File: Headings-and-Bodies.doc
Printed at: 13:05 on Saturday, 2 November, 2013

Headings and Bodies

(some clarifications)

Hugh Darwen

At the time of writing there is an ongoing debate in the *TTM* forum in which some participants argue that there is no need to define a relation to consist of both a heading and a body, it being thus defined in the relational model of data—the body alone is sufficient, they say, and what's more, they say, there is only one empty relation. Some of the arguments put forward by the "anti-heading" body (pardon the pun) appear to have shown misunderstandings of *TTM*, especially when the Inheritance Model (IM) proposed by Date and myself is superimposed on the *TTM* type model. Those misunderstandings appear to have arisen from perceived inconsistencies between IM and *TTM* and from perceived inconsistencies between IM and the algebra **A** that is intended to form a basis for defining relational operators.

I offer clarifications in two sections, each consisting of ten numbered points. Section A assumes IM is *not* in effect. Section B shows the effect of abandoning that assumption. My idea is that those who advocate a "headingless" model might like to provide ten-point counterparts in similar vein.

A. Under the *TTM* assumption: a value is of exactly one type

1. A **heading** is a set of <attribute name, type name> pairs, called **attributes,** such that if <*an*, *tn1*> is an element of heading *H* and <*an*, *tn2*> is an element of *H*, then *tn1* = *tn2*[1] and so <*an*, *tn1*> and <*an*, *tn2*> are in fact the same element.
2. An **tuple component** is a pair <*a,v*> where *a* is an attribute <*an, tn*> and *v*—the **attribute value**—is a value of type *tn*. *Note:* The term tuple component does not appear in *TTM,* which has no term for this construct—see RM Pre 9. The *TTM* definition of such a component, <*A, T, v*> in RM Pre 9, is equivalent to the one given here.
3. A **tuple** is a set of tuple components such that if <*a*, *v1*> is an element of tuple *t* and <*a*, *v2*> is an element of *t*, then *v1* = *v2* and so <*a*, *v1*> and <*a*, *v2*> are in fact the same element.
4. A tuple *t* has a heading, this being the set of attributes obtained by discarding the value component *v* from each element <*a*,*v*> of *t.*
5. Tuples of the same heading are of the same type, a **tuple type** (RM Pre 6). A name for tuple type *TT* specifies (a) that *TT* is a tuple type, and (b) a heading, defining the set of tuples constituting *TT.*

[1] It is assumed that a type has exactly one name. In practice there maybe more than one way of writing that name, in which case we can appeal to some canonical form as the assumed single name.

6. A body is a set of tuples of the same type.

7. A relation *r* has a heading *Hr* and a body *Br* such that each tuple in *Br* is of heading *Hr*. If relations *r1* and *r2* have the same heading and the same body, then *r1 = r2*; conversely, *r1* and *r2* are distinct if they differ in either heading or body.

8. Relations of the same heading are of the same type, a **relation type** (RM Pre 7). A name for relation type *RT* specifies (a) that *RT* is a relation type, and (b) a heading, defining the set of relations constituting *RT*.

   Note: Hr is implied by *Br* if and only if *Br* is nonempty. For that reason, the definition of a relational operator must be couched in terms of both headings and bodies of its input and output relations.

9. In **D**, a **tuple expression** is an invocation of some operator[2] whose invocations denote tuples. Tuple expression *tx* might reference one or more variables, in which case it possibly denotes different tuples at different times. The **declared type** of *tx* is the type of the tuple it denotes regardless of the time of evaluation.

10. In **D**, a **relation expression** is an invocation of some operator[3] whose invocations denote relations. Relation expression *rx* might reference one or more variables, in which case it possibly denotes different relations at different times. The **declared type** of *rx* is the type of the relation it denotes regardless of the time of evaluation.

B. Under IM, allowing a value to be of more than one type

Under IM a value can be of more than one type but is of exactly one most specific type *MST*, such that all of its types are supertypes of *MST*. Thus, IM overrides *TTM*'s RM Pre 1 where that prescription states that distinct type names identify disjoint value sets. As a consequence, IM also requires RM Pre 9 and RM Pre 10 to be reinterpreted in the light of that overriding of RM Pre 1, to permit a tuple or relation to be of more than one type.

In the points that follow, text that differs from its counterpart in Section A is shown in blue.

1. A **heading** is a set of <attribute name, type name> pairs, called **attributes,** such that if <*an*, *tn1*> is an element of heading *H* and <*an*, *tn2*> is an element of *H*, then *tn1 = tn2* and so <*an*, *tn1*> and <*an*, *tn2*> are in fact the same element.

2. An **tuple component** is a pair <*a,v*> where *a* is an attribute <*an, tn*> and *v*—the **attribute value**—is a value of type *tn*. *Note:* The term tuple component does not appear in *TTM,* which has no term for this construct—see RM Pre 9. The *TTM* definition of such a component, <*A, T, v*> in RM Pre 9, is equivalent to the one given here.

---

[2] Including in particular tuple selectors such as **Tutorial D**'s TUPLE{*acl*} where *acl* is a commalist of <attribute name, expression> pairs.

[3] Including in particular relation selectors such as **Tutorial D**'s RELATION{*H*}{*tcl*} where *H* is a commalist denoting a heading and *tcl* is a commalist of tuple expressions ({*H*} may be omitted if *tcl* is nonempty).

3. A **tuple** is a set of tuple components such that if <*a*, *v1*> is an element of tuple *t* and <*a*, *v2*> is an element of *t*, then *v1* = *v2* and so <*a*, *v1*> and <*a*, *v2*> are in fact the same element.

4. A tuple *t* has a heading, this being the set of attributes obtained by discarding the value component *v* from each element <*a*,*v*> of *t*.

5. Tuples of the same heading are of the same type, a **tuple type** (RM Pre 6). A name for tuple type *TT* specifies (a) that *TT* is a tuple type, and (b) a heading, defining the set of tuples constituting *TT*. *Note:* Some elements of *TT* might be of some proper subtype of *TT* (IM Pre 22) The heading of a tuple is that of its most specific type.

6. A body is a set *S* of tuples, each of some type *TT* such that *TT* is a common supertype of the types of the tuples of *S* (IM Pre 23).

7. A relation *r* has a heading *Hr* and a body *Br* such that if tuple *t* is a member of *Br*, then the type *Tt* of *t* is of some subtype of the tuple type of heading *Hr*. Moreover, if *Br* is empty, then *Hr* is the heading of some minimal relation type (IM Pre 24); otherwise, *Hr* is the most specific common supertype of the types of the tuples of *Br*. If relations *r1* and *r2* have the same heading and the same body, then *r1* = *r2*; conversely, *r1* and *r2* are distinct if they differ in either heading or body.

8. Relations of the same heading are of the same type, a **relation type** (RM Pre 7). A name for relation type *RT* specifies (a) that *RT* is a relation type, and (b) a heading, defining the set of relations constituting *RT*. *Note:* Some elements of *RT* might be of some proper subtype of *RT* (IM Pre 22).

   Note: Hr is implied by *Br* if and only if *Br* is nonempty. For that reason, the definition of a relational operator must be couched in terms of both headings and bodies of its input and output relations.

9. In **D**, a **tuple expression** is an invocation of some operator whose invocations denote tuples. Tuple expression *tx* might reference one or more variables, in which case it possibly denotes different tuples at different times. The **declared type** of *tx* is some supertype of the type of the tuple it denotes regardless of the time of evaluation.

10. In **D**, a **relation expression** is an invocation of some operator whose invocations denote relations. Relation expression *rx* might reference one or more variables, in which case it possibly denotes different relations at different times. The **declared type** of *rx* is some supertype of the type of the relation it denotes regardless of the time of evaluation.

End