**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.

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;

- распределённые. Cassandra, Hbase;

- «ключ-значение». Redis, Memcached, 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].

Хранилища «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.].

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.