Towards a Data Model for Update Propagation in MR-DLM

Sheng Zhou[1], Nicolas Regnauld[1] and Carsten Roensdorf[2]
Ordnance Survey Research[1]
Ordnance Survey GeoData Management Group[2]
Romsey Road, SOUTHAMPTON
SO16 4GU, United Kingdom

Abstract: In this paper we describe a data model for recording map generalisation process information to facilitate update propagation in a multi-resolution map database environment. The process flow in a generalisation session is modelled as a DAG of prioritised generalisation process instances. Inside each instance, data manipulation may be further divided into independent operations as the atomic functional unit for generalisation. Generalisation parameters as well as information on features that act as the source, the target or the context of operations are stored. The full generalisation history on each feature may be traced back and used in subsequent update operations.

Keywords: Multi-resolution; Digital landscape model; Update propagation; Generalisation

1. Introduction:

At Ordnance Survey, we have in the past years concentrated our efforts on creating a seamless database that stores our most detailed data (base data) centrally. The idea is that only this database is maintained, and all the products are derived from it, as automatically as possible. We are at a transition time, where our production lines do not use our base data efficiently. Different map products are often built using specific software relying of specific hardware, often obsolete. They usually exploit their own data, collected specially for a particular product.

Ordnance Survey Research, in collaboration with the GeoData Management (GDM) team and the cartographic production department, are working on bridging the gap existing between our base data (which currently mainly supports Ordnance Survey's flagship product OS MasterMap (Ordnance Survey 2008)), and other products. The aim is to bring flexibility, consistency and efficiency to our production systems.

We have therefore started a research project to design and build a multi-resolution database, to connect base data and products. The data model will be an extension of the model used for the base data. It will describe geographic concepts and their representations at different resolutions (levels of detail). The model can be enriched by adding more concepts (for example high level entities created from existing ones using part-of relationships), or by adding more abstract representations of an existing concept. One design goal for the model is that each feature in a product can then be traced back to its original features in the base data. Following the links in the opposite direction will find for each feature in the base data its representation in the different products. This approach is expected to bring opportunities to:

- Increase production efficiency by propagating updates in the main database to the relevant products.

- Flexibility to derive new products, by using the Multi-resolution database as a component library.
- Efficiency, by making all the components derived for a particular product available for others.

Generalisation operations rely heavily on spatial structures (Regnauld 2005). Important database enrichment is usually required before contextual generalisation can be applied (e.g. (Chaudhry and Mackaness 2005)), and some of those can certainly be reused.

In addition, the description of the geographic concepts can later be extended to include semantic information. This will allow us to research in a future project the derivation of semantic information for generalised products. Adding semantic information to our base data is already the object of active research at Ordnance Survey (Goodwin 2005) to increase the interoperability of our data with the outside world.

This paper discusses the modelling of this multi-resolution database, and in particular focuses on the way geographic concepts and their different representations are explicitly linked. A conceptual data model for generalisation process is presented section 2 and 3. In section 4 we present a data model for linking features and recording generalisation information at feature level. An introduction on a relational realisation of this model is given in section 5, illustrated by some simple examples. How this model can be used to facilitate update propagation is discussed in brief in section 6.

2 DLM and MR-DLM

A DLM (Digital Landscape Model) (Grünreich 1985) is a data model for representing geospatial phenomena from a certain point of view at a given scale/resolution. From a DLM, we may derive DCM (Digital Cartographic Model) by applying cartographic criteria.

2.1 From DLM to MR-DLM

To represent phenomena at different resolutions and/or from different viewpoints, different DLMs have to be created to provide appropriate levels of detail. Alternatively, we may design a multi-resolution (or more generic, multi-representation) DLM (MR-DLM) to integrate multiple DLMs into a single model.

The main difference between a MR-DLM and a simple collection of several DLMs for various resolutions is that in a MR-DLM, multiple representations of the same geospatial phenomenon at various resolutions are explicitly linked or even integrated. Consequently, cross-resolution representation and analysis can be carried out in a consistent and efficient manner.

A MR-DLM provides the basis for a multi-resolution spatial database (MR-SDB) to accommodate multiple representations of real world phenomena.

Numerous researches have been carried out on the topics of MR-DLM and MR-SDB. For example, the hierarchical structure in a map series was explored at the conceptual level in (Timpf 1998). A formal model for multi-resolution map features was described in (Puppo and Dettori 1995). "Stratified map spaces" are proposed in (Stell and Worboys 1998) as a formal basis for multi-resolution spatial databases. Intra-resolution, inter-resolution and update relations in a MR-SDB were modelled in an explicit manner in (Bobzien et al. 2006). At physical level, in addition to the conventional approach of storing explicit multiple versions, there are also some attempts to integrate multiple geometry at different scales into a single multi-scale geometry in database (Becker et al. 1991, Zhou and Jones 2001), or even integrate multiple geometry based on scale as well as other semantic criteria into a multi-representation geometry (Zhou and Jones 2003). Comprehensive reviews on these topics may be found in (Balley et al. 2004, Sarjakoski 2007).

2.2 Populating and updating a MR-DLM based MR-SDB

In general there are two approaches (Anders and Bobrich 2004) (which are often combined) for the initial population of a MR-DLM based MR-SDB:

- Manually or automatically matching and linking features at different resolutions from existing data
- Generalising large scale dataset to derive multiple representations (Figure 1)

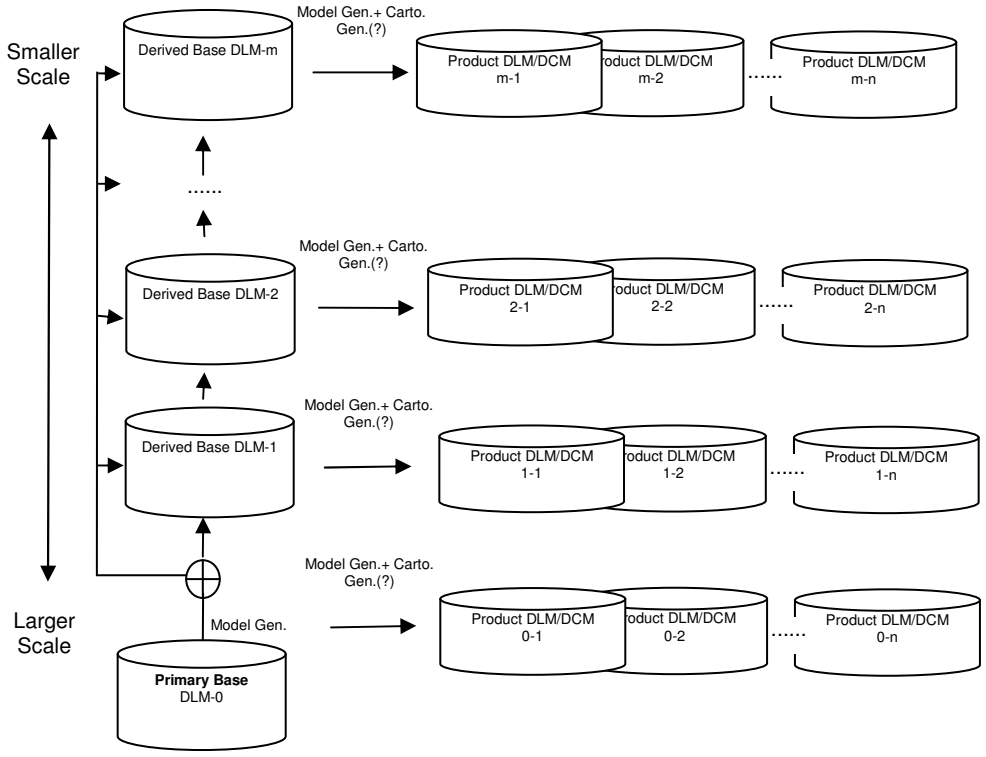


Figure 1: Populating a MR-SDB by generalisation (models are shown)

The matching and linking approach establishes relations between features (or multiple representations of the same feature) at different resolutions. However, it does not

provide means to transform features (or representations) at one resolution to those at the other resolution, which is the main strength of the generalisation approach.

A MR-SDB is inherently dynamic. It has to be updated frequently to reflect changes in the real world. Such changes will normally be surveyed at only one resolution (normally but not necessarily the finest) and has to be propagated to other resolutions (via manual or automatic generalisation). To a great extent, effective and efficient update propagation is the key to the successful application of a MR-SDB.

A solution that will guarantee overall quality and consistency is to re-run the whole process that initially builds the MR-SDB to reflect any changes. However, such an approach will be computationally intensive and inefficient, and hence not practical under most circumstances. Therefore, we need to find some way to decide the scope of the impact caused by the changes and then update the dataset locally whenever applicable. Such a local incremental update approach was introduced in (Kilpeläinen and Sarjakoski 1995) and was further elaborated in subsequent researches (e.g. (Haunert and Sester 2005, Skogan and Skagestein 2005)).

To facilitate such a strategy, we will first develop a model for automatic generalisation process. Using this model, we will design a process-centred data model for linking source and target features and logging relevant generalisation information to utilise efficient local update. In principle, this is an approach similar to the "production log" in (Skogan and Skagestein 2005).

As already mentioned, updates could occur at coarser resolutions (e.g. surveyed or third-party data at coarser resolution) and sometimes will need to be propagated to data at finer resolutions for integration purpose. In addition, derivation may take place at the same resolution, for example, deriving new structures (such as network) from surveyed data or generating a simpler representation for different purposes. In principle, such derivation may also be described using the model introduced in this paper.

3. Modelling the generalisation process

The process of automatic generalisation may be viewed as a two-dimensional (data-process) session which consists of a set of transactions in sequence to map a set of source features to a set of target features. From this point of view, we will construct a conceptual model that focuses on the spatial context of operations on each feature during generalisation. It does not attempt to describe the structural or functional details of any concrete generalisation algorithms.

3.1 The data dimension in generalisation

3.1.1 Features:

The basic data unit for generalisation is **map feature** which has a spatial depiction as well as other attributions.

To simplify our discussion, we assume **all features are single-resolution**. Therefore, any generalised feature is treated as a new feature. If the generalised feature and the

original feature represent the same real world phenomenon (possibly under different classification), we say the (real-world) **identity** is unchanged.

A map feature belongs to a **feature class**. Note that under the OO paradigm, a feature can be viewed as an instance of multiple feature classes (with one or more of them as its direct class or, in case of multiple inheritance, direct classes). A map dataset consists of features from one or more feature classes.

A set of features may be partitioned on the basis of feature classes (i.e. all features belonging to one or more classes), or on a spatial basis (e.g. all features inside a given region), or on both.

3.1.2 Mapping between source and target features

A generalisation process maps a set of source features to a set of target features (which may be nothing, i.e. empty set).

In terms of the cardinality of the mappings, they are generally M:N mapping which maps M source features into N target features. Normally, we have $M \geq N \geq 1$ where $N = 0$ represents deletions. Also, we should not completely rule out the possibility of $M < N$ (e.g. possibly some types of symbolisation or enhancement).

A special case is one to one (1:1) mapping which maps a source feature to a target feature, independent of other source features, i.e. it has no (spatial or aspatial) contexts.

Below are some examples of 1:1 generalisation:

- Ramer-Douglas-Peucker (RDP) line simplification (Ramer 1972, Douglas and Peucker 1973) (without topological consistency consideration)
- Object selection and re-classification based only on features' own attribution values (e.g. "select all residential sites with a population more than 100")

It is straightforward to represent 1:1 mapping. Unfortunately, many if not most generalisation operations are beyond simple 1:1 mapping.

Conceptually speaking, the source and target feature classes are often the same. However, as already mentioned, it improves clarity of discussion if we treat source and target feature classes as different classes (even if they are conceptually the same). This also applies to the discussion below on many to many mappings.

From the viewpoint of mapping at feature class level, there are several cases for M:N feature mapping:

- All M source features from a single feature class and all N target features from a single feature class (i.e. an 1:1 mapping at feature class level)
- All M source features from multiple feature classes and all N target features from a single feature class (i.e. a m:1 mapping at feature class level)
- All M features from multiple feature classes and all N target features from multiple features classes (i.e. a m:n mapping at feature class level)

3.2 The Process dimension in generalisation

In this section we will present a model for describing the generalisation process. This model uses a component hierarchy of **session**, **process** and **operation**.

3.2.1 Session

A generalisation session maps a source feature dataset to a target feature dataset. To an extent, a session is the analogue to an editing session between two check points in a database environment.

A session contains a set of generalisation processes which perform the actual generalisation tasks. These processes partition primarily on the process dimension (and potentially with sub-division on the data dimension) in the data-process space.

3.2.2 Process

A process encapsulates manipulations on a subset of the source dataset. Ideally a process will operate on **all** features of one or more feature classes in the source dataset.

A process normally represents the application of a clearly defined generalisation algorithm (e.g. RDP), or a set of coordinating algorithms to a set of features. Any loops in the control flow should also be encapsulated into a single process. The same process type (i.e. algorithm) may be applied to different sets of features and results in several process instances (for example, an RDP process instance for coastline simplification and another instance for contour).

A process may be functionally dependent on other processes, i.e. it must be executed **after** the execution of certain other processes. A **precedence** property will be defined for each process to specify the processes which **immediately** precede it.

Using the precedence property, a DAG (Direct Acyclic Graph) may be constructed for processes in a session with each process represented by a node in the graph (Figure 2).

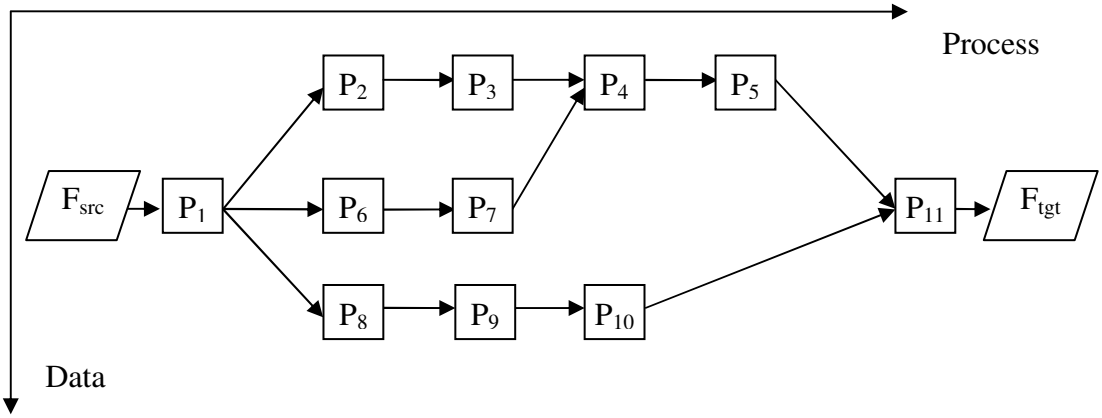


Figure 2: A Generalisation Session viewed as a DAG of processes

3.2.3 Operation

Operations further partition a process on the data dimension (figure 3). A process contains one or more operations. Each operation operates on a sub-set of features and all these sub-sets should form a partition (in a mathematical sense) of the feature set associated with the process.

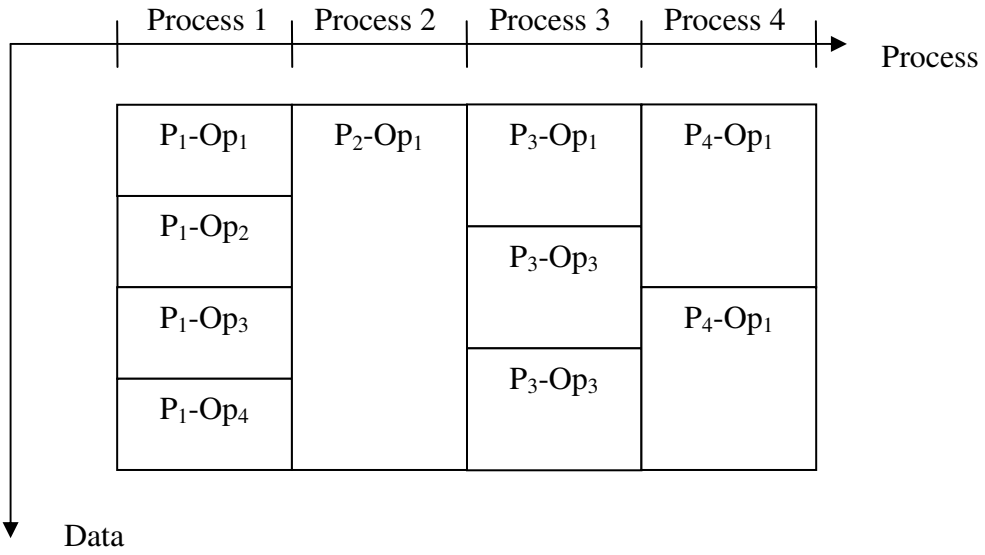


Figure 3: partitioning a session into processes and operations

For example, a RDP process applies to all contours in the contour feature class. This process contains multiple operations each of which applies to an individual contour. An operation is the basic functional unit that can't be further decomposed structurally.

It is possible that a process instance contains just a single operation so that functionally the process instance and the operation are identical.

How a process is decomposed into operations is related to the role of features in generalisation operation. In an operation, a map feature can play the role of a **participant** (i.e. being manipulated: modified, deleted, generated, etc.), or function as a **context** to other participants (e.g. setting out the topological boundary, measured against to generate some indexes). A contextual feature is not manipulated during the operation.

When a feature is changed, if it acts as a participant in an operation, any features as the contexts of this operation will not be affected by the change; however, if it acts as a context in an operation on other features, other features will be affected by this change.

At present, our design is that operations inside a process must be mutually independent, i.e. they may be executed in any order. Consequently, a feature in a process should not be a context in any operations in this process. This restriction may be lifted in the future if it is necessary to introduce precedence among operations inside a process instance.

3.2.4 Rules to decide the precedence

Having introduced the concepts of process and operation, we can now summarise the rules to decide the precedence between two processes or operations.

- Precedence is transitive (i.e. A precedes B and B precedes C then A precedes C);
- Processes with a "null" precedence property have the (equally) highest precedence;
- Multiple immediate preceding processes of a process have equal precedence
- Operations in the same process have the same precedence (if a proposed operation depends on the completion of another operation, the two should be merged into a single operation, or if appropriate, the process should be split into two)
- Precedence between two operations of two different processes is decided by the precedence of their parent processes.

The overall sequence of execution for all these processes may be decided by a "topological sort" based on the rules above. Note that the result of a topological sort may not be unique. For example, for the arbitrary session illustrated in figure 2 which contains 11 processes (labelled as P_1 to P_{11}), there are four possible execution sequences (highest priority on the left):

```
1 -> 2 -> 3 -> 6 -> 7 -> 4 -> 5 -> 8 -> 9 -> 10 -> 11
1 -> 6 -> 7 -> 2 -> 3 -> 4 -> 5 -> 8 -> 9 -> 10 -> 11
1 -> 8 -> 9 -> 10 -> 2 -> 3 -> 6 -> 7 -> 4 -> 5 -> 11
1 -> 8 -> 9 -> 10 -> 6 -> 7 -> 2 -> 3 -> 4 -> 5 -> 11
```

Nevertheless, the final output of the session following any of these sequences will be identical.

4 A data model for generalisation logging:

Based on the process-centred generalisation model described in section 3, we will now provide a conceptual design for the main components required to support **generalisation logging**, i.e. to record important meta-data for generalisation so that the generalisation process may be re-constructed at a later time.

A common component used in most other components is "selection-set", i.e. a set of one or more features of the same or several different feature classes. There are numerous ways to represent a selection-set, e.g. explicit list of feature IDs, database query statement string or implicitly represented in database schema. As this is more an issue for physical design and implementation, we will use some general and interchangeable terms (dataset, feature set, etc.) in the discussion below to refer to a selection of features.

4.1 Elements in the model

The elements in this data model are illustrated in the UML class diagram in figure 4. Here we give a brief description on these elements:

- A **GenFeatureClass** instance represent map feature tables in a generalisation database.
- A **GenDataset** instance represents a defined subset of features in the database.
- A **GenView** instance represents a view (in a DBMS sense) created on a feature table (represented by GenFeatureClass objects. At present, we do not recommend the use of views created on multiple tables.
- A **GenViewRef** instance links a GenView instance to a GenDataset. All GenViewRef instances for a GenDataset define the content of the dataset.
- A **GenSession** instance represents a generalisation session. It manipulates a source GenDataset and stores results in a target GenDataset. If preferred by the applications, the source and target datasets may be the same GenDataset.
- A **GenProcess** instance represents a generalisation process in a session denoted by the SessionID attribute.
- A **GenOperation** instance represents a generalisation operation in a process denoted by the ProcessID attribute.
- A **GenPrecedence** instance stores the information of one preceding process of a process. If there are multiple preceding processes for a process (e.g. P_{11} in figure 2), there will be more than one GenPrecedence instances created for that process.
- A **GenProcPara** instance represents the name and value of a parameter for a process.
- A **GenOperPara** instance represents the name and value of a parameter for an operation. Note that a process and its operations should have the same set of parameter. At present, we suggest that GenProcPara instances (at process level) are used to provide universal or default values for operations where GenOperPara instances at operation level supply operation-specific values that may override values from GenProcPara instances, or provide values that are absent at process level.
- A **GenFCProcess** instance records a process applied on instances of a feature class represented by a GenFeatureClass instance. When a new feature of a certain feature class is added, information on all processes applied to features of this feature class may be retrieved via the GenFCProcess instances associated with this feature class.
- A **GenFeature** represents a map feature in a generic manner. The only requirement for a feature is that it is uniquely identifiable.
- A **GenOpRef** instance links a feature to an operation. This is the key component is the model and will be described in detail in the next session.

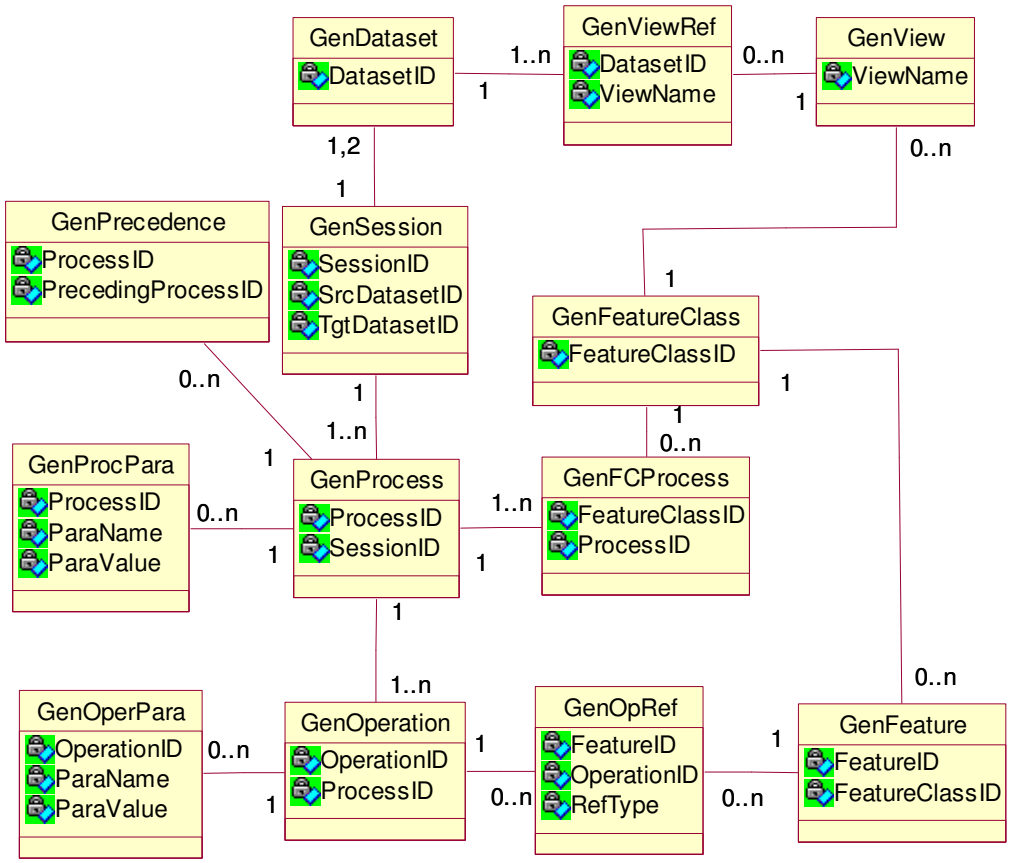


Figure 4: Generalisation logging – class diagram

4.2 Linking source and target, features and operations

A GenOpRef instance links a features and generalisation operation. GenOpRef has a RefType attribute which indicate the role of a feature in the operation. At present, we have defined three concrete role types: **S**ource, **T**arget and **C**ontext. If necessary, more role types may be created.

As mentioned previously, there is usually an M:N relation between the source and target features of an operation. The introduction of GenOpRef as an operation reference not only links features to operations, it also breaks the M:N relation between source and target features into two 1:M relations.

It is the process' responsibility to define and assign the context references. It is also worth noting that the features as contexts could be derived features that are not in the source dataset.

The generalisation history of a feature during a session may be re-constructed from the GenOpRef instances associated with this feature. From these GenOpRef instances, the GenOperation instances in which the feature plays a role can be tracked. As an operation inherits its parent process' precedence property, operations can be compared and topologically sorted to create an ordered list.

In section 5 we will give some simple examples of representing mappings between the source and target features in an operation.

4.3 Spatial scope for processes and operations

An attribute but that we intend to include is the application-specified **spatial scope** for processes and operations (this is not illustrated in figure 4). For example, the spatial scope of a process or operation could be defined as the minimum bounding rectangle (MBR) or the convex hull of all the participant features. Such an attribute will enable quick retrieve of processes and operations via spatial query.

4.4 Passing process controls from source to target

The transformation from a source dataset to a target dataset is seldom achieved in a single process. For many processes and operations in a multiple-process session, their source and target features are in intermediate states.

These intermediate states can be made persistent by physically separating the storage of an operation's source and target. Alternatively, if their identities are unchanged over an operation, features themselves may be used as vehicles to pass controls from one process or operation to the next without physically generating the (logically) new target features. An example is given in 5.2.1.

Obviously, derived features (e.g. building blocks derived from buildings) signals change of identity and they must have separate storage. Also, a non-persistent intermediate state exists inside a process/operation only, that is, an operation may create a temporary feature for certain purpose. This temporary feature can't be passed to the next process/operation unless it is made persistent and identifiable.

In practice, the two methods are likely to be used in combination. Some important intermediate states or auxiliary features may be stored in database to facilitate future update operations.

5 A relational schema for generalisation logging

The model described in section 4 can easily be realised into a relational schema. Here we will omit most straightforward details and focus on how the source and target features are linked via operation references under this schema.

To simplify the discussion, we assume all features are in one feature table so that the feature ID alone is sufficient to identify and locate a feature (in case of multiple feature tables, a table name or table ID is also required to locate a feature). Also, the process ID plus operation ID are sufficient to identify and locate an operation.

5.1 GenOpRef Table

At present, we look at two designs of the GeoOpRef tables (Oracle data types are used):

Design 1

Column	Domain
ProcessID	NUMBER
OperationID	NUMBER
FeatureID	NUMBER
RefType	CHAR(1)

Design 2

Column	Domain
ProcessID	NUMBER
OperationID	NUMBER
FeatureIDMain	NUMBER
FeatureIDSecondary	NUMBER
RefType	CHAR(1)

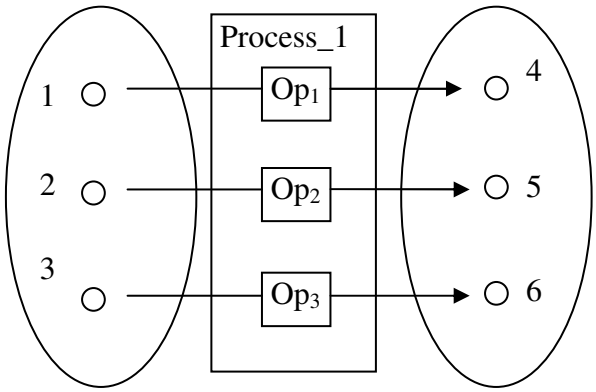
In the second design, an extra feature ID attribute (FeatureIDSecondary) is supplied for storing application specific information (see examples below). We believe the first design should be sufficient for most applications but the second design offers more flexibility and is worth further investigation.

RefType is either 'S' (for source), 'T' (for target) or 'C' (for context). This attribute enables us to store three relations (Operation-Source, Operation-Target and Operation-Context) in a single table.

5.2 Examples:

5.2.1 Line simplification (1:1 mapping)

In this case, we have a simplification process applied to a set of three coastline features. Assuming topological consistency is not considered, the process (ID: 1) may be divided into three independent operations (IDs: 1, 2, 3), each of which maps a source feature to a simplified target feature.



Source table

FeatureID
1
2
3

Target table

FeatureID
4
5
6

GenOpRef Table

ProcessID	OperationID	FeatureID	ReferenceType
1	1	1	S
1	1	4	T
1	2	2	S
1	2	5	T
1	3	3	S
1	3	6	T

We may carry out the following queries on these tables:

- Retrieve FeatureIDs of all source features of the process

```
Select FeatureID
From GenOpRefTable
Where ProcessID = 1 and ReferenceType = 'S';
```

- Retrieve FeatureID of all target features of the process

```
Select FeatureID
From GenOpRefTable
Where ProcessID = 1 and ReferenceType = 'T';
```

- Retrieve the source and target features of operation 1:

```
Select FeatureID, ReferenceType
From GenOpRefTable
Where OperationID = 1;
```

- Retrieve the target feature corresponding to source feature 1:

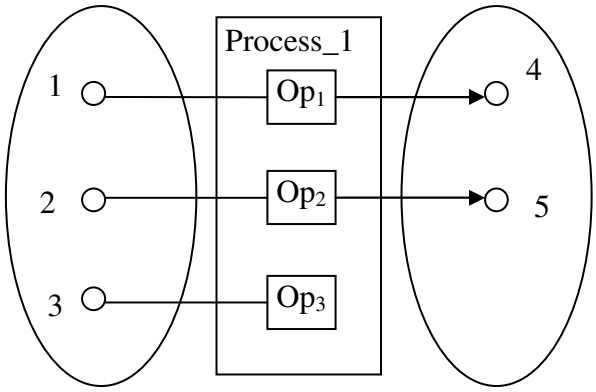
```
Select FeatureID,
From GenOpRefTable
Where OperationID =
      (Select OperationID
      From GenOpRefTable
      Where FeatureID = 1)
      AND
      ReferenceType = 'T';
```

In section 4 we described the concept of "selection-set". Effectively, the feature selection-sets of an operation are stored implicitly in GenOpRef tables. Also, the control pipeline mentioned in section 4 may be implemented by making the source and target features the same feature. In other words, a feature modifies and maps to itself via the operation. For example:

ProcessID	OperationID	FeatureID	ReferenceType
1	1	1	S
1	1	1	T
1	2	2	S
1	2	2	T
1	3	3	S
1	3	3	T

5.2.2 Selection/filtering:

Selection/filtering is the process to select some features for and omit others from the target dataset. Below is an example of context-free selection that takes place independently on individual feature, i.e. selection is solely driven by property values of a feature (e.g. the area of a building) with no spatial or a-spatial context.



Source table

FeatureID
1
2
3

Target table

FeatureID
4
5

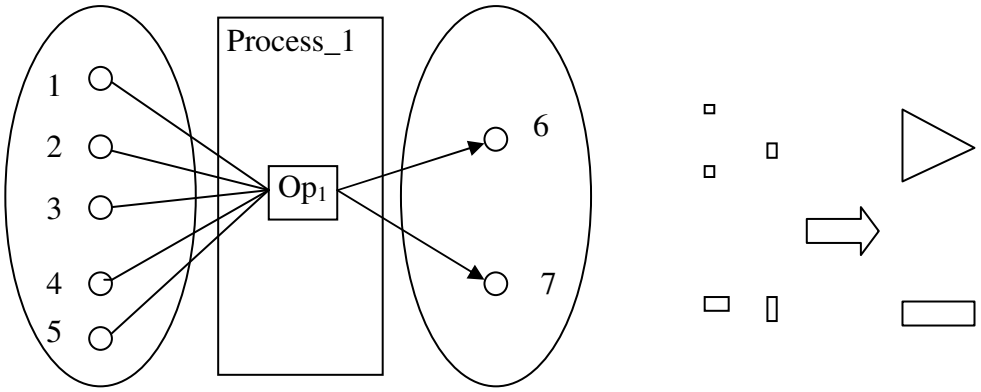
GenOpRef Table

ProcessID	OperationID	FeatureID	ReferenceType
1	1	1	S
1	1	4	T
1	2	2	S
1	2	5	T
1	3	3	S

For context-sensitive selection (which is an M:N mapping), the designs in the following sections may be used.

5.2.3 Building aggregation (M:N mapping)

In this case "aggregation" refers to the operation of representing multiple point features as a single areal feature (Regnauld and McMaster 2007).



Assuming there are 5 buildings (IDs 1, 2, 3, 4 and 5) which will be classified into two clusters and aggregated into two areal feautures (IDs 6 and 7). As the clustering is based on all buildings, a single operation (ID 1) is linked to these buildings.

Source table

FeatureID
1
2
3
4
5

Target table

FeatureID
6
7

GenOpRef Table -1

ProcessID	OperationID	FeatureID	ReferenceType
1	1	1	S
1	1	2	S
1	1	3	S
1	1	4	S
1	1	5	S
1	1	6	T
1	1	7	T

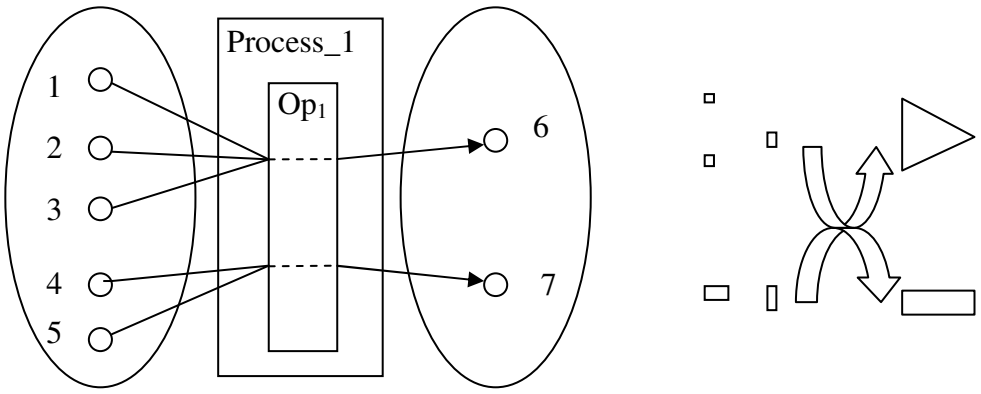
Similar queries can be performed on these tables.

The above table is based on the first design of GenOpRefTable. Consequently, the facts that source features 1, 2 and 3 are aggregated into target feature 6 and source features 4 and 5 are aggregated into feature 7 are not recorded. Should it be necessary to store such information explicitly, the second design for GenOpRefTable may be adopted:

GenOpRef Table-2

ProcessID	OperationID	FeatureIDMain	FeatureIDSecondary	ReferenceType
1	1	1	6	S
1	1	2	6	S
1	1	3	6	S
1	1	4	7	S
1	1	5	7	S
1	1	6	NULL	T
1	1	7	NULL	T

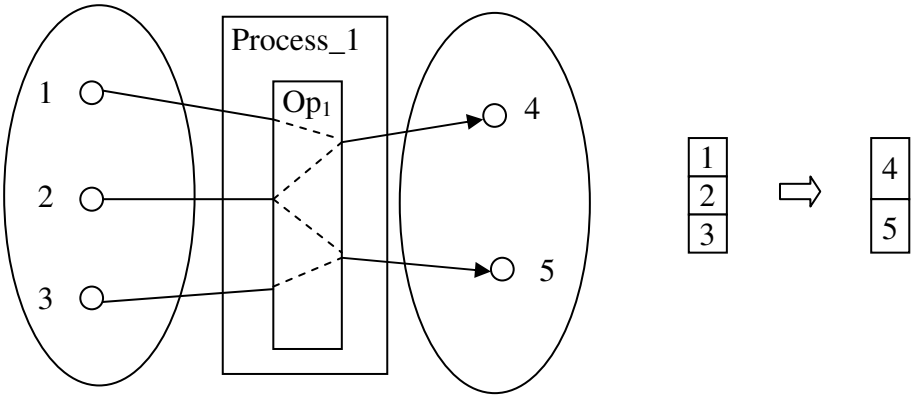
Now the sub-mappings {1, 2, 3} -> {6} and {4, 5}->{7} are also stored. The mapping on the opposite direction (target to source) is implicitly represented. This is one way of using the second feature ID attribute. Another way of using the second design in described next. Both are subject to application's interpretation.



Note that if the second design is adopted, the DISTINCT keyword is required in any queries when appropriate.

5.2.4 Generalisation of terrace houses (M:N mapping)

In previous examples, the mappings from source to target are "single-valued", i.e. a source feature, if individually linked, is mapped to at most one target feature. In some cases of generalisation, such mappings may be multi-valued. For example, three terrace houses 1, 2 and 3 are generalised into two (4 and 5). We may take the view that source feature 2 is mapped into target features 4 and 5.



Source table

FeatureID
1
2
3

Target table

FeatureID
4
5

GenOpRef Table

ProcessID	OperationID	FeatureIDMain	FeatureIDSecondary	ReferenceType
1	1	1	4	S
1	1	2	4	S
1	1	2	5	S
1	1	3	5	S
1	1	4	1	T
1	1	4	2	T
1	1	5	2	T
1	1	5	3	T

What is different from the previous example is that the bin-direction mapping is represented explicitly.

6. Issues in update propagation

In this section we will give a general analysis on the consequence of updates and how the model introduced above may facilitate update propagation.

Broadly speaking, there are three types of updates:

- Addition: a new feature is added into the source dataset
- Deletion: a feature is removed from the source dataset
- Modification: a feature in source dataset is modified (on its geometry or other properties). A modification may be substituted by a deletion and an addition in sequence

6.1 Deletion and modification

Deletion and modification is generally easier to handle as the operations relevant to a certain feature can easily be retrieved. An ordered list of operations may be re-constructed via topological sorting and all operations may be re-run accordingly.

6.2 Addition

Addition is a much more complicated issue. When a feature is added, the feature-class object for this feature will be retrieved to get the list of processes that have been applied to this feature type.

Subsequently, relevant operations will be retrieved according to the location of the new feature, using the spatial scope property of processes and operations.

6.3 Algorithm design consideration

As previously mentioned, the data model presented in this paper provides merely a generic framework for logging generalisation information. The implementation of such a framework will provide a protocol for integrating generalisation logging functionality into algorithm implementations. However, the integration is likely to be carried out on an algorithm by algorithm basis. Algorithm-specific update methods are also beyond the scope of this paper.

Such a framework may also impose some restrictions on whether certain generalisation algorithms may be feasible and how they should be implemented. In particular, the requirement of decomposing a process into independent operations could be problematic for some algorithms. We are considering the possibility of extending the model to include in-process precedence for operations so that the above requirement may be relaxed to an extent.

Another complicated task is to properly define the spatial scopes of operations. Although largely application dependent, in general, it should cover the source features as well as toughing any contextual features. A defined spatial scope could be approximate for the sake of efficiency but must not be smaller than the true scope.

For other researches relating to concrete MR-SDB update methods, please see, for example, (Bobzien et al. 2005, Sham Prasher and Xiaofang Zhou 2003, Zlatanova et al. 2004).

7. Summary and future work

In this paper we have presented a conceptual data model for linking representations for geographical phenomena at different resolutions, and recording generalisation information when these representations are generated.

A generalisation session is modelled as a DAG of generalisation processes. Each process is divided into operations which are the basic functional units in our model.

An operation transforms a set of source features to a set of target features, possibly with the help of a set of context features.

The generalisation history of operations applied or related to each feature can be stored in a database and retrieved efficiently when needed. This information may be used in subsequent update propagation to reflect changes in the real world.

The next stage of this ongoing research will focus on the physical design and prototype implementation of the data model in a DBMS environment to address any performance issue and refine or revise the physical model. In addition, a set of APIs for accessing this schema will be developed to assist the far more challenging task of implementing the generalisation logging mechanism in various generalisation algorithms.

Reference List

Anders, K.-H. and Bobrich, J., 2004, MRDB Approach for Automatic Incremental Update. *ICA Workshop on Generalisation and Multiple Representation 2004.*

Balley, S., Parent, C., and Spaccapietra, S., 2004, Modelling geographic data with multiple representations.

Becker, B., Six, H.-W., and Widmayer, P., 1991, Spatial priority search: an access technique for scaleless maps. *ACM SIGMOD*, 128-137.

Bobzien, M., Burghardt, D., Neun, M., and Weibel, R., 2006, Multi-Representation Databases with Explicitly Modelled intra-Resolution, Inter-Resolution and Update Relations.

Bobzien, M., Burghardt, D., and Petzold, I., 2005, Re-generalisation and Construction - Two Alternative Approaches to Automated Incremental Updating in MRDB. *XXII International Cartographic Conference (ICC2005).*

Chaudhry, O. and Mackaness, W. A., 2005, Visualisation of Settlements Over Large Changes In Scale. *ICA Workshop on Generalisation*, A Coruna, Spain, 7-8 July, http://ica.ign.fr/Acoruna/Papers/Chaudhry_Mackaness.pdf.

Douglas, D. and Peucker, T., 1973, Algorithms for the reduction of the number of points required to represent a digitised line or its caricature. *The Canadian Cartographer*, **10**, No. 2, 112-122.

Goodwin, J., 2005, What have Ontologies ever done for us - potential applications at a National Mapping Agency. *OWL: Experiences and Directions*, Galway, Ireland, 11-12 November, http://www.mindswap.org/OWLWorkshop/sub18.pdf.

Grünreich, D., 1985, Computer-Assisted Generalization. *Papers CERCO-Cartography Course*.

Haunert, J.-H. and Sester, M., 2005, Propagating Updates Between Linked Datasets of Different Scales.

Kilpeläinen, T. and Sarjakoski, T., 1995, Incremental Generalization for Multiple Representations of Geographic Objects. In *GIS and Generalization: Methodology and Practise* (Taylor & Francis), 209-218.

Ordnance Survey, 2008, OS MasterMap – intelligent, location-based data

Puppo, E. and Dettori, G., 1995, Towards a formal model for multiresolution spatial maps. *Advances in Spatial Databases (SSD95)*, 152-169.

Ramer, U., 1972, An iterative procedure for polygonal approximation of planar closed curves. *Computer Graphics and Image Processing*, **1**, 244-256.

Regnauld, N., 2005, Spatial structures to support automatic generalisation. *Proceedings of the 22nd International Cartographic Conference*, **CD-ROM**.

Regnauld, N. and McMaster, R. B., 2007, A Synoptic View of Generalisation Operators. In *Generalisation of Geographic Information: Cartographic Modelling and Applications* (Elsevier), pp. 37-66.

Sarjakoski, T., 2007, Conceptual Models of Generalisation and Mulitple Representation. In *Generalisation of Geographic Information: Cartographic Modelling and Application* (Elsevire Ltd), 11-35.

Sham Prasher and Xiaofang Zhou, 2003, Efficient Update and Retrieval of Objects in a Multiresolution Geospatial Database. *15th International Conference on Scientific and Statistical Database Management (SSDBM'03)*.

Skogan, D. and Skagestein, G., 2005, An Automated Incremental Generalization Method. In: Skogan, D. (Author) Multiple-Resolution Geographic Data and Consistency, PhD, Department of Informatics, University of Oslo, Norway.

Stell, J. and Worboys, M., 1998, Stratified map spaces: A formal basis for multi-resolution spatial databases. *The 8th International Sympo-sium on Spatial Data Handling*, 180-189.

Timpf, S., 1998, Hierarchical Structures in Map Series, Doctoral Thesis, Technical University Vienna.

Zhou, S. and Jones, C. B., 2001, Design and Implementation of Multi-scale Databases. *Advances in Spatial and Temporal Databases, Proceedings of the 7th International Symposium on Spatial and Temporal Databases SSTD 2001*, 365-386.

Zhou, S. and Jones, C. B., 2003, A multirepresentation spatial data model. *Advances in Spatial and Temporal Databases, Proceedings of the 8th International Symposium on Spatial and Temporal Databases SSTD 2003*, 394-411.

Zlatanova, S., Stoter, J. E., and Quak, W., 2004, Management of multiple representations in spatial DBMSs. *7th AGILE Conference on Geographic Information Science*.