Bundesamt für Landestopographie
Office fédéral de topographie
Ufficio federale di topografia
Federal office of topography

Eidgenössische Vermessungsdirektion
Direction fédérale des mensurations cadastrales
Direzione federale delle misurazioni catastali
Directorate of cadastral surveying

INTERLIS

Mécanisme d'Echange de Données Pour Systèmes d'Information du territoire

Version 1, Révision 2, mars 1998
(édition français du 1999-12-08)

Direction fédérale des mensurations cadastrales
Office fédéral de topographie
Seftigenstrasse 264, CH-3084 Wabern
Fax +41 31 963 22 97, E-Mail interlis@swisstopo.ch
Web www.swisstopo.ch ou www.gis.ethz.ch

Copyright © 1999 Direction fédérale des mensurations cadastrales, CH-3084 Wabern

All names which with an accompanying © sign are copyrighted with the respective author or vendor. Duplications and copies are permitted explicitely as long as this content remains unchanged and there is a reference to this document indicated.

Contenu

1. Vue d'ensemble

L'idée de base d'INTERLIS consiste à affirmer qu'un échange des informations d'un système d'information du territoire n'est possible que si les partenaires intéressés à l'échange se font une idée précise et unifiée sur le genre des données à échanger. C'est pourquoi INTERLIS se consacre tout d'abord à la description précise du modèle des données, puis, dans un deuxième temps seulement, à la définition du format d'échange.

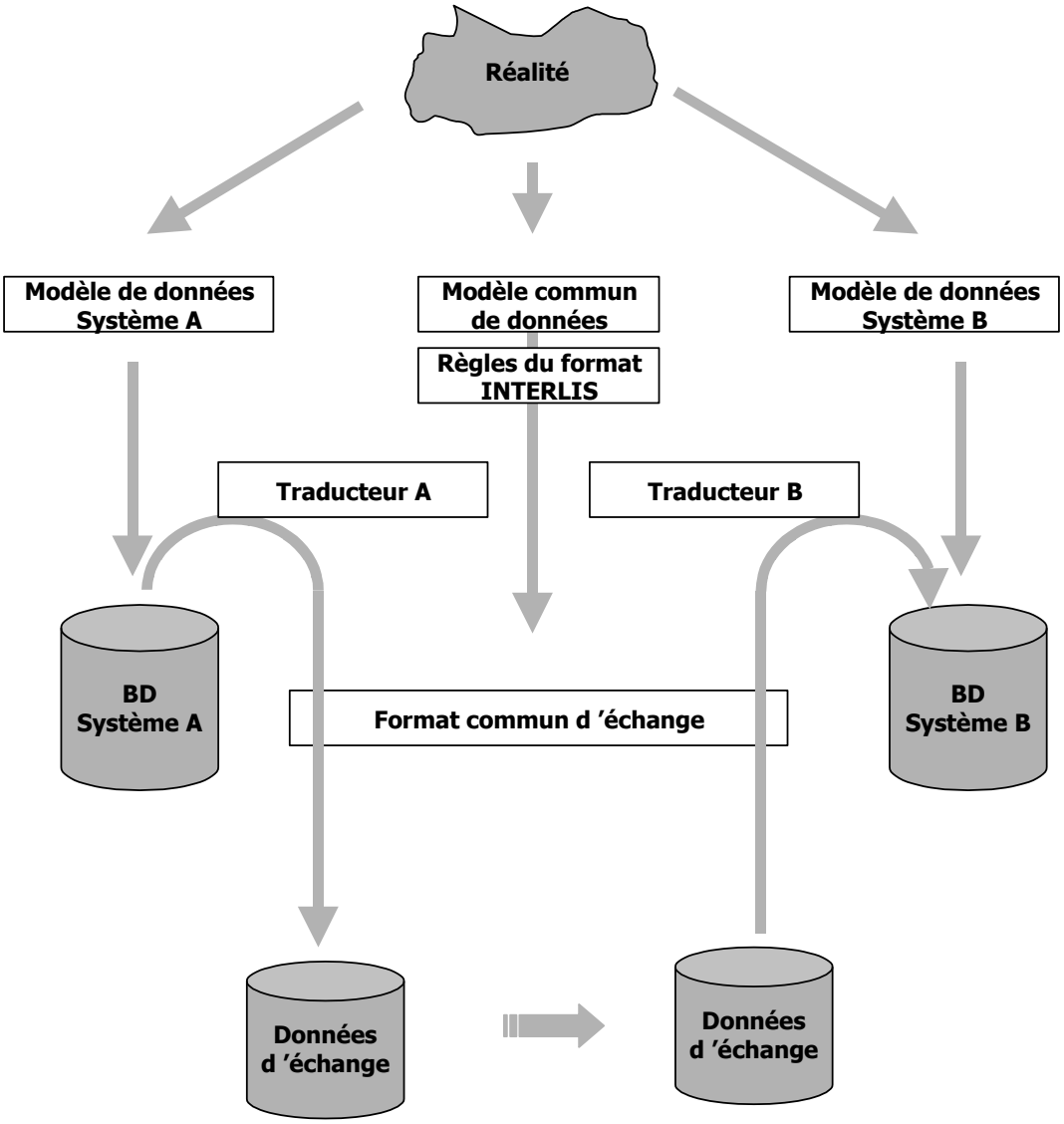


Figure 1: Transfert des données avec un modèle et un langage commun entre deux systèmes différents.

La description du modèle des données ne part pas d'une application déterminée. Bien plus, le modèle d'une application concrète peut être défini au moyen du langage de description (cf. chap. 2). Ainsi INTERLIS est-il l'outil disponible pour toutes les applications qui peuvent être décrites avec les éléments de base prévus.

Vu sous l'angle du principe de description, INTERLIS se rapproche du modèle relationnel avec extension à des éléments typiques des systèmes d'information du territoire. La syntaxe du langage suit les idées de langages modernes de programmation tels PASCAL ou MODULA2.

Pour rendre possible un échange concret de données, il faut, en sus du modèle, définir un protocole de transfert (format d'échange). Pour ne pas perdre la flexibilité de la description du modèle, le protocole de transfert est formulé de telle sorte qu'il s'adapte à la définition concrète des données. Actuellement, l'échange des données est défini au niveau fichiers de texte (cf. chap. 3).

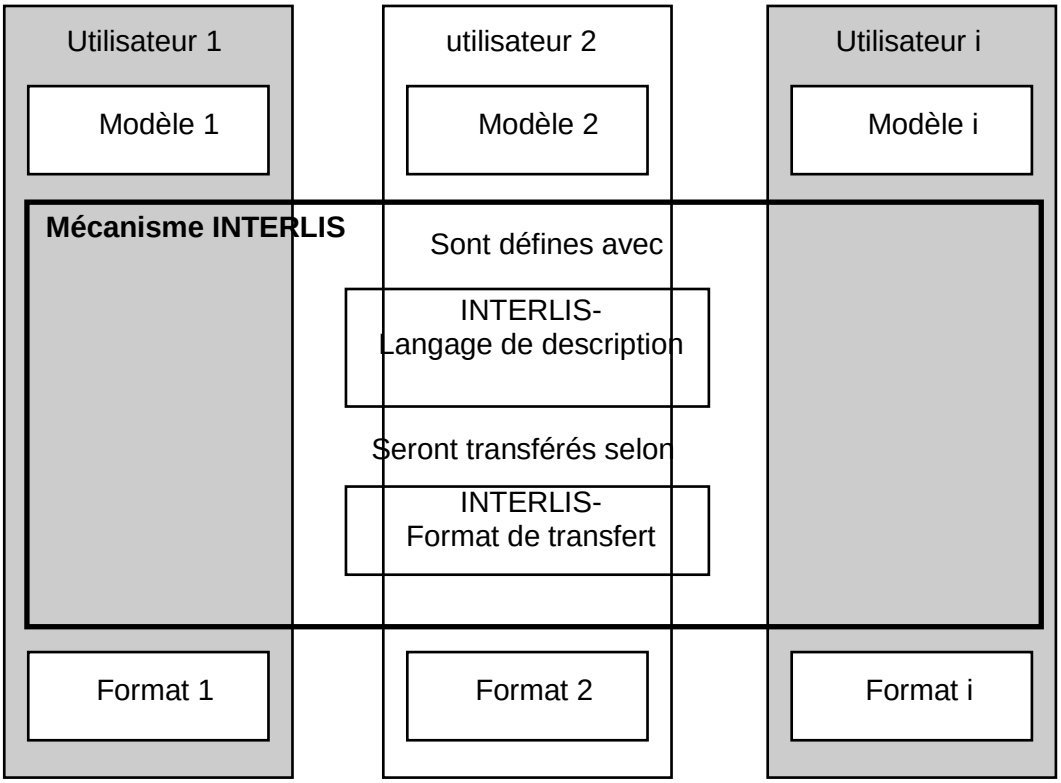


Figure 2: INTERLIS langage de description pour des données et règles de format INTERLIS.

Pour pouvoir contrôler si un modèle décrit par INTERLIS est formellement correct, il existe un compilateur qui établit une description du format de transfert correspondant au modèle (cf. chap. 4).

Le chap. 5 contient un exemple simple, tandis que le chap. 6 regroupe des indications relatives à des questions particulières.

2. Langage de description

2.1. Syntaxe utilisée

Pour déterminer le modèle des données et les paramètres de transfert d'un échange de données, les chapitres suivants définissent un langage formel. Ce langage est défini lui même d'une manière formelle. Les règles de syntaxe décrivent la suite admissible des symboles.

La description respecte ainsi les règles usuelles en matière de langages modernes de programmation. C'est pourquoi il ne sera donné ci-dessous qu'une représentation succincte, nécessaire à la compréhension. Pour des cas individuels, la littérature peut être consultée. Par exemple, une courte introduction se trouve dans "Programmer en Modula2" de Niklaus Wirth.

Au sens de la "Notation étendue Backus-Naur" (EBNF), une formule s'exprime comme suit:

```
Nom de la formule = Expression de la formule.
```

L'expression de la formule est une combinaison de:

- mots fixes (y compris signes spéciaux) du langage, enfermés entre des apostrophes, p.ex. 'BEGIN'.
- références d'autres formules par leur nom de formule.

On peut avoir les combinaisons suivantes:

Suite

```
a b c            d'abord a, puis b, puis c
```

Sélection obligatoire

```
(a | b | c)      a ou b ou c
```

Sélection facultative

```
[ a | b | c]     a, b ou c ou rien
```

Répétition facultative

```
{ a | b | c }    suite quelconque de a, b ou c.
```

Exemples: aaaaaa, abbc, accccab. Il est aussi admissible, si une suite de a, b ou c n'est pas indiquée.

Répétition obligatoire (en complément à EBNF)

```
(* a | b | c *)
```

suite quelconque de a, b ou c. Au moins une mention est exigée. "Rien" n'est pas possible. (* a *) a la même signification que a {a}, mais s'écrit plus simplement.

On aimerait souvent utiliser une même formule syntaxique dans des circonstances différentes, dans des buts différents. Pour atteindre cet objectif, il faudrait écrire une formule complémentaire:

```
Exemple = 'TABLE' Nom_de_la_table '=' Definition_de_la_table.
Nom_de_la_table = Nom.
```

Pour éviter cette relation indirecte, on utilise la notation abrégée suivante:

```
Exemple = 'TABLE' Nom-de_la_table '= ' Definition_de_la_table.
```

L'expression Nom-de_la_table n'est pas définie. Sur le plan syntaxique, la règle Nom est directement utilisée. Quant au fond, Nom est le nom d'une table. Le complément "de_la_table" devient pratiquement un commentaire.

2.2. Définition du langage de description

2.2.1. Symboles de base du langage

Le langage de description distingue les classes de symboles suivantes: noms, nombres, explications, signes particuliers et mots réservés, commentaires.

2.2.1.1. Noms

```
Nom = Lettre { Lettre | Chiffre | '_'}.
Lettre = ( 'A' | .. | 'Z' | 'a' | .. | 'z').
Chiffre = ( '0' | '1' | .. | '9' ).
```

Un nom est défini comme une suite de lettres, chiffres et traits de soulignement, où le premier caractère doit être une lettre. Les caractères accentués ne sont pas autorisés!

2.2.1.2. Nombres

Les nombres sont utilisés pour la définition de domaines et pour la codification d'un caractère donné. Dans ce deuxième cas, il est très utile de disposer également de la notation hexadécimale.

```
NoPos = (* Chiffre *) .
Nombre = [ '+' | '-' ] NoPos .
Dec = Nombre [ '.' NoPos ] [ Facteur_de_multiplication ] .
Facteur_de_multiplication = 'S' Nombre .
Code = ( NoPos | NoHex ) .
ChiffreHex = ( Chiffre |'A'|'B'|'C'|'D'|'E'|'F'|'a'|'b'|'c'|'d'|'e'|'f').
NoHex = '0' ( 'x' | 'X' ) (* ChiffreHex *).
```

Exemples :

```
NoPos:    5134523        1            23
Nombre:       123     -435         +5769
Dec:          123.456  123.456S4     123.456S-2
Code:        1234     0XA2
```

Avec le facteur de multiplication contenu dans "Dec", on obtient que l'indication de valeur reste de grandeur acceptable, sans zéros superflus. Par ex., 123.456S4 signifie 123.456*10**4, donc 1234560.

2.2.1.3. Explications

Une explication est exigée là où un fait doit être décrit avec plus de précision. Du point de vue du mécanisme standard, cette explication n'est pas interprétée et est traitée comme un commentaire. Mais il est sans autre possible de formaliser des explications plus détaillées pour permettre un traitement automatique supplémentaire. Une explication est insérée entre deux doubles traits obliques et ne doit par conséquent pas contenir de tels doubles traits.

```
Explication = '//' car_quelconques _sauf _//'//'.
```

2.2.1.4. Signes particuliers et mots réservés

Les signes particuliers et les mots réservés sont toujours écrits entre apostrophes selon les règles syntaxiques du langage, par ex. ' ; ' ou 'MODEL'. Les mots réservés s'écrivent en majuscules. En n'utilisant que des majuscules pour les noms, on s'évite ainsi facilement tout risque de conflit.

Les mots suivants sont réservés :

ANY	ARCS	AREA	BASE	BLANK
CODE	CONTINUE	CONTOUR	COORD2	COORD3
DATE	DEFAULT	DEGREES	DERIVATIVES	DIM1
DIM2	DOMAIN	END	FIX	FONT
FORMAT	FREE	GRADS	HALIGNMENT	I16
I32	IDENT	LINEATTR	LINESIZE	MODEL
NO	OPTIONAL	OVERLAPS	PERIPHERY	POLYLINE
RADIANS	STRAIGHTS	SURFACE	TABLE	TEXT
TID	TIDSIZE	TOPIC	TRANSFER	UNDEFINED
VALIGNMENT	VERTEX	VERTEXINFO	VIEW	WITH
WITHOUT				

Table 1: Mots réservés.

2.2.1.5. Commentaires

!! jusqu'à la fin de la ligne

Deux points d'exclamation consécutifs ouvrent un commentaire. Celui-ci se termine à la fin de la ligne.

2.2.1.6. Séparation des symboles

Habituellement, on sépare des symboles consécutifs par un espace (un ou plusieurs caractères blancs, des tabulateurs ou la fin de ligne). Ceci n'est toutefois indispensable que dans le cas où l'absence d'espace conduirait à interpréter deux symboles consécutifs comme un seul autre symbole.

2.2.2. Exemple introductif

Utilisons comme exemple une description simple de la surface du sol (la réforme de la mensuration officielle suisse REMO utilise pour ceci l'expression "couverture du sol"). Au chap. 5, le même exemple sera repris plus en détail, avec la définition du format d'échange et l'établissement d'un fichier concret des données.

```
TRANSFER Exemple;

MODEL Exemple

  DOMAIN
    CoordN = COORD2 480000.00   60000.00
                    850000.00  320000.00;

  TOPIC Couverture_du_sol =

    TABLE Surfaces_sol =
      Genre : (batiment, revetement_dur, verte, eau,
               boisee, sans_vegetation);
      Forme : AREA WITH (STRAIGHTS, ARCS) VERTEX CoordN
              WITHOUT OVERLAPS > 0.10;
    NO IDENT
      !! recherche via geometrie ou batiment
    END Surfaces_sol;

    TABLE Batiments =
      NoAss : TEXT*6;
```

```
      Surface : -> Surfaces_sol // Genre = batiment //;
    IDENT
      NoAss;   !! supposition que NoAss est univoque
      Surface; !! une seule surface est subordonnee a ce batiment
    END Batiments;

  END Couverture_du_sol.
END Exemple.

FORMAT FREE;

CODE
  BLANK = DEFAULT, UNDEFINED = DEFAULT, CONTINUE = DEFAULT;
  TID = ANY;
END.
```

2.2.3. Structure principale du langage

2.2.3.1. Eléments principaux

```
TransferDef =   'TRANSFER'  Nom-du_transfert ';'
                [DefDomaine_global]
                Modele_des_donnees [Exploitations]
                {Vue} Format Codification.
```

Une définition du transfert se compose de la définition des domaines valables pour toute la définition (facultatif), de la description du modèle des données et de la fixation des paramètres de transfert (format, codification). Le modèle des données permet la description précise du genre et de la structure des données à échanger. Les paramètres du transfert sont des indications complémentaires utiles au transfert.

2.2.3.2. La définition du domaine

```
DefDomaine = 'DOMAIN'
  (* Nom-du_domaine '=' Attribut_type ';' *).
```

La définition d'un domaine sert à fixer un certain type de donnée avec toutes ses caractéristiques. Cette définition peut être utilisée plus tard pour la définition des caractéristiques des attributs dans les circonstances suivantes :

- dans l'entier de la définition de transfert, lorsque la définition des domaines intervient dans le cadre d'une définition de transfert. Le nom reste réservé pour toute la définition de transfert.
- à l'intérieur du modèle ou d'une exploitation si la définition du domaine intervient dans le cadre de la définition du modèle. Le nom reste réservé dans ce cadre.
- à l'intérieur d'un thème, si la définition du domaine intervient dans le cadre de la définition du thème. Le nom reste réservé jusqu'à la fin du thème.

2.2.3.3. Le modèle des données

```
Modele_des_donnees = 'MODEL' Nom-du_modele
                       [DefDomaine-modele]
                       (* Theme *)
                     'END' Nom-du_modele '.'.
```

Le modèle des données est décrit par

- un nom de modèle
- les domaines de validité pour tout le modèle
- les définitions des différents thèmes.

2.2.3.4. Le thème

```
Theme =  'TOPIC' Nom-du_theme  '='
           (* Table | DefDomaine-local *)
         'END' Nom-du_theme '.'.
```

Un thème est tout d'abord un ensemble de tables. Du point de vue du transfert des données, les thèmes sont totalement indépendants les uns des autres. Un système qui reçoit des données doit être en mesure de recevoir ces données même si tous les thèmes ne sont pas livrés. Le transfert d'un objet d'un thème donné ne doit en aucune façon être lié à l'existence d'objets d'autres thèmes.

Cette séparation absolue des données évite, lors du transfert d'un objet, l'effet de cascade entraînant le transfert de toute une suite d'autres objets. Le transfert de bases de données partielles est ainsi possible. De même, aucune condition n'est fixée a priori sur les mesures à prendre par le système récepteur pour constater les éventuelles inconsistances entre les niveaux de données et pour les éliminer.

2.2.3.5. La table

```
Table = ['OPTIONAL' ] 'TABLE' Nom-de_la_table '='
                        Attributs Identificateurs
                      'END' Nom-de_la_table ';'.
Identificateurs = ( 'NO' 'IDENT' | 'IDENT' ( * Identdef *) ).
Identdef =  Nom-attribut { ',' Nom-attribut } ';'.
```

Une table est un ensemble d'objets ayant les mêmes caractéristiques. Celles-ci sont décrites au moyen d'attributs et d'identificateurs. Les attributs décrivent la structure exacte des données des objets. Avec chaque définition d'une identification ont définit un attribut ou une combinaison d'attributs qui identifie un objet de manière univoque.

Exemple :

```
TABLE Object =
  a: TEXT*8;
  b: TEXT*8;
  c: TEXT*8;
IDENT
  a;
  b, c;
END Object;
```

La première ligne signifie que l'attribut a identifie l'objet, la deuxième que la combinaison de b et c est identifiante (donc que b ou c seuls ne sont pas univoques).

Avec 'OPTIONAL' on montre que la table ne doit pas obligatoirement être présente.

2.2.3.6. L'attribut

```
Attributs = (* Nom-attribut ':'
```

```
                ['OPTIONAL'] (Attribut_local | Attribut_relationnel )
                [Condition_de_coherence] ';' *).
    Condition_de_coherence = explication.
```

Chaque attribut est décrit par son nom, sa structure et au besoin par les indications relatives à des conditions de cohérence spécifiques. En sus, la mention 'OPTIONAL' indique qu'il n'est pas indispensable de définir l'attribut. La définition de l'attribut peut se terminer facultativement par des conditions de cohérence, celles-ci n'étant pas formalisées mais données comme explication. Du point de vue de la structure, les attributs peuvent être classés en deux groupes:

Attributs locaux

Un attribut local n'a de sens qu'à l'intérieur de la table. Pour la description des caractéristiques d'un attribut, différents attributs-types sont à disposition. Ils peuvent grosso modo être classés en types de base et en divers types géométriques. Les spécifications exactes sont données dans les chapitres suivants.

```
Attribut_local = Domaine.
Domaine = (Nom-du_domaine | Attribut_type);
Attribut_type = (Type_de_base | Ligne | SurfLot).
```

Attributs relationnels

Les relations entre objets sont assurées par l'intermédiaire de ces attributs. Il n'est pas nécessaire de se préoccuper de la façon dont la relation s'établit. Il suffit d'indiquer vers quelle table et vers quel objet se fait la relation. Pour assurer l'indépendance des niveaux de données, on ne peut utiliser les attributs relationnels qu'à l'intérieur d'un même thème.

```
Attribut_relationnel = '-> '  Nom-de_la_table.
```

2.2.4. Types de base

```
Type de base = ( Coord2
               | Coord3
               | Longueur
               | Superficie
               | Angle
               | Domaine_numerique
               | Texte
               | Date
               | Denombrement
               | Alignement_horizontal
               | Alignement_vertical).
```

Les coordonnées, les longueurs et les superficies se réfèrent à la mesure linéaire. Pour des raisons de clarté des données transférées, il faut fixer pour ces types une même unité de base (en règle générale le mètre) pour toutes les indications les concernant.

L'indication de minimum et maximum pour les différents types définit non seulement les limites mais aussi le nombre de positions. Les décimales et le facteur de multiplication doivent concorder entre minimum et maximum. Si on veut par exemple définir un type longueur avec une limite supérieure d'un kilomètre et la représenter en millimètre, on écrit :

```
DIM1 0.000 1000.000
```

2.2.4.1. Coordonnées

```
Coord2 = 'COORD2' Dec-ymin Dec-xmin Dec-ymax Dec-xmax.
Coord3 = 'COORD3' Dec-ymin Dec-xmin Dec-zmin Dec-ymax Dec-xmax Dec-zmax.
```

Pour les coordonnées, le domaine de valeur possible est ainsi fixé pour chaque composante. Y est la coordonnée positive vers l'Est, X vers le Nord et Z l'altitude s'il n'y a pas d'autres indications (comme commentaire).

2.2.4.2. Longueur et Superficie

```
Longueur = 'DIM1' Dec-min Dec-max.
Superficie = 'DIM2' Dec-min Dec-max.
```

2.2.4.3. Angle

```
Angle = ( 'RADIANS' | 'GRADS' | 'DEGREES' ) Dec-min Dec-max.
```

Sans autre indication (comme commentaire), il est admis que GRADS et DEGREES sont définis dans le système de coordonnées national, RADIANS dans le système mathématique.

2.2.4.4. Domaine numérique

```
Domaine_numerique = '[' Dec-min '..' Dec-max ']' .
```

On définit un domaine numérique, sans que le type en fixe la signification précise. Celle-ci doit être donnée ou bien par le biais du nom d'attribut correspondant ou bien par un commentaire complémentaire.

2.2.4.5. Texte

```
Texte = 'TEXT' '*' NoPos-Longueur_max.
```

Par texte, on entent une suite quelconque de caractères qui ne doit pas excéder le nombre maximum indiqué. Pour éviter tout problème de transfert, il faut respecter les règles et recommandations du chap. 3.4.5.5. ci-dessous.

2.2.4.6. Date

```
Date = 'DATE' .
```

Pour permettre un traitement de la date pour elle-même, celle-ci n'est pas introduite comme un simple texte. Elle identifie le jour et se compose de l'année, du mois et du jour.

2.2.4.7. Enumération

```
Enumeration = '(' Element {',' Element} ')' .
Element = Nom-element [ Enumeration-subordonnee] .
```

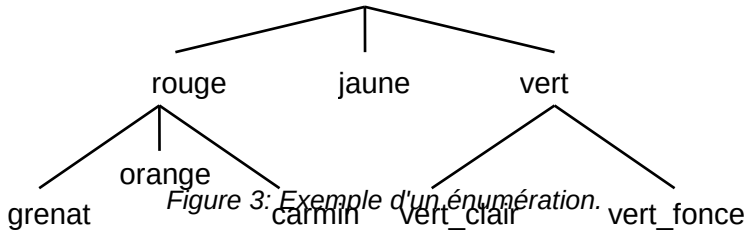


Figure 3: Exemple d'un énumération.

L'énumération fixe les valeurs admises d'un type donné. L'énumération n'est pas simplement linéaire mais a une structure arborescente.

Exemple :

```
(rouge (grenat, carmin, orange), jaune, vert (vert_clair, vert_fonce))
```

donne les valeurs possibles suivantes :

```
rouge-grenat rouge-carmin rouge-orange jaune vert-vert_clair
vert-vert_fonce.
```

Une subdivision est donnée par des parenthèses rondes. Le nombre de subdivisions est libre.

2.2.4.8. Mise en place de texte

```
Alignement_horizontal = 'HALIGNMENT'.
Alignement_vertical  = 'VALIGNMENT'.
```

Pour la préparation de plans, il faut définir la position des textes qui y figureront. Avec la notion d'alignement horizontal, on fixe le point par rapport auquel le texte se positionne. Il peut s'agir du bord gauche, du centre ou du bord droit du texte. L'alignement vertical fixe la position dans le sens de la hauteur du texte.

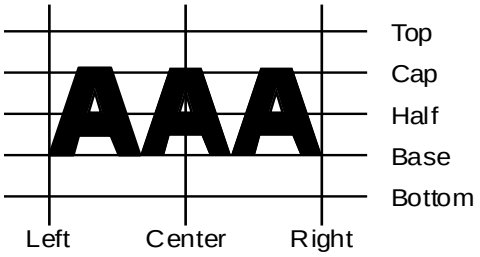


Figure 4: Mise en place de texte horizontale et verticale.

La distance Cap-Base exprime la hauteur des majuscules. Les jambages se situent dans la zone Base-Bottom.

Les alignements horizontal et vertical peuvent être compris comme énumération prédéfinie. L'effet est le même que si on les avait définis dans le cadre de la définition des domaines globaux comme suit :

```
HALIGNMENT = (Left, Centre, Right);
VALIGNMENT = (Top, Cap, Half, Base, Bottom);
```

2.2.5. Type de ligne

Par ligne, on comprend une suite continue de segments de ligne, une polyligne. La forme de la polyligne doit être décrite précisément (quelles sont les formes géométriques admises, les lignes peuvent-elles se couper ?) et les caractéristiques des points d'appui doivent être données.

```
Ligne = 'POLYLINE' Forme Points_appui [Intersec].
Forme = 'WITH' '('Type_Forme { ',' Type_Forme} ')'.
Type_Forme = ('STRAIGHTS' | 'ARCS' | Explication).
```

Figure 5: Différents suites continue de segments de ligne.

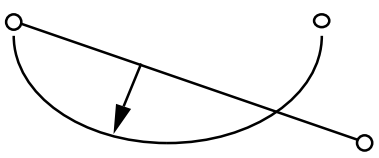
Il est donc possible d'admettre comme formes géométriques des combinaisons de droites, d'arcs de cercle et de formes spéciales. L'arc de cercle, dans le fichier de transfert, est représenté au moyen d'un point intermédiaire proche du point milieu de l'arc. Les arcs de cercle avec un très grand angle au centre, donc proches du cercle complet, sont numériquement instables et doivent donc être évités. Le point intermédiaire n'a pas la même signification qu'un point d'appui. Il ne sert qu'à la définition de l'arc de cercle. Pour le même arc, des points intermédiaires différents peuvent être utilisés d'un transfert à l'autre. Ils doivent toutefois rester proches du point-milieu. On s'assure ainsi que les systèmes concernés par le transfert restent libres dans leur définition interne de l'arc de cercle.

INTERLIS ne fixe pas des formes géométriques plus complexes. Toutefois, pour que la possibilité de décrire et transférer de telles formes subsiste, les cas particuliers peuvent être donnés comme explication (par ex. pour des courbes d'interpolation). Le mode de description et de codification du fichier de transfert est alors l'affaire des partenaires au transfert.

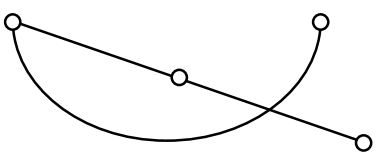
```
Intersec = 'WITHOUT' 'OVERLAPS' '>' Dec.'
```

On peut demander qu'une ligne ne se coupe ni avec elle-même ni avec une ligne de même attribut mais d'un autre objet. Des recoupements sont autorisés lorsque l'arc de cercle d'origine A coupe la ligne entre les points d'appui AB en un point X, tels qu'il n'y a pas d'autres points d'appui entre A et B, et tels que la flèche de l'arc entre A et X ne dépasse pas la tolérance admise.

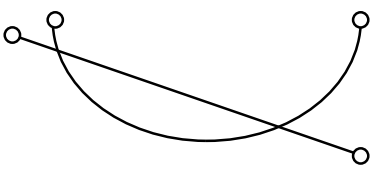
Cette règle a deux justifications: d'une part de petits recoupements sont inévitables dans certains cas pour des raisons numériques (cercle tangent), d'autre part lors de la reprise de données qui ont été saisies à l'origine graphiquement, des recoupements plus grands (p.ex. quelques cm) doivent êtres tolérés si l'on veut éviter un énorme travail d'épuration. La tolérance doit être exprimée dans les mêmes unités que les coordonnées des points d'appui.



a) La longueur de la flèche ne doit pas dépasser la tolérance donnée



b) Recoupement de segments non autorisé, car entre les points d'appui communs et le point d'intersection devrait se trouver un autre point d'appui



c) Recoupement de segments non autorisé, car n'émane pas de points d'appui communs

Figure 6: Tolérance (a) et recoupements pas tolérées (b+c) des suites continue de segments de ligne.

```
Points_appui = 'VERTEX' (Coord2 | Coord3 | Nom-du_type_de_coord)
               ['BASE' Explication].
```

Comme points d'appui de ligne, on désigne les points qui ont une signification particulière (début et fin de ligne, points d'inflexion).

En premier lieu, le domaine des coordonnées est défini. Des conditions supplémentaires peuvent être décrites au moyen d'une explication. Par exemple, on peut ainsi exiger que les coordonnées ne soient pas quelconques mais qu'elles correspondent à celles de points contenus dans une table donnée.

Les points intermédiaires d'arcs de cercle ne sont pas des points d'appui. Leurs coordonnées sont comprises dans le même domaine que celle des points d'appui.

2.2.6. Types de surface

Par surface, on comprend une portion de plan délimitée par des polylignes. Une surface est toujours délimitée par un bord extérieur contigu (limite extérieure). Si toute la portion de plan sise à l'intérieur de cette limite n'appartient pas à la surface, il s'agit alors d'enclaves qui sont également décrites par d'autres bords (bords intérieurs).

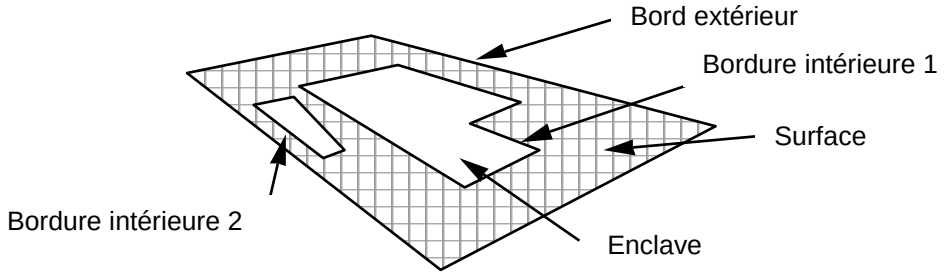


Figure 7: Surface avec bord et enclaves.

Des surfaces partielles appartiennent uniquement à la même surface, si elles ont une connexion massive. Si elles se touchent en un point seulement, elles forment deux surfaces.

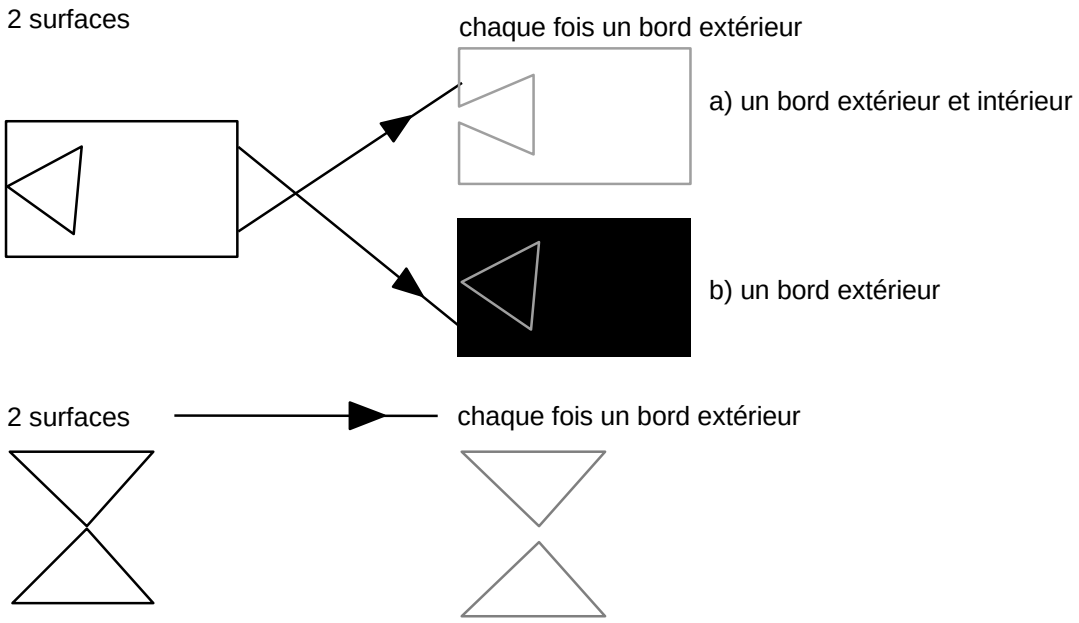


Figure 8: Exemples de surfaces partielles. INTERLIS permet les variantes a et b.

INTERLIS permet de préciser si les surfaces des différents objets d'une même table peuvent ou non se recouper.

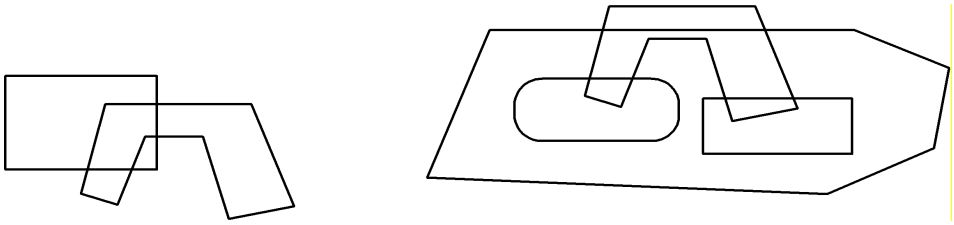


Figure 9: Surfaces (SURFACE).

Surface

Pour décrire des surfaces qui peuvent se chevaucher en entier ou partiellement, c'est-à-dire qui n'ont pas seulement des points de bords en commun, le type d'attribut géométrique SURFACE est à disposition (voir exemples figure 9).

Lot ou partition du territoire

On appelle un lot ou une partition du territoire un ensemble de surfaces qui couvrent complètement la totalité du territoire et dont les limites ne se recoupent pas.

Une partition du territoire forme une sorte de surface solde (environnement extérieur) avec un nombre quelconque d'enclaves. Pour décrire une partition du territoire, le type d'attribut géométrique AREA est à disposition.

A un objet d'une partition du territoire appartient exactement une surface du lot. A une surface du lot appartient au maximum un objet de la partition du territoire. Il est interdit qu'une des deux surfaces du lot, qui ont un bord commun, n'appartienne pas à un objet de la partition du territoire.

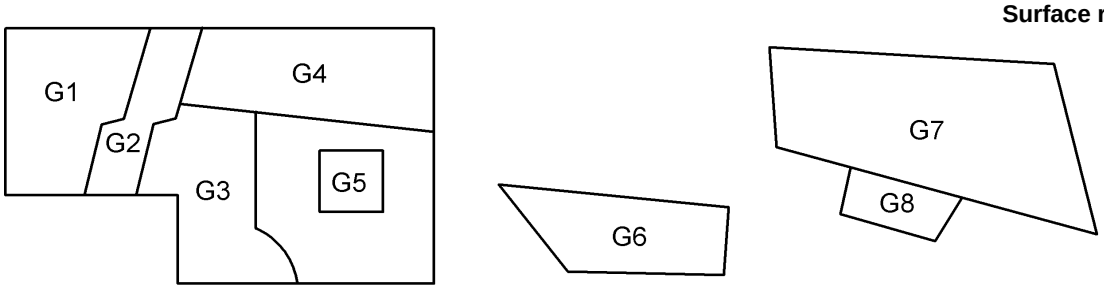


Figure 10: Partition du territoire (AREA).

Syntaxe:

```
TypeSurface = ('SURFACE' Forme Points_appui [Intersec]
              |'AREA' Forme Points_appui Intersec)
              [Linattrdef].
Linattrdef = 'LINEATTR' '='
               Attributs [Identification]
             'END'.
```

Il est interdit d'utiliser l'indication OPTIONAL avec le type d'attribut géométrique AREA.

Structure des tables

A la différence de tous les autres types d'attribut, dans le cas des surfaces (SURFACE) et des lots (AREA), une autre table est implicitement formée, qui contient les lignes. Le nom de la table est <nom-de-la-table-principale>_<nom-attribut>.

Le découpage des bords en lignes individuelles et le sens de direction des lignes est quelconque et peut être différent lors de divers transferts de données.

En cas de besoins, on peut définir des attributs pour les polylignes (Linattrdef). Dans ce cas d'autres types de surfaces ne sont pas admis.

Lignes de partitions du territoire

Des objets lignes d'une partition du territoire doivent former des vraies limites. Il est interdit d'avoir des lignes qui ont la même surface des deux côtés (figure 11).

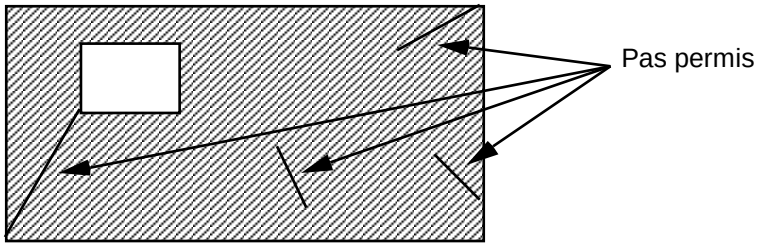


Figure 11: Configuration de surface interdite.

Sur toute la longueur d'un objet ligne, les surfaces des deux côtés ne doivent pas changer. Des lignes ne doivent pas franchir un vrai noeud (c'est-à-dire un point de rencontre de plus de deux lignes).

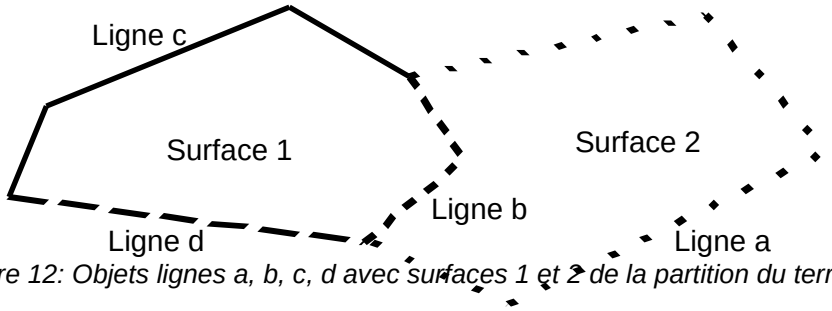


Figure 12: Objets lignes a, b, c, d avec surfaces 1 et 2 de la partition du territoire

Dans l'exemple de la figure 12, la table principale contient les objets surface 1 et surface 2 du lot, la table des lignes les objets ligne a, ligne b, ligne c et ligne d.

La subordination des objets lignes des surfaces aux objets de la partition du territoire est définie par la détermination d'un point à l'intérieur du lot comme attribut à l'objet lot. Ce point de référence ne doit pas être dans la zone de recoupement des lignes. La zone permise correspond à la surface comprise à l'intérieur des traitillés (voir figure 13).

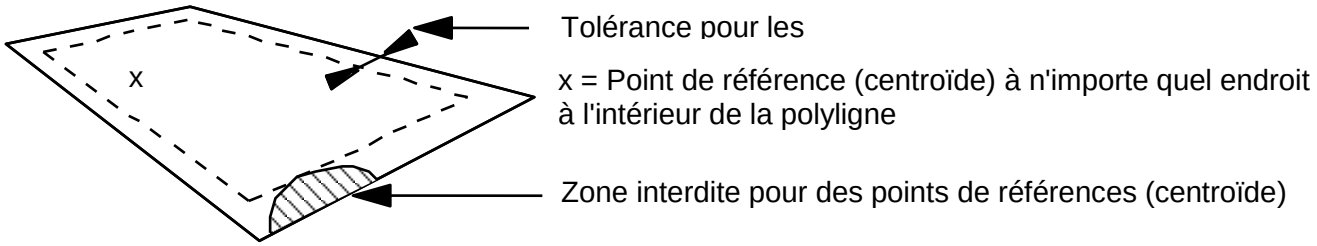


Figure 13: Secteur pour le point de référence.

D'un transfert de données à l'autre, la position du point de référence peut être différente.

2.2.7. Exploitations

```
Exploitations = 'DERIVATIVES' Nom-du_modele ';'
                  [DefDomaine-global] (* Theme *)
                'END' Nom-du_modele '.'
```

Du point de vue de la structure, une exploitation se construit comme un modèle. Elle ne définit toutefois pas de nouvelles données. Les valeurs individuelles des attributs sont des exploitations, fonctions des données du modèle.

2.2.8. Vues

Les vues permettent de décrire l'ordonnancement des données du modèle et des restitutions dans le fichier de transfert, en agissant sur quatre points :

- lors de l'édition de lignes, on peut exiger, en sus des coordonnées des points, des informations supplémentaires (VERTEXINFO)
- en cas d'attribut relationnel entre deux tables, on peut exiger que l'édition de la table désignée comprenne les données de l'autre table (<-).

Exemple:

```
MODEL X
  TOPIC Y =
    TABLE  A =
      a1 : TEXT * 10;
      a2 : TEXT * s;
    IDENT
      a1;
    END A;

    TABLE B =
      b1 : TEXT*12;
      b2 : ->  A;
    IDENT
      b1;
    END B;
:
:
VIEW X
  Y.A <- B.b2;
```

Dans le cadre de l'édition de la table A, pour chaque objet a, les données des objets de B relatives à cet objet a sont éditées comme attributs (cf. 3.4.7). Si toutes les relations sont ainsi prises en compte, la table B elle-même ne sera pas éditée.

- Pour les surfaces et les lots, on peut demander que la géométrie n'apparaisse pas dans le cadre de la table des lignes, mais directement comme attribut (cf. 3.4.7.) (CONTOUR).
- Dans le cas des lots, se pose encore la question de savoir si la surface extérieure aux limites extrêmes (périphérie, environnement) doit être restituée comme objet ou non. Ceci est l'objet de la commande WITH PERIPHERY.

Syntaxe:

```
Vue = 'VIEW' Nom-du_modele
        { Nom-du_theme '.' Nom-de_la_table ':' Def_de_la_vue
        {',' Def_de_la_vue} ';' }
      'END' Nom-du_modele '.' .

Def_de_la_vue = ('VERTEXINFO' Nom-LinAttr Explication
                 | 'WITH' 'PERIPHERY' Nom-AttrLot
                 | 'CONTOUR' Nom-AttrLot_ou_Surf ['WITH' 'PERIPHERY']
                 | '<-' Nom-de_la_table   '.'  Nom-attribut ).
```

2.2.9. Format

Pour le transfert, il faut déterminer si le fichier s'écrit selon les règles du format fixe ou du format libre (cf. 3.2.). Dans le cas du format fixe, il faut déterminer la longueur des lignes et la longueur des identificateurs.

```
Format = 'FORMAT' ('FREE'
           | 'FIX' 'WITH' 'LINESIZE' '=' NoPos ','
             'TIDSIZE' '=' 'NoPos)';'.
```

2.2.10. Codification

Pour que les caractères puissent être reconnus par le système récepteur sans ambiguïté, il faut définir le code utilisé. Si aucune définition de caractères n'est précisée, le code 8 bits de la norme ISO est applicable (ISO 8859-1:1987). Sinon il faut préciser le code utilisé par une explication. Pour les signes autorisés dans les attributs en texte, il faut tenir compte des exigences définies 3.4.5.5.

Pour quelques cas particuliers, on utilise des caractères spéciaux dans le fichier de transfert. INTERLIS définit des propositions (DEFAULT) qui peuvent être modifiées de cas en cas. Il faut alors préciser le code du caractère désiré dans la codification choisie.

Il faut régler les cas particuliers suivants :

Dans le cas d'attributs-textes, on peut rencontrer des blancs. Pour ne pas perturber la suite logique des champs (cf. 3.1.), INTERLIS utilise habituellement (DEFAULT) le trait de soulignement (_) (ASCII 0x5f).

Pour des attributs optionnels, les valeurs peuvent être non définies. Dans ces cas, à la place d'une valeur, on met un caractère spécial. INTERLIS propose habituellement @ (ASCII 0x40).

Il peut arriver que la longueur de la ligne ne suffise pas pour recevoir tous les caractères. La ligne logique est alors répartie sur plusieurs lignes physiques (cf. 3.1.) à la fin desquelles il faut insérer un caractère de "suite". INTERLIS utilise habituellement \ (ASCII 0x5c).

Lors du transfert, chaque objet reçoit un identificateur unique à l'intérieur de la table. D'une part les relations peuvent être ainsi facilement définies, d'autre part l'identification de l'objet peut être une bonne aide en cas de problème. La fabrication de l'identificateur de transfert est en principe libre. Il est considéré comme un texte de longueur quelconque. En cas de format fixe, il faut toutefois limiter sa longueur (cf. ci-dessus). Pour faciliter l'établissement de liens sur le système récepteur, il est préférable de préciser d'avantage le type d'identificateur. Avec I16 ou I32, on spécifie que l'identificateur est un nombre entier de précision 16 ou 32 bits.

```
Codification = 'CODE'  [Font] CarSpecial  ID_de_transfert  'END' '.' .
Font = 'FONT' '=' Explication  ';' .
CarSpecial = 'BLANK' '=' ('DEFAULT' | Code) ','
             'UNDEFINED' '=' ('DEFAULT' | Code) ','
             'CONTINUE' '=' ('DEFAULT' | Code) ','
ID_de_transfert = 'TID' '=' ('I16' | 'I32' | 'ANY' | Explication) ';' .
```

3. Construction du fichier de transfert

3.1. Structure propre au système

Un fichier de transfert - indépendamment de son contenu - se compose d'une suite de lignes logiques. Dans les descriptions qui suivent, la fin d'une ligne sera toujours notée NL. La construction d'une ligne logique peut se considérer de deux points de vue:

Structure physique

Une ligne logique peut se décomposer en un nombre quelconque de lignes physiques. Le début de la première et la fin de la dernière ligne physique ne nécessitent aucune mention particulière. La fin d'une ligne physique qui ne correspond pas à la fin d'une ligne logique doit comporter le caractère "suite" (habituellement on utilise \). Avec la description de la codification (cf. 2.2.10.), on peut définir un autre caractère. Au début de la ligne "suite", on introduit la marque CONT comme reconnaissance de ligne et un caractère blanc. Les caractères avant le caractère suite appartiennent à la ligne logique, et les caractères après le caractère blanc à la ligne suivante, à l'exclusion du caractère suite, du caractère CONT et du caractère blanc.

INTERLIS ne règle pas la façon dont le fichier de transfert est décomposé en lignes physiques. Ceci est l'affaire d'un niveau interface plus bas. Sans convention particulière, les lignes physiques sont fermées par LF (Line Feed, ASCII OxA). Immédiatement avant LF, on peut trouver un caractère facultatif CR (ASCII OxD).

Structure logique

La ligne logique comprend une suite de champs. L'ordonnancement des champs dans la ligne et la relation avec la structure physique est dépendante de la façon de formater la partie "données" du fichier de transfert (cf. 2.2.9.).

La description ci-après de la structuration des données et de la codification de chaque champ ne se réfère qu'à la composition logique des lignes. La question de la séparation des champs entre eux dépend du choix du format fixe ou libre (cf. 3.2.). Dans les deux cas, il faut au minimum un caractère de séparation (blanc ou tabulateur) entre deux champs d'une même ligne logique.

3.2. Format libre et format fixe

La partie "données" d'un fichier de transfert peut avoir un format libre ou fixe. Le format libre est bien adapté aux langages de programmation modernes comme PASCAL ou MODULA, alors que le format fixe convient mieux aux langages tels le BASIC et le FORTRAN.

3.2.1. Format libre

Dans le format libre, les champs sont séparés au moins par un caractère blanc ou un tabulateur. Les longueurs des champs sont égales aux longueurs des valeurs actuelles. Une ligne logique peut être interrompue à n'importe quel endroit, même au milieu d'un champ (cf. 3.1.). Les caractères nécessaires à la séparation doivent être simplement éliminés lors du décodage, pour rétablir la suite des caractères de la ligne logique.

S'il n'y a que des caractères indéfinis sur le reste d'une ligne logique, il est possible, dans le format libre, de mettre à leur place deux caractères indéfinis successifs et de fermer la ligne logique.

3.2.2. Format fixe

Dans le format fixe, les champs doivent être séparés par un seul caractère blanc. Chaque champ a la forme suivante :

Caractère initial

(si l'attribut correspondant est facultatif) Ce caractère indique si la valeur est définie ou non. Si la valeur est définie, le caractère est blanc. Si la valeur n'est pas définie, on utilise le caractère non-défini (cf. chap. 3.4.4.).

Caractères suivants

n caractères suivants pour la valeur. Pour un champ déterminé, il faut réserver le plus grand nombre de caractère qu'il est nécessaire pour la valeur en question (longueur maximum). Les définitions concrètes se trouvent au chap. 3.4. Les champs numériques sont toujours alignés à droite, les champs alphanumériques à gauche. Les caractères vides à la fin d'un champ restent des caractères vides, tandis qu'au début ou à l'intérieur d'un champ alphanumérique, ils sont occupés par le signe de remplacement (cf. chap. 3.4.5. et 2.2.10.).

Dans le cas où tous les champs restants n'ont pas suffisamment de place dans le cadre de la longueur déterminée de la ligne pour le format fixe (cf. chap. 2.2.9.), la ligne doit être subdivisée. Cette subdivision se fait de telle sorte qu'un champ donné puisse être entièrement compris dans une ligne. Une ligne "suite" ne commencera que lorsque le champ actuel - pour autant qu'il ne s'agisse pas du dernier champ d'une ligne logique - et les caractères de séparation (caractères vides) et de suite, n'auront plus la place suffisante sur la ligne actuelle. Les caractères de séparation et de suite doivent donc toujours se trouver sur l'ancienne ligne.

Le format fixe est donc ainsi fait qu'il puisse également être lu selon les règles du format libre.

Pour tenir compte de la lecture en format fixe, la succession des lignes et des marques de lignes a été choisie de telle sorte que la prochaine ligne puisse être lue avec le format prédéterminé. Une relecture (Reread) n'est donc pas nécessaire. Dans de nombreux cas, plusieurs lignes se suivent avec la même marque de ligne. La fin de la séquence est formée d'une ligne qui ne contient qu'une seule marque de ligne typique de fin de la séquence et on teste si la marque de ligne n'indique pas la fin de la séquence. Dans le cas des figures géométriques, il faut tenir compte en sus que le même format peut se rencontrer toutefois avec des marques de lignes différentes pour des formes géométriques différentes.

3.3. Structuration selon le contenu

```
Fichier_de_transfert = 'SCNT' NL Description_du_contenu
                       ('MOTR' NL Description_du_modele_et_du_transfert)
                       |'MTID' Nom-transfert NL)
                      Donnees
                       'ENDE' NL.
Description = { Ligne_de_description } '////' NL .
Ligne_de_description = quelconque_sauf_////_au_début_de_ligne NL.
Donnees = { ModeleD }.
ModeleD = 'MODL' Nom-du_modele NL
          { ThemeD }
          'EMOD' NL.
```

Un fichier de transfert contient tout d'abord une indication sur le contenu, notamment le secteur géographique. Cette indication est définie comme pur texte sans formalisation. Les partenaires intéressés au transfert peuvent eux-mêmes définir une formalisation. Puis intervient la description du transfert (cf.

chap. 2.2.) respectivement un avis, qui identifie d'une manière univoque le transfert reconnu par chaque partie. Enfin suivent les données. Celles-ci sont organisées par niveau, dans la suite de la définition du modèle. Il est admissible que tous les thèmes du modèle ne soient pas transférés. Si un thème l'est, il l'est complètement.

```
ThemeD = 'TOPI' Nom-du_theme NL
         { TablesD }
         'ETOP'.
TablesD = 'TABL' Nom-de_la_table NL
          { ObjetD }
          [ Perimetre ]
          'ETAB' NL .
ObjetD = 'OBJE' Texte-TransferID AttributD .
AttributD = { Attribut_de_base |Coordonnee-de_ref_lot } NL
            { Attribut_de_ligne } { Asurface|Alot }   { TablesA }
[ Perimetre ] = 'PERI' Texte-TransferID  NL.
```

La suite des tables est donnée par la suite des définitions à l'intérieur d'un thème. Tous les objets d'une table sont édités. L'ordre est alors quelconque. Dans le cas de lots, on indique encore, après le dernier objet, quel objet forme le périmètre, pour autant que celui-ci doive être donné (cf. chap. 2.2.8.).

L'ordonnancement des attributs par objet correspond en principe à celui des attributs dans la description. Cette règle ne souffre aucune exception dans le cas des attributs de base.

Dans les cas des lots ou des surfaces, les règles suivantes s'appliquent:

édition de la géométrie de surfaces dans le cadre de la table des lignes:

La table des lignes est éditée immédiatement après la table principale. S'il y a plusieurs attributs de surface, l'ordonnancement des tables de lignes est donné par celui de ces attributs.

Dans l'objet ligne, le premier attribut est un attribut de relation qui contient l'indication de relation à l'objet surface (cf. chap. 2.2.6.)

édition de la géométrie de lots dans le cadre de la table des lignes:

La table des lignes est éditée immédiatement avant la table principale.

Dans l'objet lot, à la place de l'attribut du lot, on trouve les coordonnées du point de référence situé à l'intérieur de l'objet (cf. chap. 2.2.6.)

édition de la géométrie avec l'objet lot ou surface (CONTOUR) :

L'ordonnancement est le même que celui de la définition des attributs. L'indication du périmètre ne se fait dans ce cas que si l'objet correspondant a été édité, c'est-à-dire si la commande WITH PERIPHERY avait été exécutée.

Les attributs de ligne et d'exploitation (attributs de surfaces-, de lots, exploitation de tables) ne sont pas édités directement sur la ligne de l'objet. Leur édition intervient en fonction des attributs de base et de ceux des lots ou des surfaces. Les attributs de ligne sont édités en premier lieu, puis suivent les attributs d'exploitation dans l'ordre de la définition.

3.4. Définition de la codification

3.4.1. Marque de ligne

Le premier champ de chaque ligne logique contient la marque de ligne. Celle-ci se compose toujours de 4 caractères.

3.4.2. Noms des couches et des tables
La longueur maximum des noms de couches et de tables est de 24 caractères. D'éventuels caractères supplémentaires sont ignorés.

3.4.3. Identification de transfert
Cette identification utilise un champ. Du point de vue structure, elle est traitée comme un texte. La longueur maximum est donnée par les paramètres de transfert (cf. chap. 2.2.9.) pour le format fixe. Elle n'est pas déterminée en cas de format libre.

3.4.4. Attributs non-définis
Les attributs qui sont signalés dans la description du modèle avec "OPTIONAL" peuvent ne pas avoir une valeur définie. Pour le montrer, on utilise le caractère "non-défini" qui est habituellement le caractère @. Dans le cadre de la définition de la codification (cf. chap. 2.2.10.), on peut en définir un autre. Dans le format libre, le caractère non-défini prend la place de la valeur. Dans le format fixe, il est positionné à l'emplacement prévu. Cet emplacement est indépendant de la place réservée pour la valeur, de sorte que le caractère non-défini n'interfère pas avec le format réservé pour la valeur elle-même (cf. chap. 3.2.2.). Aux positions prévues pour la valeur figurent alors des caractères vides.

3.4.5. Attributs de base
On définit comme attributs de base les attributs relationnels et les attributs ayant un type de base comme domaine.

3.4.5.1. Attributs relationnels
Les attributs relationnels occupent un champ. L'identification de transfert de l'objet considéré est donnée comme valeur de ces attributs (cf. chap. 3.4.3).

3.4.5.2. Attributs de nombres décimaux
Les attributs, dont le domaine est indiqué avec "Dec" (coordonnées, longueurs, etc.) sont toujours exprimés conformément à la définition du domaine. L'indication sans facteur de multiplication vaut comme format. Le point décimal est mis dans la mesure où il est indiqué dans la définition.

3.4.5.2. Coordonnées
Les coordonnées se composent de deux (COORD2) ou trois (COORD3) champs. Le premier champ contient la valeur Est, le deuxième la valeur Nord, le troisième l'altitude. Si les coordonnées ne sont pas définies, tous les champs doivent recevoir le caractère non-défini. La longueur maximum de la valeur est fixée séparément pour chaque composante.

3.4.5.3. Longueurs, superficies et angles
Ces attributs se composent chacun d'un seul champ.

3.4.5.4. Domaines numériques
Chaque valeur de domaine numérique occupe un champ.

3.4.5.5. Texte
Un attribut "texte" se compose d'un seul champ. Pour éviter que des caractères vides à l'intérieur d'un texte ne soient interprétés comme une fin de champ, il faut remplacer ces caractères vides par un autre caractère. Sans autre indication, on utilise le soulignement (ASCII-0X5F). Des différences peuvent être définies dans la description des paramètres de transfert (cf. chap. 2.2.10). Pour éviter des problèmes de représentation des caractères sur les divers systèmes, seuls sont autorisés les caractères de la liste ci-après - sans convention particulière en sus des caractères 7 bits ASCII - définis dans le jeu des

caractères PC, mais en tout cas seulement des caractères alphabétiques (cf. ISO 6937/2-1983) de la norme ISO-8859-1 qui contient tous les caractères accentués également en majuscules.

Caractères utilisables du jeu PC :

```
Ä, â, ä, à, á, æ,
Ç, ç,
É, ê, ë, è, é,
î, ï, ì, í,
Ñ, ñ,
Ö, ô, ö, ò, ó,
Ü, û, ü, ù, ú
```

3.4.5.6. Date

La date occupe un champ. Elle s'exprime sous la forme AAAAMMJJ (AAAA pour l'année, MM pour le mois [01 à 12], JJ pour le jour).

Par exemple, le 1er août 1991 s'écrit 19910801.

3.4.5.7. Enumération

Une valeur d'énumération occupe un champ. L'indication se fait numériquement par le numéro courant. Toute valeur peut être numérotée, donc tous les éléments de la définition qui ne présentent pas une sous-énumération. La numérotation commence avec 0. La longueur maximale est donnée par le numéro le plus élevé.

Exemple:

Dans le cas de l'énumération des couleurs:

```
(rouge (grenat, carmin, orange), jaune, vert (vert_clair, vert_fonce),
bleu, violet);
```

on attribue les valeurs suivantes :

```
0 rouge-grenat  1 rouge-carmin ... 3 jaune 4 vert_clair ....6 bleu 7
violet
```

3.4.5.8. Alignement

Comme les alignements horizontal et vertical peuvent se concevoir comme une énumération pré-définie, ceux-ci suivent les règles de l'énumération.

3.4.6. Attributs de lignes

Par attribut de lignes, on désigne les attributs dont le domaine est défini comme ligne. Les segments individuels d'une polyligne s'écrivent sous la forme d'une séquence de lignes. Il faut une ligne à la table par point d'appui, et des lignes supplémentaires selon le genre de liaison géométrique. Dans le cas d'arc de cercle, on donne un point supplémentaire à proximité du point-milieu. Pour des liaisons géométriques spéciales, la codification est libre, dans le cadre des règles générales de codification d'INTERLIS. Pour pouvoir être lue avec le même format, une ligne doit s'écrire sous la forme "Marque de ligne Point Autre-Indication NL".

```
Attribut_de_ligne = Pt_de_depart (* Liaison *)
                    'ELIN' NL.
Pt_de_depart = 'STPT'  Point-de_depart  NL.
Liaison = (Droite | Arc_de_cercle | Liaison_speciale)  'LIPT'  Point  NL.
Droite = .
```

```
Arc_de_cercle = 'ARCP' Point-intermediaire NL.
Liaison_speciale = Formule_de_liaison_speciale .
Point = Coordonnee [ Information_sur_le_point ].
```

Pour le transport sur de nombreux systèmes, il est pratique d'ajouter aux coordonnées des points d'autres informations. La définition est donnée au chap. 2.2.8. (VERTEX INFO). La codification est laissée au libre choix des intéressés, mais doit satisfaire aux règles des chap. 3.1 et 3.2). Si l'on veut lire en format fixe, il faut faire attention à pouvoir lire malgré tout avec le même format là où on doit s'attendre à recevoir des informations différentes.

3.4.7. Attributs d'exploitation

On désigne par attributs d'exploitation des attributs qui n'appartiennent pas obligatoirement aux tables existantes, mais qui sont issus de la compilation d'autres tables. Ceci vaut avant tout pour les types surface et lot qui impliquent une autre table et pour des attributs d'indice. Les possibilités de définition sont décrites au chap. 2.2.8. Seule la construction correspondante des fichiers est représentée ici :

```
SurfaceA = (* Bord *) 'EFLA' NL.
LotA = SurfaceA.
Bord = 'EDGE' NL (* LinAttr  Sequence_de_ligne *)
       'EEDG' NL .
LinAttr =  'LATT' AttributD .
```

La suite des bords est quelconque. Un bord peut être décomposé en un nombre quelconque de polylignes. La suite des segments de ligne doit être choisie de telle sorte que la surface décrite se situe à droite de la ligne actuelle (le bord extérieur est parcouru dans le sens des aiguilles de la montre, les enclaves dans le sens contraire). Les attributs de l'objet ligne sont édités par la marque de ligne LATT. Celui-ci contient l'identificateur de l'objet comme premier champ. Les attributs de lignes suivent dans l'ordre et dans la mesure ou ils ont été définis (cf. chap. 2.2.5.).

```
TablesA = 'TABA'  Nom-de_la_table NL
          { 'SOBJ' AttributD NL }
          'ETBA' NL.
```

On indique tous les objets de la table interprétée qui ont une relation (attribut relationnel) avec l'objet considéré. Par objet, on obtient une séquence SOBJ AttributD NL. L'attribut relationnel lui-même n'est pas édité.

4. Le compilateur INTERLIS

Le compilateur INTERLIS sert à vérifier formellement une définition de transfert.

Le compilateur INTERLIS produit, dans sa forme la plus simple, à partir d'une description de transfert, une liste avec les formats qui peuvent intervenir dans le fichier de transfert correspondant, et une liste qui contient la description donnée avec les messages d'erreurs éventuelles.

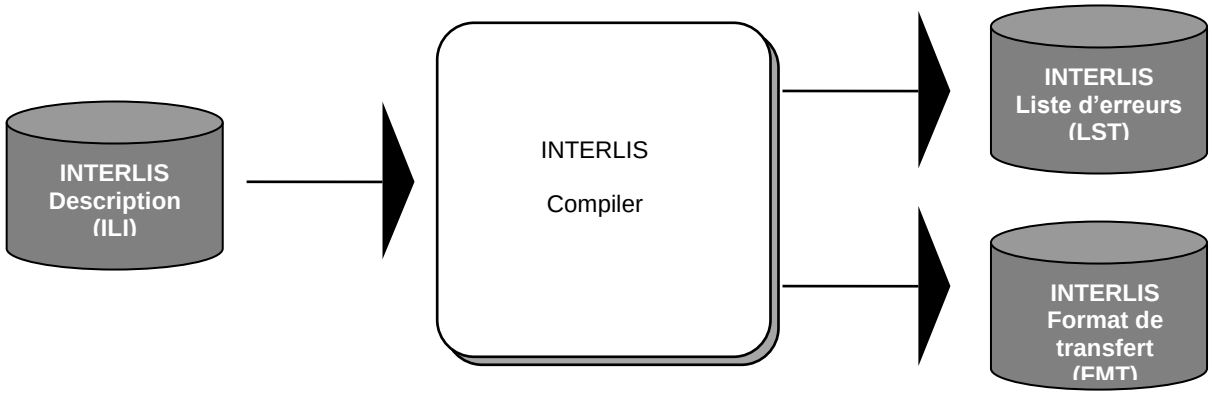


Figure 14: Le compilateur INTERLIS.

Le compilateur teste la description de transfert livrée comme input et fabrique un fichier avec tous les messages d'erreur ainsi que - en cas de définition correcte - un fichier de description des formats de toutes les tables à transférer.

Si la définition de transfert spécifie le format libre, un format d'écriture est généré avec les caractéristiques suivantes :

- longueur des lignes: 60 caractères
- dimension des identifications de transfert: 1 caractère.

5. Exemple

L'exemple traité au chap. 2.2.2. est repris ici et traité plus en détail.

```
TRANSFER Exemple;

MODEL Exemple

  DOMAIN
    CoordN = COORD2 480000.00   60000.00
                    850000.00  320000.00;

  TOPIC Couverture_du_sol =

    TABLE Surfaces_sol =
      Genre : (batiment, revetement_dur, verte, eau,
               boisee, sans_vegetation);
      Forme : AREA WITH (STRAIGHTS, ARCS) VERTEX CoordN
              WITHOUT OVERLAPS > 0.10;
    NO IDENT
      !! recherche via geometrie ou batiment
    END Surfaces_sol;

    TABLE Batiments =
      NoAss : TEXT*6;
      Surface : -> Surfaces_sol // Genre = batiment //;
    IDENT
      NoAss;   !! supposition que NoAss est univoque
      Surface; !! une seule surface est subordonnee a ce batiment
    END Batiments;

  END Couverture_du_sol.
END Exemple.

FORMAT FREE;

CODE
  BLANK = DEFAULT, UNDEFINED = DEFAULT, CONTINUE = DEFAULT;
  TID = ANY;
END.
```

Au sens du modèle de données relationnel, la structure peut être schématiquement représentée comme suit :

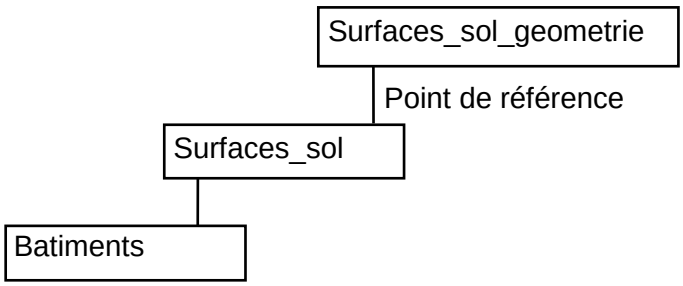


Figure 15:Schéma sémantique des liaisons de l'exemple

En premier lieu, ce sont des surfaces de la couverture du sol qui sont décrites. Pour différencier leurs caractéristiques et leurs natures, on a introduit l'attribut "genre". Dans le cas des bâtiments, il faut introduire encore d'autres informations, notamment le numéro d'assurance. Comme ces informations ne concernent que les bâtiments et non pas les autres types de surfaces de la couverture du sol, on construit une table spéciale qui est liée à celle des surfaces de la couverture du sol.

Pour montrer un fichier de transfert concret, prenons un cas simple :

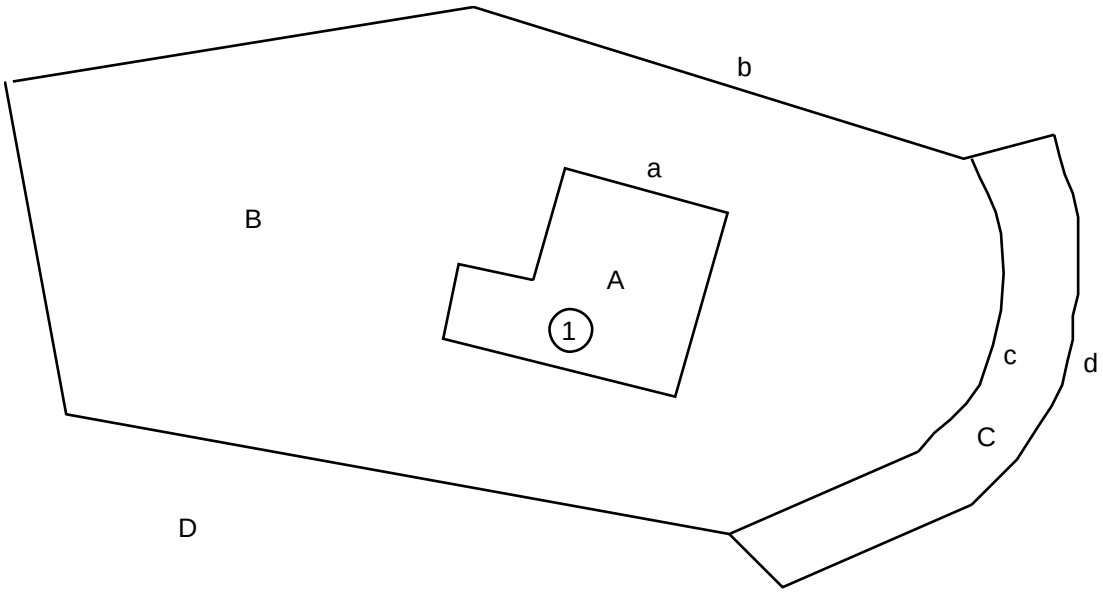


Figure 16: Exemple.

Un bâtiment (avec le numéro d'assurance 958) se trouve dans une surface verte, en bordure d'une surface à revêtement dur (route).

Les minuscules servent à l'identification des lignes, les majuscules des lots et les chiffres encerclés des bâtiments.

Le compilateur INTERLIS traduit la description du transfert dans les formats suivants :

```
Topic Couverture_du_sol
=======================

Table Surfaces_sol_geometrie
------------------------
OBJE-Format

OBJE  1

1: Ident_objet Identification de l'objet de la ligne

Followed by
Line-Records of line-attr   Line:

llab  111111.11  222222.22

1: Coordonnée
2: Coordonnée

Sequence of objects closed by
ETAB-Record

Table Surfaces_sol
------------------
OBJE-Format

OBJE  1  2 333333.33 444444.44

1: Ident_objet    La signification des champs est indiquée
2: Genre
3: Forme
4: Forme

Followed by
ETAB-Record

Table Batiments
---------------
OBJE-Format

OBJE  1  222222  3

1: Ident_objet Identification de l'objet du bâtiment
2: NoAss
3: surface Pointeur aux Surfaces_sol

Sequence of objects closed by
ETAB-Record
```

L'exemple donne donc le fichier de transfert suivant :

```
SCNT
Fichier de transfert de l'exemple
////
MTID selon description de l'exemple
MODL Exemple
TOPI Couverture_du_sol
TABL Surfaces_sol_Forme
OBJE a                          ID de la ligne
STPT 600146.92 200174.98
LIPT 600138.68 200187.51
LIPT 600147.04 200193.00
LIPT 600149.79 200188.82
LIPT 600158.15 200194.31
LIPT 600163.64 200185.96
LIPT 600146.92 200174.98
ELIN
OBJE b
STPT 600140.69 200156.63
LIPT 600118.19 200179.82
LIPT 600113.00 200219.97
LIPT 600148.30 200228.97
LIPT 600186.38 200206.82
ELIN
OBJE c
STPT 600186.38 200206.82
ARCP 600183.52 200188.65
LIPT 600170.18 200176.00
LIPT 600140.69 200156.63
ELIN
OBJE d
STPT 600186.38 200206.82
LIPT 600194.26 200208.19
ARCP 600190.75 200185.21
LIPT 600174.10 200169.00
LIPT 600145.08 200149.94
LIPT 600140.69 200156.63
ELIN
ETAB
TABL Surfaces_sol
OBJE A 0 600148.20 200183.48  A est ID_objet; 0 = batiment; Ptref.
OBJE B 2 600133.95 200206.06  B est ID_objet; 2 = verte; Ptref.
OBJE C 1 600168.27 200170.85  C est ID_objet; 1 = revetement_dur; Ptref.
ETAB
TABL Batiments
OBJE 1 958 A
ETAB
ETOP
EMOD
ENDE
```

Index

L'exemple au chapitre 5 a été corrigé par COSIG le 17.08.2016.