Постреляционные базы данных

В классической теории БД, модель данных - это формальная теория представления и обработки данных в СУБД, которая включает, по меньшей мере, три аспекта:
1) аспект структуры: методы описания типов и логических структур данных в БД;
2) аспект манипуляции: методы манипулирования данными;
3) аспект целостности: методы описания и поддержки целостности БД .

Аспект структуры определяет, что из себя логически представляет БД, аспект манипуляции определяет способы перехода между состояниями БД (то есть способы модификации данных) и способы извлечения данных из БД, аспект целостности определяет средства писаний корректных состояний БД.

Существует большое количество разновидностей БД, отличающихся по различным критериям. По признаку «модель данных» различают дореляционные, реляционные и постреляционные БД и соответствующие им СУБД. Постреляционные БД часто ассоциируют с БД NoSQL.

Англоязычная Wikipedia в статье «Database» не дает точного определения термина «постреляционная» (post-relational). Вместо этого заявляется следующее: «Жесткость реляционной модели, в которой все данные хранятся в таблицах с фиксированной структурой строк и столбцов, все чаще рассматривается как ограничение при работе с информацией, более богатой и более разнообразной по структуре, чем традиционные «книги книг» данных корпоративных информационных систем.» Проблема ограниченности реляционных баз данных решается двумя подходами:
1) Разрабатываются новые типы баз данных, которые позиционируются как постреляционные или NoSQL базы данных.
2) Поставщики реляционных баз данных расширяют возможности своих продуктов, например путем поддержки более широкого круга типов данных.

Wikipedia в статье «NoSQL» также не дает точного определения терминов постреляционные и NoSQL базы данных. Вместо этого заявляется следующее: «NoSQL - это термин, обозначающий ряд подходов, проектов, направленных на реализацию моделей баз данных, имеющих существенные отличия от используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL.» Сторонниками концепции NoSQL подчёркивается, что она не является полным отрицанием языка SQL и реляционной модели, проект исходит из того, что SQL - это важный и весьма полезный инструмент, но при этом он не может считаться универсальным.

В учебнике «Базы данных: языки и модели» автор учебника С.Д. Кузнецов пишет: «При всех достоинствах РМД, частично унаследованных в языке SQL, традиционный подход часто подвергался справедливой критике на основании того, что он породил кардинальные различия между прикладной программой и данными, хранимыми в базе данных (для обозначения этой ситуации обычно используется термин impedance mismatch - несоответствие).» Для решения этой проблемы было предложено три подхода, отраженных в трех документах, получивших название манифестов.

1. Манифест систем объектно-ориентированных баз данных (1989). Подход был развит консорциумом ODMG, последовательно опубликовавшим три версии спецификации объектной модели данных. В 2001 году ODMG завершил свою работу и был расформирован. Последний стандарт ODMG – «The Object Data Standard ODMG 3.0».
2. Манифест систем баз данных следующего поколения (1990). Необходимо сохранить все свойства SQL-ориентированных систем, но усовершенствовать их систему типов. Подход был поддержан в стандарте SQL:1999 и в последующих редакциях стандарта SQL (SQL:2003, SQL:2006, SQL:2008). Примеры реализации объектно-реляционных СУБД: Informix, Oracle, DB2.
3. Третий манифест (1995). Вернуться к первоначальной РМ, переосмыслив ее таким образом, чтобы допустить хранение в БД данных, типы которых определяются пользователем. Пример реализации: Dataphor компании Alphora.

NoSQL базы данных

Термин NoSQL (Not Only SQL - Не только SQL) введен в употребление в 2009 году для описания различных технологий баз данных, возникших для удовлетворения требований, известных как «Web-scale» или «Internet-scale». Описание схемы данных в случае использования NoSQL-решений может осуществятся через использование различных структур данных: хеш-таблиц, деревьев и т.д.

К Web-scale приложениям предъявляются следующие три требования:
1) много данных: (Facebook - 50 терабайт для поиска по входящим сообщениям; eBay - 2 петабайта);
2) «огромное» количество пользователей: исчисляются миллионами, доступ к системам выполняется одновременно и постоянно;
3) сложные данные: как правило, это приложения не простой обработки табличных данных, которые можно найти во многих коммерческих и бизнес-приложениях.

Технологии реляционных баз данных показывают свои слабые места при переходе к веб-масштабам именно в этих трех аспектах. Апологеты NoSQL поясняют, что проект NoSQL не занимается полным отрицанием языка SQL и реляционной модели, проект исходит из того, что SQL — это важный и весьма полезный инструмент, но при этом он не может считаться универсальным. Одной из проблем, которую указывают для классических реляционных БД, являются проблемы при работе с данными очень большого объема и в проектах с высокой нагрузкой. Основная цель проекта — расширить возможности БД там, где SQL недостаточно гибок, и не вытеснять его там, где он справляется со своими задачами.

В качестве одного из методологических обоснований подхода NoSQL используется эвристический принцип, известный как теорема CAP, утверждающий, что в распределённой системе невозможно одновременно обеспечить согласованность данных, доступность (англ. availability, в смысле наличия отклика по любому запросу) и устойчивость к расщеплению распределённой системы на изолированные части. Таким образом, при необходимости достижения высокой доступности и устойчивости к разделению предполагается не фокусироваться на средствах обеспечения согласованности данных, обеспечиваемых традиционными SQL-ориентированными СУБД с транзакционными механизмами на принципах ACID.

В июле 2011 компания Couchbase, разработчик CouchDB, Memcached и Membase, анонсировала создание нового SQL-подобного языка запросов – UnQL (Unstructured Data Query Language). Работы по созданию нового языка выполнили создатель SQLite Ричард Гипп (англ. Richard Hipp) и основатель проекта CouchDB Дэмиен Кац (англ. Damien Katz). Разработка передана сообществу на правах общественного достояния.

Разнообразие NoSQL баз данных велико и постоянно растет, но все они имеют общие черты:
1) обрабатываются огромные объемы данных, разделяемые между серверами;
2) поддерживается огромное количество пользователей;
3) используется упрощенная, более гибкая, не ограниченная схемой структура базы данных;

В основе NoSQL лежат следующие идеи:
1) нереляционность,
2) распределенность,
3) открытый исходный код,
4) горизонтальная масштабируемость.

Пояснение. Горизонтальная масштабируемость — это увеличение количества серверов для равномерного распределения клиентских запросом между ними и гарантии сохранения работоспособности всей системы в случае выхода из строя одного или даже нескольких узлов (технические характеристики отдельных серверов могут при этом могут быть намного ниже, чем при «вертикальном» масштабировании).

Хорошее общее введение в архитектуру NoSQL можно найти в статье Р. Твида и Дж. Джеймса «Универсальное NoSQL - введение в теорию»: http://blogerator.ru/page/nosql-vvedenie-v-teoriju-bd. На сайте http://nosql-database.org/ приводится список из 122 NoSQL СУБД, которые могут быть условно разделены на следующие категории:
- Поколоночные СУБД.
- Документо-ориентированные СУБД.
- Хранилища «ключ-значение», кортежные хранилища.
- Базы данных на основе графов.
- Другие модели данных.

Ярким представителем документо-ориентированных СУБД является MongoDB, краткое знакомство с которой можно выполнить с помощью статьи «Куда идет NoSQL с MongoDB» Тэда Ньюарда, которую можно найти по адресу https://msdn.microsoft.com/ru-ru/magazine/ee310029.aspx. **Замечание.** У этой статьи есть продолжения.

Не реляционные элементы в реляционной СУБД SQL Server

Ранее отмечалось, что многие поставщики реляционных баз данных расширяют возможности своих продуктов, например, путем поддержки более широкого круга типов данных. По такому пути пошла компания Microsoft.

xml
Тип данных, в котором хранятся XML-данные. Можно хранить экземпляры xml в столбце либо в переменной типа xml.

SQL Server поддерживает набор методов для типа данных xml:
- query() - выполняет запрос к экземпляру XML.

- value() - получает значения типа SQL из экземпляра XML.
- exist() - определяет, вернул ли запрос непустой результат.
- modify() - выполняет обновления.
- nodes() - разделяет XML на несколько строк для распространения XML-документов по наборам строк.

Для привязки реляционных данных с типом, отличным от XML, внутри XML используютсмя следующие псевдофункции:
- sql:column() - позволяет использовать значения реляционного столбца в выражении XQuery или XML DML.
- sql:variable() - позволяет использовать значения переменной SQL в выражении XQuery или XML DML.

SQL Server поддерживает поднабор языка XQuery, который используется для выполнения запросов к типу данных xml. Эта реализация XQuery совпадает с рабочим эскизом XQuery на июль 2004 г. Язык разрабатывается консорциумом W3C с участием всех основных поставщиков баз данных, а также корпорации Майкрософт. Дополнительные сведения см. в разделе Спецификация языка W3C XQuery 1.0.

geography
Географический пространственный тип данных geography в SQL Server реализуется как тип данных среды .NET CLR. Этот тип представляет данные в системе координат круглой земли и сохраняет эллиптические данные, такие как координаты широты и долготы в системе GPS.

SQL Server поддерживает набор методов для пространственного типа данных geography:
- методы для работы с типом geography, как они определены в спецификации консорциума OGC,
- набор расширений Майкрософт для этого стандарта.

geometry
Плоский пространственный тип данных geometry в SQL Server реализуется как тип данных среды CLR. Этот тип представляет данные в евклидовом пространстве (плоской системе координат).

SQL Server поддерживает набор методов для пространственного типа данных geometry:
- методы для работы с типом geometry, определенные спецификацией консорциума OGC,
- набор расширений Майкрософт для этого стандарта.

hierarchyid
Тип данных hierarchyid является системным типом данных переменной длины. Тип данных hierarchyid используется для представления положения в иерархии. Столбец типа hierarchyid не принимает древовидную структуру автоматически. Приложение должно создать и назначить значения hierarchyid таким образом, чтобы они отражали требуемые связи между строками. Значение типа данных hierarchyid представляет позицию в древовидной иерархии.

FILESTREAM
FILESTREAM позволяет приложениям на основе SQL Server хранить в файловой системе неструктурированные данные, например документы и изображения. Приложения могут одновременно использовать многопоточные API-интерфейсы и производительность файловой системы, тем самым обеспечивая транзакционную согласованность между неструктурированными и соответствующими им структурированными данными.

Хранилище FILESTREAM объединяет компонент Компонент SQL Server Database Engine с файловой системой NTFS, размещая данные больших двоичных объектов (BLOB) типа varbinary(max) в файловой системе в виде файлов. С помощью инструкций Transact-SQL можно вставлять, обновлять, запрашивать, искать и создавать резервные копии данных FILESTREAM. Интерфейсы файловой системы Win32 предоставляют потоковый доступ к этим данным.

Для кэширования данных файлов в хранилище FILESTREAM используется системный кэш NT. Это позволяет снизить возможное влияние данных FILESTREAM на производительность компонента Database Engine. Буферный пул SQL Server не используется, поэтому эта память доступна для обработки запросов.

FILESTREAM не включается автоматически при установке или обновлении SQL Server. FILESTREAM необходимо включить с помощью диспетчера конфигурации SQL Server и среды Среда SQL Server Management Studio. Для использования FILESTREAM нужно создать или изменить базу данных, которая будет содержать заданный тип файловой группы. После этого следует создать или изменить таблицу, чтобы она содержала столбец varbinary(max) с атрибутом FILESTREAM. После завершения выполнения этих задач можно будет пользоваться Transact-SQL и Win32 для управления данными FILESTREAM.