Описание данных на основе SQL

В организации базы данных задействуется большое число различных объектов. Все объекты базы данных являются либо физическими, либо логическими. *Физические объекты* связаны с организацией данных на физических устройствах (дисках). Физическими объектами являются файлы и файловые группы. *Логические объекты* являются пользовательскими представлениями базы данных. В качестве примера логических объектов можно назвать таблицы, столбцы и представления (виртуальные таблицы).

Объектом базы данных, который требуется создать в первую очередь, является сама база данных. SQL Server управляет как системными, так и пользовательскими базами данных. Пользовательские базы данных могут создаваться авторизованными пользователями, тогда как системные базы данных создаются при установке системы базы данных.

Создание базы данных

Для создания базы данных используется два основных метода. В первом методе задействуется обозреватель объектов среды SQL Server Management Studio, работа с которым рассмотрена в соответствующею лабораторной работе, а во втором применяется инструкция языка Transact-SQL CREATE DATABASE.

Ниже представлена общая форма этой инструкции, а затем в таблице подробно рассмотрены ее составляющие:

CREATE DATABASE db_name
[ON [PRIMARY] {file_spec1},.]
[LOG ON {file_spec2},.]
[COLLATE collation_name]
[FOR {ATTACH ATTACH_REBUILD_LOG}]

Параметр	Описание
db_name	имя базы данных (может содержать максимум 128 символов)
ON	указывает на то, что все файлы, в которых хранится база данных, будут указаны явно. В случае отсутствия параметра все файлы будут предоставлены неявно системой
file_spec1	спецификация файла, параметр может также содержать дополнительные опции: логическое имя файла, физическое имя и размер
PRIMARY	указывает первый (и наиболее важный) файл, который содержит системные таблицы и другую важную внутреннюю информацию о базе данных. Если параметр PRIMARY отсутствует, то в качестве первичного файла используется первый файл, указанный в спецификации
LOG ON	эта опция параметра dbo определяет один или более файлов в качестве физического хранилища журнала транзакций базы данных. Если эта опция отсутствует, то журнал транзакций базы данных все равно будет создан, поскольку каждая база данных должна иметь, по крайней мере, один журнал транзакций. См. замечание 1.
COLLATE	указывается порядок сортировки по умолчанию для базы данных. Если эта опция не указана, базе данных присваивается порядок сортировки по умолчанию базы данных model, такой же, как и порядок сортировки по умолчанию системы баз данных.

Замечание 1. Учетная запись SQL Server, применяемая для создания базы данных, называется *владельцем базы данных*. База данных может иметь только одного владельца,

который всегда соответствует учетной записи. Учетная запись, принадлежащая владельцу базы данных, имеет специальное имя **dbo**. Это имя всегда используется в отношении базы данных, которой владеет пользователь.

Присоединение и отсоединение баз данных

Все данные базы данных можно отсоединить, а потом снова присоединить к этому же или другому серверу базы данных. Эта функциональность используется при перемещении базы данных.

Для отсоединения базы данных от сервера баз используется системная процедура sp_detach_db, для присоединения – системная процедура sp_attach_db.

Инструкция CREATE TABLE

Инструкция CREATE TABLE создает новую таблицу базы данных со всеми соответствующими столбцами требуемого типа данных. Ниже представлена общая форма этой инструкции, а затем в таблице подробно рассмотрены ее составляющие:

CREATE TABLE table_name
(col_name1 type1 [NOT NULL | NULL)
[{, col_name2 type2 [NOT NULL | NULL]} ...])

Параметр	Описание
table_name	имя таблицы
col_name1, col_name2	имена столбцов таблицы
type1, type2	типы данных соответствующих столбцов таблицы
NOT NULL | NULL	если для столбца не указано, что значения NULL разрешены (NOT NULL), то данный столбец не может содержать значения NULL, и при попытке вставить такое значение система возвратит сообщение об ошибке

Кроме типа данных и свойства содержать значения NULL, в спецификации столбца можно указать следующие параметры:

- ✓ предложение **DEFAULT**: указывает значение столбца по умолчанию, т. е. когда в таблицу вставляется новая строка, ячейка этого столбца будет содержать указанное значение, которое останется в ячейке, если в нее не будет введено другое значение (здесь можно использовать константу, например, системные функции).
- ✓ свойство **IDENTITY**: задает столбец идентификаторов, который может иметь только целочисленные значения, которые системой присваиваются обычно неявно. Каждое следующее значение, вставляемое в такой столбец, вычисляется, увеличивая последнее, вставленное в этот столбец, значение (т.е. содержит явно или неявно начальное значение и шаг).

Ограничения декларативной целостности в инструкции CREATE TABLE

Одной из самых важных особенностей, которую должна предоставлять СУБД, является способ обеспечения целостности данных. Ограничения, которые используются для проверки данных при их модификации или вставке, называются *ограничениями для обеспечения целостности (integrity constraints)*. Обеспечение целостности данных может осуществляться пользователем в прикладной программе или же системой управления базами данных. Наиболее важными преимуществами предоставления ограничений целостности системой управления базами данных являются следующие:

- повышается надежность данных;
- сокращается время на программирование;
- упрощается техническое обслуживание.

Определение ограничений для обеспечения целостности посредством СУБД повышает надежность данных, поскольку устраняется возможность, что программист прикладного приложения может забыть реализовать их. Если ограничения целостности предоставляются прикладными программами, то все приложения, затрагиваемые этими ограничениями, должны содержать соответствующий код. Если код отсутствует хоть в одном приложении, то целостность данных будет поставлена под сомнение.

Если ограничения для обеспечения целостности не предоставляются системой управления базами данных, то их необходимо определить в каждой программе приложения, которая использует данные, включенные в это ограничение. В противоположность этому, если ограничения для обеспечения целостности предоставляются системой управления базами данных, то их требуется определить только один раз. Кроме этого, код для ограничений, предоставляемых приложениями, обычно более сложный, чем в случае таких же ограничений, предоставляемых СУБД.

Если ограничения для обеспечения целостности предоставляются СУБД, то в случае изменений ограничений, соответствующие изменения в коде необходимо реализовать только один раз — в системе управления базами данных. А если ограничения предоставляются приложениями, то модификацию для отражения изменений в ограничениях необходимо выполнить в каждом из этих приложений.

Системами управления базами данных предоставляются два типа ограничений для обеспечения целостности:

– декларативные ограничения для обеспечения целостности;

– процедурные ограничения для обеспечения целостности, реализуемые посредством триггеров.

Декларативные ограничения определяются с помощью инструкций языка DDL CREATE TABLE и ALTER TABLE. Эти ограничения могут быть уровня столбцов или уровня таблицы. Ограничения уровня столбцов определяются наряду с типом данных и другими свойствами столбца в объявлении столбца, тогда как ограничения уровня таблицы всегда определяются в конце инструкции CREATE TABLE или ALTER TABLE после определения всех столбцов.

Замечание 2. Между ограничениями уровня столбцов и ограничениями уровня таблицы есть лишь одно различие: ограничения уровня столбцов можно применять только к одному столбцу, в то время как ограничения уровня таблицы могут охватывать больше, чем один столбец таблицы.

Каждому декларативному ограничению присваивается имя. Это имя может быть присвоено явно посредством использования опции CONSTRAINT в инструкции CREATE TABLE или ALTER TABLE. Если опция CONSTRAINT не указывается, то имя ограничению присваивается неявно SQL Server.

Замечание 3. Настоятельно рекомендуется использовать явные имена ограничений, поскольку это может значительно улучшить поиск этих ограничений.

Декларативные ограничения можно сгруппировать в следующие категории:

– предложение **DEFAULT**;
– предложение **UNIQUE**;
– предложение **PRIMARY KEY**;
– предложение **CHECK**;
– ссылочная целостность и предложение **FOREIGN KEY**.

Предложение UNIQUE

Иногда несколько столбцов или группа столбцов таблицы имеет уникальные значения, что позволяет использовать их в качестве первичного ключа. Столбцы или группы столбцов, которые можно использовать в качестве первичного ключа, называются *потенциальными ключами (candidate key).* Каждый потенциальный ключ определяется, используя предложение **UNIQUE** в инструкции CREATE TABLE или ALTER TABLE.

Синтаксис предложения UNIQUE следующий:

[CONSTRAINT c_name]

UNIQUE [CLUSTERED NONCLUSTERED1 ({col_name1},...)

где, опция CONSTRAINT присваивает явное имя потенциальному ключу; опция CLUSTERED NONCLUSTERED1 указывает тип ключа (кластеризованный, некластеризованный), т.к. SQL Server создает индекс для каждого потенциального ключа таблицы; col_name1 обозначает имя столбца, который создает потенциальный ключ.

Предложение PRIMARY KEY

Первичным ключом таблицы является столбец или группа столбцов, значения которого разные в каждой строке. Каждый первичный ключ определяется, используя предложение **PRIMARY KEY** в инструкции CREATE TABLE или ALTER TABLE.

Синтаксис предложения PRIMARY KEY следующий:

[CONSTRAINT c_name]

PRIMARY KEY [CLUSTERED | NONCLUSTERED] ({col_name1},...)

Все параметры предложения PRIMARY KEY имеют такие же значения, как и соответствующие одноименные параметры предложения UNIQUE. Но в отличие от столбца UNIQUE, столбец PRIMARY KEY не разрешает значений NULL и имеет значение по умолчанию CLUSTERED.

Предложение CHECK

Проверочное ограничение (check constraint) определяет условия для вставляемых в столбец данных. Каждая вставляемая в таблицу строка или каждое значение, которым обновляется значение столбца, должно отвечать этим условиям. Проверочные ограничения устанавливаются посредством предложения **CHECK**, определяемого в инструкции CREATE TABLE или ALTER TABLE.

Синтаксис предложения CHECK следующий:

[CONSTRAINT c_name]

CHECK [NOT FOR REPLICATION] expression

где, параметр expression должен иметь логическое должен иметь логическое значение (true или false) и может ссылаться на любые столбцы в текущей таблице (или только на текущий столбец, если определен как ограничение уровня столбца), но не на другие таблицы.

Предложение FOREIGN KEY

Внешним ключом (foreign key) называется столбец (или группа столбцов таблицы), содержащий значения, совпадающие со значениями первичного ключа в этой же или другой таблице. Внешний ключ определяется с помощью предложения **FOREIGN KEY** в комбинации с предложением **REFERENCES**.

Синтаксис предложения FOREIGN KEY следующий:

[CONSTRAINT c_name]

[[**FOREIGN KEY**] ({col_name1},.)]

REFERENCES table_name ({col_name2},.)

[ON DELETE (NO ACTION] CASCADE | SET NULL | SET DEFAULT}]

[ON UPDATE (NO ACTION | CASCADE | SET NULL | SET DEFAULT}]

где, опция CONSTRAINT присваивает имя внешнему ключу, col_name1 после предложения FOREIGN KEY задает имена столбцов таблицы, которые будут определены как внешние ключи. В предложении REFERENCES указывается имя таблицы, содержащей столбцы, создающие соответствующий первичный ключ. Количество столбцов и их тип данных в предложении FOREIGN KEY должны совпадать с количеством соответствующих столбцов и их типом данных в предложении REFERENCES (они должны совпадать с количеством столбцов и типами данных в первичном ключе таблицы, на которую они ссылаются).

Таблица, содержащая внешний ключ, называется *ссылающейся (или дочерней) таблицей (referencing table)*, а таблица, содержащая соответствующий первичный ключ, называется *ссылочной (referenced table)* или *родительской (parent table) таблицей*.

Ссылочная целостность

Ссылочная целостность (referential integrity) обеспечивает выполнение правил для вставок и обновлений таблиц, содержащих внешний ключ и соответствующее ограничение первичного ключа.

Опции ON DELETE и ON UPDATE

SQL Server на попытку удаления и модифицирования первичного ключа может реагировать по-разному. Если попытаться обновить значения внешнего ключа, то все эти

обновления будут несогласованные с соответствующим первичным ключом, база данных откажется выполнять эти обновления и выведет сообщение об ошибке.

Но при попытке внести обновления в значения первичного ключа, вызывающие несогласованность в соответствующем внешнем ключе, система базы данных может реагировать достаточно гибко. В целом, существует четыре опции, определяющих то, как система базы данных может реагировать.

- ✓ **NO ACTION**. Модифицируются (обновляются или удаляются) только те значения в родительской таблице, для которых нет соответствующих значений во внешнем ключе дочерней (ссылающейся) таблицы.
- ✓ **CASCADE**. Разрешается модификация (обновление или удаление) любых значений в родительской таблице. При обновлении значения первичного ключа в родительской таблице или при удалении всей строки, содержащей данное значение, в дочерней (ссылающейся) таблице обновляются (т. е. удаляются) все строки с соответствующими значениями внешнего ключа.
- ✓ **SET NULL**. Разрешается модификация (обновление или удаление) любых значений в родительской таблице. Если обновление значения в родительской таблице вызывает несогласованность в дочерней таблице, система базы данных присваивает внешнему ключу всех соответствующих строк в дочерней таблице значение NULL. То же самое происходит и в случае удаления строки в родительской таблице, вызывающего несогласованность в дочерней таблице. Таким образом, все несогласованности данных пропускаются.
- ✓ **SET DEFAULT**. Аналогично опции SET NULL, но с одним исключением: всем внешним ключам, соответствующим модифицируемому первичному ключу, присваивается значение по умолчанию. Само собой разумеется, что после модификации первичный ключ родительской таблицы все равно должен содержать значение по умолчанию.

Замечание 4. В языке T-SQL поддерживаются только первые две опции.

Ограничения для обеспечения целостности и домены

Домен (domain) — это набор всех возможных разрешенных значений, которые могут содержать столбцы таблицы. Почти во всех системах управления базами данных для определения таких возможных значений столбца используются такие типы данных, как INT, CHAR и DATE. Такого метода принудительного обеспечения «целостности домена» недостаточно, как можно увидеть в следующем примере.

В таблице Студенты есть столбец Курс, в котором указывается курс, на котором обучается Студенты. Тип данных этого столбца можно определить как SMALLINT или CHAR. Определение типа данных столбца как SMALLINT будет неточным, потому что этот тип данных содержит все положительные и отрицательные целые числа в диапазоне от – 215 – 1 до 215. Объявление с использованием типа данных CHAR будет еще менее точным, поскольку в таком случае можно будет использовать все буквенно-цифровые и специальные символы. Поэтому для точного определения данных столбца индексов требуется диапазон положительных значений от 1 и до 5.

Более точно целостность домена можно принудительно обеспечить с помощью ограничений CHECK (определяемые в инструкции CREATE TABLE или ALTER TABLE), благодаря их гибкости и тому, что они всегда принудительно применяются при вставке или модифицировании данных столбца.

Язык Transact-SQL поддерживает домены посредством создания псевдонимов типов данных с помощью инструкции **CREATE TYPE.**

Псевдонимы типов данных

Псевдоним типа данных (alias data type) — это специальный тип данных, который определяется пользователем при использовании существующих базовых типов данных. Такой тип данных можно использовать в инструкции CREATE TABLE для определения

одного или большего количества столбцов таблицы. Для создания псевдонимного типа данных обычно применяется инструкция **CREATE TYPE**.

Синтаксис этой инструкции представлен ниже:

CREATE TYPE [type_schema_name.] type_name
{[FROM base_type[(precision[, scale])] [NULL | NOT NULL]]
| [EXTERNAL NAME assembly_name [.class_name]]}

Контрольные вопросы

1. Назовите виды объектов базы данных. Приведите примеры.
2. Назовите инструкции создания базы данных и создания таблицы.
3. Дайте определение понятию *владелец базы данных*.
4. Назовите системные процедуры, которые используется для отсоединения базы данных от сервера и присоединения.
5. Поясните смысл предложения DEFAULT и параметра IDENTITY, которые указываются в спецификации столбца при создании таблицы.
6. Дайте определение понятию *ограничения для обеспечения целостности (integrity constraints).*
7. Назовите типы ограничений для обеспечения целостности, которые предоставляются СУБД.
8. Перечислите инструкции языка DDL, с помощью которых определяются декларативные ограничения целостности.
9. Поясните разницу между декларативными ограничениями уровня столбцов и уровня таблицы.
10. Назовите опцию, посредством которой может быть явно присвоено имя декларативному ограничению.
11. Назовите категории декларативных ограничений.
12. Дайте определение понятию *потенциальный ключ*.
13. Дайте определение понятию *первичный ключ*.
14. Назовите основные отличия на столбец UNIQUE и столбец PRIMARY KEY.
15. Дайте определение понятиям *внешний ключ, родительская таблица, дочерняя таблица*.
16. Назовите следующие предложения в инструкции CREATE TABLE или ALTER TABLE: создание потенциального ключа, создание первичного ключа, создание проверочного ограничения, создание внешнего ключа.
17. Поясните понятие *ссылочной целостности*.
18. Назовите и поясните четыре правила, определяющие то, как SQL Server может реагировать на попытку обновления значений первичного ключа, вызывающие несогласованность в соответствующем внешнем ключе.
19. Дайте определение понятию *домен*.
20. Назовите инструкцию создания псевдонимов.
21. Дайте определение понятию *псевдоним типа данных*.

Преподаватель С.В. Банцевич