77 '), NULL));
```

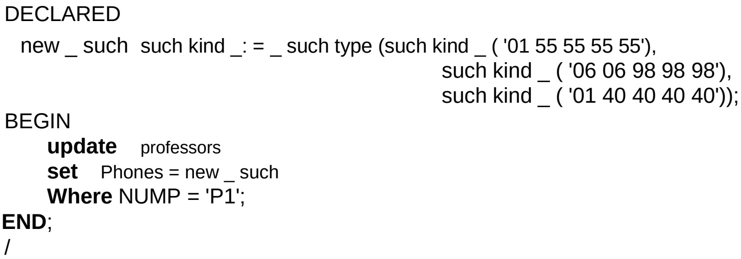
Pre-sized size arrays (VARRAY)



Nump	pname	Address				phones
		AdrNum	AdrRue	AdrVille	AdrCP	TEL NUMBR
P1	Clemency	77	Avenue of peace	Paris	75009	NULL
						NULL
						NULL
P2	Adam	7	Liberty Street	Paris	75015	01 53 80 07 99
						06 14 56 07 06
						01 49 40 07 40
P3	Saitou	1	Liberty Street	Paris	75015	01 53 80 53 80
						06 14 56 14 77
						NULL

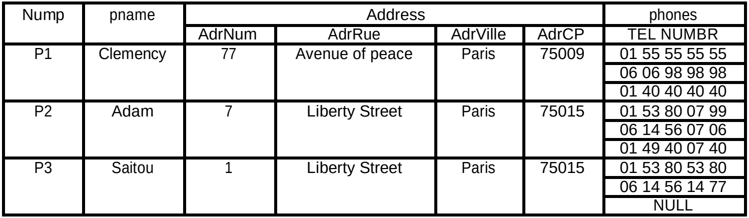
Pre-sized size arrays (VARRAY)

- Insert: INSERT into a VARRAY with PL / SQL
  - With VARRAY tables, the operator is not operational TABLE (Version 8 Oracle - to check on the V9 and v10g)
  - To manipulate the tables, it is necessary to use a program PL / SQL



```
DECLARED
  new _ such  such kind _: = _ such type (such kind _ ( '01 55 55 55 55'),
                                          such kind _ ( '06 06 98 98 98'),
                                          such kind _ ( '01 40 40 40 40'));
BEGIN
    update   professors
    set   Phones = new _ such
    Where NUMP = 'P1';
END;
/
```

Pre-sized size arrays (VARRAY)



Nump	pname	Address				phones
		AdrNum	AdrRue	AdrVille	AdrCP	TEL NUMBR
P1	Clemency	77	Avenue of peace	Paris	75009	01 55 55 55 55 06 06 98 98 98 01 40 40 40 40
P2	Adam	7	Liberty Street	Paris	75015	01 53 80 07 99 06 14 56 07 06 01 49 40 07 40
P3	Saitou	1	Liberty Street	Paris	75015	01 53 80 53 80 06 14 56 14 77 NULL

Note :

- Insert a single telephone number for Professor P1 and place it in 2nd place in the table Phones
  Writing following the trial of a ff assignment:

```
new _ such  _ such kind: = _ such kind (NULL,
                          such kind _ ( '06 06 98 98 98'), NULL);
```

Conclusion

Comparing NESTED TABLE and VARRAY

- A check by Oracle versions
- Ability to define an index in a NESTED TABLE

  The number of elements is not limited in a nested table
- No ability to set index in a VARRAY

  The number of elements is limited in a pre-sized table
- Ability to directly access records stored in both data structures functions: EXISTS, FIRST, LAST, etc.
- Performance? : NestedTable> Varray