Reference Card for ERD Notation

Gebhard Greiter, 2011

ERD Design Notation, see http://greiterweb.de/spw/ERDNotation.htm, is a formal language

- **to specify Entity Relationship Data Structures of the most general type**,

i.e. also those in which

- Relationships may have attributes, and
- Relationships may be n-ary (i.e. may not be just binary relationships).

What you describe are:

- Entity types (= entity name, and a sequence of typed entity attributes), and
- Domain types.

A typed attribute in this sense is

- an attribute with values in a specified Domain, or
- an attribute with values in a specified Entity type (generating relationships).

In order to avoid redundancy, you can – for abbreviation –

- use structures (i.e. named sets of domain-valued attributes), and
- use subclassing (i.e. make an entity type a set of attributes which is a subset of another, more specific entity type)

Whenever you want to refer to a specific domain, entity type, attribute, or set of attributes, you give its ***typed name***, i.e. its name prefixed by

- **D_** for **Domain**
- **E_ e_** for **Entity Type** respectively **Entity Type seen as a Set of Attributes**
- **A_** for **Attribute**
- **S_** for **Set of Attributes**

The code generator (e.g. **er1.exe**) takes for input an ASCII file *.er1 – or *.er1.TXT – but will see only declarations starting with a minus sign in their first column followed by exactly two spaces):

```
-  ec        E_name
-  eca       A_           e_name
-  eca       A_name       D_name
-  eca       A_name       S_name
-  d         D_name       SQL_type or name_of_class_representing_values
```

Instead of

- **eca**

you are to write

- **eca,pk**

if these attribute (or set of attributes) is part of the primary key of the entity in question. You are to write

- **eca,k***n*

n a positive integer, if the attribute (or set of attributes) is part of a secondary key called k*n*. Because keys (seen as sets of attributes) may overlap, you can have lines such as, e.g.

- **eca,pk,k***2*,**k***5*

If an attribute must not be allowed to have NULL values, say so by saying

- **eca,nn**

in the line specifying a name for this attribute (or set of attributes).

If you write

- **ec,ts**

instead of

- **ec**

the table implementing this type in SQL will have, as a kind of change log, additional attributes

- **eca A_created D_DATETIME**
- **eca A_updated D_DATETIME**
- **eca A_deleted D_DATETIME**

Code generated in C++, C# or Java is to ensure that their values are maintained automatically (if A_deleted is not NULL, an application may not be interested in such data though it might be kept to be seen by some other application).

Together with this Reference Card you may have received a code generator that will at least be able to transform ERD Notation to SQL scripts generating a corresponding physical implementation of the data model specified.

Default for such a generator is **er1.exe** (a command line utility available for free here: www.greiterweb.de/spw/useMe/er1.zip).

Code Generator

er1.exe

ERD Design Notation is a formal specification language and is, as such, capable of being transformed automatically to executable code (at least SQL).

One compiler to support this is **er1.exe** – a simple command line utility. You call it in the form

```
er1 - nnn
```

where `nnn.er1` or `nnn.er1.TXT` is the path to a TXT file containing a conceptual data model specified in ERD Notation (and possible well documented by free text between the formal sections).

Such a call will, as soon as it does no longer detect syntactical errors in your specification, create or update a folder `./xxfs` containing at least the following files:

- `nnn.sql` = Implementation of the data model in SQL
- `nnn.sql.htm` = nnn.sql presented in HTML
- `nnn.er1.htm` = the corresponding Entity Relationship Diagram (ERD)

If called in the form

```
er1 + nnn
```

additional code created automatically can then be found in `./xxfs` :

- `nnn_cs` = Implementation of the Data Layer in CSharp
- `nnn_cpp` = Implementation of the Data Layer in C++
- `nnn_java` = Implementation of the Data Layer in Java

A **Data Layer** in this sense is the implementation of a data storage & retrieval component coded on top of the SQL interface to a physical database of type `nnn.sql` – the code implementing this data layer is a set of classes that may or may not be based on LINQ, the Java Persistency API, or similar OR mappers (or none at all if the implementation language is C++).

In order to let these classes also have more dedicated data retrieval methods, you can specify class members `a_name` each of them meant to be a method returning the result of some specific SELECT statement.

The conceptual ERD in `nnn.er1.htm` will show these methods in the form

E_*name1* ->> **set of E_***name2* **a_***name3*

where `E_name1.a_name3()` is the method to retrieve a set of entities of type `E_name2` via the SELECT statement for `a_name3` in `E_name1` .

The following examples declare selection methods supported by **er1.exe** (similar code generators you might create yourself as far as needed may support more, less, or a slightly different syntax):

Example based on the Data Model in Annex 1:

```
-  for          Person x
   to           MoviesDirected
   select       Movie  m
   where        (
                     m.DirectorFirstName = x.FirstName
                AND  m.DirectorLastName  = x.LastName
                )
```

Examples based on the Data Model in Annex 2:

```
-  for          E_Person.A_Betreuer
   select       E_Person
   to           a_betreutePersonen


-  for          E_Ehe.A_Ehemann
   select       E_Ehe.A_Ehefrau
   to           a_Ehefrauen



-  for          Person x
   to           Kinder
   select       Person k
   where        (
                x = k.Mutter  OR  x = k.Vater
                )


-  for          Person x
   to           Ehepartner
   select       Person p via Ehe e
   where        ( x = e.er AND p = e.sie ) OR ( x = e.sie AND p = e.er )
```

Please note: The names following **via**, if any, must be E_ names, the name directly following **select** however can also be an S_ name.

Annex 1:

A small Sample (Movies.er1)

```
-  ec             E_Movie
                  |
-  eca,pk         A_Title             D_Title
-  eca            A_Director          E_Person


-  ec             E_Person
                  |
-  eca,pk         A_FirstName         D_Name
-  eca,pk         A_LastName          D_Name
-  eca,nn         A_Gender            D_Gender


-  d              D_Title             CHAR(80)
-  d              D_Name              CHAR(20)


-  d              D_Gender            INTEGER

                  Valid values are:

-  v              .  male
-  v              .  female
```

A slightly less trivial example – showing also the generated SQL and the ERD – is the following:

Annex 2:

Another Sample (Persons.er1)

```
-  e              E_Person
                  |
-  eca,pk         A_Vorname                 D_Name
-  eca,pk         A_Nachname                D_Name
-  eca,pk         A_GeborenAm               D_Date
-  eca            A_Adresse                 S_Adresse
-  eca            A_Vater                   E_Mann
-  eca,nn         A_Mutter                  E_Frau
-  eca            A_Betreuer                E_Person


-  ec             E_Mann
                  |
-  eca,pk         A_                        e_Person


-  ec             E_Frau
                  |
-  eca,pk         A_                        e_Person
-  eca            A_Schwangerschaften       D_Anzahl


-  e              E_Ehe
                  |
-  eca,pk         A_er_                     E_Mann
-  eca,pk         A_sie_                    E_Frau
-  eca,pk         A_VerheiratetVon          D_Date
-  eca            A_VerheiratetBis          D_Date


-  s              S_Adresse
                  |
-  eca            A_                        S_Ort
-  eca            A_Strasse                 D_Name
-  eca            A_Hausnummer              D_HausNr


-  s              S_Ort
                  |
-  eca            A_Land                    D_Name
-  eca            A_Gemeinde                D_Name


-  d              D_Name                    CHAR(30)
-  d              D_Date                    DATE
-  d              D_Anzahl                  INTEGER
-  d              D_HausNr                  CHAR(8)
```

The ERD specified by this code (and the SQL created by **er1.exe**) can be looked up in the files

www.greiterweb.de/spw/zu_Persons/o.persons.ERD.htm and

www.greiterweb.de/spw/zu_Persons/o.persons.SQL.htm