54 pages

Дата: 19.11.2012

Версия: 1.0

Статус: На утверждении

Заказчик: ЗАО «Компания «ИнтерТраст»

Исполнитель: ООО «Люксофт Профешнл»

Содержание

[Список рисунков iii](#_Toc341106861)

[Список таблиц iv](#_Toc341106862)

[1. История изменений документа 1](#_Toc341106863)

[1.1 Список изменений 1](#_Toc341106864)

[1.2 Список согласований и ревью 1](#_Toc341106865)

[1.3 Список утверждений 1](#_Toc341106866)

[2. Используемая литература 1](#_Toc341106867)

[3. Назначение документа 2](#_Toc341106868)

[4. Концепция системы 2](#_Toc341106869)

[5. Архитектура 3](#_Toc341106870)

[5.1 Внешние связи подсистемы хранения 3](#_Toc341106871)

[5.2 Логическая схема подсистемы хранения CompanyMedia 5 4](#_Toc341106872)

[5.3 Компоненты 5](#_Toc341106873)

[5.3.1 Принципы разбиения подсистемы хранения на компоненты 5](#_Toc341106874)

[5.3.2 Компонент интерфейса поисковых запросов 6](#_Toc341106875)

[5.3.3 Компонент интерфейса манипулирования данными 6](#_Toc341106876)

[5.3.4 Компонент интерфейса управления структурой данных 6](#_Toc341106877)

[5.3.5 Компонент генерации поисковых запросов 7](#_Toc341106878)

[5.3.6 Компонент генерации запросов манипулирования данными 8](#_Toc341106879)

[5.3.7 Компонент создания структуры объектов 11](#_Toc341106880)

[5.3.8 Компонент работы с версиями документов 11](#_Toc341106881)

[5.3.9 Компонент генерации идентификаторов 11](#_Toc341106882)

[5.3.10 Компонент описания диалекта реляционной базы данных 12](#_Toc341106883)

[5.3.11 Компонент кэширования запросов 12](#_Toc341106884)

[5.3.12 Компонент управления сессиями Базы Данных 13](#_Toc341106885)

[5.3.13 Компонент обслуживания словаря данных 13](#_Toc341106886)

[5.3.14 Компонент управления файлами документов 14](#_Toc341106887)

[5.3.15 Компонент конфигурации 14](#_Toc341106888)

[5.4 Стандарты и технологии 14](#_Toc341106889)

[5.4.1 Инструментальные средства разработки 15](#_Toc341106890)

[5.4.2 Стандарты разработки 15](#_Toc341106891)

[6. Типовые структуры данных, используемые для хранения информации в репозитории системы CompanyMedia 5 15](#_Toc341106892)

[6.1 Наследование в объектной модели 15](#_Toc341106893)

[6.1.1 Виртуальное и структурное наследование 16](#_Toc341106894)

[6.1.2 Правила наследования классов объектов 18](#_Toc341106895)

[6.1.3 Поиск по нескольким типам объектов 19](#_Toc341106896)

[6.2 Хранение множественных атрибутов объектов в хранилище 20](#_Toc341106897)

[6.2.1 Хранение множественных атрибутов в БД 21](#_Toc341106898)

[6.2.2 Типовые операции с множественными атрибутами 22](#_Toc341106899)

[6.2.3 Использование множественных атрибутов в операциях поиска объектов 22](#_Toc341106900)

[6.3 Типовая структура объекта в программном интерфейсе 22](#_Toc341106901)

[6.3.1 Доступ к атрибутам универсального объекта 23](#_Toc341106902)

[6.3.2 Бизнес-логика универсального объекта 24](#_Toc341106903)

[6.4 Соглашения по использованию имен объектов и таблиц объектов 24](#_Toc341106904)

[6.5 Соглашения по использованию имен полей таблицы 25](#_Toc341106905)

[6.6 Хранение файлов документов в репозитории системы CompanyMedia 5 25](#_Toc341106906)

[6.6.1 Структура хранения файлов вложений в системе CompanyMedia 5 26](#_Toc341106907)

[6.6.2 Организация областей файловых хранилищ для файлов вложений 27](#_Toc341106908)

[6.6.3 Обеспечение уникальности файлов 28](#_Toc341106909)

[7. Словарь данных репозитория объектов системы CompanyMedia 5 29](#_Toc341106910)

[7.1 Структура словаря данных 29](#_Toc341106911)

[7.2 Свойства классов объектов 30](#_Toc341106912)

[7.3 Свойства атрибутов объекта 32](#_Toc341106913)

[7.4 Создание новых и модификация существующих классов объектов 33](#_Toc341106914)

[7.5 Типы данных в репозитории объектов 33](#_Toc341106915)

[7.5.1 Отображение базовых типов объектов на объекты Java 34](#_Toc341106916)

[7.5.2 Отображение базовых типов объектов на БД Oracle 35](#_Toc341106917)

[7.5.3 Отображение базовых типов объектов на БД PostgreSQL 35](#_Toc341106918)

[8. Системные объекты 36](#_Toc341106919)

[8.1 Объекты словаря данных 37](#_Toc341106920)

[8.2 Базовые виртуальные объекты 37](#_Toc341106921)

[8.3 Служебные объекты и таблицы 39](#_Toc341106922)

[9. Алгоритмы выполнения основных операций 40](#_Toc341106923)

[9.1 Распределения объектов по БД объектов 40](#_Toc341106924)

[9.2 Алгоритм нумерации объектов 40](#_Toc341106925)

[9.3 Алгоритм оптимистической блокировки 41](#_Toc341106926)

[9.4 Алгоритм долговременной блокировки над объектом 43](#_Toc341106927)

[9.5 Алгоритмы работы с версиями документов 44](#_Toc341106928)

[9.6 Алгоритмы поддержки работы с репликацией объектов 45](#_Toc341106929)

[Приложение 1 Словарь терминов 46](#_Toc341106930)

Список рисунков

[ Рис. 1 Внешние связи подсистемы хранения 3](#_Toc341106931)

[ Рис. 2 Логическая схема подсистемы хранения 4](#_Toc341106932)

[ Рис. 3 Алгоритм обработки поискового запроса 8](#_Toc341106933)

[ Рис. 4 Алгоритм обработки запроса на манипулирование данными 10](#_Toc341106934)

[ Рис. 5 Пример иерархии классов объектов 16](#_Toc341106935)

[ Рис. 6 Пример виртуального и структурного наследования 17](#_Toc341106936)

[ Рис. 7 Примеры разрешенного и запрещенного наследования 18](#_Toc341106937)

[ Рис. 8 Запрещенное перекрытие атрибута при наследовании класса объекта 19](#_Toc341106938)

[ Рис. 9 Наследование класса-родителя по нескольким путям 19](#_Toc341106939)

[ Рис. 10 Организация хранения множественных атрибутов 21](#_Toc341106940)

[ Рис. 11 Организация хранения вложений документов 26](#_Toc341106941)

[ Рис. 12 Пример расположения файлов вложений и их связи с записями вложений 28](#_Toc341106942)

[ Рис. 13 Структура словаря данных 30](#_Toc341106943)

[ Рис. 14 Структура объектов словаря данных 37](#_Toc341106944)

[ Рис. 15 Структура базовых объектов системы CompanyMedia 5 38](#_Toc341106945)

[ Рис. 16 Служебные таблицы и объекты 39](#_Toc341106946)

[ Рис. 17 Алгоритм оптимистической блокировки 42](#_Toc341106947)

Список таблиц

[Таблица 1. Перечень свойств класса объекта в словаре данных системы CompanyMedia 5 30](#_Toc341106948)

[Таблица 2. Перечень свойств атрибутов объекта в словаре данных системы CompanyMedia 5 32](#_Toc341106949)

[Таблица 3. Типы данных в подсистеме хранения системы CompanyMedia 5 33](#_Toc341106950)

[Таблица 4. Отображение типов данных атрибутов классов объектов системы CompanyMedia 5 на типы данных Java 34](#_Toc341106951)

[Таблица 5. Отображение типов данных атрибутов классов объектов системы CompanyMedia 5 на типы данных РСУБД Oracle 35](#_Toc341106952)

[Таблица 6. Отображение типов данных атрибутов классов объектов системы CompanyMedia 5 на типы данных РСУБД PostgreSQL 35](#_Toc341106953)

[Таблица 7. Структура идентификатора объекта в CompanyMedia 5 41](#_Toc341106954)

[Таблица 8. Операции версионности 44](#_Toc341106955)

# История изменений документа

## Список изменений

| Версия | Дата | Предмет изменения | Причина изменений | Кем сделано |
| --- | --- | --- | --- | --- |
| 0.1 | 15.11.2012 | Создана начальная версия документа |  | А. Афанасьев  Г. Кушнир |
|  |  |  |  |  |
|  |  |  |  |  |

## Список согласований и ревью

|  |  |  |  |
| --- | --- | --- | --- |
| Версия | Дата | Ревьюер | Результат (ссылка на документ) |
|  |  |  |  |
|  |  |  |  |

## Список утверждений

|  |  |  |  |
| --- | --- | --- | --- |
| Версия | Дата | Кто утвердил | Ссылка на документ (или подпись) |
|  |  |  |  |
|  |  |  |  |

# Используемая литература

При разработке данного документа использовались следующие документы:

1. Бизнес требования к платформе электронного документооборота, документированного электронного взаимодействия и распределенной коллективной работы нового поколения «CompanyMedia» 5, Бизнес требования к СМ 5 \_3\_10\_2\_(чист).doc
2. CM5 -Треб к платформе - Панов \_20121023.doc
3. СМ5\_Платформа (Панов В.А.) v0.1 20121029.doc
4. СМ5\_ПХ Версионирование бизнес-объектов в СМ5 Панов В.А. 20121010.doc
5. Требования к поиску, Поиск.docx, 14.10.2012
6. Требования к механизму прав, Права.doc, 13.11.2012
7. Требования к поддержке версионности, Версионность.docx, 12.11.2012
8. Письмо:   
   From: Vladimir A. Panov [panov@inttrust.ru](mailto:panov@inttrust.ru)  
   To: Afanasyev, Andrey [AAfanasyev@luxoft.com](mailto:AAfanasyev@luxoft.com)  
   Cc: Kushnir, Grigoriy [GKushnir@luxoft.com](mailto:GKushnir@luxoft.com); Kravchenko,   
    Mikhail [MKravchenko@luxoft.com](mailto:MKravchenko@luxoft.com);   
    Taranchenko, Oleg [otaranchenko@inttrust.ru](mailto:otaranchenko@inttrust.ru);   
    Veselov, Oleg [OVeselov@luxoft.com](mailto:OVeselov@luxoft.com);   
    Trishina, Svetlana [STrishina@luxoft.com](mailto:STrishina@luxoft.com)  
   Subject: RE: Коллекции  
   Date: Пн 12.11.2012 15:18

# Назначение документа

В документе описывается модель архитектуры подсистемы хранения CompanyMedia 5, приводятся принципы разбиения на подсистемы, и дается описание существующих подсистем.

Документ также включает в себя описание используемых технологий и стандартов.

# Концепция системы

Система разбивается на подсистемы согласно требуемым Заказчику функциям [1, 2]. Каждая подсистема выполняет обработку замкнутого набора событий.

Подсистемы разбиваются на компоненты. В тексте компонентом называется как любая часть системы ПХ CM5.

Взаимодействие между отдельными подсистемами и модулями осуществляется только через общую системную среду. Прямые вызовы между подсистемами отсутствуют.

Разрабатываемая Система должен удовлетворять основным принципам построения современных информационных систем:

1. **Защищенность**. Отдельные компоненты системы имеют минимально необходимый доступ к защищенной информации.
2. **Централизованность**. данный принцип предполагает то, что вся общая для системы информация хранится в одном месте. Все потребители этой информации получают её, используя стандартные интерфейсы системы.
3. **Масштабируемость*.*** Наращивание мощностей определяется только характеристиками аппаратного обеспечения, на котором функционирует данная система, и каналов передачи данных, используемых для обмена информацией.
4. **Распределенность**. Система позволяет распределять информационные ресурсы и процессы их обработки по нескольким серверам, функционирующих в её составе. Сервера могут быть размещены на различных серверных площадках.
5. **Модульность**. В связи с разнородностью решаемых задач Система состоит из отдельных интегрируемых подсистем.
6. **Открытость**. Система обладает открытым программным API интерфейсом. Подсистемы предоставляют сервисы для обращения сторонних систем к своим функциям.
7. **Гибкость**. Данный принцип определяет возможность добавления новых функций в систему без нарушения её функционирования.
8. **Комплексность**. При декомпозиции должны быть установлены такие связи между структурными элементами системы (подсистемами, компонентами подсистем и комплексами ПО), которые обеспечивают цельность системы и возможность его сопряжения с другими подсистемами.
9. **Сервис-ориентированный подход** (Service Oriented Architecture - SOA). При таком подходе приложения декомпозируется на ряд сервисов, взаимодействие которых обеспечивает выполнение требуемых от приложения функций.

# Архитектура

## Внешние связи подсистемы хранения

Подсистема хранения (ПХ) CM5 взаимодействует с несколькими подсистемами CM5, который являются внешними по отношению к ПХ CM5 (см. Рис. 1).



* Рис. 1 Внешние связи подсистемы хранения

Внешние подсистемы выполняют следующие задачи:

1. Подсистема методов доступа обеспечивают реализацию методов для хранимых объектов. Данная подсистема использует ПХ CM5 для получения и модификации объектов в БД объектов.
2. Подсистема коллекций выполняет задачу по обновлению промежуточных объектов, хранимых в БД коллекций. Информацию об изменении объектов, хранимых в БД объектов, она получает от ПХ CM5. В ходе работы использует ПХ CM5 для получения данных из БД объектов и для модификации информации в БД коллекций.
3. Подсистема полнотекстовой индексации обеспечивает формирование и обновление полнотекстового индекса для хранимых объектов, подвергающейся полнотекстовой индексации.
4. Подсистема регистрации и учёта используется подсистемой хранения для регистрации событий по созданию и изменению объектов БД объектов. В свою очередь ПХ CM5 предоставляет подсистеме «Регистрация и учёт» сервисы для записи
5. Подсистема разграничения доступа обеспечивает проверку требуемых прав доступа при операциях чтения, модификации и удаления объектов, хранимых в ПХ CM5.
6. Подсистема сессий сохраняет информацию о правах Пользователя и о месте Пользователя в организационной структуре. Данная информация не покидает защищенного ядра системы. Все остальные пользователи оперируют с идентификатором сессий.
7. Предлагаю добавить внешнюю по отношению к ПХ, но связанную с ней, «подсистему кэширования», т.к. ПХ уже пользуется кэшем в своем «компоненте кэширования запросов», а платформа в целом не может быть масштабируемой без кэширования. Реализация ее может в 5.0 быть частичной, но архитектура должна ее учитывать сразу.

## Логическая схема подсистемы хранения CompanyMedia 5

Подсистема хранения CompanyMedia 5 состоит из ряда взаимодействующих между собой компонентов (см. Рис. 2).



* Рис. 2 Логическая схема подсистемы хранения

## Компоненты

### Принципы разбиения подсистемы хранения на компоненты

ПХ CM5 разделена на компоненты по принципу клиент-серверной архитектуры согласно принципам SOA. По функциональности в подсистеме хранения выделены следующие основные классы запросов к ПХ:

1. Поиск.
2. Модификация объектов.
3. Модификация структуры хранимых объектов.

#### Интерфейсный уровень подсистемы хранения CompanyMedia 5

На верхнем уровне выделены интерфейсные модули, реализующие сервисы для внешних подсистем:

1. компонент интерфейса поисковых запросов. Служит для поиска информации с учетом прав доступа к объектам;
2. компонент интерфейса манипулирования данными. В основном используется внешней подсистемой «Высокоуровневые прикладные методы доступа» для выполнения операций по созданию, модификации и удаления объектов, хранимых в системе CM5. Так же может использоваться напрямую из других компонентов;
3. компонент интерфейса управления структурой данных. Данный компонент требуется для поддержания изменений прикладной информационной модели по запросам пользователя.

#### Уровень бизнес-логики подсистемы хранения CompanyMedia 5

Бизнес-логику данной подсистемы образуют компоненты, выполняющие обработку основных классов запросов к ПХ:

1. компонент генерации поисковых запросов;
2. компонент генерации запросов манипулирования данных;
3. компонент создания структуры объектов.

К этому же уровню относятся два компонента, которые используются для выполнения некоторых функций компонента генерации запросов и манипулирования данных:

1. компонент работы с версиями документов;
2. компонент генерации идентификаторов.

#### Уровень логики манипулирования данными подсистемы хранения CompanyMedia 5

На данном уровне расположены модули, обеспечивающие работу в внешними по отношению к ПХ CM5 ресурсами. Так же эти модули обеспечивают функции повышения производительности за счет предоставления локальных хранилищ для часто используемых повторяемых данных, для вычисления которых могут потребоваться значительные ресурсы (примером таких данных могут являться поисковые запросы и запросы на манипулирование данными). К этому уровню относятся:

1. компонент описания диалекта реляционной базы данных;
2. компонент кэширования запросов;
3. компонент управления сессиями БД;
4. компонент обслуживания словаря данных;
5. компонент управления файлами документов.

### Компонент интерфейса поисковых запросов

Обеспечивает внешний интерфейс для поисковых запросов, направляемых на выполнение в ПХ CM5.

Обеспечивает восстановление мандата пользователя по идентификатору сессии. Для этого обращается к подсистеме сессий.

### Компонент интерфейса манипулирования данными

Обеспечивает внешний интерфейс для запросов на создание, создание версий, модификацию и удаление объектов, направляемых на выполнение в ПХ CM5.

Следует отметить, что данный компонент должен обеспечить двойной набор интерфейсов:

1. Интерфейсы стандартного манипулирования объектами.
2. Интерфейс для подсистемы Регистрации и учета, который обеспечивает манипулирование объектами без записи в журнал объектного аудита.

Обеспечивает восстановление мандата пользователя по идентификатору сессии. Для этого обращается к подсистеме сессий.

### Компонент интерфейса управления структурой данных

Обеспечивает внешний интерфейс для запросов на модификацию структуры объектов, направляемых на выполнение в ПХ CM5.

Данный компонент так же обеспечивает доступ к словарю данных, хранящем информацию о всех объектах, существующих в системе.

Обеспечивает восстановление мандата пользователя по идентификатору сессии. Для этого обращается к подсистеме сессий.

### Компонент генерации поисковых запросов

Компонент служит для обеспечения требований поиска по атрибутам объектов с учетом прав доступа к объектам [5].

Основные функции компонента:

1. Проверка наличия готового откомпилированного запроса в кэше запросов.
2. Генерация поискового запроса по его описанию, переданного от интерфейса поисковых запросов, в случае если готового запроса нет.
3. Обеспечение защиты информации на уровне просмотра объектов.

Рассмотрим основные этапы обработки поискового запроса в компоненте генерации поисковых запросов (см. Рис. 3).

Компонент генерирует текст SQL запросов поиска информации. На вход компоненту подается объектная модель поискового запроса. Объектная модель поискового запроса создается на этапе разработки спецификации ПХ CM5.

Прежде всего, компонент генерации поисковых запросов пытается найти готовый запрос, соответствующий объектной модели запроса, в модуле кэширования запросов. Если такой запрос не найден, то производится его создание.

При генерации запроса используется словарь доступа данных. Из него получается информация о классах объектов и составе их атрибутов.

Основное назначение данного компонента – обеспечение защиты информации. С целью предотвращения доступа к информации, на получение которой у Пользователя системы нет прав. Для этого применяется предварительная модификация поискового запроса для проверки прав пользователя на получение каждого объекта, участвующего в запросе.

При генерации запроса используется компонент описания диалекта РСУБД, который позволяет определить какие конструкции при генерации запроса можно использовать и создает отдельные конструкции SQL оператора SELECT.

После получения запроса, готового к выполнению в БД, производится выполнение запроса с использованием компонента управления сессиями БД.

При использовании расширенного мандатного доступа [4, 6] может потребоваться пост-обработка результатов выполнения запроса. В этом случае с каждой строкой результат, полученной от БД, компонент генерации поисковых запросов обращается к подсистеме разграничения прав доступа, которая выдает ответ, может ли быть текущая строка включена в результаты запроса.



* Рис. 3 Алгоритм обработки поискового запроса

Пост-обработка результатов запроса с обращением к подсистеме разграничения прав доступа может значительно увеличить время получения результатов при выполнении поисковых запросов. На этапе разработки спецификации ПХ CM5 должны быть приняты меры для отказа от пост-обработки результатов запроса. Отказ от обращения к подсистеме разграничения прав доступа может привести к значительному росту числа строк, получаемых в ходе запроса. При этом пост-обработка остается, но она будет проходить методом группировки записей поискового запроса (возможно с промежуточными проверками).

### Компонент генерации запросов манипулирования данными

Следует отметить, что для изменения одного объекта может потребоваться несколько операторов подмножества data manipulation language (DML) языка SQL.

Основные функции компонента генерации запросов манипулирования данными:

1. Проверка наличия готового откомпилированного запроса в кэше запросов.
2. Создание запросов для выполнения базовых операций CRVUD (Create, Read, Version, Update, Delete):
   1. Создания объекта (Create)
   2. Чтения объекта (Read). Для объектов с множественными атрибутами возможно отложенное чтение множественных атрибутов. Решение принимается при разработке спецификации ПХ CM5.
   3. Создания новой версии (Version).
   4. Изменения объекта (Update).
   5. Удаления объекта (Delete).
3. Обеспечение защиты информации на уровне объектов.
4. Ведение журналов аудита по изменению объектов.
5. Передача информации в подсистему коллекций информации об измененных объектах.

Рассмотрим основные этапы обработки запроса в компоненте генерации запросов манипулирования данными (см. Рис. 4).

Прежде всего, компонент генерации запросов манипулирования данными проверяет допустимость запрашиваемой операции. Для этого он передает в подсистему разграничения доступа объект (или имя класса объекта при создании нового объекта), пользовательский профиль и запрашиваемую операцию. Подсистема разграничения доступа к объектам разрешает или запрещает обработку.

В случае разрешения обработки запрашивается компонент кэширования запросов. Если запрос на выполнения требуемой операции уже был ранее создан, то он возвращается. Для него возвращаются все операторы Data Manipulate Language (далее по тексту используется сокращение - DML).

Если подсистема кэширования не вернула запрос, то запрос на выполнение требуемой операции создается заново.

При генерации запроса используется словарь доступа данных. Из него получается информация о классах объектов и составе их атрибутов.

При генерации запроса используется компонент описания диалекта РСУБД, который позволяет определить какие конструкции при генерации запроса можно использовать и создает отдельные конструкции DML операторов SQL.

После создания запросов на основании анализа атрибутов универсального объекта отбираются только те DML операторы, которые необходимы для выполнения сохранения переданного в подсистему манипулирования данными объекта.

После отбора операторов они выполняются с использованием подсистемы управления сессиями БД. Все операторы DML выполняются в рамках одной транзакции БД.

После выполнения всех операторов DML (или после ошибки во время выполнения одного из них) производится вызов подсистемы регистрации и учета. В которую передается универсальный объект, содержащий как старые, так и новые значения полей (см. п. 6.3.2); время, затраченное на выполнение операции; результат выполнения операции (успешное выполнение или код ошибки); информация о пользователе. Подсистем регистрации и учета фиксирует это событие согласно своим правилам работы. Вызов подсистемы регистрации и учета производиться в асинхронном режиме.

Если был изменен объект, хранимый в БД объектов, то об этом событии оповещается подсистема коллекций. При этом оповещении ей передается универсальный объект, содержащий как старые, так и новые значения полей (см. п. 6.3.2); информация о пользователе. Подсистема коллекций оповещается только в результате успешного выполнения операции по обновлению объекта. Подсистема коллекций оповещается в синхронном режиме.



* Рис. 4 Алгоритм обработки запроса на манипулирование данными

После этого анализируется «были ли изменены поля, входящие в полнотекстовой индекс?». Если такие поля были изменены, то об изменении объекта оповещается полнотекстовой индекс. Полнотекстовой индекс оповещается в асинхронном режиме.

### Компонент создания структуры объектов

По запросу пользователя создаётся запросы на изменения структуры объектов на языке Data Definition Language (далее по тексту используется сокращение DDL) языка SQL. В первой версии будут предусмотрены следующие действия:

1. Создать новый класс объект
2. Создать новый атрибут в классе объекта
3. Удалить атрибут в классе объекта
4. Удалить класс объекта

Следует отметить то, что компонент создания структуры объектов в отличие от других компонентов бизнес-уровня не использует компонент кэширования запросов для хранения своих запросов. Это связано с двумя факторами:

1. запросы на модификацию структуры классов объектов очень редки;
2. после выполнения модификации структур объектов некоторые скомпилированные запросы в кэше запросов становятся недействительными.

После изменения структуры хранимых объектов очищаются кэш сохраненных запросов и кэш словаря объектов. Кэш словаря объектов сразу же целиком заполняется заново измененными данными словаря объектов. Кэш запросов заполняется по мере выполнения запросов по поиску и манипулирования данными.

### Компонент работы с версиями документов

Данный компонент обеспечивает выполнение запросов по работе с версиями хранимых объектов для компонента генерации запросов манипулирования данными. Список поддерживаемых функций в первой версии компонента работы с версиями:

1. Создать новую версию
2. Получить текущую версию
3. Взять указанную версию
4. Удалить документ (включая все версии)

Алгоритмы работы описан в пункте 9.5.

### Компонент генерации идентификаторов

Данный компонент отвечает за создание уникальных идентификаторов. Структура идентификатора рассмотрена в п. 9.2.

Для создания уникального идентификатора используются специальные структуры хранения счетчиков, расположенных в БД (см. п. 8.3). Для хранения счетчиков не используются конструкции SEQUENCE, которые имеются не во всех РСУБД. Кроме того, использование счетчиков выполненных на конструкциях SEQUNCE значительно снижается производительность системы при выбранной структуре объектов.

### Компонент описания диалекта реляционной базы данных

Компонент описания диалекта реляционной базы данных предоставляет универсальный инструмент для описания произвольной реляционной базы данных. Компонент используется всеми компонентами бизнес-уровня для получения основных характеристик используемой (используемых) РСУБД.

Компонент построен по plug-in архитектуре:

1. сам компонент содержит стандартные сервисы, предоставляемые компонентам бизнес-уровня ПХ CM5;
2. компонент определяет интерфейс (API) для plug-in описания РСУБД;
3. компонент содержит механизм подключения произвольного plug-in, соответствующего API для plug-in и описывающего конкретную РСУБД.

Большинство современных баз данных декларирует поддержку SQL2003, но в то же время встречается целый ряд различий, которые необходимо учесть при генерации SQL предложений. Основными различиями для современных РСУБД являются:

1. Типы данных. Практически не одна РСУБД не поддерживает всех типов, определенных в SQL2003. В большинстве РСУБД определены собственные типы данных, работа с которыми реализована гораздо более эффективно, чем со стандартизованными типами.
2. Операции JOIN в поисковых запросах.
3. Ограничения на вложения операторов SELECT (глубина вложения, места в которых могут появляться вложенные операторы).
4. Реализация статистических функций в операторах SQL могут различаться.
5. В некоторых базах существуют расширения операторов UPDATE, DELETE.
6. В большинстве РСУБД для эффективного использования данных используются различные расширения в языке описания данных DDL.

В первой версии системы поставляются два plug-in с описаниями следующих целевых баз данных:

1. Oracle 11g R2
2. PostgreSQL version 9.1

### Компонент кэширования запросов

Компонент кэширования запросов выполняет следующие функции:

1. Сохранение запроса на поиск информации.
2. Сохранение группы операторов DML на модификацию информации.
3. Поиск запроса на поиск информации.
4. Поиск группы операторов DML на модификацию информации.

Особое внимание при разработке этого компонента следует обратить на следующие алгоритмы и структуры данных:

1. алгоритм однозначной идентификации запроса на поиск информации. В него входит не только объектный запрос, но и конкретные данные для поиска информации. В частности в зависимости от количества элементов в списках IN оператора SELECT создаются различные запросы к БД;
2. алгоритм хэш-функции для запросов на поиск информации;
3. структуру хранения групп операторов DML на модификацию информации. Данная структура должна разрабатываться при тесном взаимодействии с разработкой алгоритмов генерации запросов на поиск информации в компоненте генерации запросов на поиск информации;
4. алгоритм хэш-функции для запросов модификации информации.

### Компонент управления сессиями Базы Данных

Данный компонент выполняет следующие функции:

1. Создание соединений с СУБД
2. Создание пулов соединений с СУБД
3. Закрытие соединений
4. Выполнение SQL операторов.

### Компонент обслуживания словаря данных

Основной компонент описания всех прикладных объектов системы, которые могут быть сохранены в СУБД. Функции компонента обслуживания словаря данных:

1. Создание нового класса объекта.
2. Модификация описания существующего класса объекта.
3. Удаление описания класса объекта.
4. Получение полной информации о классе объекта.
5. Получение информации о классе объекта для генерации запроса на модификацию или поиск данных. Данная операция выдает сокращенное описание объекта, удобное для выполнения операции генерации запросов.

Данный компонент используется всеми компонентами бизнес уровня ПХ CM5.

### Компонент управления файлами документов

Компонент отвечает за доступ и управление к внешнему хранилищу файлов. Решает следующие задачи:

1. Создать
2. Найти файл по идентификатору.
3. Прочитать файл.
4. Изменить файл.
5. Удалить файл.
6. Связать файл с объектом.
7. Разорвать связь файла с объектом.

Файловое хранилище реализовано внешними средствами. В первой версии – средствами операционной системы (ОС). Отказоустойчивость должна быть достигнута внешними средствами, в том числе средствами RAID.

В первой версии права доступа на файл не хранятся.

### Компонент конфигурации

Подсистема хранения рассчитана на работу с несколькими СУБД. Текущие конфигурируемые параметры:

1. Текущие используемые диалекты системы
2. Протоколы подключения БД
3. Используемый алгоритм распределения данных (в первом релизе только round robin)

В первой версии компонент хранит конфигурацию в файле.

## Стандарты и технологии

В рамках разработки каркаса планируется использовать следующие стандарты:

1. Java Enterprise Edition 6
2. Java Standard Edition 7

В качестве технологических подходов к построению приложения должно использоваться используется:

1. Multi-tier architecture
2. SOA
3. Inversion of control

### Инструментальные средства разработки

Будут использоваться следующие инструментальные средства разработки.

1. Eclipse IDE for Java Enterprise Edition Developers
2. JetBrains IntelliJ IDEA
3. Tomcat version 7
4. Oracle 11g Release 2
5. PostgreSQL version 9.1

Инструментальные библиотеки, используемые при разработке определяются на этапе разработки спецификации ПХ CM5.

### Стандарты разработки

При разработке системы с данной архитектурой предполагается использование стандартного итеративного подхода, на основе Luxoft Standard Development Process.

# Типовые структуры данных, используемые для хранения информации в репозитории системы CompanyMedia 5

В данном разделе описываются правила генерации структур данных в реляционной БД, используемые для создания структур для хранения объектов в системе CM5. Данные правила используются как для создания объектов в БД хранилищ объектов, так и для БД коллекций.

В целом предлагаемое решение по классам объектов, определяемым во время работы системы, весьма интересно. Но предварительно мы с вами обсуждали вариант на основе заранее заданного набора таблиц, но с заранее заготовленными дополнительными колонками, предназначенными для расширения состава атрибутов уже без изменения схемы БД. Плюс такого решения – в изменениях бизнес-классов без изменения схемы БД, то есть быстро. Минус – в ограниченности возможностей.

Еще одно сомнение, крепнущее по мере въезжания в написанное. Предлагаемый механизм более «низкоуровневый», чем я предполагал изначально. А именно, я предполагал (и писал об этом), что не только в платформе, но на уровне ПХ, будут жестко определены базовые классы, на основе которых (и только их) можно будет в БР определять прикладные классы. Теперь получается, если я правильно понял, что это так и будет, но не в ПХ, а выше (в платформе вообще). Естественный вопрос: а зачем нужен этот совершенно абстрактный промежуточный уровень? Ведь мы не собираемся строить с этой ПХ произвольные информ.системы.   
Возможно, если бы этот уровень был не в динамическом виде, а в программном коде (с компиляцией под заданные базовые классы), реализация обошлась бы проще, да и производительность была бы выше?

## Наследование в объектной модели

CM5 предоставляет возможность определять наследование классов объектов (см. Рис. 5). С точки зрения подсистемы хранения объектов наследование представляет из себя включение в класс-наследник атрибутов из базового класса.



* Рис. 5 Пример иерархии классов объектов

Среди классов объектов можно выделить:

1. листовые классы объектов. Это такие классы объекты от которых не наследуется ни один другой объект системы. Любому листовому объекту в ПХ всегда соответствует таблица (таблицы) в БД;
2. объекты иерархии системы. Это объекты от которых наследуются какие-то классы объектов. Классы иерархии можно подразделить на два вида:
3. виртуальные классы. Таким классам в БД не соответствует никакой таблицы. Они используются для определения атрибутов, которые потом включаются в классы-наследники. Является некоторым аналогом абстрактного класса в программировании;
4. структурные классы объектов. Таким классам в БД соответствует отдельная таблица (таблицы). При наследовании устанавливается связь между таблицами класса-наследника и таблицами структурного класса.

Система позволяет обращаться как к листовому классу объектов, так к структурным классам иерархии. К виртуальным классам напрямую обращаться нельзя, т.к. они не существуют вне контекста классов-наследников.

### Виртуальное и структурное наследование

По способу включения атрибутов базового класса в ПХ выделяют два типа наследования:

1. виртуальное наследование;
2. структурное наследование.

Тип наследование определяется типом базового класса. Для иллюстрации разных механизмов наследования рассмотрим конкретный пример (см. Рис. 6).



* Рис. 6 Пример виртуального и структурного наследования

В исходной схеме определяются 6 объектов. Из них:

1. два виртуальных класса объекта:
2. persistent\_object. Находится в начале иерархии;
3. base\_doc. Находится в середине иерархии после структурного объекта.
4. один структурный класс объект:
5. doc\_root;
6. три листовых класса объекта:
7. doc\_links;
8. input\_doc;
9. output\_doc.

Из этой схемы ПХ согласно правилам наследования создает 4 таблицы (показано на правой части схемы):

1. одну таблицу для структурного объекта (doc\_root\_s);
2. 3 таблицы для листовых классов объектов (doc\_link\_s, input\_doc\_s, output\_doc\_s).

Для упрощения сопоставления в описании таблиц применяются те же типы, что и в описании объектов.

Виртуальные классы включены в таблицы классов-наследников в виде множества атрибутов. Такие атрибуты в таблицах показаны курсивом.

### Правила наследования классов объектов

В первой версии ПХ поддерживается только одиночное наследование от структурных классов объектов, т.е. у любого класса-наследника не может быть больше одного структурного базового класса. Для виртуальных классов допускается множественное наследование. Таким образом у класса объекта может быть больше одного базового класса, но среди них структурным классом может быть не больше одного класса. Структурного класса среди базовых классов может не быть вовсе.

В первой версии подсистемы хранения для виртуального класса невозможно его изменение если он имеет более одного класса-наследника. Это связано с тем, что при изменении виртуального класса должны быть изменены все его классы-наследники. Такая операция может быть очень длительной для некоторых РСУБД.

При наследовании классов в СM5 существуют некоторые ограничения:

1. При наследовании для любого класса объекта-наследника на может быть более одного структурного класса среди всех путей классов объектов-родителей (см. Рис. 7). При ошибочном наследовании на двух путях используются структурные классы объектов (т.е. классы имеющие собственные таблицы).



* Рис. 7 Примеры разрешенного и запрещенного наследования

Предложение

Предлагается в первой версии ограничится более простым правилом множественного наследования:

Множественно наследование от виртуальных классов объектов возможно только если все классы объектов-родителей являются виртуальными.

1. Нельзя перекрывать атрибут классов объектов-родителей. Т.е. все имена атрибутов в объектах, входящих в дерево наследование должны быть уникальными. Таким образом класс объекта-наследника может только добавлять новые атрибуты. Запрещенное наследование показано на Рис. 8 красным цветом.



* Рис. 8 Запрещенное перекрытие атрибута при наследовании класса объекта

1. В случае если класс-родитель включается в класс-наследник по нескольким путам, то он входит в класс-наследник только один раз (аналогично интерфейсам в Java). На Рис. 9 в правой части показано реальный состав атрибутов объекта при множественном наследовании при котором, базовый объект включается в объект-наследник по нескольким путям.



* Рис. 9 Наследование класса-родителя по нескольким путям

1. Среди базовых классов объектов не может быть более одного структурного класса. Все остальные классы должны быть виртуальными. Структурный базовый класс может отсутствовать.

### Поиск по нескольким типам объектов

Структурные объекты предназначены для организации поиска информации в различных классах объектов. Согласно требованию система должна позволять искать информацию в различных типах объектов по общим атрибутам. В подсистеме хранения такой поиск организуется с помощью поиска структурного класса, общего для всех классов объектов по которым проводится поиск.

В частности на приведенной схеме (см. Рис. 6) для классов input\_doc и output\_doc возможен одновременный поиск в обоих классах по атрибутам:

1. title
2. version\_label
3. id\_\_
4. update\_cnt\_\_
5. acl\_\_

ПХ хранения возвращает все найденные записи, попадающие под критерий отбора без учета к какому классу она относится.

Для ограничения классов найденных документов заданным списком при разработке хранилища надо предпринимать специальные меры.

Таким образом можно сказать о том, что структурные классы используются там, где необходим поиск по нескольким классам объектов. В остальных случаях лучше использовать виртуальные классы.

Это связано с тем, что количество записей в таблице структурного объекта является как минимум суммой количества записей во всех таблицах классов-наследников.

Альтернативой поиску по совместным атрибутам классов объектов, хранимых в таблицах структурных классов, может служить поиск по нескольким таблицам с объединением результатов по конструкции UNION. В этом случае следует отметить то, что операция UNION является для большинства РСУБД очень ресурсоемкой (требуется много места для хранения и объединения промежуточных результатов) и сравнительно медленной (т.к. требует обработки большого объема информации не только, хранимой в БД, но и записываемой и модифицируемой во временные рабочие наборы).

В первой версии ПХ поиск по совместным атрибутам среди классов объектов, находящихся в различных таблицах, не реализуется. Поддерживается только поиск по совместным атрибутам, находящимся в общем для различных классов структурном классе.

## Хранение множественных атрибутов объектов в хранилище

В системах документооборота в хранимых объектах многие атрибуты являются множественными. Для удобства проектирования и реализации системы ПХ обеспечивает для объектов хранение и оперирование с множественными атрибутами.

Множественный атрибут представляет собой одномерный массив данных. Индекс массива начинается с 0 и должен быть меньше чем максимальное знаковое четырехбайтовое число (значение – 2415919103). Массив множественных атрибутов не может содержать пустое значение (значение null).

### Хранение множественных атрибутов в БД

Если класс содержит множественные атрибуты то для его хранения создается две таблицы в БД (см. Рис. 10). Множественные атрибуты определяются с помощью символа звездочка (“\*”) после описания типа атрибута.



* Рис. 10 Организация хранения множественных атрибутов

Первая таблица содержит все одиночные атрибуты всех объектов данного класса, вторая таблица содержит все значения множественных атрибутов всех объектов данного класса.

Имена таблиц:

Таблица с множественными атрибутами содержит служебную колонку, содержащее номер строки массивов множественных атрибутов. На схеме (см. Рис. 10) данная колонка имеет имя row\_\_.

Любой объект в системе всегда имеет поле идентификатора объекта (в примерах его имя – id\_\_). В таблице с не множественными атрибутами идентификатор объекта является первичным ключом таблицы. В таблице с множественным ключом используется составной первичный ключ: <id\_\_, row\_\_>.

В приведенном примере (см. Рис. 10) показаны таблицы класса, содержащие 3 объекта. При этом значения объектов:

1. {id\_\_=15, title=”aaa”, alc\_\_=32, version=null, themes=null, authors=null}
2. {id\_\_=48, title=”bbb”, alc\_\_=32, version=[“1.2”], themes=[“apple”, “iPad”], authors=[134, 2568, 3]}
3. {id\_\_=867, title=”query”, alc\_\_=46896, version=[“3.1”], themes=null, authors=null}

Нулевое значение во множественных атрибутах не допускается из-за того, что первое же нулевое значение в колонке множественного атрибута означает то, что список значений множественного атрибута завершился.

Для каждого объекта имеется ровно одна запись в таблице с не множественными атрибутами и от 0 до 2 415 919 103 записей с таблице с множественными атрибутами. Число записей в таблице с множественными атрибутами, соответствующих конкретному объекту, равно максимальной размерности массива множественных атрибутов, хранимых в этом объекте.

### Типовые операции с множественными атрибутами

Основными операциями с множественными объектами являются:

1. Присвоить значение множественному атрибуту с заданным индексом.
2. Вставить новое значение во множественный атрибут перед элементом с указанным индексом.
3. Добавить новый элемент в конец множественного атрибута.
4. Удалить элемент множественного атрибута с заданным индексом.
5. Усечь элемент начиная после элемента с указанным индексом.

### Использование множественных атрибутов в операциях поиска объектов

Поиск объектов по множественным атрибутам немножко отличается от поиска по не множественным атрибутам.

Собственно говоря, поиск по множественному атрибуту может проходить как поиск по списку значений. Отсюда проистекают некоторые ограничения на поиск по множественным атрибутам[[1]](#footnote-2):

1. Нельзя непосредственно сравнивать множественный атрибут с другим атрибутом. Следует использовать конструкцию:  
   <не‑множественный‑атрибут> **[NOT] IN** <множественный‑атрибут>
2. Нельзя сравнивать константу со значением множественного атрибута. Следует использовать конструкцию сравнения с множеством:  
   <константа> **[NOT] IN** <множественный‑атрибут>
3. Нельзя сравнивать список значений с множественным атрибутом.
4. Над множественным атрибутом возможны операции сравнения (*больше, меньше, больше или равно, меньше или равно, лежит в интервале*). При этом операция сравнения считается удавшейся, если хотя бы одно значение множественного атрибута дает для этой операции результат true (истина).

## Типовая структура объекта в программном интерфейсе

Все объекты в программном интерфейсе являются наследником «универсального» объекта.

Данный объект позволяет хранить в себе любые атрибуты.

Универсальный объект имеет несколько предопределённых атрибутов, к которым осуществляется доступ через его методы:

1. Идентификатор объекта. Имя – id. Методы доступа:
   1. getId – дать идентификатор объекта
   2. setId – установить идентификатор объекта. Запрещается изменять идентификатор объекта если значение идентификатора объекта не пустое.
2. Тип объекта. Имя – type. Методы доступа:
   1. getType – дать тип объекта.
   2. setType – установить тип объекта. Запрещается изменять тип объекта, если значение типа не пустое.

### Доступ к атрибутам универсального объекта

Помимо этого универсальный объект имеет группу методов для доступа к атрибутам по именам:

1. void setAttr(String attrName, Object newValue) – установить новое значение атрибута. Если такого атрибута не было, то он заводится в состав универсального объекта.
2. Object getAttr(String attrName) – дать значение атрибута. Если атрибута с таким именем нет, то возвращается null.
3. void removeAttr(String name) – удалить атрибут с заданным именем.
4. boolean hasAttr(String name) – посмотреть имеется ли атрибут с указанным именем в составе объекта.

Для облегчения программирования необходимо создать группы методов, позволяющих оперировать со множественными атрибутами:

1. void setAttrValue(String name, int index, Object newValue) – установить значение множественного атрибута с заданным индексом
2. void insertAtrrValue(String name, int index, Object newValue) – вставить значение нового элемента множественного атрибута перед элементом с заданным индексом
3. void addAttrValue(String name, Object newValue) – добавить новое значение множественного атрибута в конец списка значений множественного атрибута
4. void removeAttrValue(String name, int index) – удалить значение множественного атрибута с указанным индексом. Индексы элементов, лежащих после удаляемого уменьшаются на единичку.
5. void truncateAttrValue(String name, int index) – Удалить все элементы начиная с элемента с указанным индексом и до конца списка значений множественного атрибута.
6. Object getAttrValue(String name, int index) – дать значение элемента множественного атрибута с указанным индексом.
7. List<Object> getAllAttrValues(String name) – дать список значений множественного атрибута.

Вполне вероятно для уменьшения ошибок программирования нужно будет создать методы, дублирующие методы присвоения и получения значений как одиночных, так и множественных атрибутов, но для определенных типов данных. Например:

1. Integer getIntAttr(String name)
2. void setDoubleAttr(String name, Double value)
3. Long getLongAttrValue(String name, int index)
4. List<String> getStringAllValues(String name)
5. void setDateAttrValue(String name, int index, Calendar newDate)

Полный список методов универсального хранимого объекта определяется на этапе разработки спецификации подсистемы хранения.

### Бизнес-логика универсального объекта

Основная бизнес логика универсального объекта заключается в отслеживании изменения своих атрибутов.

Универсальный объект должен в любой момент дать список атрибутов, которые изменялись у него с некоторого момента времени. Под таким моментом времени может пониматься, например, вызов специального метода, который сбрасывает список изменений в универсальном объекте.

Было бы очень желательно для универсального объекта возможность выдать для измененных атрибутов как старое, так и новое значение атрибутов. Такое свойство универсального объекта очень сильно облегчит ведение журналов объектного аудита.

Универсальный объект не содержит в себе логики сохранения данных – данная функция лежит на компоненте интерфейса манипулирования данных для системы в целом и на компоненте генерации запросов манипулирования данных для подсистемы хранения.

## Соглашения по использованию имен объектов и таблиц объектов

Предлагаются следующие соглашения по именам объектов и таблиц объектов.

Имя объект должно отвечать следующим требованиям:

1. Имя объекта может содержать следующие символы:
   1. Буквы латинского алфавита ([a-z|A-Z])
   2. Цифры ([0-9])
   3. Знак подчеркивания (“\_”)
2. Имя объекта всегда должно начинаться с буквы латинского алфавита
3. Длина имени объекта не может превышать 26 символов
4. В имени объектов не различается регистр букв (т.е. не различают строчных и прописных букв: “a” == “A”)

При описании системы рекомендуется использовать строчные буквы для имен объектов.

Имена таблиц зависят от имени объекта:

1. Для таблицы с не множественными атрибутами имя таблицы  
   <имя-объекта>\_s
2. Для таблицы с множественными атрибутами имя таблицы  
   <имя-объекта>\_m

## Соглашения по использованию имен полей таблицы

Имя атрибута должно отвечать следующим требованиям:

1. Имя атрибута может содержать следующие символы:
   1. Буквы латинского алфавита ([a-z|A-Z])
   2. Цифры ([0-9])
   3. Знак подчеркивания (“\_”)
2. Имя атрибута всегда должно начинаться с буквы латинского алфавита
3. Длина имени атрибута не может превышать 26 символов
4. В имени атрибута не различается регистр букв (т.е. не различают строчных и прописных букв: “a” == “A”)

Существует специальное назначение использования символов подчеркивания:

1. Рекомендуется использовать один символ подчеркивания (“\_”) для атрибутов виртуальных объектов.
2. Рекомендуется использовать два символа подчеркивания (“\_\_”) для служебных полей, доступ и модификация которых прикладными программистами ограничена. Такие поля могут быть изменены через специальные методы объектов.

## Хранение файлов документов в репозитории системы CompanyMedia 5

Компонент работы с файлами документов позволяет хранить документы в нескольких областях хранения. В общем случае под областью хранения имеется в виду некий способ организации хранения файлов вложений документов. В первой версии реализуется только один из возможных способов хранения вложений документов – хранение документов в файловой системе.

В последующем можно реализовать хранение файлов в БД (в виде BLOB или в виде группы записей), хранение на устройствах с потенциальной адресацией.

### Структура хранения файлов вложений в системе CompanyMedia 5

При хранении в файловой системе область хранения файлов вложений представляет собой директорию в файловой системе операционной системы (ОС) (см. Рис. 11).

Доступ к такой директории и файлам, расположенной в ней имеет только тот пользователь операционной системы, от лица которого работает сервер приложений CM5. Каждая такая область описана записью в базе данных. Дополнительно может быть указана максимальная емкость данной области хранения в Мб.



* Рис. 11 Организация хранения вложений документов

Файлы с вложениями располагаются в поддиректориях директории области хранения. Эти директории создаются динамически.

Каждому файлу ролла (файла содержания ролла) соответствует запись в БД CM5. Данная запись так же хранит дополнительную информацию о вложении, например:

1. дата создания файла вложения (дата его помещения в систему);
2. mime тип вложения;
3. длину вложения;
4. контрольную сумму вложения (MD5 или SHA1)
5. количество ссылок на данную точку входа в хранилище файлов (во скольких объектах данное вложение используется).

На запись с описанием файла вложения ссылается запись ссылки на файл вложения. На запись ссылки на файл вложения ссылаются записи хранимых объектов (с помощью атрибутов, имеющих тип «файл вложения»). Запись ссылки на файл вложения хранит следующую информацию:

1. имя вложения;
2. имя исходного файла вложения;
3. указание на то, каким процессором обрабатывать данное вложение;
4. дата включения данного вложения;
5. пользователь, включивший данный файл;
6. дата последней модификации файла;
7. пользователь выполнивший последнюю модификацию.

Полный список хранимой информации в записи вложения и в записи ссылки на файл вложения формируется на этапе разработки спецификации подсистемы хранения CM5.

Предлагаю учесть, что число файлов, хранимых в системе, может быть не только соизмеримо с числом объектов (полным, не только документов), но даже в несколько раз больше. Например, если как в экспериментах по прототипу ПХ в базе было 20 миллионов документов и еще больше поручений, надо расчитывать на 100 миллионов файлов, соответствующих этому множеству объектов.

А значит, следует экономить, например, на числе записей в СУБД, соответствующих хранимым файлам. Поэтому предлагаемые здесь как минимум по 2 записи на каждый файл – это недопустимо.

Однако, вполне можно использовать то обстоятельство, что файл в нашей системе никогда не используется «сам по себе», а всегда через объект, к которому он присоединен. А точнее, файл всегда присоединяется в определенное «поле» (реквизит, атрибут типа контент) в объекте. Таких атрибутов у объекта (в соответствии с его классом) может быть несколько, и в каждом – любое число файлов.

Проблему дублирования содержимого файлов можно не учитывать, т.к. СЭД – не почта, сама система не должна способствовать своими функциями размножению копий файлов.

Таким образом, исходя из этих обстоятельств, более правильно будет общий список файлов вообще убрать, а записи, соответствующие ссылкам на файлы в объектах, разместить в тех таблицах для классов этих объектов, которые хранят множественные атрибуты. Ссылка на файл – это и есть множественный атрибут, но надо учесть, что таковых в классе может быть несколько с разными названиями.

### Организация областей файловых хранилищ для файлов вложений

Запись вложения содержит уникальный идентификатор файла с вложением (целое число длиной 8 байтов). Данный идентификатор используется для формирования пути и имени файла с содержанием ролла (см. Рис. 12). При этом используется следующий алгоритм:

1. Каждый байт преобразуется в 2 шестнадцатеричные цифры.
2. Каждая пара шестнадцатеричных цифр формирует имя (или имя директории или имя файла).
3. Первые 7 имен используются для создания динамических директорий.
4. Восьмая пара цифр используется для создания имени файла, содержащего вложение.



* Рис. 12 Пример расположения файлов вложений и их связи с записями вложений

В каждой директории не будет находиться ссылок более чем 256 объектов следующего уровня. Общее число хранимых файлов в области хранения может достигать 264-1 ~ 1.8 \* 1019.

### Обеспечение уникальности файлов

В случае необходимости хранить только уникальные файлы, проверку уникальности файлов можно реализовать в операции сохранения файлов в файловом хранилище. При этом при определении уникальности файла может использоваться длина файла и его контрольная сумма. В случае совпадения двух данных характеристик файлов придется выполнять их полное побайтное сравнение.

Дело в том что использование даже контрольной суммы и длины файла не дает полной гарантии в том, что для двух различных файлов эти величины не могут совпасть. Это очень редкий вариант, но его необходимо тоже корректно обрабатывать. В данном случае такая ситуация обрабатывается с помощью прямого сравнения файлов.

# Словарь данных репозитория объектов системы CompanyMedia 5

Словарь данных системы является компонентом подсистемы хранения, который содержит описание всех объектов в системе (в том числе и самого словаря). Любой объект созданный в системе помещается в словарь данных.

Словарь данных устанавливается в систему при создании БД хранилища объектов при создании репозитория. Словарь данных модифицируется компонентом создания структур объектов. Они используется следующими компонентами:

1. компонентом создания структур объектов;
2. компонентом генерации запросов манипулирования данными;
3. компонентом генерации поисковых запросов.

Словарь данных должен обеспечивать выполнение следующих основных функций:

1. Для любого класса объектов необходимо получить базовый класс для данного объекта.
2. Для любого класс объекта необходимо получить полный список классов объектов-родителей.
3. Для любого класса объекта необходимо получить список таблиц, в которых храниться данный объект.
4. Для любого класса объекта необходимо получить полный список атрибутов данного класса. При этом к данному списку предъявляются следующие требования:
   1. Для каждого атрибута необходимо знать его имя.
   2. Для каждого атрибута необходимо знать его тип.
   3. Для каждого атрибута необходимо знать множественны он или одиночный.
   4. Для каждого атрибута необходимо знать его модификатор длины (размерность для чисел, длину для строковых значений).
   5. Для каждого атрибута необходимо знать в каком классе объекта-родителя он определен.
   6. Для каждого атрибута необходимо знать имя таблицы БД, в которой он хранится.

## Структура словаря данных

Структура словаря данных состоит из двух объектов (см. Рис. 13):

1. описание объектов;
2. описание атрибутов объектов.



* Рис. 13 Структура словаря данных

Для атрибутов виртуальных классов объектов имени таблицы не указывается. В классах объектов-наследников виртуальных объектов атрибуты виртуальных объектов повторяются с указанием имен таблиц, в которых хранятся данные атрибуты, унаследованные от виртуального объекта (на Рис. 13 так наследуются атрибуты объекта root\_object: id\_\_, update\_cnt\_\_, last\_modify\_time). Для атрибутов, унаследованных от атрибутов виртуальных объектов указывается ссылка на описание атрибута, от которого унаследован данный атрибут объекта.

Такая организация данных обеспечивает наличие всех атрибутов, реально хранимых в БД, в справочнике атрибутов словаря данных с указанием реальных таблиц, используемых для хранения информации. Это облегчает алгоритмы построения операторов манипулирования данными в компонентах генерации поисковых запросов и генерации запросов манипулирования данными.

## Свойства классов объектов

Данный раздел содержит перечень свойств классов объектов, хранимых в словаре данных (см. Таблица 1). Перечень свойств класса объекта уточняется на этапе разработки спецификации подсистемы хранения CM5.

Таблица 1. Перечень свойств класса объекта   
в словаре данных системы CompanyMedia 5

| № | Имя свойства | Тип | Комментарий |
| --- | --- | --- | --- |
| 1 | class\_id | int | Идентификатор класса |
| 2 | name | string(26) | Имя класса объекта. Максимальная длина имени – 26 символов.  Имена таблиц для хранения объектов получаются по зашитым в систему правилам (см. п. 6.4) |
| 3 | virtual | boolean | Признак виртуального класса объекта. |
| 4 | is\_multiply | boolean | Класс объекта имеет множественные атрибуты.  Если значение true, то для такого объекта существует таблица множественных атрибутов. |
| 5 | full\_text | boolean | Объекты данного класса подвергаются полнотекстовому индексированию.  Такие объекты обрабатываются подсистемой полнотекстовой индексации. |
| 6 | restricted | boolean | Доступ к объектам контролируется с помощью ACL.  Достигается наследованием от виртуального класса restricted\_obj.  Атрибут избыточный, но упрощает алгоритмы для подсистемы хранения. Автоматически проставляется для класса объекта если среди классов объектов-родителей встречается требуемый класс. |
| 7 | version | boolean | Объект может иметь версии.  Среди классов объектов-родителей должен присутствовать виртуальный объект version\_obj.  Атрибут избыточный, но упрощает алгоритмы для подсистемы хранения. Автоматически проставляется для класса объекта если среди классов объектов-родителей встречается требуемый класс. |
| 8 | replicative | boolean | Объект может участвовать в репликации.  Среди классов объектов-родителей должен присутствовать виртуальный объект replicative\_obj.  Атрибут избыточный, но упрощает алгоритмы для подсистемы хранения. Автоматически проставляется для класса объекта если среди классов объектов-родителей встречается требуемый класс. |
| 9 | parent\_objects | short\* | Множественный атрибут. Содержит ссылки на базовые классы для данного класса.  Если имен базовых классов больше одного (множественное наследование), то все базовые классы (возможно кроме одного) должны быть виртуальными. Среди базовых классов не может быть более одного структурного класса (см. п. 6.1.2). |
| 10 | system | boolean | Признак системного класса объекта. Такой объект служит для правильного функционирования системы. Его нельзя менять.  Все атрибуты такого объекта всегда имеют установленный флаг business. |
| 11 | business | boolean | Класс объекта бизнес-решения. Такой класс объекта поставляется вместе с бизнес-решением. Для такого объекта возможно добавление новых атрибутов.  Модификация и удаления атрибутов такого объекта с установленным флагом business невозможна, т.к. они определены для бизнес-решения и должны оставаться неизменными. |
| 12 | collection | boolean | Признак того, что данный класс объекта принадлежит к промежуточным данным коллекций.  Для таких классов существуют специальные |
| 13 | db\_id | id\* | Список ссылок на записи БД, в которых храниться объект |

Для объектов коллекций class\_id < 0, для хранимых объектов class\_id > 0.

class\_id лежит в диапазоне от -32 768 до 32 767. Значение “0” не используется. Таким образом в системе может быть до 32768 классов объектов коллекций и до 32768 классов хранимых объектов.

## Свойства атрибутов объекта

Данный раздел содержит перечень свойств атрибутов объектов, хранимых в словаре данных. Перечень свойств атрибутов объекта уточняется на этапе разработки спецификации подсистемы хранения CM5.

Таблица 2. Перечень свойств атрибутов объекта   
в словаре данных системы CompanyMedia 5

| № | Имя свойства | Тип | Комментарий |
| --- | --- | --- | --- |
| 1 | attr\_id | int | Идентификатор атрибута |
| 2 | name | string(26) | Имя атрибута. Максимальная длина имени атрибута – 26 символов. |
| 3 | type | short | Код типа атрибута. |
| 4 | length | int | Длина атрибута. сейчас используется только для строковых атрибутов. |
| 5 | precision | int | Точность атрибута. В данной версии не используется. |
| 6 | is\_multiply | boolean | Признак множественного атрибута. |
| 7 | full\_text | boolean | Данный атрибут входит в полнотекстовой индекс. |
| 8 | business | boolean | Атрибут бизнес-решения. Такой атрибут не может быть изменен или удален. |
| 9 | table | string(28) | Имя таблицы для данного атрибута. |
| 10 | parent\_attr | int | Ссылка на запись атрибута виртуального класса объекта. Если данное свойство не равно null, то такой атрибут не может быть изменен. Для его изменения необходимо изменить описание виртуального класса. |

Значение идентификатора атрибута attr\_id находится в диапазоне от 1 и до 2 147 483 647. Атрибуты одного объекта могут иметь не последовательные идентификаторы. Это происходит в том случае, когда атрибуты добавляются при модификации объектов.

## Создание новых и модификация существующих классов объектов

Созданием и модификацией классов объектов занимается компонент создания структуры объектов.

При создании новых объектов производятся проверки правил наследования, принятых в системе (см. п. 6.1.2).

При создании виртуального объекта компонент создания структуры объектов модифицирует словарь и на этом работа заканчивается.

При создании структурного или листового класса сначала создаются или модифицируются объекты описания классов объектов словаря данных. Затем выполняются операторы SQL для модификации структуры БД. После успешного выполнения модификации БД созданные (модифицированные) описания объектов сохраняются в словаре.

## Типы данных в репозитории объектов

Подсистема хранения поддерживает фиксированное число типов атрибутов (см. Таблица 3). Подсистема хранения не поддерживает механизма расширения типов атрибутов.

Таблица 3. Типы данных в подсистеме хранения  
системы CompanyMedia 5

| № | Наименование типа | Код типа | Описание |
| --- | --- | --- | --- |
| 1 | id | 1 | Идентификатор объекта. Содержит информацию для однозначной идентификации объекта. |
| 2 | string | 2 | Строка. Для строки всегда должна задаваться максимально возможная длина.  Максимальная длина ограничена величиной, зависящей от РСУБД, используемой для построения системы. В большинстве случаев можно считать то, что максимальны длина будет больше или равна 4096 байтам. |
| 3 | int | 3 | Целое число. |
| 4 | long | 4 | Длинное целое число. Количество знаков больше, чем в целом числе типа int. |
| 5 | date | 5 | Дата без времени. Дата содержит информацию о часовом поясе. |
| 6 | datetime | 6 | Дата с временем. Дата содержит информацию о часовом поясе. |
| 7 | timestamp | 7 | Точная дата и время без информации о часовом поясе. |
| 8 | boolean | 8 | Логическое значение. |
| 9 | double | 9 | Число с плавающей точкой. |
| 10 | placement | 10 | Ссылка на файл вложения (см. п. 6.6). |
| 11 | short | 11 | Короткое целое число. Количество знаков меньше, чем в целом числе типа int. |
| 12 | text | 12 | Длинный текст, хранимый непосредственно в базе данных.  *В первой версии подсистемы хранения не реализуется.* |
| 13 | binary | 13 | Длинные бинарные данные, хранимые непосредственно в базе данных.  *В первой версии подсистемы хранения не реализуется.* |

### Отображение базовых типов объектов на объекты Java

Данный раздел содержит отображение типов данных атрибутов подсистемы хранения на типы данных языка программирования Java (см. Таблица 4).

Таблица 4. Отображение типов данных атрибутов классов объектов   
системы CompanyMedia 5 на типы данных Java

| № | Наименование типа | Тип Java | Описание |
| --- | --- | --- | --- |
| 1 | id | ObjectId | Тип данных для идентификаторов системы CM5 |
| 2 | string | String |  |
| 3 | int | Integer, int[[2]](#footnote-3) |  |
| 4 | long | Long, long |  |
| 5 | date | Calendar | Календарь используется для того, чтобы иметь возможность хранить часовой пояс. |
| 6 | datetime | Calendar | Календарь используется для того, чтобы иметь возможность хранить часовой пояс. |
| 7 | timestamp | Calendar | Календарь используется для общности интерфейса. |
| 8 | boolean | Boolean, boolean |  |
| 9 | double | Double, double |  |
| 10 | placement | Placement | Тип данных для ссылки на файлы вложений. В первой версии совпадает с типом ObjectId. |
| 11 | short | Short, short |  |
| 12 | text | LongText | Специализированный объект работы с длинным текстом. |
| 13 | binary | LongBinary | Специализированный объект работы с длинными двоичными данными. |

### Отображение базовых типов объектов на БД Oracle

Данный раздел содержит отображение типов данных атрибутов подсистемы хранения на типы данных языка программирования Java (см. Таблица 4).

Таблица 5. Отображение типов данных атрибутов классов объектов   
системы CompanyMedia 5 на типы данных РСУБД Oracle

| № | Наименование типа | Тип Oracle | Описание |
| --- | --- | --- | --- |
| 1 | id | VARCHAR2(30) | Тип данных для идентификаторов системы CM5 (см. п. ). Длина - 27 байт. |
| 2 | string(n) | NVARCHAR2(n) | Данный переменная - максимум 3\*n+3 байт. |
| 3 | int | NUMBER(10,0) | Длина переменная – максимум 7 байт. |
| 4 | long | NUMBER(19,0) | Длина переменная – максимум 11 байт. |
| 5 | date | DATE | Длина – 7 байт. |
| 6 | datetime | TIMESTAMP WITH TIME ZONE | Длина – 13 байт. Время с точностью до миллисекунд. |
| 7 | timestamp | TIMESTAMP | Длина – 9 байт. Время с точностью до миллисекунд. |
| 8 | boolean | CHAR(1) | Длина – 1 байт. |
| 9 | double | BINARY\_DOUBLE | Длина – 9 байт. |
| 10 | placement | VARCHAR2(26) | Тип данных для ссылки на файлы вложений. Длина - 27 байт. |
| 11 | short | NUMBER(5,0) | Длина 4 байта. |
| 12 | text | NCLOB | Длина неизвестна. Данные хранятся отдельно от записи. |
| 13 | binary | BLOB | Длина неизвестна. Данные хранятся отдельно от записи. |

### Отображение базовых типов объектов на БД PostgreSQL

Данный раздел содержит отображение типов данных атрибутов подсистемы хранения на типы данных языка программирования Java (см. Таблица 4).

Таблица 6. Отображение типов данных атрибутов классов объектов   
системы CompanyMedia 5 на типы данных РСУБД PostgreSQL

| № | Наименование типа | Тип PostgreSQL | Описание |
| --- | --- | --- | --- |
| 1 | id | varchar(30) | Тип данных для идентификаторов системы CM5. Занимает 27 байт. |
| 2 | string(n) | character varying(n),  varchar(n)[[3]](#footnote-4) | Для работы с несколькими строками кодировка базы должна быть установлена в UTF-8 |
| 3 | int | integer, int, int4 | Занимает 4 байта. |
| 4 | long | bigint, int8 | Занимает 8 байт. |
| 5 | date | date | Занимает 4 байта. |
| 6 | datetime | timestamp with time zone | Занимает 8 байт |
| 7 | timestamp | timestamp | Занимает 8 байт |
| 8 | boolean | char(1) | Несмотря на то, что PostgreSQL имеет тип данных bool (boolean) для хранения логических значений будет использоваться строковый тип (как и в остальных РСУБД) для обеспечения единого подходя для хранения логических значений во всех РСУБД. Занимает 1 байт. |
| 9 | double | double precision, float8 | Занимает 8 байт. |
| 10 | placement | varchar(26) | Тип данных для ссылки на файлы вложений. Занимает 27 байт. |
| 11 | short | smallint, int2 | Занимает 2 байта. |
| 12 | text | text[[4]](#footnote-5) | В следствие особенности реализации JDBC интерфейса работа с данным типом ведется через временные файлы на дисках. Точная длина неизвестна. |
| 13 | binary | bytea[[5]](#footnote-6) | В следствие особенности реализации JDBC интерфейса работа с данным типом ведется через временные файлы на дисках. Точная длина неизвестна. |

# Системные объекты

В данном разделе содержаться некоторые сведения о базовых объектах CM5. Окончательно список системных объектов и их атрибутов определяется на этапе проектирования спецификации подсистемы хранения CM5.

## Объекты словаря данных

Структура словаря данных описывается в терминах таблиц БД. При этом используются типы данных, определенные для ПХ CМ5, что позволяет эти таблицы отобразить на любые РСУБД с которыми может функционировать СМ5.

Структура словаря данных состоит из следующих таблиц (см. Рис. 14):

1. class\_dict – таблица с описанием классов объектов CM5
2. class\_virtual\_parent\_dict – таблица для множественной связи класса объекта с базовыми виртуальными классами
3. db\_link\_dict – список ссылок на объекты БД, в которых могут сохранятся объекты данного класса
4. attribute\_dict – таблица с описанием атрибутов классов объектов.



* Рис. 14 Структура объектов словаря данных

Каждой таблице соответствует объект ПХ. Программный интерфейс доступа к словарю данных определяется на этапе разработки спецификации подсистемы хранения CM5.

## Базовые виртуальные объекты

Предлагаемая схема (см. Рис. 15) базовых объектов является предварительной. Окончательный состав базовых объектов для построения хранимых объектов CM5 определяется на этапе разработки спецификации подсистемы хранения CM5.

Все виртуальные объекты в системе CM5 имеют суффикс “\_obj”.

Основное назначение объектов:

1. persistent\_obj – объект, обеспечивающий оптимистическую блокировку объектов в системе.
2. checkout\_obj – объект, обеспечивающий долговременную блокировку объектов.



* Рис. 15 Структура базовых объектов системы CompanyMedia 5

1. restricted\_obj – объект, обеспечивающий работу со списками доступа к объекту.
2. version\_obj – объект, обеспечивающий работу с версиями объектов
3. replicative\_obj – объект, обеспечивающий работу репликации объектов между несколькими удаленными хранилищами.
4. his\_event\_obj – объект обеспечивающий быструю регистрацию основных событий в жизненном цикле объекта (создание и последнее изменение)
5. document\_root – структурный объект, являющийся корнем для всех документов системы.
6. dict\_obj – коревой виртуальный объект для справочников системы.
7. hierarchy\_dict\_obj – корневой виртуальный объект для иерархических справочников системы.
8. restricted\_dict\_obj – объект, обеспечивающий ведение справочников с «историчностью», т.е. с учетом времен начала и конца действия позиции справочника.

## Служебные объекты и таблицы

Предлагаемая схема (см. Рис. 16) служебных таблиц и объектов является предварительной. Окончательный состав базовых объектов для построения хранимых объектов CM5 определяется на этапе разработки спецификации подсистемы хранения CM5.

В состав служебных таблиц и объектов входят следующие сущности:

1. Counts – объект, содержащий значения счетчиков для всех объектов в этой базе данных



* Рис. 16 Служебные таблицы и объекты

1. cm5\_config – конфигурация текущей установки
2. db\_config – конфигурация текущей базы данных
3. site\_dict – словарь всех сайтов системы. Словарь реплицируется между всеми серверами системы.
4. server\_dict – словарь серверов системы. Словарь реплицируется между всеми серверами системы.
5. db\_dict –словарь баз данных текущего сервера.
6. object\_history – Объект с записями истории обработки объектов системы.
7. acl – объект для хранения списков доступа на чтение для объектов системы.

# Алгоритмы выполнения основных операций

## Распределения объектов по БД объектов

В качестве базового алгоритма распределения данных между БД объектов хранилища предполагается использование алгоритма «Round robin».

При этом следует учесть тот факт, что для хранилища, основанного на реляционной модели, все объекты одного агрегата для повышения эффективности работы хранилища желательно помещать в одну БД объектов. В то же время о структуре агрегатов знает только прикладное решение (бизнес-решение)

В связи с необходимостью уметь размещать объекты в определённой БД хранилища объектов API ПХ должно предусматривать как автоматический выбор целевой БД, так и возможность задать БД для размещения объекта принудительно.

## Алгоритм нумерации объектов

Алгоритм нумерации объектов должен соответствовать следующим требованиям [2]:

1. Уникально идентифицировать объект относительно хранилища объектов
2. Уникально идентифицировать репозиторий на котором находится объект
3. Поиск объекта по хранилищу должен выполняться без справочников, определяющих размещение объектов, и без поиска по всей сети
4. Идентификатор объекта не меняется в течение всего его жизненного цикла.
5. Те же требования справедливы для идентификаторов версий.

В связи с этим уникальный идентификатор включает в сой состав сведения о местоположении объекта при создании.

Уникальный идентификатор объекта состоит из нескольких частей:

1. Идентификатор сайта (1 ‑ 255). Целое. Длина - 1 байт.
2. Идентификатор хранилища (1 ‑ 255). Целое. Длина – 1 байт.
3. Идентификатор БД (1 ‑ 255). Целое. Длина – 1 байт.
4. Идентификатор типа объекта (-32768 ‑ 32767). Целое. Длина – 2 байта.
5. Уникальный идентификатор объекта в БД хранилища внутри типа (1 ‑ 9 223 372 036 854 775 807). Целое. Длина – 8 байт.

Идентификатор представляет собой строку длиной 26 ? символов. Строка идентификатора формируется по следующим правилам:

1. Из всех чисел формируется байтовая строка длиной 13 ? байт. Числа берутся в порядке перечисления частей идентификатора:
2. идентификатор сайта;
3. идентификатор хранилища;
4. идентификатор БД;
5. идентификатор типа объекта;
6. идентификатор объекта в БД хранилища.
7. После этого полученная строка байт преобразуется в строку шестнадцатеричного представления этой строки длиной 30 байт (2 символа на 1 байт исходной байтовой строки)

В результате получаем уникальный идентификатор объекта (см. ).

Таблица 7. Структура идентификатора объекта в CompanyMedia 5

| Смещение | Длина в символах | Содержание |
| --- | --- | --- |
| 0 | 2 | Шестнадцатеричное представление идентификатора сайта |
| 2 | 2 | Шестнадцатеричное представление идентификатора репозитория |
| 4 | 2 | Шестнадцатеричное представление идентификатора БД |
| 8 | 4 | Шестнадцатеричное представление идентификатора типа объекта |
| 12 | 16 | Шестнадцатеричное представление идентификатора объекта внутри типа |

Примеры идентификаторов.

1. 010101000a00000000000003a4 – объект, созданный на первом сайте, первом репозитории, находится в базе номер 1, имеет 10 класс объекта и идентификатор 932. Объект относится к БД хранилища объектов.
2. 030105fffb0000000000001e32 – объект, созданный на сайте 3, первом репозитории, находится в базе номер 5, имеет -5 класс объекта и идентификатор 7730. Объект относится к БД коллекций (номер класса объекта отрицательный).

## Алгоритм оптимистической блокировки

Алгоритм предотвращает конфликты между конкурирующими транзакциями, выявляя их и откатывая транзакцию назад. Алгоритм оптимистической блокировки позволяет уменьшит время удержания записей БД в заблокированном состоянии, что позволяет уменьшит нагрузку на БД.

При разработке программного обеспечения необходимо выбирать стратегию блокировок данных. При этом следует учитывать следующее:

1. Если ситуация обновления одних и тех же данных в один момент времени относительно редка, то выгоднее использовать оптимистичную блокировку. Если же возможность возникновения ситуации обновления одних и тех же данных достаточно высока, то лучше использовать пессимистичную блокировку, это снизит количество прерванных транзакций.
2. Также следует учитывать, что при оптимистичной блокировке в случае прерывания транзакции её нужно запускать заново.
3. Пессимистичный алгоритм использует более дорогие механизмы блокировок на уровне БД.
4. При долговременном обновлении записей (например, в пользовательском интерфейсе) необходимо применять или оптимистическую блокировку или долговременную блокировку (cм. п. 9.4).
5. Оптимистическая блокировка позволяет

Оптимистическая блокировка требует наличия в записи БД специального счетчика изменения. Такой счетчик обеспечивается виртуальным объектом persistent\_obj (см. п. 8.2). Имя счетчика write\_cnt\_\_.

Рассмотрим алгоритм оптимистической блокировки по шагам (см. Рис. 17):

1. Читаем объект в бизнес-логику



* Рис. 17 Алгоритм оптимистической блокировки

1. При этом ПХ CM5 выдает оператор SELECT> возвращающий все атрибуты объекта. Среди этих атрибутов передается текущее значение счетчика изменений write\_cnt\_\_.
2. В бизнес-логике производится обработка объекта.
3. Бизнес-логика сохраняет объект.
4. Для этого вызывается метод «Изменить объект» в ПХ CM5.
5. Первым делом выдается оператор   
   SELECT write\_cnt\_\_ FROM some\_obj WHERE id\_\_ = ? FOR UPDATE,  
   который возвращает текущее значение счетчика изменений в хранимом в БД объекте.  
   Этот оператор так же блокирует объект от изменения другим процессом до конца транзакции.
6. Если счетчик, хранимый в объекте, используемом в бизнес-логике, (old write\_cnt\_\_) совпадает с текущем значением счетчика, полученного из объекта хранимого в базе (new write\_cnt\_\_), то производится сохранение данных. Равенство счетчиков означает то, что за время обработки никто еще не изменил объект в БД.
7. В процессе сохранения прежде всего производится приращение счетчика числа изменений.
8. Затем выполняется обновление объекта в БД.
9. После чего производится нормальное завершение транзакции БД.
10. После этого управление возвращается в бизнес-логику.
11. Если же счетчик, хранимый в объекте, используемом в бизнес-логике, (old write\_cnt\_\_) не совпадает с текущем значением счетчика, полученного из объекта хранимого в базе (new write\_cnt\_\_), то производится обработка столкновения транзакций. неравенство счетчиков означает то, что кто-то за время обработки объекта уже изменил его.
12. При обработке столкновения транзакций выполняется откат транзакции БД.
13. Управление в бизнес-логику возвращается с помощью вызова исключительной ситуации, соответствующей столкновению транзакций.

## Алгоритм долговременной блокировки над объектом

Долговременная блокировка должна быть «глобальной» - объект «захватывается» с учетом того, что его «реплики» могут существовать на нескольких СМ-сайтах. Для этого используется понятие «мастер-сайта» объекта, см. в моем документе «СМ5\_Платформа (Панов В.А.) v0.2 20121115.doc».

Описанные ниже правила работают, если текущий СМ-сайт совпадает с мастер-сайтом объекта. Иначе при попытке check-out сначала надо делать «запрос» на мастер-сайт для временного переноса мастер-сайта объекта на текущий.

Долговременная блокировка требует наличия в записи БД специальных полей для отметки факта блокировки. Такие поля обеспечивается виртуальным объектом checkout\_obj (см. п. 8.2). В ПХ CM5 в качестве таких полей используют идентификатор пользователя, осуществившего блокировку и время установки долговременной блокировки.

Объект можно редактировать только если идентификатор пользователя, осуществившего блокировку равен null. Данная проверка лежит на бизнес-логике системы.

ПХ CM5 сможет сохранить объект только если выполнены следующие условия:

1. Новое значение идентификатора пользователя, заблокировавшего объект, или пустой (null) или равно идентификатору пользователя, проводящего данное изменение объекта.
2. В хранимом в базе данных объекте значение идентификатора пользователя, заблокировавшего объект, или пустой (null) или равно идентификатору пользователя проводящего данное изменение объекта.
3. Если идентификатора пользователя, заблокировавшего объект, имеет пустое значение (null), то поле времени установки блокировки так же очищается.
4. Если идентификатор пользователя, заблокировавшего объект, имеет значение идентификатора пользователя, проводящего данное изменение объекта, и в БД идентификатор пользователя пуст, то полю времени установки блокировки присваивается значение текущего момента времени.
5. Если идентификатор пользователя, заблокировавшего объект, имеет значение идентификатора пользователя, проводящего данное изменение объекта, и в БД идентификатор пользователя не пуст, то поле времени установки блокировки остается без изменения.

При любом другом значении атрибутов операция модификации объекта, поддерживающем долговременную блокировку, завершается с помощью вызова исключительной ситуации, соответствующей столкновению долговременных транзакций.

## Алгоритмы работы с версиями документов

Для поддержки работы бизнес логики системы ПХ CM5 должна поддерживать работу в версиями объектов [2, 3, 4, 7]

Для работы с версиями предусмотрены следующие операции:

1. Создать новую версию
2. Получить текущую версию
3. Взять указанную версию
4. Удалить документ (включая все версии)

При создании объекта всегда создаётся версия по умолчанию 1. Дальнейшие запросы по изменению версии приводят к увеличению версии. Возможен откат на версию назад только через создание новой версии. Удаление или вставка в середину не возможна. Ветвление версий невозможно.

Таблица 8. Операции версионности

| Операция | Действие |
| --- | --- |
| Создать новую версию | Текущая версия увеличивается на 1. Создаётся новая строка с новыми данными и помечается текущей. |
| Получить текущую версию | Получить объект по уникальному идентификатору и версию, которая помечена как текущая |
| Взять необходимую версию | Получить объект по уникальному идентификатору версии |
| Удалить документ (включая все версии) | Удалить все версии документа |

## Алгоритмы поддержки работы с репликацией объектов

Репликация объектов между базами требует наличия в записи БД специальных полей для отметки факта репликации. Такие поля обеспечивается виртуальным объектом replicative\_obj (см. п. 8.2). В ПХ CM5 в качестве таких полей используют идентификатор исходного объекта (r\_id\_\_) и счетчик изменений исходного объекта (r\_write\_cnt\_\_).

При создании нового объекта поле r\_id\_\_ устанавливается в то же значение,что и поле идентификатора объекта id \_\_.

Поиск объектов, которые могут быть реплицированы, всегда производится по полю r\_id\_\_. Если объект по данному полю не найден, то такой объект недоступен в данном хранилище.

При обратной репликации сравнение полей реплицируемого объекта r\_write\_cnt\_\_ и поля исходного объекта write\_cnt\_\_ позволяет определить конкурентное изменение объектов. Конкурентное изменение реплики и исходного объекта разрешается на уровне бизнес-логики системы.

При репликации в качестве идентификатора реплицируемого объекта всегда передается поле r\_write\_cnt\_\_, но в качестве значения текущего счетчика изменений передается значение поля write\_cnt\_\_. Это позволяет при многоступенчатой репликации сохранят идентификатор исходного объекта и отмечать факты изменения объекта не только в исходном репозитории, но и в промежуточном репозитории.

Более подробно алгоритмы репликации описаны в документе «Архитектура подсистемы репликации».

Приложение 1 Словарь терминов

| Термин, сокращение | Расшифровка |
| --- | --- |
| API | Application Program Interface.  Интерфейс программирования приложений — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах.  API определяет функциональность, которую предоставляет программа (модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована. |
| Application Server | Application Server.  Сервер приложений - это программная платформа предназначенная для эффективного исполнения процедур (программ, механических операций, скриптов), которые поддерживают построение приложений. Сервер приложений действует как набор компонентов, доступных разработчику программного обеспечения через API, который определен самой платформой.  В данном документе под Application Server подразумевается Lightweight J2EE Application Server, т.е. J2EE AS без EJB контейнера. |
| AS | Application Server |
| CM5 | CompanyMedia version 5. |
| CRUD | Create, Read, Update, Delete.  Стандартный набор операций над объектами базы данных, который позволяет описать полный алгоритм взаимодействия в хранимыми данными. |
| CRVUD | Create, Read, Version, Update, Delete.  Расширенное множество CRUD операций. Множество операций расширено операцией Version, позволяющей создать новую версию объекта Может использоваться вместо операции Update. Обычно имеет более низкий уровень защиты, чем операция Update.  Применяется для систем хранения, поддерживающих версионность объектов. |
| IoC | Inversion of Control.  Инверсия управления— важный принцип объектно-ориентированного программирования, используемый для уменьшения связанности в компьютерных программах.  Формулировка:   * Модули верхнего уровня не должны зависеть от модулей нижнего уровня. Оба должны зависеть от абстракции. * Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.   Для Java реализуется в виде IoC контейнеров. Одна из реализаций – Spring Framework.  Метод применения: Программное обеспечение запрашивает объект из контейнера, и контейнер создаёт объект и его зависимости. |
| J2EE | Java Platform, Enterprise Edition, сокращенно Java EE (до версии 5.0 — Java 2 Enterprise Edition или J2EE) — набор спецификаций и соответствующей документации для языка Java, описывающей архитектуру серверной платформы для задач средних и крупных предприятий. |
| J2SE | Java Platform, Standard Edition, сокращенно Java SE (ранее Java 2 Standard Edition или J2SE) — стандартная версия платформы Java 2, предназначенная для создания и исполнения аплетов и приложений, рассчитанных на индивидуальное пользование или на использование в масштабах малого предприятия. Не включает в себя многие возможности, предоставляемые более мощной и расширенной платформой Java 2 Enterprise Edition (J2EE), рассчитанной на создание коммерческих приложений масштаба крупных и средних предприятий. |
| Java EE | См. J2EE |
| Java SE | См. J2SE |
| JDBC | Java DataBase Connectivity.  Соединение с базами данных на Java — платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE. |
| Round-robin | Round-robin — алгоритм распределения нагрузки распределённой вычислительной системы методом перебора и упорядочения её элементов по круговому циклу. |
| SOA | Service-oriented architecture Сервис-ориентированная архитектура (SOA, англ. service-oriented architecture) — модульный подход к разработке программного обеспечения, основанный на использовании распределённых, слабо связанных (англ. loose coupling) заменяемых компонентов, оснащённых стандартизированными интерфейсами для взаимодействия по стандартизированным протоколам.  Программные комплексы, разработанные в соответствии с сервис-ориентированной архитектурой, могут быть реализованы, например, как набор Web служб, взаимодействующих по протоколу SOAP, но возможны и другие реализации (например, на базе jini, CORBA).  Интерфейсы компонентов в сервис-ориентированной архитектуре инкапсулируют детали реализации (операционную систему, платформу, язык программирования) от остальных компонентов, таким образом обеспечивая комбинирование и многократное использование компонентов для построения сложных распределённых программных комплексов, обеспечивая независимость от используемых платформ и инструментов разработки, способствуя масштабируемости и управляемости создаваемых систем. |
| БД | База данных — представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ)  (Гражданский кодекс РФ, ст. 1260). |
| Бизнес-решение | Бизнес-решение – это относительно самодостаточный прикладной модуль, способный к самостоятельному развертыванию и использованию (без других БР), но интегрирующиеся с ними на принципах слабой связности. [2] |
| БР | Бизнес-решение. |
| Виртуальный класс объектов | Класс объектов, не имеющих таблиц в БД.  Используются в ПХ для описания шаблонов структур хранения, которые могут использоваться в других классах объектов ПХ. |
| Листовой класс объектов | Класс объектов, порождающий таблицы в БД, но не имеющий наследников. |
| ОС | Операционная система. |
| ПХ | Подсистема хранения. Компонент системы CompanyMedia 5, отвечающий за манипулирование информацией, хранимой в БД системы и в хранилище файлов вложений. |
| РСУБД | Реляционная Система управления Базами данных. СУБД, использующая реляционные принципы манипулирования с информацией, хранимой в БД. |
| Структурный класс объектов | Класс объектов, порождающий таблицы в БД, и имеющий наследников. |
| СУБД | Система Управления Базами Данных - совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных[. |

**Замечания, добавленные 22.11.2012**

Некоторые моменты, относящиеся к ПХ, являются ключевыми для работы всей системы, они должны быть описаны:

1. Как будут реализованы связи (включая их разновидности и раскладку объектов по множеству БД репозитория).
2. Как будет реализован ACL, особенно с учетом связей и распределения объектов по БД
3. Механизм распределения объектов по множеству БД репозитория, особенно с учетом "жестких связей" между некоторыми типами объектов, например, иерархия поручений/отчетов - со своим "корневым" документом.

По 3-му пункту есть предложение:

1. В UID любого объекта (и версии тоже) выделить небольшой фрагмент (достаточно пары байтов), на основе которого вычисляется "номер" БД репозитория. Только это не в коем случае не сам номер БД, как это написано сейчас. Это может быть случайное число, а номер БД получается его делением на число БД.
2. При создании любого объекта, жестко связанного с уже существующим объектом (обычно корневым в иерархии или просто в отношении 1-ко-многим), в его UID копируется часть UID корневого объекта, отвечающая за его размещение в БД. Объект-потомок как бы наследует "фрагмент ДНК" от прародителя.

1. Предложения являются предварительными. Существует различные синтаксисы работы с множественными атрибутами (см. документацию на PostrgeSQL, Documentum, FileNet) [↑](#footnote-ref-2)
2. Через запятую в данной таблице указывается элементарный тип Java, соответствующий [↑](#footnote-ref-3)
3. Через запятую указан синоним типа, который может быть использован наравне с основным типом. [↑](#footnote-ref-4)
4. В документации информации о том, как храниться тип text нет. [↑](#footnote-ref-5)
5. В документации информации о том, как храниться тип bytea нет. [↑](#footnote-ref-6)