面向对象型数据库：

虽然关系型数据库在存储管理方面性能强大，但由于以上提到的结构僵化等问题，80年代出现了面向对象数据库技术，其中面向对象数据库系统宣言是这个思想振聋发聩的宣告。其将面向对象性数据库管理系统归总为必备特性（复杂对象，对象标识，封装性，类型，继承性，重载，可扩充性，计算完备性，辅存管理，并发性，恢复，即时查询）和可选特性（多重继承，类型检查和推理，分布式，类型系统）。面向对象的数据库系统在逻辑上和物理上从面向记录上升为面向对象、面向可具有复杂结构的一个逻辑整体。允许用自然的方法，并结合数据抽象机制在结构和行为上对复杂对象建立模型，从而大幅度提高效率，降低用户使用复杂性。后来对象型数据库思想不断发展分化，数据库发展趋势大致如：面向对象数据库->对象关系型数据库->对象代理数据库。

面向对象数据库主要是数据库与面向对象语言的结合。由以下几部分组成：对象模型（主要包括对象的特性，对象间的联系），对象描述语言：ODL（对象定义语言），OIF（对象交互格式），对象查询语言: OQL(对象查询语言。)，对象语言绑定（常见的是C++,JAVA。目的主要是实现ODL和ODMG缺乏的OML）。这里我们先提一下对象模型，有种说法是“面向对象数据库模式是类的集合”。面向对象的数据模型提供了一种类层次结构。在面向对象数据库模式中，一组类可以形成一个类层次。一个面向对象数据库可能有多个类层次。在一个类层次中，一个类继承其所有超类的全部属性、方法和消息。其次，对象语言绑定。这个特点出现的原因我认为主要是面向对象特点决定的，绑定可以方便的加入面向对象的功能，由用户灵活的实现定义和操作对象。但我认为引入语言绑定后，为了满足对多语言绑定的支持，还是牺牲了一些良好的特性。

接下来，我们在数据建模上与关系型进行对比分析。假设现在面对的对象是个复杂对象，它结构复杂，维度多样，且对象之间有较多组合继承关系。针对这种复杂对象建模时，1.关系数据库需要为所有数据在每种尺度下建立自己的表，访问时，通过不同尺度的查询条件，查询不同的表来返回信息。往往同种数据在不同尺寸下的表会有大量数据冗余，这加大了同种数据一致性维护难度。而如果通过集成，可以很好的降低这一难度。2.其次，我们无法在关系型数据库中实现对数据的计算，如添加c=a\*b，c属性必须也要建立一个字段，这是不必要的。对象型可以在表上定义一个计算的函数或者方法来实现这类查询，这是关系型所不能支持的。3.一些结构关系复杂的数据类型无法在关系型中储存，而是通过基础的数据类型去模拟记录，这往往导致数据库表单的设计过于复杂，而且需要大量的join来实现复杂数据的表达，而对象型为了满足多尺度的要求，可以对统一对象创建多个类并根据类层次结构来进行管理。

分析其组成并结合表现，我们可以得出结论，面向对象型数据库支持复杂对象，且建模方法自然，易理解。对象具有独立对象标识，使得查询速度极大加快。对象的封装性实现了良好的透明度，使用者不需知道操作的细节，加快了效率。因为面向对象数据库对复杂对象的表现优异，它广泛应用于：多媒体，工程数据，地理信息等复杂信息管理系统。

但由于它没有原型数据库做基础，完全从零开始实现一个对象型数据库，虽然足够灵活，但也开发费时费力，开发者需要手动实现数据库功能，自主开发需要的扩展数据类型，同时还要考虑性能优化问题，而且与其他数据库的转接兼容也相对麻烦。所以这种思路虽然很强大，但受到的限制还是太多：缺乏数学模型的支持，复杂难用，成熟度低，最后没有撼动关系型的地位。

之后，人们在对象型数据库的基础上建立了对象关系型数据库。对象关系数据模型是从关系数据模型出发，指向对象思维，以关系数据库和sql为基础的扩展关系模型，具有扩展复杂数据类型和允许用户自定义函数功能。它具有面向对象特性，收通用性强的sql3标准支持。对象关系型数据库管理系统既满足了传统关系型数据库用户的要求，同时又能支持复杂数据管理需要的应用，具有强大的查询语言功能。关系数据库只能做到对复杂数据的存储和对简单数据的查询。而对象关系数据库的最大功能是能做到对复杂数据的查询且速度很快，具有可扩充性。对象关系型数据库的典型有oracle数据库。

Db4o

面向对象数据库的概念提出后，对象数据库管理系统为面向对象编程语言增加了持久的概念。最早的商品化 ODBMS 出现在 1986 年，是 Servio 公司和 Ontos 公司推出的。后来，九十年代 Object Design、 Versant 、 Objectivity 、 Poet 、UniSQL等公司也加入了这个开拓行列。然而目前为止成熟的对象型数据库还是比较少，而且开源的更少。2004年，**db4o作为免费的开源ODBMS（纯面向对象数据库管理系统）发布**，2005年，db4o首先将Native Queries 作为面向对象的数据访问API 来实现，该API完全依赖于编程语言（Java / C＃）本身，由此它成为了100%原生数据库。目前国内相对比较流行的也是db4o,虽然很可惜这个项目已经在2014年被官方停止维护。

Db4o 提供两种运行模式，分别是本地模式和服务器模式。本地模式是指直接在程序里打开 db4o 数据库文件进行操作，而服务器模式则是客户端通过 IP 、端口来访问服务器。一般来说，javeEE更适用于服务器模式（这里假设api语言用的是java），嵌入式开发更适用于本地模式。它有几大特点：一：轻量级。实现该数据库只需要下载400k左右的jar包或者dll文件，并将其引入到系统中。这个特点也让它非常适用于嵌入式系统。二：原生。因为其为原生数据库，开发者不需下载管理工具，不需学习新的数据库语言，可以直接用编程语言管理数据库。三：多平台支持。db4o 支持从 Java 1.1 到 Java 5.0，此外还支持 .NET 、 CompactFramework 、 Mono 等 .NET 平台，也可以运行在 CDC 、 PersonalProfile 、 Symbian 、 Savaje 以及 Zaurus 这种支持反射的 J2ME 方言环境中，还可以运行在 CLDC 、 MIDP 、 RIM/Blackberry 、 Palm OS 这种不支持反射的 J2ME 环境中。四：性能优秀。下图为db4o 官方公布的基准测试数据，db4o 我们可以非常惊讶的发现在读写查删时，它比采用 Hibernate/MySQL 方案速度高出几到几十倍。



图2. db4o 官方基准测试数据

接下来我将使用db4o，结合我们的食品场景进行数据库设计。因为db4o是原生数据库，我们拟采用java进行数据库开发。首先，在设计类时，我们可以发现面向对象的思想给方便面复杂的属性管理提供了不少便利。我们可以设计两个父类：用户，方便面。接下来使用继承的思路，把人分为测试员（测试用的机器为了方便也可以放在这个子类里，作为人的特性被设置为null即可，主键则专门设置规则）和用户两个子类。方便面和人之间存在两种关系：一：用户—吃&评价是否喜欢🡪方便面，二：测试员从多个角度（时间，定性，定量，数据）--打分🡪方便面。类模型可以简单描述为：

|  |
| --- |
| 此处有图或者代码 |

建立类并将数据存入后，当我们需要新增食物属性时，我们可以直接在食物的类里定义新的属性和对应的方法。当我们需要查询食物信息时，我们有三种查询方式：QBE（Query by Example）、NQ（Native Queries）、SODA（Simple Object Database Access）。QBE 标准是由IBM提供的，是一种比较传统成熟的查询。SODA是 db4o 最底层的查询 API，虽然写起来较麻烦，但当遇到动态生成查询的时候 SODA 实在非常高效。db4o 更推荐使用 NQ 进行查询，NQ 方式支持原生语言，也就是说，我们可以直接使用java或者.NET来写数据库查询语句，这无疑非常方便。但我们也需要注意，NQ的本质是转换接口，数据库实现内部还是会设法把NQ转换为SODA。这种转换大多数情况下是成功的，但当有些查询表达式的流向图非常难于分析时，db4o 将不得不实例化一些持久对象来真实地运行 NQ 表达式。这个过程会影响性能。我在示例开发中使用的是最特色的NQ方法。但这里对类间的联系并没有很直接的体现。

|  |
| --- |
| 其次，它是一个面向对象的系统：只有支持面向对象数据库模型，支持复杂对象，具有运用各种构造机制从简单对象组成复杂对象的能力。复杂对象构造能力加强了对客观现实世界的模拟能力，且方法自然、易理解、具有对象标识，对象标识独立于其值而存在的特性可以极大地加快查询速度；具有封装性，对象既封装了数据，又封装操作，实现了信息隐藏，使用户不必知道操作的实现细节，只利用设计者 提供的消息即可访问对象。 |

|  |
| --- |
| 面向对象数据库的概念提出是很早的， “面向对象数据库系统”这一术语第一次出现于 1985 年。著名的研究项目包括：Encore-Ob/Server ( 布朗大学）， EXODUS（Wisconsin 大学）， IRIS （惠普）， ODE （ Bell 实验室）， ORION （MCC ） ，Vodak （GMD-IPSI）和 Zeitgeist （Texas Instruments）。其中以 ORION 项目发表的论文数为最多。 MCC 的 Won Kim 将这些论文中最有价值的一部分汇编成书并由 MIT 出版社出版。对象数据库管理系统为面向对象编程语言增加了持久的概念。最早的商品化 ODBMS 出现在 1986 年，是 Servio 公司（现在的 GemStone 公司）和 Ontos 公司推出的。后来（九十年代） Object Design （ ODI ）、 Versant 、 Objectivity 、 O2 Technology 、 Poet 、 Ibex 、 UniSQL 和 ADB MATISSE 等公司也加入了这个开拓行列。然而目前为止成熟的对象型数据库还是比较少，而且开源的更少。2004年，**db4o作为免费的开源ODBMS发布**，2005年，db4o首先将Native Queries 作为面向对象的数据访问API 来实现，该API完全依赖于编程语言（Java / C＃）本身。国内相对比较流行的也是db4o,但是这个项目也已经在XX年被官方停止维护。  b4o 提供两种运行模式，分别是本地模式和服务器模式。本地模式是指直接在程序里打开 db4o 数据库文件进行操作，而服务器模式则是客户端通过 IP 地址、端口以及授权口令来访问服务器。 |

面向对象数据库笔记

|  |
| --- |
| 发展趋势：面向对象数据库->对象关系型数据库->对象代理数据库 |

|  |
| --- |
| 面向对象数据库：数据库与面向对象语言的结合。由以下四部分组成：   * 对象模型（对象的特性，对象间的联系） * 对象描述语言：ODL（对象定义语言） OIF（对象交互格式） * 对象查询语言: OQL(对象查询语言) * 对象语言绑定：C++,JAVA 目的主要是实现ODL和ODMG缺乏的OML，利用语言灵活性。   面向对象模型：   * 类层次结构（子类/超类；外延） * 对象（原子对象，结构对象，聚集对象） * 对象的状态（属性，联系（只支持二元联系））   对象语言绑定：   * 原因1.面向对象特点决定的，绑定可以方便的加入面向对象的功能，由用户灵活的实现定义和操作对象。2.可以扩大在掌握一门以上语言的用户中的应用度。（为了开发者方便？是这个意思吗？） * 缺点：1.为了适应多种语言的特性，牺牲了一些良好的特性来满足对多语言绑定的支持。2.虽然有多语言的绑定，对一些普通用户来说应用依然是十分困难的。   面向对象数据库系统：  （非常重要的面向对象数据库系统宣言）包括了：   * 必备特性（复杂对象，对象标识，封装性，类型，继承性，重载，可扩充性，计算完备性，辅存管理，并发性，恢复，即时查询） * 可选特性（多重继承，类型检查和推理，分布式，类型系统）   面向对象数据库应用：  广泛应用于：多媒体，工程数据，地理信息等复杂信息管理系统。  针对这种复杂对象建模时，1.关系数据库需要为所有数据在每种尺度下建立自己的表，访问时，通过不同尺度的查询条件，查询不同的表来返回信息。往往同种数据在不同尺寸下的表会有大量数据冗余，这加大了同种数据一致性维护难度。而如果通过集成，可以很好的降低这一难度。2.其次，无法再关系型数据库中实现对数据的计算，如添加c=a\*b，c属性必须也要建立一个字段，这是不必要的。对象型可以在表上定义一个计算的函数或者方法来实现这类查询，这是关系型所不能支持的。3.一些结构关系复杂的数据类型无法再关系型中准出，二是通过基础的数据类型去记录地理数据，往往导致数据库表单的设计过于复杂，而且需要大量的join来实现复杂数据的表达，而对象型为了满足多尺度的要求，可以对统一对象创建多个类并根据类层次结构来进行管理。  但由于它没有原型数据库做基础，完全从零开始实现一个对象型数据库，虽然足够灵活，但也开发费时费力，开发者需要手动实现数据库功能，自主开发需要的扩展数据类型，同时还要考虑性能优化问题，而且与其他数据库的转接兼容也相对麻烦。所以这种思路虽然很强大，但受到的限制还是太多。缺乏数学模型的支持，复杂难用，成熟度低，最后没有撼动关系型的地位。 |

|  |
| --- |
| 对象关系型数据库  对象关系数据模型是从关系数据模型出发，一想对象思维，一关系数据库和sql为基础的扩展关系模型，具有扩展复杂数据类型和允许用户自定义函数功能。她是具有面向对象特性，收通用性强的sql3标准支持。对象关系型数据库管理系统既满足了传统关系型数据库用户的要求，同时又能次吃具有复杂数据管理需要的应用，具有强大的查询语言功能。  关系数据库只能做到对复杂数据的存储和对简单数据的查询。而对象关系数据库的最大功能是能做到对复杂数据的查询且速度很快，具有可扩充性。  特点：   * 扩充数据类型 * 支持复杂对象 * 支持集继承概念 * 提供通用的规则系统 |

|  |
| --- |
| 面向对象数据库优势 面向对象数据库相比于传统数据库（特别是关系数据 库）的优势主要体现在以下几点： 2.1 面向对象数据库更易于掌握 面向对象数据库从面向程序设计语言的扩充着手使之 成为基于面向对象程序设计语言的面向对象数据库。例如： ONTOS、ORION 等，它们均是 C++的扩充，熟悉 C++的人 均能很方便地掌握并使用这类系统。 2.2 支持复杂应用 面向对象模型主要用于复杂应用，应用中数据的复杂性 越高，数据间相互的关系越复杂， 性能的提高也越大。相 关对象的寻找可通过类层次或其他相关关系来完成，将特定对象放入高缓冲区或内存的技术，可通过预测用户或应用程序可能存取的类及其实例而得到优化。当数据复杂性较高时，聚集和缓冲技术对性能的提高更显著，而这正是关系数 据库无法达到的。 2.3 存储大型数据结构 面向对象数据库不仅能存储复杂的应用程序，而且还能存储较大的数据结构。尽管关系数据库支持大量的元组，但单个元组的大小受到限制，虽然有些关系数据库在这方面有所放松，但带来了数据库重组与管理低效等问题，面向对象数据库不会因为有大量对象的存在而降低了性能。因为不管对象的特性有多复杂，应用程序都没有必要把对象分离或装 配。 2.4 直接引用对象 面向对象的数据库支持对象的直接引用，这样做不仅减少了系统的数据冗余，提供了数据共享能力，有利于数据完整性维护，同时还大大提高了搜索和导航访问能力。在关系数据库中则相反，复杂的数据集必须由应用程序组装，如低 效的连接运算。 2.5 长事务处理 在传统的意义下，设计事务不是原子的。因为事务具有较长的持续时间，所以事务的部分结果应该是可见的。在事务进行阶段，系统可以通过发布一个修改令牌来实现单一设计者对数据的更新，也可以通过发布多个浏览令牌来实现多个设计者对数据的同时浏览。系统能够实现对系统崩溃点的保存，进行系统恢复。除此之外，面向对象的数据库将对不同版本的支持和事务管理集成为一体，以此来简化当前的存 取和恢复。 2.6 优良的应用开发环境 从应用程序的开发环境来看，面向对象数据库又显示出其优势。关系数据库的应用开发离不开数据操作语言和相应的宿主语言等，程序员必须同时掌握这两种语言，两者缺一 不可， 否则不能建立完整的应用程序而面向对象数据库的 应用开发并没有这些限制。 2.7 简化并发控制 在关系数据库中，并发控制理论已经很成熟了，但实现起来却比较复杂，应用程序必须显式地对数据进行封锁，封锁类型也要考虑。对某张表表或其中一部分元组的封锁往往带来对相关的其他表或元组的封锁而面向对象数据库的并发控制以对象为封锁单位，相关数据由对象本身的结构决定，一些控制可由对象的方法和触发器完成，并发控制简单 而有效。 2.8 实现了无缝连接 面向对象数据库的产生主要是为了解决“阻抗失配”，它 强调高级程序设计语言与数据库的无缝连接。由于实现了无缝连接，使得面向对象数据库能够支持非常复杂的数据模 型，从而特别适用于工程设计领域。  3.1 性能方面 由于面向对象数据库中数据被存放在许多地方，因此，有效对象聚集是性能好坏的关键因素。提高面向对象数据库的性能尤其是在分布式环境中的性能的一种方法是把访问数据库的应用程序也看作是对象，以使它们在数据库中可像数据对象那样到处移动。在进行查询时，数据库可以选择将 数据移至程序还是将程序移至数据。 3.2 模式修改 当需要面向对象数据库的升级或新版本时，数据库的模 式修改或重构将是个问题。 3.3 标准化 标准化和形式化是面向对象数据库系统研究和发展的一个重要方向。几年来，人们在核心面向对象概念方面基本达成了共识，但在面向对象数据模型的其它方面，如体系结构、编程接口语言上的理解尚未达到一致。有待于在系统研 制和应用过程中进行标准化。 3.4 技术须趋向成熟 190 刘江枫：面向对象数据库技术及其前景分析 我们知道，面向对象数据库的发展有赖于面向对象语言的发展，随着面向对象语言的发展，面向对象数据库将趋于成熟 |

|  |
| --- |
| **Db4o特点**  **性能问题**  db4o 是一个开源的纯面向对象数据库引擎，对于 Java 与 .NET 开发者来说都是一个简单易用的对象持久化工具，使用简单。同时，db4o 已经被第三方验证为具有优秀性能的面向对象数据库， 下面的基准测试图对 db4o 和一些传统的持久方案进行了比较。db4o 在这次比较中排名第二，仅仅落后于JDBC。通过图 1 的基准测试结果，值得我们细细品味的是采用 Hibernate/HSQLDB 的方案和 JDBC/HSQLDB 的方案在性能方面有着显著差距，这也证实了业界对 Hibernate 的担忧。而 db4o 的优异性能，让我们相信： 更 OO 并不一定会牺牲性能。  å¾1. HSQLDB åºåæµè¯  图1. HSQLDB 基准测试  图2为 db4o 官方公布的基准测试数据，db4o 比采用 Hibernate/MySQL 方案在某些测试线路上速度高出 44 倍之多！并且安装简单，仅仅需要 400Kb 左右的 .jar 或 .dll 库文件。  å¾2. db4o å®æ¹åºåæµè¯æ°æ®  图2. db4o 官方基准测试数据  **零管理，易嵌入，原生数据库，支持多种平台**  同时，db4o 的一个特点就是**无需 DBA** 的管理（零管理），占用资源很小（使用 db4o 仅需引入 400 多 k 的 jar 文件或是 dll 文件，**内存消耗极小**。），这很适合**嵌入式**应用以及 Cache 应用。同时，db4o 是 100% 原生的面向对象数据库，**直接使用编程语言来操作数据库**。程序员无需进行 OR 映射来存储对象，大大节省了程序员在存储数据的开发时间。支持多种平台。**db4o 支持**从 Java 1.1 到 Java 5.0，此外还支持 .NET 、 CompactFramework 、 Mono 等 .NET 平台，也可以运行在 CDC 、 PersonalProfile 、 Symbian 、 Savaje 以及 Zaurus 这种支持反射的 J2ME 方言环境中，还可以运行在 CLDC 、 MIDP 、 RIM/Blackberry 、 Palm OS 这种不支持反射的 J2ME 环境中。所以自从 db4o 发布以来，迅速吸引了大批用户将 db4o 用于各种各样的嵌入式系统，包括流动软件、医疗设备和实时控制系统。  或许开发者会问，如果现有的应用环境已经有了[关系型数据库](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93" \t "_blank)怎么办？没关系，db4o 的 dRS（db4o Replication System）可实现 db4o 与关系型数据库的双向同步（复制） |

|  |
| --- |
| 参考文献：   * **1995 – The OODBMS Manifesto**   Malcolm Atkinson et. al. release “[The Object-Oriented Database System Manifesto](http://www.cs.cmu.edu/afs/cs.cmu.edu/user/clamen/OODBMS/Manifesto/)”   * **Introduction to Object-Oriented Databases (Computer Systems Series)**   [Introduction to Object-Oriented Databases (Computer Systems Series)](https://img1.doubanio.com/view/subject/l/public/s4071129.jpg)  作者: [Won Kim](https://book.douban.com/search/Won%20Kim)  出版社: The MIT Press 出版年: 1990-11-19 页数: 256 定价: USD 30.00 装帧: Paperback ISBN: 9780262  有两篇论文专门讲解了 NQ 的基本概念和设计思路，分别是 《Cook/Rosenberger，持久对象原生数据库查询语言》 和 《Cook/Rai，Safe Query Objects: Statically Typed Objects as Remotely Executable Queries》。 |