**初次部署到生产环境前的敏捷数据方法**

在还没有部署到生产环境前，数据库元结构的的变化还是很方便的。可以引入敏捷数据方法，避免过度设计和提前固化。

具体做法：

1. 先有一个大的愿景，团队的每个成员都知道需要实现的功能，领域的初略结构（即我们要设计的东西是干什么用的，有哪些大致的对象，对象之间的关系是什么），以电网结构为例，我们都知道电网结构大大致构成，有线路、变压器、低压线路、台区、变电站、开关站等等对象，我们知道我们的目标是设计一个采集系统，我们还不能确定是否要设计电能损耗的计算。
2. 只实现目前功能需要的数据结构。我们决定先不实现电能损耗的计算，所以可以先设计实现当前功能所需要的最低要求的对象。
3. 注意好的实践。比如遵循范式话是关系数据库的好的实践。
4. 保留继续演变的可能性，当新的需求要求增加新的对象时，我们对原来的设计进行扩充，修改原来的设计。比如我们原来设计的“测量点”表，包括了电表的表地址、所属采集设备等等信息，当要实现换表功能时，需要将原来的表至少拆分成两个部分，将电表相关的信息分拆出来构成独立的表，为了表示换表这个事实，还需要增加一个电表变更的历史表。（历史表的一般做法或者说好的实践就是包括起始时间和结束时间，而新记录的结束时间还不知道，这时指定一个很大的时间常量来表示Null是一个好的实践，可以加快查询的速度并简化查询——因为SQL的null比较在不同的数据库中是不一样的，大部分是一个散值逻辑（是、否、未知或未定义），使用大的常量可以避免三值逻辑。
5. 每次演变都要包括文档和脚本。文档对变更加以说明，记录变更的目的，以及数据结构的版本。脚本对变更的步骤用sql脚本的形式表达出来，同时更新版本号。
6. 需要有一个版本记录表，记录版本变更历史，包括版本号、实现者、说明文档名。

好处：

可以避免过度设计，不需要的功能，不会出现在表结构中。

可以保证数据结构的质量，当需要进行扩充变更时，都是为了实现新的功能才进行变更的。增加新的表结构前都先对原来的元数据进行了重构。重构总是遵循好的实践。

变通：

可以将标准的参考设计作为参考，但是我们又不是原封不动的原样引用，而是根据自己的需要加以取舍。

敏捷数据的其他实践：

我们省略了重构的测试环节。测试是很重要的，但是作为初次实践敏捷数据这种设计方法，我们可以暂时先不引入测试环节，而靠经验来保证重构的质量。