Oracle DBA 使用手册

--设计篇

DOYENSEER

doyenseer@sohu.com

前言

研究数据库主要涉及到三个方面：一是数据库原理，主要研究数据库的基本概念，优化算法和使用数据库的语言；二是数据库设计方法学，主要研究如何设计数据库和建立用户使用的数据库系统；三是数据库应用，即各种行业的数据库应用管理系统。

本文档重点介绍数据库设计方法学，根据 Oracle 系统的特点探讨一下设计和创建数据库的方法，通过简洁而实用的设计方法，构建稳定，高效的数据库应用管理系统。

设计目标

本节主题

- 数据库设计的重要性
- 良好设计的要求
- 良好设计的优点
- 数据库设计的方法

数据库设计的重要性

数据库设计是保证数据库中数据的一致性，完整性和正确性的重要过程。如果数据库设计的不合理，会造成应用系统检索信息困难，甚至得不到正确的检索信息。从而会影响公司的日常运营，或者公司的未来发展。

设想一下：如果你打算为自己建一所别墅，你首先要作的事是什么呢？显然，不会立即雇佣承包商，让他按照他的想法来为你建房（除非你有钱，不怕返工）。肯定要先雇设计师针对你的想法进行结构设计并画出蓝图，直到符合你的要求为止，然后再雇承包商建住宅。设计师会在蓝图上详细记录房子的大小，布局，各种设施，而承包商会根据蓝图估计工作量和需要的成本，雇佣劳工，购买材料，按照蓝图设计来建房。

数据库系统也是如此。可以把逻辑数据库设计看做结构蓝图，而物理数据库实现看做别墅。逻辑数据库设计描述了数据库的“大小”，“布局”和“各种设施”，它主要是确定事务的信息和操作需求。然后使用 RDBMS 软件(Oracle)构造逻辑数据库的物理实现。通过建表和表之间的联系和约束，完成数据库的构建。最后就可以通过应用系统，使用户可以方便快捷的访问数据库中的数据信息。

所以说数据库设计的好坏直接关系到数据库是否能产生正确的信息，良好的数据库设计对以后的管理和维护带来许多益处。

良好设计的要求

为了设计一个良好的，可靠的数据库结构，必须达到一定的设计要求：

- 保证数据库中的数据的有效性，完整性和正确性。数据完整性是数据库设计过程的最重要问题，是通过表，字段，关系以及业务规则等四个方面来实现的。
- 数据库中的数据应尽可能的小，数据库中的每个表都代表一个主题，由一些相关的字段组成。
- 数据库符合用户的业务规则。数据必须提供合法和正确的信息，这些信息对于用户都是有意义的。
- 数据库支持扩充。随着信息增长和用户需求的变化，数据库的结构应易于修改和扩展。

良好设计的优点

经验告诉我们，花在设计可靠的数据库结构上的时间是值得的。因为有以下优点：

- 信息易于检索。可以很容易地创建查询，根据表之间的正确关系，可以很方便快捷的检索出正确信息。
- 数据易于修改。因为数据库中的重复字段尽可能少，所以修改表中某一个字段不会影响到表中其它字段以及其它表中的字段。
- 数据库结构易于管理。对数据库结构中对象的修改不会对数据库的整体性能造成不利的影响。
- 用户应用易于开发。可以更多的关注业务系统的软件开发，不必疲于应对糟糕设计带来的各种问题。

数据库设计的方法

数据库的设计可分为两大部分：一数据库的逻辑设计，即数据库管理系统要处理的数据库逻辑结构建模，包括概念建模和逻辑建模；二数据库的物理设计，即在逻辑结构已确定的前提下设计数据库的存储结构，也就是物理建模。整个设计过程可分为 5 个阶段：需求分析，概念设计，逻辑设计，物理设计，数据库实施。

设计阶段

本节主题

- 需求分析
- 概念设计
- 逻辑设计
- 物理设计
- 数据库实施

需求分析

需求分析阶段涉及考察待建模的企业，访问用户和管理人员，了解当前使用的系统或收集与业务有关的数据，比如各种表格，统计报表等资料。分析用户当前和未来的需求（包括数据和处理）。需求分析是整个设计过程的基础，是最困难，最耗精力的一步（主要是怎么与客户有效沟通交流）。需求分析做不好，很有可能导致整个设计返工。不要考虑如何使自己走捷径，要牢记**“没有时间做好，就总有时间重做”**的经验教训。

在系统分析阶段，设计者和用户双方要密切合作，共同收集和分析数据管理中信息的内容和用户对信息处理的要求。主要从以下几方面调研：

- 机构与人员的情况

与系统相关的各级机构和组织的名称是什么？分几级管理？各级之间的相互关系如何？谁管谁？管理流程怎样?

各级机构的任务是什么？管理范围多大？管理办法（政策、规定）是什么？

各级机构的人员安排是什么样的?各自的责任的是什么？谁负责制订政策？谁负责执行日常管理？谁进行实际操作?

进行机构改革的趋向？管理政策、管理办法是否会发生变化？怎样变?

- 数据的情况

现在用什么方式描述所管理的对象？是卡片记录，或表格登记，或记帐本，还是档案等。

所有有关的数据自然形成了哪些记录型？各自的名称是什么?每种记录型可能的记录值有多少？哪些记录型语义上靠得近些，形成相对独立的区域？

每种记录型包含什么属性？它们的名称、类型、取值范围？哪些属性被多个记录型重复地使用?属性之间的信息关系是怎样的？如依赖关系、存在性关系、决定关系、相互约束关系等。

记录型及属性是否会随着时间的推移发生变化？会增加或减少记录型、属性吗？会增加或减少它们之间的信息关系吗?

各种记录型包含的记录数量变化的趋势，随时间的变化率多大？

- 操作的情况

列出用户要求的各种报表。各自的格式，有哪些统计项？涉及哪些记录型、属性？多长时间进行一次报表统计？

列出各项查询要求。它们涉及哪些记录型、属性?经常查询的是什么？查询的频率是多少？有哪些偶尔的查询或统计？查询要求的响应时间多长？

预测未来可能要求的查询、报表、格式要求、涉及信息等。

各项查询、报表涉及的记录型、属性等信息在前面是否已调查？如果没有，则进行必要的补充调查。

各机构对各类数据是否有专门要求和保密措施?如：对某记录型只能读取而不能修改，对某些记录型只准某些人读、写和修改等。

其实，大多数数据库项目都不是从头开始建立的，在设计一个新数据库时，不但应该仔细研究业务需求而且还要考察现有的系统。从现有系统中可以找出旧系统的优点和缺点，需求中被忽略的问题等有价值的信息。分析这些信息，充分完善当前需求。

总之，分析结果是否能准确地反映实际系统的业务流程情况和用户对数据库的要求，会直接影响以后各阶段的工作，并影响到数据库系统将来运行的效率。因此分析阶段是整个数据库设计的基础。

注：分析阶段的难点在于如何与客户沟通，沟通是从参与者获得信息的工具，就是得到关于数据库系统得新想法，澄清不理解的问题，所以沟通的效果也会直接影响到需求分析的质量。沟通是一项技能，非常讲究方法和技巧的，本文档不对此问题进行深入探讨（可以 email 联系作者），在这里提出 4 点建议以供参考:

1. 要让参与者明白你的想法。在每次讨论前，都应该让其他参与者有时间了解这次讨论的主题。避免有些参与者过于谨慎，怕陷入难堪，而不想或不知道提出自己的观点。
2. 要让参与者相信他们的意见是有价值的。对参与者的任何建议都应该认真对待，并且有价值的观点，也应该真诚地表达谢意。这样才能提高参与者的积极性，并与之逐步建立起最基本的信任感。
3. 在有分歧时要让参与者知道你是决策者。在讨论问题时难免会有分歧，僵持不下的局面，作为数据库系统设计者，应该客观做出有利于数据库结构设计的决定，避免紧张气氛延续下去，从而影响分析阶段的进度。
4. 对待不同层次的参与者，应该分开讨论。因为不同层次的人看待问题的角度不同，分开可以避免更多争执。

概念设计

数据库概念结构的设计是数据库设计过程中非常重要的环节。概念设计是独立于硬件和软件系统的，它的目的是以用户可以理解的形式来表达信息的流程，从而方便与客户交流沟通。

良好的概念结构设计应该满足以下要求：

1. 能准确反映用户的需求，信息描述具有一致性，没有自相矛盾，并且不含语义二义性。
2. 尽可能消除冗余的数据和不必要的数据关系，保证最小冗余。
3. 具有很强的扩展性，当用户需求发生变化时，可方便更改。
4. 具有一定的可靠性，从而不会影响对数据处理的要求。

概念设计是通过对需求阶段得到的信息进行汇总，归纳并做出合理的抽象解释，形成一个独立于具体DBMS的数据模型，它定义了一组规则，包括数据结构的生成规则，数据完整性的约束规则以及在数据上的操作规则。通常用实体—联系数据模型（E-R Model）来描述。

E-R 数据模型的结构生成规则定义了三种基本结构：实体、属性以及联系。实体用于描述客观实体的整体特征，一个客观实体内信息之间的关系应是比较紧密的。联系描述客观实体之间的语义联系。不同实体之间的语义联系应是较松散的。属性描述客观实体及其联系的

特征。在 E-R 数据模型中，每一个实体、联系和属性都必须有一个可唯一标识的名字。它们分别用带有名字的方框、棱形框和椭圆框来表示。根据一个特定的信息系统的需求，把有关的实体、联系和属性按实际的情况连接起来，并标明它们之间的对应关系，就得到一个具体的 E-R 图，即该信息系统概念模式的 E-R 图表示。

为保证概念模式设计正确性，对概念模式的有效性检查是必不可少的步骤。检查的主要内容是：

- 减少冗余度　应当最大可能遵守第三范式。简单的理解三范式：
  第一范式：1NF 是对属性的原子性约束，要求属性具有原子性，不可再分解；
  第二范式：2NF 是对记录的惟一性约束，要求记录有惟一标识，即实体的惟一性；
  第三范式：3NF 是对字段冗余性的约束，即任何字段不能由其他字段派生出来，它要求字段没有冗余。
  但是，没有冗余的数据库未必是最好的数据库，有时为了提高运行效率，就必须降低范式标准，适当保留冗余数据。
- 要善于识别与正确处理实体间的关系　　尽量消除实体之间的多对多的关系。
- 各项描述是否清楚，是否存在语义上的二义性。
- 概念结构是否满足用户的需求。

逻辑设计

逻辑结构设计是把概念结构转换为数据库管理系统能处理的数据模型，并对其进行优化。将概念结构（E-R 图）转换成具体的数据库产品(Oracle)支持的数据模型，就是把实体和实体之间的关系转换为相应的二维关系表，并进一步规范和优化：

- 规范化
  定义符合 Oracle 约束条件的规则，满足数据完整性；
  确定有效的主键和索引，尽量不要选择需编辑的字段作键值，避免使用复合键；
  选择适当的存储参数；
  为了安全性建立必要的视图；
- 优化
  减少过多的关系，关系越多，开销越大，效率越低；
  要考虑到未来的各种变化。

数据库的逻辑设计一般遵循以下原则：

1. 对象命名规范
   - 表名

   前缀为 tbl_。表名称为前缀_分类代码_有特征含义的单词或缩写组成。总长度不超过 30 个字符。将所有表根据应用系统功能分类，比如资源表，则分类代码为 RES。
   - 字段名

   前缀为表名的首字母组合。字段名为前缀_有特征含义的单词或缩写组成。总长度不超过 30 个字符。
   - 主键

   前缀为 pk_。主键名称为前缀_表名(简写)_构成的字段名。如果复合主键的构成字段较多，则只包含第一个字段。
   - 外键

前缀为 fk_。外键名称为前缀_表名(简写)_外键字段名_外键表名(简写)_外键表构成的字段名。

- 索引

前缀为 idx_。索引名称为前缀_表名_构成的字段名。如果复合索引的构成字段较多，则只包含第一个字段。

- 视图

前缀为 V_。索引名称为前缀_有特征含义的单词或缩写组成。

- 序列

前缀为 seq_。序列名称为前缀_有特征含义的单词或缩写组成。如果是为某张表创建序列，也可以为前缀_表名。

- 函数

前缀为 Fun_。函数名称为前缀_有特征含义的单词或缩写组成。

- 过程

前缀为 prc_。过程名称为前缀_有特征含义的单词或缩写组成。

- 触发器

前缀为 trg_。触发器名称为前缀_有特征含义的单词或缩写组成_触发形式（增，删，改）。

- 包

前缀为 pkg_。包名称为前缀_有特征含义的单词或缩写组成。

- 表空间

前缀为 tbs_。表空间名称为前缀_有特征含义的单词或缩写组成。

- 数据文件

<表空间名>nn.dbf 。nn =1，2，3，4，…等。

- 数据库链接

前缀为 dbl_。数据库链接名称为前缀_链接数据库名。

2. 语句编写规范

命名应该使用英文单词，避免使用拼音，特别不应该使用拼音简写。命名不允许使用中文或者特殊字符。英文单词使用用对象本身意义相对或相近的单词。选择最简单或最通用的单词。不能使用毫不相干的单词来命名。

当一个单词不能表达对象含义时，用词组组合，如果组合太长时，采用用简或缩写，缩写要基本能表达原单词的意义。当出现对象名重名时，是不同类型对象时，加类型前缀或后缀以示区别。

名称一律大写，以方便不同数据库移植，以及避免程序调用问题。命名的各单词之间可以使用下划线进行分隔。

命名不允许使用 SQL 保留字。

3. 数据类型选取

Oracle 数据类型表：

数据类型	描述
char(n)	定长字符串，n 字节长，n<=2000。字节如果不指定长度，缺省为 1 个字节长（一个汉字为 2 字节）
varchar2(n)	可变长的字符串，具体定义时指明最大长度 n，n<=4000。这种数据类型可以放数字、字母以及 ASCII 码字符集(或者 EBCDIC 等数据库系统接受的字符集标准)中的所有符号。
nchar(n)	用来存储 Unicode 字符集的定长字符型数据，n<=1000。
nvarchar2(n)	用来存储 Unicode 字符集的变长字符型数据，n<=1000。
number(m,n)	可变长的数值列，允许 0、正值及负值，m 是所有有效数字的位数，n 是小数点以后的位数。 如：number(5,2)，则这个字段的最大值是 99,999，如果数值超出了位数限制就会被截取多余的位数。 如：number(5,2)，但在一行数据中的这个字段输入 575.316，则真正保存到字段中的数值是 575.32。 如：number(3,0)，输入 575.316，真正保存的数据是 575。
date	用来存储日期数据
long	可变长字符列，最大长度限制是 2GB，用于不需要作字符串搜索的长串数据，如果要进行字符搜索就要用 varchar2 类型。 long 是一种较老的数据类型，将来会逐渐被 BLOB、CLOB、NCLOB 等大的对象数据类型所取代。
raw(n)	可变长二进制数据，在具体定义字段的时候必须指明最大长度 n，Oracle 用这种格式来保存较小的图形文件或带格式的文本文件，如 Miceosoft Word 文档。 raw 是一种较老的数据类型，将来会逐渐被 BLOB、CLOB、NCLOB 等大的对象数据类型所取代。
long raw	可变长二进制数据，最大长度是 2GB。Oracle 用这种格式来保存较大的图形文件或带格式的文本文件，如 Miceosoft Word 文档，以及音频、视频等非文本文件。 在同一张表中不能同时有 long 类型和 long raw 类型，long raw 也是一种较老的数据类型，将来会逐渐被 BLOB、CLOB、NCLOB 等大的对象数据类型所取代。
blob	用来存储多达 4GB 的非结构化的二进制数据
clob	用来存储多达 4GB 的字符数据
nclob	用来存储多达 4GB 的 Unicode 字符数据
bfile	在数据库外部保存的大型二进制对象文件，最大长度是 4GB。 这种外部的 LOB 类型，通过数据库记录变化情况，但是数据的具体保存是在数据库外部进行的。 Oracle 8i 可以读取、查询 BFILE，但是不能写入。 大小由操作系统决定。
rowid	用来存储表中列的物理地址的二进制数据，占用固定的 10 个字节。

urowid	用来存储表示任何类型列地址的二进制数据。
float	用来存储浮点数。

4. 主键定义原则

每个表都应该定义主键，选择标准：

- 能够唯一标识表中某一行的属性或属性组；
- 应尽量选择更改较少的字段，作为主键；如果没有合适的，应采用序列来创建主键；
- 应尽量少用复合主键；

5. 索引创建原则

索引的建立必须慎重，对每个索引的必要性都应该经过仔细分析，要有建立的依据。一般遵循以下原则：

- 经常出现在 Where 子句中的字段，特别是大表的字段，应该建立索引；
- 经常与其他表进行连接的表，在连接字段上应该建立索引；
- 数据量超过 300 的表应该有索引；
- 索引应该建在选择性高的字段上；
- 索引应该建在小字段上，对于大的文本字段甚至超长字段，不要建索引；
- 频繁进行数据操作的表，不要建立太多的索引；
- 删除无用的索引，避免对执行计划造成负面影响；
- 复合索引的建立需要进行仔细分析；尽量考虑用单字段索引代替；
- 正确选择复合索引中的主列字段，一般是选择性较好的字段；
- 复合索引的几个字段是否经常同时以 AND 方式出现在 Where 子句中？单字段查询是否极少甚至没有？如果是，则可以建立复合索引；否则考虑单字段索引；
- 如果复合索引中包含的字段经常单独出现在 Where 子句中，则分解为多个单字段索引；
- 如果复合索引所包含的字段超过 3 个，那么仔细考虑其必要性，考虑减少复合的字段；
- 如果既有单字段索引，又有这几个字段上的复合索引，一般可以删除复合索引；

物理设计

- 服务器操作系统

服务器是数据库系统主要的运行环境，如何选择服务器通常要考虑使用者对系统响应时间的苛刻要求、动态变化和难以预计的未来负荷、未知的升级周期等特殊的情况。所以在选择服务器的主机时，除了遵循高性能、网络吞吐量大、可靠性和可用性好这样一些基本原则外，还应当注意以下问题：

1. 可靠性及安全性

系统支持 24 小时不间断地运行，系统处理的数据很多是重要或敏感数据，所以还要具备良好的安全性及容灾备份机制。

2. 可扩展性

当系统的负荷增大到一定程度时，需要对系统进行扩展。系统扩展方式主要有两种：一种是增加系统的配置，例如增加内存、更换 CPU、增加系统外部存储设备等；另一种是通过增加服务器，建立服务器集群来满足需求增长的要求。不管是哪种方式都要求系统具有可扩展的体系结构（例如冗余插槽、托架、电源等）。

3. 网络吞吐量及网络接口能力

在大量用户访问的情况下，仍然能够具备良好的响应时间。在选择过程中，需要注意选择网络适配器类型和接口都较多的产品。

4. 开放的体系结构

服务器是否具有开放的体系结构直接影响到系统日后的升级换代和维护问题。专用体系结构的计算机设备（例如 IBM AS／400、Compaq Tandam 系列）虽然有良好的整体性能，但在系统升级和维护方面有一定风险。

目前操作系统分成三大主流：UNIX 系统、Windows 系统和 Linux 系统。这三大类操作系统各有特点：UNIX 系统目前所占的份额最大，应用最为广泛，且安全可靠性经过时间和实践的检验，但是它的缺点就是版本太多，难以统一； Windows 系统是由 Microsoft 公司提供，价格较低，支持工具较多，缺点是开放性差，安全性不高；Linux 系统是一个开放的、免费的平台，发展速度迅猛，是一个全新的、富有生命力的系统。

对于中小型数据库系统，采用 linux 操作系统比较合适，对于数据库冗余要求负载均衡能力要求较高的系统，可以采用集群数据库的方案。

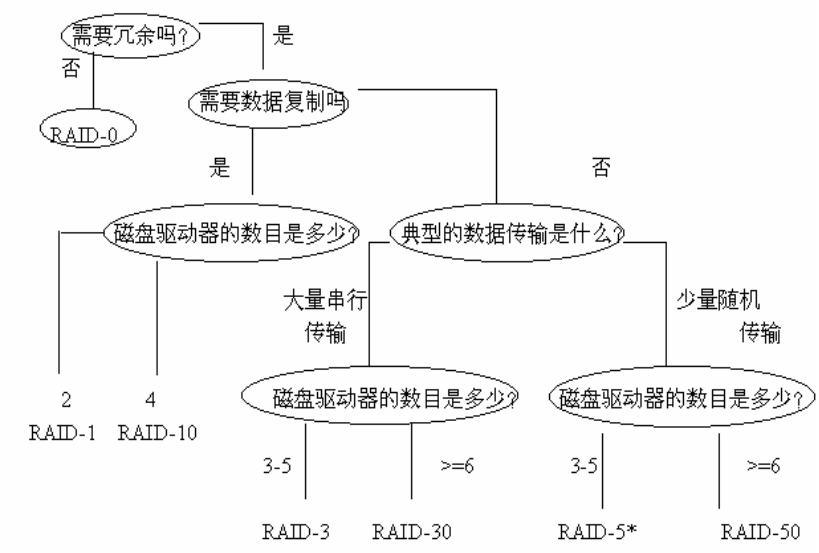
- 磁盘存储

数据库服务器对数据存储的性能以及安全性也有很高的要求。所以数据库服务器一般都采用 SCSI 硬盘作为存储介质（有些企业还采用光纤通道的硬盘），在数据存储方面通常都要实现 RAID。

RAID（冗余独立磁盘阵列）是一种通过结合磁盘阵列特性与数据条块化方法来提高数据可用率和系统可靠性，并预防磁盘故障的一种结构。其原理是利用 RAID 算法来计算丢失的信息数据，然后，再将找回的数据存放在冗余备用的磁盘上。RAID 技术的应用范围非常广泛，它还可以有效地预防各种类型的系统故障，比如说，凡是在数据写入的过程中发生的错误，一般只有在该数据被调用时才能被监测到。

RAID 级别可以通过软件或硬件实现。基于软件的 RAID 需要使用主机 CPU 周期和系统内存，从而增加了系统开销，直接影响系统的性能。磁盘阵列控制器把 RAID 的计算和操纵工作由软件移到了专门的硬件上，一般比软件实现 RAID 的系统性能要好。

RAID 级别的选择有三个主要因素:可用性(数据冗余)、性能和成本。如果不要求可用性，选择 RAID0 以获得最佳性能。如果可用性和性能是重要的而成本不是一个主要因素，则根据硬盘数量选择 RAID 1。如果可用性、成本和性能都同样重要，则根据一般的数据传输和硬盘的数量选择 RAID3、RAID5。也可以参考下面的流程图进行选择：



选择 RAID 级别图

对于 Oracle 数据库而言，很多 RAID 方案都不具备 Oracle 数据库所要求的高性能。大多数 Oracle DBA 都会选择一种结合了镜像（mirroring）和数据块分段（block-level striping）的 RAID 方案。

要注意的是，使用 RAID 并不能防止磁盘灾难性的故障。Oracle 专门推荐将所有的付诸使用的数据库都运行在 ARCHIVELOG 模式下，而不去过分依赖 RAID 的架构。Oracle 还建议定期进行 Oracle 的备份。

Oracle 数据库最常用的 RAID 架构：

1. RAID 0

RAID 0 通常指的是数据块的分段技术，它是在磁盘设备上实现 Oracle 数据库负载平衡的卓越方法，但是由于它没有提供数据的备份，因而完全无法提供高可用性。和手动的数据文件分段（你要手动将 Oracle 的表格空间分割放进小的数据文件里）不一样，Oracle 会利用 RAID 0 自动地将一个数据块进行分段并一次放进所有的磁盘设备里。在这种方式下，每个数据文件在每个磁盘上都存有其一部分内容，这样磁盘 I/O 的负载会变得非常平衡。

2. RAID 1

RAID 1 也叫做磁盘镜像。由于磁盘都是被相互复制，所以 RAID 1 可以做成双重或者三重镜像。根据 RAID 1 架构的设计，如果一个磁盘发生错误，那么 I/O 子系统就会自动切换到各个复制磁盘中的一个上，而不需要中断服务。Oracle 的专家会在要求高可用性的时候使用 RAID 1。对于三重镜像而言，Oracle 数据库的平均无故障时间（mean time to failure，MTTF）可以长达数十年。

3. RAID 0+1（RAID 10）

RAID 0+1 是数据块分段和磁盘镜像的组合。RAID 0+1 一出现就淘汰了 Oracle 这一层的分段技术，因为 RAID 0+1 的分段是在数据块这一层的，它分配表格块的方式是：每个磁盘上一个数据块，跨越每个磁盘设备。

RAID 0+1 也是一个远比（单纯的）分段技术好得多的替代方案，因为它将负载平均地分配到所有的磁盘设备上，也就是说负载的上升和降低都被平均地分配到了所有的

磁盘上。这就减轻了 Oracle 系统管理员在各个磁盘上手动地进行 Oracle 表格分段的负担。

4. RAID 5

RAID 5 是打造 Oracle 数据仓库和 OLAP 的好方法，因为在这里负载的速度不是很重要，而且系统 I/O 的主要职责在于只读的活动。

- 内存要求

对于 linux 操作系统下的数据库,由于在正常情况下 Oracle 对 SGA 的管理能力不超过 1.7G。所以总的物理内存在 4G 以下。SGA 的大小为物理内存的 50%—75%。对于 64 位的小型系统，Oracle 数据库对 SGA 的管理超过 2G 的限制，SGA 设计在一个合适的范围内：物理内存的 50%—70%，当 SGA 过大的时候会导致内存分页，影响系统性能。

- 交换区

当物理内存在 2G 以下的情况下，交换分区 swap 为物理内存的 3 倍，当物理内存>2G 的情况下，swap 大小为物理内存的 1—2 倍。

- 数据库 SID

数据库 SID 是唯一标志数据库的符号，命名长度不能超过 5 个字符。对于单节点数据库，以字符开头的 5 个长度以内字串作为 SID 的命名。对于集群数据库，当命名 SID 后，各节点 SID 自动命名为 SIDnn，其中 nn 为节点号：1，2，…,64。例如 rac1、rac2、rac24。

更多信息，请参看<< Oracle DBA 使用手册-安装篇>>

- 数据库类型选择

对于海量数据库系统，采用 data warehouse(数据仓库)的类型。对于小型数据库或 OLTP 类型的数据库，采用 Transaction Processing（事务处理）或通用类型。

- 数据库连接类型选择

Oracle 数据库有专用服务器连接类型和多线程服务器 MTS 连接类型。对于内存不是很高，并发用户连接数量比较多的 OLTP 服务,则 MTS 的连接方式比较合适，比如网站。而其他一般情况下最好选择专用服务器连接方式，比如批处理服务，数据仓库等等。

- 数据库 SGA 配置

SGA 随着不同的环境而不同，没有一种普通的最佳方案，在设置它直前要先考虑以下的几个方面：物理内存多大；操作系统是那种及占多大的内存，数据库系统是文件系统还是裸设备；数据库运行的模式。 SGA 包括：Fixed size、Variable size、Database Buffers、Redo Buffers。SGA 占有物理内存的比例没有严格的规定，只能遵从一般的规则：SGA 占据物理内存的 40%--70%左右

更多信息，请参看<< Oracle DBA 使用手册-安装篇>>

- 数据库字符集选择

一旦数据库创建后，数据库的字符集是不应改变的。因此，考虑使用哪一种字符集是十分重要的。数据库字符集应该是操作系统本地字符集的一个超集。存取数据库的客户使用的字符集将决定选择哪一个超集，即数据库字符集应该是所有客户字符集的超集。

- 数据库文件

数据库文件之间的 I/O 竞争是数据库之大忌, 所以对数据库规划之前要先对数据文件的 I/O 进行初步的评估，通常情况下：应用的表和索引通常应该被分配或分区到多个表空间中, 以降低单个数据文件的 I/O, 最好把每一种功能相同的区域对象建立单独的表空间，除了数据字典和系统回滚段外，其他数据对象应移出系统表空间。

- 数据库块大小

数据仓库，OLAP 类型： db_block_size 尽可能大，采用 16K 或 更高。

OLTP 类型： db_block_size 用比较小的取值范围，采用 4K 或 8K，建议 8K。

- 数据库控制文件

应该建立多个（至少 3 个）控制文件，并放在不同的物理位置，控制文件的位置在实例初始化参数文件中指定的。

- 数据库日志文件

日志文件的大小由数据库事务处理量决定，在设计过程中，确保每 20 分钟切换一个日志文件。所以对于批处理系统，日志文件大小为几百 M 到几 G 的大小。对于 OLTP 系统，日志文件大小为几百 M 以内。

对于批处理系统，日志文件组为 5—10 组；对于 OLTP 系统，日志文件组为 3—5 组，每组日志大小保持一致；对于集群数据库系统，每节点有各自独立的日志组。为了确保日志能够镜象作用，每日志组的成员为 2 个。

日志文件应该放在不同的物理位置。

- 数据库回滚表空间

在 Oracle9i 数据库中，设计 Undo 表空间取代以前版本的回滚段表空间。Undo 表空间大小的设计规范由以下公式计算：

Undospace = UR * UPS *db_block_size+ 冗余量

UR： 表示在 undo 中保持的最长时间数（秒），由数据库参数 UNDO_RETENTION 值决定。

UPS：表示在 undo 中，每秒产生的数据库块数量。

例如：在数据库中保留 2 小时的回退数据，假定每小时产生 200 个数据库块。则 Undospace = 2 * 3600 * 200 * 4K = 5.8G

- 数据库临时段表空间

数据库临时段表空间根据实际生产环境情况调整其大小，表空间属性为自动扩展。

- 数据库系统表空间

系统表空间大小 1G 左右，除了存放数据库数据字典的数据外，其他数据不得存储在系统表空间。

- 数据库非系统表空间

当表空间 大小小于操作系统对最大文件限制时，表空间由一个文件组成。如果表空间大小大于操作系统对最大文件限制时，该表空间由多个数据文件组成，表空间的总大小为估算为：

Tablespace + sum (数据段+索引段)*150%

表空间数据文件采用自动扩展的方式，扩展容量快大小按 2 的整数倍（1M、2M、4M、8M、16M、32M、64M）进行扩展，创建表空间时尽量采用 nologing 选项。表空间的最大限制一般采用 unlimited，除非确切知道表空间数据文件的最大使用范围。（一般 windows 3 2 位系统的文件最大 2 G， 6 4 位的 unix 系统系统文件最大 1 2 8 G,但也要注意文件格式设定的文件大小）,建议最大为 2G。表空间采用 local 管理方式。

数据库实施

数据库实施主要包括以下工作：

- 定义数据库结构

确定了数据库的逻辑结构与物理结构后，就可以用所选用的 DBMS 提供的数据定义语言（DDL）来严格描述数据库结构。

- 数据装载

数据库结构建立好后，就可以向数据库中装载数据了。组织数据入库是数据库实施阶段最主要的工作。对于数据量不是很大的小型系统，可以用人工方法完成数据的入库，其步骤为：

1. 筛选数据

需要装入数据库中的数据通常都分散在各个部门的数据文件或原始凭证中，所以首先必须把需要入库的数据筛选出来。

2. 转换数据格式

筛选出来的需要入库的数据，其格式往往不符合数据库要求，还需要进行转换。这种转换有时可能很复杂。

3. 输入数据

将转换好的数据输入计算机中。

4. 校验数据

检查输入的数据是否有误。

对于中大型系统，由于数据量极大，用人工方式组织数据入库将会耗费大量人力物力，而且很难保证数据的正确性。因此应该设计一个数据输入子系统由计算机辅助数据的入库工作。

- 编制与调试应用程序

数据库应用程序的设计应该与数据设计并行进行。在数据库实施阶段，当数据库结构建立好后，就可以开始编制与调试数据库的应用程序，也就是说，编制与调试应用程序是与组织数据入库同步进行的。调试应用程序时由于数据入库尚未完成，可先使用模拟数据。

- 数据库试运行

应用程序调试完成，并且已有一小部分数据入库后，就可以开始数据库的试运行。数据库试运行也称为联合调试，其主要工作包括：

1. 功能测试。即实际运行应用程序，执行对数据库的各种操作，测试应用程序的各种功能。
2. 性能测试。即测量系统的性能指标，分析是否符合设计目标。