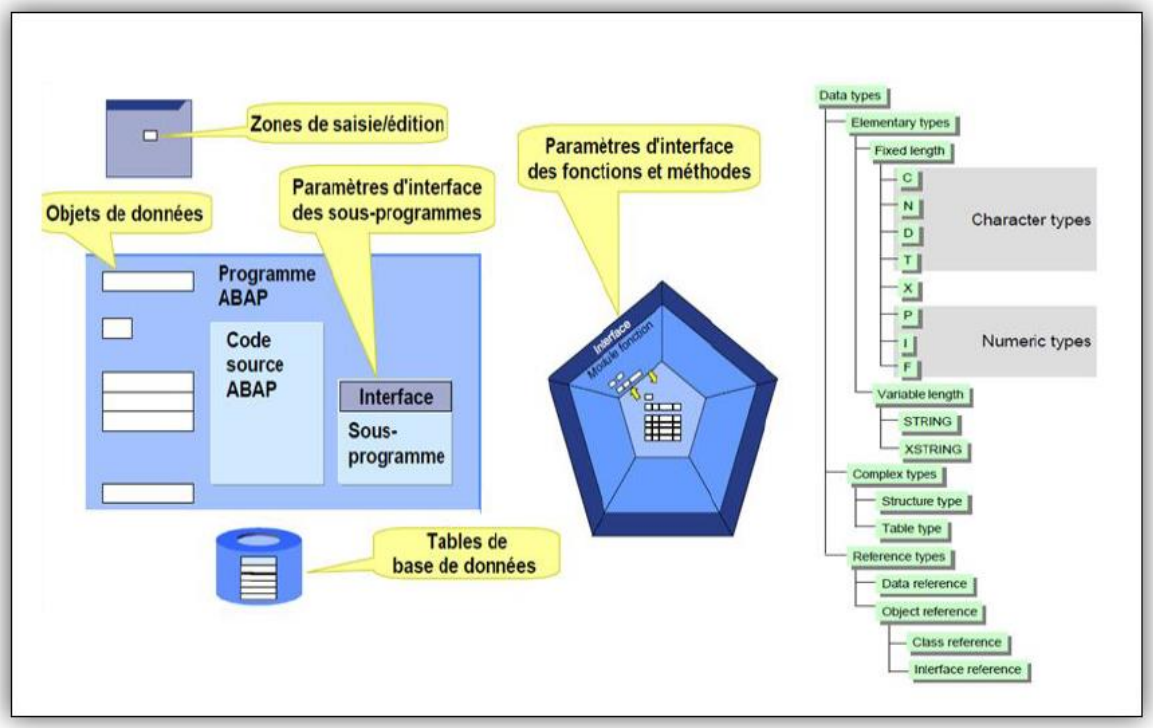
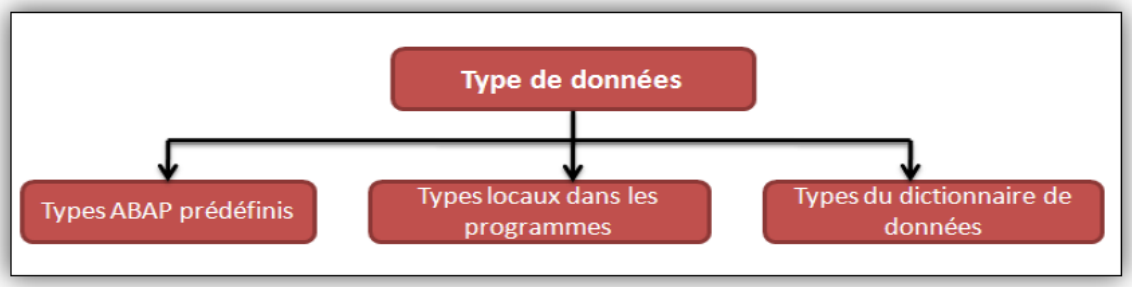
LES TYPES DE DONNEES

- Les types décrivent les attributs des:
- Zones de saisie et de sortie sur les écrans.
- Objets de données.
- Paramètres d'interfaces : les contrôles de types sont exécutés chaque fois qu'une fonction ou qu'un sous-programme est appelé.
- Les tables de base de données.



Les déclarations de données dans un programme peuvent se baser sur un type ABAP prédéfini, sur un type créé dans le dictionnaire de données ou sur un type créé localement dans le programme :



LES TYPES PREDEFINIS D'ABAP

Ce sont des types prédéfinis dans le noyau du système SAP R/3 et visibles pour tous les programmes ABAP. Ils sont utilisés pour définir des types de données et des objets locaux et aussi pour spécifier les types des fields-symboles et les types des paramètres utilisés dans des écrans ABAP.

	Type de données	Signification	Valeur initiale	Valeurs possibles	
Numérique	I	Nombre entier	0	[-2147483648 ; 2147483647]	
	F	Nombre en virgule flottante	0.0...E+000	[2.2...E-308 ; 1.7...E+308]	
				Longueur par défaut (carac.)	Longueur maximale (carac.)
	P	Nombre condensé	0	15	31
Alphanumérique	N	Chaîne numérique	00 ... 0	1	65535
	C	Chaîne de caractères	_ _ ... _	1	65535
	STRING	Chaîne de caractères	vide	0	indifférent
	D	Date JJMMAAAA	00000000	8	8
	T	Temps HHMMSS	000000	6	6
	X	Code hexadécimal	X'00'	1	65535
	XSTRING	Code hexadécimal	vide	0	indifférent

Remarque : **STRING** et **XSTRING** se caractérisent par une longueur variable.

LES TYPES LOCAUX

Un nouveau type de données local peut être défini en fonction d'un *type existant*, en utilisant le mot-clé **TYPES**.

```
TYPES <type> [TYPE <type>|LIKE <objetdonnées>].
```

- Le nom du type peut contenir jusqu'à 30 caractères.
- Le nom du type peut contenir des lettres, chiffres et le caractère _.
- Le nom de type ne doit pas être le même qu'un type prédéfini ou un mot clé ABAP.
- Le nom du type doit être le plus significatif possible.
- Utiliser le caractère _ pour séparer les mots composés.
- Le premier caractère doit être une lettre.

Un type existant peut être :

- Un type prédéfini (*TYPE*).
- Un type local précédemment défini dans le programme (*TYPE*).
- Un type d'un objet local défini (avec *DATA*) dans le programme (*LIKE*).
- Un type créé dans le dictionnaire de données (*TYPE*).
  Remarque : Dans des anciennes versions, *LIKE* était utilisé pour faire référence aux tables transparentes définies dans le dictionnaire de données.

Si la déclaration va se baser sur un type générique, des options sont ajoutées afin de définir les attributs du type <t> qui ne sont pas encore définis.

LES TYPES DE DONNEES ELEMENTAIRES

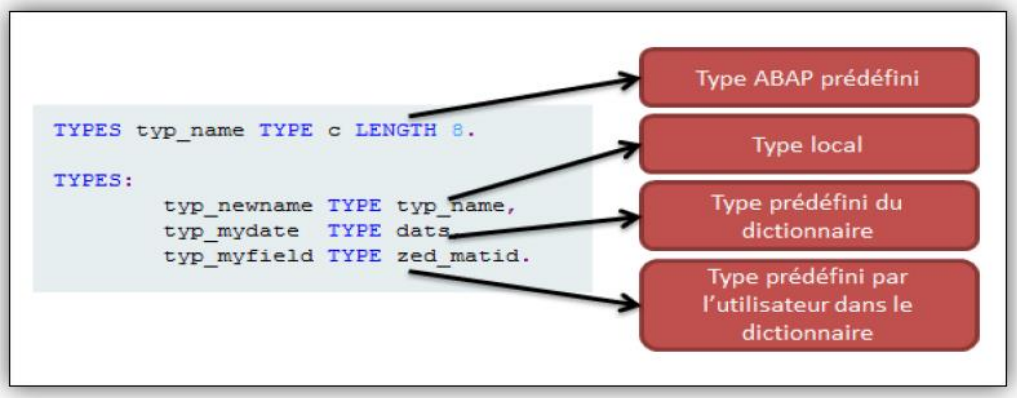
Les types de données élémentaires dans un programme sont définis en utilisant les types élémentaires prédéfinis.

La syntaxe suivante permet de définir un nouveau type de données élémentaire:

```
TYPES <t>[(<length>)] [TYPE <type>|LIKE <obj>] [DECIMALS <dec>].
```

- <*type*> peut-être un type ABAP prédéfini, un type élémentaire local ou un élément de données défini dans le dictionnaire ABAP.
- Si <*type*> est un des types prédéfinis dont la longueur est fixe (C, N, P et X), il faut définir la longueur en utilisant l'option <*length*>.
- Lorsque l'option *LIKE* est utilisée, <*obj*> peut être un objet de données existant avec un type de données élémentaire.

Remarque : Si aucune option n'est utilisée le système prend par défaut le type **C** de longueur 1.



LES TYPES STRUCTURES

Pour créer une nouvelle structure dans un programme, il faut utiliser la syntaxe suivante :

```
TYPES:
        BEGIN OF <typestruct>,
        * ...composantes...
        END OF <typestruct>.
```

Cette déclaration crée une structure contenant toutes les variables entre :
TYPES BEGIN OF<typestruct> et **TYPES END OF <typestruct>.**

```
TYPES typ_name TYPE c LENGTH 8.

TYPES:
      BEGIN OF typ_linetype,
         name    TYPE typ_name,
         mydate  TYPE dats,
         myfield TYPE zed_matid,
      END OF typ_linetype.

TYPES:
      typ_linenew TYPE typ_linetype,
      typ_mystruc TYPE zst_structure. "zst_structure:structure défini dans le dictionnaire ABAP4
```

Un type structuré peut aussi être défini par rapport à un type de ligne d'une table interne par l'instruction **LIKE LINE OF <tabi>** et par rapport à un type de table par l'instruction **TYPE LINE OF <typetabi>**

LES TYPES DE TABLE

LES TYPES DE TABLE

Les tables locales dans un programme sont appelées des **tables internes**. Pour définir un nouveau type de table interne, la syntaxe suivante est utilisée :

```
TYPES <t>
        TYPE|LIKE <tabkind> OF <linetype>
        [WITH <key>]
        [INITIAL SIZE <n>].
```

Après TYPE ou LIKE, il n'y a aucune référence à un type de données existant mais il y a le constructeur de type:

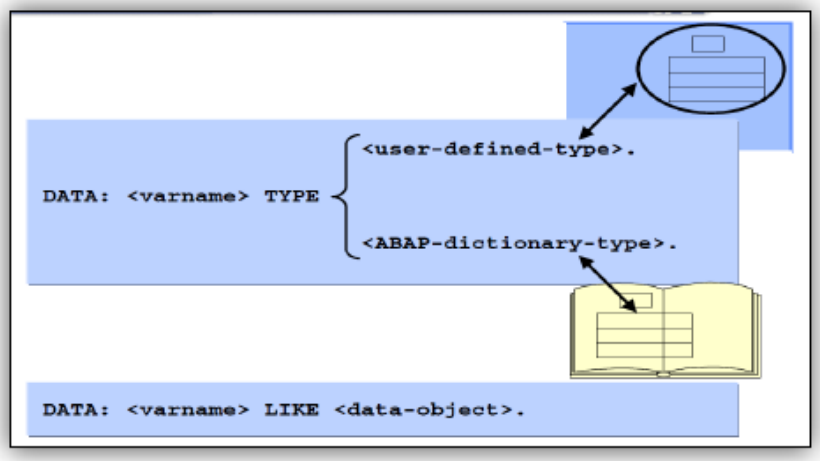
```
TYPES typ_name TYPE c LENGTH 8.

TYPES:
        typ_newname TYPE typ_name,
        typ_mydate  TYPE dats,
        typ_myfield TYPE zed_matid.

TYPES:
      BEGIN OF typ_linetype,
         name    TYPE typ_name,
         mydate  TYPE dats,
         myfield TYPE zed_matid,
      END OF typ_linetype.

TYPES:
      typ_linenew TYPE typ_linetype,
      typ_mystruc TYPE zst_structure. "zst_structure:structure défini dans le dictionnaire ABAP4

TYPES  typ_sophtab
       TYPE SORTED TABLE OF typ_linenew
       WITH UNIQUE KEY myfield name.

TYPES:
      typ_mytabnew TYPE typ_sophtab,
      typ_othertab TYPE ztt_mytabletype. "ztt_muytabletype: type de table créé dans le dictionnaire ABAP
```

Objets de données :

Un **objet de données** est une zone de mémoire nommée, structurée en fonction d'un type de données spécifique. Ces objets contiennent des données nécessaires lors de l'exécution du programme mais qui ne sont pas persistantes (n'existent que pour la durée de l'exécution du programme).

Remarque : Avant de traiter les données persistantes telles que les données d'une table transparente ou à partir d'un fichier séquentiel, il faut les lire dans des objets de données. Par contre, si l'utilisateur veut garder le contenu d'un objet de données après la fin du programme, il faut l'enregistrer sous une forme persistante.

Pour définir un objet de données dans un programme, il faut utiliser le mot clé **DATA** :



LES VARIABLES:

Les variables sont définis selon le format suivant :

```
xv_.....
```

- **x** prend la valeur :
  - **l** dans le cas d'une variable locale.
  - **g** lorsqu'il s'agit d'une variable globale.

```
TYPES typ_name TYPE c LENGTH 8.


DATA:
      lv_value      TYPE p DECIMALS 2,
      lv_newname    TYPE typ_name,
      lv_mydate     TYPE dats,
      lv_myfield    TYPE zed_matid,
      lv_mynewfield LIKE lv_myfield.
```

LES OBJETS DE DONNEES PREDEFINIS

Au moment de l'exécution, les objets de données suivants sont toujours présents et il n'est pas nécessaire de les déclarer.

- **SPACE** : une constante de type **C** et de longueur 1. Il contient un seul espace.
- **Les zones systèmes de la structure SY :** en fonction de la structure **SYST** du dictionnaire ABAP, le système crée automatiquement une structure appelée **SY** pour chaque programme. Les composantes individuelles de la structure sont appelées **zones système**. Elles contiennent des valeurs de l'état courant du système. Remarque : Ces valeurs sont mises à jour automatiquement par l'environnement d'exécution ABAP.

La notation **sy-<zone_système>** permet d'accéder aux zones systèmes.

Parmi les zones systèmes, il y a :

- **sy-tcode** : représente code de transaction courant.
- **sy-mandt :** représente le mandant courant.
- **sy-uname :** représente l'utilisateur courant.
- **sy-datum :** représente la date courante.
- **sy-langu** : représente la langue courante.
- **sy-subrc** : représente le code retour des instructions ABAP.

LES STRUCTURES

Les structures sont définies dans un programme ABAP de la même façon que les objets de données élémentaires, en utilisant le mot clé **DATA**.

Le nom de la structure doit commencer par **LS** pour une structure locale et **GS** pour une structure globale.

```
TYPES typ_name TYPE c LENGTH 8.

TYPES:
       typ_newname TYPE typ_name,
       typ_mydate  TYPE dats,
       typ_myfield TYPE zed_matid.

TYPES:
      BEGIN OF typ_linetype,
        name    TYPE typ_name,
        mydate  TYPE dats,
        myfield TYPE zed_matid,
      END OF typ_linetype.
TYPES:
      typ_linenew TYPE typ_linetype,
      typ_mystruc TYPE zst_structure. "zst_structure:structure défini dans le dictionnaire ABAP

TYPES  typ_sophtab
       TYPE SORTED TABLE OF typ_linenew
       WITH UNIQUE KEY myfield name.

DATA:
      ls_linetype   TYPE         typ_linetype,
      ls_mystruc    TYPE         zst_structure,"zst_structure: structure défini dans le dictionnaire ABAP
      ls_sflight    TYPE         sflight_t,    "sflight_t : une structure standard défini dans le dictionnaire ABAP
      ls_mynewstruc LIKE         ls_sflight,
      ls_mysophtab  TYPE LINE OF typ_sophtab.
```

Les options **TYPE LINE OF <typetabi>** et **LIKE LINE OF <tabi>** sont utilisés respectivement pour consulter le **type de ligne** d'un type de table et d'une table interne.

Une structure peut être créée directement dans une instruction *DATA* sans avoir à définir au préalable un propre type de données.

```
DATA: BEGIN OF st_structurename,

        …..,

        END OF st_structurename.
```

LES TABLES INTERNES

Les tables internes sont définies suivant la syntaxe suivante :

```
DATA <objetdonnéestabi>
{TYPE|LIKE} {[STANDARD]|SORTED|HASHED|INDEX|ANY}
TABLE OF {<typestruc>|<structureobjetdonnées>}
[WITH {[NON-UNIQUE]|UNIQUE}
{KEY {<f1> ... <fn>|TABLE LINE} |DEFAULT KEY}]
[INITIAL SIZE <n>]
[WITH HEADER LINE].
```

À l'exception de l'option **WITH HEADER LINE**, la syntaxe permettant de déclarer les objets de la table interne, est identique à celle utilisée pour définir les types de table.

Remarque : L'option **WITH HEADER LINE** permet de créer une **table interne avec un en-tête**. Cependant, cette technique de programmation étant obsolète, il est recommandé de ne plus l'utiliser.

```
TYPES typ_name TYPE c LENGTH 8.

TYPES:
       typ_newname TYPE typ_name,
       typ_mydate  TYPE dats,
       typ_myfield TYPE zed_matid.

TYPES:
      BEGIN OF typ_linetype,
        name    TYPE typ_name,
        mydate  TYPE dats,
        myfield TYPE zed_matid,
      END OF typ_linetype.
TYPES:
      typ_linenew TYPE typ_linetype,
      typ_mystruc TYPE zst_structure. "zst_structure:structure défini dans le dictionnaire ABAP

DATA lt_simptab TYPE STANDARD TABLE OF typ_linetype.
DATA :
      lt_sophtab
      TYPE SORTED TABLE OF t_linenew
      WITH UNIQUE KEY myfield name.

DATA :
      lt_mytabnew LIKE lt_scphtab,
      lt_othertab TYPE ztt_mytabletype."ztt_mytabletype: type de table défini dans le dictionnaire ABAP
```

TP

1) Créer un report sans TOP Include, nommé **ZTR_RPT_<quadrigramme>_TYPES_MANAGE.**

   Remarque : Le programme doit respecter la structure d'un programme ABAP.

2) Déclarer un type de données :

Nom	Type
TYP_NEWP	p de longueur 10 et 2 décimales après la virgule
TYP_NAME	Chaine de caractères de longueur 18
TYP_ADRESSE	Chaine de caractères de longueur variable
TYP_PARTNER	BU_PARTNER
TYP_STREET	ZED_STREET_##

3) Déclarer un type de structure :

   - TYP_CLIENT avec les champs suivants :

Champ	Type
ID	TYP_PARTNER
NOM	Chaine de caractères de longueur 18
PRENOM	BU_NAMEP_F
STREET	ZED_STREET_##

   - TYP_SNEWCLIENT de type TYP_ CLIENT.
   - TYP_SFICHE_CLIENT en utilisant le type de structures ZST_FICHE_CLIENT_##.

4) Déclarer un type de table :

Nom	Type
TYP_TNEWCLIENT	Le type de ligne est TYP_CLIENT
TYP_TFICHE_CLIENT	Une table standard avec la structure ZST_FICHE_CLIENT_## comme type de ligne et identifiant_client comme clé non unique.
TYP_TNEWFICHE_CLIENT	TYP_TFICHE_CLIENT
TYP_TFCLIENT	ZTT_FICHE_CLIENT_##