ОСОБЕННОСТИ СОЗДАНИЯ ТАБЛИЦ В SQL SERVER, MYSQL и ORACLE

САЖИН Юлиана
Coordonator: САРАНЧУК Дориан

Технический Университет Молдовы

***Аннотация**: В данной статье рассматриваются вопросы создания таблиц базы данных: именование столбцов и определение типов данных. Приводится обзор правил и рекомендаций для создания таблиц. Также даются некоторые полезные советы, которые в будущем облегчат работу с базой данных. Рассматриваются типы данных, определенные стандартом SQL и типы данных определенные в СУБД. Так же рассматриваются некоторые особенности различных СУБД: SQL Server, MySQL и Oracle. От того насколько правильно создана таблицы и ограничения зависит корректность и эффективность работы всей базы данных.*

***Ключевые слова:** база данных, таблица, атрибут, тип данных.*

1. Введение

После создания базы данных, следующим шагом должно стать создание таблиц. Таблицы являются объектами, в которых организовываются и хранятся данные. Таблица базы данных хранит информацию об одной однородной группе объектов. Например, таблица может хранить справочник адресов, список магазинов, телефонный справочник или прайс-лист. База данных может содержать одну или несколько таблиц. Каждая таблица состоит из строк и столбцов. Данные об объекте, содержащиеся в одной строке, называются записью. Каждая запись состоит из одного или более полей (значений свойств объекта), а каждое поле имеет свой тип.

При проектировании таблиц базы данных необходимо принять некоторые решения, относящиеся к их структуре. К этим решениям относится определение того, какие элементы данных должны храниться в этих таблицах и как таблицы будут связаны друг с другом. Для того чтобы правильно реализовать поставленную задачу, необходимо ответить на следующие вопросы: Какие данные будут храниться в каждой из таблиц? Какие поля будут созданы для хранения данных? Какие будут требования к диапазону данных, хранящихся в колонках (какие данные разрешено в них хранить), и какие типы данных СУБД должны применяться для каждой из колонок?

2. Создание таблиц

Таблицы создаются командой CREATE TABLE. Эта команда создает пустую таблицу. Команда CREATE TABLE определяет имя таблицы и описание набора имён столбцов, указанных в определенном порядке. Она также определяет типы данных и размеры столбцов. Каждая таблица должна иметь, по крайней мере, один столбец.

Простейший синтаксис команды CREATE TABLE, определенный стандартом SQL:

CREATE TABLE table-name
(<column name> <data type>[(size)],
<column name> <data type> [(size)] ...);

При создании таблиц следует учитывать специфику конкретной СУБД. Например, при создании таблиц в MySQL следует явно указывать тип создаваемых таблиц. В MySQL существуют несколько типов таблиц, которые определяют набор допустимых действия с таблицами базы данных, а также влияют на характеристики таблиц. Так, например, тип InnoDB поддерживает транзакции и внешние ключи. Формат данных InnoDB обеспечивает надежное хранение данных за счет транзакционности и блокировки данных на уровне строки. Таблицы типа MyISAM чаще всего используются для веб-приложений и в других средах, где преобладают запросы на чтение. Таблицы типа MyISAM показывают хорошие результаты при выборках SELECT. Во многом это связано с отсутствием поддержки транзакций и внешних ключей. Однако при модификации и добавлении записей вся таблица кратковременно блокируется, это может привести к серьёзным задержкам при большой загрузке. Тип хранения данных MEMORY создает таблицу с данными в памяти. Так как данные хранятся в памяти, то данный тип таблиц обладает большим быстродействием, но при этом велика

вероятность потери данных связанных с возможными сбоями в работе сервера (при перезагрузке сервера все данные для таблиц с типом MEMORY пропадают). Исходя из вышесказанного данный тип таблиц идеально подходит для временного хранения данных.

При создании нужно использовать понятные и значимые имена для таблиц и полей. Лучше именовать таблицы в единственном числе (т.е. Group вместо Groups). Таблица и так представляет коллекцию сущностей, нет никакой необходимости во множественном числе. Не следует использовать символ пробела в именах таблиц или полей, иначе придется использовать специальные знаки (например,‘{‘, ‘[‘, ‘“‘) для обозначения таблиц или полей. К примеру, для доступа к таблице Order Description придется писать [Order Description] (обозначение в SQL Server) или “Order Description ”.

Современные СУБД допускают записывать имена полей и таблиц, используя кириллицу. Однако стоит избегать подобных имен. Во-первых, обязательно придется использовать специальные символы для обозначения таких таблиц или полей, а во-вторых, очень неудобно писать скрипты и запросы – будет необходимо постоянно переключать раскладку клавиатуры.

3. Типы данных

В языке SQL имеются средства, позволяющие для каждого атрибута указывать тип данных, которому должны соответствовать все значения этого атрибута. Следует отметить, что определение типов данных является той частью, в которой коммерческие реализации языка не полностью согласуются с требованиями официального стандарта SQL. Это объясняется, в частности, желанием обеспечить совместимость SQL с другими языками программирования.

3.1 Типы данных определенные стандартом SQL

Символьные данные состоят из последовательности символов, входящих в определенный создателями СУБД набор символов. Поскольку наборы символов являются специфическими для различных диалектов языка SQL, перечень символов, которые могут входить в состав значений данных символьного типа, также зависит от конкретной реализации. Чаще всего используются наборы символов ASCII и EBCDIC. Для определения данных символьного типа используется следующий формат:

<символьный_тип>::={ CHARACTER [VARYING][длина] | [CHAR | VARCHAR][длина]}.

При определении столбца с символьным типом данных параметр длина применяется для указания максимального количества символов, которые могут быть помещены в данный столбец (по умолчанию принимается значение 1). Символьная строка может быть определена как имеющая фиксированную или переменную (VARYING) длину. Если строка определена как фиксированное количество символов, то при вводе в нее меньшего количества символов она дополняется до указанной длины пробелами, добавляемыми справа. Если строка определена с переменной длиной значений, то при вводе в нее меньшего количества символов в базе данных будут сохранены только введенные символы, что позволит достичь определенной экономии внешней памяти.

Битовый тип данных используется для определения битовых строк, т.е. последовательности двоичных цифр (битов), каждая из которых может иметь значение либо 0, либо 1 . Данные битового типа определяются при помощи следующего формата:

<битовый_тип>::=BIT [VARYING][длина].

Тип точных числовых данных применяется для определения чисел, которые имеют точное представление, т.е. числа состоят из цифр, необязательной десятичной точки и необязательного символа знака. Данные точного числового типа определяются точностью и длиной дробной части. Точность задает общее количество значащих десятичных цифр числа, в которое входит длина как целой части, так и дробной, но без учета самой десятичной точки. Масштаб указывает количество дробных десятичных разрядов числа:

<фиксированный_тип>::={NUMERIC[точность[,масштаб]]|{DECIMAL|DEC}[точность [,масштаб]]| {INTEGER |INT}| SMALLINT}.

Типы NUMERIC и DECIMAL предназначены для хранения чисел в десятичном формате. По умолчанию длина дробной части равна нулю, а принимаемая по умолчанию точность зависит от реализации. Тип INTEGER (INT) используется для хранения больших положительных или отрицательных целых чисел. Тип SMALLINT – для хранения небольших положительных или отрицательных целых чисел; в этом случае расход внешней памяти существенно сокращается.

Тип округленных чисел применяется для описания данных, которые нельзя точно представить в компьютере, в частности действительных чисел. Округленные числа (или числа с плавающей

запятой) представляются в научной нотации, при которой число записывается с помощью мантиссы, умноженной на определенную степень десяти (порядок), например: 10E3, +5.2E6, -0.2E-4. Для определения данных вещественного типа используется формат:

<вещественный_тип>::={ FLOAT [точность]| REAL |DOUBLE PRECISION}.

Параметр точность задает количество значащих цифр мантиссы. Точность типов REAL и DOUBLE PRECISION зависит от конкретной реализации.

Тип данных "дата/время" используется для определения моментов времени с некоторой установленной точностью. Стандарт SQL поддерживает следующий формат:

<тип_даты/времени>::= {DATE | TIME[точность][WITH TIME ZONE]| TIMESTAMP [точность] [WITH TIME ZONE]}.

Тип данных DATE используется для хранения календарных дат, включающих поля YEAR (год), MONTH (месяц) и DAY (день). Тип данных TIME предназначен для хранения отметок времени, включающих поля HOUR (часы), MINUTE (минуты) и SECOND (секунды). Тип данных TIMESTAMP используется для совместного хранения даты и времени. Параметр точность задает количество дробных десятичных знаков, определяющих точность сохранения значения в поле SECOND. Тип данных INTERVAL используется для представления периодов времени.

3.2 Типы данных определенные стандартом СУБД

Каждая конкретная реализация СУБД дополняет и расширяет типы данных, определённые стандартом SQL. Например, в SQL Server были добавлены следующие типы данных:

- *CURSOR* — специальный тип данных, используемый для описания курсора в форме переменной или параметра хранимой процедуры OUTPUT. Тип нельзя использовать в инструкции CREATE TABLE. Тип CURSOR может принимать значение NULL.
- *IMAGE* — хранит двоичное значение переменной длины до 2 147 483 647 байт. Этот тип данных часто используется для хранения графики, звука и файлов (таких как документы MS Word и электронные таблицы MS Excel). Значениями типа IMAGE нельзя свободно манипулировать. Столбцы типа IMAGE и TEXT имеют множество ограничений на способы использования.
- *MONEY* — хранит денежные значения в диапазоне от -922337203685477.5808 до 922337203685477.5807. Значение занимает 8 байт.
- *TEXT* — хранит очень большие фрагменты текста длиной до 2 147 483 647 символов. Значениями типа TEXT и IMAGE часто гораздо труднее манипулировать, чем, скажем, значениями типа VARCHAR. Например, нельзя создавать индекс по столбцу типа TEXT или IMAGE.
- *UNIQUEIDENTIFIER* — представляет собой значение, уникальное для всех баз данных и всех серверов. Представлено в виде xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx, в котором каждый «x» представляет собой шестнадцатеричное число в диапазоне 0-9 или a — f. Единственными операциями, которые можно производить над значениями этого типа, являются сравнение и проверка на NULL. В столбцах этого типа можно использовать ограничения и свойства, за исключением свойства IDENTITY.

В Oracle были добавлены следующие типы данных:

- *RAW* — используется для хранения двоичных данных до 2000 байт.
- *LONG* — используется для хранения текстовых данных длиной до 2 ГБ.
- *LONG RAW* — используется для хранения двоичных данных до 2 ГБ.
- *ROWID* — используется для хранения идентификаторов ROWID базы данных Oracle в специальном формате (адреса строк таблицы).
- *BLOB* — сохраняется до 4 ГБ двоичных данных. Данные этого типа хранятся вне таблицы, а в таблице Oracle находятся лишь указатели на объекты.
- *CLOB, NCLOB* — сохраняется до 4 ГБ текстовых данных. *NCLOB* – это тип данных *NLS* большой фиксированной длины (*NLS* означает *National Language Set*) и используется для работы в Oracle на языках, отличных от английского. В английском для хранения одного символа нужен 1 байт, а в некоторых языках мира с наборами больших символов (японском, китайском, корейском), а так же на языках, где текст читается справа налево (арабский, иврит) для хранения одного символа требуется несколько байт. Данные этого типа хранятся вне таблицы, а в таблице находятся лишь указатели на объекты.

– *BFILE* — сохраняется до 4 ГБ неструктурированных данных, причем в файлах операционной системы (внешние файлы).

MySQL поддерживает несколько типов столбцов, которые можно разделить на три категории: числовые типы данных, типы данных для хранения даты и времени и символьные (строковые) типы данных.

При создании таблиц желательно использовать целочисленные типы меньшей длины, чтобы получить таблицы меньшего размера. К примеру, в 90% случаев программисты используют для целочисленных значений тип *INT* (диапазон со знаком от -2147483648 до 2147483647, диапазон без знака от 0 до 4294967295). Однако, маловероятно что в таблице будет 4.2 миллиарда записей. Скорее всего, достаточно использования:

– *MEDIUMINT* -8388608 до +8388607. UNSIGNED - знак не учитывается. Диапазон без знака от 0 до 16777215.
– *SMALLINT* -32768 до +32767. Диапазон без знака от 0 до 65535.
– *TINYINT* от -128 до +127. Диапазон без знака от 0 до 255.

Рекомендуется использовать минимально возможные типы данных. Чем больше тип данных, тем объемнее таблица, и тем больше обращений к дискам нужно для получения данных.

Часто приходится сталкиваться с такой проблемой, как точное представление денежных величин. В MySQL для представления таких величин необходимо использовать тип данных *DECIMAL*. Поскольку данные этого типа хранятся в виде строки, потерь в точности не происходит. А в случаях, когда точность не имеет слишком большого значения, вполне подойдет и тип данных DOUBLE.

Используйте тип bit для хранения логических (boolean) полей. Использование integer и varchar затрачивает гораздо больше ресурсов

Поля, содержащие бинарные данные, не должны объявляться в часто используемых таблицах из-за проблем с производительностью. Эти данные необходимо расположить в другой таблице. А их указатели могут быть использованы в часто используемых таблицах. Лучше всего вообще не хранить бинарные данные в таблицах. Это связано с ресурсоемкостью, производительностью и безопасностью базы данных. Изображения, файлы и прочее следует хранить в отдельных папках, а в таблице только путь к ним.

Заключение

Перед проектированием таблиц следует четко определить, какие цели и задачи будет выполнять проектируемая система, чтобы осознать общую конструкцию таблиц вашей базы данных, прежде чем начать создавать их. Нужно также знать, каким образом будет осуществляться доступ к данным. Учесть, какие запросы будут выполняться чаще всего и из каких колонок будут извлекаться данные. Определите, какая информация действительно необходима для базы данных, а какую хранить не надо. Ответы на эти вопросы помогут вам принять решения о том, как создавать таблицы и индексы, какие ограничения могут применяться, какие значения по умолчанию могут оказаться полезными.

Библиография

1. *Библиотека TechNet. Типы данных Transact-SQL*. URL: http://technet.microsoft.com/ru/library/ms187752%28v=sql.105%29.aspx (дата обращения: 20.11.13).
2. *Oracle Database Documentation Library. 26 Oracle Data Types*. URL: http://docs.oracle.com/cd/B28359_01/server.111/b28318/datatype.htm#CNCPT012 (дата обращения 20.11.2013).
3. Юрий Шиканов. *MySQL - обзор типов таблиц.* URL: http://dizballanze.com/%D0%91%D0%94/mysql-obzor-tipov-tablits/ (дата обращения 21.11.2013)
4. *SQL в примерах и задачах. Учеб. пособие* / И.Ф. Астахова, А.П. Толстобров, В.М. Мельников.— Мн.: Новое знание, 2002. — 176 с.