Лабораторная работа № 2.
Создание реляционной базы данных

Цель работы

Целью выполнения данной лабораторной работы является изучение основных понятий и определений, связанных с физической схемой базы данных, реляционной моделью данных и типов данных языка SQL; получение навыков создания физической схемы базы данных на основе диаграммы «сущность — связь» и ее реализации с помощью операторов языка SQL.

Теоретические сведения

Физическая схема базы данных является описанием структуры данных в терминах платформы реализации — конкретной СУБД; эта модель уже содержит информацию о различных деталях реализации — индексах и ключах, типах атрибутов и т. д., которые определены в терминах целевого языка программирования. Физическая модель фактически является диаграммным представлением части программного кода, определяющего схему данных.

Система управления базами данных (СУБД) — комплекс языковых и программных средств, предназначенный для создания, ведения и использования баз данных.

Здесь и далее в качестве конкретной модели логической организации данных в БД будет использоваться реляционная модель.

Реляционная модель данных — логическая модель данных, основанная на математическом понятии «отношение». Упрощенно отношение можно понимать, как синоним слова «таблица». При использовании реляционной модели данных в таблице в виде строк хранятся сведения об однотипных объектах. При этом каждый столбец таблицы соответствует какой-либо определенной характеристике (атрибуту) объекта. Для описания сложных структур предметной области применяют связывание таблиц друг с другом (например, можно связать друг с другом таблицу, хранящую сведения о студентах, и таблицу, хранящую сведе-

ния об учебных группах).

Приведем алгоритм создания реляционной модели базы данных из ER-диаграммы.

1. Каждая сущность превращается в таблицу. Имя сущности становится именем таблицы.
2. Каждый атрибут становится возможным столбцом с тем же именем; может выбираться более точный формат. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения; столбцы, соответствующие обязательным атрибутам, — не могут. Для каждого столбца таблицы выбирается тип данных, соответствующий виду хранимой в нем информации.
3. Компоненты ключа сущности превращаются в первичный ключ таблицы. Если имеется несколько возможных ключей сущности, выбирается наиболее используемый. Если в состав ключа сущности входят связи, к числу столбцов первичного ключа добавляется копия ключа сущности, находящейся на дальнем конце связи (этот процесс может продолжаться рекурсивно). Для именования этих столбцов используются имена концов связей и/или имена сущностей.
4. Связи «многие к одному» (и «один к одному») становятся внешними ключами. Т. е. делается копия ключа с конца связи «один», и соответствующие столбцы составляют внешний ключ. Необязательные связи соответствуют столбцам, допускающим неопределенные значения; обязательные связи — столбцам, не допускающим неопределенные значения.

SQL (Structured Query Language — структурированный язык запросов) — язык программирования, предназначенный для выполнения операций над таблицами (создание, удаление, изменение структуры) и над данными таблиц (выборка, изменение, добавление, удаление).

Перечислим типы данных в языке SQL.

Символьные типы данных — содержат буквы, цифры и специальные сим-

волы.

- `CHAR` или `CHAR(n)` — символьные строки фиксированной длины. Длина строки определяется параметром `n`. CHAR без параметра соответствует `CHAR(1)`. Для хранения таких данных всегда отводится `n` байт вне зависимости от реальной длины строки.
- `VARCHAR(n)` — символьная строка переменной длины. Для хранения данных этого типа отводится число байт, соответствующее реальной длине строки, но не более `n`.

Целые типы данных — поддерживают только целые числа (дробные части и десятичные точки не допускаются). Над этими типами разрешается выполнять арифметические операции и применять к ним агрегирующие функции (определение максимального, минимального, среднего и суммарного значения столбца реляционной таблицы).

- `INTEGER` (`INT`) — целое, для хранения которого отводится, как правило, 4 байта. (Замечание: число байт, отводимое для хранения того или иного числового типа данных зависит от используемой СУБД и аппаратной платформы, здесь приводятся наиболее «типичные» значения) Интервал значений от -2147483647 до +2147483648.
- `SMALLINT` — короткое целое (2 байта), интервал значений от -32767 до +32768.

Вещественные типы данных — описывают числа с дробной частью.

- `FLOAT` и `SMALLFLOAT` — числа с плавающей точкой (для хранения отводится обычно 8 и 4 байта соответсвенно).
- `DECIMAL(p)` — тип данных аналогичный `FLOAT` с числом значащих цифр `p`.
- `DECIMAL(p,n)` — аналогично предыдущему, `p` — общее количество десятичных цифр, `n` — количество цифр после десятичной запятой.

Денежные типы данных — описывают, естественно, денежные величины.

Если в ваша система такого типа данных не поддерживает, то используйте DECIMAL(p,n).

– MONEY(p,n) — все аналогично типу DECIMAL(p,n). Вводится только потому, что некоторые СУБД предусматривают для него специальные методы форматирования.

Дата и время — используются для хранения даты, времени и их комбинаций. Большинство СУБД умеет определять интервал между двумя датами, а также уменьшать или увеличивать дату на определенное количество времени.

– DATE — тип данных для хранения даты.
– TIME — тип данных для хранения времени.
– INTERVAL — тип данных для хранения временного интервала.
– DATETIME — тип данных для хранения моментов времени (год + месяц + день + часы + минуты + секунды + доли секунд).

Двоичные типы данных — позволяют хранить данные любого объема в двоичном коде (оцифрованные изображения, исполняемые файлы и т. д.). Определения этих типов наиболее сильно различаются от системы к системе, часто используются ключевые слова:

– BINARY
– BYTE
– BLOB

Последовательные типы данных — используются для представления возрастающих числовых последовательностей.

– SERIAL — тип данных на основе INTEGER, позволяющий сформировать уникальное значение (например, для первичного ключа). При добавлении записи СУБД автоматически присваивает полю данного типа значение, получаемое из возрастающей последовательности целых чисел.

В заключение следует сказать, что для всех типов данных имеется общее значение NULL — «не определено». Это значение имеет каждый элемент столб-

ца до тех пор, пока в него не будут введены данные. При создании таблицы можно явно указать СУБД могут ли элементы того или иного столбца иметь значения NULL (это не допустимо, например, для столбца, являющего первичным ключом).

Создание таблиц в языке SQL осуществляется с помощью оператора CREATE TABLE, синтаксис которого приведен ниже.

```
CREATE TABLE <имя_таблицы>
(<имя_столбца> <тип_столбца>
[NOT NULL]
[UNIQUE | PRIMARY KEY]
[REFERENCES <имя_мастер_таблицы> [<имя_столбца>]]
, ...)
```

Пользователь обязан указать имя таблицы и список столбцов. Для каждого столбца обязательно указываются его имя и тип (см. таблицу в предыдущем разделе), а также опционально могут быть указаны параметры:

- NOT NULL — в этом случае элементы столбца всегда должны иметь определенное значение (не NULL);
- один из взаимоисключающих параметров UNIQUE — значение каждого элемента столбца должно быть уникальным или PRIMARY KEY — столбец является первичным ключом;
- REFERNECES <имя_мастер_таблицы> [<имя_столбца>] — эта конструкция определяет, что данный столбец является внешним ключом и указывает на ключ какой мастер_таблицы он ссылается.

Контроль за выполнением указанных условий осуществляет СУБД.

Приведем в качестве примера операторы для создания БД publications:

```
CREATE DATABASE publications;

CREATE TABLE authors (au_id INT PRIMARY KEY,
```

```
author VARCHAR(25) NOT NULL);

CREATE TABLE publishers (pub_id INT PRIMARY KEY,
publisher VARCHAR(255) NOT NULL,url VARCHAR(255));

CREATE TABLE titles (title_id INT PRIMARY KEY,
title VARCHAR(255) NOT NULL,
yearpub INT,
pub_id INT REFERENCES publishers(pub_id));

CREATE TABLE titleautors (au_id INT
REFERENCES authors(au_id),
title_id INT REFERENCES titles(title_id));

CREATE TABLE wwwsites (site_id INT PRIMARY KEY,
site VARCHAR(255) NOT NULL,
url VARCHAR(255));

CREATE TABLE wwwsiteauthors (au_id INT
REFERENCES authors(au_id),
site_id INT REFERENCES wwwsites(site_id));
```

Удаление таблиц в языке SQL осуществляется с помощью оператора `DROP TABLE`, синтаксис которого приведен ниже.

```
DROP TABLE <имя_таблицы>
```

Модификация таблиц в языке SQL осуществляется с помощью оператора `ALTER TABLE`, синтаксис которого приведен ниже.

Добавить столбцы

```
ALTER TABLE <имя_таблицы> ADD
(<имя_столбца> <тип_столбца>
[NOT NULL]
[UNIQUE | PRIMARY KEY]
[REFERENCES <имя_мастер_таблицы> [<имя_столбца>]],...)
```

Удалить столбцы

```
ALTER TABLE <имя_таблицы> DROP (<имя_столбца>,...)
```

Модификация типа столбцов

```
ALTER TABLE <имя_таблицы> MODIFY
(<имя_столбца> <тип_столбца>
[NOT NULL]
[UNIQUE | PRIMARY KEY]
[REFERENCES <имя_мастер_таблицы> <имя_столбца>]],...)
```

Ход выполнения работы

1. Изучить теоретические сведения.
2. Для спроектированной ранее концептуальной схемы БД создать физическую схему.
3. Создать БД по созданной физической схеме с использованием операторов языка SQL.
4. Составить отчет о выполнении лабораторной работы.
5. Подготовить ответы на контрольные вопросы.

Контрольные вопросы

1. Дайте определение физической схемы БД.
2. Что такое СУБД?
3. Что такое реляционная модель данных?

4. Опишите алгоритм создания реляционной базы данных по ER-диаграмме.
5. Перечислите и охарактеризуйте основные типы данных в языке SQL.
6. Запишите и прокомментируйте синтаксис оператора SQL для создания таблицы.
7. Запишите и прокомментируйте синтаксис оператора SQL для удаления таблицы.
8. Запишите и прокомментируйте синтаксис оператора SQL для модификации таблицы.