QL基础

关于QL语言

QL的属性

QL和面向对象

QL和通用编程语言

参考文献

关于QL语言

QL是一种声明性的，面向对象的查询语言，经过优化可实现对分层数据结构（尤其是表示软件工件的数据库）的高效分析。
数据库是有组织的数据集合。最常用的数据库模型是将数据存储在表中的关系模型，而SQL（结构化查询语言）是关系数据库最常用的查询语言。
查询语言的目的是提供一个编程平台，您可以在其中询问有关存储在数据库中的信息的问题。数据库管理系统管理数据的存储和管理并提供查询机制。查询通常引用相关的数据库实体，并指定结果必须满足的各种条件（称为谓词）。查询评估涉及检查这些谓词并生成结果。好的查询语言及其实现的一些理想属性包括：

- 声明性规范–声明性规范描述了结果必须满足的属性，而不是提供计算结果的过程。在数据库查询语言的上下文中，声明性规范抽象了基础数据库管理系统和查询处理技术的详细信息。这大大简化了查询编写。
- 富有表现力–强大的查询语言可让您编写复杂的查询。这使得该语言广泛适用。
- 高效的执行–查询可能很复杂，数据库可能非常庞大，因此对于查询语言实现而言，有效地处理和执行查询至关重要。

QL的属性

QL的语法类似于SQL，但是QL的语义基于Datalog，Datalog是一种声明性逻辑编程语言，通常用作查询语言。这使得QL主要是一种逻辑语言，并且QL中的所有操作都是逻辑操作。此外，QL从Datalog继承递归谓词，并增加了对聚合的支持，从而使复杂的查询也变得简洁明了。例如，考虑一个包含人与父母的亲子关系的数据库。如果我们想找到一个人的后代数量，通常我们将：

- 查找给定人的后代，即一个孩子或一个孩子的后代。
- 计算使用上一步找到的后代的数量。

当您在QL中编写此过程时，它与上述结构非常相似。请注意，我们使用递归来查找给定人员的所有后代，并使用总计来计算后代的数量。由于语言具有声明性，因此可以将这些步骤转换为最终查询而无需添加任何过程详细信息。QL代码如下所示：

```
Person getADescendant(Person p) {
  result = p.getAChild() or
  result = getADescendant(p.getAChild())
}

int getNumberOfDescendants(Person p) {
  result = count(getADescendant(p))
}
```

有关QL重要概念和语法构造的更多信息，请参见各个参考主题，例如“表达式”和“递归”。这些说明和示例可帮助您了解该语言的工作原理以及如何编写更高级的QL代码。
有关QL语言的正式规范，请参阅“QL语言规范”。

QL和面向对象

面向对象是QL的重要特征。面向对象的好处是众所周知的一它提高了模块性，实现了信息隐藏，并允许代码重用。QL在不损害其逻辑基础的情况下提供了所有这些好处。这是通过定义一个简单的对象模型实现的，其中将类建模为谓词，将继承建模为隐含。可用于所有受支持语言的库广泛使用了类和继承。

QL和通用编程语言

这是通用编程语言和QL之间的一些突出的概念和功能差异：

- QL没有任何命令性功能，例如分配变量或文件系统操作。
- QL对元组集合进行操作，并且查询可以视为定义查询结果的一组复杂的集合操作序列。
- QL基于集合的语义使处理值的集合变得很自然，而不必担心有效地存储，索引和遍历它们。
- 在面向对象的编程语言中，实例化一个类涉及通过分配物理内存来保存该类实例状态的对象来创建对象。在QL中，类只是描述已经存在的值的集合的逻辑属性。

参考文献

学术参考文件 还概述了QL及其语义。有关数据库查询语言和数据日志的其他有用参考：

- Database theory: Query languages
- Logic Programming and Databases book
- Foundations of Databases
- Datalog