Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Département fédéral de la défense,
de la protection de la population et des sports DDPS

Office fédéral de topographie swisstopo

Transposition de modèles de géodonnées conceptuels

A l'exemple de modèles INTERLIS 2 retranscrits en SQL

Exemples de meilleures pratiques

Date de création: mai 2014

Mandant:
COSIG, Office fédéral de topographie swisstopo

Auteurs:
Inser SA & Eisenhut Informatik AG

Table des matières

1 Introduction

Le présent recueil d'exemples de transposition est à considérer comme un répertoire de meilleures pratiques et vise à aider toutes celles et tous ceux qui doivent passer du modèle conceptuel orienté-objet en INTERLIS 2 à un modèle relationnel simplifié en SQL. Pourquoi avoir dressé cet inventaire? Pour deux raisons principales qui sont d'une part les expériences réalisées dans le cadre de projets de transposition menés par COSIG et l'OFEV et d'autre part le fait que swisstopo assume, conformément à la LGéo / à l'OGéo, la tâche de fixer les exigences qualitatives et techniques à satisfaire par les géodonnées de base relevant du droit fédéral, notamment en matière de modélisation.
La liste des cas de figure types n'est pas close et peut naturellement être étendue. On peut aussi imaginer d'autres options de transposition pour les cas de figure dévoilés. Il s'agissait surtout de répertorier quelques solutions courantes. Il va de soi que le présent document est ouvert à d'autres possibilités, si bien qu'il est «dynamique» en ce sens.

2 Indications de portée générale

Un modèle conceptuel sert à la documentation et à l'harmonisation de géodonnées. Il fournit des informations sur la structure et la sémantique d'un jeu de géodonnées indépendamment de tout système. Il convient donc de réfléchir à certains points de portée générale avant de transposer un tel modèle conceptuel dans des systèmes de banques de données logiques. On peut par exemple s'interroger sur la finalité de l'application logique et sur les exigences techniques à satisfaire. Il faut surtout se défaire d'emblée de l'idée qu'un modèle conceptuel orienté-objet peut être converti tel quel en un modèle logique.
Le présent document vise à mettre en lumière des points particulièrement délicats de la transposition à l'aide d'exemples. On veillera toutefois à ne pas se concentrer sur une seule technologie de banques de données; on tâchera plutôt de fournir des exemples et de donner des conseils relatifs aux points qu'il s'agit d'observer.
Il y a généralement plusieurs possibilités pour convertir un modèle conceptuel en un modèle de banque de données concret. La question suivante revêt donc une grande importance: comment le modèle implémenté sera-t-il utilisé? Est-il par exemple opportun de recourir à une structure à plat ou vaut-il mieux créer une banque de données à la modélisation plus élaborée comportant de nombreuses tables et conditions? Quelles personnes ou quels processus (groupes d'utilisateurs) utiliseront par ailleurs la banque de données et avec quels logiciels? Il est également recommandé de faire appel à un spécialiste du domaine concerné issu du groupe des utilisateurs afin qu'il vérifie si la structure créée est effectivement utilisable par le public visé.
Les technologies mises en œuvre pour implémenter la banque de données jouent elles aussi un rôle prépondérant. Certains systèmes de banques de données permettent d'implémenter des conditions d'une telle sévérité qu'il est impossible de stocker des données non conformes dans la banque de données.
Les points suivants constituent des indications de portée générale dont il convient de tenir compte lors de la conversion de schéma:

- INTERLIS 2 est orienté-objet. Un modèle INTERLIS 2 doit donc être «désemboîté» et converti en un modèle relationnel; une clé primaire et une clé étrangère doivent par ailleurs être générées pour que des relations puissent être reproduites.

- INTERLIS 2 permet d'importer d'autres modèles ou des parties de modèles. La complexité réelle d'un modèle INTERLIS 2 peut donc être bien supérieure à ce que la première lecture laisse entrevoir.

- Les modèles INTERLIS 2 sont structurés en thèmes (topics) et en classes. De nombreux systèmes de banques de données ne connaissent que les schémas et les tables.

- A l'inverse, INTERLIS n'accepte des objets multiples (Multipart) que via des structures telles que celles prédéfinies dans les modèles de données CH-Base. Le cas échéant, les banques de données doivent être modélisées de telle façon qu'elles n'acceptent que des objets uniques (Singlepart).

- INTERLIS 2 accepte plusieurs géométries par classe. Rares sont les systèmes de banques de données de SIG à prévoir de tels cas de figure, si bien qu'une classe avec deux géométries doit généralement être scindée en deux ou trois tables.

- INTERLIS 2 accepte les attributs de structure. Ce n'est pas le cas de la plupart des systèmes de banques de données et des SIG proposés sur le marché, de sorte qu'une classe avec des attributs de structure doit généralement être scindée en deux tables ou plus.

- INTERLIS 2 accepte les conditions au sein d'une même classe, par exemple le fait qu'un attribut doive être plus grand ou plus petit qu'un autre attribut de cette classe. Dans de nombreux systèmes de banques de données, de telles conditions ne sont modélisables qu'avec difficulté.

- INTERLIS 2 accepte les énumérations / les domaines codés, ce qui n'est pas le cas de tous les systèmes de banques de données. Les tables de référence constituent par exemple un moyen pour transposer des énumérations.

Il convient par ailleurs d'indiquer qu'une banque de données qui prend en charge des données conformes à un modèle n'a pas forcément besoin d'être bâtie conformément à ce modèle elle-même. Une structure à plat peut par exemple être légitime. C'est pourquoi il faut préciser ici que le respect de nombreuses conditions peut être assuré par des processus d'importation et d'exportation.

3 Points importants de la conversion de schéma

3.1 Conversion de structures orientées-objet

Remarque: la numérotation permet d'associer plusieurs possibilités de simplification (colonne 5) à un même cas de figure (exemple identique, colonnes 2 et 4).
PK = primary key (clé primaire)
FK = foreign key (clé étrangère)

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
1.a	Héritage	1:1	CLASS A (ABSTRACT) = Attribut_1; END A; CLASS B EXTENDS A = Attribut_2; END B;	CREATE TABLE A (ID PK, Attribut_1) CREATE TABLE B (ID PK, Attribut_2)	Une table est créée pour chaque classe.
1.b	Héritage	Sur-classe	CLASS A (ABSTRACT) = Attribut_1; END A; CLASS B EXTENDS A = Attribut_2; END B;	CREATE TABLE A (ID PK, Attribut_1, Attribut_2)	Même exemple que pour le numéro 1.a. Cette fois-ci, une table est créée pour chaque classe racine. Restriction: lorsque de nombreuses tables renvoient à la classe racine.
1.c	Héritage	Sous-classe	CLASS A (ABSTRACT) = Attribut_1; END A; CLASS B EXTENDS A = Attribut_2; END B;	CREATE TABLE B (ID PK, Attribut_1, Attribut_2)	Même exemple que pour le numéro 1.a. Une table est créée pour chaque classe concrète.
2.a	Structures	Clé étrangère pour une	STRUCTURE A =	CREATE TABLE A (	La structure est modélisée sous forme de

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
		structure	Attribut_1 ; Attribut_2 ; END A; CLASS B = Attribut_3: A; END B;	ID PK, B_Attribut_3, Attribut_1, Attribut_2, FOREIGN KEY B_Attribut_3_FK (B_Attribut_3) REFERENCES B (ID)) CREATE TABLE B (ID PK)	table contenant une clé étrangère pour établir le lien avec la table principale B. Lorsque l'attribut de structure est de type LIST/BAG OF, il faut encore un champ d'index dans la table de la structure.
2.b	Structures	Clé étrangère pour un élément de structure dans la table principale	STRUCTURE A = Attribut_1 ; Attribut_2 ; END A; CLASS B = Attribut_3: A; END B;	CREATE TABLE A (ID PK, Attribut_1, Attribut_2) CREATE TABLE B (ID PK, Attribut_3, FOREIGN KEY Attribut_3_FK (Attribut_3) REFERENCES A(ID))	Même exemple que 2.a. Ici, la clé étrangère est contenue dans la table principale. Cette solution n'est pas envisageable lorsqu'un attribut de structure est défini comme LIST/BAG OF. La possibilité de définir un champ d'index fait défaut dans ce cas.
2.c	Structures	Eléments de structure comme JSON (ou XML) dans la table principale	STRUCTURE A = Attribut_1 ; Attribut_2 ; END A; CLASS B = Attribut_1: A; END B;	CREATE TABLE B (ID PK, Attribut_1 CLOB)	Le type de données dépend de la technologie, CLOB ou TEXT peuvent par exemple être utilisés.
2.d	Structures	Clé étrangère générique	STRUCTURE A = Attribut_1 ; Attribut_2 ; END A;	CREATE TABLE A (ID PK, Attribut_1, Attribut_2,	PARENTID est la clé étrangère générique dans la table de la classe avec l'attribut de structure. Toutefois, la même structure pouvant être utilisée dans différentes

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
			CLASS B = Attribut_1: A; END B;	PARENTID, PARENT_FQNAME VARCHAR(1000)) CREATE TABLE B (ID PK)	classes pour les attributs de structure, on ne sait pas dans quelle table la clé étrangère doit figurer. PARENT_FQNAME contient donc le nom qualifié de l'attribut de structure, soit «ModelName.TopicName.ClassName. AttrName».
3	Structures	Insérer les éléments de structure dans la table principale	STRUCTURE A = Attribut_1 : (D1,D2) ; Attribut_2 : (D3,D4) ; END A CLASS B = Attribut_3 : A; Attribut_4 : A ; END B;	CREATE TABLE B (ID PK, Attribut_3_1 (D1,D2), Attribut_3_2 (D3,D4), Attribut_4_1 (D1,D2), Attribut_4_2 (D3,D4))	On peut ainsi éviter la création d'un nombre trop élevé de tables. Restriction: lorsque les domaines sont très étendus. Les énumérations peuvent alors être transposées comme elles le sont aux exemples 17 et 18. Cette solution n'est pas envisageable lorsqu'un attribut de structure est défini comme LIST/BAG OF.
4.a	Associations	Sans table intermédiaire: uniquement pour des relations 1:1 et 1:n, sans contraintes de BD (DB-Constraints)	CLASS A = Attribut_1; END A; CLASS B = Attribut_2; END B; ASSOCIATION C = rA -- {1..*} A; rB -- {1} B; END C;	CREATE TABLE A (ID PK, Attribut_1, rB,) CREATE TABLE B (ID PK, Attribut_2)	Le lien entre deux tables est établi avec une clé étrangère. Cette solution devient malaisée lorsque l'association d'une extension contient des attributs.
4.b	Associations	Sans table intermédiaire: uniquement pour des relations 1:1 et 1:n, avec contraintes de BD (DB-Constraints)	CLASS A = Attribut_1; END A; CLASS B = Attribut_2; END B; ASSOCIATION C =	CREATE TABLE A (ID PK, Attribut_1, rB, FOREIGN KEY rB_FK (rB) REFERENCES B(ID))	Le lien entre deux tables est établi avec une clé étrangère. Cette solution devient malaisée lorsque l'association d'une extension contient des attributs. ESRI GDB: non transposable.

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
			rA -- {1..*} A; rB -- {1} B; END C;	CREATE TABLE B (ID PK, Attribut_2)	
5.a	Associations	Avec table intermédiaire: pour des relations n:m, sans contraintes de BD (DB-Constraints)	CLASS A = Attribut_1; END A; CLASS B = Attribut_2; END B; ASSOCIATION C = rA -- {1..*} A; rB -- {1..*} B; END C;	CREATE TABLE A (ID PK, Attribut_1,) CREATE TABLE B (ID PK, Attribut_2) CREATE TABLE C (rA, rB)	Le lien entre deux tables est établi avec une table intermédiaire.
5.b	Associations	Avec table intermédiaire: pour des relations n:m, avec contraintes de BD (DB-Constraints)	CLASS A = Attribut_1; END A; CLASS B = Attribut_2; END B; ASSOCIATION C = rA -- {1..*} A; rB -- {1..*} B; END C;	CREATE TABLE A (ID PK, Attribut_1,) CREATE TABLE B (ID PK, Attribut_2) CREATE TABLE C (rA, rB, FOREIGN KEY rA_FK (rA) REFERENCES A(ID), FOREIGN KEY rB_FK (rB) REFERENCES B(ID))	Le lien entre deux tables est établi avec une table intermédiaire. ESRI GDB: non transposable.

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
6	Associations	Avec suppression en cascade	CLASS A = Attribut_1; END A; CLASS B = Attribut_2; END B; ASSOCIATION C = A -- {1..*} A; B -<#> {1} B; END C;	CREATE TABLE A (ID PK, Attribut_1, Attribut_3, CONSTRAINT Attribut_3_FK FOREIGN KEY (Attribut3) REFERENCES B(ID) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE CASCADE,) CREATE TABLE B (ID PK, Attribut_2)	• Banque de données (base SQL): les restrictions requises sont à définir comme des contraintes (constraints). Elles peuvent être lourdes lors de l'importation, l'ordre de succession des objets / enregistrements étant important. Il est ainsi impossible d'importer des «enfants» lorsque les «parents» n'existent pas. • ESRI GDB: • ESRI GDB: les possibilités sont différentes suivant la licence détenue. Deux types de classes de relations (RelationshipClasses) sont possibles: 1. Simple: elles ne comportent aucune restriction, le lien est établi pour la visualisation dans ArcMap. Composite: restrictions en «cascade» lorsqu'un objet est supprimé.
7.a	Associations	Mettre à plat le modèle récursif dans la BD	CLASS A = Attribut_1; END A; ASSOCIATION C = rA1 -- {1..*} A; rA2 -- {1} A; END C;	CREATE TABLE A (ID PK, k0_Attribut_1, k1_ID, k1_rA2, k1_Attribut_1, k2_ID, k2_rA2, k2_Attribut_1, k3_ID, k3_rA2,	En règle générale, une structure récursive est modélisée pour prendre en charge une structure d'objet infinie. Si l'on veut mettre à plat une telle structure dans une table, il faut se limiter à un nombre d'objets fini: une table possède un nombre de colonnes fixe. Dans l'exemple, 5 objets directs ou indirects de A peuvent être mémorisés au plus avec un objet racine. (Correspond à la transposition #10 appliquée à une association récursive). Les valeurs d'attributs existent de manière

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
				k3_Attribut_1, k4_ID, k4_rA2, k4_Attribut_1, k5_ID, k5_rA2, k5_Attribut_1)	redondante. Lors de la saisie, l'utilisateur / l'application doit garantir que les données disponibles à plusieurs reprises présentent des contenus identiques pour des objets identiques.
7.b	Associations	Mettre à plat le modèle récursif dans la BD via JSON (ou XML) dans la table principale	CLASS A = Attribut_1; END A; ASSOCIATION C = rA1 -- {1..*} A; rA2 -- {1} A; END C;	CREATE TABLE A (ID PK, k0_Attribut_1, kn_Objekte CLOB)	Au lieu de stocker des enregistrements séparés (un par objet avec les valeurs associées), le réseau atteignable depuis un objet racine peut être sérialisé dans un format adapté et stocké (valeur de colonne) avec l'objet racine. (Correspond à la transposition #2.c appliquée à un réseau d'objets récursif). Les requêtes avec des critères portant sur les objets du réseau deviennent alors compliquées. Les valeurs d'attributs existent de manière redondante. Lors de la saisie, l'utilisateur / l'application doit garantir que les données disponibles à plusieurs reprises présentent des contenus identiques pour des objets identiques.
8	Associations	Mettre à plat les classes ensemble, dans une table	CLASS A = Attribut_1; END A; CLASS B = Attribut_2; END B; ASSOCIATION C =	CREATE TABLE A (ID_CLASSA, ID_CLASSB, Attribut_1, Attribut_2)	Avantage: les exploitations ne requièrent pas de jointure de BD (DB-Join). Inconvénients: Les valeurs d'attributs existent de manière redondante. Lors de la saisie, l'utilisateur / l'application doit garantir que les données disponibles à

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
			rA -- {1..*} A; rB -- {1..*} B; END C;		plusieurs reprises présentent des contenus identiques pour des objets identiques.
9	Conteneurs / baskets	Plusieurs conteneurs du même thème par banque de données	TOPIC A = CLASS AA = Attribut_1; END AA; END A ;	CREATE TABLE A (ID PK, Attribut_1, BasketID FOREIGN KEY BasketID _FK (BasketID) REFERENCES Basket(ID),) CREATE TABLE Basket (ID PK)	Il est possible d'avoir plusieurs conteneurs par thème dans le fichier de transfert. Un champ supplémentaire est requis si l'on veut savoir quelles données appartiennent à quel conteneur INTERLIS.

3.2 Noms de classes, noms d'attributs

PK = primary key (clé primaire)
FK = foreign key (clé étrangère)

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
10	Noms	Renommer des tables ou des colonnes à cause de conflits avec les règles de BD ou de prescriptions impératives touchant les: 1. mots-clés 2. longueurs 3. majuscules et minuscules	1. Table «Where» 2. ESRI GDB: longueur maximale d'un nom: 64 caractères 3. Nom d'attribut KW_Ueberschwemmung	1. Table «Where_1» 2. S'il est trop long, le nom doit être coupé. 3. Nom d'attribut kw_ueberschwemmung	1. Mots-clés qui ne sont pas acceptés dans les banques de données (par exemple Value) 2. Longueurs de noms d'attributs qui ne sont pas acceptées dans les banques de données. 3. Les majuscules ne peuvent pas être acceptées.
11	Thèmes / TOPICs	Plusieurs thèmes par modèle / banque de données	TOPIC A = CLASS AA = Attribut_1; END AA; END A ; TOPIC B = CLASS AA = Attribut_2; END AA; END B ;	CREATE TABLE A_AA (ID PK, Attribut_1) CREATE TABLE B_AA (ID PK, Attribut_2)	Parce qu'il n'existe pas de thèmes dans la BD, les noms des tables peuvent éventuellement être complétés par le nom du thème pour éviter toute ambiguïté. Plusieurs conteneurs peuvent coexister dans le fichier de transfert (un conteneur par thème). Si l'on veut savoir quelles données appartiennent à quel thème, il convient de donner des noms appropriés aux tables.

3.3 OID, Primary Key

PK = primary key (clé primaire)
FK = foreign key (clé étrangère)

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
12	OID	OID/TID utilisé comme clé interne	CLASS A = Attribut_1; END A;	CREATE TABLE A (ID PK, Attribut_1)	Si un TID est sans équivoque dans un fichier de transfert (.xtf), il ne l'est généralement pas lorsque plusieurs fichiers XTF sont importés. Une renumérotation doit donc être entreprise. Un OID est globalement univoque. Plusieurs fichiers XTF peuvent être importés sans qu'une renumérotation soit requise. Inconvénient: lorsque le système de numérotation externe change, les références internes doivent être adaptées.
13	OID	OID/TID non utilisé comme clé interne	CLASS A = Attribut_1; END A;	CREATE TABLE A (ID PK, xtf_id, Attribut_1)	Lors de l'importation, un nouveau numéro doit être attribué à tous les objets, y compris ceux ayant un OID. Il n'y a pas non plus de conflits d'ID pour les TID, puisqu'ils ne sont pas utilisés comme PK en interne. Avantage: si le système de numérotation externe change, les références internes n'ont pas besoin d'être adaptées.
14.a	Renvoi vers des objets non connus (EXTERNAL)	Avec valeur de référence de remplacement	CLASS A = Attribut_1; END A; CLASS B = Attribut_2; END B; ASSOCIATION C =	CREATE TABLE A (ID PK, Attribut_1, rB, FOREIGN KEY rB_FK (rB) REFERENCES B(ID) rBext,	rBext est utilisé à la place des TID / OID de l'objet inconnu (qui n'existe pas dans la BD). Par exemple dans des catalogues externes.

			rA -- {1..*} A; rB (EXTERNAL) -- {1} B; END C;	) CREATE TABLE B (ID PK, Attribut_2)	
14.b	Renvoi vers des objets non connus (EXTERNAL)	Avec objet proxy	CLASS A = Attribut 1; END A; CLASS B = Attribut 2; END B; ASSOCIATION C = rA -- {1..*} A; rB (EXTERNAL) -- {1} B; END C;	CREATE TABLE A (ID PK, Attribut 1, rB, FOREIGN KEY rB_FK (rB) REFERENCES B(ID)) CREATE TABLE B (ID PK, IST_UNBEKANNT Attribut 2)	Les objets inconnus sont inscrits dans la table, mais sont signalés comme étant inconnus (drapeau IST_UNBEKANNT). Les clés étrangères peuvent donc renvoyer vers un enregistrement, comme c'est normalement le cas.

3.4 Types de données de base

PK = primary key (clé primaire)
FK = foreign key (clé étrangère)

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
15.a	Enumérations	En tant que nom	DOMAIN D = (D1, D2, D3) CLASS A = Attribut_1: D; END A;	CREATE TABLE D (ID VARCHAR(100) PK, Description) CREATE TABLE A (ID PK, Attribut_1, CONSTRAINT Attribut_1_FK FOREIGN KEY (Attribut1) REFERENCES D(ID) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,)	• Banque de données (base SQL): tables avec contraintes (Constraints) • ESRI GDB: CodedValueDomain Reprendre les noms des éléments énumérés comme ID.
15.b	Enumérations	En tant que code	DOMAIN D = (D1, D2, D3) CLASS A = Attribut_1: D; END A;	CREATE TABLE D (ID BIGINT PK, NAME VARCHAR(100) PK, Description) CREATE TABLE A (ID PK, Attribut_1, CONSTRAINT Attribut_1_FK FOREIGN KEY (Attribut1) REFERENCES D(ID) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,)	• Banque de données (base SQL): tables avec contraintes (Constraints) • ESRI GDB: CodedValueDomain La liste des codes peut revêtir une importance propre au domaine de spécialité concerné. Point important dont il faut tenir compte: la liste doit être générée comme des valeurs Integer, par exemple «1, 2, 3, 4, …» ou «11, 12, 15, 16, …».
16	Types de données INTERLIS qui	En tant que TEXT, CLOB ou BLOB	CLASS A = Attribut_1 : HALIGNMENT;	CREATE TABLE A (ID PK,	Ces types de données ne peuvent pas être utilisés directement.

	n'existent pas dans la BD		Attribut_2 : BLACKBOX BINARY; Attribut_3: BLACKBOX XML; END A;	Attribut_1 VARCHAR(20), Attribut_2 BLOB, Attribut_3 CLOB)	La table d'aide (annexe 4.1) répertorie les différents types d'attributs (nombres et texte) acceptés dans PostGreSQL et ESRI GDB.
17.a	Plus d'une géométrie par table	Reprendre la structure telle quelle (1:1) dans la BD	CLASS A = Attribut_1; Geom_1: Line; Geom_2: Surface; END A;	CREATE TABLE A (ID PK, Attribut_1, Geom_1 geometry(Linestring), Geom_2 geometry(Polygon))	Tout dépend de la technologie employée, par exemple: • PostGIS: cette option est possible, mais ne peut pas être utilisée avec les logiciels de SIG courants. • ESRI GDB: elle n'est pas acceptée Il faut aussi tenir compte des possibilités éventuellement limitées de l'application.
17.b	Plus d'une géométrie par table	Une table spécifique pour chacun des attributs géométriques supplémentaires	CLASS A = Attribut_1; Geom_1: Line; Geom_2: Surface; Attribut_2; END A;	CREATE TABLE A (ID PK, Attribut_1, Geom_1 geometry(Linestring), Attribut_2,) CREATE TABLE A_geom_2 (ID PK, Geom_2 geometry(Polygon))	
17.c	Plus d'une géométrie par table	Dissocier la table en présence d'attributs géométriques supplémentaires	CLASS A = Attribut_1; Geom_1: Line; Geom_2: Surface; Attribut_2; END A;	CREATE TABLE A (ID PK, Attribut_1, Geom_1 geometry(Linestring),) CREATE TABLE A_geom_2 (ID PK, Geom_2 geometry(Polygon) Attribut_2,)	
17.d	Plus d'une géométrie par table	Un enregistrement redondant par attribut géomé-	CLASS A = Attribut_1;	CREATE TABLE A (ID PK,	Deux enregistrements sont créés pour un même objet avec les mêmes valeurs pour

		trique, pour lequel seule la valeur géométrique est différente	Geom_1: Surface; Geom_2: Line; Attribut_2; END A;	xtf_id, Attribut_1, Geom, Attribut_2,)	xtf_id, Attribut 1 et Attribut 2. La seule différence concerne la géométrie: le premier enregistrement contient la valeur Geom_1 et le second la valeur Geom_2. Inconvénient: le type de la colonne Geom doit pouvoir contenir des genres différents: points et/ou lignes et/ou surfaces. Inconvénient: l'application ou l'utilisateur doit garantir que les valeurs redondantes pour xtf_id, Attribut 1 et Attribut 2 sont bien identiques dans les différents enregistrements du même objet.
18	Arcs de cercle	Ils ne sont pas acceptés dans la BD			De nombreuses BD acceptent les arcs de cercle. S'ils ne le sont pas (par la BD ou par l'application SIG), la géométrie doit être modifiée. Pour éviter des pertes de données dans un tel cas, la géométrie doit faire l'objet d'un stockage (et d'une mise à jour) supplémentaire dans une structure de données propre (un BLOB par exemple).
19	Recouvrements	Conversion en une géométrie conforme à l'ISO/OGC (élément simple ou simple feature)	A EXTENDS Surface = SURFACE WITHOUT OVERLAPS > 0.002;		La géométrie doit être modifiée parce qu'elle est incompatible avec les normes internationales. Pour éviter des pertes de données, la géométrie doit faire l'objet d'un stockage (et d'une mise à jour) supplémentaire dans une structure de données propre (un BLOB par exemple).
20	Recouvrements	Condition: deux surfaces ne peuvent pas se recouvrir de plus de 0,002 mètre	A EXTENDS Surface = SURFACE WITHOUT OVERLAPS > 0.002;		Condition difficile à intégrer: • Banque de données (base SQL): avec déclencheur (Trigger) • ESRI GDB: contrôle seul

					Le respect de telles restrictions peut aussi être assuré au stade de l'importation.
21	Restriction	Un attribut d'une classe est dépendant d'un autre attribut	CLASS A = Attribut_1; Attribut_2 ; MANDATORY CONSTRAINT Attribut_1 <= Attribut_2; END A;		Condition difficile à intégrer: • Banque de données (base SQL): avec déclencheur (Trigger) • ESRI GDB: contrôle seul Le respect de telles restrictions pourrait aussi être assuré au stade de l'importation.
22	Précision	INTERLIS indique la précision des coordonnées via le nombre de chiffres figurant après la virgule (soit 1 mm dans le cas présent)	COORD 480000.000 .. 840000.000 [m], 7000.000 .. 300000.000 [m], 200.000 .. 5000.000 [m];	• ESRI GDB: Résolution : 1mm Tolérance (recommandation ESRI): Au moins le double de la résolution, soit 2mm	• PostGreSQL: les coordonnées sont stockées à leur pleine résolution (avec tous les chiffres après la virgule). • ESRI GDB: les concepts de tolérance et de résolution sont connus. Les coordonnées sont donc «alignées» sur une grille virtuelle avec une certaine distance entre les mailles. La résolution ne doit pas être définie de manière trop grossière.
23	Type de données OID	Un type d'attribut est défini comme OID UUID avec un domaine de valeurs	CLASS A = Attribut_1 : MANDATORY INTERLIS.UUIDOID; END A;		• PostGreSQL: GUID • ESRI GDB: GUID • Oracle: RAW(16)

3.5 CHBase

PK = primary key (clé primaire)
FK = foreign key (clé étrangère)

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
24	chbase: LocalisedText	Un type d'attribut est défini comme LocalisedText.	STRUCTURE LocalisedText = Language: LanguageCode_ISO639_1; Text: MANDATORY TEXT; END LocalisedText; CLASS A = Attribut_1 : ….LocalisedText; END A;	CREATE TABLE A (ID PK, Attribut_1, Attribut_1_Language)	Une seule langue étant acceptée, un attribut peut être inséré dans la table principale.
25.a	chbase:MultilingualText	Attribut défini comme une table spécifique	STRUCTURE LocalisedText = Language: LanguageCode_ISO639_1; Text: MANDATORY TEXT; END LocalisedText; STRUCTURE MultilingualText = LocalisedText : BAG {1..*} OF LocalisedText; UNIQUE (LOCAL) LocalisedText:Language; END MultilingualText; CLASS A = Attribut 1 : LocalisationCH_V1. MultilingualText; END A;	CREATE TABLE A_Attribut1 (ID PK, A_Attribut1, Language, Text CONSTRAINT A_Attribut1_FK FOREIGN KEY (A_Attribut1) REFERENCES A(ID) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,) CREATE TABLE A (ID PK)	La structure MultilingualText est modélisée comme une table contenant une clé étrangère pour établir le lien avec la table principale A.

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
25.b	chbase:MultilingualText	Insérer dans la table principale	STRUCTURE LocalisedText = Language: LanguageCode_ISO639_1; Text: MANDATORY TEXT; END LocalisedText; STRUCTURE MultilingualText = LocalisedText : BAG {1..*} OF LocalisedText; UNIQUE (LOCAL) LocalisedText:Language; END MultilingualText; CLASS A = Attribut 1 : LocalisationCH_V1. MultilingualText; END A;	CREATE TABLE A (ID PK, Attribut1_DE, Attribut1_FR, Attribut1_IT)	Cette solution est envisageable, mais peut se révéler déplaisante parce que la table principale contient de nombreux attributs.
26	chbase: CatalogueReference	Un type d'attribut est défini comme CatalogueReference.	CLASS X EXTENDS …Catalogues.Item = Code : MANDATORY TEXT; Description : MANDATORY TEXT; END X; STRUCTURE Y EXTENDS ...Catalogues.CatalogueReference = Reference (EXTENDED) : REFERENCE TO X; END Y; CLASS A = Attribut_1 : Y;	CREATE TABLE X (ID PK, Code, Description) CREATE TABLE A (ID PK, Attribut_1 CONSTRAINTAttribut_1_FK FOREIGN KEY (Attribut_1) REFERENCES X(Code) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,)	• Banque de données (base SQL): la table X contient toutes les valeurs possibles du catalogue. Le lien avec la table principale est alors établi à l'aide d'une clé étrangère. • ESRI GDB: CodedValueDomains

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
			END A;		
27.a	chbase: Item		CLASS X EXTENDS …Catalogues.Item = Code : MANDATORY TEXT; Description : MANDATORY TEXT; END X;	CREATE TABLE Item (ID PK, Class, // z.B. ‚X' Code, Description)	Une table de codes pour toutes les classes qui étendent Item. Facilite/permet l'utilisation éventuelle de la bibliothèque de logiciels existants.
27.b	chbase: mettre à plat Item	Un type d'attribut est défini comme CatalogueReference.	CLASS X EXTENDS …Catalogues.Item = Code : MANDATORY TEXT; Description : MANDATORY TEXT; END X; STRUCTURE Y EXTENDS ...Catalogues.CatalogueReference = Reference (EXTENDED) : REFERENCE TO X; END Y; CLASS A = Attribut_1 : Y; END A;	CREATE TABLE A (ID PK, Attribut_1_Code, Attribut_1_Description)	Forte simplification pour les exploitations, modification de la table de codes relativement malaisée en revanche.
28	chbase: MultLine	Un type géométrique est défini comme MultiLine.	Line = POLYLINE WITH (STRAIGHTS, ARCS) VERTEX Coord2; STRUCTURE LineStructure = Line: Line; END LineStructure; STRUCTURE MultiLine = Lines: BAG {1..*} OF LineStructure;	CREATE TABLE X (ID PK, Geo_Obj geometry MultiLinestring,)	Dépend de la technologie : • PostGIS: MultiPart accepté • ESRI GDB: MulitPart accepté

Numéro	Cas de figure (Use case)	Description	Exemple	Simplification (exemple SQL)	Commentaires
			END MultiLine; CLASS X = Geo_Obj : MANDATORY MultiLine; END X;		
29	chbase: MultiSurface	Un type géométrique est défini comme MultiSurface.	Surface = SURFACE WITH (STRAIGHTS, ARCS) VERTEX Coord2; STRUCTURE SurfaceStructure = Surface: Surface; END SurfaceStructure; STRUCTURE MultiSurface = Surfaces: BAG {1..*} OF SurfaceStructure; END MultiSurface; CLASS X = Geo_Obj : MANDATORY MultiSurface; END X;	CREATE TABLE X (ID PK, Geo_Obj geometry MultiPolygon,)	Dépend de la technologie: • PostGIS: MultiPart accepté • ESRI GDB: MultiPart accepté

4 Annexe

4.1 Types de données «Nombres» et «Texte»: INTERLIS/PostGreSQL/ESRI GDB

	INTERLIS	PostGreSQL		ESRI GDB	
Nombres	Domaines de valeurs: par exemple 0 .. 130	Smallint	De -32768 à +32767	Short integer	De -32,768 à 32,767
		Integer	De -2147483648 à +2147483647	Long integer	De -2,147,483,648 à 2,147,483,647
		Bigint	De -9223372036854775808 à 9223372036854775807		
	Domaines de valeurs: par exemple 0.0 .. 130.0	decimal	no limit	Single-precision floating-point number (float)	De -3.4E38 à 1.2E38 env.
		numeric	no limit	Double-precision floating-point number (double)	De -2.2E308 à 1.8E308 env.
		real	6 decimal digits precision		
		double precision	15 decimal digits precision		
Texte	Avec définition de longueur: TEXT*30	character varying(n), var-char(n)	variable-length with limit	Texte(longueur)	
		character(n), char(n)	fixed-length, blank padded		
		text	variable unlimited length		