**Правительство Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Национальный исследовательский университет   
«Высшая школа экономики»

**Факультет Бизнес-информатики**

**Кафедра корпоративных информационных систем**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

на тему   
«Методы моделирования данных в аналитических информационных системах»

Студент группы № 471

Исерсон Никита Маркович

Руководитель ВКР  
Моргунов Александр Федорович, к.т.н., доцент

Москва, 2013

Содержание

[Используемые сокращения 2](#_Toc357713403)

[Введение 3](#_Toc357713404)

[1. Постановка задачи 5](#_Toc357713405)

[1.1 Техническая постановка задачи 6](#_Toc357713406)

[1.2 Постановка задачи с точки зрения бизнеса 8](#_Toc357713407)

[1.3 Текущее решение на платформе SAP BPC on SAP BW 15](#_Toc357713408)

[1.4 Описание существующей проблемы 18](#_Toc357713409)

[2. Предлагаемое решение на основе платформы SAP HANA 19](#_Toc357713410)

[2.1 Анализ существующих решений 19](#_Toc357713411)

[2.2 Модель данных системы планирования 22](#_Toc357713412)

[2.3 Преимущества модели данных на основе SAP HANA 32](#_Toc357713413)

[3. Результаты 39](#_Toc357713414)

[3.1 Форма «Нормы-драйверы» 39](#_Toc357713415)

[3.2 Форма «Амортизация» 41](#_Toc357713416)

[3.2 Основная форма планирования 42](#_Toc357713417)

[Заключение 48](#_Toc357713418)

[Используемая литература 49](#_Toc357713419)

# Используемые сокращения

|  |  |
| --- | --- |
| **Обозначение** | **Описание** |
| SAP BW | SAP Business Warehouse |
| SAP BPC | SAP Business Planning and Consolidation |
| SAP HANA | SAP High-Performance Analytical Appliance |
| OLTP | Online Transaction Processing |
| OLAP | Online Analytical Processing |
| MDX | Multi-Dimensional Expressions |
| SSD | Solid-State Drive |
| ETL | Extraction, Transformation and Loading |
| ИСУ ХД | Информационная система управления хранилищем данных |
| ССП | Самостоятельное структурное подразделение |
| ГО | Головной офис |
| ПП | Профильные подразделения |
| ФД | Финансовый департамент |

# Введение

В рамках данной работы рассмотрена проблема моделирования данных в аналитических системах, связанная с классическими принципами их построения. Объектом исследования являются системы бюджетного планирования, которые традиционно строятся на основе аналитической модели данных. Однако такая модель далеко не полностью удовлетворяет требованиям, предъявляемым к системам такого типа. Кроме аналитической обработки, системы бюджетирования должны также поддерживать транзакционную обработку. Последнее включает в себя нормализацию данных, предотвращение дублирования и несогласованности. Это необходимо для эффективного выполнения операций вставки, изменения и удаления, которые в наибольшей степени влияют на производительность системы.

Цель данной работы – применение методов моделирования для разработки новой модели данных, позволяющей эффективно обрабатывать как аналитические запросы, так и транзакционные. При этом высокие требования предъявляются к производительности, объему хранимых данных и легкости проектирования интерфейсов отчетных и входных форм.

Актуальность данной проблемы широко подтверждена как на практике, так и в теоретическом плане. До последнего момента нам не приходилось думать о совмещении в рамках одной модели OLTP и OLAP подходов, поскольку это не соответствует главной парадигме реляционных баз данных. Дело в том, что на настоящий момент эти принципы пересмотрены производителями программного обеспечения, поскольку определены другие концепции построения баз данных. Отказ от классической абстракции необходим, чтобы использовать варианты физического хранения данных для построения оптимальной модели.

Нашей задача в данной работе состоит в том, чтобы создать такую модель, максимально используя преимущества одной из таких современных платформ. Для этого в первой главе приведено описание концептуальных требований, предъявляемых к модели. Во второй главе проводится анализ существующих платформ, реализующих инновационные технологии построения баз данных. Также во второй главе описана модель и показаны ее преимущества на основе выбранной нами платформы. В третьей главе приведены интерфейсы системы, перенесенные на новую модель. Данная модель используется на практике в проекте внедрения системы бюджетирования в крупном российском банке.

# Постановка задачи

Начнем, конечно, с того, что такое система бюджетного планирования. В первую очередь система бюджетирования считается аналитической системой, или, по крайней мере, использующей аналитическую платформу. Но во вторую очередь – это инструмент сбора данных, стратегических и тактических бюджетов. Но в этом и кроется некоторая несогласованность, бытующая и сегодня в подходе к моделированию схемы данных крупных коммерческих систем бюджетного планирования.

Мы знаем, что схема данных аналитических систем строится сейчас по некоторым принципам, уже ставшим традиционными. Речь идет о концепции хранилищ данных, или OLAP, то есть о подходе к проектированию, который ставит во главу угла быстроту выполнения запросов определенного типа – таких, которые требуются в данной отчетной системе. В системах класса OLTP другие требования: модель должна работать одинаково для всех возможных типов запросов, и предотвращать дублирование данных.

Здесь нам придется коснуться такого важного термина как нормализация. Схемы транзакционных баз данных должны быть нормализованы во избежание аномалий удаления, обновления, чтения и т.д. Схемы аналитических систем являются денормализованными или многомерными – это позволяет наиболее оперативно получать информацию по запросам чтения, а конкретнее – в разрезе определенных аналитик (slicing and dicing).

Также для повышения оперативности ответа на запрос данные в многомерной схеме (иначе – схема-звезда, или куб) хранятся не только детальные, но и агрегированные – в соответствие с настроенными иерархиями, их узлами и т.д. Обычно размер хранилищ данных существенно больше, чем размер транзакционных систем.

В чем же несогласованность, когда речь идет о системах бюджетирования? Именно в том, что они должны умело совмещать свойства аналитических систем. Во-первых, они служат для построения топ-менеджментом компании аналитических бюджетных форм. Во-вторых – являются системами обработки транзакций (обновления, удаления), которыми является сбор данных на этапе формирования бюджета. Также необходим контроль объемов данных, которые обрабатывает такая смешанная система. Ведь зачастую сложно с уверенностью сказать, что такое тактический бюджет, а что такое операционный. А данные, как известно, появляются в системе только с боевым стартом, и поэтому важно на этапе проектирования установить границы «гранулярности» обрабатываемых системой данных.

## 1.1 Техническая постановка задачи

Таким образом, на этапе постановки задачи были выявлены следующие концептуальные требования к модели данных классической системы бюджетирования:

1. OLTP требования:

* Ввод данных (insert/update)

Ручной ввод данных в формы планирования необходим для создания бюджета силами линейных руководителей предприятия, а также контроля сотрудниками финансового блока. Здесь необходима возможность отслеживать изменения в реальном времени, чтобы исключить появление несогласованности в данных, а также сдвиг сроков сбора бюджета.

* Статусы блокировки срезов данных

Отслеживание данных по статусам необходимо для контроля версий данных в процессе согласования бюджета.

* Ввод комментариев по записям

Комментарии по записям необходимы для ведения дополнительной информации о планируемых затратах в ходе процесса планирования. Размер поля комментария может сильно варьироваться. Также комментарии постоянно изменяются в процессе согласования бюджета.

* Аудит данных и действий

Полная история изменений, включая изменения статусов, комментариев и данных необходима специалистам финансового блока для мониторинга процесса планирования.

* Драйверы затрат и курсы валют

Драйверы затрат необходимы для расчета нормируемых затрат. Курсы валют используются для ведения значений показателей в разрезе различных валют операций.

* Измерения в реляционном отношении

Измерения в модели планирования могут находиться в реляционном отношении, причем это случается чаще, чем в хранилищах данных. Например, определенный объект инвестиций может относиться к определенным дочерним организациям, или вид расходов – к определенной статье.

1. OLAP требования:

* Быстрые аналитические запросы по срезам данных (slice and dice)

Аналитические запросы по срезам данных необходимы для возможности анализа сводной и агрегированной информации специалистами финансового блока.

* Ведение иерархий и навигация по ним (drill-down and drill-up)

Как и в любом хранилище данных, измерения часто должны быть представлены в виде иерархии (организационная структура, план счетов)

* Расчетная логика (MDX, скрипты)

Расчетная логика необходима для реализации функций планирования (распределение затрат, централизация, вычисление амортизации, нормируемых затрат)

* Сводный многомерный анализ (pivoting)

Возможность построения сводных таблиц также относится к требованиям анализа агрегированной информации на всех имеющихся на текущий момент в бюджете данных.

## 1.2 Постановка задачи с точки зрения бизнеса

В данной части работы будет приведена часть концептуального плана проекта по внедрению системы бюджетного планирования. Практический пример позволит нам понять, действительно ли требуют решения, озвученные нами выше задачи.

**Организационная структура**

Бюджетная структура банка представляет собой иерархическое дерево, «листьями» которого являются минимальные организационные единицы, на которые планируются расходы – ССП. К ССП относятся:

* ССП головного офиса;
* дополнительные офисы Москвы и Московской области;
* головные офисы (ГО) филиалов (региональные офисы - операционные офисы 1-го уровня);
* дополнительные офисы филиалов (операционные офисы 2-го уровня);
* международные представительства;
* дочерние компании.

**Нормативно-справочная информация**

Для целей планирования сметы расходов и доходов в Системе планирования реализованы следующие справочники:

* справочник ССП;
* справочник статей;
* справочник видов расходов;
* справочник курсов валют;
* справочник драйверов;

**Подход к реализации процесса планирования**

Процесс планирования Сметы АХР состоит из двух групп бизнес-процессов:

* БП01.Годовое планирование АХР
* БП02.Полугодовая коррекция бюджета АХР

Ниже представлен весь перечень процессов и подпроцессов планирования сметы АХР.

1. БП01. Годовое планирование сметы АХР
   1. БП01.01.Подготовка к планированию АХР
      1. БП01.01.01. Обновление справочников
      2. БП01.01.02. Настройка алгоритмов расчета
      3. БП01.01.03. Обновление прав доступа
      4. БП01.01.04. Загрузка графиков платежей из системы ИСУ АХД
   2. БП01.01.05. Загрузка фактических значений драйверов и норм
   3. БП01.02. Сбор данных ССП
      1. БП01.02.01. Сбор заявок по расходам от ССП ГО
      2. БП01.02.02. Сбор заявок по расходам от подразделений Московского региона
      3. БП01.02.03. Сбор заявок по расходам от региональных подразделений
      4. БП01.02.04. Сбор заявок по расходам от региональных подразделений (Филиалы)
   4. БП01.03. Контроль и согласование ПП
   5. БП01.04. Контроль Куратором
   6. БП01.05. Ревизия ПП
   7. БП01.06. Согласование ФД
   8. БП01.07. Внесение изменений в бюджет
   9. БП01.08. Агрегация данных и передача в смету АХР. Факт и внешние системы
2. БП02. Полугодовая коррекция сметы АХР
   1. БП02.01.Подготовка к полугодовой коррекции АХР
      1. БП02.01.01. Создание версии плана для полугодовой коррекции
      2. БП02.01.02. Обновление справочников
      3. БП02.01.03. Настройка алгоритмов расчета
      4. БП02.01.04. Обновление прав доступа
      5. БП02.01.05. Загрузка плана из системы ИСУ АХД.
      6. БП02.01.06. Загрузка графиков платежей из системы ИСУ АХД (с обратным знаком)
3. БП02.02. Ввод корректировок Профильными Подразделениями
4. БП02.03. Согласование ФД
5. БП02.04. Внесение изменений в бюджет
6. БП02.05. Агрегация данных и передача в смету АХР. Факт и внешние системы

Процесс планирования реализован с максимальной унификацией всех входящих в него вспомогательных процессов (подпроцессов).

Под вспомогательными подпроцессами понимаются процессы подготовки к планированию - загрузка данных из внешних источников, процесс выгрузки агрегированных плановых данных во внешние файлы для последующей загрузки во внешние системы, процесс настройки/обновления полномочий пользователей.

Основными процессами являются: сбор данных, проверка профильными подразделениями, проверка кураторами, согласование ФД, внесение изменений, утверждение бюджета.

Каждый основной процесс состоит из одного или нескольких шагов. По каждому шагу назначается собственник. Собственник (инициатор) шага закрепляется за каждым ССП.

В системе реализованы следующие процессы (подпроцессы) и шаги:

1. Подготовка планирования: загрузка данных из системы «ИСУ АХД» (графики платежей, справочники), загрузка данных из внешних файлов (справочники, реестры), настройка полномочий пользователей.
2. Сбор данных.
3. Контроль ПП.
4. Согласование ФД.

**Ручной ввод плановых затрат**

При помощи ручного ввода будут планироваться все ненормируемые расходы. Планирование вручную должно осуществляться следующим образом: пользователь определяет все аналитики планирования – ССП, статью, вид расхода, объект и проект. Далее пользователь задает распределение плановой суммы кассовым методом. Также пользователь задает даты «с» и «по» для распределения плановой суммы методом начисления.

**Документооборот и статусы**

Статусы данных реализованы для ограничения возможности ввода на разных этапах планирования. Статусы данных соответствуют этапам процесса планирования. Статусы данных устанавливаются в разрезе следующих аналитик: «ССП», «Статья», «Время», «Категория» (план, факт, и т. п.).

В системе реализованы следующие статусы:

* «Ввод данных» - значение по умолчанию;
* «Контроль ПП» – проставляется на этапе контроля ПП;
* «Контроль ФД» – проставляется на записи, на этапе контроля ФД;
* статус «Согласовано» является конечным (никакие изменения данных невозможны).

В случае, когда требуется полностью исключить из планирования какую-либо запись, то ее сумма обнуляется с помощью механизма отклонения записей. При этом отклоняемые данные копируются в специальную категорию Rejected, инициатору ССП, к которому относятся отклоняемые данные, отправляется уведомление по электронной почте, в поле «Автор изменения» устанавливается ключевое слово «Отклонено» и указывается автор отклонения записи.

**Описание интерфейсов ввода данных**

В системе планирования сметы доходов и расходов реализовано три группы интерфейсов: интерфейсы ввода данных для расчета планируемых сумм по нормам, интерфейсы расчета плановой амортизации по кассе и начислению и интерфейсы ручного ввода данных.

В системе планирования интерфейс ручного ввода подразделяются на следующие категории:

* интерфейс просмотра данных;
* интерфейс ручного ввода плановых сумм;
* интерфейс ручного ввода;
* интерфейс ввода драйверов;
* интерфейс расчета плановой амортизации;

**Планирование на основе нормируемых затрат**

Расчет расходов по нормируемым статьям осуществляется автоматически путём умножения установленных норм на количество драйвера.

Драйверы привязаны к ССП, а нормы устанавливаются в разрезе ССП, статьи расхода, вида расхода, драйвера. Для реализации планирования по нормам и драйверам используется ввод норм в разрезе узлов ССП или ССП, статей, видов расходов, драйверов и периодов планирования через загрузку предварительно настроенного файла.

Значения драйверов (фактические за прошедшие периоды) могут загружаться из внешних файлов, а также заноситься вручную через интерфейс ввода значений драйверов.

Расчет сумм планирования по нормам и драйверам осуществляется автоматически при сохранении значений норм или драйверов.

**Расчет амортизации**

Фактическая амортизация рассчитывается внесистемно и загружается в систему планирования из внешнего файла. Плановая амортизация рассчитывается следующим образом: в качестве базы используются суммы, запланированные по статьям капитальных затрат и определенных видов расходов. Амортизация рассчитывается не по каждой записи, а по всем записям, агрегированным по аналитикам ССП-статья-вид расхода. Алгоритм расчета амортизации следующий: сумма по ССП-статье-виду расхода делится на соответствующую статье-виду расхода норму амортизации и распределяется по всем периодам планирования до конца года, начиная с периода, следующего за периодом, в котором запланирована сумма по статье капитальных затрат.

**Планирование общебанковских расходов**

Требования к планированию общебанковских расходов (ОБР) заключаются в следующем.

В случае централизации расходов на ОБР: ССП осуществляет планирование статьи кассовым методом и методом начисления. Если по статье-ССП-виду расхода нужно осуществить централизацию расходов по кассе, то с ССП расходы, запланированные по кассе, переносятся на ОБР профильного подразделения. Расходы по начислению остаются на ССП. Если по статье-ССП-виду расхода нужно осуществить централизацию расходов по начислению, то расходы, запланированные по начислению, переносятся на ОБР, расходы, запланированные по кассовому методу, остаются на ССП. При децентрализации ОБР: на ОБР расходы планируются по кассе и начислению, далее определяется база распределения по ССП – выбираются ССП, по которым происходит распределение, в качестве базы распределения используется ранее определенный драйвер.

**Отчеты по комментариям**

Отчет по комментариям позволяет просматривать все комментарии, относящиеся к текущему приложению (в текущем ракурсе).

Возможно, применение фильтров по времени, пользователю, приоритету и отдельным измерениям, для которых сохранен комментарий.

**Отчет по аудиту данных**

В отчете показано, кем изменены данные, в какое время, как (например, с помощью логики или интерфейса для Excel), а также подробные сведения об измененной записи. В отчете отражается новое введенное значение.

**Отчет по рабочим статусам**

В этом отчете можно задать параметры, включающие даты и время начала и окончания, а также значения элементов измерений, отслеживающих рабочий статус, а именно: статьи бюджета, ССП, категории.

В отчете о рабочем статусе выводится статус данных, являющийся текущим рабочим статусом для этих данных. Здесь указан текущий этап данных в общем процессе утверждения.

## 1.3 Текущее решение на платформе SAP BPC on SAP BW

Изначально, модель данных классической системы бюджетирования будет опираться на технологическую платформу SAP BW [2]. SAP BW представляет собой решение SAP [2] для организации корпоративного хранилища данных, позволяющего использовать при построении и моделировании системы корпоративной отчетности следующие преимущества:

1. OLAP, MDX Engine – возможности многомерного, slice-n-dice анализа
2. ETL процессы [1] – использование потоков трансформации данных, областей временного хранения подготовки данных (Persistent Staging Area), отслеживание загрузки только измененных записей или «дельты» (Change-Data Capture)
3. Широкие возможности моделирования и управления производительностью системы на уровне аналитических кубов (разбиение на секции, компрессия, индексы, предопределенные агрегаты для выбранных измерений схемы данных)

В соответствии с текущими «лучшими практиками», модель системы бюджетирования представляет собой OLAP-куб, или многомерную таблицу факта, находящуюся в отношении с измерениями, также известную как Star Schema Benchmark [3] (Рис.1):

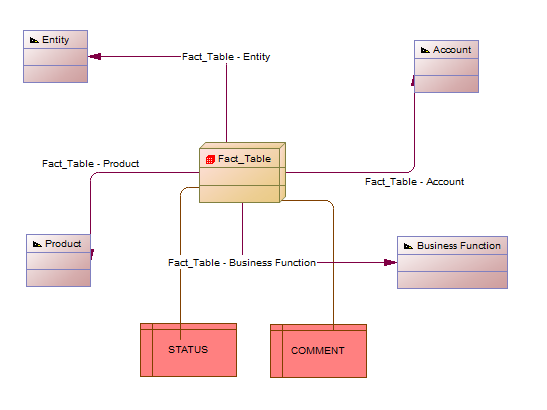


Рис.1 Схема-звезда

Для ведения данных планирования, создания бизнес-процессов согласования бюджета, аудита данных, рабочих статусов и комментариев используется специальное приложение SAP Business Planning and Consolidation [4]. Данное приложение предоставляет возможности работы с построенными моделями данных через веб-интерфейс и клиент Microsoft Excel. Концептуальные требования, описанные выше при постановке задачи, полностью реализованы в данном программном продукте.

Однако, вследствие того, что модель данных, основанная на схеме-звезде или таблице фактов, не является оптимальной для выполнения операций записи и обновления, при достаточно высокой гранулярности данных производительность системы резко падает. Данный факт объясняется, например, наличием частых операций соединения (JOIN) между «денормализованной» и как следствие, избыточной таблицей фактов и таблицами статусов и комментариев. Таблица фактов находится в отношении один ко многим с этими таблицами, поскольку для каждой ее записи, в целях вышеупомянутого аудита данных, необходимо хранить все версии статусов и комментариев, измененных пользователем.

## 1.4 Описание существующей проблемы

Итак, требования по автоматизации бюджетного процесса, собранные в концептуальном проекте внедрения системы бюджетирования в крупном банке, подтверждают описанные в технической постановке задачи требования к модели данных. С одной стороны, данные должны быть оптимизированы для ввода записей, статусов, комментариев, что обычно реализуется с помощью высоко нормализованной схемы таблиц (OLTP). С другой стороны, в рамках решения аналитических задач, необходимо оптимизировать данную модель для выполнения быстрых аналитических запросов (OLAP). Модель системы бюджетирования должна удовлетворять обоим принципам одновременно, что несовместимо с классической концепцией реляционных баз данных. Далее мы рассмотрим предлагаемое в рамках данной работы решение, использующее последние технологии построения баз данных.

# Предлагаемое решение на основе платформы SAP HANA

## 2.1 Анализ существующих решений

Технологии баз данных в настоящий момент переживают в некотором смысле переломный момент. Сейчас классические подходы пересматриваются вплоть до физического уровня хранения и оперирования данными. Общераспространенная теория реляционных баз данных больше не является «панацеей», и чтобы, преодолеть ограничения, накладываемые данной абстракцией, приходится спускаться до низких уровней, вплоть до организации «железа» и физического хранения записей.

Это подразумевает под собой определенный вызов для ИТ-архитекторов и профессионалов по моделированию, поскольку каждый конкретный бизнес-сценарий может требовать индивидуального подхода на всех этапах проектирования системы.

Последними инновациями организации хранения и доступа к данным в аналитических системах являются:

1. СУБД на графических процессорах [5];

Высокая параллелизация запросов за счет большего количества однотипных процессоров, управление распределением памяти, многопоточность. Примером реализации данной технологии является СУБД empulse ParStream.

1. In-Memory базы данных;

Высокая скорость чтения, возможности кэширования и параллелизации за счет хранения всех данных не на диске, а в основной памяти. Примером реализации данной технологии являются решения SAP HANA [6], QlikView In-Memory.

1. Поколоночные базы данных [12];

Возможность выполнения быстрых операций группировки, агрегации, компрессии за счет того, что хранение данных организовано не построчно, а поколоночно в виде массива значений. Примером реализации данной технологии являются решения SAP HANA, MongoDB, Apache Cassandra [7].

1. Алгоритмические подходы для параллельного выполнения запросов на низком уровне

* Map-Reduce [8] (примеры – Sybase IQ, Apache Hadoop,   
   Tenzing)
* Parallel Aggregation (пример – SAP HANA) [9, 10]

1. Распределенные, или согласованные в «конечном счете» системы systems [11];

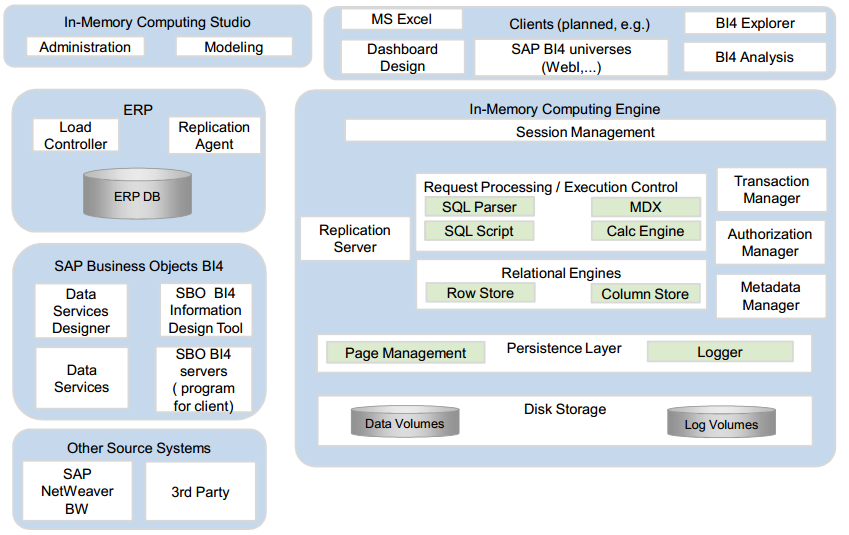
Обеспечивают высокую доступность за счет хранения нескольких копий всего объема данных – операции чтения могут выполняться с любой копией, а операции вставки и обновления выполняются последовательно на всех копиях в течение определенного периода времени. Примером реализации данной технологии являются решения Apache Cassandra, Amazon Dynamo.

В данной работе будет описан пример реализации модели данных бюджетного планирования на технологической платформе SAP HANA. Этот продукт сочетает в себе такие инновационные технологии, как поколоночное хранение, in-memory хранение данных и высокий коэффициент параллельного выполнения запросов на уровне процессоров.

SAP HANA поставляется как «appliance», то есть в виде программно-аппаратного комплекса (сервера с предустановленным на нем программным обеспечением).

Программные компоненты решения оптимизированы для работы на специальном сертифицированном оборудовании, поставляемом ведущими производителями (HP, IBM, Fujitsu). Требования к оборудованию включают в себя требования к оперативной памяти (main data), SSD памяти (non-active data) и дисковой памяти, как средству резервного копирования. Также определена спецификация процессоров, использования ими кэшей на нескольких уровнях.

Архитектура программной части комплекса выглядит следующим образом (Рис.2):

Рис.2 Архитектура SAP HANA

В поставку входят также клиентские приложения для построения системы корпоративной отчетности. В правой части рисунка изображено устройство главной вычислительной части, содержащей несколько оптимизирующих запросы механизмов (SQL Script, Calc Engine, MDX), а также два основных реляционных механизма – построчное и поколоночное хранилище.   
В нижней части изображено дисковое хранилище, однако данные в нем хранятся исключительно для резервного копирования. Вся работа с данными на уровне приложения происходит в основной памяти.  
Таким образом, SAP HANA обладает следующими преимуществами:

1. Технология In-Memory
2. Поколоночное хранение
3. Вычисление агрегатных данных без материализации
4. Алгоритмизированное параллельное выполнение запросов (parallel aggregation)

## 2.2 Модель данных системы планирования

В данной работе будет изучено построение модели данных, комбинирующей реализацию OLTP и OLAP требований в рамках одной системы. Поскольку для классической теории реляционных баз данных объединение двух данных подходов теоретически неоправданно, мы будем широко использовать технологические и аппаратные преимущества выбранной платформы.

В общем виде подход к построению модели заключается в следующем:

1. Ввод данных в процессе планирования бюджета будет осуществляться в высоко нормализованные таблицы с построчным хранением
2. Аналитические запросы к этим данным будут выполняться к таблице с поколоночным хранением.

Модель, реализующая второй пункт, будет построена по принципу Google Big Table [13] – таблица фактов без измерений. Добавление атрибутов измерений становится возможным благодаря свойству поколоночных таблиц – в них можно добавлять столбцы, не затрагивая текущие данные. Отсутствие операций соединения в модели с поколоночным хранением благоприятным образом влияет на производительность аналитических запросов.

Начнем с первого пункта, то есть с модели данных, предназначенной для ввода данных в систему.

**Модель данных для ввода и изменения**

Модель данных, предназначенная для ввода и изменения записей в процессе планирования, построена на основе нормализованных таблиц с построчным хранением и соответствует третьей нормальной форме (3NF). Данный подход обеспечивает защиту от дублирования данных, появления несогласованных данных, но самое главное – позволяет работать с данными системы планирования быстро и эффективно в режиме транзакций изменения, добавления и удаления. Также в модели будут использованы технологические преимущества выбранной платформы. Именно с этой целью аналогичные таблицы для ручного ввода, амортизации и нормируемых затрат разделены на схеме. Мы коснемся позднее применения данного подхода.

Инфологическая модель такой базы данных выглядит следующим образом (Рис.3):

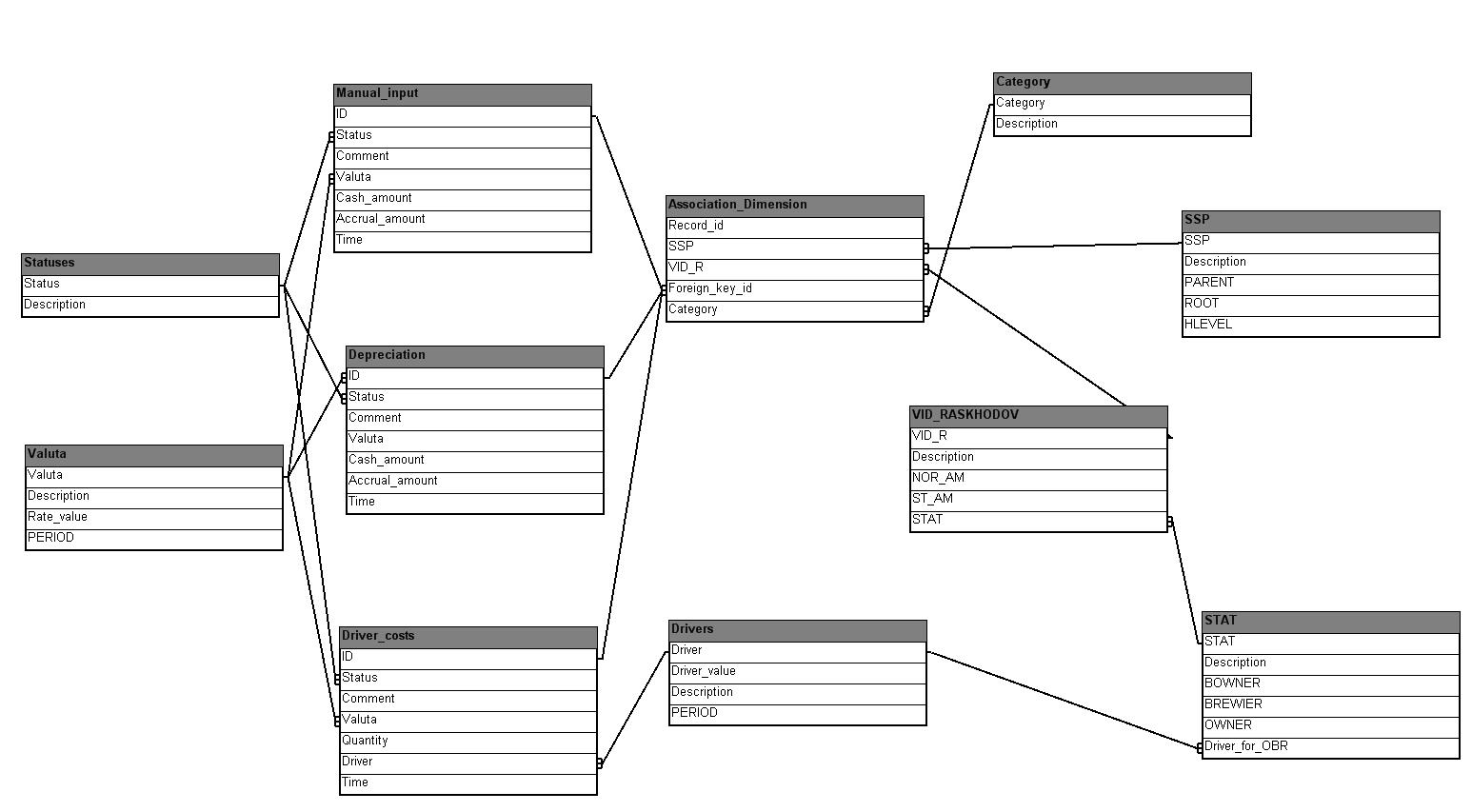


Рис.3 Модель для ввода/изменения

Главной таблицей в этой модели является таблица Association\_Dimension. Она содержит следующие поля: уникальный номер записи (Record\_id), ССП (SSP), вид расходов (VID\_R), категория (Category). Также в таблице присутствует внешний суррогатный ключ к транзакционным таблицам, который также входит в первичный ключ таблицы Association\_Dimension. Это необходимо, для того чтобы исключить совпадения в значениях ключа между транзакционными таблицами.

Также на схеме изображены три аналогичные таблицы, связанные «один-ко-одному» с таблицей Association\_Dimension: Manual\_input (Ручной ввод), Depreciation (Амортизация) и Driver\_costs (Нормируемые затраты). Данное решение пояснено в описании связей схемы данных и пункте 2.3, т.к. касается физических основ реализации аппаратной платформы. Следующие поля в этих таблицах совпадают:

* уникальный первичный ключ (ID)
* статус (Status)
* комментарий (Comment)
* валюта (Valuta)
* период (Time)
* Record\_id – внешний ключ (Association\_Dimension)

Таблицы Manual\_input и Depreciation также имеют аналогичные показатели: сумма по кассовому методу (Cash\_amount) и сумма по методу начисления (Accrual\_amount). Таблица нормируемых затрат Driver\_costs содержит показатель количество (Quantity) и ссылку на справочник норм драйвера (Driver). Также все три транзакционные таблицы имеют ссылки на справочник статусов и валют (Statuses и Valuta).

**Описание справочников модели:**

1. Statuses – содержит уникальный первичный ключ (Status) и название статуса (Description)
2. Valuta - содержит уникальный первичный ключ (Valuta) и название валюты (Description), значение курса валюты (Rate\_value) на определенный период времени (PERIOD).
3. Drivers - содержит уникальный первичный ключ (Driver) и название драйвера затрат (Description), значение нормы затрат (Driver\_value) на определенный период времени (PERIOD).
4. Category - содержит уникальный первичный ключ (Category) и название категории (План, Факт) (Description).
5. VID\_RASKHODOV – содержит уникальный первичный ключ (VID\_R) и название вида расходов (Description), атрибут NOR\_AM, норма амортизации (иначе, срок полезного использования ОС), используется для расчета амортизации, атрибут STAT – хранит связь «один ко многим» между Статьями и Видами Расхода, атрибут ST\_AM – хранит информацию о амортизационной статье, на которую распределяются списания стоимости с текущей статьи.
6. STAT – содержит уникальный первичный ключ (STAT) и название статьи (Description), атрибут BOWNER, используется для настройки видимости и прав открытия шагов потока бизнес-процессов, атрибут BREVIEWER – утверждающий в потоках бизнес-процесса, атрибут OWNER – используется разрешения на редактирование для статусов, Driver\_for\_OBR – используется для распределения общебанковских расходов при расчете децентрализации.
7. SSP – содержит уникальный первичный ключ (SSP) и название ССП (Description), а также Parent-Child иерархию на основе полей: ROOT – корневой элемент иерархии, PARENT – родитель в текущей иерархии, HLEVEL – уровень в текущей иерархии.

**Описание связей отношений в схеме данных:**

1. Таблицы измерений STAT, VID\_R, Category связаны 1:М с таблицей Association\_Dimension.
2. Таблицы транзакционных данных Manual\_input, Depreciation и Driver\_costs связаны М:1 с таблицей Association\_Dimension. Это связано с тем, что для определенного ССП и вида расхода заранее определен метод планирования: ручной, амортизация (для капитальных затрат) или нормируемые затраты. Для каждой комбинации в таблице Association\_Dimension существует несколько соответствий в одной из трех таблиц, отличающихся только периодом.
3. Таблицы Statuses, Valuta связаны 1:М с транзакционными таблицами Manual\_input, Depreciation и Driver\_costs.
4. Таблица-справочник нормируемых затрат Driver связана 1:М с таблицами STAT и Driver\_costs.
5. Таблица измерений VID\_RASKHODOV связана М:1 с таблицей STAT. Так как одной статье соответствует несколько видов расходов (например, статья – Канцелярские расходы, виды расходов – карандаши, ручки, скрепки).

**Описание реализации заявленных требований в модели:**

1. Ведение организационной структуры реализовано в модели с помощью Parent-Child иерархии измерения SSP (дополнительные офисы, профильные подразделения)
2. В модели реализованы справочники ССП, статей, видов расходов, категорий, драйверов и курсов валют
3. В модели реализованы справочники Статусов, ведение Комментариев, однако история изменения Статусов и Комментариев будет храниться в таблице с хранением по колонкам, описанной во второй части данного пункта. При изменении статуса или комментария в таблицах транзакционных данных модели происходит изменение (update) записи.
4. В модели реализована таблица для ведения данных по ручному вводу затрат (Manual\_input), она имеет показатели суммы по начислению и по кассовому методу.
5. В модели реализована таблица для ведения нормируемых затрат (Driver\_costs), она содержит значения количества драйвера и ссылку на справочник драйверов – для вычисления суммы затрат как произведения количества драйвера и его значения на определенный период.
6. В модели реализована таблица для ведения амортизации (Depreciation), она также имеет показатели суммы по начислению и по кассовому методу. Для каждой комбинации ССП – статья – вид расхода модель позволяет однозначно определить норму амортизации (NOR\_AM, атрибут измерения STAT), необходимую для распределения амортизации по периодам.
7. В модели реализована структура данных для расчета общебанковских расходов (ОБР). Для каждой комбинации ССП – статья – вид расхода модель позволяет однозначно определить драйвер распределения расходов (Driver\_for\_OBR, атрибут STAT), необходимый для распределения общебанковских расходов по ССП.

**Модель данных для аналитических запросов**

Модель данных для аналитических запросов представлена в виде одной таблицы, многомерной и не соответствующей принципам нормализации. Модель построена по принципу построения поколоночных баз данных, использованному в Google Big Table. Модель состоит из таблицы фактов без измерений и содержит естественные значения ключей. В поколоночных таблицах операция соединения является самой тяжелой с точки зрения производительности запросов. Таким образом, используя свойство поколоночных таблиц – возможность добавлять новые столбцы, не затрагивая текущие данные, мы можем вести аналитику по всем необходимым атрибутам измерений, описаниям, а не только по значениям естественных ключей. Кроме того, поколоночные таблицы используют при хранении механизм компрессии, помещающий уникальные значения столбца в «словарь значений» (value dictionary), и присваивающий этим значениям определенный коды, которые и играют на внутреннем уровне роль суррогатных ключей, или SID-ов. Также, определив фиксированную длину для каждого из атрибутов таблицы фактов, мы можем еще больше увеличить производительность запросов, позволив приложению обращаться к данным столбца как к массиву значений произвольного доступа.

Инфологическая модель таблицы фактов выглядит следующим образом (Рис.4):

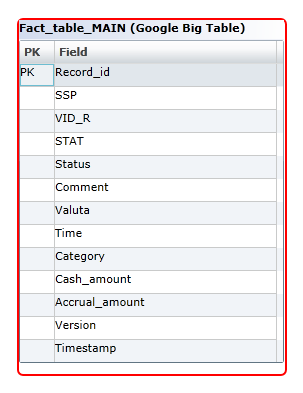


Рис.4 Модель для чтения

Таблица фактов содержит следующие поля, или атрибуты:

1. Record\_id – уникальный первичный ключ, однозначно идентифицирующий запись (INT)
2. SSP – название самостоятельного структурного подразделения (VARCHAR 40)
3. VID\_R – название вида расхода (VARCHAR 40)
4. STAT – название статьи затрат (VARCHAR 40)
5. Status – статус записи (VARCHAR 10)
6. Comment – комментарий по записи (VARCHAR 60)
7. Valuta - валюта операции (VARCHAR 3)
8. Time – период (DATETIME)
9. Category – категория данных (VARCHAR 10)
10. Cash\_amount (DECIMAL 10,2) – сумма по методу начисления
11. Accrual\_amount (DECIMAL 10,2) – сумма по кассовому методу
12. Version (BIT) – версия данных, указатель на актуальную (последнюю) версию записи. В отчетности по бюджету мы будем видеть только актуальные версии записей, остальные сохраняются для ведения историчности изменений комментариев и статусов, показателей.
13. Timestamp – время добавления записи, необходим для ведения историчности изменений комментариев, статусов и показателей (TIMESTAMP).

**Интеграция модели данных для ввода-изменения и модели для аналитических запросов**

Интеграция модели данных, представляющей собой схему данных соответствующей третьей нормальной форме, и многомерной таблицы фактов будет реализована с помощью репликации данных на основе триггерных конструкций (SQL Data Definition Language).

Пример триггера, основанного на данных схемах, достаточно прост, и выглядит следующим образом:

CREATE TRIGGER TEST\_TRIGGER

AFTER INSERT/UPDATE ON Manual\_input

REFERENCING NEW ROW mynewrow, OLD ROW myoldrow

FOR EACH ROW

BEGIN

SELECT SSP,

VID\_R,

STAT,

Status,

Comment,

Time,

Category,

Cash\_amount,

Accrual\_amount

INTO TEMP FROM Manual\_input

INNER JOIN Association\_Dimension ON Manual\_input.record\_id = Association\_Dimension.record\_id

INNER JOIN VID\_RASKHODOV ON VID\_RASKHODOV.VID\_R = Association\_Dimension.VID\_R

INNER JOIN Category ON Association\_Dimension.Category = Category.Category

INNER JOIN SSP ON Association\_Dimension.SSP = SSP.SSP

INNER JOIN STAT ON VID\_R.STAT = STAT.STAT

INNER JOIN Statuses ON Manual\_input.Status = Statuses.Status;

INNER JOIN Valuta ON Manual\_input.Valuta = Valuta.Valuta

WHERE

Manual\_input.ID= :mynewrow.Record\_id

INSERT INTO Fact\_table\_main VALUES(:TEMP, 1,CURRENT\_TIMESTAMP);

UPDATE Fact\_table\_main SET Version = 0

Record\_id = (SELECT MAX(Record\_id) FROM Fact\_table\_main

WHERE SSP = TEMP.SSP AND VID\_R = TEMP.VID\_R AND STAT = TEMP.STAT AND   
 Category = TEMP.Category)

END;

Сначала производится операция выборки, соединяющая необходимые таблицы для получения всех полей только что вставленной записи. Затем данная запись вставляется в таблицу фактов как актуальная, а предыдущая запись с таким же набором измерений помечается как неактуальная.

Таким образом, при изменении значения суммы, статуса или комментария в таблицу фактов будет происходить добавление актуальной записи, и пометка старой записи неактуальной. Это позволяет, как быстро обращаться к актуальным данным бюджета, так и хранить историю изменений всех данных.

## 2.3 Преимущества модели данных на основе SAP HANA

В данной работе мы уже упоминали о том, что в настоящее время преодолевать ограничения классических реляционных баз данных, используя более низкоуровневые абстракции. Провозглашение того, что исключительно в рамках модели мы можем получить оптимальную производительность системы, сейчас не совсем оправданно. Именно на стыке технологий, относящихся к оптимизации «железа», и специального программного обеспечения, модели данных, учитывающей эту специфику, мы можем получить оптимальный результат. В этой главе мы хотели бы показать преимущества, которые дает нам именно наша модель данных, построенная на основе базы данных SAP HANA.

**Механизмы сжатия и «словарь значений»**

Таблица фактов, используемая в нашей модели, содержит естественные значения ключей, а также прочих необходимых для аналитики атрибутов, т.к. для повышения производительности мы исключили из модели таблицы измерений. Однако тогда большая часть полей является текстовыми и это также может повлиять на скорость работы, но в противоположную сторону. Здесь нам на помощь приходят внутренние механизмы компрессии данных, используемые в SAP HANA [15].

Внутренний механизм компрессии SAP HANA изображен на рисунке (Рис.5):

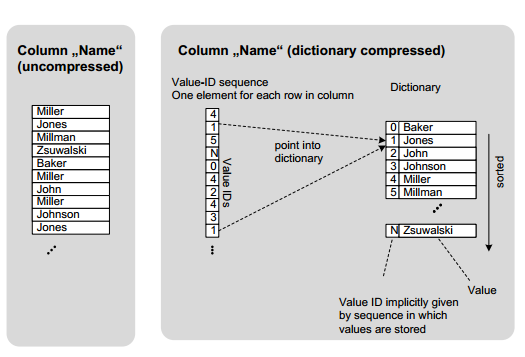
****

Рис.5 Алгоритм компрессии данных

В левой части изображен несжатый столбец значений фамилий сотрудников. В правой части показано, как устроена компрессия и физическое хранение таких данных. Сначала уникальные значения столбца сортируются по возрастанию, и затем каждому из них присваивается порядковый номер. Это хранится в оперативной памяти системы и называется «словарь значений». Сам столбец содержит ссылки на порядковые номера из словаря значений. Это позволяет использовать быстрый бинарный поиск при обработке запросов, применяя бинарное кодирование с помощью log2(NDICT) бит.  
Поиск осуществляется следующим образом: мы находим значение в словаре, затем просматриваем значения столбца, используя индекс. Индекс представлен в виде массива значений строк для каждого значения из словаря.  
Кроме того, мы получаем уменьшение размеров таблицы и базы данных соответственно за счет сжатия с использованием одного из следующих методов кодирования (префиксное кодирование, кодирование длин серий, кластерное кодирование, косвенное кодирование).

Таким образом, благодаря методам хранения данных, используемым в SAP HANA, для нас не имеет значения, какой тип имеет столбец, т.к. наличие порядковых номеров из словаря значений дает нам возможность осуществлять быстрый поиск по данным числового типа.

**Секционирование таблиц и параллельная агрегация**

Здесь мы объясним, для чего мы разделили транзакционные таблицы в первой модели на три – Manual\_input, Depreciation, Driver\_costs и почему установили связь между ними и таблицей Association\_Dimension как 1:1. Дело в том, что измерения ССП и вид расходов содержат наибольшее количество значений членов измерений (организационная структура и детализированный управленческий план счетов). Однако в таблице Association\_Dimension содержатся лишь внесенные комбинации, но лишь по одному разу. Это происходит потому, что пересечение комбинаций невозможно – для каждого ССП и вида расходов предопределен метод планирования (ручной, капитальные затраты или по нормам). Это позволяет максимально уменьшить количество записей, храня каждую комбинацию лишь по одному разу. Таким образом, мы можем разбить таблицу Association\_Dimension на секции по двум измерениям: ССП и вид расходов, которые позволяют нам обрабатывать данные из различных частей с помощью различных ядер процессора и даже различных серверов. Устройство данного механизма в SAP HANA изображено на рисунке (Рис.6) (разбиение на секции возможно как по строкам, так и по столбцам):

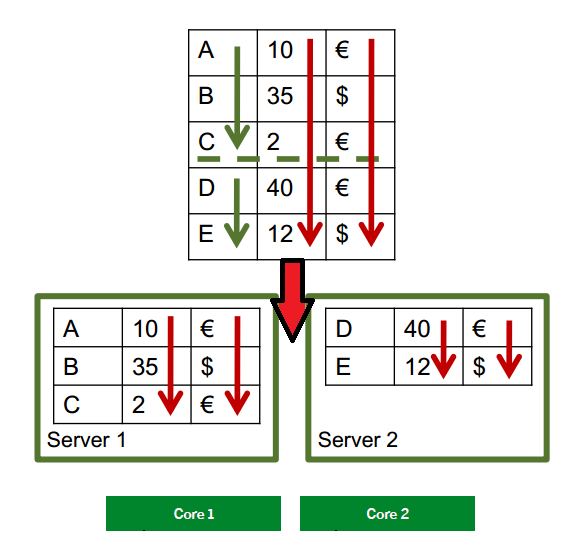
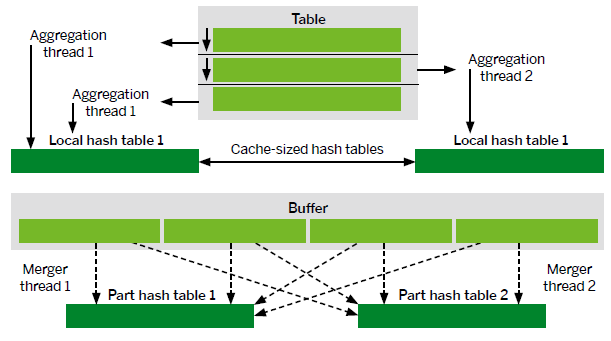
****

Рис.6 Технология параллельной обработки секций

В SAP HANA предусмотрены следующие алгоритмы секционирования: циклическое, хэш-секционирование и секционирование по диапазонам. Так как значения измерений ССП и вид расходов нам заранее известны и будут встречаться равновероятно, для них следует применить секционирование по диапазонам. Также важный для секционирования атрибут – Время (Time), для него целесообразнее применить хэш-секционирование (нет информации о вероятности появления значений) [15].

Таким образом, за счет данных возможностей SAP HANA мы сможем достичь наибольшего параллелизма на аппаратном уровне. Это относится ко всем операциям – чтения, изменения. В данном случае мы получаем практически горизонтальное масштабирование производительности системы.

Секционирование также может помочь нам в модели таблицы фактов. Таблица фактов является многомерной и будет содержать большое количество столбцов. Кроме того, так как мы храним в ней все исторические данные по изменениям данных, таблица будет также большой горизонтально (обладать высокой кардинальностью). Для ускорения выполнения аналитических запросов мы можем использовать аналогичное секционирование по атрибутам ССП, вид расхода, статья (диапазонное) и время (хэш), т.к. эти атрибуты содержат наибольшее количество значений. Также необходимо настроить диапазонное секционирование по атрибуту версия, т.к. в аналитических отчетах нам будут нужны только актуальные данные, которые составляют лишь часть всей таблицы фактов. Следовательно, мы сможем добиться большего параллелизма. Но, кроме этого, разбиение на секции поможет нам применить механизм параллельной агрегации данных, реализованный в SAP HANA. Быстрая агрегация чрезвычайно эффективна для аналитических запросов по срезам данных.  
Механизм параллельной агрегации схематично изображен на рисунке (Рис.7):

****Рис.7 Технология параллельной агрегации

Параллельная агрегация работает следующим образом [15]:

1. Выбираются секции входящего набора записей
2. Секции помещаются в хэш-таблицы для обеспечения О(1) поиска
3. Выполняется агрегация данных для данных секций в параллельных потоках и с использованием различных ядер процессора
4. Затем агрегированные данные собираются в параллельных потоках, используя алгоритмы параллельного соединения (hash, range join, radix join, map-reduce, mergesort)

Таким образом, мы можем добиться горизонтальной масштабируемости системы также для модели таблицы фактов, эффективно используя параллельную агрегацию и секционирование. Это позволяет избежать проблем с производительностью аналитических запросов на достаточно объемной таблице фактов, хранящей исторические данные.

**Обработка OLTP информации таблицами с поколоночным хранением**

Таблицы с поколоночным хранением, используемые нами для моделирования таблицы фактов, используются и в других базах данных. В первую очередь они помогают оптимизировать операции чтения, когда нет необходимости читать для каждой записи значения всех столбцов, а можно прочитать из памяти лишь значения тех столбцов, которые участвуют в запросе. Однако SAP HANA также поддерживает эффективное выполнение операций добавления для поколоночных таблиц, или обработку OLTP-транзакций [14]. Это важно для нас, так как в любом случае мы создаем такую нагрузку на таблицу фактов, используя репликацию данных на триггерах. Кроме того, данные, поступающие для добавления, должны быть добавлены в «словарь значений», т.е. быть сжаты. Для этого в SAP HANA разработана концепция промежуточного хранения данных при вставке в таблицы с поколоночным хранением. Эта концепция состоит из трех стадий обработки записи. Данный жизненный цикл записи при добавлении изображен на рисунке (Рис.8):

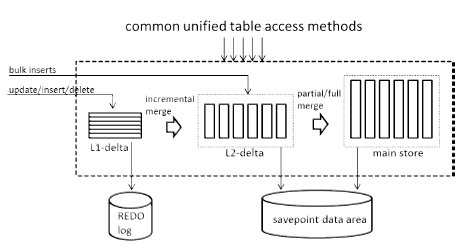


Рис.8 Алгоритм обработки запросов к таблицам

1. L1-delta: принимает на вход все запросы к таблице, сохраняя их в строчно-ориентированном формате, что более оптимально для ввода, удаления и изменения. Также здесь данные не сжимаются, оптимальный размер для запросов – 10,000-100,000 строк.
2. L2-delta: во второй фазе записи уже хранятся в поколоночном формате, а также кодируются с использованием словаря значения. Но он остается не отсортированным, чтобы не снижать производительность OLTP-запросов по вторичным индексам, например, при проверке уникальности определенного значения. Оптимальный размер запросов – до 10 млн строк.
3. Main store: здесь данные уже сжаты используя возможные техники кодирования и битовых векторов, а также отсортированы, т.е. хранятся в обычном, доступном для быстрого чтения формате.

Для нас здесь важно то, что первая и вторая фаза оптимизированы для операций добавления, изменения и удаления, как единичного, так и пакетного. Оптимизация алгоритмов соединения данных, хранящихся в каждой из трех фаз, более подробно описана в [15]. Главное, что поколоночная таблица будет легко справляться с операциями ввода, изменения и удаления, а при временном снижении нагрузки на систему будет добавлять данные из первых двух фаз в основной индекс. Таким образом, нам не следует опасаться падения производительности в результате наличия операций вставки в таблицы такого типа, хранящие сжатые данные.

# Результаты

В качестве описания результата проделанной работы, приведем описание интерфейсов и форм, доступных пользователю после реализации указанной модели. Данные интерфейсы взяты из продуктивной системы для демонстрации процесса работы пользователя с интерфейсами ввода данных в ходе процесса бюджетного планирования.

## 3.1 Форма «Нормы-драйверы»

**Описание формы «Нормы-Драйверы»**

Форма «Нормы-Драйверы» используется для планирования заявок в количественном выражении. Записи, вводимые через данную форму, сохраняются в реестре драйверов и используются для дальнейшего расчета сумм планирования.

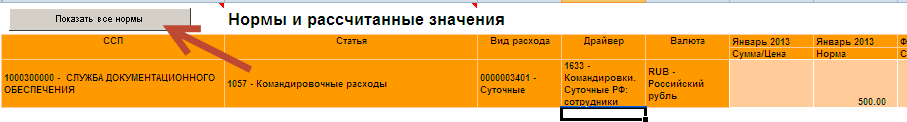
Форма состоит из двух листов:

* Лист «Нормы» предназначен для просмотра норм и сумм/цен, образованных путём умножения нормы на количество по драйверу.
* Лист «Драйверы» предназначен для ввода количественных значений драйвера.

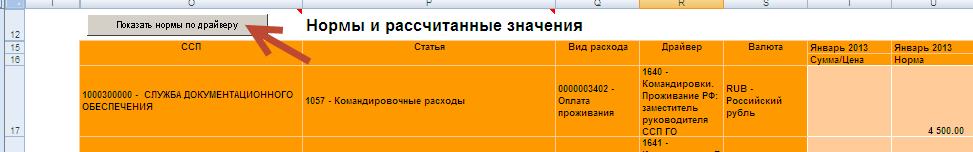
**Ввод количества по драйверу**

Для ввода количества по драйверу необходимо:

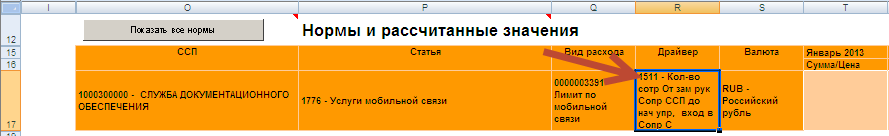
1. Установить требуемые фильтры с помощью контекста (см. раздел «Контекст»)
2. На листе «Нормы» нажать кнопку «Показать все нормы» для отображения всех норм в рамках ограничений фильтра (Рис.9).

Рис.9 Форма «Нормы»

1. Если требуется вывести данные только по определенным драйверам, то ограничить фильтр требуемым значением драйвера, и нажать на кнопку «Показать нормы по драйверу» (Рис.10).

Рис.10 Форма «Нормы»

1. В столбце «Драйвер» щелкнуть по ячейке с названием драйвера, требуемого для ввода/корректировки (Рис.11).

Рис.11 Форма «Нормы»

1. Перейти на лист «Драйверы» (Рис.12)



Рис.12 Форма «Нормы»

1. Для ввода/корректировки количественных значений драйвера необходимо ввести значения в соответствующие столбцы с помесячным разбитием количества драйвера в периоде планирования (Рис.13).

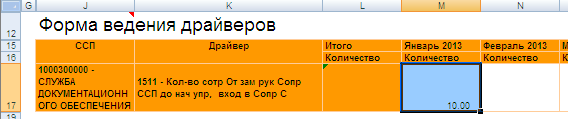


Рис.13 Форма «Нормы»

1. Для сохранения данных после ввода количества необходимо нажать кнопку «Сохранить данные» (Рис.14).

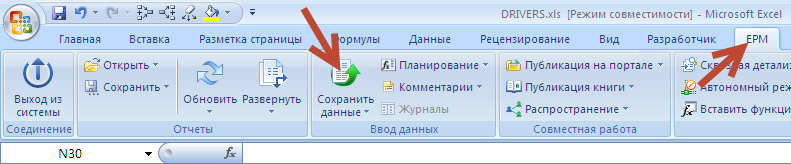


Рис.14 Форма «Нормы»

1. Подтвердить сохранение нажатием на кнопку «Да» (Рис.15).

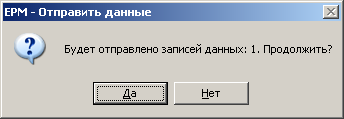


Рис.15 Форма «Нормы»

1. Если данные успешно сохранены, то в появившемся окне отразится соответствующее состояние и количество сохраненных данных (Рис.16).

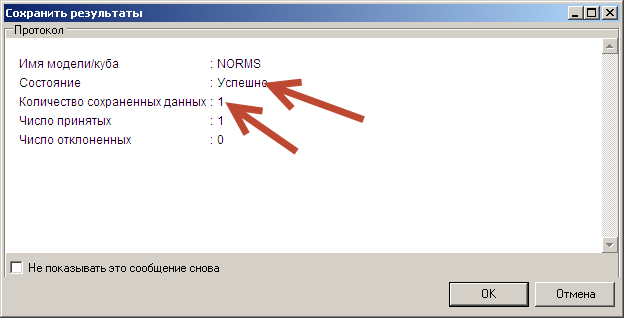


Рис.16 Форма «Нормы»

1. Для ввода/корректировки количественных значений по другому драйверу, требуется вернуться на лист «Нормы» и перейти к шагу 3.

## 3.2 Форма «Амортизация»

**Расчет плановой амортизации**

Для расчета плановой амортизации необходимо:

1. Открыть форму «Амортизация»
2. Перейти на лист «База», где будут отображены запланированные расходы.
3. Установить требуемые фильтры по измерениям
4. Нажать на кнопку «Сохранить». Будет произведен расчет плановой амортизации по выбранным в фильтре ограничениям (Рис.17).



Рис.17 Форма «Амортизация»

1. Подтвердить сохранение данных нажатием на кнопку «Да» (Рис.18):

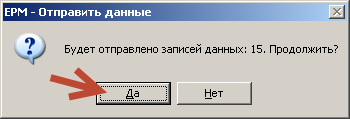


Рис.18 Форма «Амортизация»

1. Будет выведен протокол о сохранении. Убедиться, что напротив поля «Состояние» стоит значение «Успешно» и нажать на кнопку «ОК» (Рис.19):

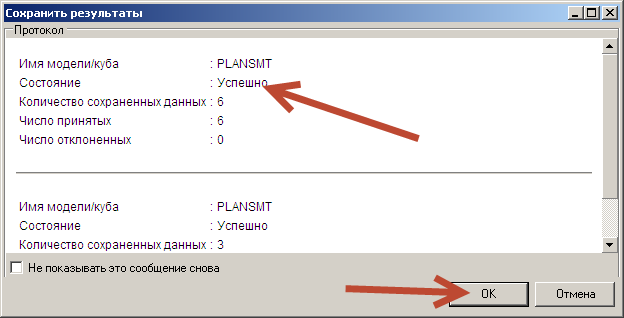


Рис.19 Форма «Амортизация»

1. Перейти на лист «Результат»
2. Установить требуемые фильтры, чтобы просмотреть результаты расчета плановой амортизации.

## 3.2 Основная форма планирования

**Описание основной формы планирования**

Основная форма планирования представляет собой Excel-книгу с 3 листами:

Лист «Просмотр» - используется для просмотра всех плановых данных.

Лист «Изменение» - используется для ввода всех плановых данных, кроме нормируемых расходов.

**Просмотр плановых данных**

Для просмотра плановых данных необходимо:

1. Открыть Основную форму планирования
2. Перейти на лист «Просмотр»
3. Нажать на кнопку «Фильтр» для ограничения выборки данных.

При необходимости детализации данных встать на любую ячейку в требуемой для детализации строке и нажать на кнопку «Детализация». При этом Вы перейдете на лист «Изменение».

**Ввод плановых данных**

Для ввода плановых данных необходимо:

1. Открыть Основную форму планирования
2. Перейти на лист «Изменение»
3. Для добавления новой строки нажать на кнопку «Добавить»
4. Откроется окно выбора элементов новой строки. Причем, если перед нажатием кнопки «Добавить» была выделена ячейка в строке с данными, то они автоматически подтянуться в окно выбора элементов новой строки. Если ячейка была пустой, то значения в полях измерений не будут заполнены (Рис.20).

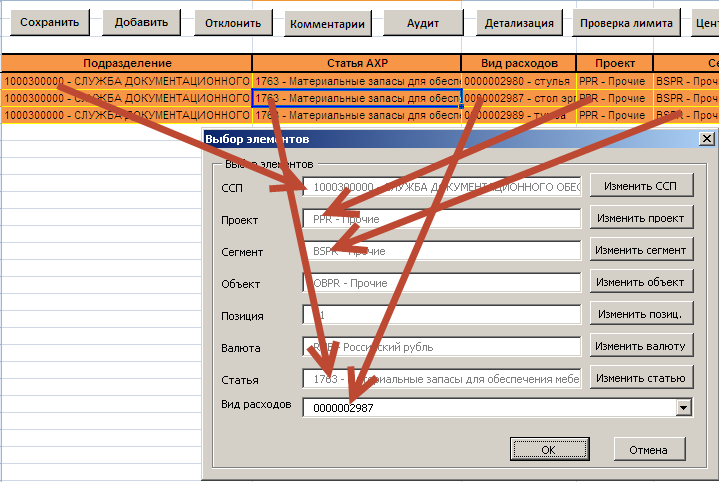


Рис.20 Форма «Основная»

1. Выбрать необходимые элементы, нажав на кнопки изменения, для каждого измерения. По завершении выбора элементов для всех измерений нажать на кнопку «ОК» для добавления новой строки (Рис.21).

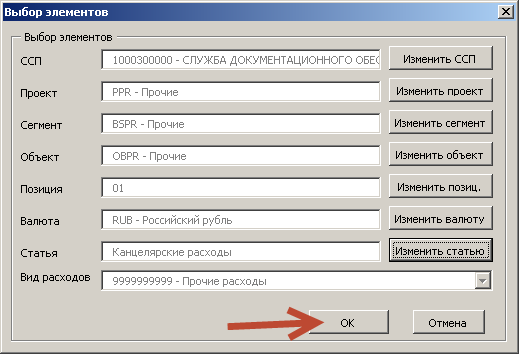


Рис.21 Форма «Основная»

1. Ввести обоснование в поле «Комментарий»
2. Нажать на кнопку «Сохранить» (Рис.22).



Рис.22 Форма «Основная»

1. Подтвердить сохранение данных нажатием на кнопку «Да» (Рис.23).

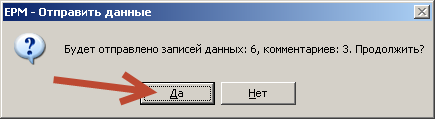


Рис.23 Форма «Основная»

1. Будет выведен протокол о сохранении. Убедиться, что напротив поля «Состояние» стоит значение «Успешно» и нажать на кнопку «ОК» (Рис.24).

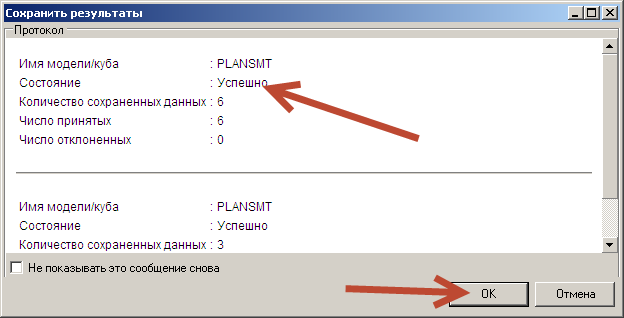


Рис.24 Форма «Основная»

**Централизация (Перенос данных с ССП на ССП ОБР)**

Для централизации расходов необходимо:

1. Открыть Основную форму планирования
2. Перейти на лист «Просмотр»
3. Нажать на кнопку «Фильтр» для ограничения выборки данных (см. раздел «Множественный фильтр»)
4. Встать на любую ячейку строки с требуемыми данными и нажать «Детализация», при этом Вы перейдете на лист «Изменение»
5. Нажать правой кнопкой мыши по любой ячейке строки и в контекстном меню выбрать «Централизация (касса)» - для централизации значений раздела «Касса», либо выбрать «Централизация (начисление)» - для централизации значений раздела «Начисление» (Рис.25).

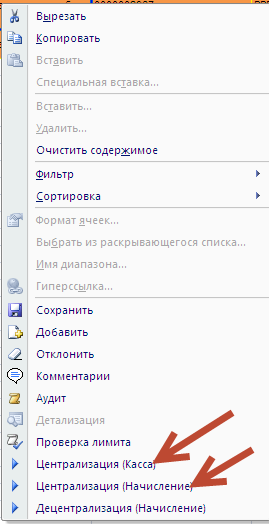


Рис.25 Форма «Основная»

1. Выбрать ССП ОБР, на которое требуется перенести значения.
2. После нажатии на кнопку «ОК» окна выбора элементов, будет произведен перенос данных на выбранное ССП ОБР.

**Децентрализация (Перенос данных с ССП ОБР на ССП)**

Для децентрализации расходов необходимо:

1. Открыть Основную форму планирования
2. Перейти на лист «Просмотр»
3. Нажать на кнопку «Фильтр» для ограничения выборки данных (см. раздел «Множественный фильтр»)
4. Встать на любую ячейку строки с требуемыми данными и нажать «Детализация», при этом Вы перейдете на лист «Изменение»
5. Нажать правой кнопкой мыши по любой ячейке строки и в контекстном меню выбрать «Децентрализация (начисление)» - для децентрализации значений раздела «Начисление» (Рис.26).

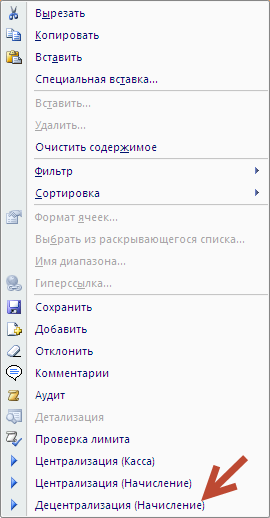


Рис.26 Форма «Основная»

1. Выбрать ССП, на которое требуется перенести значения.
2. После нажатии на кнопку «ОК» окна выбора элементов, будет произведен перенос данных на выбранное ССП.

# Заключение

В рамках решения поставленных проблем модели данных систем бюджетирования была построена модель, отвечающая как требованиям по вводу, так и по оперативному анализу данных.

Предполагается, что данная модель позволит обеспечить real-time доступ ко всей корпоративной финансовой информации, а также соответствовать требованиям по производительности операций вставки, обновления и удаления, согласованности данных.

Несмотря на то, что модель привязана к деталям физической реализации вычислений и требует значительных технических изменений, она отвечает самым последним технологиям СУБД.

Сфера баз данных сейчас переживает серьезные преобразования, которые с ростом объемов данных в ближайшем будущем, коснутся и полного пересмотра физических основ БД. Это своеобразная не только абстракция, но и фундамент, который на определенном этапе развития должен быть пересмотрен. И здесь новые концепции хранения и обработки данных позволяют нам добиться как эффективного масштабирования производительности, так и других технических возможностей.

# Используемая литература

1. Kimball, Ralph «The data warehouse ETL toolkit: practical techniques for extracting, cleaning, conforming, and delivering data» / Ralph Kimball – Indianapolis: Wiley Publishing Inc., 2004 – 526 стр.

2. Palekar, Amol «SAP NetWeaver BW 7.3 — Practical Guide (2nd Edition)» / Amol Palekar, Bharat Patel, and Shiralkar Shreekant – Boston: SAP PRESS, 2012 – 784 стр.

3. Patrick O'Neil, Elizabeth O'Neil, Xuedong Chen and Steve Revilak «The Star Schema Benchmark and Augmented Fact Table Indexing, Presentation at TPCTC» // University of Massachusetts Boston Library – 2009. [Электронный ресурс]. URL: <http://www.cs.umb.edu/~poneil/TPC_Talk082409.pdf> (дата обращения 20.05.2013).

4. Jones, Peter «Implementing SAP Business Planning and Consolidation» / Peter Jones – Boston: SAP PRESS, 2012 – 648 стр.

5. Peter Bakkum, Srimat Chakradhar «Efﬁcient Data Management for GPU Databases» // University of Delaware Library – 2012. [Электронный ресурс]. URL: <http://www.eecis.udel.edu/~cavazos/cisc879-spring2012/papers/datamanagement.pdf> (дата обращения 20.05.2013).

6. Franz Färber, Sang Kyun Cha, Jürgen Primsch, Christof Bornhövd, Stefan Sigg, Wolfgang Lehner «SAP HANA Database - Data Management for Modern Business Applications» // Association for Computing Machinery Digital Library – 2012. [Электронный ресурс]. URL: <http://dl.acm.org/citation.cfm?id=2094126> (дата обращения 20.05.2013).

7. Venkata Ponnam, Indiana University «A survey on Cassandra» //Indiana University Library – 2011: [Электронный ресурс]. URL: <http://salsahpc.indiana.edu/b534projects/sites/default/files/public/1_A%20survey%20on%20Cassandra_Ponnam,%20Venkata%20Shravan.pdf> (дата обращения 20.05.2013).

8. Biswapesh Chattopadhyay, Liang Lin, Weiran Liu, Sagar Mittal, Prathyusha

Aragonda, Vera Lychagina, Younghee Kwon, Michael Wong «Tenzing

A SQL Implementation On The MapReduce Framework» // Google Research Library – 2010: [Электронный ресурс]. URL: [http://static.googleusercontent.com/external\_content/untrusted\_dlcp/research.google.com/ru//pubs/archive/37200.pdf](http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/ru/pubs/archive/37200.pdf) (дата обращения 20.05.2013).

9. Martina-Cezara Albutiu, Alfons Kemper, Thomas Neumann «Massively Parallel Sort-Merge Joins in Main Memory Multi-Core Database Systems» // Association for Computing Machinery Digital Library – 2012. [Электронный ресурс]. URL: <http://dl.acm.org/citation.cfm?id=2336678> (дата обращения 20.05.2013).

10. Dengfeng Gao, Jose Alvin G. Gendrano, Bongki Moon, Richard T. Snodgrass, Minseok Park, Bruce C. Huang, Jim M. Rodrigue «Main Memory-Based Algorithms for Efﬁcient Parallel Aggregation for Temporal Databases» // Association for Computing Machinery Digital Library – 2012. [Электронный ресурс]. URL: <http://dl.acm.org/citation.cfm?id=992379> (дата обращения 20.05.2013).

11. Werner Vogels «Eventually Consistent» // All Things Distributed Library – 2008. [Электронный ресурс]. URL: <http://www.allthingsdistributed.com/2008/12/eventually_consistent.html> (дата обращения 20.05.2013).

12. Daniel J. Abadi, Samuel R. Madden, Nabil Hachem «Column-Stores vs. Row-Stores: How Different Are They Really?» // MIT Computer Science and Artificial Intelligence Lab – 2008: [Электронный ресурс]. URL: <http://db.csail.mit.edu/projects/cstore/abadi-sigmod08.pdf> (дата обращения 20.05.2013).

13. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber at Google «Bigtable: A Distributed Storage System for Structured Data» // Google Research Library – 2006: [Электронный ресурс]. URL: [http://static.googleusercontent.com/external\_content/untrusted\_dlcp/research.google.com/ru//archive/bigtable-osdi06.pdf](http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/ru/archive/bigtable-osdi06.pdf) (дата обращения 20.05.2013).

14. Vishal Sikka, Franz Faerber, Wolfgang Lehner, Sang Kyun Cha, Thomas Peh, Christof Bornhoevd «Efficient Transaction Processing in SAP HANA Database» // Michigan State University Library – 2012: [Электронный ресурс]. URL: <http://www.cse.msu.edu/~pramanik/teaching/courses/cse880/12f/seminars/2012f/p731-sikka.pdf> (дата обращения 20.05.2013).

15. Berg, Bjarne «SAP HANA: An Introduction» / Bjarne Berg, Penny Silvia – Boston: SAP PRESS, 2012 – 428 стр.

|  |  |
| --- | --- |
|  |  |