Использование базы данных при принятии решений в процессе стандартизации

Д.В. Силаков, Институт системного программирования РАН
silakov@ispras.ru

Аннотация. В статье рассказывается о подходе к сбору и анализу данных, необходимых для принятия решений в процессе стандартизации, основанном на использовании базы данных и связанной с ней инфраструктуры, затрагиваются вопросы обеспечения быстродействия. Описывается применение подхода к разработке стандарта Linux Standard Base [1].

Database Usage for Decision Making during Standardization Process

D.V. Silakov, Institute of System Programming, RAS
silakov@ispras.ru

Abstract. This paper describes an approach to collection and analysis of data required for decision making during standardization process, based on the database usage. The paper also concerns operating speed questions. The approach is demonstrated on Linux Standard Base [1] specification development process.

1. Введение

Разработка стандартов в области программного обеспечения – непростой и ресурсоемкий процесс, требующий проведения большого объема аналитической работы. Одной из важных составляющих разработки многих стандартов является исследование существующих реализаций целевых систем – например, для определения их общих черт либо для выявления каких-то перспективных свойств.

Еще одним важным аспектом разработки стандартов является то, что интерес может представлять не только конечный продукт в виде текста спецификации, но и различные промежуточные аналитические результаты. Выводы аналитиков могут понадобиться и в дальнейшем, существенно облегчая разработку последующих версий стандарта. Например, если некоторый объект не был включен в стандарт из-за того, что его свойства сильно отличаются в различных реализациях, то, возможно, следует рассмотреть возможность проведения повторного исследования ситуации при разработке следующих версии стандарта. Если же, наоборот, объект не включили по причине его постепенного вытеснения более совершенными аналогами, то его можно больше не принимать во внимание. Кроме того, зачастую в процессе исследований выявляется набор свойств, которыми должен удовлетворять объект для того, чтобы быть включенным в стандарт (либо наоборот, может быть выявлен набор свойств, наличие которых у объекта является препятствием для его стандартизации).

В качестве примера достаточно глубокого анализа реализаций можно привести процесс разработки стандарта Single UNIX Specification [2]. При его создании в 1992-93 гг. было исследовано пятьдесят ведущих на тот момент приложений для UNIX, в результате чего было выявлено 130 функций-кандидатов на включение в стандарт.

Однако в подавляющем большинстве случаев все результаты аналитической деятельности представляются в виде некоторых документов. При достаточно большом объеме информации такой подход сильно затрудняет ее использование – много времени может уйти просто на поиск необходимых данных и сведения их в единый отчет. Кроме того, достаточно ресурсоемким является процесс обновления документов.

Одним из возможных подходов к облегчению труда разработчиков стандартов является использование базы данных для хранения материалов для анализа и,

возможно, некоторых правил, которые следует учитывать при принятии решений. Для проектирования такой базы предлагается использовать абстрактную модель систем, на которые нацелен стандарт. База данных должна сопровождаться набором инструментов, позволяющих оперативно получать необходимые аналитику сведения. Данная статья рассказывает о реализации такого подхода, разрабатываемой в ИСП РАН в рамках совместного проекта с Linux Foundation для процесса создания стандарта Linux Standard Base [3].

2. Построение модели целевых систем

В основе рассматриваемого подхода лежит использование базы данных для хранения информации о существующих системах, на которые нацелен стандарт. При проектировании базы данных предлагается построить абстрактную модель целевой системы, выделив в ней следующие сущности:

- Объекты, подлежащие стандартизации, и те их свойства, которые планируется определить в стандарте.
- Объекты, стандартизация которых не планируется, но которые могут быть полезны при проведении анализа. У таких объектов также необходимо выделить свойства, необходимые при для проведения анализа – конструируемая модель должна отражать те свойства объектов, которые принимаются во внимание при принятии решений.

Также необходимо установить связи между выделенными объектами. Имея набор объектов, их свойств и связей между ними, можно описать модель системы с помощью некоторой формальной нотации, подходящей для проектирования базы данных. В роли такой нотации может выступать, например, диаграмма «сущность-связь». Рассмотрим проектирование базы данных на примере стандарта Linux Standard Base.

3. Проектирование БД для Linux Standard Base

Целью разработки стандарта Linux Standard Base (LSB) является обеспечение переносимости приложений между различными дистрибутивами Linux; именно дистрибутивы являются целевой системой в случае LSB. Для достижения переносимости приложений между дистрибутивами осуществляется стандартизация части объектов, которые должны предоставляться операционной системой. К таким объектам относятся:

- Бинарные интерфейсы (Application Binary Interface, ABI).
- Библиотеки – наборы бинарных интерфейсов.
- Команды и утилиты.
- Классы.
- Модули интерпретируемых языков Perl и Python.

Разработчики приложений могут быть уверены, что их продукты будут работать на любом дистрибутиве, удовлетворяющем стандарту LSB, при условии, что используются только стандартизованные объекты.

В случае LSB достаточно четко определено, какая информация необходима для принятия решений о стандартизации. Официальная позиция The Linux Foundation при разработке LSB состоит в том, что в стандарт включаются только элементы, использование которых уже стало "устоявшейся практикой". Более точно, для включения в LSB некоторого элемента (библиотеки, команды, интерфейса и пр.) для него должны должны выполняться следующие условия:

- Рассматриваемый элемент должен присутствовать во всех основных

современных дистрибутивах (под «основными» подразумеваются дистрибутивы, которым принадлежит основная доля рынка Linux-систем – SUSE, RedHat и основанный на нем Oracle Unbreakable Linux, Ubuntu, Debian, Mandriva и др.). Для удобства дистрибутивы делятся на компоненты (в соответствии со своей внутренней структурой), каждый из которых предоставляет определенные команды, библиотеки и пр.

- Добавляемый элемент должен быть востребован приложениями.
- Крайне желательно наличие документации, специфицирующей добавляемую сущность. Стоит отметить, что отсутствие качественной документации является одной из главных проблем многих свободных программ, и различные библиотеки не являются исключением. Для небольшого количества интерфейсов возможна разработка документации непосредственно силами Linux Foundation, однако для больших библиотек, состоящих из сотен интерфейсов, эта задача достаточно трудоемка.

Положительным аспектом при принятии решения о включении в стандарт интерфейса либо команды является наличие тестов, проверяющих конкретную реализацию на соответствие спецификации. Однако именно отсутствие качественного тестирования является слабой стороной многих программных продуктов, поэтому в большинстве случаев разработку тестов Linux Foundation берет на себя.

Наличие некоторого интерфейса во всех дистрибутивах и востребованность его приложениями является необходимым, но не достаточным условием для включения в стандарт LSB. Например, достаточно часты случаи, когда поведение некоторой функции еще не устоялось и в будущих версиях библиотеки возможна ее кардинальная переработка. Другим примером являются устаревшие интерфейсы, которые разработчики планируют удалить в следующих версиях. Поэтому помимо анализа выполнения формальных критериев для включения интерфейса в LSB, необходимо знать мнение разработчиков библиотеки, в которую он входит.

После изучения сущностей, фигурирующих в критериях для принятия решений о стандартизации, была предложена схема таблиц, содержащих информацию о собираемых данных, представленная на рисунке 1.

Помимо представленных на схеме сущностей, существуют вспомогательные таблицы LibraryAttribute, InterfaceAttribute и CommandAttribute, предназначенные для хранения произвольной информации (в текстовом виде) для любой библиотеки, функции и команды соответственно. Как правило, в эти таблицы заносятся комментарии для элементов, рассматривавшихся в качестве кандидатов на включение в LSB и для которых были выполнены все формальные критерии, но которые не были включены в стандарт после консультаций с разработчиками. Указывается причина, по которой элемент не был включен (устарел, является небезопасным и т.п.), а также, по возможности, входящая в LSB альтернатива, которую следует использовать вместо него.

Отметим, что указанные таблицы являются частью спецификационной базы данных LSB [5], которая помимо рассматриваемых данных содержит подробную информацию об элементах, уже включенных в стандарт.

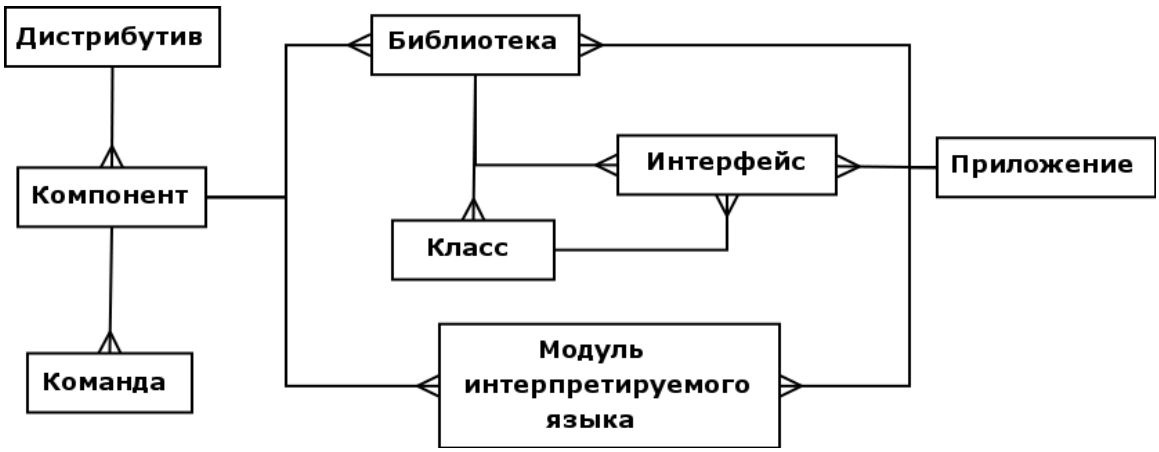

Рисунок 1. Схема таблиц БД, содержащих данные о дистрибутивах и приложениях

4. Обеспечение инструментальной поддержки

Наличие базы данных, содержащей необходимую для аналитиков информацию, облегчает их труд, однако непосредственная работа с базой посредством SQL запросов является достаточно трудоемким занятием; формирование нужных запросов само по себе может отнимать существенную часть времени. Существует необходимость в программном комплексе, включающем в себя базу данных, облегчающем взаимодействие с ней человека и способном создавать на основе этой базы отчеты в необходимой для анализа форме.

Так, для принятия решения о включении того или иного элемента в стандарт LSB должна быть проделана достаточно большая аналитическая работа. Схожая работа выполняется и при разработке других стандартов и спецификаций. Как уже упоминалось ранее, при разработке Single UNIX Specification было проведено исследование пятидесяти приложений и выявлено 130 функций-кандидатов на включение в стандарт. Однако объем данных, с которыми приходится работать при создании LSB, существенно больше – на данный момент анализируются несколько десятков дистрибутивов и сотни приложений, а кандидатами на включение в стандарт выступают десятки тысяч интерфейсов. В отсутствии инструментальной поддержки анализ такого объема данных требует чрезвычайно больших человеческих ресурсов.

Программный комплекс, предоставляющий удобные возможности по анализу и принятию решений в процессе создания LSB, разрабатывается в ИСП РАН в рамках совместного проекта с Linux Foundation по развитию инфраструктуры стандарта LSB [3]. Комплекс включает в себя следующие компоненты:

- Инструменты по сбору и анализу информации, необходимой для вынесения вердикта по перечисленным выше необходимым критериям включения элемента в стандарт. Сюда входят инструменты по анализу дистрибутивов, приложений, тестовых наборов, а также документации.
- База данных, в которую помещается собранная информация. Также схема базы поддерживает хранение произвольной информации в текстовом виде, привязанной к произвольному анализируемому элементу.
- Инструменты, составляющие отчеты по заданным элементам. Такие отчеты используются аналитиками при принятии решения о включении элемента в стандарт.

Рассмотрим составляющие комплекса подробнее.

5. Подготовка данных для анализа

Сбор необходимых данных является первым этапом любого анализа. В нашем случае осуществляется сбор различной информации по дистрибутивам, приложениям, существующей документации и тестовым наборам.

5.1. Анализ дистрибутивов

Типичный дистрибутив Linux в варианте поставки на одном DVD диске включает в себя несколько тысяч разделяемых библиотек, экспортирующих сотни тысяч интерфейсов. При принятии решений о добавлении некоторого компонента рассматривается около десятка дистрибутивов. Кроме того, LSB поддерживает 7 различных архитектур (x86, x86-64, IA64, PPC32, PPC64, S390 и S390X), и зачастую рассматриваемый компонент имеет разные свойства на различных архитектурах (так, одна и та же библиотека может иметь разный состав интерфейсов, а интерфейсы – различные версии). Помимо бинарных библиотек, LSB специфицирует некоторые аспекты интерпретаторов и их окружения для языков Perl и Python; информация о том, как устроено это окружение в каждом дистрибутиве, также представляет интерес.

В ИСП РАН разработан набор инструментов, позволяющий составить перечень библиотек, предоставляемых дистрибутивом, вместе с экспортируемыми функциями, список предоставляемых модулей Perl и Python, а также список присутствующих в дистрибутиве команд и утилит. При сборе данных о бинарных файлах используются системные утилиты Linux – readelf и objdump [4], для анализа файлов интерпретируемых языков используются средства, предоставляемые соответствующими интерпретаторами. Если дистрибутив распространяется в виде пакетов в формате rpm либо deb, то установка дистрибутива не требуется – инструменты могут работать непосредственно с пакетами указанных форматов.

Анализу подвергается не весь дистрибутив; в целях сокращения объема собираемых данных поддерживается список библиотек (содержащий около 800 записей), используемых проанализированными на данный момент приложениями. Спектр охвата приложений достаточно широк, и «за бортом» анализа фактически остаются только библиотеки, специфические для конкретного дистрибутива, а также различные компоненты, предназначенные для отладочных целей и в процессе штатной работы программ не использующиеся. В среднем для одного дистрибутива анализируются порядка 2000 библиотек, содержащих около полумиллиона интерфейсов.

5.2. Анализ приложений

Помимо инструментов для анализа дистрибутивов, в ИСП РАН разработаны схожие по структуре средства обработки приложений. В отличие от исследования дистрибутивов, для приложений анализируются интерфейсы и библиотеки, необходимые для их успешного запуска и функционирования.

Общее число приложений для Linux с трудом поддается оценке. В дистрибутиве, поставляемом на одном диске, содержится порядка 3000 приложений. Основной репозиторий дистрибутива Gentoo содержит около 12000 различных программ, а одним из самых обширных является основной репозиторий популярного дистрибутива Ubuntu, содержащий более 22000 приложений.

Приложения крайне разносторонни, и при анализе полезно разбивать их на группы согласно некоторой классификации. В LSB используются следующие характеристики:

- Размер – по количеству используемых внешних интерфейсов приложения делятся на маленькие (используют менее 100 внешних функций), средние (от

100 до 999) и большие (1000 и более). Стоит отметить, что этот "размер" не обязательно отражает объем исполняемых файлов приложения либо объем его кода – приложение с большим объемом исходного кода может оказаться "маленьким" с точки зрения базы данных, если практически не использует внешних (не реализованном в нем самом) функций.

- Производитель. Данная характеристика интересна, в основном, для коммерческих приложений. Для свободных программ в качестве производителя указывается дистрибутив, из которого была взята сборка приложения для анализа.
- Функциональная категория – каждое приложение относятся к одной из 12 категорий, включающих в себя «Data Management», «Network» и др.
- Тип лицензии, по которой распространяется приложение (свободная, проприетарная либо смешанная).

Собранные данные по приложениям и дистрибутивам представляются в текстовом формате и с помощью дополнительных инструментов загружаются в реляционную базу данных. Типичный размер данных, собираемых для дистрибутива, поставляемого на одном DVD диске, составляет около 500 Мб; после предварительной обработки этих данных в базу загружается около 50 Мб. По состоянию на март 2008 года в базе содержится информация о 22 дистрибутивах и 850 приложениях Linux, общий объем данных в базе – 1.5 Гб. На данный момент собираются данные в основном для двух самых распространенных архитектур – x86 и x86-64, однако в будущем планируется расширение этого диапазона.

5.3. Анализ документации

Разработчики LSB предпочитают использовать готовые спецификации элементов, включаемых в стандарт. Так, если описание функции уже содержится в некотором стандарте (например, в POSIX или ISO C) и это описание полностью удовлетворяет потребностям разработчиков приложений, то в текст LSB вместо описания функции заносится ссылка на соответствующую спецификацию. Из почти сорока тысяч функций, включенных в LSB 3.2, непосредственно в стандарте описано около семисот.

Для многих библиотек, рассматриваемых в качестве кандидатов на включение в стандарт LSB, существует некоторая документация. Однако зачастую она неполна и охватывает не все предоставляемые библиотекой интерфейсы; поэтому важной является задача определения конкретных функций, для которых есть документация. При большом количестве функций такая задача может занять существенное время, однако в ИСП РАН разработаны инструменты, облегчающие такой анализ для документации в текстовом формате, в форматах HTML, PDF и других, для которых возможно преобразование в HTML либо в текст.

5.4. Анализ тестов

В случае, когда рассматривается возможность добавления в LSB новой библиотеки и для этой библиотеки существуют некоторые тестовые наборы, необходимо провести анализ этих наборов и определить, какие именно интерфейсы они проверяют. Этот процесс также производится автоматически с помощью соответствующих инструментов. Помимо определения списка тестируемых интерфейсов, для каждого из них определяется качество тестирования; на данный момент различаются три уровня качества – поверхностный (Shallow), нормальный (Normal) и глубокий (Deep).

5.5. Получение информации от разработчиков

Как уже отмечалось ранее, мнение разработчиков программных компонентов, включаемых в стандарт, является чрезвычайно важным. Однако непосредственное обращение к разработчикам по каждому возникающему вопросу зачастую оказывается малоэффективным и может сильно затянуть процесс принятия решений. Особенно это заметно в мире свободного ПО, где даже многие ключевые компоненты разрабатываются добровольцами в свободное от основной работы время. Поэтому важно не переносить на разработчиков львиную долю работы по принятию решений о стандартизации тех или иных элементов, а получать от них некоторые дополнительные критерии (специфические для данного компонента), которым должны удовлетворять элементы, добавляемые в стандарт. Например, разработчик может указать, что не надо специфицировать методы определенного класса, либо функции, использующие определенные типы параметров. Такие знания играют важную роль в процессе принятия решений и достаточно часто позволяют существенно сократить число кандидатов на стандартизацию. Кроме того, важно их сохранять, поскольку с течением времени может встать вопрос об обновлении информации о том или ином компоненте стандарта, и наличие накопленных ранее знаний сократит время повторного исследования компонента.

6. Создание отчетов

Автоматизация сбора данных для анализа позволяет сильно сократить время принятия решений по стандартизации того или иного набора элементов. Однако наличие данных само по себе решает проблему только частично – необходимы средства для представления собранных данных в удобной для аналитиков форме.

Такие средства в случае LSB обеспечивает LSB Navigator [6], предоставляющий Web-интерфейс к различным частям базы данных LSB, в том числе и к таблицам, содержащим данные о дистрибутивах и приложениях. Рабочая версия LSB Navigator развернута на сайте linux-foundation.org и доступна всем желающим. Navigator распространяется по лицензии GNU GPL version 2 и его исходный код может быть свободно получен в Linux Foundation Bazaar [7]. Там же находятся и скрипты, необходимые для развертывания локальной копии базы данных LSB.

Для каждого элемента, который потенциально может быть стандартизован LSB (библиотеки, команды, класса, и т.д.), Navigator отображает «домашнюю страницу» со всей информацией, необходимой для анализа (присутствие в дистрибутивах, использование в приложениях, ссылки на документацию и т.п.). Существует возможность построения статистического отчета для произвольной группы интерфейсов из некоторой библиотеки. Отчеты не просто содержат численные показатели; для каждого из них может быть получена детальная расшифровка – где именно используется интерфейс, в каких тестовых наборах проверяется его поведение и т.п. Также доступны отчеты, позволяющие определить потенциальных кандидитов на включение в стандарт – наиболее часто используемые библиотеки и функции.

По мере необходимости отображаются доступные комментарии и рекомендации по командам, библиотекам и интерфейсам из таблиц CommandAttribute, LibraryAttribute и InterfaceAttribute, которые также играют важную роль при стандартизации.

7. Обеспечение быстродействия

Отчеты, создаваемые автоматически на основе имеющейся в базе данных информации, являются важным, но не главным критерием для принятия решения о включении того или иного элемента в стандарт. Для каждого элемента, проходящего по

всем формальным критериям, содержащимся в отчетах, зачастую необходим дополнительный анализ, проводимый человеком. Например, на практике встречались ситуации, когда некоторая функция присутствует во всех системах и даже используется приложениями, однако разработчики считают эту функцию ненадежной и рекомендуют к ней не обращаться.

Таким образом, основная функция отчетов заключается в отсечении набора элементов, для которых проведение глубоких исследований нецелесообразно. Проведение таких исследований – достаточно трудоемкое занятие, поэтому крайне желательно получать отчеты по произвольным элементам за короткое время. Кроме того, информация о том, может ли некоторый объект рассматриваться как кандидат на включение в LSB, полезна для разработчиков приложений. Такая информация может быть использована либо при принятии решений об использовании той или иной функции в приложении, либо для запроса на включение в стандарт LSB уже используемых приложением функций. Естественно, разработчики должны получать интересующие их сведения оперативно – требуется, чтобы человек мог увидеть все необходимую информацию с помощью LSB Navigator в режиме «on-line»; задержка в ответе на запрос в несколько секунд уже рассматривается как слишком долгая. В то же время запросы к базе данных, используемые при составлении отчетов, имеют достаточно сложную структуру; поскольку таблицы, к которым происходит обращение, содержат большой объем данных, то время выполнения запросов может быть достаточно велико. Ниже рассматриваются способы, которыми можно это время сократить.

7.1. Использование средств СУБД

Для улучшения производительности всегда важно использовать средства, предоставляемые используемой СУБД. В качестве СУБД для базы данных LSB используется MySQL, являющаяся открытой и в то же время полнофункциональной и производительной системой. Для таблиц используется тип MyISAM; его выбор обусловлен главным образом тем, что добавление данных производится редко и при необходимости может быть произведена заново. Ввиду этого отсутствует необходимость поддержки долгих транзакций. Кроме того, инструменты, заполняющие базу, достаточно надежны и обеспечивают целостность данных, а изменений уже загруженных данных никогда производится. Это позволяет обходиться без внешних ключей. При таких условиях MyISAM обладает наилучшей производительностью.

Для MySQL существует ряд рекомендаций от разработчиков как по оптимизации структуры базы данных, так и по оптимизации запросов. Ознакомиться с ними можно в соответствующем разделе портала MySQL [8]. Как показала практика разработки инфраструктуры LSB, использование этих рекомендаций позволяет в некоторых случаях увеличить производительность в несколько раз.

7.2. Проектирование схемы БД

Как и при разработке любой достаточно большой базы данных, крайне важным является проектирование ее схемы. При этом рассматриваются по крайне мере два важных аспекта работы с базой – сложность манипулирования данными и скорость работы с ними. Эти два аспекта достаточно часто приводят к противоречивым требованиям; так, с точки зрения удобства манипулирования данными, желательно производить нормализацию таблиц. Однако нормализация приводит к увеличению числа таблиц, что влечет за собой необходимость выполнения более сложных запросов, затрагивающих больше таблиц, чем в ненормализованном случае. Такое усложнение

запросов зачастую ведет к увеличению времени их выполнения.

В процессе разработки схемы базы данных стандарта LSB такие проблемы также рассматриваются. При этом была учтена важная особенность процесса сбора данных для принятия решений – добавление новых данных происходит достаточно редко (в среднем – один раз в неделю), а необходимости изменения уже существующих данных практически не возникает (в случае обнаружения ошибок в инструментах, загружающих информацию в базу, все данные просто перезагружаются заново).

С учетом этого было принято решение использовать базу данных, таблицы которой лежат в третьей нормальной форме, однако для создания отчетов использовать не непосредственно таблицы этой базы, а создаваемые автоматически на их основе денормализованные таблицы. Какие именно таблицы нужны для ускорения работы генераторов отчетов, решается на основе анализа запросов, выполняемых этими генераторам; необходимы такие таблицы, которые позволят упростить структуру запросов и сократить время их выполнения.

Создаваемые автоматически таблицы должны перегенерироваться после каждого обновления данных. При этом время их создания может быть достаточно существенным; например, для базы данных LSB он составляет порядка 20 минут. В случае частого обновления данных использование такого подхода может оказаться нецелесообразным ведь во время автоматического создания таблиц генераторы отчетов фактически недееспособны, и может оказаться, что существенную часть времени ими просто нельзя пользоваться.

Стоит отметить, что возможно использование денормализованных таблиц и непосредственно для хранения данных. При этом возрастает сложность инструментов, загружающих данные в таблицу, а также усложняется задача поддержания согласованности данных. По крайне часть этих задач может быть возложена на СУБД (например, посредством использования триггеров и механизмов поддержки целостности данных). Однако сложность всего комплекса в целом при этом возрастает, потенциально ослабляя его надежность, и не давая никаких преимуществ в скорости по сравнению с автоматически создаваемыми вспомогательными таблицами. Кроме того, при необходимости структура вспомогательных таблиц может быть достаточно легко изменена, либо могут быть добавлены новые таблицы, в то время как изменения схемы основной части базы данных, как правило, требуют переработки всех работающих с ней инструментов.

8. Заключение

При разработке стандартов в области программного обеспечения часто встает необходимость сбора и анализа больших объемов данных. Анализируемая информация может содержать как статистические сведения, так и некоторые комментарии и рекомендации, полученные от сторонних консультантов. Автоматизация процесса сбора данных, а также предоставление аналитикам инструментов для получения различного вида отчетов позволяет существенно сократить время принятия решений о стандартизации тех или иных элементов.

Одним из возможных подходов к такой автоматизации является создание программного комплекса, в основе которого лежит база данных, содержащая аналитическую информацию различного рода; схема базы данных строится на основе абстрактной модели целевых систем. Такой подход применяется при разработке стандарта Linux Standard Base; соответствующие инструменты разрабатываются в ИСП РАН в рамках совместного проекта с Linux Foundation. Объем обрабатываемых данных достаточно велик; в то же время предъявляются строгие требования к быстродействию системы, поэтому важную роль играют различного вида оптимизации. В проекте

используются как независимые от используемой СУБД техники оптимизации, так и специфичные для нее, в совокупности позволяющие достичь желаемого быстродействия.

Литература

[1] Linux Standard Base (LSB). http://www.linux-foundation.org/en/LSB
[2] API Standards for Open Systems. Andrew Josey, The Open Group. http://www.opengroup.org/austin/papers/wp-apis.txt.
[3] LSB Infrastructure Program. http://ispras.linux-foundation.org
[4] GNU Binutils. http://www.gnu.org/software/binutils/
[5] LSB Specification Database.
http://ispras.linux-foundation.org//index.php/LSB_Database_Home
[6] LSB Navigator. http://linux-foundation.org/navigator
[7] Linux Foundation Unofficial Bazaar Repository. http://bzr.linux-foundation.org/unofficial/
[8] MySQL 5.1 Reference Manual: Optimization.
http://dev.mysql.com/doc/refman/5.1/en/optimization.html