数据库设计通用规范

引言

1. 目的
本规范主要为数据库设计和SQL规范。设计规范主要约束命名、表、索引设计等内容，请数据架构人员注意表设计规范，选择适合的表类型。SQL规范主要包含书写规范、索引使用、性能等需要SQL编写人员注意的内容。
2. 使用对象
本规范适用所有开发人员，DBA，在整个软件开发过程中必须遵循此规范。
3. DBA原则
重设计，轻需求，从项目设计阶段开始参与。不过多干预需求，针对重点部分详细Review ,非重点部分只检查规范性内容。务必对数据库对象，以及代码SQL语句严格把控，开发成形或投产后再要修改则难上加难。

数据库设计过程规范

综述

数据库设计的目的是为某个特定的应用系统设计最优的数据模式，使之可以满足各类业务要求，能够高效的存储数据。
在实现数据库对象前需要对系统的业务和数据进行分析，从而了解对象实体以及它们的关联关系，区分数据类型，范围，依赖关系（约束条件），然后进行逻辑数据模型设计和物理数据库设计，最后产生实际数据库。
数据库设计的完整过程包含：逻辑数据模型设计、物理数据库设计和实现数据库三个步骤，他们在逻辑上是连续的先后关系，实际设计中可以借助PowerDesigner等工具来进行全部三个阶段的工作。
按照业务模块分别构建数据库模型（ER图），进行物理数据库设计，完成后生成数据字典，生成相应数据库的DDL脚本，创建特定DBMS的数据库对象。

设计步骤

数据库设计步骤按下面的表格列表说明：

阶段 活动 输出 工具/说明
命名规范 1. 建立命名规范 命名规范 命名规范由产品线或架构组提出
逻辑数据模型设计 1. 识别实体类型
2. 确定实体属性
3. 定义主键
4. 确定实体间关系和外键约束
5. 确定实体完整性约束 数据字典
数据库模型（ER图） PowerDesigner等
物理数据库设计 1. 表设计
2. 字段的标准化和冗余
3. 创建索引和约束
4. 创建数据库文档 数据库说明文档 PowerDesigner等
数据库实现 1.创建DDL脚本
2.生成数据库对象。 DDL脚本
物理数据库 PowerDesigner等

数据库对象命名规范

基本要求

所有需要命名的对象需遵守以下规则：
• 数据库对象名称长度不应超过30个字符。
• 对象名称不应大小写混用，不使用双引号指定数据库对象名称的大小写拼写规则，并不含系统保留字。
• 分布式架构的OLTP系统尽量不使用外键、触发器、函数、存储过程、包、物化视图等有碍扩展性的数据库对象。
• 出于安全性考虑，尽量不使用dblink。

TABLE命名规范

<三位模块名>_可读性较强的表名。
英文字母和数字以及下划线组成。

COLUMN命名规范

使用可读性较强的列名,不超过15个字符。
英文字母和数字以及下划线组成。

PRIMARY KEY命名规范

1.主键列命名：<TABLE_NAME_SUFFIX>_ID,比如 MAS_TRACE表主键列为TRACE_ID。

2.主键约束命名：PK_<TABLE_NAME>。主键的格式为：前缀_表名。比如MAS_TRACE表主键约束命名为PK_MAS_TRACE。

INDEX命名规范

普通索引

IDX_<TABLE_NAME>_<COLUMN_NAME>
前缀为IDX_。索引的格式为：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，如果出现重名情况，则包含这些字段中的第n个字段即可。

唯一索引

IDX_UK_<TABLE_NAME>_<COLUMN_NAME>
前缀为IDX_UK_。索引的格式为：前缀_表名_构成的字段名。

函数索引

IDX_FUNC_<TABLE_NAME>_<NAME>
前缀为IDX_FUNC_。索引的格式为：前缀_表名_构成的特征表达字符。

VIEW命名规范

V_<VIEW_NAME>
前缀为V_。按业务操作命名视图，不超过15个字符。
英文字母和数字以及下划线组成。

逻辑数据模型设计规范

逻辑数据模型分析一个数据模型结构的数据要求，是一个设计说明，对实体、事件、关系及其细节提供文档，它告诉最终的数据库实现必需要有的信息和功能，但模型并不指出功能如何实现。
数据库逻辑设计的主要目的是设计数据库的逻辑结构。通过分析数据实体抽象出数据模型结构，为实体、事件、关系及其细节提供文档，和具体数据库产品、部署环境无关，主要反映业务逻辑，是一个设计说明。在这一阶段采用标准化工具（如，PowerDesigner）产生逻辑设计的实体关系模型（ER图）。
以下规范逻辑数据模型建立的原则和方法。

识别实体类型

通过与用户的讨论及分析，从用户详细需求（软件需求）或旧系统中识别实体及特性。
主要活动：
 确定实体名称：识别实体的名称
 确定文本描述：使用简短的文本描述实体，可以使用中文，该描述可以用于该实体对应的数据库表对象的注释。

确定实体属性（字段）

通过与用户的讨论及分析，从用户详细需求（软件需求）或旧系统中识别并确认每个实体的属性及属性完整性。一般来说在这个阶段的设计需要遵守第三范式。
主要活动：
 确定实体所有的属性。
 确定属性名称，并进行简短描述。
 为每个属性定义允许值的域，在这个阶段只定义基本类型和长度
 确定每个属性是必须的还是任选的

审计字段

需要添加审计字段的表，添加一下4个字段记录审计信息：
字段名 类型 是否为空 描述
created_by Varchar2(50) 否 创建者ID
created_date timestamp 否 创建时间
last_modified_by Varchar2(50) 是 修改者ID
last_modified_date timestamp 是 修改时间

确定主键

除弱实体外的每一个实体都必须有一个唯一标识属性的最小化集合，即主键。作为主键的字段应尽量少改变，含有的字段尽量少。好的主键应是毫无业务意义的字段。
主要活动：
 使用无意义字段，或从实体属性中选择一个或多个属性用于唯一标识一条记录，从而确定主键
数据库中，所有表都需要一个无业务含义的id作为主键（UUID），由客户端生成主键，速度更快。UUID对应数据库类型为varchar2（50）。

确定实体间关系和外键约束

识别上面设计出的逻辑实体之间的关系，不必强制建立关系，实体关系尽量限制在模块内部。
主要活动：
 确定实体间的关系，一对一还是一对多，还是从属关系
 对于实体间用于关联的属性，标记为外键，外键总是关联唯一的字段

原则上所有数据库表不设置外键约束，业务上的关联，采用表的业务意义上的主键（code）做关联。
如果主表存在有效期，则子表需要建立两个字段，一个是主表的UUID字段，一个是主表的code字段。
数据库的外键约束不需要建立。

确定实体完整性约束

关系完整性是为保证数据库中数据的正确性和相容性，对关系模型提出的某种约束条件或规则。上面的主键和外键也属于完整性约束。
主要活动：
 确定实体属性的完整性约束（是否允许空、允许的值、是否唯一等）

物理数据库设计规范

完成数据库逻辑模型设计后，应结合实际使用的关系型数据库系统（RDBMS）、实际业务需求、预估数据量、使用频率等对数据库逻辑模型进行实现，即进行物理数据库设计，并设计数据库的存储结构、存取方式。
在物理设计阶段把数据实体映射为数据库表，把实体属性映射为数据表的字段，对字段进行标准化与冗余处理，在数据库表上设计必要的索引和约束。
第三代数据库不使用sequence、存储过程，视图等特有特性。

表设计

逻辑设计中的数据实体对应的就是数据库表，在数据库物理设计中需要对其进一步完善
指导原则：
 使用可读性强的名称作为表名
 为表添加合理的注释
 评估单表规模，如果单表字段数过多（如Oracle单表最多不应超过255个字段，否则肯定会引起行链接），则需要重新设计或进一步拆表。尽量精简，不宜过多，否则必定影响性能。
 估算数据量，对大表明确业务数据的生命周期，指定表的数据迁移、保留策略
 分区表
对于大量数据，且业务处理上有明确的分区条件的表应做分区处理。分区表对应的绝大多数业务处理规则均应带有明显的分区键，可以把一次处理的数据限定在某个具体范围内，控制一次查询需要扫描的数据量。表的数据量和是否采用分区表没有必然联系。在不使用全局索引的情况下，使用分区表的查询一般应带有分区键，把一次查询需要访问的数据限制在一个或少量分区内。
单表行数超过500万行或者单表容量超过2GB，才推荐进行分库分表。

字段标准化和冗余

逻辑设计中的数据实体属性对应的就是数据库表中的字段
指导原则：
 采用可读性强的名称作为字段名
 为字段添加合理的注释
 设计用于记录创建时间和更新时间的字段作为数据清理的参考
 根据实际业务查询的需要，在必要时采取反范式设计，增加冗余列避免在实现业务逻辑时产生过多影响性能的表连接。字段允许适当冗余，以提高性能，但是必须考虑数据同步的情况。冗余字段应遵循： 1）不是频繁修改的字段。 2）不是varchar超长字段，更不能是text字段。 正例：商品类目名称使用频率高，字段长度短，名称基本一成不变，可在相关联的表中冗余存储类目名称，避免关联查询。

业务查询条件字段不宜超过10个字符
varchar是可变长字符串，不预先分配存储空间，长度不要超过5000，如果存储长度大于此值，定义字段类型为text，独立出来一张表，用主键来对应，避免影响其它字段索引效率。

创建索引和约束

索引是从数据库获取数据的最高效方式之一。大部分的数据库性能问题都可以通过使用索引技术解决，在数据库设计阶段，最重要的工作之一就是创建索引。但表上的索引过多可能会导致更新和插入操作非常缓慢，也不应在频繁更新的列上创建索引，应仔细检查和设计每一个索引，来保证它在保证检索性能的同时不会太影响更新和插入操作。
小规模表（少于1000行记录）、OLAP系统可以不创建索引，检索小表或一个表中的大部分数据时，有索引不一定比没索引好。不对大型字段创建索引。
指导原则：
为经常作为查询条件或经常用于排序的字段创建索引
不为选择率低的列创建索引
组合索引的本质是为了提高索引的可选择率，所以如果col1上的索引选择率足够好，就可以不为col1和col2的查询再创建联合索引
不得使用外键与级联，一切外键概念必须在应用层解决。 说明：

（概念解释）学生表中的student_id是主键，那么成绩表中的student_id则为外键。如果更新学生表中的student_id，同时触发成绩表中的student_id更新，则为级联更新。外键与级联更新适用于单机低并发，不适合分布式、高并发集群；级联更新是强阻塞，存在数据库更新风暴的风险；外键影响数据库的插入速度。
谨慎设计位图索引
和普通的B树索引相比，位图索引在distinct值很少的列上可以提高查询效率，但是在更新索引列时会锁住一部分键值相同的行。如果对索引列没有更新操作，那么可以创建位图索引，反之则不建议使用
在数据库上设计约束，但可以不在数据库端实现，由应用程序控制，以减少应用程序和数据库端的交互。

业务上具有唯一特性的字段，即使是组合字段，也必须建成唯一索引。 说明：不要以为唯一索引影响了insert速度，这个速度损耗可以忽略，但提高查找速度是明显的；另外，即使在应用层做了非常完善的校验和控制，只要没有唯一索引，根据墨菲定律，必然有脏数据产生。

创建数据库文档

物理数据库设计的最后应生成数据库设计文档，留档备查，为后续修改提供文档基础。
指导原则：
完善ER图。仔细检查ER图的设计，确保前面已经完成数据库的逻辑及物理设计在ER图中已经得到体现。
生成数据库文档。以人工或软件自动处理的方式，生成数据库设计的相关文档。文档内应至少包含：
实体设计，属性
表名，字段名，用注释说明其业务含义和作用
近似的或者估计的数据量，按量级区别（百级/千级/万级/十万/百万/千万）。
实体的被使用属性：只读/只写/少写多读/多写少读/多读多写
实体支持的关系型数据库系统（Oracle/DB2/EDB）
创建时间
最近修改时间

数据库实现

数据库设计的最后应对设计出的数据库进行物理实现。
主要活动：
利用PowerDesigner的数据库生成工具生成所有表的DDL语句脚本，并在对应的RDBMS中执行

SQL使用规则

1. 需要join的字段，数据类型保持绝对一致；多表关联查询时，保证被关联的字段需要有索引。 说明：即使双表join也要注意表索引、SQL性能。

2.不要使用count(列名)或count(常量)来替代count(*)，count(*)就是SQL92定义的标准统计行数的语法，跟数据库无关，跟NULL和非NULL无关。
说明：count(*)会统计值为NULL的行，而count(列名)不会统计此列为NULL值的行。

3.count(distinct col) 计算该列除NULL之外的不重复数量。注意 count(distinct col1, col2) 如果其中一列全为NULL，那么即使另一列有不同的值，也返回为0。

4.数据订正时，删除和修改记录时，要先select，避免出现误删除，确认无误才能执行更新语句。

5.in操作能避免则避免，若实在避免不了，需要仔细评估in后边的集合元素数量，控制在1000个之内。