

The **tkz-orm** package Object-Role Model Drawing Library

Jakob Voß*, Camil Staps†

Version 0.1.4
January 15, 2016

<http://purl.org/net/tkz-orm>

Abstract

This package provides styles for drawing Object-Role Model (ORM2) diagrams in \TeX based on the PGF and *TikZ* picture environment.

Contents

1	Introduction	2
2	Object Types	3
3	Predicates and Roles	4
4	Constraints	6
4.1	Uniqueness Constraints	7
4.2	Mandatory Role Constraints	9
4.3	External constraints	10
4.4	Ring Constraints	11
4.5	Number and Value Constraints	12
4.6	Textual constraints	13
5	Subtyping	13
6	Additional Features	15
6.1	Duplicated and implied parts of a model	15
6.2	Arrow heads	17
6.3	Macros for text layout	17
7	Settings and Utilities	18
	References and Index	18

*jakob.voss@gbv.de

†info@camilstaps.nl

1 Introduction

tkz-orm is intended to help you creating Object-Role Model (ORM) diagrams. It is based on the PGF and TikZ¹ picture macro package for T_EX. and provides additional styles and commands to typeset ORM2 diagrams. With tkz-orm you can “program” ORM diagrams just as you “program” your document when you use L^AT_EX – including the inherent lack of WYSIWYG. Unless multi-touch e-paper interfaces become usable, tkz-orm can best be combined with a whiteboard or paper and pencil — but you may also find ways to automatically create ORM diagrams with tkz-orm.

Status of this package

This is the developer version of tkz-orm. Please send your comments to the author so the package can be improved. All parts of the package are available at least under the L^AT_EX Project Public License[LPP08] and the GNU Public license[GPL91]. For details have a look at the file LICENSE that is part of this package. The permanent URL of tkz-orm is <http://purl.org/net/tkz-orm> which redirects you to its current location and a collection of examples.

ORM in a nutshell

Object-Role Modeling (ORM)² is a fact-oriented modeling language that evolved from the *Natural-language Information Analysis Method* (NIAM) by G.M. Nijssen. The current version (ORM2) is mainly based on works of Terry Halpin. Like ERM, UML, and other data modeling languages ORM helps to identify and abstract information objects, relationships, and rules of a Universe of Discourse to be formalized and implemented on another level. ORM includes a graphical notation and a precise verbalization in natural language. Models can further be validated by populating fact tables with sample data. An overview of the ORM2 graphical notation is given in [Hal05] and more details in [HM08]. An ORM model consists of object types (section 2) and predicates (section 3). Each of the n roles of an n -ary predicate is connected to an object type that plays the specific role in this predicate. Furthermore a model can contain constraints (section 4), subtypes (section 5), and other features (section 6). tkz-orm also allows you to change the appearance of ORM diagrams (section 7).



Figure 1: Examples of object types in ORM

¹Available at <http://sourceforge.net/projects/pgf/>.

This document was created with TikZ version 3.0.1a

²See <http://www.orm.net/>.

2 Object Types

Object types are drawn as rectangles with rounded borders. The object's type name is written as node text inside. *Entity types* use solid border lines and *value types* use dashed border lines. The minimal size of an object is set to 6mm×6mm. This package provides the following styles for entities and values:

/tikz/entity

This style is to be used with nodes that represent entity types.



```
\begin{tikzpicture}
\node[entity] at (0,0) {Foo};
\node[entity] (unnamed) at (1.2,0) {};
\node[entity] at (2.5,0) {Person\\(.name)};
\end{tikzpicture}
```

/tikz/value

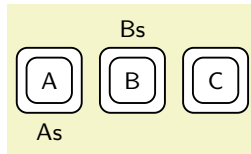
This style is to be used with nodes that represent value types.



```
\begin{tikzpicture}
\node[value] {Name};
\end{tikzpicture}
```

/tikz/power

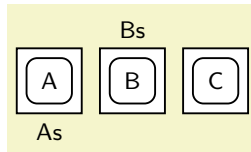
This style is to be used with nodes that represent power types.



```
\begin{tikzpicture}
\node[entity,power=below:As] {A};
\node[entity,power=Bs] at (1.1,0) {B};
\node[entity,power] at (2.2,0) {C};
\end{tikzpicture}
```

/tikz/sequence

This style is to be used with nodes that represent sequence types.



```
\begin{tikzpicture}
\node[entity,sequence=below:As] {A};
\node[entity,sequence=Bs] at (1.1,0) {B};
\node[entity,sequence] at (2.2,0) {C};
\end{tikzpicture}
```

/tikz/every entity

/tikz/every value

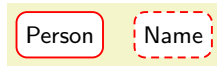
Each of these styles is invoked by the styles `entity` or `value`. Change one of these styles to change the appearance of entity or value types.



```
\begin{tikzpicture}[
every entity/.style={draw=blue!50,fill=blue!20},
every value/.style={draw=green!50,fill=green!20}]
\node[entity] (P) at (0,0) {Person};
\node[value] (N) at (1.5,0) {Name};
\end{tikzpicture}
```

`/tikz/every object`

This style is invoked by the styles `entity` and `value`. Change this style to change the common appearance of entity and value types.



```
\begin{tikzpicture}[
  every object/.style={shape=rectangle,draw=red}]
  \node[entity] (P) at (0,0) {Person};
  \node[value] (N) at (1.5,0) {Name};
\end{tikzpicture}
```

Since entity types and value types are very frequent node types in an ORM diagram, there are two special abbreviations for creating object types:

`\entity`

Inside `{tikzpicture}` this is an abbreviation for `\node[entity]`.

`\value`

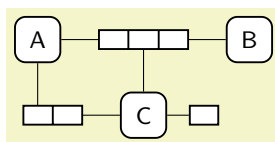
Inside `{tikzpicture}` this is an abbreviation for `\node[value]`.

3 Predicates and Roles

Relationship parts (*roles*) played by objects are shown as boxes of fixed size (4mm×2.5mm). A *predicate* is a sequence of one or more concatenated role boxes. Predicates can be created with the following styles:

`/tikz/roles=number of roles`
`/tikz/role=1`

Shapes the current node as predicate with a given number of role boxes. Numbers from 1 to 20 are supported. The default value is 2 (binary).



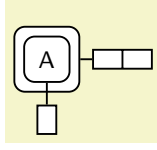
```
\begin{tikzpicture}[orm]
  \entity (A) at (0,1) {A};
  \entity (B) at (2.8,1) {B};
  edge node[roles=3] (p) {} (A);
  \entity (C) at (1.4,0) {C} edge (p.south);
  \node[role] at (2.2,0){} edge (C);
  \draw (A) |- node[roles,xshift=2mm]{} (C);
\end{tikzpicture}
```

`/tikz/vroles=number of roles`
`/tikz/vrole=1`

Shapes the current node as predicate rotated by 90 degree (vertical).

`/tikz/relation`
`/tikz/relationship`
`/tikz/plays`

These equivalent styles are to be used with connection lines between objects and roles. By default it just includes the style `every orm line` which results in a solid, black line of 0.25mm width.



```
\begin{tikzpicture}
  \node[entity,power] (A) {A};
  \node[roles] (r) at (1,0) {};
  \draw[relation] (A-power) -- (r);
  \node[vrole] at (0,-.8){} edge[relation] (A);
\end{tikzpicture}
```

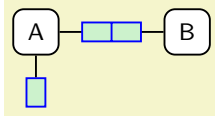
The following table lists abbreviations that can be used for creating predicate nodes and relationship lines inside `{tikzpicture}`:

command(s)	abbreviation for	
<code>\unary</code> or <code>\role</code>	<code>\node[role]</code>	
<code>\binary</code> or <code>\roles</code>	<code>\node[roles]</code>	
<code>\ternary</code>	<code>\node[roles=3]</code>	
<code>\vunary</code> or <code>\vrole</code>	<code>\node[vrole]</code>	
<code>\vbinary</code> or <code>\vroles</code>	<code>\node[vroles]</code>	
<code>\vternary</code>	<code>\node[vroles=3]</code>	
<code>\plays</code>	<code>\draw[relationship]</code>	

The general style of predicates and roles can be modified by the following keys:

`/tikz/every predicate`

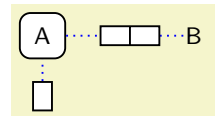
Changing this style to modify the common appearance of predicates.



```
\begin{tikzpicture}[
  every predicate/.style={draw=blue,fill=green!20}
]
  \entity at (2,0) (B) {B};
  \entity (A) {A} edge[relation] node[roles]{} (B);
  \unary at (0,-0.8) (r) {} edge[relation] (A);
\end{tikzpicture}
```

`/tikz/every relationship`

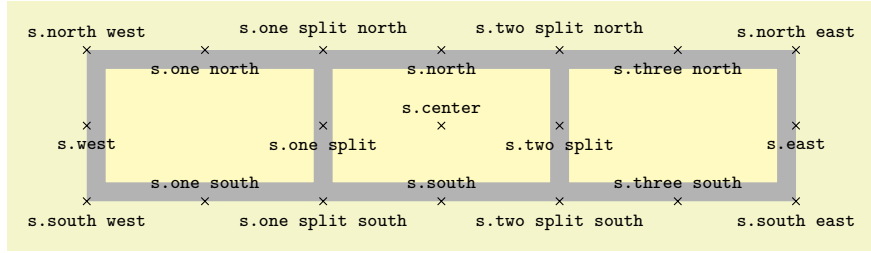
This style is invoked by the style `relationship`. To change the appearance of explicit relationship lines you can change this style. Please keep in mind that nodes placed on a line by `node` in one operation inherit properties from the line they refer to, so you should create relationship lines with `\plays`.



```
\begin{tikzpicture}[orm,
  every relationship/.style={draw=blue,dotted}
]
  \entity (A) {A};
  \plays (A) to node[roles]{} (2,0) node(B){B};
  \unary at (0,-0.8) (r) {} edge[relation] (A);
\end{tikzpicture}
```

Predicates are drawn either horizontally (`roles`) or vertically (`vroles`) as nodes with one or more parts. Figure 2 shows some of the anchors. Vertical predicates are rotated by 90 degree so `north` is at the left, `west` is at the bottom etc.

The verbalization of a predicate can be given as `label` next to a predicate. For binary relationships forward and inverse readings can be separated by a slash. To show the inverse reading, add an arrow tip with the commands `\ormleft` or `\ormup`. Labels for predicates with more than two roles must contain three dots (`\ldots`) for each inner role. Role names and indices can be added by different styles.

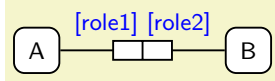


```
\Large
\begin{tikzpicture}
\node[roles=3,shape example,inner ysep=0.75cm] (s) {};
\foreach \anchor/\placement in
{one north/below, one south/above,
three north/below, three south/above,
one split/below, one split north/above, one split south/below,
two split/below, two split north/above, two split south/below,
north/below, south/above, east/below, west/below, center/above,
north west/above, north east/above, south west/below, south east/below}
\draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
node[\placement] {\scriptsize\texttt{s.\anchor}};
\end{tikzpicture}
```

Figure 2: Node anchors of an ORM predicate

/tikz/role name

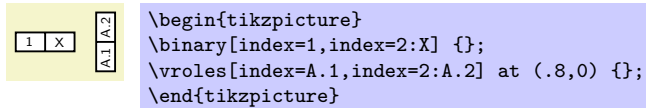
This style is to be used with role names. Role names can be displayed in square brackets and blue color next to a role box.



```
\begin{tikzpicture}[orm]
\entity (A) at (0,0) {A};
\entity (B) at (2.8,0) {B};
\plays (A) edge node(r)[roles]{} (B);
\node[role name,
at=(r.north),anchor=south east] {[role1]};
\node[role name,
at=(r.north),anchor=south west] {[role2]};
\end{tikzpicture}
```

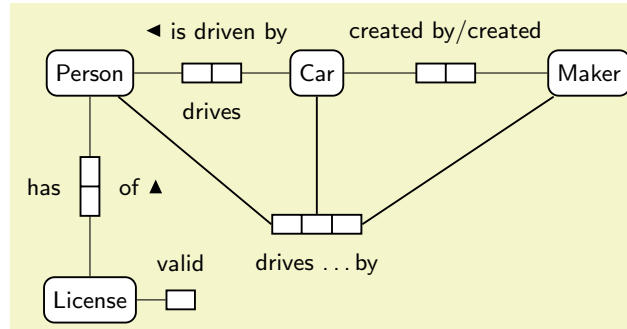
/tikz/index=<n>:<index>

Adds a role index as small label at the n th role box (default: n).



4 Constraints

ORM provides a rich set of constraints: Mandatory constraints (section 4.2) and uniqueness constraints (section 4.1) limit the way objects can be combined



```

\begin{tikzpicture}[orm]
  \entity at (0,3) (P) {Person};
  \entity at (3,3) (C) {Car};
  \entity at (0,0) (L) {License};
  \entity at (6.6,3) (M) {Maker};
  \unary[label=valid] at (1.2,0) (V) {} edge (L);
  \draw (P) to node[roles,
    label=below:drives,label=\ormleft{is driven by}]{} (C);
  \draw (P) to node[vroles,label=has,label=below:\ormup{of}]{} (L);
  \draw (C) to node[roles,label=created by/created]{} (M);
  \ternary[label=below:drives \ldots by] at (3,1) (t) {};
  \plays (P) -- (t.west);
  \plays (C) -- (t);
  \plays (M) -- (t.east);
\end{tikzpicture}

```

in predicates. External constraints (section 4.3) and subtype constraints (section 5) involve multiple roles or object types. All constraints are displayed in magenta and either drawn directly at an object type or role, or linked to one or more object types or role with dotted or dashed lines or arrows (see the styles `limits` and `limits to`). ORM2 defines a set of symbols for external (section 4.3), ring (section 4.4) and other types of constraints. The general TKZ-ORM constraint key `constraint` only sets the font to violet. An optional key value can be used to add a predefined constraint symbol at the current position.

`/tikz/constraintcolor=<color>`

Changes the constraint color (default: `magenta!100`).

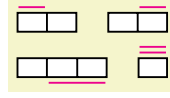
4.1 Uniqueness Constraints

By default every row in a fact table is unique. To express additional uniqueness constraints on one or more roles of a fact table or to explicitly express the uniqueness on the full predicate, a *uniqueness bar* is drawn above or below the fact roles. If the bar spans two or more non-adjacent roles, it is drawn as dotted line above or below the excluded roles. Bars can be stacked in multiple levels. To draw uniqueness bars you can use the following styles at predicate nodes:

`/tikz/unique=<from>-<to>:<level>`

Draws a uniqueness constraint bar above one or more roles. All parts of the key value are optional. As default a simple uniqueness bar above (`<level>=1`)

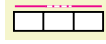
the first role ($\langle from \rangle=1$) is drawn. To make a bar span multiple roles, use the $\langle from \rangle-\langle to \rangle$ syntax. Negative levels drawn the bar below the roles.



```
\begin{tikzpicture}
\binary[unique] at (0,0) {};
\binary[unique=2] at (1.2,0) {};
\ternary[unique=2-3:-1] at (0.2,-0.6) {};
\unary[unique=1,unique=1:2] at (1.4,-0.6) {};
\end{tikzpicture}
```

/tikz/skip unique= $\langle from \rangle-\langle to \rangle:\langle level \rangle$

Draws a dotted uniqueness constraint bar. The syntax is the same as at the **unique** key. The bar includes background color in the gaps between dots, so it can be drawn on top of another bar.

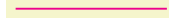


```
\begin{tikzpicture}
\ternary[unique=1-3,skip unique=2] {};
\end{tikzpicture}
```

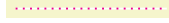
/tikz/uniqueness bar

/tikz/skipped uniqueness bar

This styles can be used to draw a line in the same style as a uniqueness constraint bar or a dotted uniqueness constraint bar.

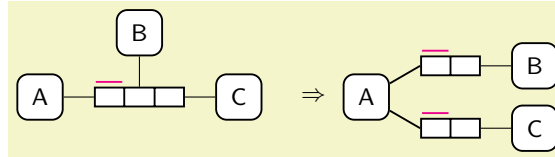


```
\tikz\draw[uniqueness bar] (0,0) -- (2,0);
```



```
\tikz\draw[skipped uniqueness bar] (0,0) -- (2,0);
```

Please note that elementary n -ary predicates should only have uniqueness constraints of at least $n - 1$ roles. Picture 3 shows how to split a ternary predicate with unique constraint bar on one role.



```
\begin{tikzpicture}[orm] % needs positioning library
\ternary[unique] (t) at (0,0) {};
\entity[left=of t] {A} edge (t);
\entity[above=of t] {B} edge (t);
\entity[right=of t] {C} edge (t);
\node at (2.3,0) {$\Rightarrow$};
\entity (A) at (3,0) {A};
\binary[right=of A.north east,yshift=1mm,unique] (t1) {};
\binary[right=of A.south east,yshift=-1mm,unique] (t2) {};
\plays (A) -- (t1.west); \plays (A) -- (t2.west);
\entity[right=of t1] {B} edge (t1);
\entity[right=of t2] {C} edge (t2);
\end{tikzpicture}
```

Figure 3: A ternary predicate can be split into to binary predicates

4.2 Mandatory Role Constraints

To indicate explicitly that a role is mandatory, a mandatory role dot is added to either end of the line that connects the role to its object. Usually it is placed at the object type end. This package defines the style key `constraint dot` (alias `cdot`) and the following keys which can be used to add mandatory role dots to lines drawn with the `to` operation.

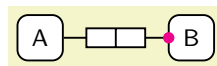
`/tikz/constraint dot`
`/tikz/cdot`

Draws the current node as mandatory role dot.

● `\tikz \node[cdot] {};`

`/tikz/mandatory`
`/tikz/required`

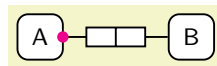
This styles enables the `relationship` style and adds a mandatory role dot at the start of a straight line.



```
\begin{tikzpicture}
\entity (A) {A};
\entity at (2,0) {B} edge[mandatory]
node[roles] (p) {} (A);
\end{tikzpicture}
```

`/tikz/required by`

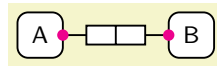
This styles enables the `relationship` style and adds a mandatory role dot at the end of a straight line.



```
\begin{tikzpicture}
\entity (A) {A};
\entity at (2,0) {B} edge[required by]
node[roles] (p) {} (A);
\end{tikzpicture}
```

`/tikz/both required`
`/tikz/both mandatory`

This styles enables the `relationship` style and adds mandatory role dots at both ends of a straight line.



```
\begin{tikzpicture}
\entity (A) {A};
\entity at (2,0) {B} edge[both required]
node[roles] (p) {} (A);
\end{tikzpicture}
```










To show that either of many roles is mandatory, you can add an *inclusive-or* (*disjunctive mandatory*) role constraint with `constraint=mandatory` as shown in section 4.3. By default it is assumed that each entity or value must play at least some role. *Independent object types* whose roles are collectively optional can be marked by an exclamation mark appended to its name. It is recommended not to include implied mandatory constraints unless they refer to subtypes (section 5).

4.3 External constraints

External constraints span multiple roles that may come from different predicates. They are depicted by several circle symbols next the roles they limit, possibly linked to them with a dotted or dashed line (style `limits` and `limits to`). `tkz-orm` implements external constraint symbols as node shapes.

`/tikz/constraint=<constraint type>`

This style sets the font to ORM style on constraint color (violet). If you provide a constraint type as key value, the current node is shaped as constraint circle and the symbol of the specified constraint type is drawn. The most common constraint types are **exclusive** (alias **x**) to indicate that populations of two or more role-sequences must be mutually exclusive, **mandatory** (alias **required**, **total**, and **or**) to indicate that each at least on of two more roles must be played by an object type, and **xor** (alias **partition**) to indicates that exactly one of two or more roles must be played by an object type. These constraints can also be used in subtyping (section 5). The constraint type **unique** and **preferred unique** enforces combinations of object types that play a given set of roles to always be the same. The types **equal**, **subset**, and **supset** indicate that tuples of roles have to be equal, subset or superset compared to each other (**supset** is not included in standard ORM2). The constraint type is **external** only draws the circle and can be used for custom constraints.

	exclusive / x		mandatory / total required / or		xor / partition
	unique		preferred unique		external
	equal		subset		supset

```
\begin{tikzpicture}[orm]
  \matrix[column sep=2mm, row sep=2mm]{
    \node[constraint=x]{}; & \node[right]{exclusive / x}; &
    \node[constraint=or]{}; & \node[right,text width=2.8cm]
      {mandatory / total required / or}; &
    \node[constraint=xor]{}; & \node[right]{xor / partition}; \\
    \node[constraint=unique]{}; & \node[right]{unique}; &
    \node[constraint=preferred unique]{}; & \node[right]{preferred unique}; &
    \node[constraint=external]{}; & \node[right]{external}; \\
    \node[constraint=equal]{}; & \node[right]{equal}; &
    \node[constraint=subset]{}; & \node[right]{subset}; &
    \node[constraint=supset]{}; & \node[right]{supset}; \\
  };
\end{tikzpicture}
```

`/tikz/limits`

This style is to be used with lines that connect constraint circles and roles. It can also be used to link other kinds of constraints (for instance value constraints) to the entity, value, or role they belong to.



```
\begin{tikzpicture}
  \unary (r1) at (0,0) {};
  \unary (r2) at (0,-1.4) {};
  \draw[limits] (r1) to node[constraint=x] {} (r2);
\end{tikzpicture}
```

/tikz/limits to

This style is to be used with directed lines that connect constraint circles and roles. The line is drawn in the same style as `limits` but dashed and with an arrow tip of style `orm arrow` at the head.



Each object that
plays role 1 also
plays role 2

```
\begin{tikzpicture}[orm]
  \unary[index=2] (a) at (0,0) {};
  \unary[index=1] (b) at (0,-1.4) {};
  \draw[limits to] (b) -- (a)
    node[pos=.4,constraint=subset,name=s]{};
  \node[right=2mm of s,text width=2.3cm]
    {Each object that plays role 1 also plays role 2};
\end{tikzpicture}
```

\limits

\limitsto

Inside `{tikzpicture}` these commands can be used as abbreviations for `\draw[limits]` and `\draw[limits to]`.



```
\begin{tikzpicture}[orm]
  \limits (0,.4) to (1,.4); \limitsto (0,0) to (1,0);
\end{tikzpicture}
```

\constraintdeclare {<constraint type name>}{<path code>}

This command declares a new constraint type. The `{<path code>}` is passed to the `append after command` key to be drawn after the constraint circle. Unless you want to extend ORM you do not need to declare new constraint types. This command is for internal use only!

\constraintdeclarealias {<alias name>}{<existing constraint type name>}

This command can be used to create an alias (another name) for an existing constraint type. This command is for internal use only!

/tikz/every constraint

This style is invoked at every constraint. You can change this style to change for instance the constraint color.

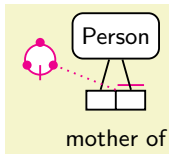
4.4 Ring Constraints

A ring constraint can be applied to any two roles of a predicate that are played by the same object type (or the same supertype). Such constraints can also be viewed as properties of a binary relation or as properties of a directed graph. There are 10 ring constraints that combined can be used in 26 forms. The graphical syntax of ‘irreflexive’ and ‘antisymmetric’ provided by this package are

slightly changed compared to the official ORM2 syntax and some combinations are omitted or changed.



```
\centering
\begin{tikzpicture}[orm]
  \foreach \n/\s in {0/irreflexive,1/asymmetric,2/strongly intransitive,
    3/antisymmetric,4/acyclic,5/acyclic intransitive,
    6/symmetric,7/purely reflexive,8/symmetric irreflexive,
    9/intransitive,10/reflexive,11/transitive}{
    \path ($mod(\n,3)*(3.4,0)-int(\n/3)*(0,0.8)$) node [constraint=\s] {}
      +(4mm,0) node[anchor=west] {\s}; }
\end{tikzpicture}
```



```
\begin{tikzpicture}[orm]
\entity (P) {Person};
\binary[below=of P,unique=2,label=below:mother of] (r) {};
\plays (P) to (r.one north) (P) to (r.two north);
\limits (r.north) to +(-1,0.4) node[constraint=acyclic]{};
\end{tikzpicture}
```

4.5 Number and Value Constraints

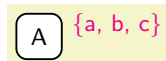
Value constraints, cardinality constraints, and occurrence frequencies can simply be drawn beside the object type or role they refer to, optionally linked to with a dotted or dashed limitation line.

Frequency Constraints specify the number of times an object can play a role. Usually it is connected to the roles with a limitation line.



```
\begin{tikzpicture}
\binary[index=1:1] (b) {};
\limits (b.one south) -- +(0,-.4) node[constraint]{f};
\end{tikzpicture}
```

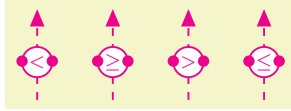
A **Value constraint** indicates which values are allowed in an object type or role. It can be defined by declaring the set of possible values enclosed in curly brackets next to an object or role type. The commands `\ormbraces` and `\ormvalues` are handy abbreviations to create curly brackets.



```
\begin{tikzpicture}
\entity (A) {A};
\node[constraint,anchor=north west,inner ysep=0]
  at (A.north east) {\ormbraces{a, b, c}};
\end{tikzpicture}
```

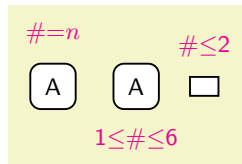
Value comparison-constraints are depicted by one of four comparison operators $<$, \leq (`le`), $>$, and \geq (`ge`). The constraints are shown at a dashed arrow between two roles in the same way as `constraint=subset` and `constraint=supset`

(but the value-comparison is between instances not between sets). Equality can be stated with `constraint=equal` which should not be confused with similar looking `constraint=purely reflexive`.



```
\tikz \foreach \x/\s in {0/<,1/ge,2/>,3/le}{
  \draw[limits to] (\x,0) to (\x,1.2);
  \node[constraint=\s] at (\x,.5) {};
};
```

Cardinality constraints are rarely included in ORM diagrams since they are often implied by other constraints. However you can explicitly say that each population of an object type or a role includes exactly, at most, or at least a given number of instances. This is done by adding a cardinality constraint next to the object or role. The hash sign (“#”) stands for the cardinality.



```
\begin{tikzpicture}[orm]
  \entity[label={[constraint]\#=$n$}] {A};
  \entity[label={[constraint]below:1$\leq$\#$\leq$6}]
    at (1.1,0) {A};
  \role[label={[constraint]\#$\leq$2}] at (2,0) {};
\end{tikzpicture}
```

4.6 Textual constraints

Constraints not expressed by predefined graphical notation may be specified as textual rules. Textual rules can be displayed as footnotes with footnote numbers or signs that mark the involved elements in the diagram.

`/tikz/rule=<mark>`

This key is to be used with nodes that contain textual rules. The optional `<mark>` is shown as footnotes index left to the rule.

¹ **Each Number identifies at most one Room.**

```
\begin{tikzpicture}
  \node[rule=1] {{\bfseries Each} Number identifies {\bfseries at most one} Room.};
\end{tikzpicture}
```

`\rules`

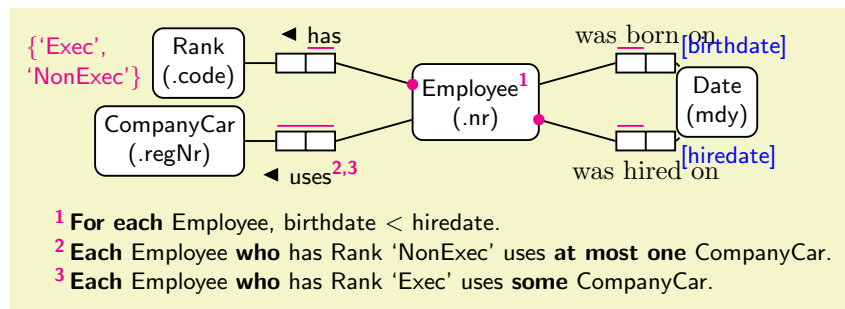
This command is an abbreviation for `\matrix[row sep=0mm,nodes={right}]` inside `{tikzpicture}`. Matrices are useful to draw multiple textual rules below each other.

5 Subtyping

To draw type hierarchies you can use the tree syntax of TikZ. Euler diagrams are a less used alternative for simple type hierarchies.

`/tikz/subtype`

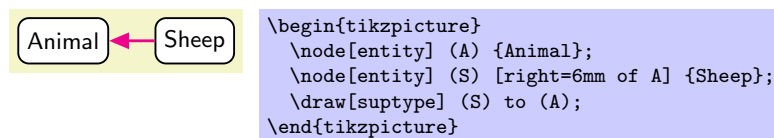
Draws a subtype relationship arrow from the supertype to the subtype.



```
\begin{tikzpicture}
\entity (E) {Employee\ormind{1}\(.nr)};
\binary[left=of E.north west,unique=2,label=\ormleft{has}] (h) {};
\binary[left=of E.south west,unique=1-2,
label=below:\ormleft{uses\ormind{2,3}}] (u) {};
\entity[left=of h] (Rank) {Rank\(.code)};
\entity[left=of u] (Car) {CompanyCar\(.regNr)};
\node[constraint=text,align=left,anchor=east] at (Rank.west)
{\textbraceleft'Exec',\('NonExec'\textbraceright};
\plays[mandatory] (E) to (h.east);
\plays (h) to (Rank) (E) to (u.east) (u) to (Car);
\binary[right=of E.north east,unique,label=was born on] (b) {};
\binary[right=of E.south east,unique,label=below:was hired on] (i) {};
\entity[right=1.8 of E] (Date) {Date\(.mdy)};
\plays[mandatory] (E) to (b.west) (E) to (i.west);
\plays (b.east) to (Date) (i.east) to (Date);
\node[role name,anchor=south west] at (b.east) {[birthdate]};
\node[role name,anchor=north west] at (i.east) {[hiredate]};
\rules at (-.4,-2) {
\node[rule=1] {\(\ormbf For each} Employee, birthdate $\<$ hiredate.}; \\\
\node[rule=2] {
\(\ormbf Each} Employee {\(\ormbf who} has Rank 'NonExec' uses
\(\ormbf at most one} CompanyCar.}; \\\
\node[rule=3] {
\(\ormbf Each} Employee {\(\ormbf who} has Rank 'Exec' uses
\(\ormbf some} CompanyCar.}; \\\
};
\end{tikzpicture}
```

/tikz/suptype

Works in the same way as subtype but with reverse direction.

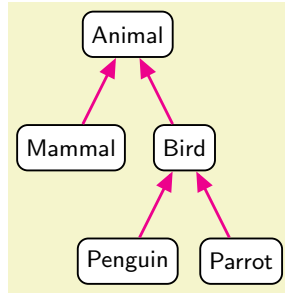


Multiple inheritance may require to select one path as primary. You can distinguish primary and secondary subtypes by drawing the latter with a dashed arrow (subinterface or supinterface).

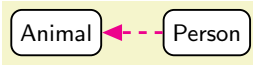
/tikz/subinterface

/tikz/supinterface

Draw secondary subtype/supertype relationship arrows.

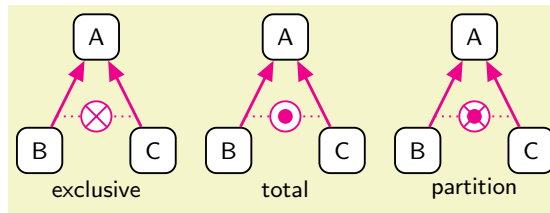


```
\begin{tikzpicture}[
  edge from parent/.style=subtype]
\node[entity] {Animal}
  child {node[entity] {Mammal}}
  child {node[entity] {Bird}
    child {node[entity] {Penguin}}
    child {node[entity] {Parrot}}
  };
\end{tikzpicture}
```



```
\begin{tikzpicture}
\node[entity] (A) {Animal};
\node[entity] (P) [right=8mm of A] {Person};
\draw[supinterface] (P) to (A);
\end{tikzpicture}
```

Subtype constraints can be shown linked to the subtype arrows:



```
\begin{tikzpicture}[orm]
\foreach \c/\x in {exclusive/0,total/2.5,partition/5}{
  \entity (A) at (\x,0) {A} [edge from parent/.style=subtype]
  child {node [entity] (B) {B}} child {node [entity] (C) {C}};
  \limits ($ (A)! .7! (B)$) to node[constraint=\c] {} ($ (A)! .7! (C)$);
  \node at (\x,-2) {\c};
};
\end{tikzpicture}
```

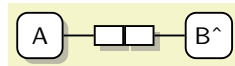
6 Additional Features

6.1 Duplicated and implied parts of a model

Sometimes an object type or predicate is referred to without describing all its details because it is defined in an external model or because it is shown duplicated at some other place in the same model. To indicate such an external or duplicated object type or a predicate, a shadow is added to its shape. Alternatively ORM2 allows to add a circumflex “^” to an object type’s name. A different kind of redundancy are roles and constraints that deduce from other parts of the model. ORM2 includes the possibility to shade redundant roles. This is useful for instance to show conceptual pathes or join fact types that are normally excluded. Moreover ORM allows a *zooming* on object types. This means that only objects and roles connected to a given object type are shown.

/tikz/duplicated model

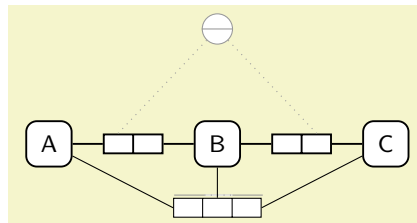
This style modifies the styles `every object` and `every predicate` so all object types and predicates in the current scope get a shadow.



```
\begin{tikzpicture}[orm]
\begin{scope}[duplicated model]
\entity (A) {A};
\node[role] (r1) [right=of A] {};
\node[role] (r2) [right=0 of r1] {};
\draw[relationship] (A) -- (r1);
\end{scope}
\entity (B) [right=of r2] {B^{\{}}};
\plays (r2) -- (B);
\end{tikzpicture}
```

/tikz/IMPLIED model

This style modifies the styles `every orm line` and `every object` in the current scope to draw all lines thin and all objects filled gray. The style is currently broken.

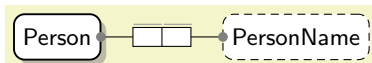


```
\begin{tikzpicture}
\matrix[column sep=4mm] {
\entity (A) {A}; & \binary (ab) {}; & \entity (B) {B}; & \\
\binary (bc) {}; & & \entity (C) {C}; & \\
\plays (A) -- (ab) -- (B) -- (bc) -- (C);
\begin{scope}[IMPLIED model]
\node[constraint=unique] (con) [above=of B] {};
\limits (ab.one north) -- (con) -- (bc.two north);
\ternary[unique=1-3,skip unique=2] (abc) [below=4mm of B] {};
\plays (A) -- (abc.west); \plays (B) -- (abc); \plays (C) -- (abc.east);
\end{scope}
\end{tikzpicture}
```

/tikz/duplicated

/tikz/IMPLIED

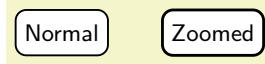
This styles work like `duplicated model` and `IMPLIED model` but only affect the current element. The style are currently broken.



```
\begin{tikzpicture}
\entity[duplicated] (P) {Person};
\value[IMPLIED,right=1.6 of P] (V) {PersonName};
\draw[IMPLIED,both required] (P) to
node[roles,unique=1,unique=2]{} (V);
\end{tikzpicture}
```


`/tikz/zoomed`

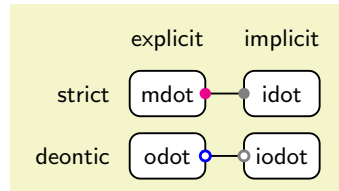
This styles visualizes an object type as *zoomed* by using a thicker line.



```
\begin{tikzpicture}
\entity at (0,0) {Normal};
\entity[zoomed] at (2,0) {Zoomed};
\end{tikzpicture}
```

6.2 Arrow heads

Constraint dots are implemented as arrow heads. The normal constraint dot is drawn with arrow head `mdot` for mandatory. Furthermore there is an implicit mandatory constraint dot (`idot`), a deontic mandatory constraint dot (`odot`), and a implied deontic mandatory constraint dot (`iodot`). You *should not* directly use this arrow heads but the mandatory role constraints `cdot`, `required` etc. (4.2) which can be modified by other styles. This feature is not fully tested yet.



```
\begin{tikzpicture}[orm]
\matrix [row sep=2mm,column sep=3mm,every entity/.style={minimum width=10mm}]{
\entity[label=left:strict,label=above:explicit,right] (m) {mdot}; &
\entity[label=above:implicit] (i) {idot}; \\
\entity[label=left:deontic,right] (o) {odot}; &
\entity (io) {iodot}; \\
\plays[mdot-idot] (m) to (i);
\plays[odot-iodot] (o) to (io);
\end{tikzpicture}
```

6.3 Macros for text layout

The following macros can be used both in TikZ pictures or normal text:

`\ormtext`

Sets the font to the same sans-serif variant which is used in ORM diagrams.

`\ormbf`

Sets the font to a bold variant of `\ormtext` .

`\ormc`

Sets the font to a `\ormtext` in constraint color.

`\ormsup` $\{\langle text \rangle\}$

Puts some text in a superscript variant of `\ormtext` .

`\ormsub` $\{\langle text \rangle\}$

Puts some text in a subscript variant of `\ormtext` .

`\ormind {\text}`

Puts some text in a superscript variant of `\ormbf` .

`\ormbraces {\text}`

Puts some text as `\ormtext` in braces.

`\ormvalues {\text}`

Puts some text as `\ormc` in braces in constraint color.

¹A Person is not {0,1}, Male or Female, up \blacktriangle or \blacktriangleleft left but $\text{queer}\text{multi}\text{gender}!$

```
\ormind{1}A {\ormtext Person} is not \ormbraces{0,1},~
{\ormc Male} {\ormbf or} {\ormtext Female}, \ormup{up}~
{\ormbf or} \ormleft{left} but \ormsub{queer}\multi\ormsup{gender}!
```

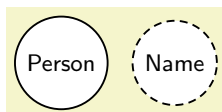
7 Settings and Utilities

`/tikz/orm`

This style sets the font and line width and the default node distance

`/tikz/orm-spacious`

If you prefer to have entities and labels typeset as circles, you can use `orm-spacious` instead of `orm`.



```
\begin{tikzpicture}[orm-spacious]
\node[entity] (P) at (0,0) {Person};
\node[value] (N) at (1.5,0) {Name};
\end{tikzpicture}
```

`/tikz/every orm line`

This style is invoked by all styles of this package that draw lines. By default it sets the line width to 0.3mm.

Changes

0.1.4, January 15, 2016 Unstable developer version (at Github).

0.1.4, January 15, 2016 Power types, sequence types, `orm-spacious` added.

0.1, January 25, 2010 First release (at CTAN).

References

[GPL91] GNU Public License Version 2, 1991.

[Hal05] Terry Halpin. ORM 2 Graphical Notation, 2005.

[HM08] Terry Halpin and Tony Morgan. *Information Modeling and Relational Databases*. Morgan Kaufmann, 2008.

[LPP08] LaTeX Project Public License (LPPL) Version 1.3c, 2008.

Index

- < constraint, 13
- > constraint, 13
- custom constraint, 10
- equal constraint, 10
- ge constraint, 13
- le constraint, 13
- mandatory constraint, 10
- or constraint, 10
- partition constraint, 10
- preferred unique constraint, 10
- required constraint, 10
- subset constraint, 10
- supset constraint, 10
- total constraint, 10
- unique constraint, 10
- xor constraint, 10
- x constraint, 10
- Arrow tips
 - mandatory, 11
- `\binary`, 5
- both mandatory key, 9
- both required key, 9
- cardinality constraints, 13
- `cdot` key, 9, 9
- constraint key, 7, 10
- constraint dot key, 9, 9
- constraintcolor key, 7
- `\constraintdeclare`, 11
- `\constraintdeclarealias`, 11
- Constraints, 6
- constraints
 - cardinality, 13
 - textual, 13
 - value-comparison, 13
 - values, 12
- duplicate key, 16
- duplicate model key, 16, 16
- Entities, 3
- `\entity`, 4
- entity key, 3, 3, 4
- every constraint key, 11
- every entity key, 3
- every object key, 4, 16
- every orm line key, 4, 16, 18
- every predicate key, 5, 16
- every relationship key, 5
- every value key, 3
- implied key, 16
- implied model key, 16, 16
- Independent object types, 9
- index key, 6
- `\limits`, 11
- limits key, 7, 10, 10, 11
- limits to key, 7, 10, 11
- `\limitsto`, 11
- mandatory key, 9
- Object types, 3
 - Independent, 9
- orm key, 18
- orm-spacious key, 18
- `\ormbf`, 17, 18
- `\ormbraces`, 12, 18
- `\ormc`, 17, 18
- `\ormind`, 18
- `\ormleft`, 5
- `\ormsub`, 17
- `\ormsup`, 17
- `\ormtext`, 17, 17, 18
- `\ormup`, 5
- `\ormvalues`, 12, 18
- `\plays`, 5, 5
- plays key, 4
- power key, 3
- predicates, 4
- relation key, 4
- relationship key, 4, 5, 9
- required key, 9
- required by key, 9
- `\role`, 5
- role key, 4
- role name key, 6
- roles, 4
- `\roles`, 5
- roles key, 4, 5
- rule key, 13

- `\rules`, 13
- sequence key, 3
- skip unique key, 8
- skipped uniqueness bar key, 8
- subinterface key, 14, 14
- subtype key, 13, 14
- supinterface key, 14, 14
- suptype key, 14
- `\ternary`, 5
- textual constraints, 13
- `\unary`, 5
- unique key, 7, 8
- uniqueness bar key, 8
- `\value`, 4
- value key, 3, 3, 4
- value constraints, 12
- value-comparison constraints, 13
- Values, 3
- `\vbinary`, 5
- `\vrole`, 5
- vrole key, 4
- `\vroles`, 5
- vroles key, 4, 5
- `\vternary`, 5
- `\vunary`, 5
- zoomed key, 17