# Anchor Modeling: Naming Convention

L. Rönnbäck
lars.ronnback@resight.se
Resight, Kungsholms Strand 179, 112 48 Stockholm

O. Regardt
olle.regardt@teracom.se
Teracom, Kaknästornet, 102 52 Stockholm

M. Bergholtz, P. Johannesson, P. Wohed
maria@dsv.su.se, pajo@dsv.su.se, petia@dsv.su.se
DSV, Stockholm University, Sweden

September 15, 2010

Entities in an anchor schema can be named in any manner, but having a naming convention outlining how names should be constructed increases understandability and simplifies working with a model. If the same convention is used consistently over many installations the induced familiarity will significantly speed up recognition. A good naming convention should fulfill a number of criteria, some of which may conflict with others. Names should be short, but long enough to be intelligible. They should be unique, but have parts in common for related entities. They should be unambiguous, without too many rules having to be introduced. Furthermore, in order to be used in many different representations, the less "exotic" characters they contain the better. The suggested conventions use only regular letters, numbers, and the underscore character. This ensures that the same names can be used in a variety of representations, such as in a relational database, in XML, or in a programming language.

The conventions suggested in Figure 1 are described using a variant of EBNF (Extended Backus-Naur Form). This variant allows regular expressions to be used in the production rules, such as allowed ranges and different specifications of repetitions, while keeping the comma sign for concatenation. An example of an allowed range can be seen in the production rule for the primitive 'lower' with the regular expression: [a-z], allowing all lower case letters. Examples of repetitions can be seen in the production rule for the primitive 'lwords' with the regular expression: (lower)+, (uword)?, where the primitive 'lower' must occur one or more times and may be followed by the primitive 'uword' once. Another example of a repetition can be seen in the production rule for the mnemonic

| Parts | |
|---|---|
| $N_{\mathbb{T}}(T)$ | ::= $T$, delim, historization |
| $N_{\mathbb{I}K}(T)$ | ::= M$K$, delim, identity, delim, $R_K$ |
| $N_{\mathbb{I}A}(T)$ | ::= M$A$, delim, identity, delim, $R_A$ |
| $N_{\mathbb{T}}(B)$ | ::= M$B$, delim, historization |
| $N_{\mathbb{D}}(B)$ | ::= $B$ |
| $N_{\mathbb{I}K}(B)$ | ::= M$K$, delim, identity |
| $N_{\mathbb{I}A}(B)$ | ::= M$A$, delim, identity |
| $N_{\mathbb{D}}(K)$ | ::= $K$ |
| $N_{\mathbb{I}}(K)$ | ::= M$K$, delim, identity |
| $N_{\mathbb{I}}(A)$ | ::= M$A$, delim, identity |
| **Entities** | |
| $T$ | ::= M$A$, delim, $R_A$, (delim, M$A$, delim, $R_A$)+, (delim, M$K$, delim, $R_K$)* |
| $B$ | ::= M$A$, delim, M$B$, delim, D$A$, delim, D$B$ |
| $K$ | ::= M$K$, delim, D$K$ |
| $A$ | ::= M$A$, delim, D$A$ |
| **Descriptors** | |
| D$B$ | ::= uwords |
| D$K$ | ::= uwords |
| D$A$ | ::= uwords |
| **Mnemonics** | |
| M$B$ | ::= (upper \| number){3} |
| M$K$ | ::= (upper \| number){3} |
| M$A$ | ::= (upper \| number){2} |
| **Roles** | |
| $R_K$ | ::= lwords |
| $R_A$ | ::= lwords |
| **Primitives** | |
| historization | ::= 'ValidFrom' |
| identity | ::= 'ID' |
| delim | ::= '_' |
| lwords | ::= (lower)+, (uword)? |
| uwords | ::= (upper, (lower \| number)*)+ |
| number | ::= [0-9] |
| upper | ::= [A-Z] |
| lower | ::= [a-z] |

Figure 1: A suggested naming convention described in Extended Backus-Naur Form with regular expressions.

'M*A*' with the regular expression: (upper | number){2}, where exactly two times either of the primitives 'upper' or 'number' occur.

EBNF can only describe syntax. However, the names derived from the suggested namning conventions also contain semantics. From the names it should be possible to deduce the relationships between entities. The procedure to get names with such semantics is simple. The EBNF in Figure 1 should be followed bottom-up and every production rule exhausted before moving up to the next. If a production rule contains a reference to a previous production rule, a string among those already created in the referenced rule must be selected. For example, when the production rule for the mnemonic 'M*A*' is exhausted, the result is a set of mnemonics (two upper case letters or numbers) for anchors. In the anchor model in Figure 3, there are four anchors, yielding the set of mnemonics {'AC', 'PE', 'ST', 'PR'}. The production rule for the name of an attribute '*B*' is 'M*A*, delim, M*B*, delim, D*A*, delim, D*B*', which starts with a reference to 'M*A*', allowing only one of the four strings in the set to be used. Which of the four is selected is determined by the actual relationships, such that an attribute takes the mnemonic of the anchor it belongs to. Given the example, exhausting the production rule for '*B*' and the anchor having the mnemonic 'AC' will produce the set {'AC_NAM_*Actor_Name*', 'AC_PLV_*Actor_ProfessionalLevel*', 'AC_GEN_*Actor_Gender*'}. From these attribute names it is easy to determine which anchor they belong to.

While mnemonics are short unique strings, names also contain descriptors, which are longer strings intelligible enough to give an understanding of what entities represent. For example, the anchor named 'AC_*Actor*' has the mnemonic 'AC' and the descriptor '*Actor*'. One attribute of this anchor is named 'AC_NAM_*Actor_Name*', starting with the mnemonic of the anchor to which it belongs, followed by a mnemonic for the attribute, 'NAM', and the descriptors for both the anchor and the attribute. In the tie named 'PE_*in*_AC_*wasCast*' the mnemonics of the adjoining anchors, 'AC' and 'PE', are present together with the names of the roles they have in the tie. These conventions will yield unique, but not unambiguous names, i.e. the order of the adjoined anchors and knots in a tie may change. For example the two names 'PE_*in*_AC_*wasCast*' and 'AC_*wasCast*_PE_*in*' both refer to the same tie.

Parts are named according to the functions described in Figure 2. The conventions for parts in Figure 1 also contain references to earlier production rules, giving those names semantics as well. The production rule naming the anchor identity for an attribute is 'M*A*, delim, identity'. From the set of available anchor mnemonics, the one for the anchor that is the domain of the attribute should be chosen. For example, in the attribute named 'AC_NAM_*Actor_Name*' the identity part is named 'AC_ID', the data type part is named 'AC_NAM_*Actor_Name*', and the time type part is named 'AC_NAM_ValidFrom'. Likewise the anchor that is the type of an anchor role in a tie should be chosen.

| Function | Naming part | Input |
|---|---|---|
| $N_{\mathbb{T}}(T)$ | time type | tie |
| $N_{\mathbb{I}_K}(T)$ | knot identity | tie |
| $N_{\mathbb{I}_A}(T)$ | anchor identity | tie |
| $N_{\mathbb{T}}(B)$ | time type | attribute |
| $N_{\mathbb{D}}(B)$ | data type | attribute |
| $N_{\mathbb{I}_K}(B)$ | knot identity | attribute |
| $N_{\mathbb{I}_A}(B)$ | anchor identity | attribute |
| $N_{\mathbb{D}}(K)$ | data type | knot |
| $N_{\mathbb{I}}(K)$ | identity | knot |
| $N_{\mathbb{I}}(A)$ | identity | anchor |

Figure 2: Functions naming parts for anchors, knots, attributes, and ties.

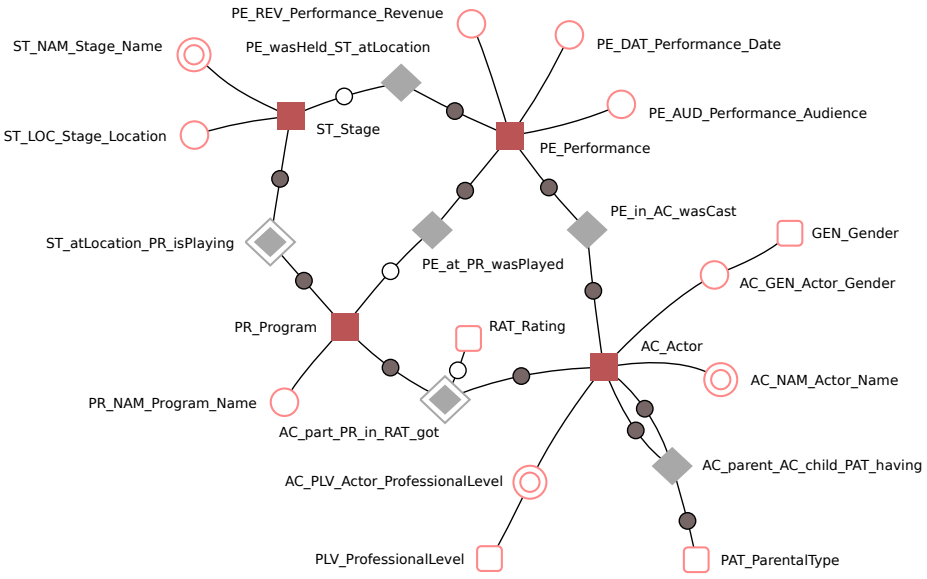


Figure 3: An anchor model