Что относится к ммбд

-) В кратце о ммбд (1-2 ист)

Вот есть реляционные их краткое описание

А вот есть ммд источник 1

И что-то из источника 2

#### Multi-model Database A \*\*multi-model database\*\* is a database management system designed to support multiple data models against a single, integrated backend. This means that it can store, index, and query data in more than one model, such as document, graph, relational, and key-value models. This type of database provides a unified interface for data consistency, security, and access, and it eliminates the need for complex transformations and migrations between different databases [[1]](<https://en.wikipedia.org/wiki/Multi-model_database>).

Многомодельная база данных — это база данных, предназначенная для поддержки нескольких моделей данных в одной системе хранения данных. Это означает, что такая система может хранить, индексировать и запрашивать данные в нескольких моделях. Этот тип базы данных обеспечивает единый интерфейс для обеспечения согласованности, безопасности и доступа к данным, а также устраняет необходимость сложных преобразований и миграций между различными базами данных [1].

-) Какие бывают модели данных и краткое их описание (3-4+ ист)

Right now there are four main categories of NoSQL databases: key-value, column family, document and graph ones [2].

В настоящее время существует четыре основные модели баз данных: ключ-значение, семейство столбцов, документальные и графовые [2].

Возможно не хватает перехода с объяснением про схемы и NoSQL

Описание каждой и источник к ней

This is where Key Value stores come into play. Key value stores allow theapplication developer to store schema-less data. This data is usually consisting of a string which represents the key and the actual data which isconsidered to be the value in the ”key - value” relationship. The data itselfis usually some kind of primitive of the programming language (a string,an integer, an array) or an object that is being marshalled by the programming languages bindings to the key value store. This replaces the need forfixed data model and makes the requirement for properly formatted dataless strict.To make this more clear, here’s a short example (using ruby and the simplebut limited ”pstore” standard library) that should make that concept clearto any programmer.

Базы данных ключ-значения имеют хранилище ключ-значение. Они позволяют разработчику приложения хранить данные без схемы. Эти данные обычно состоят из строки, которая представляет ключ, и фактических данных, которые считаются значением в отношениях «ключ-значение». Сами данные обычно представляют собой своего рода примитив языка программирования (строка, целое число, массив) или объект, который создаётся привязками языков программирования к хранилищу ключ-значение. Это заменяет необходимость в фиксированной модели данных и делает строгими требования к правильному форматированию данных без данных [3].

In summary, a graph db-model is a model in which the data structures for the schemaand/or instances are modeled as a directed, possibly labeled, graph, or generalizationsof the graph data structure, where data manipulation is expressed by graph-orientedoperations and type constructors, and appropriate integrity constraints can be definedover the graph structure..

Графовая модель базы данных — это модель, в которой структуры данных для схемы и/или экземпляров моделируются как направленный, возможно, помеченный граф или обобщение структуры данных графа, где манипулирование данными выражается с помощью графо-ориентированных операций и конструкторов типов, а соответствующие ограничения целостности могут быть определены в структуре графа. [4]

The column-family database is another NoSQL databasethat stores data using a column-wise approach, unlikerelational ones, which organize data by rows.Data stored in a column family database is partitionedvertically, which makes partial read more efficient as only asubset of row attributes is loaded.

База данных семейства столбцов - это база данных NoSQL, которая хранит данные с использованием столбцового подхода, в отличие от реляционных, которые упорядочивают данные по строкам. Данные, хранящиеся в базе данных семейства столбцов, выбираются вертикально, что делает частичное чтение более эффективным, поскольку загружается только набор атрибутов строки [5].

Instead of storing data in fixed rows and columns, document databases use flexible documents. A document is a record in a document database. A document typically stores information about one object and any of its related metadata. Documents store data in field-value pairs. The values can be a variety of types and structures, including strings, numbers, dates, arrays, or objects.

Вместо хранения данных в фиксированных строках и столбцах базы данных документов используют гибкие документы. Документ – это запись в базе данных документов. Документ обычно хранит информацию об одном объекте и любых связанных с ним метаданных. Документы хранят данные в парах поле-значение. Значения могут быть различных типов и структур, включая строки, числа, даты, массивы или объекты [6].

-) Примеры ммбд с описанием какие categories of NoSQL они поддерживают (1-2+ ист + желательно с оф сайта)

Для каждой categories of NoSQL существую собственные ммд:

Перечисление с описанием и источником

Here are some of the most popular NoSQL databases:

1. ArangoDB is a free and open-source database manager that supports key-value, document, and graph database models.

2. The OrientDB engine supports Graph, Document, Key/Value, and Object models, so you can use OrientDB as a replacement for a product in any of these categories.

3. Azure Cosmos DB accommodates operational data models: document, vector, key-value, graph.

Далее представлены некоторые из наиболее популярных мультимодельных баз данных:

1. ArangoDB — это бесплатный менеджер баз данных с открытым исходным кодом, который поддерживает модели баз данных ключ-значение, документ и граф [7].

2. OrientDB поддерживает следующие модели баз данных: граф, документа, ключ-значение [8].

3. Azure Cosmos DB поддерживает следующие модели баз данных: документ, ключ-значение, граф [9].

-) Проблемы ммбд (потенциально в последнюю главу вынести) (1-3 ист)

Ммд имеет проблемы (перефразировать):

Перечисление проблем

—- Multi-model query processing and optimization. Despite ORDBMSs are capable of storing data with various formats (models), they do not provide a cross-modeldata processing language, inter-model compilation or respective multi-model query optimization. In contrast, a multi-model database attempts to embrace this challenge bydeveloping a unified query language to accommodate all the supported data models.As mentioned in the previous sections, there exist proposals of multi-model query languages. For example, AQL provided by ArangoDB enables one to access both graphand document data. However, the existing query languages are immature, and it isstill an open challenge to develop a full-fledged query language for multi-model data.A closely related problem is a proposal of an approach for identification of the optimal query plan for efficient evaluation of a given cross-model query [Lu 2017; Zhanget al. 2018]. Wavelets and histograms enable one to exploit the knowledge of distribution of data and thus optimize query evaluation strategies. However, the currenttechniques (e.g. [Alway and Nica 2016]) are developed for RDBMSs having a fixed relational schema, whereas multi-model DBMSs support both flexible and diverse schema.Thus, new dynamic techniques should be developed capable of adaptation to schemachanges.Currently the single-model DBMSs usually build a separate domain-specific indexfor different domains. Cross-domain queries are then evaluated by (1) separating indexsearches specifically for the individual domain, and (2) integrating the partial resultsto find all solutions. In the multi-model world we can use this approach too. For each ofthe models there exist verified types of indices, such as B-tree and B+-tree for relationaldata, TreePi [Zhang et al. 2007] and gIndex [Yan et al. 2004] for graph data, or XBtree [Bruno et al. 2002] for hierarchical XML data. However, the efficiency of suchapproach is questionable. A natural hypothesis is that a universal index comprisingvarious data models should quite probably be a better solution.In addition, the cloud-based distributed technologies are going forward. Cloud datacan be very diverse, including text, streaming data, unstructured and semi-structureddata. And cloud users and developers may be in high numbers, but not DBMS experts.Therefore, one challenge is to extend the technology of distributed database management and parallel database programming to fulfill the requirement of the scalability,simplicity and flexibility of the cloud-based multi-model data management.—- Multi-model schema design and optimization. A good design of the databaseschema is a critical part influencing many aspects, such as efficiency of query processing, application extensibility etc. There are critical decisions about both the physicaland logical schema of the data. For example, as shown in [Scherzinger et al. 2013] forthe case of key/value stores, a naive schema design will result in 20–35% of databasetransactions failing for a certain workload, whereas this problem can be alleviatedthrough the design of an appropriate schema. A similar paper [Mior 2014] provides acost-based approach to schema optimization in column stores. Contrary to relationaldatabases, NoSQL databases usually use significantly denormalized physical schemawhich requires additional space. Hence, in the world of multi-model systems we encounter contradictory requirements for the distinct models and thus it calls for a newsolution for multi-model schema design to balance and trade-off the diverse requirement of multi-model data.ACM Computing Surveys, Vol. 0, No. 0, Article 0, Publication date: 2019.0:31Even the question of existence of a schema differs significantly – traditional relational databases are based on existence of a pre-defined schema, whereas NoSQLdatabases are based on the assumption of schemalessness. A possible solution mayfind an inspiration, e.g., in the proposal of the NoSQL AbstractModel (NoAM) [Bugiottiet al. 2014], an abstract data model for NoSQL databases that specifies a systemindependent data representation. However, the proposal covers only aggregateoriented NoSQL databases (i.e., key/value, column, and document).A closely related problem of schema inference from a sample set of data instances isanother open issue in the multi-model context. There exists a number of approachesdealing with inference of, e.g., JSON [Baazizi et al. 2017] or XML [Mlynkov ´ a and ´Necasky 2013] schemas. Recently there have appeared approaches inferring a schema ´for NoSQL document stores [Gallinucci et al. 2018a], or in general for aggregateoriented databases [Sevilla Ruiz et al. 2015; Chillon et al. 2017]. There are even meth- ´ods which identify aggregation hierarchies in RDF data [Gallinucci et al. 2018b]. However, in the world of multi-model data we need to infer also references between thedistinct models. In addition, the inference approaches may benefit from informationextracted from related data with distinct models.—- Multi-model evolution. In general, it is a difficult task to efficiently managedata schema evolution and the propagation of the changes to the relevant portions ina database system, such as data instances, queries, indices, or even storage strategies.In some smaller applications a company can rely on a skilled database administrator tomanage the data evolution and to propagate the modification to other impacted partsmanually. But in most cases, it is a complicated and error-prone job.In the context of multi-model databases, this task is more subtle and difficult. Wecan distinguish intra-model and inter-model changes. In the former case we can reuse the existing approaches for single models. In the latter case, however, they cannotbe straightforwardly applied. The state-of-the-art solutions [Polak et al. 2015], usingthe classical Model-Driven Architecture, deal with multiple data models which represent distinct and overlapping views of a common model of the considered reality viawhich a change can be propagated to all affected parts. Then the change propagationcan be solved within particular data models separately. In the case of multi-modeldatabases the distinct models cover separate parts of the reality which are interconnected using references, foreign keys, or similar entities. Hence the evolution management has to be solved across all the supported data models. In addition, the challengeof query rewrite [Curino et al. 2008; Manousis et al. 2013], i.e. propagation of changesto queries, also becomes more complex in case of inter-model changes which requirechanges in data access constructs.—- Multi-model extensibility. The last but not least open problem is the challengeof model extensibility, which can be considered in several scopes. First, we may consider intra-model extensibility which means extending one of the models with newconstructs, e.g., extending the XML model with the support for the query on IDs andIDREF(S). Second, we may consider inter-model extensibility which adds new constructs expressing relations between the models, e.g. the ability to express a CHECKconstraint from the relational model across both relational and XML data. And third,we can provide extra-model extensibility which involves adding a whole new model,together with respective data and query, e.g. adding time series data with the supportof time series analysis.

—- Multi-model query processing and optimization. Despite ORDBMSs are capable of storing data with various formats (models), they do not provide a cross-modeldata processing language, inter-model compilation or respective multi-model query optimization. In contrast, a multi-model database attempts to embrace this challenge bydeveloping a unified query language to accommodate all the supported data models. However, the existing query languages are immature, and it isstill an open challenge to develop a full-fledged query language for multi-model data.A closely related problem is a proposal of an approach for identification of the optimal query plan for efficient evaluation of a given cross-model query.

—- Multi-model schema design and optimization. A good design of the databaseschema is a critical part influencing many aspects, such as efficiency of query processing, application extensibility etc. Contrary to relationaldatabases, NoSQL databases usually use significantly denormalized physical schemawhich requires additional space. Hence, in the world of multi-model systems we encounter contradictory requirements for the distinct models and thus it calls for a newsolution for multi-model schema design to balance and trade-off the diverse requirement of multi-model data. Even the question of existence of a schema differs significantly – traditional relational databases are based on existence of a pre-defined schema, whereas NoSQLdatabases are based on the assumption of schemalessness.

—- Multi-model evolution. In general, it is a difficult task to efficiently managedata schema evolution and the propagation of the changes to the relevant portions ina database system, such as data instances, queries, indices, or even storage strategies.In some smaller applications a company can rely on a skilled database administrator tomanage the data evolution and to propagate the modification to other impacted partsmanually. But in most cases, it is a complicated and error-prone job.In the context of multi-model databases, this task is more subtle and difficult.

—- Multi-model extensibility. The last open problem is the challengeof model extensibility, which can be considered in several scopes. First, we may consider intra-model extensibility which means extending one of the models with newconstructs. Second, we may consider inter-model extensibility which adds new constructs expressing relations between the models. And third,we can provide extra-model extensibility which involves adding a whole new model,together with respective data and query.

Многомодельная обработка и оптимизация запросов. Несмотря на то, что многомодельные БД способны хранить данные в различных форматах (моделях), они не обеспечивают язык межмодельной обработки данных, межмодельную компиляцию или соответствующую оптимизацию многомодельных запросов. Напротив, многомодельная база данных пытается решить эту проблему путем разработки унифицированного языка запросов, который будет охватывать все поддерживаемые модели данных. Однако существующие языки запросов еще незрелы, и разработка полноценного языка запросов для многомодельных данных все еще остается открытой задачей. Тесно связанной проблемой является предложение подхода к определению оптимального плана запроса для эффективной оценки заданный межмодельный запрос.

Проектирование и оптимизация многомодельной схемы. Хороший дизайн схемы базы данных является важной частью, влияющей на многие аспекты, такие как эффективность обработки запросов, расширяемость приложения и т.д. В отличие от реляционных баз данных, базы данных NoSQL обычно используют значительно денормализованную физическую схему, которая требует дополнительного места. Следовательно, в многомодельных системах мы сталкиваемся с противоречивыми требованиями к различным моделям, и, таким образом, требуется новое решение для проектирования многомодельных схем, чтобы сбалансировать и найти компромисс между разнообразными требованиями к многомодельным данным. Даже вопрос существования схемы существенно различается: традиционные реляционные базы данных основаны на существовании заранее определенной схемы, тогда как базы данных NoSQL основаны на предположении об отсутствии схемы.

Мультимодельная эволюция. В целом, эффективное управление развитием схемы данных и распространением изменений в соответствующих частях системы базы данных, таких как экземпляры данных, запросы, индексы или даже стратегии хранения, является сложной задачей. В некоторых небольших приложениях компания может положиться на квалифицированный администратор базы данных для управления развитием данных и распространения изменений на другие затронутые части вручную. Но в большинстве случаев это сложная и подверженная ошибкам работа. В контексте многомодельных баз данных эта задача более тонкая и трудная.

Многомодельная расширяемость. Последней открытой проблемой является проблема расширяемости модели, которую можно рассматривать в нескольких аспектах. Во-первых, мы можем рассмотреть возможность расширения внутри модели, что означает расширение одной из моделей новыми конструкциями. Во-вторых, мы можем рассмотреть межмодельную расширяемость, которая добавляет новые конструкции, выражающие отношения между моделями. И в-третьих, мы можем обеспечить дополнительную расширяемость модели, которая включает добавление совершенно новой модели вместе с соответствующими данными и запросом [10].

**Категории взаимодействия**

Описание каждого (перефразировать):

1. Polyglot persistence.

Одним из самых известных способов построения БД является полиглот (назвать по-другому и добавить на английском) и указать источник. [11]

1. Лямбда [11] (?)
2. Мультимодельные СУБД на основе реляционной модели
3. Мультимодельные СУБД на основе документной модели
4. СУБД «без основной модели»

Методы общения

Краткое описание каждого

Проблемы и сложности методов(?) работы с каждым

Текст

Источники (нормально оформить)

1. https://en.wikipedia.org/wiki/Multi-model\_database
2. The Multi-model Databases – A Review Ewa Płuciennik & Kamil Zgorzałek
3. Key-Value stores:a practical overview Marc Seeger Computer Science and MediaUltra-Large-Sites SS09 Stuttgart, Germany
4. Survey of graph database models Authors: Renzo Angles, Claudio Gutierrez
5. An Approach for Implementing Online Analytical Processing Systems under ColumnFamily Databases Abdelhak Khalil and Mustapha Belaissaoui
6. <https://www.mongodb.com/document-databases>
7. <https://docs.arangodb.com/3.11/concepts/data-models/>
8. <https://orientdb.org/docs/3.0.x/datamodeling/Tutorial-Document-and-graph-model.html>
9. <https://learn.microsoft.com/en-us/azure/cosmos-db/introduction>
10. Lu , J & Holubová , I 2019 , ' Multi-model Databases : A New Journey to Handle the Variety of Data ' , ACM Computing Surveys , vol. 52 , no. 3 , 55 . <https://doi.org/10.1145/3323214>
11. Polyglot database architectures = polyglot challenges