

Docs » Data Handling Category » Repository Project » Querying

Querying

Query methods, identifier lookup, filtering, sorting and Doctrine Criteria

Anonymous · 20/10/2026

Querying

`RepositoryInterface` defines the following operations. Examples in this guide use the in-memory `Repository`; custom implementations execute their own backend queries.

Method	Result
<code>find(\$id)</code>	Entity object or null
<code>findAll()</code>	Array of entity objects
<code>findBy(\$criteria, \$orderBy, \$limit, \$offset)</code>	Array of matching entity objects
<code>findOneBy(\$criteria, \$orderBy)</code>	First matching entity or null
<code>count(\$criteria)</code>	Number of matching rows
<code>findByCriteria(\$criteria)</code>	Array of entity objects selected with Doctrine Criteria
<code>getClassName()</code>	Entity class name

Identifier Lookup

```
$product = $repository->find('prod-001');  
$missing = $repository->find('unknown'); // null.
```

The in-memory implementation accepts string and integer identifiers and looks up the stored row key. Its optional `find()` lock arguments are unused. An `id` field inside the row does not change the lookup key.

Filtering and Counting

```
$computers = $repository->findBy(['category' => 'computers']);  
$selected = $repository->findBy([  
    'category' => ['computers', 'accessories'],  
]);  
$first = $repository->findOneBy(['category' => 'computers']);  
$total = $repository->count();
```

```
$computerCount = $repository->count(['category' => 'computers']);
```

Fields in a criteria array are combined with AND. An array value means any of the listed values may match that field. The in-memory implementation uses PHP's non-strict `in_array()` comparison. A field that is absent or null does not match this filtering path, including when the requested value is null.

An empty criteria array matches every row. `findAll()` and the filtering methods return lists of entities rather than preserving source keys.

Ordering and Pagination

```
$page = $repository->findBy(  
    criteria: ['category' => 'computers'],  
    orderBy: ['price' => 'DESC', 'name' => 'ASC'],  
    limit: 20,  
    offset: 40,  
);
```

The in-memory implementation processes ordering fields in the supplied order. DESC selects descending comparison; other direction values select ascending comparison. Missing or null ordering fields are skipped for that comparison. An offset or limit below zero raises a translatable invalid-argument exception.

Doctrine Criteria

```
use Doctrine\Common\Collections\Criteria;  
use Doctrine\Common\Collections\Order;  
  
$expensiveProducts = $repository->findByCriteria(  
    Criteria::create()  
        ->where(Criteria::expr()->gt('price', 1000))  
        ->orderBy(['price' => Order::Descending])  
        ->setMaxResults(10),  
);
```

For Repository, criteria are applied to the stored rows before entity creation. The result contains entity objects. This path uses Doctrine Collections' expression semantics, which differ from the basic `findBy()` membership checks.

A custom SQL repository is responsible for implementing these query methods and translating supported criteria to its backend. The manager does not translate expressions or issue SQL.

Raw Rows and Entities

`Repository::all()` returns raw rows, while `Repository::findAll()` returns entities. Container mutations change the in-memory row collection and do not write to a source file or database. Query operations that process rows validate that each row is an array.

Last updated on 20/10/2026