Евгеньев Р.А. Обработка диссоциативных наборов данных при решении задач аудита // Информационно-экономические аспекты стандартизации и технического регулирования, 2018. № 6(46).

УДК 004.89

ОБРАБОТКА ДИССОЦИАТИВНЫХ НАБОРОВ ДАННЫХ ПРИ РЕШЕНИИ ЗАДАЧ АУДИТА

Евгеньев Р.А., соискатель, ФГБУН ВИНИТИ РАН

Аннотация. *Все хозяйствующие субъекты окружены потоком разрозненной информации, которую необходимо сохранить и обработать не только для коммерческих целей, но и в рамках обязательств перед контролирующими органами. Наличие регуляторов и аудиторов обуславливает архитектуру информационной системы предприятия, способы хранения данных и их типологию.*

Ключевые слова: Data Warehouse, Data Lake, Lambda архитектура, Kappa архитектура, Iota архитектура, Audit-Driven архитектура.

UDC 004.89

AUDIT – DRIVEN HETEROGENIC DATASETS PROCEEDING

Evgenyev R.A., Institute of Scientific and technical information of the Russian Academy of Sciences

Abstract. *The information that surrounds any company is huge and various. This information is critical not only from the point of view of business, but also from the legal point of view. The existence of regulators and auditors influences the type of the database architecture and the way of proceding the incoming data.*

Keywords: Data Warehouse, Data Lake, Lambda architecture, Kappa architecture, Iota architecture, Audit-Driven architecture.

Вопрос хранения и обработки информации об экономической деятельности предприятия и его окружения является критическим не только с коммерческой точки зрения, но и с точки зрения нормативно-правовой базы. Предприятие постоянно получает разрозненные данные, имеющие разную степень важности и частоту обновления. Весь массив этой информации нуждается в структурировании и анализе. Однако начальным этапом

обработки данных является их запись в архив или общую базу данных. Основными типами хранилищ данных являются Data Warehouse и Data Lake. Data Warehouse – это одна из старейших систем агрегации разрозненных данных [2]. Изначально её можно найти в продуктах Oracle Server, например.

Data Lake. Эта архитектура является более гибкой, чем Data Warehouse. Она позволяет обрабатывать неструктурированные и денормализованные данные. Важной особенностью Data Lake является возможность распределенной обработки данных с горизонтальным скалированием мощности для вычислений. Стоит оговориться, что данная система не является универсальной. Очень популярный среди разработчиков и аналитиков Hadoop под ней работает не очень хорошо. Это связано с тем, что Hadoop приспособлен для параллельных данных, которые не зависят друг от друга. Например, у нас есть выборка из нескольких миллионов объектов, между которыми нет практически никакой зависимости. С помощью Hadoop мы можем разбить нашу выборку на кусочки и далее обрабатывать партиционированный массив по мере необходимости. За счёт того, что в системе существует репликация файлов, иными словами, у нас есть более, чем одна копия одного и того же файла, мы можем производить расчёты с более высокой скоростью. Однако на этапе анализа результатов появляются сложности со сведением данных. Поэтому аналитика на Hadoop работает весьма ограниченно.

В таблице, представленной ниже, перечислены основные характеристики этих двух типов хранилищ данных. (Таблица 1)

Таблица 1

Основные характеристики двух типов хранилищ данных: Data Warehouse; Data Lake

Data Warehouse	Свойства	Data Lake
Структурированные, обработанные	Данные	Неструктурированные, мультиструктурные
Предопределённая структура	Обработка	On read схема

Высокая цена для больших объёмов данных	Хранение	Спроектирована для большого количества наименований
Маленькая гибкость, преконфигурированная структура	Гибкость	Большая гибкость, но сложная перенастройка
Высокий уровень безопасности	Безопасность	Средний уровень безопасности
Бизнес-сообщество	Пользователи	Аналитики данных и инженеры

Лямбда архитектура. В любой системе существует два вида данных, обрабатываемых либо в пакетном режиме, либо в поточном. Именно из-за сложности обработки последнего типа данных с помощью Hadoop, для которого пришлось разрабатывать определённое количество расширений, таких как Storm, например, инженеры пришли к Лямбда архитектуре [1]. Она состоит из трёх слоёв (см. рисунок 1) :

- «Медленная» память (Batch ; high-latence layer for historical data)
- Слой для обработки поточных данных (Speed layer for recent/stream data)
- Слой для сведения данных (Smart reconcilation layer)

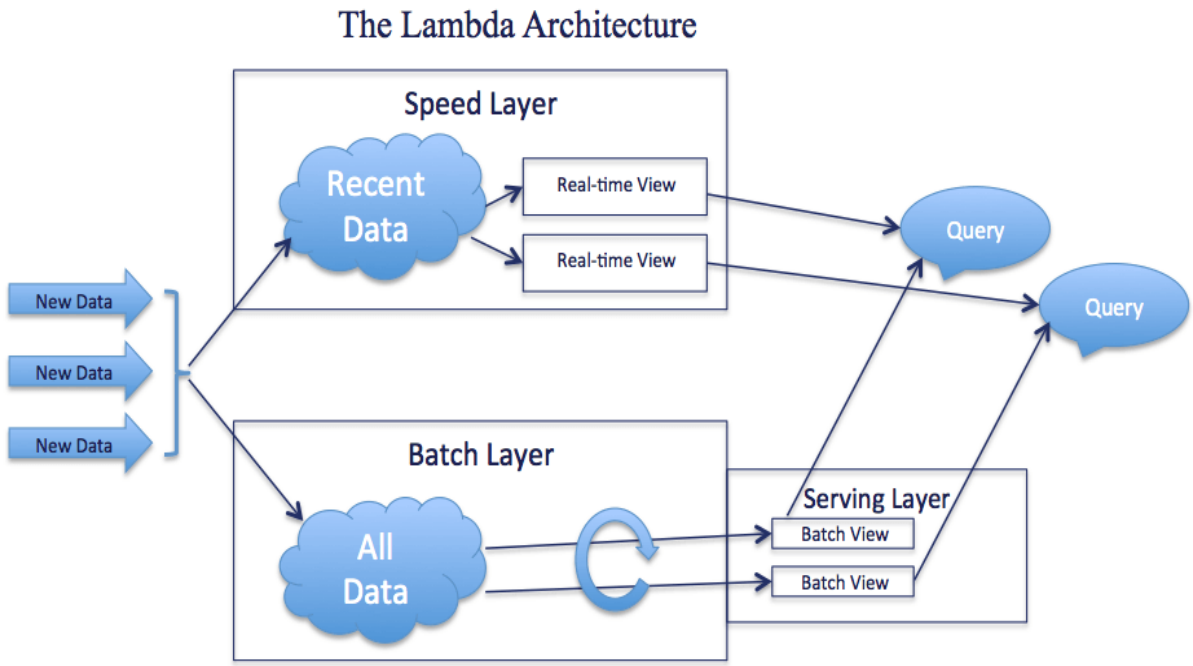


Рисунок 1. Лямбда архитектура обработки данных.

К недостаткам Лямбда архитектуры можно отнести, во-первых, необходимость загрузки данных только через speed layer и, во-вторых, большое количество неточностей в обрабатываемом массиве данных в слое Batch. Два вышеперечисленных недостатка весьма усложняют и замедляют весь процесс : происходит не только отъём памяти на вычисления, но и увеличивается шаффл по сети.

Kappa архитектура. Чтобы устранить недостатки лямбда архитектуры, была предложена Каппа архитектура [3]. В её основе лежит идея отказа от слоя Batch и интерпретация всех данных как поточных (см. рисунок 2).

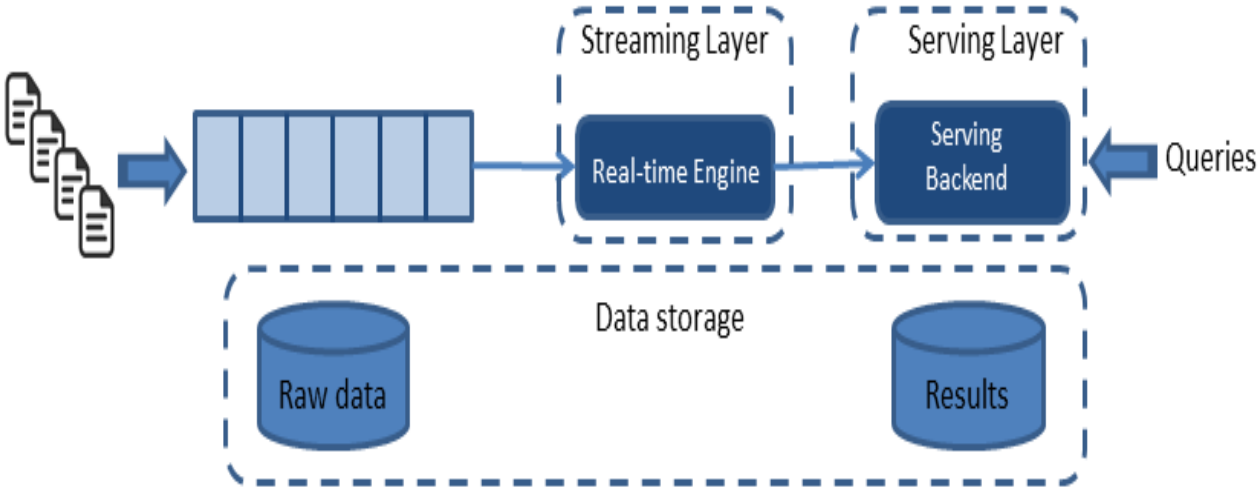


Рисунок 2. Каппа архитектура обработки данных.

Таким образом, в Каппа архитектуре стрим обрабатывает и исторические данные, и данные, входящие в систему в реальном времени. Преимуществом данной схемы является относительная простота пересчёта данных. Не нужно пересчитывать весь массив, достаточно запустить новый поток (stream job). Это свойство Каппа архитектуры значительно снижает её оперативную сложность (operational complexity).

In – Memory Data Fabric. С удешевлением стоимости и увеличением мощности накопителей, а так же с усовершенствованием архитектуры хранилищ, обработка и хранение данных в памяти стало весьма простым. За прошедшие двадцать лет цена гигабайта памяти упала примерно в 20000 раз. Одновременно с этим увеличилась скорость чтения данных в сети и обогнала скорость чтения с диска. Это иллюстрирует слова Джона Гейджа из Sun Microsystems, сказавшего в 1984 году : « Network is the computer. »

IMDF состоит из трёх уровней (см. рисунок 3.) На нижнем уровне находятся источники. Это могут быть различные базы данных, облачные хранилища, Enterprise Data Warehouse, и т.д. На верхнем уровне находятся потребители, использующие, например SQL. А в середине расположен огромный дистрибутивный кластер, у которого Compute совмещён с In-memory хранилищем. Это даёт возможность обрабатывать данные напрямую из памяти не сохраняя их на диске или в сети.

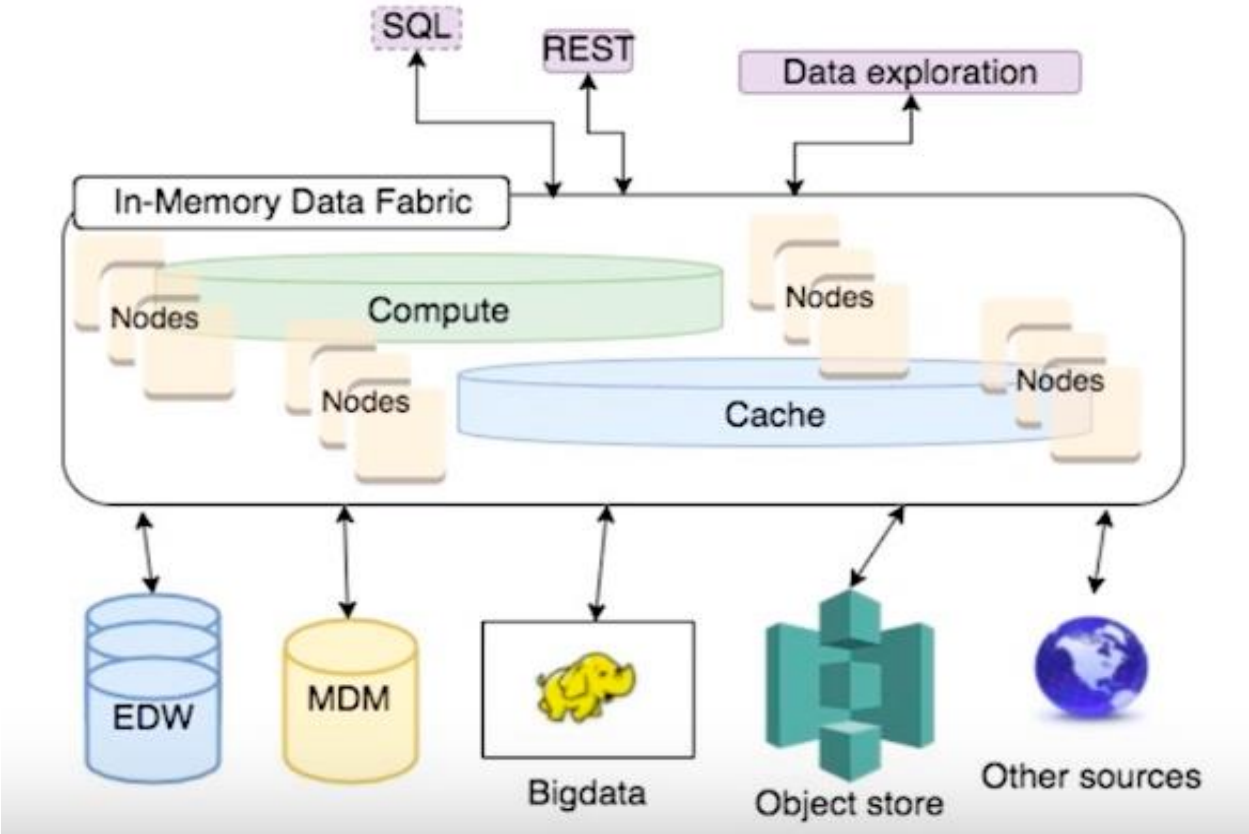


Рисунок 3. Устройство In-Memory Data Fabric.

IMDF представляет собой универсальный взгляд на данные, которые подсчитываются в кэш (cache) по мере необходимости или удаляются из него. В дополнение, на нодах связанных с кэшем можно выполнять различные вычисления : стриминг, map-reduce. Так же есть возможность распределять задачи из кэша между конкретными процессорами.

IOTA архитектура стала естественным продолжением развития вычислительных мощностей и оперативной памяти и имеет в своей основе вышеописанную IMDF (см. рисунок 4). Data Fabric находится на среднем уровне данной системы и состоит из кэша и множества нодов. На нижнем уровне расположены источники данных. Из них информация поступает с разной скоростью в кэш : некоторые источники обновляются раз в месяц, а некоторые – десятки раз в секунду. За счёт того, что в IMDF присутствует большое количество нодов и вычислительных фреймворков, данные обрабатываются с разной скоростью в зависимости от частоты обновления источников. На фоне данной конструкции встроен кэш с блоками данных и дата-каталог с репертуаром всех этих блоков, позволяющий строить необходимые нам дата-сеты.

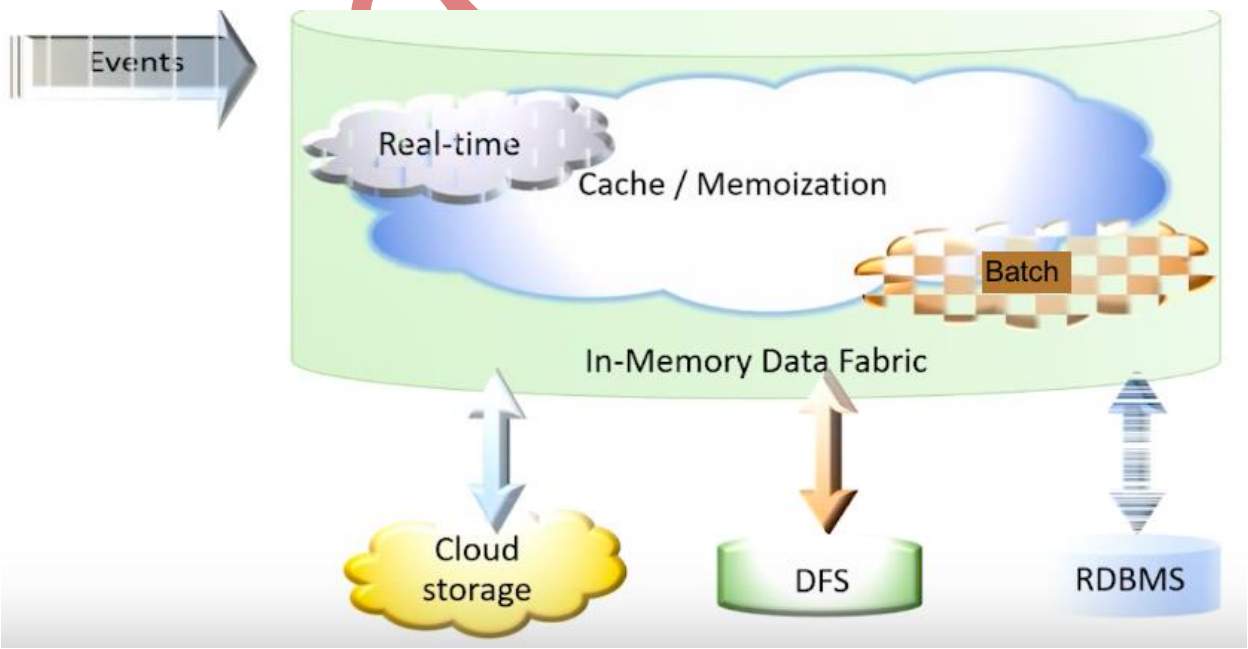
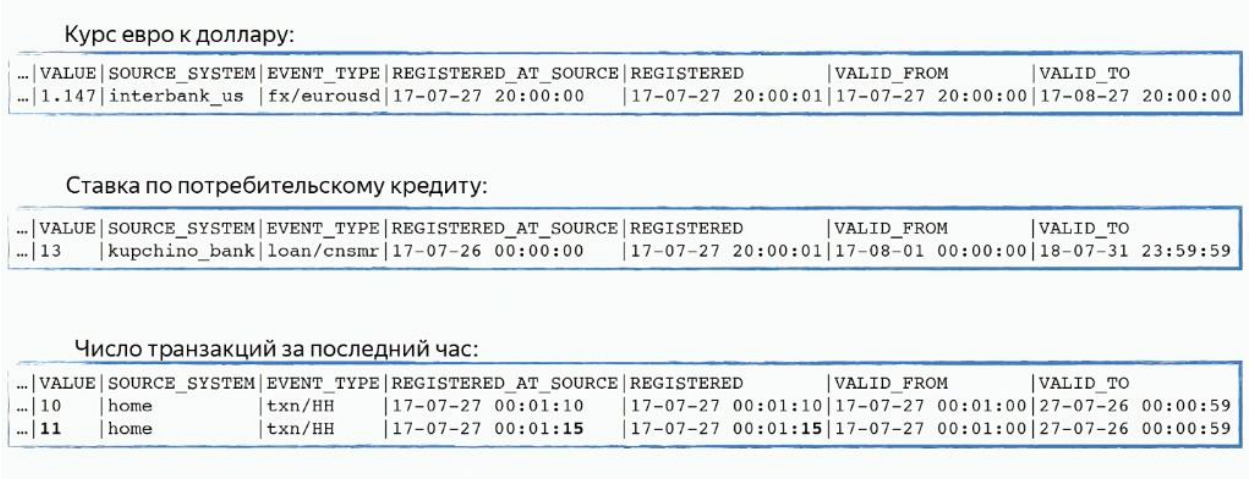


Рисунок 4. IOTA архитектура.

Audit-Driven архитектура. Данная архитектура специально адаптирована под требования регуляторов и позволяет обеспечить консистентонсть и верифицируемость данных на любом историческом отрезке времени. Основной проблематикой во взаимодействии компании с аудитором являются неточности в отчётах из-за задержек в обработке транзакций внутри системы, что в свою очередь влияет на итоговые месячные, квартальные и годовые показатели [4]. Общие очертания данной архитектуры весьма схожи с IOTA. Существует определённое количество входных данных с разной частотностью и сроком действия (курсы валют, изменяющиеся ежесекундно, данные поставщиков и продажи, обновляемые несколько раз в час, нормативные документы и регламенты, которые могут оставаться без изменений годами). Выходящий поток информации обращён к клиентам и контролирующим органам. А между этих двух потоков данных заключено хранилище и некая бизнес-логика, которая трансформирует входящую информацию в конечный продукт.

Основным свойством AD архитектуры является битемпоральность [5]. Для обеспечения консистентности данных принимают во внимание не только физическое время ознакомления с каким-либо фактом, но и техническое время, или время, в которое факт стал известен системе. На рисунке 5 представлены примеры обработки информации в хранилище на примерах курса валют, кредитной ставки банка и количества транзакций клиентов в системе.



Курс евро к доллару:

…	VALUE	SOURCE_SYSTEM	EVENT_TYPE	REGISTERED_AT_SOURCE	REGISTERED	VALID_FROM	VALID_TO
…	1.147	interbank_us	fx/eurousd	17-07-27 20:00:00	17-07-27 20:00:01	17-07-27 20:00:00	17-08-27 20:00:00

Ставка по потребительскому кредиту:

…	VALUE	SOURCE_SYSTEM	EVENT_TYPE	REGISTERED_AT_SOURCE	REGISTERED	VALID_FROM	VALID_TO
…	13	kupchino_bank	loan/cnsmr	17-07-26 00:00:00	17-07-27 20:00:01	17-08-01 00:00:00	18-07-31 23:59:59

Число транзакций за последний час:

…	VALUE	SOURCE_SYSTEM	EVENT_TYPE	REGISTERED_AT_SOURCE	REGISTERED	VALID_FROM	VALID_TO
…	10	home	txn/HH	17-07-27 00:01:10	17-07-27 00:01:10	17-07-27 00:01:00	27-07-26 00:00:59
…	**11**	home	txn/HH	17-07-27 00:01:**15**	17-07-27 00:01:**15**	17-07-27 00:01:00	27-07-26 00:00:59

Рисунок 5. Иллюстрация битемпоральности в хранилище данных.

Следующим важным свойством архитектуры направленной на аудит является хранение данных без изменений. В отличие от IOTA архитектуры, предполагающей удаление массивов информации, которую можно восстановить позднее путём пересчёта, в данном случае предпочтительней не использовать протокол CRUD (create, read, update, delete), что позволит не только обезопасить данные, но и гарантировать их консистентность. Таким образом, если необходимо обновлять, преобразовывать или удалять данные, то следует предварительно создать их новые версии.

Структура данных и способы их обработки в AD системе обуславливают и способ их хранения. Оптимальным является каскадное хранилище [5] (см. рисунок 6). Сначала данные попадают в хранилище первого порядка (например, HDFS, S3). На следующем этапе можно извлекать различные факты из этого массива данных и складировать их в хранилище фактов, а в дальнейшем строить необходимое количество каскадов хранилищ, в которые помещать информацию из хранилища фактов и обрабатывать её в зависимости от типа конкретной задачи. Таким образом, данные в предлагаемом хранилище структурируются и обрабатываются проходя через различные каскады и на выходе могут быть сведены в аналитический отчёт или графическую визуализацию.

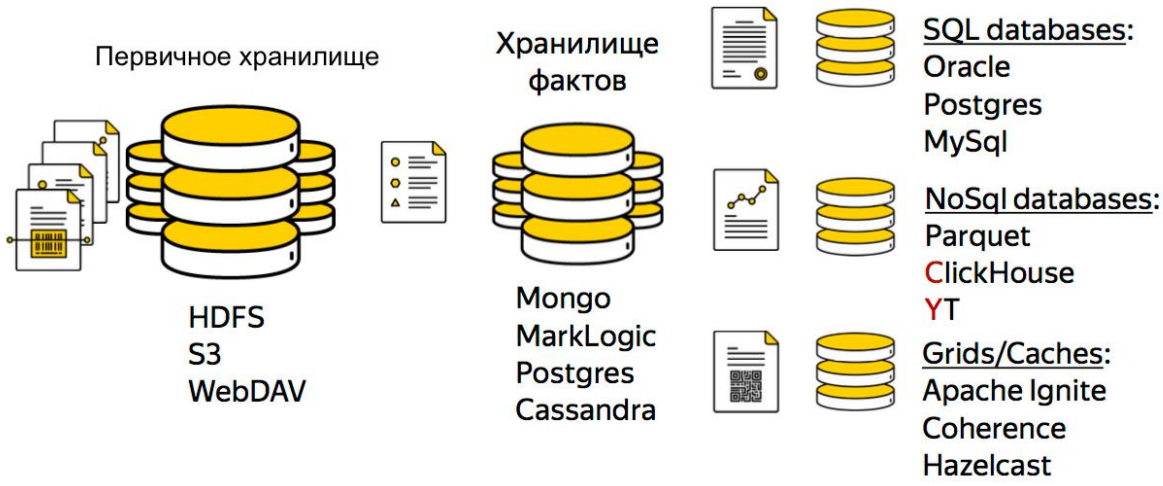


Рисунок 6. Каскадное хранилище данных.

Данные, обрабатываемые системой аудита можно разделить на быстро изменяющиеся и относительно статичные. К первой категории относятся, например, курсы валют, биржевые котировки акций или транзакции клиентов, ко второй – нормативная база или уставные данные клиентов. Таким образом, для первого типа данных оптимальным протоколом передачи является подписка (subscription), а для второго REST, SOAP или распределённые модули кэша (grid/cache). Следует оговориться, что информация от регуляторов, несмотря на свою статичную природу, должна передаваться клиенту через подписку. Это связано с тем, что изменения нормативной базы должны приниматься и актуализироваться системой незамедлительно с момента их публикации.

Список использованных источников и литературы

1. Mariam Kiran, Peter Murphy, Inder Monga, Jon Dugan, Sartaj Singh Baveja, «Lambda architecture for cost-effective batch and speed big data processing», 2015 IEEE International Conference on Big Data (Big Data)
2. S. Chaudhuri and U. Dayal, «An Overview of Data Warehousing and OLAP Technology», ACM SIGMOD Record, vol. 26, no. 1, 1997, pp. 65–74.
3. J. Lin, « The Lambda and the Kappa », IEEE Internet Computing, 2017
4. Hoisl B., Strembeck M. (2012) A UML Extension for the Model-Driven Specification of Audit Rules. In: Bajec M., Eder J. (eds) Advanced

Information Systems Engineering Workshops. CAiSE 2012. Lecture Notes in Business Information Processing, vol 112. Springer, Berlin, Heidelberg, p.24

5. Владимир Красильщик, Яндекс, Санкт-Петербург, Россия, SmartDataConf, 2017