1. **细化接口**

根据资源库的个体需求定义查询方法，方法命名也应体现领域逻辑。将查询条件封装进查询方法内（穷举所有可能出现的操作，将所有的方法都实现。）

不是一种通用的以sql语句为输入参数的操作，而是根据不同的方法限制了不同的查询参数。

例如：allInProgressOrdersOfCustomer(customerId)只需要具体的条件，而不需要使用通用的sql语言。

优点：

1、减轻调用者的负担。

2、有查询条件的查询方法有业务含义，可读性更高

缺点：

1. 增加一个新的查询条件可能会导致更多的查询方法的增加，扩展性较差。
2. 过于具体，接口方法膨胀，会出现大量的接口。
3. **通用接口**

通用的查询，仅做一些基础的、通用的接口，而在具体的业务实施中，由调用者再对查询方法进行扩展。

不限制查询参数，查询参数由调用者决定，由调用者根据实际的业务需求设定自己的查询参数实现接口功能。

比如一个查询方法 它只定义Findallmatch(string sql)这个方法，后面相关的具体的sql语言，查询条件等，都由调用者自己去实现（使用查询条件表达式）。

优点：

1. 以通用性换取接口的可扩展
2. 灵活的运用于各种业务规则。
3. 能适应各种的业务变化，而不需要对接口进行增改。

缺点：

1、可读性差，他只定义了一个大的框架，具体的内容则是在具体实现时被定义，具体功能的实现由调用者自己决定。

1. **规格模式方式**

使用规格模式来封装查询条件。相较于直接使用查询条件，规格模式的封装性更好，可以实现按照业务规则定义不同的规格子类。

规格模式例子： 见代码

主要思想： 规格模式有四个接口方法，分别为And、Not、Or和Issatisfy四个方法，其中最后一个方法为判断条件，前三个为组合条件，根据不同的Issatisfy条件结合And、Not、Or三个方法，可以形成自定义的查询条件组合。由此实现对通用查询条件语句的封装。

缺点：

1. 实现不同的组合需要定义不同的实例
2. 实际上，他并没有改变通用接口的可读性问题，对于具体功能的实现仍然需要调用者自己实现。

优点：

1、主要是对通用接口内的查询语句进行封装，封装性好

2、同时也具备了可扩展性

**4、个人理解：**

1、根据项目情况:

若是项目较小，关于数据库的操作较少，在后续的业务扩展等方面对数据库的改动较小的，可以使用细化接口，毕竟若是变化不大，细化接口还是造成不了太大的接口膨胀的问题。

而若是涉及到大量的不同的数据查询方法的，则需要定义通用接口，为了保证封装性，再使用规格模式来对查询条件进行封装。保证扩展性。

若是当前数据查询操作较少，但不保证未来数据库操作情况变化的，也可以采用，先定义必须的细化接口，同时也定义通用接口以保证扩展性。通用接口的查询条件也可以用规格模式进行封装。

2、根据领域

使用方向是面向领域，那么最好仓储库接口的定义还是与具体的业务逻辑相关，即面对相应的服务提供相应的存储库方法接口，这样的定义会更为清晰。

参考（<https://juejin.cn/post/6845166890554228744#heading-9>）接口定义规范中也提及，应避免“通用”的Repository模式