别名（Aliases）

别名（Aliases）

别名是现有QL实体的替代名称。
定义别名后，您可以使用该新名称来引用当前模块名称空间中的实体。

定义别名（Defining an alias）

您可以在任何模块的主体中定义别名。为此，您应指定：

1. 关键字`module`，`class`或`predicate`分别为module， type或non-member谓词定义别名。
2. 别名的名称。这应该是该类型实体的有效名称。例如，有效的谓词别名以小写字母开头。
3. 对QL实体的引用。这包括实体的原始名称，对于谓词，还包括谓词的Arity。

您也可以注释别名。请参阅 可用于别名的注释列表。
请注意，这些注释适用于别名引入的名称（而不适用于基础QL实体本身）。例如，别名对其别名的名称可以具有不同的可见性。

模块别名（Module aliases）

使用以下语法为模块定义别名：

```
module ModAlias = ModuleName;
```

例如，如果您创建的新模块`NewVersion`是`OldVersion`的更新版本，则可以弃用该名称`OldVersion`，如下所示：

```
deprecated module OldVersion = NewVersion;
```

这样，两个名称都解析为相同的模块，但是如果使用名称`OldVersion`，则会显示弃用警告。

类型别名（Type aliases）

使用以下语法为类型定义别名：

```
class TypeAlias = TypeName;
```

请注意，这`class`只是一个关键字。您可以为任何类型（即原始类型， 数据库类型和用户定义的类）定义别名。
例如，您可以使用别名将基本类型的名称缩写`boolean`为`bool`：

```
class bool = boolean;
```

或者，要使用在`OneTwoThreeLib.qll`中的模块 `M`中定义的`OneTwo`类，您可以创建一个别名来使用较短的名称`OT`：

```
import OneTwoThreeLib

class OT = M::OneTwo;

...

from OT ot
select ot
```

谓词别名（Predicate aliases）

使用以下语法为非成员谓词定义别名：

```
predicate PredAlias = PredicateName/Arity;
```

这适用于有结果或无结果的谓词。
例如，假设您经常使用以下谓词，该谓词计算小于10的正整数的后继：

```
int getSuccessor(int i) {
```

```
  result = i + 1 and
  i in [1 .. 9]
}
```

您可以使用别名将名称缩写为`succ`：

```
predicate succ = getSuccessor/1;
```

作为没有结果的谓词的示例，假设您拥有一个谓词，该谓词可容纳小于10的任何正整数：

```
predicate isSmall(int i) {
  i in [1 .. 9]
}
```

您可以给谓词一个更具描述性的名称，如下所示：

```
predicate lessThanTen = isSmall/1;
```