3数据规范化管理

3.3数据存储

在整个系统中涉及到海量的有关数据，如风洞设备的试验数据、设备使用过程中产生的故障数据和传感器采集到的各种信号数据等，如果不对这些数据进行规范处理并存储，将会使得后期无法有效利用这些数据。本节介绍对这些数据的编码、结构化处理及存储方式。

3.3.1数据编码

风洞设备种类繁多，为了清晰地区分各类数据并对其统一管理，应该对这些数据进行统一编码。本部分结合气动中心装备管理编码规范，对相关数据的编码结构提出具体的规范和要求。下面将介绍数据编码的范围、引用文件、术语定义、编码要求和具体的数据编码规范。

（1）范围

本部分规定了风洞设备健康管理系统相关数据的编码要求。

本规范适用于风洞设备健康管理系统及相关信息管理系统的开发。

（2）引用文件

下列文件中的条款通过引用而成为本规范的条款。

《全军信息服务基础平台白皮书》

GB/T 2260 中华人民共和国行政区划分代码

GB/T 10113 分类编码通用术语

GB/T 24463.3-2009 交互式电子技术手册 第3部分：公共源数据库要求

GJB 4017-2000 风洞试验代码

GJB 7000-2010 军用物资和装备分类

GJB 6600.2-2009 装备交互式电子技术手册 第2部分：数据模块编码和信息控制编码

Q/QD 9-2021中国空气动力研究与发展中心标准 气动中心装备管理编码规范

Q/J QD 2.1-2014 气动数据库数据规范 第1部分 常规气动数据

Q/J QD 2.2-2015 气动数据库数据规范 第2部分 特种气动数据

GB/T 36625.2-2018 智慧城市 数据融合 第2部分：数据编码规范

（3）术语和定义

①数据

本部分术语“数据”是指在风洞设备健康管理系统中所涉及到的各种设备数据、试验数据、故障数据等数据的集合。

②编码结构

编码结构指编码符号排列的逻辑顺序。

（4）编码要求

①通用要求

a.具有唯一性；

b.采取分段式编码

②字符要求

a.数字字符“0”至“9”；

b.大写英文字母“A”至“Z”，但不包括“I”和“O”（易与数字“1”和“0”混淆）。

（5）编码规范

在整个风洞设备健康管理系统中，对所有的数据进行分类可以分为设备数据、试验数据、故障数据、维护数据等。数据的整体编码结构如图X所示。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 510×× | - | ××× | - | ××× | - | ×× | - | ××× | - | ×× | - | ×× |
| 管理机构码（前5位） |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 部门编码 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 所属系统类别码 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 所属系统序号 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 数据类型码 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 设备类别码 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 设备序号 |  |  |  |  |  |  |  |  |  |  |  |  |  |

图X 数据整体编码结构

①“管理机构码”字段由5位数字表示，用来表示管理结构的识别码如低速空间动力研究所的标识码51013。

②“部门编码”字段由5位数字表示，用来表示管理及使用知识的具体部门，如90101。

③“所属系统类别码”及“所属系统序号”表示部门管理的具体风洞类别及序号，以“字母”+“数字”组成，如4米×3米低速风洞设备可以被表示为“AFL12Z-0104001”。

④“数据类型码”字段由1位大写英文字母组成，表示数据所属的类别。设备数据由“D”表示，试验数据由“T”表示，故障数据由“F”表示，维护数据由“M”表示。

⑤“设备类别码”及“设备序号”用来表示风洞中涉及到的设备的类别及序号，以“字母”+“数字”组成，如编号为001的风扇可以被表示为“FS-001”。

3.3.2数据结构化处理

在构建系统中各部分模型的时候，数据处理相关的任务往往要占据大部分时间，数据的质量直接决定了模型的预测能力和泛化能力的好坏。数据质量涉及到很多因素，包括准确性、完整性、一致性、时效性、可信性和解释性。在真实数据中，获取到的数据可能包含了大量的缺失值，可能包含大量的噪音，也可能因为人工录入错误导致有异常点存在，非常不利于算法模型的训练。数据预处理是数据分析、挖掘前一个非常重要的数据准备工作。一方面它可以保证挖掘数据的正确性和有效性，另一方面通过对数据格式和内容的调整，使数据更符合挖掘、分析的需要。数据预处理的结果是对各种源数据进行相应的处理，得到结构化的、标准的数据，提供给后续的分析使用。

原始数据主要有结构化数据、半结构化数据和非结构化数据三种。结构化数据类型包括预定义的数据类型、格式和结构的数据，常见的如关系型数据库中数据表里的数据；半结构化数据指具有可识别的模式并可以解析的文本数据文件，比如XML数据文件和JSON文件；非结构化数据指没有固定结构的数据，各种文档、图片、视频、音频等都属于非结构化数据。在预处理时应先对半结构化数据和非结构化数据进行相应的转换，将其转换为结构化数据，再针对结构化数据使用统一的预处理方法，下面对结构化数据的预处理方法及半结构化数据和非结构化数据向结构化数据的转换方法进行说明。

（1）结构化数据的预处理方法规范

结构化数据的预处理步骤主要分为数据清洗、数据集成、数据规约和数据变换。下面将从这四个方面介绍具体方法。

①数据清理

数据清洗（data cleaning）指检测数据中存在冗余、错误、不一致等噪声数据，利用各种清洗技术，形成“干净”的一致性数据集合。如下图X所示。

在桌子上

低可信度描述已自动生成

图X 数据清洗

其主要思想是通过清除重复数据、填补缺失值、消除噪声数据，平滑或删除离群点，并解决数据的不一致性来“清理”数据。数据清理又可分为清除重复数据、缺失值处理、离群点处理和噪声处理，下面对这几种处理方式进行介绍。

a.清除重复数据

为了提高数据挖掘的速度和精度，有必要去除数据集合中的重复记录。如果有两个及以上的实例表示的是同一实体，那么即为重复记录。为了发现重复实例，通常的做法是将每一个实例都与其他实例进行对比，找出与之相同的实例。对于实例中的数值型属性，可以采用统计学的方法来检测，根据不同的数值型属性的均值和标准方差值，设置不同属性的置信区间来识别异常属性对应的记录，识别出数据集合中的重复记录，并加以消除。

相似度计算是重复数据清洗过程中的常用方法，通过计算记录的各属性的相似度，再考虑每个属性的不同权重值，加权平均后得到记录的相似度。如果两条记录相似度超过了某一阈值，则认为两条记录是匹配的，否则，认为这两条记录指向不同实体。

另一种相似度计算算法基于基本近邻排序算法。核心思想是为了减少记录的比较次数，在按关键字排序后的数据集上移动一个大小固定的窗口，通过检测窗口内的记录来判定它们是否相似，从而确定重复记录。

b.缺失值处理

在获取信息和数据的过程中，由于手动输入的失误操作、部分信息需要保密或者数据来源不可靠等各种各样的原因，使得数据集中的内容残缺不完整。会存在各类的原因导致数据丢失和空缺。比如某条记录的属性值被标记为NULL、空缺或“未知”等。一旦不完整、不准确的数据用于分析，则会影响抽取模式的正确性和导出规则的准确性，进而导致分析结果和执行决策出现严重偏差。如下图X所示。

图片包含 游戏机, 钟表, 画, 桌子

描述已自动生成

图X 缺失数据清洗

针对这些缺失值的处理方法，主要是基于变量的分布特性和变量的重要性（信息量和预测能力）而采用不同的方法，具体见下表X。

表X 缺失值处理方法

|  |  |  |  |
| --- | --- | --- | --- |
| 方法名称 | 处理 | 优点 | 缺点 |
| 删除变量 | 若变量的缺失率较高（大于80%），覆盖率较低，且重要性较低，可以直接将变量删除。 | 执行效率高 | 数据的可靠性降低 |
| 统计量填充 | 若缺失率较低（小于95%）且重要性较低，则根据数据分布的情况进行填充。如果数据符合均匀分布，用该变量的均值填补缺失；如果数据存在倾斜分布的情况，采用中位数进行填补。 | 贴近原始数据分布，不易产生数据分析偏差 | 有可能会影响缺失数据与其他数据之间原本的相关性 |
| 插值法填充 | 使用随机插值、多重差补法、热平台插补、拉格朗日插值、牛顿插值等进行插值填充。 | 贴近原始数据分布，不易产生数据分析偏差 | 有可能会影响缺失数据与其他数据之间原本的相关性 |
| 模型填充 | 使用回归、贝叶斯、随机森林、决策树等模型对缺失数据进行预测 | 填充效率高 | 填充的数据依赖于模型的预测能力 |

c.离群点处理

异常值是数据分布的常态，处于特定分布区域或范围之外的数据通常被定义为异常或噪声。异常分为两种：“伪异常”，由于特定的业务运营动作产生，是正常反应业务的状态，而不是数据本身的异常；“真异常”，不是由于特定的业务运营动作产生，而是数据本身分布异常，即离群点。检测离群点的几种主要方法如下表X所示：

表X 离群点检测方法

|  |  |
| --- | --- |
| 检测方法 | 处理 |
| 简单统计分析 | 根据箱线图、各分位点判断是否存在异常。 |
| 基于绝对离差中位数 | 这是一种稳健对抗离群数据的距离值方法，采用计算各观测值与平均值的距离总和的方法，放大了离群值的影响。 |
| 基于距离 | 通过定义对象之间的临近性度量，根据距离判断异常对象是否远离其他对象。 |
| 基于密度 | 离群点的局部密度显著低于大部分近邻点，适用于非均匀的数据集。 |
| 基于聚类 | 利用聚类算法，丢弃远离其他簇的小簇。 |

在检测出离群点后，可以按照如下方法处理：根据异常点的数量和影响，考虑是否将该条记录删除，这种方法易损失较多信息；使用平均值或中位数替代异常点，简单高效，这种方法信息的损失较少。

d.噪声处理

噪声是变量的随机误差和方差，是观测点和真实点之间的误差。常用的消除噪声数据的方法分为两种，分别是噪声平滑方法（data polishing）和噪声过滤（data filters）。

噪声平滑常用分箱法。将预处理数据分布到不同的箱中，通过参考周围实例平滑噪声数据，包括等宽分箱和等深分箱两大类。具体的分箱技术包括: 按箱平均值平滑，即求取箱中的所有值的平均值，然后使用均值替代箱中所有数据; 按中位数平滑，和上一种方法类似，采用中位数进行平滑; 按设定的箱边界平滑，定义箱边界是箱中的最大和最小值，用最近的箱边界值替换每一个值。

噪声过滤利用聚类方法对离群点进行分析、过滤。在训练集中明确并去除噪声实例。噪声过滤的常用算法包括IPF 算法（iterative partitioning filter）、EF 算法（ensemble filter）。

②数据集成

数据集成（data integration）是将多个源文件或多个源数据库中的数据进行合并处理，解决语义的模糊性。这些源可能包括多个数据库、数据方或一般文件，在集成时主要涉及到实体识别问题、冗余问题和数据值的冲突问题。

a.实体识别问题

数据分析者或计算机如何才能确信一个数据库中的customer\_id和另一个数据库中的customer\_number指的是同一实体？通常，数据库和数据仓库中包含有元数据——关于数据的数据。这种元数据可以帮助避免模式集成中的错误。

b.冗余问题

如果一个属性能够由另一个表“导出”，那么称这个属性是冗余的，如年薪。属性命名的不一致也可能导致数据集中的冗余。应用相关性检测冗余：数值型变量可计算相关系数矩阵，标称型变量可计算卡方检验。

c.数据值的冲突问题

为了避免此类问题，不同数据源在统一合并时，应保持规范化并去重。

③数据规约

数据归约（data reduction）是在对发现任务和数据本身内容理解的基础上，寻找依赖于发现目标的表达数据的有用特征，以缩减数据模型，从而在尽可能保持数据原貌的前提下最大限度地精简数据量，促进大数据挖掘更高效。主要有维度规约和维度变换两种方式。

a.维度规约

用于数据分析的数据可能包含数以百计的属性，其中大部分属性与挖掘任务不相关，是冗余的。维度归约通过删除不相关的属性，来减少数据量，并保证信息的损失最小。

b.维度变换

维度变换是将现有的数据降低到更小的维度，尽量保证数据信息的完整性。主要方法有主成分分析（PCA）、因子分析（FA）、奇异值分解（SVD）等方法。

④数据变换

数据变换（data transformation）指对数据进行规范化、离散化、稀疏化处理，以达到用于挖掘分析的目的。

a.规范化处理

数据中不同特征的量纲可能不一致，数值间的差别可能很大，不进行处理可能会影响到数据分析的结果，因此，需要对数据按照一定比例进行缩放，使之落在一个特定的区域，便于进行综合分析。特别是基于距离的挖掘方法，聚类，KNN，SVM一定要做规范化处理。在进行规范化处理时，可使用最大/小规范化、Z-Score标准化、Log变换等方法进行处理。

b.离散化处理

数据离散化是指将连续的数据进行分段，使其变为一段段离散化的区间。分段的原则有基于等距离、等频率或优化的方法。

c.稀疏化处理

针对离散型且标称变量，无法进行有序的Label Encoder时，通常考虑将变量做0，1哑变量的稀疏化处理，例如动物类型变量中含有猫，狗，猪，羊四个不同值，将该变量转换成is\_猪，is\_猫，is\_狗，is\_羊四个哑变量。若是变量的不同值较多，则根据频数，将出现次数较少的值统一归为一类“rare”。稀疏化处理既有利于模型快速收敛，又能提升模型的抗噪能力。

（2）半结构化转结构化数据

常见的半结构化数据格式有XML格式和JSON格式等，在对数据预处理前先将其转换为结构化数据，即可按照结构化数据的预处理方法对其进行统一处理。

可扩展标记语言（XML）以分层格式作为数据交换的标准应用在各种Web程序中，用于在万维网的信息交流。XML文档由嵌套元素结构从根元素开始，每个元素都有一个与它关联的标记。除嵌套元素外，一个元素或子元素可以有属性和值，这使得非结构化地Web内容能够更好地被用户理解。格式良好的XML文档（它的标签是正确的嵌套）不需要符合特定的文档类型定义（DTD）或者架构。因此，XML是比较适合存储半结构化的数据，它具有易扩展的优势，也即扩展过程只需要更改相应的DTD。下面对XML数据转换为结构化数据的方法进行介绍。

在将XML格式的数据转换为结构化数据时可以通过xml-parser来解析XML数据。xml-parser是一种XML的解析器，常见的xml-parser有：DOM、SAX、JDPM和DOM4J等，其中DOM是XML解析技术的基础方法，能使用户很好地理解XML文件结构。SAX因其解析速度快且占用内存少而适用于移动设备，是目前较为流行的解析方法。

DOM即文档对象模型，DOM定义了所有XML元素的对象和属性，以及访问它们的方法（接口）。因为DOM分析器是以树的形式把XML文档放在了内存中，所以DOM中每个节点都可以通过Java或其他编程语言来访问。

SAX属于事件驱动，是应用于实际案例之后产生的标准，不需要建立文档树。它的原理是使用事件机制和方法回调技术来实现XML文档的解析。

XML文件通过解析器使用DOM和SAX两种方法对其进行解析。DOM方法是读入XML文档并构建一个树形结构，解析时对内存的要求较高，当文档过大、过于复杂时，对其遍历会相当费时，导致解析效率比较低。SAX方法是把所有的数据都封装成一个个事件，，需要在解析时通知解析器调用相应方法对其解析，因此不需要将整个文档存储之后再进行读取，极大地提高了整个解析过程的效率。两种方法都可以达到解析XML文档的目的，但相比内存占用过大的DOM，SAX是较好的选择。

（3）非结构化转半结构化数据

非结构化数据可以先转换为半结构化数据，然后再转换为结构化数据，这种方法叫做间接转换法，运用较为广泛。

间接转换法中的半结构化数据常常用XML文件来承载。目前广泛使用的文件类型有PDF、Office文档、文本数据以及HTML网页等，它们不具有通用性，只能根据某一种文件的某一个特征进行分析、转换为XML。

①Word文档需要Jacob技术对其进行操作；

②Excel文档需要Java Excel API对其进行操作；

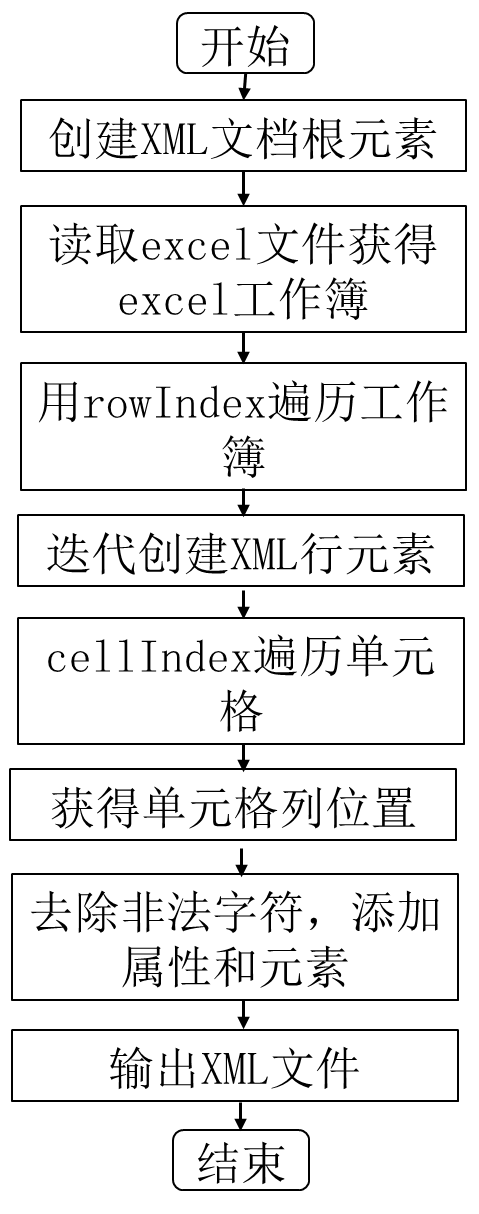
③HTML网页需要使用Webharvest、htmlparser或者htmlclean等技术对其进行XML数据的转换；

④图片、图像、音频、视频这些非文本的以二进制形式存储的文件，则不能转换为XML文档，只能采用新建XML文档对这些文件的属性信息进行管理，即创建对应的XML文档对图片、图像、音频、视频等文件的文件名称、文件大小、文件的分类、文件路径等相关属性进行记录。用户需要使用这些文件就可以根据XML文档中的属性信息进行筛选和查找，并根据文档中记录的文件路径进行调用。

本小节描述将非结构化的XLS格式数据转换为XML数据的方法作为非结构化数据转换为半结构化数据的示例。

在转换excel表时可以使用函数库（Apache POI）对excel表进行读写，POI提供API给Java程序对Microsoft Office格式数据实现读和写的功能。XML的解析过程即是对文档数据进行顺序扫描的过程，当扫描到文档与元素的开始与结束时通知时间处理函数，再由时间处理函数做处理，然后继续同样的扫描，直至文档结束。

通过POI函数库提供给Java程序的API对excel文档进行读取时通过遍历XLS文件、获取单元格的位置等算法，最终生成XML文件并输出。其流程如下图X所示：



图X excel表转化XML文件流程

3.3.3数据存储

在对数据进行结构化处理后，可以得到格式、结构统一的各类试验数据、生产数据、故障数据等，其统一的逻辑结构均为二维表格的形式。在二维表格中，各部分数据被划分为一个个字段，每条数据对于同一个字段可能会有着不同的值，在存储此种类型数据时可以使用关系型数据库统一存储。

关系型数据库（Relational Database）是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。关系模型可以简单理解为二维表格模型，而一个关系型数据库就是由二维表及其之间的关系组成的一个数据组织。因此，关系型数据库以行和列的形式存储数据，其格式与要存储的二维表数据格式完全统一，用户也可以通过查询来检索数据库中的数据。

关系型数据库具有如下几个特点：

存储方式：传统的关系型数据库采用表格的储存方式，数据以行和列的方式进行存储，要读取和查询都十分方便。

存储结构：关系型数据库按照结构化的方法存储数据，每个数据表都必须对各个字段定义好（也就是先定义好表的结构），再根据表的结构存入数据，这样做的好处就是由于数据的形式和内容在存入数据之前就已经定义好了，所以整个数据表的可靠性和稳定性都比较高，但带来的问题就是一旦存入数据后，如果需要修改数据表的结构就会十分困难。

存储规范：关系型数据库为了避免重复、规范化数据以及充分利用好存储空间，把数据按照最小关系表的形式进行存储，这样数据的管理就可以变得很清晰、一目了然。

扩展方式：由于关系型数据库将数据存储在数据表中，数据操作的瓶颈出现在多张数据表的操作中，而且数据表越多这个问题越严重，如果要缓解这个问题，只能提高处理能力，也就是选择速度更快性能更高的计算机，这样的方法虽然可以一定的拓展空间，但这样的拓展空间一定有非常有限的，也就是关系型数据库只具备纵向扩展能力。

查询方式：关系型数据库采用结构化查询语言（即SQL）来对数据库进行查询，SQL早已获得了各个数据库厂商的支持，成为数据库行业的标准，它能够支持数据库的CRUD（增加，查询，更新，删除）操作，具有非常强大的功能，SQL可以采用类似索引的方法来加快查询操作。

规范化：在数据库的设计开发过程中开发人员通常会面对同时需要对一个或者多个数据实体（包括数组、列表和嵌套数据）进行操作，这样在关系型数据库中，一个数据实体一般首先要分割成多个部分，然后再对分割的部分进行规范化，规范化以后再分别存入到多张关系型数据表中，这是一个复杂的过程。随着软件技术的发展，相当多的软件开发平台都提供一些简单的解决方法，例如，可以利用ORM层（也就是对象关系映射）来将数据库中对象模型映射到基于SQL的关系型数据库中去以及进行不同类型系统的数据之间的转换。

事务性：关系型数据库强调ACID规则（原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）），可以满足对事务性要求较高或者需要进行复杂数据查询的数据操作，而且可以充分满足数据库操作的高性能和操作稳定性的要求。并且关系型数据库十分强调数据的强一致性，对于事务的操作有很好的支持。关系型数据库可以控制事务原子性细粒度，并且一旦操作有误或者有需要，可以马上回滚事务。

读写性能：关系型数据库十分强调数据的一致性，并为此降低读写性能付出了巨大的代价，虽然关系型数据库存储数据和处理数据的可靠性很不错，但一旦面对海量数据的处理的时候效率就会变得很差，特别是遇到高并发读写的时候性能就会下降的非常厉害。

在关系型数据库中，数据由不同的表组成，每个表包含有不同的字段，还可以对各个字段施加多种限制，其结构如下表X所示。

表X 关系型数据表结构

|  |  |  |
| --- | --- | --- |
| 字段1 | 字段2 | 字段3 |
| A | E | I |
| B | F | J |
| C | G | K |

在使用关系型数据库存储二维表数据时，可以使用当前流行的MySQL数据库管理系统。MySQL是目前最流行的RDBMS（Relational Database Management System，关系数据库管理系统）应用软件之一。MySQL所使用的SQL语言也是用于访问数据库的最常用标准化语言。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择MySQL作为网站数据库。使用MySQL数据库存储二维表数据时非常方便快速：只需要将已经规范化处理的表数据存储为CSV文件、EXCEL文件等格式，然后就可以使用各种各样的数据库管理工具从这些文件中批量导入数据。

在数据库中存储知识时，还应该满足以下要求：

（1）在存储数据的时候，为了能够较好地支持中文，应该使用utf-8对数据进行编码处理。

（2）数据库中存储的数据应该遵循统一的命名方式，以保证全局数据的一致性。应该能够通过统一的命名直观地感受到知识所属的类别及其具体信息。

（3）在使用数据库存储数据文件时，可能会由于各种原因导致数据丢失或损坏，因此应该对数据库中的数据进行备份，以便各种意外事件发生时能够迅速恢复数据。

（4）对于数据库中有加密需求的部分数据，应选用合适的加密方法对其进行加密处理，以防数据泄露。

3.4数据备份

在数据库中存储着海量数据，可能会因为管理人员操作不当、软件BUG、硬件故障、外界入侵等原因导致部分数据或全部数据丢失。因此，为了避免数据丢失时严重影响系统的正常运行，应该对数据库中存储的数据进行备份。本节对几种常见的数据备份方式及备份流程进行介绍。

3.4.1 备份方式

数据的备份方式主要分为完全备份、拆分备份、事务日志备份、差异备份、在线备份和离线备份几种方式，下面对各种备份方式分别进行介绍。

（1）完全备份

完全备份是指对数据库整体进行复制备份，包括数据库中的用户表、系统表、索引、视图和存储过程等所有数据库对象。由于备份整个数据库，因此这种方式需要耗费较多的时间和空间。

（2）拆分备份

拆分备份是指当数据库的容量大于备份介质的容量时，将数据库拆分成容量小于备份介质容量的若干个子数据库，然后将各子数据库分别备份到不同的备份介质上，保留原有的访问控制策略，并保证原数据库的完整性。

（3）事务日志备份

这种方式需要创建一个事务日志文件，用来记录数据库的改变。在备份的时候只需要复制自上次备份以来对数据库所做的改变，所以只需要很少的时间。

（4）差异备份

差异备份也叫增量备份。它是只备份数据库一部分的另一种方法，它不使用事务日志，相反，它使用整个数据库的一种新映象。它比最初的完全备份小，因为它只包含自上次完全备份以来所改变的数据库。这种备份方式的优点是存储和恢复速度较快。

3.4.2 备份流程

在对数据库整体进行备份时，可以直接使用使用mysqldump备份工具或成熟的数据库管理工具提供的可视化界面快速完成，本节主要介绍对数据库中单张数据表的备份及恢复流程。

在对数据库中存储的数据表进行单独备份时，需要对该表提前进行锁定，允许其他线程继续查询该表但是不能对表中数据进行修改。锁定表之后可以使用SQL语句对数据进行备份和恢复，备份完成之后再解除对该表的锁定。可以使用SELECT INTO OUTFILE语句备份数据，并用LOAD DATA INFILE语句恢复数据。但是这种方法只能导出数据的内容，而不包括表的结构，如果表的结构文件损坏，必须要先恢复原来的表的结构。对表中数据进行备份的步骤及SQL语句大概如下：

（1）首先锁定数据表，避免在备份过程中表被更新，使用SQL语句“LOCK TABLES tbl\_name READ”；

（2）然后导出数据，使用SQL语句“SELECT \* FROM tbl\_name INTO OUTFILE ‘tbl\_name.bak’”或是使用mysqldump备份工具在命令行窗口直接备份；

（3）备份完成后解锁表，使用SQL语句“UNLOCK TABLES”。

使用备份数据还原数据表时的步骤及SQL语句如下：

（1）首先为表增加锁定，使用SQL语句“LOCK TABLES tbl\_name WRITE”；

（2）然后恢复数据，使用SQL语句“LOAD DATA INFILE ‘tbl\_name.bak’ REPLACE INTO TABLE tbl\_name”；

（3）最后对表进行解锁。

数据表备份及恢复的完整流程如下图X所示：

形状

低可信度描述已自动生成

图X 数据表备份及恢复流程

3.5数据更新

随着整个系统的不断运行，系统中的设备数量不断增多，传感器获取到的数据以及和设备相关的各类信息也随之不断增加，这些都要求系统中存储的数据应该能够随着时间不断更新。本节对数据更新相关的要求进行介绍。

数据更新（data revision）是以新数据项或记录、替换数据文件或数据库中与之相对应的旧数据项或记录的过程。数据更新主要包括数据的插入、修改和删除三类操作。在进行数据更新时，应该遵循一下要求：

（1）在增加新的数据时，应该遵守命名时的规范，正确设置其名称和编号，便于和之前的数据统一存取。

（2）在更新已有数据的时候，不能够破坏其原有的命名方式，并在更新时同步对涉及到的其它相关数据进行更新，以保持数据库中全局数据的一致性。

（3）在对数据库中的数据进行更新、增加、删除的时候，应该由特定的数据库管理人员进行操作。对于系统中的其它人员，可适当给予一定的权限，不应该将数据库中数据的权限全部放开。

（4）对于应加密处理的数据，新增或更新时仍要进行加密处理。

（5）在新增或更新数据库中数据后，应按照选用的备份方法对其进行备份处理。

（6）数据库中应该保存有日志记录文件，每次更新数据后应将更新内容写入到日志文件中。