04.09.2014------------------------------------------------------------------------

Просуков Евгений Алексеевич

Алексеев Юрий Евтихович

Кузнецов с д - Базы данных, языки и модели

Дейт - Введение в системы баз данных

Ульман – Введение в системы базы данных

Insight SQL server 2008

SQL server и визуалка 12

База в сйлсервере – датабаз енжин. Остальные службы необязательны и запускаются по требованию.

[[[в рамках лабы

Три предметные сущности: поставщики, детали, проекты.

Атрибуты: имя поставщика, статус, город, (суррогатный)ключ(целоечисло) ; название, цвет, вес, склад, (суррогатный)ключ(целоечисло) ; название проекта, город, (суррогатный)ключ(целоечисло).

Сущности связываются между собой - предлагается тернарная(3) связь: поставщик А поставляет деталь Б для проекта Ц, определяется количество поставок.

Для связи требуется указать номера (ключи); в СЙЛ выступают в роли внешних ключей. Определяется связь между тремя таблицами и наделяется дополнительным свойством (ещё одна таблица).

ER-диаграммы:

Сущность - прямоугольник. Свойство - овал, связывается с сущностью прямой. Отношение - ромб. В ^ случае: "поставки", свойство отношения: "количество". Отношение оперирует ссылками на объекты, в данном случае - ключами (номерами).

В таблице поставщиков могут быть поставщики, ничего не поставляющие; детали, никем не поставляемые; проекты, для которых никто ничего не поставляет.

Один поставщик поставляет одну деталь для одного проекта. Тройка "номер поставщика, номер детали, номер проекта" в отношении может служить первичным составным ключа.

На основе ЕРдиаграммы нужно через инструкции СЙЛ создать датабазу. Переключить контекст на создание таблиц в датабазе.

При использовании строк - лучше использовать N char и N varchar, юникодовские строки, для избежания проблем импорта.

Для городов можно ограничить выбор строк из набора (таблицы). Допускаются ограничения NULL | NOT NULL; пустая строка из пробелов != отсутствующей строке.

Атрибуты могут быть объявлены в виде PRIMARY KEY - первичный ключ, уникальным образом идентифицирующий данную запись; ключ может быть многостолбцовым. Атрибуты могут быть объявлены UNIQUE. FOREIGN KEY - атрибут ссылающийся на другую таблицу (как правило - на первичный ключ или на уник). CHECK(проверка) - может использоваться в предикатах принадлежности.

В ЛР 2 части. 1 часть - СЙЛ(подъязык DDL) и сервер, создание баз и таблиц, добавление ограничений через AlterTable, оформление сценариев.sql. 2 часть - программа генерации правдоподобных тестовых данных для заполнения ими таблицы; ~1000поставщиков, \*2 поставок

Загрузка даты в датабазу: утилита bcp в командной строке; инструкция BulkInsert СЙЛ; средства экспорта\импорта СЙЛ сервера

]]]

Четыре РК, первый – на 4 неделе в конце лекции, в письменном виде. Структурная, целостная, манипулятическая части. В качестве примеров используется учебная БД. На РК – записать три сценария разными методами.

**1. Основные понятия и определения**

Базы данных. Системы БД (банки данных – Григорьев, Ревунков). Системы управления БД.

БД **O**n**L**ine**T**ransaction**P**rocessing и БД **O**n**L**ine**A**nalitical**P**rocessing.

ОЛТП – выполняются единицы работы, в результате которых происходит обновление записей. Работают по принципу «всё или ничего», нельзя выполнить часть действий и не выполнить другую часть. Транзакции должны быть короткими и выполняться быстро; ведётся журнал транзакций. Одновременно с одной БД обычно выполняется большое число транзакций.

ОЛАП – аналитическая обработка неизменной БД (хранилища данных, data warehouse).

Требования к данным ОЛТП:

1. данные должны быть неизбыточны (каждое данное присутствует в базе в единственном экземпляре)

2. совместное использование данных многими пользователями; отсутствие конфликтов. Достигается использованием блокировки данных (минимальный размер – одна запись) и таймаутом при долгой невозможности доступа

3. эффективность доступа к БД (малое время отклика). Достигается путём сортировки, различных алгоритмов поиска и способов реализации данных (сбалансированные деревья)

4. целостность (integrity) – данные должны быть корректными и согласованными

11.09.2014------------------------------------------------------------------------

5. безопасность данных – защита данных от как преднамеренного так и непреднамеренного разрушения. За это отвечает система управления БД

6. восстановление данных после программных и аппаратных сбоев – с использованием информационной избыточности, предоставляемой журналом транзакций

7. независимость хранимых данных от прикладных программ. Прикладная программа должна иметь представление только о логическом способе хранения данных, но не о физическом их представлении.

Система УПРАВЛЕНИЯ БД – приложение, обеспечивающее создание, хранение, обновление и поиск информации в БД.

СИСТЕМА БД – совокупность одной или нескольких БД и комплекса информационных\программных\технических средств, обеспечивающих работу системы.

Основные функции СУБД:

- непосредственное управление данными во внешней памяти

- управление буферами оперативной памяти (помимо той буферизации, которую организует ОС) для повышения производительности выполнения запросов

- управление транзакциями (с использованием механизма блокировок или управления версиями строк)

- журнализация (ведение журнала транзакций, его использование для фиксации транзакций, а также откатов\накатов)

- поддержка языков БД

Существует большое число признаков, по которым классифицируются СУБД. Основным является модель данных, положенных в основу базы. Признаки:

• модель данных: до-реляционные, реляционные, пост-реляционные. ДР: инвертированные файлы или списки, иерархические или древовидные, сетевые (похожи на графы).

В случае инвертированных файлов: создаётся дополнительная таблица (индекс) из двух столбцов: наиболее часто используемый атрибут, и идентификатор строки в оригинале; эта таблица сортируется. Выигрываем во времени, но проигрываем по памяти. По одному из атрибутов в оригинале таблица скорее всего будет упорядочена – это называется кластерным ключом. Этот способ организации называется IMS.

ПР (no-sql) базы делятся на множество категорий: active database, cloud –‘’-, temporal –‘’-, federated –‘’-, graph –‘’-, ...

• архитектура организация хранения: локальные и распределённые. Локальные – на одной машине, распределённые – на физически разных. Если на одной машине на одной инсталяции СЙЛ есть две БД, они считаются локальными; две инсталяции на одной машине – базы данных на них считаются распределёнными.

• способ доступа к БД: файл-серверные (устаревшие), клиент-серверные (текущего дня), встраиваемые, сервисно-ориентированные, прочие. *В ФС*: сервер пассивен и хранит файлы, клиенты копируют информацию и работают с копией. *В КС*: клиент формирует запрос, а сервер обрабатывает запросы (получает команду, выполняет действия, возвращает ограниченное количество строк не перегружая трафик). *В СО*: промежуточный слой представляет набор сервисом с конечными точками; конечные точки подсоединяются к серверу БД и к клиенту.

Большинство БД являются реляционными; пост-реляционные пока не заняли рынок.

**2. Реляционная модель БД**

69-70 год, Э. Кодд. Рассматривать модель будем в интерпретации К. Дейта и Хью Дарвена. Их представление со временем меняется, равно как и язык реляционной алгебры, предназначенный для работы с БД. Позже был разработан язык TutorialD, по объёму сравнимый с SQL. Rel – система, на которой можно выполнять запросы и объявления на языке TutorialD.

В реализации Дейта реляционная модель состоит из трёх частей: структурной, целостностной, манипуляционной.

Основные понятия **структурной** части: тип данных; домен; атрибут; схема отношения; схема базы данных; кортеж; отношение; потенциальный, первичный, и альтернативный ключи; реляционная база данных.

*Тип данных* – множество значений, операций и способ представления значений.

*Домен* – можно считать уточнением типа данных. Подмножество определенного типа, на которое наложено ограничение. Для определения домена могут применяться специальные инструкции, используются: имя, тип данных (или другой домен), логическое условие для описания подмножества.

*Атрибут* – упорядоченная пара <имя-домен>.

*Схема отношения* – «заголовок» будущей таблицы; именованая совокупность пар <имя-домен>. В учебной БД: S(Sno:int, Sname:string, status:STATUS, city:string); P(Pno:int, Pname:string, color:COLOR, weight:float, city:string)

*Схема базы данных* – совокупность схем отношений, без указаний свойств целостности.

*Кортеж* (соответствующий данной схеме отношения) – «строка» таблицы; совокупность упорядоченных троек <имя атрибута – имя домена – значение атрибута>. Значение берётся из домена. Количество этих троек называется степенью отношения.

*Отношение* – совокупность произвольного числа кортежей. Число кортежей в отношении – кардинальное число. Имеем прямоугольную структуру с шириной (степенью отношения) и высоту. В отношении нет одинаковых кортежей; кортежи не упорядочены сверху вниз; атрибуты в кортеже не упорядочены слева направо; каждый кортеж содержит ровно одно значение для каждого атрибута.

*Ключи*: потенциальный – набор атрибутов, обладающих свойством уникальности и неизбыточности (нельзя убрать ни один атрибут из ключа не потеряв уникальности). Как правило один из потенциальных ключей назначается первичным, остальные считаются альтернативными. Также можно определить суперключ: множество атрибутов, подмножеством которого является потенциальный ключ.

*Реляционная база данных* – набор отношений, имена которых совпадают с именами схем отношений.

В **целостной** части фиксируется два базовых понятия – целостность сущностей и ссылочная целостность.

ЦС: не должно быть двух одинаковых кортежей.

СС: если атрибуты внутри отношения как-то связаны, то эта связь определяется функциональными связями. Если же связаны отношения, то выделяют понятие родительского отношения с потенциальным ключом (который может быть первичным) и дочернего отношения с копией потенциального ключа родительского.

Таким образом, если есть связанные отношения и внешний ключ, то он должен ссылаться на существующий кортеж.

Манипуляционная часть: управление базой данных, с помощью языков реляционной алгебры и реляционного счисления. РА замкнута на множестве отношений.

В РА определяется восемь операций над множествами:

1. UNION, INTERSECT, MINUS, TIMES – «традиционные».
2. Выбор (ограничение) , где R – множество, c – логическое выражение. Соединение JOIN - |><| или |><|c (условие). Проекция , где А – элементы, по которым ведётся проекция. Деление DIVIDE BY.
3. Переименование RENAME.

*.* Соединение – проекция выбранного декартового произведения.

|  |
| --- |
| Sn0 |
| 1 |
| 4 |

Пример деления:

A DIVIDE BY B ==

|  |  |
| --- | --- |
| Sn | Pn0 |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |
| 1 | 4 |
| 1 | 5 |
| 1 | 6 |
| 2 | 1 |
| 2 | 2 |
| 3 | 2 |
| 4 | 2 |
| 4 | 4 |
| 4 | 5 |

|  |
| --- |
| Pn0 |
| 2 |
| 4 |

Из А выбираются все такие значения, для которых в Pn0 находятся значения из B.

Множество всех операций , где | логическое или (над множествами работает как объединение).

Унарные – переименование (::= T RENAME A1 AS A2; T::=R|(E) – множество, либо результат работы другого операция), выбор (::=T where C), проекция (::=T|T[A1,A2…An]).

Бинарные – бинарная проекция BinOp (UNION | INTERSECT | JOIN | MINUS | DIVIDE BY | TIMES), E. Интерсект, джойн и дивайд являются нетривиальными – могут быть выражены через другие.

Примеры.

Схема БД:

S { S#, SNAME, STATUS, CITY } PRIMARY KEY { S# }

P { P#, PNAME, COLOR, WEIGHT, CITY } PRIMARY KEY { P# }

J { J#, JNAME, CITY } PRIMARY KEY { J# }

SPJ { S#, P#, J#, QTY } PRIMARY KEY { S#, P#, J# }  
FOREIGN KEY { S# } REFERENCES S  
FOREIGN KEY { P# } REFERENCES P  
FOREIGN KEY { J# } REFERENCES J

А. Получить имена поставщиков, которые поставляют деталь под номером 2

((SPJ JOIN S)WHERE Pno = 2) [Sname]

Б. Получить имена поставщиков, которые поставляют по крайней мере одну красную деталь

((P WHERE Color = ‘Красный’) JOIN SPJ)[Sno] JOIN S)[Sname]

(((P WHERE Color = ‘Красный’)[Pno] JOIN SPJ) JOIN S)[Sname]

18.09.2014------------------------------------------------------------------------

25.09.2014------------------------------------------------------------------------

Манипуляционная часть реляционной модели: реляционная алгебра и реляционная счисление. РА рассматривается в варианте Кодда в интерпретации Дейта.

Возникает вопрос – достаточно ли восьми операторов РА чтобы выразить все пожелания пользователя. Язык РА – декларативный, не содержит процедурных расширений. SQL их содержит – поэтому на сйл можно запрограммировать задачи, нетипичные для РА.

Существуют два варианта дополнительных операторов РА:

1) Расширение: имеется таблица из n атрибутов и m кортежей; её нужно дополнить справа набором новых столбцов, значение каждого вычисляется с помощью скалярного выражения. Если в оригинале было N столбцов и в дополнении M, то становится N+M, но при этом кардинальное число (количество кортежей) не меняется.

2) Суммирование: имеется таблица степени n с кардинальным числом k; справа она дополняется новыми столбцами m, но при этом значение в этих столбцах является итоговыми функциями, которые применяются к столбцам исходной таблицы. Кардинальное число изменится; k1 в доп.столбцах <= K за счет удаления повторений (десять строк с поставщиком S превратятся в одну строку S 10)

Таким образом, расширение – вычисление по горизонтали; суммирование – вычисление по вертикали.

1)

Синтаксис:

EXTEND реляц.выражение ADD скаляр.выражение AS имя.атрибута [, ск.выр AS имя, …]. Скалярное выражение зависит от типов данных: умножение, сложение, и\или, круглые скобки, конкатенация, …

Примеры:  
EXTEND (P JOIN SP) ADD (Weight \* Qty) AS ShipWT  
EXTEND S ADD COUNT**(**(SP RENAME S# AS X) WHERE S# = X**)** AS N – сколько различных видов поставок осуществляет каждый поставщик

2)

Синтаксис:

SUMMARIZE рел.выр PER рел.выр ADD обобщение AS имя

РВ – могут быть именами таблиц. РВ2 должно состоять из тех атрибутов, которые являются подмножеством атрибутов слева. Обобщением является любая функция над атрибутами, в основном COUNT, SUM, MIN, MAX, AVG.  
В качестве РВ2 зачастую будет использоваться проекция.

Примеры:  
SUMMARIZE (P JOIN SP) PER P[City] ADD COUNT AS N – сколько раз каждый город участвует в поставках  
SUMMARIZE SP PER S[S#] AD COUNT AS N – сколько раз каждый поставщик участвовал в поставках

Суммирование является не примитивным выражением – его можно выразить через EXTEND. В жанном случае **(**EXTEND S[S#] ADD *(*(SP RENAME S# AS X) WHERE X = S#*)* AS Y, COUNT (Y) AS N**)**[S#,N]

Помимо этих двух операций можно добавить операции реляционного сравнения <, >, <=, >= в терминах множеств – собственное подмножество, супермножество и т.д..

Реляционное исчисление (вариант кортежей)

Пусть Р(х) – предикат (правильно построенная формула), в состав которого входит переменная Х. === wff; одновременно является булевой функцией.

=== EXISTS x P(x)

=== FORALL x P(x)

NOT (EXISTS x P(x)) === FORALL x (NOT P(x))

NOT (FORALL x P(x)) === EXISTS x (NOT P(x))

EXISTS x P(x) = false OR(i=1..n) (P(x\_i))

FORALL x P(x) = true AND(i=1..n) (P(x\_i))

FORALL x P(x) = NOT (EXISTS x (NOT P(x)))

Объявление кортежной переменной:

RANGE имя-переменной IS список-реляционных-выражений. Все эти выражения должны быть совместимыми по типу – одинаковых заголовков. Таким образом кортежная переменная задаётся на объединении всех РВ задаваемых справа.

Реляционное-выражение ::= (список-целевых-элементов)[WHERE wff]|имя-отношения

Целевой-элемент ::= переменная|переменная.атрибут[AS атрибут]

Wff ::= условие | NOT wff | условие AND wff | условие OR wff | IF условие2 THEN wff | EXISTS переменная(wff) | FORALL переменная(wff) | (wff)

Условие ::= (wff) | компаранд ОПЕРАЦИЯ\_ОТНОШЕНИЯ компаранд ; где компаранд – выражение которое можно сравнивать.

Примеры:

(SX.Sname) WHERE EXISTS SPJX (SPX.Sno=SX.Sno AND SPX.Pno=2) - получить имена поставщиков, которые поставляют деталь 2. Номер поставщика в таблице поставщиков совпадает с номером в таблице поставок – если такие существуют, то получаем имя.

(SX.Sname) WHERE EXISTS SPJX (SPX.Sno=SX.Sno AND EXISTS PX (PX.Pno=SPJX.Pno AND PX.Color=’Red’)) - получить имена поставщиков, которые поставляют по крайней мере одну красную деталь.

(SX.Sname) WHERE FORALL PX (EXISTS SPX (SPX.Sno=SX.Sno AND SPX.Pno=PX.Pno)) - получить имена поставщиков, которые поставляют все типы деталей.

Для каждой детали получить её номер и общее поставляемое количество:

(PX.Pno, SUM (SPX WHERE SPX.Pno=PX.Pno, Qty) AS T)

Получить общее количество поставляемых деталей:

(SUM (SPX, Qty) AS Total)

Для каждого поставщика получить его номер и общее количество осуществленных им поставок:

(SX.Sno, COUNT (SPX WHERE SPX.Sno = SX.Sno) AS N)

02.10.2014------------------------------------------------------------------------

Связь РА и РИ

1. Автор реляционной модели (Кодд) предложил алгоритм редукции Кодда, с помощью которого любое выражение РИ можно преобразовать в последовательность операций РА. Этим он доказал, что РА ПО-КРАЙНЕЙ МЕРЕ мощнее РИ.

В книге Дейта сам алгоритм редукции не приводится, хотя приводится развернутый пример его работы.

2. Ульман доказал, что любое выражение РА можно представить в виде выражения РИ. Этим он доказал, что РИ ПО-КРАЙНЕЙ МЕРЕ мощнее РА.

Следовательно, эти два способа выражения являются эквивалентными.

**3. Теория проектирования Реляционных Баз Данных**

Существуют два подхода к проектированию:

1) снизу вверх (bottom up) – метод синтеза. В рамках предметной области рассматривается набор свойств и атрибутов, и каким-то образом устанавливается связь между этими атрибутами. Ничего кроме этой связи нет; отправной точки являются связи, а сущности нет. Дальше необходимо на основе этого набора связей спроектировать набор отношений (схем отношений), который даст схему базы данных.

На практике этот метод не используется, ввиду трудоемкости формулирования связей в предметной области.

2) сверху вниз (top down) – метод анализа. Имея одно отношение (или набор отношений в виде ЕР-диаграммы), и в пределах отношения мы располагаем набором зависимости между атрибутами. Ставится задача проектирования БД, которая в некотором смысле будет лучше изначальной.

Основным приемом метода является декомпозиция – разбиение исходного отношения на более мелкие. Предполагается, что отношение(схема) претерпевает изменения, которые заключаются в том, что каждое отношение в составе схемы БД принимает ту или иную **нормальную форму:** 1NF, 2NF, 3НФ (НФ Бойса-Кодда, BCNF), 4НФ, 5НФ и другие.

Нормальная форма

1НФ – первоначально определенная БД (см. определение реляционной модели) – домены содержат неделимые данные, каждый атрибут имеет определенное значение. Аномалия обновления – трудности, связанные с обновлением данных, если какое-то отношение находится в 1НФ.

Схема в НФ под номером N тогда и только тогда, когда она находится в НФ N-1 и обладает некоторыми дополнительными свойствами. Новые НФ достигаются путем декомпозиции; оператором декомпозиции является оператор PROJECT. Декомпозиция выполняется чтобы выполнялись обязательное и желательное условия: должна быть без потерь информации и может быть без потери функциональных зависимостей между атрибутами отношения.

Если отношение R декомпозируется (PROJECT) в отношения R1 и R2, а R1 и R2 JOIN в R’, то в R’ должны быть те же функциональные зависимости.

Выделим три типа зависимостей атрибутов:

- функциональные (functional dependencies) – принципиальны для 1NF..BCNF; играют основную роль

- многозначные (MV d) – для 4NF

- зависимости соединения (join d) – для 5NF.

В 1НФ каждый атрибут атомарен. Пусть в отношении имеется набор атрибутов, которые являются потенциальным ключом, и этот набор единственный – одновременно первичный ключ. Тогда каждый неключевой атрибут должен зависеть от первичного ключа – но зависеть он может только от части ключа. Тогда это отношение находится в 1НФ, но не в 2НФ – найдутся неключевые атрибуты, зависящие от части ключа.

Пусть отношение находится в 2НФ – каждый неключевой атрибут ПОЛНОСТЬЮ зависит от ключевого. Если в отношении наблюдаются транзитивные функциональные зависимости, то оно не находится в 3НФ.

Устранив транзитивные зависимости, на этом можно остановиться – теперь все отншения являются вполне удовлетворительными.

НФ Бойса-Кодда: потенциальных ключей несколько (отношение состоит из большого числа атрибутов) и они составные – требуется уточнить понятие функциональной зависимости.

По существу, ФЗ – ограничение, которое должно проверяться при любой попытке обновления (требует время). Если изначально определить набор зависимостей, то можно найти другой набор, который будет более компактный, но эквивалентным заданному – т.е. получить замыкание множества ФЗ.

Аксиомы Армстронга: позволяют получить эти зависимости.  
R(A1…An), FD\_s = {FD1…FDk} – как получить замыкание . Используют эти аксиомы (которые однако не являются алгоритмом) до тех пор, пока ими можно пользоваться.

Closure(X,S) – алгоритм поиска замыкания множества атрибутов X для заданного множества ФЗ S. Алгоритм может ответить на вопрос, будет ли какая-нибудь ФЗ принадлежать замыканию, которое нам неизвестно.

09.10.2014------------------------------------------------------------------------

После нахождения замыкания встает задача – найти множество функциональных зависимостей, которое было бы наименьшим – задача минимального покрытия.

Перевод 1НФ -> 2НФ.

Отношение (конкретное) r над схемой отношений R находится во 2-й нормальной форме тогда и только тогда, когда r находится в 1-й нормальной форме, и каждый неключевой атрибут полностью зависит от ключа (речь идет об атрибутах, которые не входят в состав потенциальных ключей).

Пример: рассмотрим R(A,B,C,D), первичным ключом является пара Pk{A,B}. Дана функциональная зависимость вида FD = (A->D). Предлагается отношение R декомпозировать на два: R1(A,D),Pk1{A} ; R2(A,B,C), Pk2{A,B} и внешним ключом Fk2{A} REF R1(A), ссылающимся на отношение R1.

Перевод 2НФ -> 3НФ.

Онтошение r над схемой R находится в 3-й нормальной форме тогда и только тогда, когда r находится во 2-й нормальной форме, и каждый неключевой атрибут нетранзитивно зависит от ключа.

Пример: схема отношения R(A,B,C), Pk{A}, функциональная зависимость FD(B->C). Таким образом В зависит от А, а С зависит от В – транзитивная зависимость. Производим декомпозицию в R1(B,C), Pk1{B} ; R2(A,B), Pk2{A}, Fk1{B} REF R1(B)

BCNF.

Формулировка НФ Бойса-Кодда исходила из: потенциальных ключей может быть несколько; ключи могут быть составными; пересечение между атрибутами ключей может быть не пусто.

Отношение r над схемой R находится в НФ БК тогда и только тогда, когда каждая нетривиальная и неприводимая слева функциональная зависимость имеет свою левую часть в качестве суперключа отношения R.

Суперключ – подмножество множества атрибутов, которое в качестве подмножества (не обязательно собственного) содержит потенциальный ключ. Потенциальный ключ – неприводимый суперключ.

Аксиомы Армстронга можно использовать при первой попытке получения множества функциональных зависимостей относительно заданного множества.

Аксиома1: правило рефлексивности. Если (-> функциональная зависимость).

Аксиома2: дополнение. , где С – новый атрибут в пределах данной схемы. АС – объединение множеств.

Аксиома3: транзитивность. .

Существует доказательство этих правил из определения функциональных зависимостей.

Функциональная зависимость – А функционально определяет В (В функционально зависит от В) тогда и только тогда, когда для любых двух кортежей T1,T2 отношение r над схемой R из равенства t1[A]=t2[A] следует t1[B]=t2[B].

Набор правил Армстронга является полным – для заданного множества ФЗ минимальный набор может быть выведен только с помощью этих трех правил; является исчерпывающим – в результате применений этих ФЗ никакие дополнительные ФЗ не могут быть получены.

Тем не менее, набор был дополнен.

4: самоопределение. .

5: декомпозиция. .

6: объединение. .

7: композиция. .

8: правило унификации (общая теорема определения). .

Применяя избыточные правила, можно получить замыкание заданного множества функциональных зависимостей.

Задача: для множества отношений задана функциональная зависимость; проверить, будет ли другая ФЗ принадлежать замыканию.

Пример на работу с ^ правилами.

Дано: схема отношения R(A,B,C,D,E,F), множество ФЗ FD\_p = {(A->BC), (B->E), (CD->EF)}. Показать, что для R также выполняется FD=(AD->F) (FD выводима из множества FD\_p).

1. A->BC – дано

2. A->C – из 1 согласно правилу 5) декомпозиции

3. AD->CD – правило 2) дополнения

4. CD->EF – дано

5. AD->EF – правило 3) транзитивности

6. AD->F – правило 5) декомпозиции. Конец примера

Алгоритмы ФЗ.

*Алгоритм 1*. Дано: множество ФЗ S над схемой отношений R; множество атрибутов Х над схемой R. Найти: замыкание множества X относительно множества S, Closure(X,S).

Решение: полагаем что изначально некоторое множество совпадает с Х. Вводим понятия нового и старого множества, и повторяем процесс, пока множества не перестанут меняться.

Jnew = X;

repeat

Jold = Jnew;

for each (X->Y) in S do

if () then Jnew = ;

until (Jold == Jnew)

return Jold;

Упражнение (типа на дом). Дано: R(A..F); S = {(A->BC), (E->CF), (B->E), (CD->EF)}. Найти Closure({A,B},S). Ответ: {A,B,C,E,F}.

Множество ФЗ является неприводимым (минимальным покрытием в канонической форме), если оно обладает тремя свойствами: 1) правая часть каждой ФЗ одноэлементна, 2) левая часть неприводима, 3) нельзя удалить ни одну ФЗ не изменив замыкания.

*Алгоритм 2*. Дано: множество ФЗ S над схемой R, сама схема R.

1. пусть F = S

2. заменить каждую ФЗ вида (X->{A1,A2,...,An}) на n ФЗ вида (X->A1),...,(X->An)

3. для каждой ФЗ из F вида (X->A)

для каждого атрибута B, который является элементом Х

проверять, если {{F-{(X->A)}} {{(X-{B})->A}} эквивалентно F, то

заменить (X->A) на (X-{B})->A

4. для каждой оставшейся ФЗ вида (X->A) из F

если {F-{(X->A)}} эквивалентно F, то

удалить (X->A) из F.

//здесь "-" обозначает вычитание множеств. Замыкание – множество всех ФЗ, выводимых из данного. Множество ФЗ S эквивалентно множеству ФЗ T, если замыкание S == замыканию Т (). Т является покрытием S, если T+ содержит все ФЗ из S. Нетривиальная – зависимость вида {A,B} -> {A}.

16.10.2014------------------------------------------------------------------------

*На 7й неделе – второй модуль, лабы №3-6, рк2. Наиболее важна 5я лаба, написание 25ти запросов, с использованием предикатов (between, like, … семь штук). РК на компах во время лабораторной (вторник). РК2 – дана база данных Family из двух таблиц Person, Marriage (супружеские пары). На лекции – вопрос про функциональные зависимости (а также суперключи и всё остальное), 1-2-3-НФ\_БК. См. Дейта (глава про ФЗ). Другие книги: The Theory of Relational Databases (David Maier) гл.4-8; Fundamentals of Database System (Elmasri, Navatre) гл.15-16часть6.*

Поскольку ^ процесс исключения задается циклом for each, то можно для одного и того же суперключа получить несколько потенциальных ключей, в зависимости от порядка в котором проходил for each.

Неприводимое (минимальное в нормальной форме) покрытие: правая часть всегда одноэлементна; левая часть неприводима; нет лишних зависимостей (минимальное – объединяем зависимости с одинаковой левой частью).

Пример.

Дано: R(A,B,D). ФЗ: E={B->A, D->A, AB->D}. Найти минимальное покрытие.

X -> Y ⬄ детерминант (LeftHandSide) -> зависимая часть (RHS).

Ответ: G:{B->D, A->D}

Тест (*Алгоритм 3*). Проверяет, является ли декомпозиция без потерь информации (NJD) для бинарной декомпозиции. Было отношение R, стало R1, R2.

Дано: БД D(R1, R2) над R. Декомпозиция происходит без потерь тогда и только тогда, когда выполняется одно из двух условий: либо .

*Алгоритм 4*. 3НФ декомпозиция с сохранением ФЗ (DPD).

Дано: универсальная схема отношений R и множество ФЗ F.

1. найти минимальное покрытие (min cover) G для F (по алгоритму2)

2. для каждой левой части Х ФЗ из G создать схему отношений в D с атрибутами , где X->A1, … X->Ak – ФЗ из G, и Х – ключ

3. все оставшиеся атрибуты помещаем в одно единственное отношение в D.

Пример.

Дано: R(1SSN, 2ProjNO, 3EmployeeSalary, 4EmployeePhone, 5DepartamentNO, 6Pname, 7PLocation). 1->5, 1->4, 2->5, 2->6, 2->7, 12->3,4,5,6,7, т.е. G = {SSN->{Esal, Ephone, Dno}, Pno->{Pname, Plocation}}.

D(R1(SSN, Esal, Ephone, Dno), R2(Pno, Pname, Plocation)).

23.10.2014------------------------------------------------------------------------

Пусть имеется отношение R(x,y,z) и есть ФЗ X->Y; декомпозируется в R1(x,y), R2(x,z). По теореме Хита, должно выполняться natural\_join(R1,R2)=R.

Известно, что можно выполнить одновременно декомпозицию без потери информации и без потери ФЗ, но не более чем до 3НФ; до НФБК (в общем случае) уже нельзя.

Пример (алг4).

Дано: U(a,b,c,d,e,f,g). S = {a-cde, b-fg, ab-cdefg}. Множество АБ является первичным ключом – все остальные зависят от этих двух.  
Используя алг2, получаем новый набор ФЗ S = {a-cde, b-fg}, который можно рассматривать как минимальное покрытие; другими словами прошла декомпозиция на r1(a,c,d,e) и r2(b,f,g).

*Алгоритм 5.* Декомпозиция NJD(Non-addiive Join Decomposition) для НФБК.

Дано: универсальное отношение U(или R) и множество ФЗ F.

1. пусть D = {U}

2. while в Д существует отношение, не находящееся в НФБК do

выбрать это отношение Q;

найти ФЗ X->Y, которая нарушает НФБК;

заменить Q в D на

*Алгоритм 6*. Декомпозиция NJD & DPD(Dependency Preservation Decomposition – сохранение зависимостей) для 3НФ.

Дано: универсальное отношение U(или R) и множество ФЗ F.

1. найти минимальное покрытие G для F (алг2)

2. Для каждой левой части Х в ФЗ из G создать отношение в Д с атрибутами {X \cup(i=1..k){Ai}}, где X->Ai являются ФЗ из G (А - одноэлементные правые части ФЗ), Х является ключом отношения

3. Если ни одно из отношений не содержит ключа универсального отношения U, то создать ещё одно отношение, содержащие все атрибуты этого ключа.

Может оказаться, что после добавления этого отношения (если только ключ не содержится в Х->А) какие-то отношения являются "лишними", т.е. являются проекциями других отношений

4. Из результирующей схемы БД D исключить лишние отношения

Пример.

См. пример к алг4 на базе сотрудников. В пункте3 нужно будет добавить ещё одно отношение для первичного ключа: R3(SSN, Pno).

Пример.

Универсальное отношение R, для которого справедливы ФЗ F={P->LCA, LC->AP, A->C}.  
Применяя алг2 (при этом исключая его пункт.3): имеем G = {P->L, P->C, P->A, LC->A, LC->P, A->C}. С пунктом.3 будет GX={P->LC, LC->AP, A->C}  
После пункта.2 алг6 будем иметь R1(P,L,C), R2(L,C,A,P), R3(A,C).  
Пункт.4 алг6: третье является проекцией второго, и первое является проекцией второго; отбрасывая их, оставляем R2(L,C,A,P) находящееся в 3НФ.

В этом примере могло оказаться, что на заключительном этапе поиска минимального покрытия мы бы получили другое множество GY={P->LA, LC->P, A->C}. В этом случае применение пункта.2 алг6 привело бы к R1(P,A,L), R2(L,C,P), R3(A,C). Ни одно из этих отношений не является проекцией другое, и мы бы получили декомпозицию на три отношения в 3НФ, при этом получили бы БД из трёх таблиц.

Между ними всё равно остаются общие атрибуты. Запросы к БД использовали бы JOIN, забирающие временные ресурсы.

Более высокие нормальные формы (4НФ и 5НФ).

1. 4NF и MultiValueDependencies.

Пример – иерархические структуры данных в формате ХМЛ; студенты, преподаватели и учебники.

<hctx>

<course>

<name>physics</name>

<teachers>

<teacher>фейнман</teacher>

<teacher>арнольд</teacher>

</teachers>

<texts>

<text>механика</text>

<text>оптика</text>

</texts>

</course>

<course>

<name>математика</name>

<teachers>

<teacher>арнольд</teacher>

</teachers>

<texts>

<text>механика</text>

<text>анализ</text>

<text>алгебра</text>

</texts>

</course>

</hctx>

Эквивалентно таблице

|  |  |  |
| --- | --- | --- |
| Курс | Тичер | Текст |
| Физ | Фейн | Мех |
| Физ | Фейн | Опт |
| Физ | Арн | Мех |
| Физ | Арн | Опт |
| Мат | Арн | Мех |
| Мат | Арн | Анализ |
| Мат | Арн | Алг |

Таблица эта обладает избыточностью, которая ведёт к аномалиям обновления, но находится в НФБК. Варианты декомпозиции: курс-преподаватель и курс-текст. В первом случае: взять физику ключом нельзя, потому что одной физике будут соответствовать два препода => оба атрибута будут составным ключом.

30.10.2014------------------------------------------------------------------------

Основным языком для микрософтовского Т-СЙЛ является С#. Он позволяет создавать хранимые процедуры, возвращающие скалярные и табличные значения. Создаваемые в шарпе типы данных (де факто – классы) могут являться типами столбцов в СЙЛ таблицах.

Лабораторная: 6 шарповых классов – процедуры + один тип, и протестировать как оно работает.

Определение: пусть R(A1,A2,…An) – схема отношений, Многозначная зависимость (MVD) Х->->У указывает ограничение на любое отношение r над схемой отношения R: .

По лемме Фейгина (Фейджина) имеем, что всегда наблюдается следующее x->>y|z

Многозначная зависимость называется тривиальной, если .

Схема R находится в 4 НФ, если каждая многозначная зависимость в качестве детерминанта имеет суперключ.

Зависимость соединения (Join Dependencies) и 5НФ.

Пример: отношение SPJ из БД dbSPJ без атрибута Qty, SPJ(Sno, Pno, Jno) можно декомпозировать на три: SP, PJ, JS. Также можно декомпозировать на SP и SPJ' в котором будут ранее отсутствующие кортежи, и уже его декомпозировать на PJ, JS.

Зависимость соединения над схемой отношения также определяет ограничение для конкретного экземпляра r: каждое частное отношение над схемой должно неаддитивное соединение частной декомпозиции. Многозначная зависимость является частным случаем ЗС для k=2.

Определение: JD(R1, R2, … Rk) – Natural Join(\*)(R1,R2…Rk)=R должно иметь неаддитивное соединение (новых атрибутов не добавляется).

Рассмотрим JD(R1,R2) – возможны (R1\cupR2) ->>(R1-R2) и (R1\cupR2) ->>(R2-R1).

Находится в 5нф если для каждой нетривиальной зависимости соединения каждое частное отношения Ri является суперключом по отношению к исходному R.

06.11.2014------------------------------------------------------------------------

**Введение в SQL.**

1. краткая историческая справка

1969,70 – доктор Э.Ф. Кодд, реляционная модель данных. 1970 – Кодд предложил язык «Alpha» для работы с РБД. В середине 70х в IBM была создана экспериментальная СУБД System R, работающая на языке SEQueL (structured English queries language); по юридическим соображениям язык был переименован в SQL. Главным образом язык предназначен для формулировки запросов на выборку.

2. стандартизация СЙЛ

Было несколько этапов выработки стандарта. SQL-86 (SQL1), SQL-92 (SQL2), SQL:1999 (SQL3), SQL:2003,2006,2008,2011. Стандартом является ISO/IEC 9075 “database language SQL”, он же сйл2.

Структура стандарта: 1.SQL/Framework, 2.SQL/Foundation(Core), 3.SQL/CallLanguageInterface(подобие ODBC 3), 4.SQL/PersistentModuleStorage, 9.SQL/MED, 10.SQL/OLB, 11.SQL/Schema, 13.SQL/JavaRunTime, 14.SQL/XML.

3. SQL для MS SQL Server = Transact-SQL (T-SQL)

Подход Top-Down.

Основные конструкции Т-СЙЛ – инструкции (statements). Из них составляется сценарий, делящийся на пакеты (через GO). Инструкции классифицируются:

a) DDL: CREATE / ALTER / DROP (54/49/52 вариантов – креат датабейз, альтер тейбл, …)

b) DML: SELECT / UPDATE / DELETE / INSERT / MERGE. При этом балк инсерт сюда относится «боком» – он не фиксируется в журнале транзакций и на него не срабатывают триггеры.

c) DCL (инструкции безопасности): GRANT / REVOLVE / DENY. Грант – предоставляет разрешение выполнять операции над таблицами. Револв – отзывает разрешение, денай – запрещает.

d) TCL (инструкции управления транзакциями): BEGIN DISTRIBUTED TRANSACTION, BEGIN TRANSACTION, COMMIT TRANSACTION, ROLLBACK TRANSACTION, SAVE TRANSACTION (устанавливает точку сохранения внутри транзакции – откат до этой точки, а не до начала).

e) CFL (инструкции управления потоком): BEGIN / END, BREAK, CONTINUE, GOTO, IF .. ELSE .., WHILE, TRY .. CATCH .., PRINT, EXECUTE, в 14й версии THROW.

f) инструкции курсоров: DECLARE CURSOR, OPEN, FETCH, CLOSE, DEALLOCATE

g) инструкции BACKUP, RESTORE

h) команды управления

i) инструкции SET

Система типов данных SQL

а) в стандарте СЙЛ: точные числа, приближенные числа, символьные и битовые строки, дата и время, интервалы, булевские. Также в зависимости от производителя: коллекции (массивы, множества, …), анонимные строчные (row) типы, пользовательские (UserDefinedTypes)

б) в Т-СЙЛ: точные, приблизительные, символьные, … . Булевский тип не реализован, но есть DECLARE @var bit = ‘TRUE’ или ‘FALSE’

Подход Down-Top.

Алфавит: юникод.

Лексика: идентификаторы, ключевые слова, константы (литералы в СЙЛ), операторы, символы пунктуации. Максимальная длина идентификатора: 128 символов, переменной: 116. Идентификаторы могут начинаться с буквы, \_, @, #; дальше могут идти буквы, цифры, \_ @ # $.

Большинство ссылок на объекты используют трехкомпонентные имена: таблица, схема, столбец. Кроме идентификаторов используются ключевые слова (зарезервировано 185), КС для работы с ODBC-драйверами (235)и зарезервированные на будущее(273).

13.11.2014------------------------------------------------------------------------

…

20.11.2014------------------------------------------------------------------------

…

CREATE FUNCTION fnFullSPJ()

RETURNS TABLE

AS

RETURN( SELECT S.sname, P.pname, J.jname, SPJ.qty

FROM S JOIN SPJ ON S.sno=SPJ.sno JOIN P

ON P.pno = SPJ.pno JOIN J ON J.jno=SPJ.jno)

GO

Происходит слишком много затрат на вызов процедуры – сохранение параметров в стеке, передача параметров, сохранение адреса возврата и так далее, при этом в функцию не передаётся никаких параметров, а возвращает она («статическое») табличное значение. В данном случае лучше использовать CREATE VIEW vwFullSPJ AS SELECT … GO.

При использовании многотабличного представления, возникает проблема обновления данных в исходных таблицах. В этом случае можно использовать триггер INSTEAD OF.

Триггер – процедурное средство обеспечение целостности БД. Используются если нужны проверки не булевого типа (CHECK поддерживает только булевые). Триггер не вызывается, а срабатывает; ничего не возвращает. Взаимодействует с двумя псевдотаблицами – inserted, deleted; некоторое продолжение журнала транзакций. В инсертед – вставляемые строки, в делетед – удаляемые; обновляемые строки – одна в делет, другая в инсерт. Внутри триггера можно использовать UPDATE (возвращает истину, если в столбце происходило обновление) и COLUMN UPDATED (определяет по битовой маске, в которых происходили изменения; проверяет не более 8 столбцов за раз).

Внутри триггера можно записать блог BEGIN TRY END TRY, BEGIN CATCH END CATCH; RAISE ERROR; ROLLBACK TRANSACTION (действует на весь пакет – нужно учитывать, что происходило до вызова триггера).

Пример первый – триггер AFTER. После обновления в базовой таблице, будет обновлять таблицу аудита. Работаем в БД TempDB.

use tempdb

go

create schema hr --human resource

go

create table hr.Employee(

EmpID char(6) not null primary key,

Fname char(20) not null,

Lname char(20) not null,

Salary decimal(12,2) not null

)

create table hr.EmployeeAudit(

EmpID char(6) not null,

/\*\*\*/ DateChange datetime2 not null default(sysdatetime()),

Fname char(20) not null,

Lname char(20) not null,

Salary decimal(12,2) not null,

/\*\*\*/ Action char(6) not null check( Action in ('delete', 'update') ),

/\*\*\*/ ChangedByUserName sysname not null default(original\_login()), --имя пользователя подключившегося к серверу

constraint pk\_EA primary key (EmpID, DateChange)

)

create trigger hr.Employee$UpdateAndDeleteAuditTrigger --DML триггер привязывается к таблице; просмотреть можно в менеджере студии

on hr.Employee

after update, delete --условия срабатывания

as begin

declare @rowsAffected int=(select count(\*) from deleted); --@ - локальная переменная

if @rowsAffected=0 return

begin try --пытаемся определить значение поля экшн

declare @action char(6)

set @action = case when (select count(\*) from inserted)>0 then 'update' else 'delete' end --end к кейзу

insert /\*hr. не пишем, потомучто триггер внутри схемы\*/ EmployeeAudit(EmpID, Fname, Lname, Salary, @action)

select (EmpID, Fname, Lname, Salary, @action) from deleted

end try

begin catch

if @@TRANcount > 0 --можно throw

rollback tran

end catch

end

go

update hr.Employee set Salary=Salary\*1.1 where EmpID='1' --триггер сработает; добавит в аудит строку с экшн=делет

Пример второй – INSTEAD OF, обработка строк (имя, фамилия). Приводит строки к виду <Загл><строч><строч>...<строч> <Загл><строч>...<строч>

Триггер при вставках будет вставлять вместо вводимых строк – преобразованные.

use tempdb

go

create table student(...)

go

create function dbo.fnTitleCase(@inputString varchar(2000))

returns varchar(2000)

as

... --в строке из нескольких слов каждое слово теперь начинается с заглавной буквы и продолжается строчными

go

create trigger student$InsteadOfInsertTrigger

on student

instead of insert

as begin

declare @rowsAffected int=(select count(\*) from inserted)

begin try

insert into student(studentIDname, Fname, Lname, rowCreateDate, rowCreateUser)

select studentIDname, dbo.fnTitleCase(Fname), dbo.fnTitleCase(Lname), current\_timestamp, original\_login() from inserted

end try

begin catch

if @@trancount>0

rollback tran

end catch

end

go

27.11.2014------------------------------------------------------------------------

Внутреннее устройство СЙЛ сервера

I. Данные

1) база данных

2) файлы (аналог устройств в старых ОС)

В один файл нельзя записать больше одной БД. В минимальной конфигурации содержит два файла: .mdf (файл данных), .ldf (журнал)

3) Экстенты (разделяемые – страницы разных типов; однородные – страницы одного типа)

4) Страницы (8192 байта) – единицы памяти, выделяемые внутри экстента.

В СЙЛ существуют страницы данных, страницы индексов, BLOB-страницы.

5)

04.12.2014------------------------------------------------------------------------

In Memory OLTP база данных – БД, которая полностью загружается в оперативную память (до 128 ГБ)

При выполнении запросов, оптимизатор строит план выполнения запросов – некое подобие объектного кода. Процессор запросов (главная часть СУБД) строит план оптимальным образом с учетом инедксации и накопленной ранее статистики по выполнению запросов.

Индексация T-SQL.

CREATE / DROP / ALTER . Если указать поле автоинкрементным, оно автоматически становится (кластерным) индексом. Это повышает производительность работы; но если по этому полю не происходит поиск (не используется в WHERE), то получается лишний перерасход памяти.

Индексы используются для повышения производительности. В СЙЛ сервере хранятся в виде сбалансированного дерева; узлами и листьями являются страницы.

Индексы делятся на кластерные и некластерные. Кластерный – в листовых узлах находятся сами строки данных. В некластерном - 16байтные указатели на данные, находящиеся в куче. Если некластерный индекс построен на основе кластерного - то указатели на корень кластерного индекса, если тот существует.

Рассмотрим пример. Одна страница имеет размер 8 Кбайт (8192 байта).

CREATE TABLE (Col char(60)); требуется построить для неё индекс.

Пусть в таблице 100 строк; для хранения потребуется 6000 байт, < 8192. Пока все строки будут храниться в одной таблице, то индекс будет состоять только из одной страницы; максимум так можно хранить 134 строки; данные хранятся в листьях.

+135'я строка – влечет добавление двух страниц (чтобы получилось дерево).

Для 17957 строк индекс будет состоять из 134 страниц листового уровня и корневой страницы

+17958'я строка: СЙЛ сервер добавляет промежуточный уровень, содержащий 2 страницы (дерево станет глубины 3).

+~2млн'я строка: дерево станет глубины 4.

Журнал транзакций используется для восстановления (откатов) отдельных транзакций; восстановления всех незавершенных транзакций при запуске СЙЛ сервера.

Логическая архитектура журнала транзакций.

ЖТ – последовательность записей: а) log sequence number; б) prev LSN; в) Translation ID number; г) тип записи в лог-файле; д) прочее. Основные типы: Translation Log Operation Code (Код выполненной логической операции) и Update Log Record (исходный и результирующий образ изменённых данных).

Физическая структура ЖТ.

Каждый физический журнал разбивается на несколько виртуальных. Количество виртуальных журналов и их размер определяется системой; программист в это вмешаться не может. Если исчерпывается дисковое пространство для физических журналов, администратор СУБД может выделить в конфигурационном файле дополнительную память.

DBCC Log(имя БД, тип вывода) – функция доступа к журналу транзакций. Тип вывода – число 0..4, указывает насколько полную информацию о ЖТ нужно получить.

Fn\_dblog() – «недокументированная» функция доступа к ЖТ.

Транзакция – «единица работы» СЙЛ сервера, состоящая из набора инструкций языка. Граница транзакции определяется или явно через BEGIN \ ROLLBACK \ COMMIT, либо неявно, когда бегин отсутствует – транзакция начинается сразу же после выполнения предыдущей (SET IMPLICIT\_TRANSACTION ON – действует на одно соединение). Также существует режим, в котором каждая команда СЙЛ является отдельной транзакцией.

11.12.2014------------------------------------------------------------------------

Инструкции SET связанные с управлением транзакциями.

SET LOC\_TIMEOUT <число> //по умолчанию время ожидания бесконечно; -1 – также бесконечность

SET IMPLICIT\_TRANSACTION ON/OFF //по умолчанию on

SET REMOTE\_PROC\_TRANSACTION ON/OFF

SET XACT\_ABORT ON/OFF //по умолчанию офф

SET TRANSACTION ISOLATION LEVEL {READ COMMITED (по умолчанию) / READ UNCOMMITED / REPEATABLE READ / SERIALIZABLE }

Журнал транзакций используется при сбоях оборудования и т.п. По какой-то копии ЖТ происходит восстановление ЖТ, после чего происходит накат – изменения, зафиксированные на диске, возвращаются назад; затем происходит откат – БД возвращается в исходное состояние, если транзакция была начата но не завершилась.

Пример транзакции на dbSPJ.

BEGIN TRAN

INSERT INTO S VALUES(N'qwe',100,N'asd');

INSERT INTO S VALUES(N'zxc',100000,N'rty'); --статус smallint, переполнение

--UPDATE S SET Sno=2 WHERE Sno=1; --это ограничение окажется более приоритетно чем ^

COMMIT TRAN

СЙЛ не производит отката транзакции, если инструкция внутри оной привела к оной. Для автоматически отката необходимо установить SET XACT\_ABORT ON.

25.12.2014------------------------------------------------------------------------

Пост-реляционные базы данных

NoSQL – Not Only SQL базы данных. MangDB, RaptorDB и т.д.. В них нет реляционных таблиц; внутренняя форма представления – JSON, более экономичная чем XML.

ODMG – выпустили манифест реляционных баз данных. Кроме них - Манифест объектно-реляционных СУБД. Третий – Дейт и Дарвин, «назад к реляционным БД».

ООБД – информация представлена в виде объектов.