The content model for ORM users

Content classes, objects, versions, translations, nodes, datatypes, sections and object states explained in the terms of an ORM - what each is in the database and in PHP, how a typical blog or catalog maps onto it, and...

Source file: `doc/eng-us/help/web-framework-developers/the-content-model-for-orm-users.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/web-framework-developers/the-content-model-for-orm-users.md

This page is for developers who would start a project by writing entity classes and migrations. In Exponential, most data that editors manage is **content**: you define its types in the administration (or ship them as packages), and the kernel stores, versions, translates, permits, caches and indexes it. Your code reads and writes content through the content API instead of a model class of its own.

Contents

1. The pieces in ORM terms
2. Datatypes
3. Versions and translations
4. Nodes: one object, several places
5. Sections, states and relations
6. A blog, mapped
7. Reading and writing from code

The pieces in ORM terms

Exponential	Closest ORM idea	Class
Content class (Article: Title, Intro, Body, Image)	an entity or model definition	`eZContentClass`
Class attribute	a field definition with its type and settings	`eZContentClassAttribute`
Content object	one row of the entity, with its history	`eZContentObject`
Version	a revision of the row: draft, published, archived and more	`eZContentObjectVersion`
Object attribute	one field's value, per version and language	`eZContentObjectAttribute`
Node (location)	where the row appears in the site tree; parent and children	`eZContentObjectTreeNode`
Remote ID	a stable external key you choose, unlike the numeric id	`remote_id` of objects and nodes

The administrator's view of the same terms: The content model and editing content.

Datatypes

Each class attribute has a **datatype**, which is a column type, a form widget, a validator and a renderer in one: `ezstring` (one line), `eztext`, `ezxmltext` (rich text), `ezinteger`, `ezfloat`, `ezboolean`, `ezdate`, `ezdatetime`, `ezemail`, `ezurl`, `ezimage`, `ezbinaryfile`, `ezobjectrelationlist`, `ezselection`, `ezmatrix`, `ezuser` and more (the shipped ones are in `kernel/classes/datatypes/`). The ones offered for new attributes are `content.ini [DataTypeSettings] AvailableDataTypes[]`. A new kind of field is a datatype class extending `eZDataType` in your extension (Custom datatypes).

Versions and translations

- Every edit creates a **version**; publishing makes it current and archives the previous one. Drafts are versions with their own status (`eZContentObjectVersion::STATUS_DRAFT`, `STATUS_PUBLISHED`, `STATUS_ARCHIVED` and others).
- Each version holds the attribute values for each **language** it is translated into. Which language a siteaccess shows follows its `SiteLanguageList` (Content languages).
- History, compare and restore come with it; you write none of it.

Nodes: one object, several places

An object is placed in the tree by **nodes**. One object can have several nodes (the same article in "News" and in "Sport"); one is its main node. The tree gives you navigation, URL aliases, subtree permissions and subtree queries (`path_string`, `depth`). There is no separate "category" table to design unless you want one.

Sections, states and relations

Feature	Use	Read
Section	a group of content for permissions and design (Standard, Users, Media...)	Sections
Object state	a workflow-like status per object (for example "locked", "in review") usable in policies and request rules	Object states
Object relation	a link from one object to others, kept with reverse lookups	The content API

A blog, mapped

Framework approach	Exponential approach
`posts` table, `Post` model, migration	class `blog_post` (title, intro, body, image, tags) created in **Setup > Classes** or installed from a package
`categories` table, foreign key	folders in the tree, or a relation attribute, or tags from `eztags`
`post_revisions` table	versions, built in
`post_translations` table	translations, built in
`slug` column, unique index	URL alias, built in (Pretty URLs)

Framework approach	Exponential approach
Admin CRUD screens	the administration, built in
Policy class per model	roles with `content/read`, `content/edit` limited by class, section, subtree (Authorization)
Controller and templates for the front end	override templates for the class's full and line views (Template resolution)

Reading and writing from code

PHP

```php
$node   = eZContentObjectTreeNode::fetchByURLPath( 'about_us' );
$object = eZContentObject::fetchByRemoteID( 'expdemo-settings-page' );
$title  = $object->dataMap()['title']->content();

$article = eZContentFunctions::createAndPublishObject( array(
    'creator_id'       => $userID,
    'class_identifier' => 'article',
    'parent_node_id'   => $parentNodeID,
    'remote_id'        => 'feed-item-' . $externalID,
    'attributes'       => array( 'title' => 'A new article' ) ) );
```

Rules:

- **Write content only through the content API** (`eZContentFunctions`, operations), never with SQL on the content tables: publishing runs workflows, clears caches and updates the search index.
- **Prefer remote IDs** for content your code depends on; numeric IDs differ between installations.
- **Set a** `Limit` on every list fetch.

Full reference: The content API in PHP and Creating and updating content in code.

Related pages

- ORM and active record: eZPersistentObject
- Designing your content model
- Datatype life cycle
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

> **Become an Exponential Partner**
>
> Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.
>
> Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.