**Глава 1 Литературный обзор**

Данная глава последовательно даёт обзор источников информации из области современных баз данных, фреймворков (программных каркасов для построения информационных систем) и кластерного анализа данных, в качестве которых рассматриваются литература, электронные документы, компьютерные программы.

Активное развитие сферы информационных технологий значительно отразилось и на посвящённом ей литературном фонде: практически всегда можно найти несколько книг от авторитетных изданий на любую тематику. В то же время видна тенденция перехода формата обучения от печатных источников к видеоурокам и интерактивным обучающим курсам из-за более наглядного представления конечного результата и возможности сразу же приступить к выполнению практических заданий.

Стоит также отметить быстрое устаревание источников информации на тему информационных технологий: к актуальным можно отнести источники за последние 5-10 лет. Темп развития отрасли также сказывается и на объёме источников информации: значительная часть востребованных знаний представлена в виде коротких статей на веб-ресурсах, ссылаться на которые невозможно по ряду объективных причин.

Источники информации на тему кластерного анализа данных представлены немногочисленными книгами, больше информации можно найти в формате статей. Зачастую вопрос кластеризации данных раскрывается в источниках на смежную тематику.

1.1 Обзор современных СУБД

Первоочередной задачей является определение наиболее подходящей системы управления базами данных (СУБД) для хранения знаний.

На данный момент выделяют следующие основные группы СУБД:

- реляционные. Наиболее популярные представители: Oracle, MySQL, MS SQL Server, PostgreSQL, DB2, SQLite;

- столбцовые: C-Store;

- семейство столбцов: Cassandra, Hbase, Hypertable;

- «ключ-значение». Redis, Memcached, Riak, Amazon DynamoDB;

- документоориентированные. MongoDB, Couchbase, CouchDB;

- графовые: Neo4j;

- мультимодельные: OrientDB, ArangoDB [Elmasri].

Вкратце рассмотрим основные особенности каждой из групп.

Один из основных идеологов реляционного подхода к базам данных Эдгар Кодд предложил использовать для обработки данных аппарат теории множеств. Он продемонстрировал, что представление данных является совокупностью двумерных таблиц особого вида, называемых в математике «отношением» (англ. relation). Основными понятиями реляционных БД являются сущность, атрибут, первичный и внешний ключ. На практике сущностями являются таблицы, атрибутами — колонки таблиц, а ключи используются для установления отношений между таблицами [Кириллов].

Реляционные БД поддерживают три типа связи между сущностями: один-к-одному, один-ко-многим, многие-ко-многим. Связи между записями разных сущностей не поддерживаются.

Помимо теоретической работы над реляционным подходом к БД, Эдгар Кодд также создал и практический инструмент для работы с отношениями — реляционную алгебру. Каждая операция данной алгебры использует одну или несколько таблиц в качестве операндов и в итоге создаёт новую таблицу [Кириллов].

Столбцовые СУБД возникли вследствие недостатков производительности реляционных СУБД. В отличие от реляционной СУБД, где вся база данных хранится в одном файле, столбцовые СУБД хранят значения столбцов отдельно друг от друга, в разных файлах. Такой подход ограничивает загружаемый объём данных за каждый запрос (в реляционных СУБД записи загружаются со всеми столбцами), что снижает время совершения запросов и занимаемое дисковое пространство. Столбцовые СУБД используют язык запросов SQL [Фаулер].

Следующую группу направлений развития баз данных специалисты относят к так называемым NoSQL-базам данных. Мы не рассматриваем подробно происхождение и значение это термина (во многом из-за его неоднозначности), отметим его основное отличие — базы данных этой группы не используют SQL в качестве языка запросов по умолчанию (однако могут поддерживать ради совместимости) и привычную табличную структуру представления данных.

Отличительной особенностью баз данных «семейств столбцов» (англ. column-family, тж. wide-column) является распределение данных как на основе срок, так и на основе столбцов. При этом происходит объединение столбцов в группы или в семейства с целью показать, какие столбцы лучше хранить вместе. Базы данных данного типа позволяют каждой строке иметь различную структуру столбцов без каких-либо ограничений со стороны схемы, размер строк также может быть разным. В качестве параллели с базами данных «семейства столбцов» можно привести двумерный массив данных. Преимуществом данного вида БД является возможность легко добавить новые столбцы к существующим строкам; таблица может быть разреженной (множество null-значений) без каких-либо накладных расходов. Структурно БД «семейства столбцов» занимают положение между реляционными СУБД и БД «ключ-значение» [Редмонд]

Хранилища данных «ключ-значение» состоят из простых пар «ключа» и ассоциируемого с ним «значений», которое обычно является массивом данных. Подобные базы данных обеспечивают структуру, которая позволяет хранить и читать значения на основе «ключа». «Ключ» обычно является строкой и во многих отношениях схож с первичным ключом в реляционной БД. Отдельные записи в «значении» не отслеживаются и не различаются, поэтому при необходимости их изменения необходимо обновление всего «значения». Для базы данных «значение» представляет собой произвольный набор байтов, любая обработка которого отводится на использующую БД систему. Единственные операции, которые позволяют осуществлять БД «ключ-значение», - это put (для записи «значения»), get (для чтения «значения») и delete (для удаления пары «ключ-значение»). Операция обновления данных не поддерживается. [Hoffner]

Документоориентированная СУБД хранит данные в виде структурированных документов, обычно в формате XML или JSON. При этом определение «документоориентированная СУБД» не подразумевает какую-либо специфику насчёт модели хранения: документоориентированные СУБД могут выполнять ACID-транзакции или другие функции традиционных реляционных СУБД, хотя популярные документоориентированные обеспечивают относительно скромную транзакционную поддержку.

Документоориентированные базы данных, позволяя описывать данные без использования схемы, возможно, являются золотой серединой между жёсткой схемой реляционных баз данных и свободных от схемы хранилищ «ключ-значение». Сочетание с практикой веб-разработки вылилось в появление JSON-баз данных (MongoDB в частности), которые стали выбором по умолчанию для многих веб-разработчиков [Harrison G. Next Gen.].

Графовая база данных состоит из набора вершин (узлов, сущностей) и граней (связей, отношений, рёбер). Узлы воспринимаются как объекты со свойствами, между которыми моделируются отношения с помощью граней, которые также могут иметь свойства. Отношения имеют направления, на их основе происходит организация узлов, что позволяет единожды записать данные и затем по-разному их интерпретировать [Jordan] [Фаулер NoSQL].

Графовая структура позволяет представить данные в более естественном виде без искажений, как это может произойти в реляционных базах данных, а также применить различные типы графовых алгоритмов к этим данным. Одна из ключевых особенностей графовых БД — возможность обхода графа по его узлам и граням, перемещения от одного узла к другому, следуя направленным отношениям. Эта возможность называется «index free adjacency» (примерно переводится как смежность без индекса), смысл которой заключается в поиске прилежащих узлов без использования поиска по индексу, что значительным образом сказывается на производительности [Bruggen].

1.2 Выбор СУБД

Реляционный подход к СУБД зародился в конце 1960-х годов. К концу 1980-х годов реляционные СУБД стали наиболее популярным решением и сохраняют это положение на данный момент [Кузнецов Основы БД].

Несколько десятилетий разработчики программного обеспечения пытались приспособить связанные, полуструктурированные наборы данных к хранению в реляционных СУБД. Но хотя реляционные СУБД были изначально спроектированы для систематизации бланков и табличных структур, они плохо приспособлены для хранения ситуативных, исключительных связей, которые неожиданно возникают на практике [Robinson *Graph Databases*].

Связи между данными являются неотъемлемой частью реляционных СУБД, однако только на уровне моделирования, как средство объединения таблиц. Зачастую необходимо снять неоднозначность семантики связей, связывающих сущности, так же как и определить их вес или силу.

Реляционные СУБД не располагают подобным функционалом. Помимо этого, по мере накопления резко разнящихся значений и всенаправленного усложнения и размытия набора данных, реляционная модель перегружается соединёнными операцией JOIN таблицами, частично заполненными записями и множеством условий на отсутствие значений.

Рост взаимосвязанности в реляционной СУБД превращается в множество JOIN-операций, которые отрицательно сказываются на производительности и усложняют дальнейшую адаптацию существующего набора данных к дальнейшим возможным изменениям в бизнес-логике [Robinson *Graph Databases*].

СУБД только с коммерческой лицензией не соответствуют поставленным требованиям к создаваемой системе, поэтому СУБД Oracle в качестве СУБД для хранения знаний не рассматривается.

1.3 Современное состояние области кластерного анализа данных

Рассматривая область кластерного анализа данных, нужно отметить многообразие алгоритмов. На данный момент не существует общепринятой классификации алгоритмов кластеризации. Отдельные исследователи предлагают различные модели классификации, однако среди каждой из них можно выделить несколько основных направлений.

Иерархические алгоритмы: аггломеративные, дивизимные (BIRCH, CURE, ROCK, Chameleon, Echidna)

Разделяющие (K-means, K-medoids, K-modes, PAM, CLARANS, CLARA, FCM)

Плотностные (DBSCAN, OPTICS, DBCLASD, DENCLUE)

Сеточные (Wave-Cluster, STING, CLIQUE, OptiGrid)

Моделируемые (EM, COBWEB, CLASSIT, SOMs)

Иерархические алгоритмы разделяются на две группы: аггломеративные и дивизимные. Аггломеративные алгоритмы более популярны и многочисленны, нежели дивизимные, представляющие скорее исторический интерес к этапам развития кластерного анализа.

Иерархическим алгоритмам кластеризации присуще плохое масштабирование (O(n2) и хуже). В общем случае сложность аггломеративной кластеризации равна O(n3), что делает её слишком медленной для больших наборов данных. Сложность дивизимной кластеризации со всесторонним поиском равна O(2n), что хуже. Однако в некоторых исключительных случаях сложность аггломеративных алгоритмов равна O(n2): SLINK для одинарной связи и CLINK для полносвязной кластеризации [Sasirekha].

Результаты иерархического анализа обычно представляются в виде дендрограммы.