An Internationalised Object Data Model: a Locale-based Approach

Alex G. Büchner[*], David A. Bell[†], John G. Hughes[*]

[*] Northern Ireland Knowledge Engineering Laboratory, University of Ulster
[†] School of Information and Software Engineering, University of Ulster
email: {ag.buchner, da.bell, jg.hughes}@ulst.ac.uk

Abstract

Internationalisation of data is a special case of semantic heterogeneity which arises due to differences between national methods of representation and organisation. We argue that a specialised data model has advantages over a generic solution, with respect to accuracy, exhaustiveness and performance gains. To represent international data, the concept of locales is introduced, which encompasses locale identity, localised equivalence specifications for atomic and complex types, locale mediation, as well as a hierarchical topology of locales, which represents a shared ontology. The introduced concept is then embedded in an object data model, viz. ODMG, and its object definition and query language are extended accordingly. A prototype which has been developed is described and the encouraging results are evaluated.

Keywords:
semantic interoperability, heterogeneous databases, internationalisation, context mediation, ontologies, object oriented databases.

1 Introduction and Motivation

Internationalisation in database systems refers to pre-existing heterogeneity, which arises due to national differences in language and geography. It has hitherto been tackled with special representation and query constructs for character sets, information from multiple calendaric systems, as well as numeric data.

Although we believe that for future applications there is substantial need for tailored internationalised data models, somewhat in the same way as tailored temporal data models, we see internationalisation of databases as a special case of semantic heterogeneity, for various reasons. Firstly, a domain-specific underlying ontologies promise more accurate results in the field of internationalised database systems. This is essential in areas like currency risk management, stock markets or any other discipline in which world-wide transactions are involved. [Mad96] lists a wide range of financially related examples of this type of large-scale semantic heterogeneity. Secondly, the performance of storing and querying world-wide information electronically, on media such as the Internet, can be increased by a specialised approach. This is particularly relevant in time-critical operations. Lastly, an internationalised approach is more likely to cover a quasi-exhaustive range of requirements, including social, political, and environmental facts covering, for example, the needs of ethnic minorities. The literature reference example in Figure 1, which is used throughout the paper, has been created to illustrate possible internationalised semantic heterogeneities, among others, which have to be reconciled.

ref_1 ('R. Pike, K. Thomson'; 'Hello World or Καλημερα κοσμε or こんいちは世界', 10-11-1993; {US$2.50; £1.80; 3,20 DM; FF 9.20})
ref_2 ('K. Lunde'; '日┌ ┌ 情┌ ┌ 理'; 1993年10月12日; {IR£21.95; ¥ 2800; HK$29.50})
ref_3 ('François Bancilhon, et. al.'; 'Building an Object-Oriented Database System: The Story of O_2'; ┌ ┌ ┌ ┌ ,┌ ,┌ ┌ ; {FF195; ┌ ,┌ ┌ ┌ ┌ })

Figure 1. An example of international literature references

The contribution of this work is the development of an internationalised object data model which is based on the concept of locales. In order to achieve this goal, we have adopted [Sci94]'s approach of semantic values and mapped its model onto an object-oriented counterpart.

The outline of this paper is follows. §2 introduces the concept of locales, which deals with locale identity, localised equivalence specifications for atomic and complex types, locale mediation, and the hierarchical organisation of locales. In §3 to §5, we extend the ODMG model with these concepts, and its object definition and query language. A prototype which has been developed is described in §6 and the encouraging results are evaluated. §7 compares related work, before conclusions are drawn and further research is outlined in §8.

2 The Concept of Locales

2.1 Locales and Locale Identity

A locale represents structural and behavioural characteristics which are shared by a geographical, political or abstract region, represented as a shared ontology.

Definition 1. A **locale** l contains a set of properties $P_l = \{p_{l_1}, p_{l_2}, p_{l_3}, \ldots, p_{l_n}\}$ where $P_l \subseteq P$ and $P = \{p_1, p_2, p_3, \ldots, p_m\}$, which represents an application-specific ontology O.

The general idea of the locale identity is that every single attribute instance has an additional attribute locale identifier.

Definition 2. A **locale identifier** *lid* is a unique representation of a locale l, which is chosen from a set of locales $L = \{l_1, l_2, l_3, \ldots, l_n\}$, and allotted to a semantic (that is localised) value s of type t with value v, such that each localised entity in an information space is represented as the triple $s = (t, v, lid)$.

Example 1. ref_1 - ref_3 ontologise internationalisation and localisation features in which possible properties are the exchange rate of a currency, the scale of a numeric value and the granularity of any attribute. A semantic value s_1 = (currency, 2.50, USA) would be represented as s_2 = (currency, 293.69, Japan) when interchanged from locale USA to locale Japan, given an exchange rate of 1: 117.48.

This approach is a generalisation of proposals, in which every attribute – but not each attribute instance – has a specific locale allotted to it, for instance, the COLLATE and CHARACTER SET keywords in SQL-92 [Mel93]. This tagging method has been used in a wide range of applications, such as class variables in object-oriented systems, meta-knowledge in expert systems, and rendering information in word processors and mark-up languages. Consequently, every locale contains type- *and* locale-sensitive semantics and, thus, controls the behaviour of these opaque data types. Behaviour in a generic database context means that equality has to be re-defined to allow a uniform representation of new functionality in order to handle information in internationalised (multi)databases. Therefore, traditional comparison operations have to replaced by localised

ones, that is the concept of equality has to be substituted by that of equivalence (also referred to as semantic equality). A type-specific semantics is required to express the behaviour of the taxonomies of operations which can occur when dealing with international data.

2.2 Localised Comparison Operations

Inhabitants from culturally different environments often judge an identical situation with different measures of discernibility which requires diplomatic services among the participants when it comes to multinational co-operations. This natural and widely accepted phenomenon is represented by comparisons between and among values from different locales.

Definition 3. Let $s_1 = (t_1, v_1, lid_1)$ and $s_2 = (t_2, v_2, lid_2)$ be two semantic values and q be a **comparison operator** such that $q \in \{=, <, <=, >, >=, <>\}$. Then the expression $e = s_1\ q\ s_2$ is valid iff $t_1 = t_2$ or t_1 can be converted to t_2 and vice versa[1].

Dyadic operations upon identical data types with the same locale can be performed traditionally; dealing with data from different locales depends crucially on q-operations. With dyadic operations from different locales, the operands are either in canonical forms, have to be transformed to such, or a particular q-operation has to be computed which is sensitive to both locales l_1 and l_2. While unary operations are straightforward, operations involving comparison require a mechanism which guarantees semantic equivalence among values from different locales.

2.3 Locale Comparison Specification

Formulating comparisons to resolve semantic heterogeneity among values from different locales cannot be expressed in a single mechanism. Examples of the diversity of comparisons in the area of internationalisation are collation sequences (including complex orders, like multiscript sorting [LaB93]), transformations of different calendaric systems, or operations upon exchange rates in different currencies. To perform equivalence operations as specified in Definition 3, each value has to be converted into a form in which traditional q-operations can be computed. Due to the fact that there *is* no canonical form of localised values of certain types, a more sophisticated approach is required, which takes the locales of values into account. Sciore, et. al. [Sci94] have identified five different ways of declaring semantic value specifications, namely rules, predicates in logic, functional expressions, tables, and tagged attribute properties.

While numeric-based data types, for example, currencies or dates, can usually be transformed into a locale-independent canonical form, this is not the same for strings, abstract and complex data types, which might behave inconsistently in different locales (also known as source context). Additionally, different users (called receiver context) might want to specify individual comparison operations.

Example 2. In a Greek library database, one might want to put all Greek titles first (according to ISO 8859-7), and then all non-Greek European titles (according to ISO 8859-0). On the other hand, a linguist who is familiar with both writing systems and not concerned about lexicographics, might want to collate words according to an order table in which all letters have their equivalents in each writing system, for instance, x and Ξ, y and Ψ, z and Ω.

The maximum number of specifications for each type is of complexity $O(|L|) = |L|^2$, where $|L|$ denotes the cardinality of the set of locales L. Although this quadratic order could theoretically lead to an explosion of comparison specifications, in reality often few specifications are required. Additionally, in §2.5, a hierarchical

[1] Whether a semantic value can be converted will be specified in §2.6, when locale mediation is introduced.

locale inheritance mechanism will be introduced to reduce the number of comparison specifications. To specify the comparison of two localised values, a locale mediator, which takes the two semantic values and returns their type-and locale-specific order, has to be incorporated.

Definition 4. A **locale mediator** m is an evaluation mechanism that takes the localised values s_1 and s_2 as input and returns the order of that value pair, such that

$$m(s_1, s_2) = \begin{cases} -1 & \text{if } s_1 > s_2 \\ 0 & \text{if } s_1 = s_2 \\ 1 & \text{if } s_1 < s_2 \\ \text{Null} & \text{if undefined} \end{cases}$$

The output of the mediator m is also called the order o between s_1 and s_2, which simplifies defining the specification of orders for complex data types (see §2.4). Example mediators are defined in §4.2. The specification of a generic locale mediator is described in detail in §2.6. Facilitating the mediator as defined, although currently a black box, it is now possible to define the following.

Definition 5a. Let $s_1 = (t_1, v_1, lid_1)$ and $s_2 = (t_2, v_2, lid_2)$ be two semantic values and $e = s_1\ q\ s_2$ a valid expression. s_1 is **equal** to s_2 iff $(v_1 = v_2) \wedge (t_1 = t_2) \wedge (lid_1 = lid_2)$. s_1 is **equivalent** to s_2, iff $m(s_1, s_2) = 0 \wedge (lid_1 \neq lid_2)$.

Since locale equivalence is a superset of locale equality, the former can be embedded in the latter and all comparisons between two values in an information space can be carried out using the locale mediator m — independently of the content of the two locale identifiers:

Definition 5b. Let $s_1 = (t_1, v_1, lid_1)$ and $s_2 = (t_2, v_2, lid_2)$ be two semantic values and $e = s_1\ q\ s_2$ a valid expression. s_1 is **semantically equal** to s_2, iff $m(s_1, s_2) = 0$.

2.4 Complex Data Types

In addition to atomic data types, the concept of locale identity also has to be supported by complex types such as sets, bags, lists, arrays and dictionaries, as well as any arbitrary user-defined abstract type. Also, it is insufficient to calculate semantic in/equality only, since an order of data of complex type is required for various database-related operations, such as ordering, indexing and grouping. The examples, which are used to illustrate the mechanism and purpose of each defined order, are taken from hypothetical stock market information across the globe. The complexity is kept as minimal as possible for didactical reasons.

To compute the order of two complex objects with the same structure, we define the following measure, which gives elements with a smaller index higher priority and which is normalised to the integer interval [–1,1] via the signum function.

Definition 6. Given two complex objects $o_1(s_{1_1}, s_{1_2}, s_{1_3}, ..., s_{1_n})$ and $o_2(s_{2_1}, s_{2_2}, s_{2_3}, ..., s_{2_n})$, the **object order** $o(o_1,o_2)$ is defined as

$$o(o_1, o_2) = \operatorname{sgn}\left(\sum_{i=1}^{n} (n - i + 1)^2 * m(s_{1_i}, s_{2_i}) \right)$$

Example 3. Using an open-close scenario and given o_1 = ((currency, 12.00, USA), (currency, 14.00, USA)), o_2 = ((currency, 1404.00, Japan), (currency, 1652.00, Japan)) and an exchange rate of 1:117.50, which is facilitated by the mediator, the object order o(o_1, o_2) is calculated as following. sgn(4 * $m(s_{1_1},s_{2_1})$ + (1 * $m(s_{1_2},s_{2_2})$) = sgn(4 * -1 + 1 * 1)) = -1, and thus $o_1 > o_2$.

Operations upon objects with different structure depend on the type of the (complex) semantic values. We now define generic orders for three kinds of complex types, which can be overridden in application-specific mediators. The collection types considered in here are sets, vectors and lists. For sets, the results of the mediator for each individual element of the difference sets are calculated and mapped onto the result set {−1,0,1}.

Definition 7. Let $c_1^s = \{s_{1_1}, s_{1_2}, s_{1_3}, ..., s_{1_n}\}$ and $c_2^s = \{s_{2_1}, s_{2_2}, s_{2_3}, ..., s_{2_m}\}$ be two internationalised sets, and $\Delta_1 = c_1^s - c_2^s$ and $\Delta_2 = c_2^s - c_1^s$ be the two difference sets. The **set order** o(c_1^s, c_2^s) is defined as

$$o(c_1^s, c_2^s) = \begin{cases} \operatorname{sgn}\left(|\Delta_1| - |\Delta_2|\right) & \text{if } \Delta_1 = \varnothing \vee \Delta_2 = \varnothing \\ \operatorname{sgn}\left(\sum_{i=1}^{|\Delta_1|}\sum_{j=1}^{|\Delta_2|} m\left(s_{1_i}, s_{2_j}\right)\right) & \text{otherwise} \end{cases}$$

Example 4. Given are c_1^s = {(currency, 12.00, USA), (currency, 20.00, HK)}, c_2^s = {(currency, 12.00, USA), (currency, 8.785, Egypt), (currency, 17.00, Germany)}, and an exchange rate between the HK$ and the Egyptian Pound of 1: 0.4393. Since the first two elements in each set are semantically equal, the order between the two sets is only dependent on the third element in the second set, and thus $c_1^s < c_2^s$.

For set operations ($\cap$, $\cup$, $-$, $\supset$, $\subset$, $\in$ and derivatives thereof), as well as composite predicates ($\forall$ and $\exists$), the existing locale mediators are used to evaluate the results. The second collection type to be defined is type vector, which re-uses the (lexicographic) object order of Definition 6 for physically overlapping components and adds a fraction for the remaining part which is only relevant in case of equality.

Definition 8. Let $c_1^v = \langle s_{1_1}, s_{1_2}, s_{1_3}, ..., s_{1_n}\rangle$ and $c_2^v = \langle s_{2_1}, s_{2_2}, s_{2_3}, ..., s_{2_m}\rangle$ be two internationalised vectors**.** The **vector order** o(c_1^v, c_2^v) is defined as

$$o(c_1^v, c_2^v) = \operatorname{sgn}\left(\left(\sum_{i=1}^{\min(n,m)} \left(\min(n,m) - i + 1\right)^2 * m\left(s_{1_i}, s_{2_j}\right)\right) + \frac{m-n}{\max(n,m)}\right)$$

Further examples can be derived straightforwardly from Example 4. The third collection type for which an order is to be defined are lists, which is calculated similarly to vectors, the only difference being the declaration of lists.

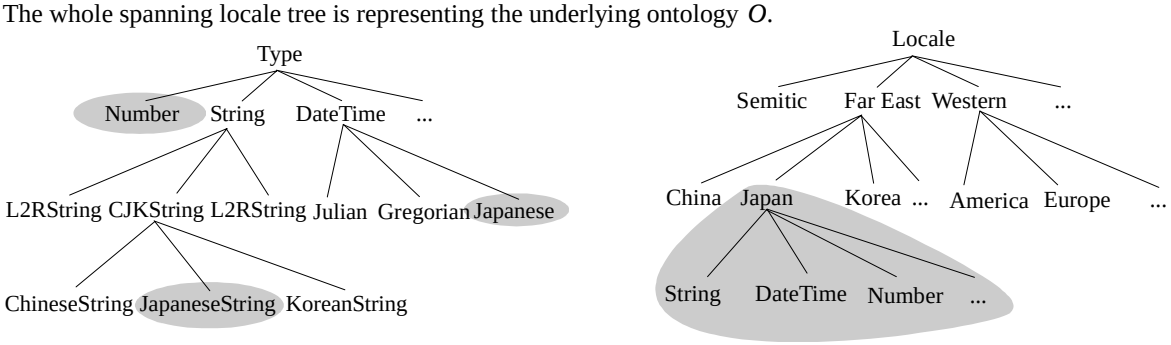
Definition 9. Let $o_1^l = \langle s_{1_1}, s_{1_2}, s_{1_3}, ..., s_{1_n} \rangle$ and $o_2^l = \langle s_{2_1}, s_{2_2}, s_{2_3}, ..., s_{2_m} \rangle$ be two internationalised lists**.** The **list order** $o(o_1^l, o_2^l)$ is defined as

$$o(o_1^l, o_2^l) = \mathrm{sgn}\left(\left(\sum_{i=1}^{\min(n,m)} (\min(n,m) - i + 1)^2 * med\left(s_{1_i}, s_{2_i}\right)\right) + \frac{m-n}{\max(n,m)}\right)$$

2.5 Topological Organisation of Locales

To minimise redundancy of locale declarations (for example, linguistically related writing systems or dates in different time zones but from a single calendaric system), locales are organised in a hierarchy. Thus, structure and behaviour can be inherited, and overloading and overriding mechanisms can be applied to locales.

There are two ways in which to organise locales hierarchically. One is by types, that is for every data type a locale-based hierarchy is built (Figure 2a). The other is by locales, that is locales are organised in a hierarchy and for every type, structure and mediators are specified (Figure 2b). Depending on the degree of fluctuation of types and locales respectively, a system is easier to maintain, which has been indicated by the locale-related shaded parts in each hierarchy. It is felt that it is more natural to group types around locales, rather than the other way round, and hence, this approach has been chosen. Also, new locales are usually added to a system, rather than new data types with *different* locale characteristics. The root locale node is always Locale, which contains some basic generic structure (like the name and properties) as well as behaviour (for instance, the mediation execution mechanism and order computations) which is shared by all inheritors, unless overridden. The whole spanning locale tree is representing the underlying ontology *O*.



(a) Type-based Locale Hierarchy (b) Locale-based Type Hierarchy

Figure 2. Topological Organisation of Locales

Definition 10. A **locale hierarchy** *O* is an undirected, connected, acyclic graph which is defined as the tuple $O = (L,E)$, where $L = \{l_0, l_{1_1}, l_{1_2}, ..., l_{1_{|l_1|}}, l_{2_1}, l_{2_2}, ..., l_{2_{|l_2|}}, ..., l_{n_1}, l_{n_2}, ..., l_{n_{|l_n|}}\}$ and $E = \{e_1, e_2, e_3, ..., e_m\}$. Each e_k has the form $e_k = \{l_i, l_j\}$; $l_i, l_j \in L$, l_0 has indegree 0, $l_1 ... l_n$ have indegree 1. l_i is **sublocale** of l_j iff $l_i \subset l_j$; l_i is **superlocale** of l_j iff $l_j \subset l_i$.

A further advantage of the hierarchical arrangement of locales is the possibility of packaging locale sub-trees; we refer to these sub-trees as resources.

Definition 11. A **resource** r is defined as a subgraph of a locale hierarchy O, such that $r \subset O$, that is $r \in O \wedge r \neq O$.

Example 5. The highlighted Japan subgraph in Figure 2b is a resource, which could contain the following properties:

JapaneseString(WritingSystem ∈ {Hiragana, Katakana, Kanji})
JapaneseDate(Calendar = Emperor; TZ = +8)
JapaneseNumber()

In this particular case, Japanese Strings have one property, which can take three possible values, dates have two properties and numbers inherit all properties from its superlocale, FarEast. The operational part of locales will be outlined in the next subsection.

2.6 A Locale Mediator

The purpose of the locale mediator is to compute the order of values from two locales. A variety of constructs has been suggested in the literature (mostly based on [Wie92]), but to reconcile locale heterogeneity, functions (which encompass arithmetic expressions as well as rules) and tables are sufficient. Although functions have been designed to convert numbers and derivations thereof (like datetimes), they can also be used by strings or data of abstract type. Similarly, tables have originally been designed to represent collation sequences of characters, but they are not limited to them; data from any other type can be organised in tables, too. The generic locale mediator skeleton, which is based on Definition 5b, has the following layout:

m ← s1, s2
    function <conversion_function> |
    table <collation>
m → {–1, 0, 1}

The locale mediator m takes two semantic values s_1 and s_2; their types and locales can be, but are not mandatorily, different. <conversion_function> is an implementation-specific set of linguistic programming constructs, which should cover minimal, but sufficient functionality, and <collation> is the name of an ordering table. The mediator returns the order of the two semantic input values where the flags –1, 0 and 1 represent 'less than', 'equal' and 'greater than', respectively. Note that the mediator does not return the locale of the result. This will either be requested explicitly by a formulated query, or implicitly given by default locales (see §4.1 and §5)

2.7 Recapitulation

The topology independent layer view in Figure 3 graphically presents the concept of locale identity. For simplicity, only a minimised locale hierarchy is used from the internationalisation example, and only a few attribute instances are allocated to properties. Referring to the literature reference example given in Figure 1, an entity is the definition of a reference, ref_1 – ref_3 are instances thereof, and every single attribute instance is allotted to a locale identifier from the locale hierarchy.

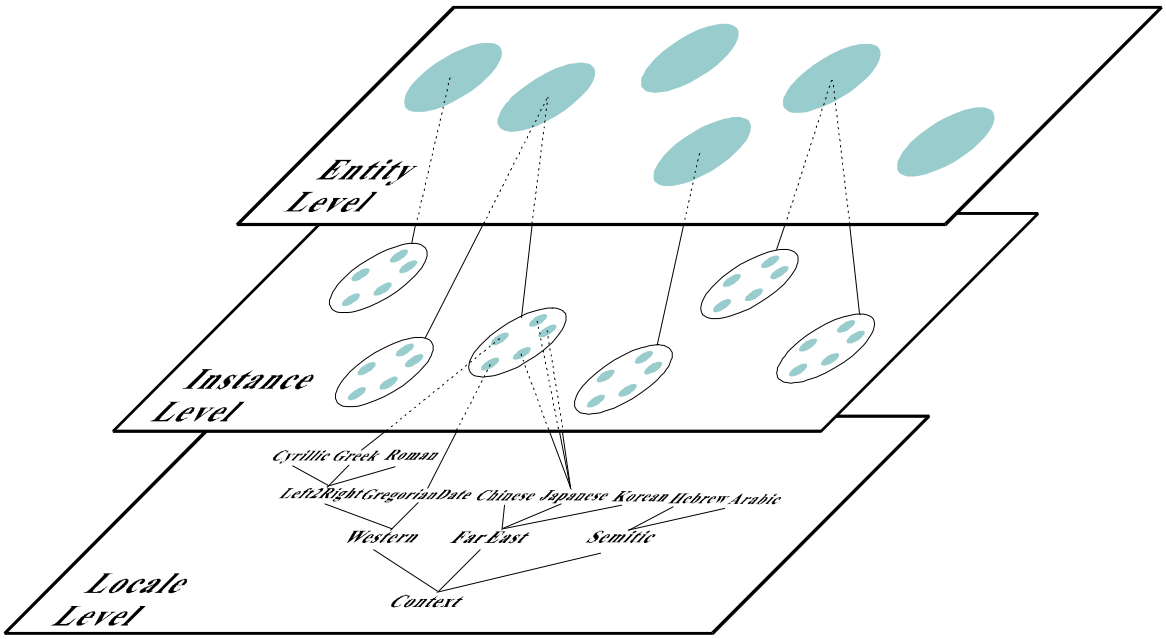


Figure 3. Topology-independent layer view

The four proposed fundamental constructs (locale identity, localised equivalence, inheritance of locales, and a locale mediator), will now be embedded in an appropriate data model.

3 Locale Object Data Model

Embedding the concept of locales in a data model requires the representation of locale semantics and inheritance, as well as polymorphism mechanisms in order to build a locale hierarchy. Traditional data models are inappropriate in providing these facilities, and so an object data model has been chosen [Hug91]. For the purposes of this paper, an evolutionary approach has been preferred, and the ODMG object model [Cat97] has been used. However, none of the extensions made here are ODMG-specific; they can be applied to any object data model.

A new meta data type Locale is created which contains information about locales themselves. Its structural part consists of a unique name identifier (locale identity), a set of properties and a set of super_locales. To model single locale inheritance, a recursive relationship, referred to as 'sublocalising', has been embedded. Two methods to define and remove locales (create and delete) as well as two methods to attach and detach mediators to a locale (add_mediator and remove_mediator) are needed. To check the existence of a locale, an exists? method is required.

To connect the type Locale a 1:n relationship Localisation to the abstract type Attribute_Instance has been added. To keep the principle of encapsulation, the type Attribute_Instance has been enhanced by the mutator method set_locale to change the locale of an attribute instance, and the accessor method get_locale to retrieve it.

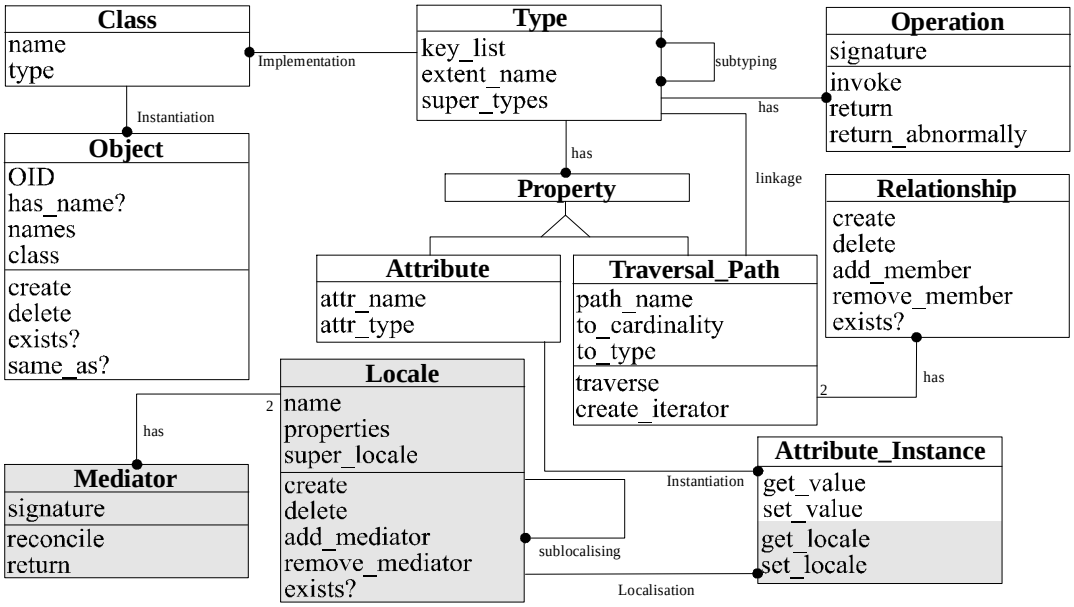


Figure 4. Localised ODMG meta model[2]

The structural part of the Mediator contains a signature, which consists of the mediator name, the mediator type (function or table), as well as the types and locales of the two semantic input values analogous to the specification in §2.6. To send a message to a mediator, the reconcile method is called, which uses the return method for the response. The Mediator type is connected through a 2:n has-relationship to the Locale type. Note that the mediator could have been modelled as part of the locale, but for reasons of encapsulation and scalability it has been designed as a separate component.

The three outlined extensions (introduction of the Locale and Mediator types as well as attribute instance enhancements) lead to the simplified ODMG meta model depicted in Figure 4. The diagram makes use of the graphical notation of OMT, highlighting the added internationalisation features. In addition, to represent locale properties and behaviour transparently, the model is capable of packaging locales and connected resources, and using them when needed.

4 Locale Object Definition Language

To keep consistency with the ODMG model enhancements incorporated above, the ODMG-ODL is extended in this section. Again, the extensions are not ODMG-specific; syntactical statements are just borrowed for convenience.

In general, in ODMG every type is defined as an interface, to be compatible with the OMG interface definition language IDL. An interface consists of an extent, a set of keys, a number of attributes and relationships, as well as a range of operation signatures. Only a few syntactical extensions are necessary to support all localised semantic constructs of the locale object data model.

[2] Modified from [Loo93].

4.1 Attribute Declaration

An optional default locale, which can be allotted to every attribute definition, has been added. Another optional constraint is the limitation of one attribute to a non-empty set of locales. The BNF for specifying an attribute looks as follows, where the sidebar illustrates the extensions made to the original form.

```
<attrib_dcl> ::=
    [readonly] attribute
    <domain_type> <identifier>
    [[<positive_int_const>]]
    [LocaleDefault <locale>]
    [LocaleConstraints <locale_set>

<locale_set> ::=
    <locale> | <locale> , <locale_set>

<locale> ::= <identifier>
```

If both properties are assigned, LocaleDefault must be a member of LocaleConstraints. If a type has a supertype, locale attribute properties are inherited. To follow principles of object-orientation, these properties can also be overridden. If either LocaleDefault or LocaleConstraints is overridden, clashes have to be detected automatically, that is LocaleDefault still has to be a member of LocaleConstraints.

Example 6. The interface of the reference type of the literature example in Figure 1, can be declared as follows, assuming that an abstract data type Currency has been defined:

```
interface Reference
(
    extent references
    keys author, title
)
{
    attribute String author LocaleDefault USA;
    attribute String title LocaleDefault USA;
    attribute Date published LocaleDefault Gregorian LocaleConstraints Gregorian, Julian;
    attribute Set<Currency> prices LocaleDefault UK
}
```

Allocation of a LocaleDefault to the currency set can lead to misinterpretation, because it is unclear whether the locale UK is meant to be default for the whole set of prices, or for every single element of the set. We assume that it is both, since it seems to be a natural way of declaring default locales for complex data types.

4.2 Locale and Mediator Declaration

Due to the fact that locales are treated similarly to types and mediators similarly to operations, they too have to be part of the database schema, and therefore syntactical constructs have to be provided for their declaration. Declaration of locales consists of naming a locale identifier and superlocales, which are implicitly given in the specification of a locale. Declaration of mediators requires a signature (identifier and a mediator type) as well as the mediator implementation. The full BNF for Locale and Mediator declaration has been listed in the appendix.

Example 7. The following is the declaration of the three locales Western (sublocale of Locale), USA (sublocale of Western) and USMountain (sublocale of USA) for the data type Date. WesternDate contains one property Gregorian which is the only permitted value, USADate is a kind of virtual locale without any additional properties, and USMountainDate inherits the Calendar property from the locale Western and adds TZ as additional property.

```
Locale Western (Date) : Locale
    Properties Calendar PropertyDefault Gregorian PropertyConstraints Gregorian

Locale USA (Date) : Western

Locale USMountain (Date) : USA
    Properties TZ PropertyDefault —6
```

Example 8. The specification of two simple alternative locale mediators — one based on a function, one on a table — to specify the comparison from US$ to HK$ with and without 10% overhead for amounts over 10.000 HK$ can be defined as follows.

```
Mediator USA_HK Function USA(Currency) HongKong(Currency)
(
    if HongKong.Value > 10000
    then (HongKong.Value * 0.12) * 1.1
    else HongKong.Value * 0.12
)

Mediator USA_HK Table USA(Currency) HongKong(Currency)
(
    NYSE
)
```

As can be seen from the given example, values which can be accessed through the dot notation (as can all other properties) are implicitly converted to some canonical form, which then can be evaluated by the mediator execution mechanism and the comparison result returned accordingly. It is important to note that it is possible that semantic values of different types (atomic, complex or abstract) can be compared, which is regularly required for semantic heterogeneity conflict resolution. Also, different users in different contexts (that is locales) can replace each mediator with their own perceptive version, which guarantees the autonomy of the source and receiver sites. After specifying syntactical extensions of the structural section, the operational part has to be enhanced, too.

5 Locale Object Query Language

To represent locale information within queries minor syntactical extensions have been introduced to the ODMG-OQL. Every locale property of every attribute has to be accessible, which is realised through the following notation, where attribute can be either an atomic or a complex object:

```
<attribute>@<property>
```

To guarantee orthogonality of the query language, locale identity can be used with every attribute in all valid query statements.

Example 9. To retrieve all titles which are allocated to the Japanese locale and written by American authors (authors who have been allotted to the locale USA!), the following query has to be formulated:

```
SELECT r.title, r.title@WritingSystem
FROM References r
WHERE r.author@WritingSystem = USA AND r.title@WritingSystem = Japan
```

Internally, the locale information must be accessed through the defined accessor method get_locale. Joins now have enhanced semantics, because *q*-operators check not only for values, but also for localised semantics.

Example 10. Hence, the following OQL statement returns all authors with identical locales, who wrote at least one reference and one or more book (specified in a separate class):

```
SELECT r.author, r.title, r.title@WritingSystem
FROM References r, Books b
WHERE r.author = b.author
```

With the introduction of the locale indicator and the usage of localised comparison operators all queries can be expressed. This also holds for aggregate functions, such as the order by clause and the group by clause. If no specific locale information is given, the newly defined comparison operators are used to imitate these operators. But it might also be of interest to sort and/or group a retrieved set by its locale.

Example 11. The following query groups the retrieved data by price categories and orders them by the price locale and author.

```
SELECT r.author, r.title
FROM References r
GROUP BY   cheap: r.price <= 20@Locale = USA,
           normal: r.price > 20@Locale = USA AND r.price < 30@Locale = USA,
           expensive: r.price >= 30@Locale = USA
ORDER BY r.price@Locale, r.author
```

The few examples above illustrate the usage of the localised OQL and show that only one simple extension, viz. the @ notation, has been necessary to formulate queries dealing with locale heterogeneities.

6 Prototype Implementation and Results

A locale mediator prototype has been developed to verify the proposed concepts and to evaluate results as well as the performance of the system (see screenshot in Figure 5). The system supports hierarchically organised locales, comparison specifications based on functions and tables, and a locale mediation mechanism for standard data types. The concepts for these operations have been described and specified in §§2.1-2.3 and §§2.5-2.6. Currently neither complex data types have been implemented, nor the ODL and OQL extensions outlined in the previous two sections. The prototype shows the applicability of the proposed concepts and verifies the validity of suggested reconciliation mechanisms.

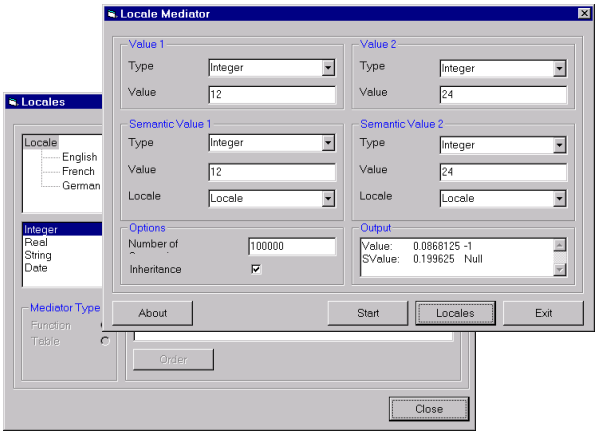


Figure 5. Screenshot of the *Locale* Prototype

Figure 6 shows graphically the results benchmarking the overhead for four standard atomic data types which is incurred by the introduction of internationalisation features on attribute instance level. The results show the increase in computation time by a factor 9.5 to 12.5 when locale mediation between two semantic values has to be performed. The support of locale inheritance increases the time spent by another 8-10% for each level.

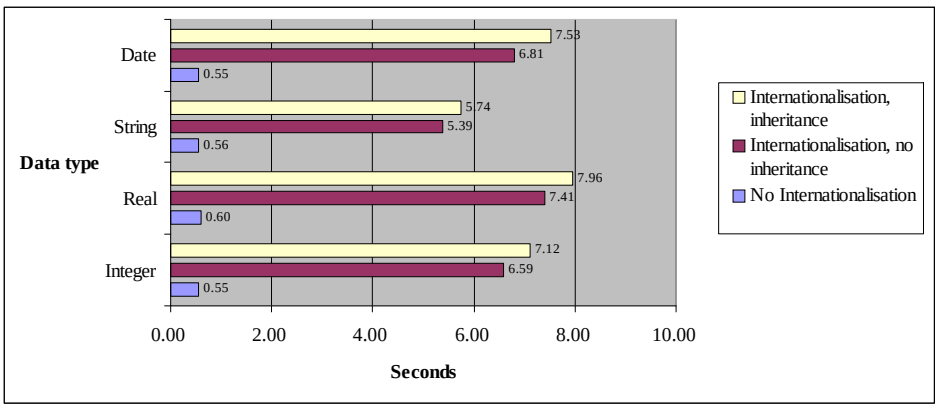


Figure 6: Benchmark Results of the *Locale* Prototype

7 Related Work

Related work can be subdivided into two main groups, namely internationalised data models[3] and systems that tackle semantic heterogeneity generically, which can be applied to internationalisation problems. Relevant research of latter category concentrates on context information and ontologies; the reader is referred to [Hul97] for a general survey.

[3] Limited data models, which support national language support only are not considered in here.

Yoshioka and Melton [Yos93] have enhanced the SQL standard to handle multilingual strings, by extending the standard CHARACTER data type to NATIONAL CHARACTER. Following this monolithic approach, the same data types (localised strings) can be incompatible with each other. Therefore, coercibility rules, based on the coercibility attributes 'explicit', 'implicit', and 'coercible' (in order of priority), have been introduced to keep the consistency of the underlying relational calculus and to guarantee operations upon *q*-operators [Mel93]. Most approaches (including SQL and ODMG) represent multidatetime information, that is date, time, and timezones using [Dat88]'s recommendation for adding date and time support to SQL. More sophisticated approaches which support multiple calendars can be found in the area of temporal databases [Sno95]. Although all these approaches and their proprietary derivatives support international types, they carry some substantive limitations. Firstly, attribute instances cannot be allotted with localised information; the property remains on attribute level. Secondly, conversion mechanisms are not visible to the user and so they are not modifiable. Thirdly, internationalisation properties are static and thus, they can neither be modified nor extended. Lastly, complex types are not supported at all.

Our research has mainly been motivated by Sciore, Siegel, and Rosenthal's [Sci94] concept of semantic values to represent semantically heterogeneous information. The system, which is based on the relational calculus, lacks the handling of complex objects (*"[...] context information needs to be attached to objects larger than single attributes; this may me be easier in object-oriented models.")* and, due to the lattice-like representation, leads to an explosion of conversion functions. The usage of an object-oriented data model has removed both remedies as shown in this paper.

[Ouk96] define context (locale information) as the semantic knowledge which is required to perform queries in heterogeneous environments. Each context is specified as a set of inter-schema correspondence assertions describing the semantic relationship between two entities, each containing conflicts about naming, abstraction and level of heterogeneity. Our approach is currently not concerned about locale building, but locale representation and interchange. [Kas96] represent context as a collection of contextual co-ordinates and values assigned to them, represented as description logic expressions. Semantic similarities are used to identify different levels of semantic proximity of two objects. Context is represented at an intensional level, as opposed to our extensional representation. The incorporation of context uncertainty in our object model is work in progress (see §8).

8 Conclusions and Further Work

We have presented an approach to the modelling of internationalised data, which is intended to be more accurate and has less overhead than more generic resolution mechanisms which deal with semantic heterogeneity. The concept of locale has been the basis of our work. This encompasses locale identity, localised equivalence specifications, locale mediation and a hierarchically organised locale ontology. We have extended the ODMG data model with the proposed concepts, including its object definition and object query language. A prototype has been described briefly and results have been evaluated.

Future work is twofold. Firstly, as described in §6, performance losses are inevitable and thus query optimisation techniques for localised semantics are required. Secondly, the concept of internationalisation reconciliation has been described as specialisation of generic context mediation. The data model and its definition and query language have been generalised to be used in the wider context of semantic heterogeneity [Büc98]. In order to model more realistic scenarios, uncertainty representation and reasoning is inevitable. Work includes fuzzy context representation [Bü98b] and similarities among context (locales) [Büc98c]. But, for

more advanced systems a holistic view, with multiple ontologies and full reasoning support, is required, which current work in progress.

Acknowledgements
We would like to thank Colin J. Miller and Louis D. Natanson from the University of Abertay Dundee for their helpful comments on an earlier draft of this paper.

9 References

[Büc98a] A.G. Büchner, D.A. Bell, J.G. Hughes. A Contextualised Object Data Model based on Semantic Values, in *Proc. 11th Int'l. Conf. on Parallel and Distributed Computing Systems*, pages 171-176, 1998.

[Büc98b] A.G. Büchner, D.A. Bell, J.G. Hughes. Context Interchange among Fuzzy Senders and Receivers, submitted for publication, 1998.

[Büc98c] A.G. Büchner, D.A. Bell, J.G. Hughes. Orders and Similarities among Semantic Values, University of Ulster, Faculty of Ulster, internal report, 1998.

[Cat97] R.G.G. Cattell, D. Barry, D. Bartels, M. Berler, J. Eastman, S. Gamerman, D. Jordan, A. Springer, H. Strickland, D. Wade (eds.). The Object Database Standard: ODMG-93 Release 2.0, Morgan Kaufmann Publishers, San Mateo, CA, 1997.

[Dat88] C.J. Date. A Proposal for Adding Date and Time Support to SQL, *ACM SIGMOD Record*, 17(2):53-76, 1988.

[Hug91] J.G. Hughes. Object-Oriented Databases, Int'l. Series in Computer Science, Prentice-Hall, New York, 1991.

[Hul97] R. Hull. Managing Semantic Heterogeneity in Databases: A Theoretical Perspective, *Proc. 16th ACM Symp. on Principles of Database Systems,* pages 51-61, 1997.

[Kas96] V. Kashyap, A. Sheth. Semantic and Schematic Similarities between Database Objects: A Context-based Approach, *The VLDB Journal*, 5(4):276-304, 1996.

[LaB93] A. LaBonté. Multi Script Ordering for Unicode, *Proc. SHARE Europe Spring Meeting, Distributed Applications*, pages. 377-399, 1993.

[Loo93] M.E.S. Loomis. The ODMG object model, *Journal of Object-Oriented Programming*, 6(3):64-69, 1993.

[Mad96] S.E. Madnick. Are we moving toward an Information SuperHighway or a Tower of Babel. The Challenge of Large-Scale Semantic Heterogeneity, keynote speech *12th Int'l. Conf. on Data Engineering*, 1996.

[Mel93] J. Melton, A.R. Simon. Internationalization Aspects of SQL-92, *Understanding the new SQL: A complete Guide*, Chapter 18, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[Ouk96] A.M. Ouksel, C.F. Naiman. Coordinating Context Building in Heterogeneous Information Systems, *Journal of Intelligent Information Systems*, 3:151-183, 1996.

[Sci94] E. Sciore, M. Siegel, A. Rosenthal. Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems, *ACM Transactions on Database Systems*, 19(2):254-290, 1994.

[Sno95] R.T. Snodgrass (ed.). TSQL2 Temporal Query Language, Kluwer Academic Publishers, 1995.

[Wie92] G. Wiederhold. Mediators in the Architecture of Future Information Systems, *IEEE Computer*, 25(3):38-49, 1992.

[Yos93] H. Yoshioka, J. Melton. Character internationalization in databases: a case study, *Digital Technical Journal*, 5(3):80-96, 1993.

Appendix: Locale and Mediator BNF

<locale_dcl> ::=
Locale <identifier> (<Type>) [: <locale>]
[Properties <locale_properties>]

<Type> ::= <atomic_types> | <complex_types> | <abstract_types>

<locale_properties> ::=
<locale_property> | <locale_property > , <locale_properties>

<locale_property> ::=
<identifier> PropertyDefault <value> PropertyConstraints <locale_set>

<mediator> ::=
Mediator <mediator_signature>
(
<mediator_body>
)

<mediator_signature> ::=
<identifier> Function | Table <locale> (<Type>) <locale> (<Type>)

<mediator_body> ::=
<mediator_function> | <mediator_table>

<mediator_function> ::= <mediator_conversion_function>

<mediator_table> ::= <identifier>