

1.0 版本

# db4o 复制系统 (dRS)

db4o 复制系统 (dRS)，基于 [Hibernate](#) 的强大能力，允许用户在领先的开源对象数据库 [db4o](#) 的分布式实例和类似 [Oracle](#)、[MySQL](#) 这样的所有普通关系型数据库之间构建双向对象同步应用程序。当处理现有 IT 环境中关系型数据库技术的数据一致性时，dRS 使 db4o 的原生对象持久化体系能适用于所有的 Java 和 .NET 开发者。

dRS 100% 的面向对象，尤其适合敏捷企业开发和软件制造商的产品快速更替，以及大多数的移动业务环境。相比上世纪 80 年代利用关系型数据库技术设计的集中式数据存储，在复制数据方面软件管理者能削减 90% 的开发费用和时间，并且关系型数据库与 Java、.NET 面向对象软件体系也有着本质的不协调。

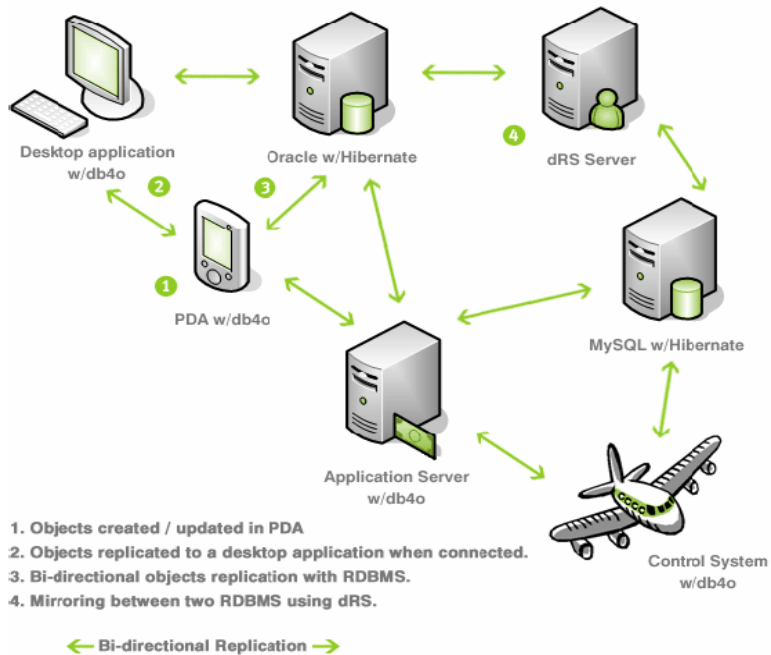
db4o 是开源对象数据库，它能使 Java 与 .NET 开发者大幅削减开发时间以及开支，并能达到史无前例的性能。

db4o 原生对象数据库引擎独一无二的设计使它成为了嵌入式设备和装置、移动套装软件或桌面平台或实时控制系统的理想选择——简而言之：适用于所有不需要 DBA 的环境。

db4o 对象数据库的设计目标是嵌入进分布式应用软件以及需要软件支持的设备上，尤其是资源受限和性能要求苛刻以及无需 DBA 的场合。列举一下 db4o 客户的应用 [Indra 高速列车控制系统](#)，[波音 P8-A 军用飞机](#)，[能运行在现有 PDA 上的 Easterndata 移动入户送货系统](#)。在大部分案例中，客户端 db4o 数据库实例部分地连接到运行了 Oracle 和 MySQL 等关系型数据库 (RDBMS) 的企业级服务器端环境中。基于这些连接，在分布式数据集之间，dRS 提供了自动的，单向或双向的数据同步。

db4o 复制系统 (dRS)，适用于 db4o-到-Hibernate/RDBMS、db4o-到-db4o、以及 Hibernate/RDBMS-到-Hibernate/RDBMS 的复制。dRS 目前版本是 1.0 并运行在 Java 1.2 或更高版本上。

基于 GPL 许可，你可以到 db4o 下载中心 (<http://www.db4o.com/community/ontheroad/downloadcenter/>) 免费下载，付费商业授权 (非 GPL) 可发邮件到 [sales@db4o.com](mailto:sales@db4o.com) 申请。



## 为企业赋予敏捷和移动特性

dRS 的面向对象复制手段是专门为敏捷企业的产品快速更替和大量的移动业务环境而设计。dRS 在简化分布式数据库实例同步以及频繁的重构软件代码方面做了很多优化，这在以前是无法同时完成的。

假设这样的范例，业务对象“customer”（包含消费者账户信息）需要在售货员的已联网 PDA 和后台企业服务器之间同步。哪些组成了对象“customer”，哪些又应该和父对象一起被复制，类中的父—子关系提供了自然的描述。为了完成这些业务对象的完整复制，开发者只需要连接两个数据库实例并请求更新“customer”对象再同步他们即可——仅仅需要一行代码：

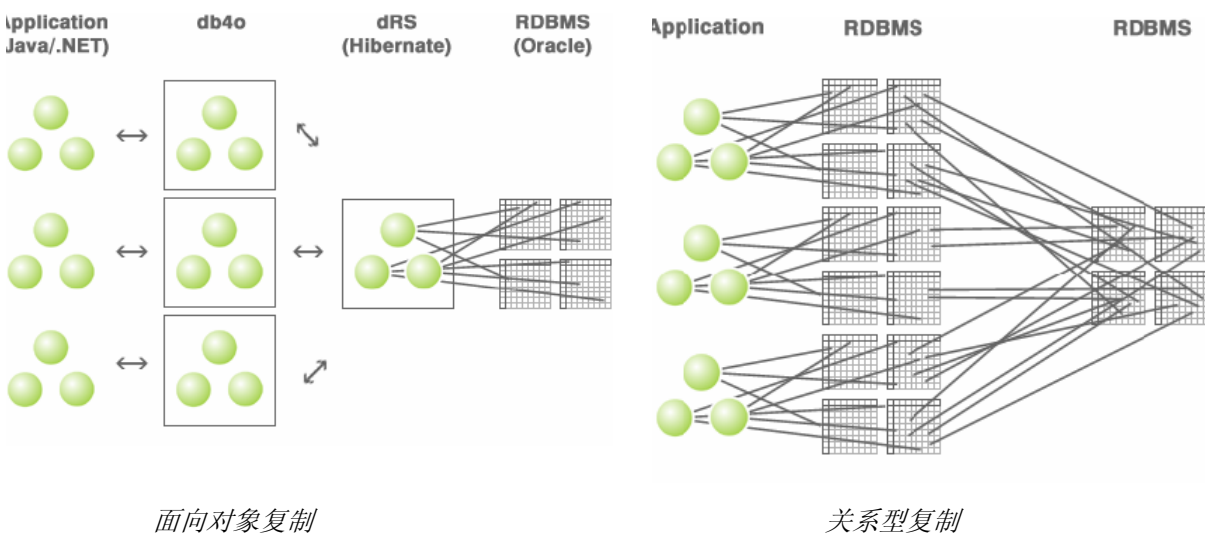
```
//Start a new ReplicationSession
ReplicationSession replication = new GenericReplicationSession(db4oProvider,
hibernateProvider);

//Query for objects changed from db4o
Iterator4 itor = db4oProvider.objectsChangedSinceLastReplication();

//Iterate changed objects, replicate them
while (itor.hasNext()) {

Customer customer = (Customer) itor.next();
replication.replicate(customer);
}
//commit the replication session
replication.commit();
```

通常的关系型技术，业务对象“customer”将包含很多有关联的表，这些表由硬连线（hard-wired）、静态自定义代码（static application code）以及由多行非原生 SQL 书写的字符串互相参考，导致重构非常棘手。



当正在争论这些问题时（参考 Scott Ambler 著，“[Agile Techniques for Object Databases](#)”），对象—关系的不协调匹配将会对软件自身的重构带来严重的制约，一旦分布式数据库需要进行数据同步的话，这个问题将会变得更加严重。

利用 **db4o** 原生面向对象持久化手段，开发者可以通过敏捷开发提高生产力，再也无需被原有的持久化方案所约束。因为自身的 **schema** 自动化上的进步。**db4o** 能够自动适应任何被更新的对象模型，使频繁的软件重构成为可能，甚至是高度分布式的数据体系。

因此，企业能更容易的适应快速变化的环境，无需担忧急速膨胀的花费和软件质量以及逐渐降低的可维护性。无需遵循“不要改变正在运行的系统”原则，企业部署 **db4objects** 的原生对象持久化方案，能抑制用户日益增长的需求给企业带来的冲击，并提供更加丰富的功能和细化的软件解决方案，尤其适合移动方面的应用。

### 特色：面向对象的同步冲突解决方案

**dRS** 独一无二的特色是它的应用程序数据驱动（**application-data-driven**）冲突解决方案。应用程序数据模型应该是唯一存储面向数据业务逻辑的地方，可作为解决冲突问题的方案，这是不二的选择。

当复制会话断开后，某个对象在断开的数据库实例上同时被修改就会产生同步冲突。一般情况用户会选择模拟业务规则从源头到目标重新更替一次，或者跳过冲突对象，或者停止复制以防脏数据，或者嵌入通知规范触发用户行为或其他业务规则来解决冲突。

```
ConflictResolver resolver = new ConflictResolver() {
    public Object resolve (Object copyA, Object copyB) {
        if(((Customer)copyA).changedBy().isCustomerOwner()
            return copyA;
        if(((Customer)copyB).changedBy().isCustomerOwner()
            return copyB;
        return null;
    }
};
```

以上范例，业务对象“**customer**”在售货员手持 **PDA**（离线状态）和后台 **CRM** 数据库中都有一个可修改的电话号码。业务逻辑需要知道售货员是否已确认客户的“身份”，以便能覆盖后台数据，除非有特例发生（比如，对售货员来说，用户的某些特殊权限或例如自助服务交易等这些更“高级”的身份）。所以业务逻辑不能被固化（比如用关系型复制技术）在应用程序中，但是可以和对象一起存储，让它有足够的“聪明”自动同步所有相关联的和最接近的业务规则。

## 安装

安装 *db4o 复制系统* 是为了更简洁地操作关系型数据库。只需从 *db4o* 下载中心 [www.db4o.com/community/ontheroad/downloadcenter/](http://www.db4o.com/community/ontheroad/downloadcenter/) 下载 *db4o* 核心包以及包括 *Hibernate* 在内的 *db4o 复制系统*。安装并配置 *Hibernate* 连接到你的关系型数据库，接着为持久类创建 *Hibernate* 映射文件。

如果在你的关系型数据库中已经有了数据，*dRS* 将更新 *schema* 以便记录对象版本。如果你的数据库是空的，*dRS* 系统将自动创建存储对象的表。接着，启动复制会话，传递你想要复制的对象。最后提交——进行两个数据库之间的同步。更多信息请参考用户指南，其中包含了大量的范例，能帮助你快速学习，并在 10 分钟内运行成功。