МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Севастопольский государственный университет»

СОЗДАНИЕ СХЕМЫ БД. ССЫЛОЧНАЯ ЦЕЛОСТНОСТЬ

Методические указания
к лабораторной работе №3
по дисциплине
«Теория баз данных»
для студентов, обучающихся по направлениям
09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика» по учебному плану подготовки бакалавров
дневной и заочной форм обучения

Севастополь
2020

УДК 004.92

Создание схемы БД. Ссылочная целостность. Методические указания к лабораторной работе №3 по дисциплине «Теория баз данных», для студентов, обучающихся по направлениям 09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика» по учебному плану подготовки бакалавров дневной и заочной форм обучения /Сост. Ю.В. Доронина, А.В. Волкова – Севастополь: Изд-во СГУ, 2020. – 9 с.

Цель методических указаний: научиться анализировать предметную область с целью создания схемы БД, учитывая ссылочную целостность набора.

Допущено учебно-методическим центром в качестве методических указаний.

1 ОСНОВНЫЕ ПОЛОЖЕНИЯ

1.1 Типы связей между отношениями. Требования к ссылочной целостности данных

Свое название реляционные базы данных получили именно по той причине, что таблицы в БД не существуют независимо друг от друга. Таблицы взаимосвязаны друг с другом, т.е. действие, произведенное в одной таблице, вызовет некоторые действия в другой таблице.

Существует три основных класса связей между таблицами:

- один к одному (1:1);
- один ко многим (1:М);
- многие ко многим (М:М).

На практике связи первого типа используются редко. Связи третьего типа не реализуются в РБД напрямую, одну связь многие ко многим приводят к двум связям один ко многим. Поэтому связь 1:М используется наиболее часто, ее следует рассмотреть наиболее подробно.

Пусть имеется две таблицы следующего вида:

1. **Клиент** (Номер_клиента, ФИО_клиента, Счет_клиента)
2. **Продажи** (Номер_клиента, Наименование_товара, Количество_товара)

Таблица **Клиент** представлена на рисунке 1. Таблица **Продажи** представлена на рисунке 2.

Таблицы связанны отношением 1:М, т.е. один клиент может совершить много покупок, каждая покупка совершается только одним клиентом. В таблицах хранятся следующие данные:

Номер_клиента	**ФИО_клиента**	**Счет_клиента**
1	Сидоров И.И.	1923563
2	Петров А.Б.	1736523
3	Федоров Н.Г.	9265623

Рисунок 1 – Таблица **Клиент**.

Номер_клиента	**Наименование_товара**	**Количество_товара**
1	Шоколад «Корона»	2
1	Шейки Куринные	1,5
2	Крупа манная	3

Рисунок 2 – Таблица **Продажи**

Очевидно, что поле Номер_клиента в обеих таблицах имеет одинаковый смысл - оно обозначает номер, присвоенный клиенту. В случае если таблицы не имеют связи друг с другом, ничего не мешает добавить в таблицу «Продажи» запись, например, (8, «Хлеб белый»,1), несмотря на то, что в таблице «Клиент» нет покупателя с номеров 8. Это нарушение целостности данных, так как такая строка может быть занесена, но она не соответствует действительности. Чтобы подобная ситуация стала невозможной, необходимо установить связь между таблицами по полю «Номер_клиента». В этом случае сервер БД автоматически проследит, чтобы поле «Номер_клиента» из вставляемой строки существовало в таблице «Клиенты».

Подобной проверки достаточно, чтобы условие целостности данных выполнялось при добавлении данных в таблицы. Но его недостаточно при манипулировании данными.

Рассмотрим ситуацию, когда необходимо удалить из таблицы «Клиент» некоторую запись, например, клиента с номером 1. В случае если таблицы не связанны, удаление клиента повлечет за собой изменение только одной таблицы. В таблице «Продажи» останутся сведения о покупках покупателя с номером 1. Такая ситуация – также нарушение целостности данных, так как о данном покупателе, после его удаления, базе данных ничего не известно. В случае если таблицы связанны, удаление покупателя может повлечь за собой удаление всех его покупок (говорят, что удаление каскадируется). Использование конструкций оператора CREATE TABLE для обеспечения целостности данных приведено в Приложении А.

Такая же ситуация с модификацией данных в родительской таблице. Если покупатель с номером 1 изменил номер на 5, то в связанных таблицах изменение родительской таблицы повлечет за собой автоматическое изменение дочерних таблиц.

С помощью идентифицирующей связи устанавливается взаимосвязь между зависимыми сущностями (клиент и его покупки). Не идентифицирующая связь устанавливает взаимосвязь между независимыми сущностями (автомобиль и его цвет). Разница между идентифицирующей и не идентифицирующей связью состоит в том, что при идентифицирующей связи внешний ключ является частью первичного ключа дочерней сущности, а при не идентифицирующей связи внешний ключ не входит в первичный ключ дочерней сущности. На ER-диаграмме идентифицирующая связь изображается непрерывной линией, не идентифицирующая – пунктирной.

1.2 Стандартная нотация ER-диаграмм

Приведем пояснения к стандартной нотации ER-диаграмм, в которой представлены варианты к лабораторной работе.

Сущности. Сущность изображается прямоугольником, над которым пишется имя сущности. Одна сущность соответствует одной таблице. Прямоугольник разделяется линией на две части. В верхней части указываются ключевые атрибуты. После имени атрибута могут стоять символы, уточняющие назначение атрибута. Допустимые символы:

- PK – первичный ключ;
- AK – альтернативный ключ;
- FK – внешний ключ;
- IE – inversion entry, говорит о том, что по данному полю должен быть создан индекс.

Связи. Связь между двумя отношениями изображается с помощью линии. Идентифицирующая связь изображается сплошной линией, не идентифицирующая - пунктирной. Арность связи указывается следующим образом: со стороны "многие" ставится жирная точка, со стороны «один» точка не ставится. Допустимость Null – значений изображается ромбиком с той стороны связи, где позволяются Null – значения.

Категории. В случае если у нас есть некоторая общая сущность (например – люди), и необходимо хранить информацию о некоторых разновидностях данной сущности (например – родители и дети, мужчины и женщины, работающие и безработные и т.д.) имеет место понятие категории.

Рассмотрим на примере использование понятия категории. Допустим, нам необходимо хранить следующую информацию о родителях и детях:

Родитель (Номер, ФИО, Адрес, Образование)
Ребенок (Номер, ФИО, Адрес, Номер_детского_садика)
ЯвляетсяРебенком (Номер_родителя, Номер_ребенка)

Схема данных:
Родитель ------- 1:M -------> **Ребенок**

Очевидно, что для ребенка поле «Образование» не имеет смысла, как и поле «Номер_детского_садика» для родителя. Кроме того, в случае, если подобную информацию хранить в виде двух таблиц, то возникает дублирование информации и проблемы с целостностью данных. Например, адрес ребенка совпадает с адресом родителей, и если данные вводятся вручную, то можно ввести адрес ребенка отличный от адреса родителя.

Подобная проблема решается следующим образом: вводится обобщающая категория «Личность», в нее выносятся общие поля из двух дочерних таблиц. Дочерние таблицы привязываются к родительской связью типа «один к одному».

Личность (Номер, ФИО);
Родитель (Номер, Адрес, Образование);
Ребенок (Номер, Номер_детского_садика);
ЯвляетсяРебенком(Номер_родителя, Номер_ребенка)

Схема данных:
Родитель --------1:1--------> Личность<--------1:1-------- Ребенок

Для того чтобы добавить данные о ребенке, нужно занести данные в две таблицы – «личность» и «ребенок». Например:

```
ISERT INTO Личность VALUES (2,'Иванов И.И.');
ISERT INTO Ребенок VALUES (2,'д/с №23');
```

Нумерация в таблицах сквозная!

На рисунке 3 изображена схема категориальной связи.

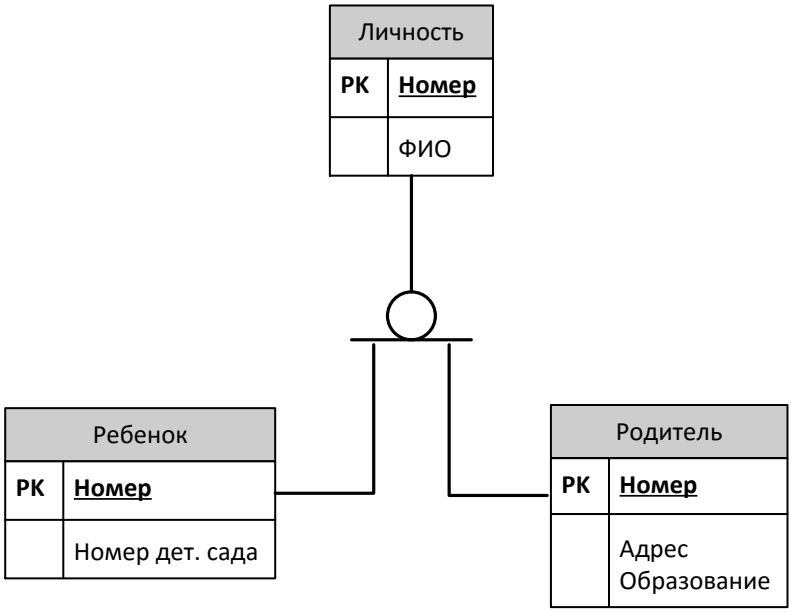

Рисунок 3 – Схеме категориальной связи

Допустимость NULL-значений связи на диаграмме указывается ромбиком.

Связь таблиц в реляционной БД устанавливается при создании таблиц с помощью описателя FOREIGEN KEY (см. приложение А).

2 ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Проанализировать схему БД своего варианта задания (вариант то же, что и в лабораторной работе №1), выделить и классифицировать все существующие связи, определить необходимые ограничения целостности.
2. Создать все еще не созданные таблицы, изменить существующие таким образом, чтобы они могли участвовать в связях (описание ALTER TABLE).
3. Установить связи между таблицами.
4. Проверить работу ограничений целостности (каскадирование удаления, модификации и др.)
5. Подготовить и защитить отчет.

3 СОДЕРЖАНИЕ ОТЧЕТА

1. Отчет состоит из титульного листа, цели работы, описания процесса выполнения работы и вывода.
2. Отчет должен содержать исходные и модифицированные таблицы.
3. В отчете необходимо привести список связей по арности, описать ограничения целостности, допустимость Null-значений, идентифицируемость.
4. Отражение работы по проверке целостности данных.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Требования к ссылочной целостности данных?
2. Типы связей между отношениями?
3. Стандартная нотация ER-диаграмм?
4. Нормальные формы для баз данных?
5. Необходимость процесса нормализации БД?
6. Приведение БД к нормальной форме Бойса-Кодда?
7. Обосновать в какой нормальной форме находится полученная схема БД?
8. Способы повышения надежности данных?

ПРИЛОЖЕНИЕ А

Ограничения целостности данных

Ограничения целостности - это правила, которые отслеживают достоверность данных. Ограничения целостности могут проверять значение, вводимое в один столбец (например, возраст не может быть отрицательным, и не может быть больше 150 лет), может проверять значение, вводимое в несколько столбцов например, если в поле "возраст" введено значение 10 лет и менее, то в поле «количество детей» не может быть значение большее 0, может проверять несколько таблиц (например, при удалении заказчика надо удалить все его заказы). Ограничения, проверяющие значение, вводимое в столбец, называются ограничениями на уровне атрибута, проверяющие несколько столбцов – ограничения на уровне таблицы, проверяющие несколько таблиц – ограничения на уровне связи.

CREATE TABLE может создавать следующие типы ограничений целостности:

− PRIMARY KEY (первичный ключ) - уникально идентифицирует каждую строку таблицы. Значение в указанном столбце либо в упорядоченном наборе столбцов не могут повторяться в более чем одной строке. Столбец PRIMARY KEY должен быть определен только с атрибутом NOT NULL. Таблица может иметь только один PRIMARY KEY, который может быть определен на одном или более столбцов.

− UNIQUE (уникальные) ключи гарантируют, что не существует двух строк, имеющих одно и тоже значение в указанном столбце или упорядоченном наборе столбцов. Уникальный столбец должен быть определен с атрибутом NOT NULL. Таблица может иметь один или более UNIQUE ключей.

В случае, если первичный или альтернативный ключ состоит из одного поля, его можно описать в той же строке, что и само поле (такое ограничение называется ограничением на уровне столбца). Если первичный или альтернативный ключ состоит из нескольких атрибутов, он описывается после определения всех полей таблицы (такое ограничение называется ограничением на уровне таблицы).

Ограничение на уровне столбца имеет следующий синтаксис:

```
<col_constraint> = [CONSTRAINT <имя ограничения>] <определение ограничения>
```

т.е. определение ограничения состоит из необязательной части, состоящей из служебного слова CONSTRAINT и имени ограничения, и обязательной части, состоящей из собственно определения ограничения. В случае, если первая часть опущена, сервер генерирует имя ограничения автоматически. Посмотреть имя ограничения можно просмотрев системную таблицу RDB$RELATION_CONSTRAINTS.

Определение ограничения на уровне столбца описывается так:

```
<определение ограничения> = {
UNIQUE | PRIMARY KEY|CHECK (<условие проверки>)
| REFERENCES <ИмяДругойТаблицы> [( <имя столбца>[, <имя столбца> …])]
[ON DELETE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]
[ON UPDATE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]
}
```

Поле можно определить как первичный ключ, его значение можно сделать уникальным по таблице, либо можно указать, что поле ссылается (REFERENCES) на некоторое поле из другой таблицы. Таблица, на которую ссылается поле, называется родительской. Смысл ограничения REFERENCES следующий: значение в поле в дочерней таблицы можно занести только в том случае, если это значение есть в соответствующем поле родительской таблицы. Например, если у нас есть таблица клиентов, и таблица покупок, поле «Номер клиента» из таблицы покупок должно ссылаться на поле «Номер клиента» из таблицы клиентов. Таким образом в таблицу покупое невозможно будет занести номер несуществующего клиента. Вторая часть описателя REFERENCES описывает, какие действия необходимо предпринять при удалении из родительской таблицы (ON DELETE) и при обновлении родительской таблицы (ON UPDATE).

Всего возможно четыре варианта:

- NO ACTION – ничего не предпринимать;
- CASCADE – каскадировать. В случае удаления из родительской таблицы будут удалены все связанные записи из дочерней таблицы. Например, удаление клиента повлечет удаление всех его покупок. При модификации записи из родительской таблицы будут модифицированы также записи в дочерней таблице. Например, если клиент поменял номер с пятого на десятый, во всех его покупках поле “Номер клиента” также изменит свое значение с 5 на 10;
- SET DEFAULT – поле получает свое значение по умолчанию;
- SET NULL – поле устанавливается в NULL.

Какое именно действие необходимо предпринимать зависит от смысла таблиц. При удалении клиента удалять все его покупки имеет смысл. Но при удалении желтого цвета из таблицы цветов не стоит удалять все желтые автомобили.

Ограничения целостности на уровне таблицы описывается так же, как и ограничение на уровне столбца, за исключением определения самого ограничения .

```
<ограничение> = {{PRIMARY KEY | UNIQUE} ( <имя столбца> [,<имя столбца> …])
| FOREIGN KEY (<имя столбца> [,<имя столбца> …]) REFERENCES <ИмяДругойТаблицы>
[ON DELETE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]
[ON UPDATE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]
| CHECK (<условие>)}
```

Отличие в том, что в PRIMARY KEY, UNIQUE и FOREIGN KEY можно описать более одного столбца.

Ссылочные ограничения гарантируют, что значения в наборе столбцов, которые определены в FOREIGN KEY принимают те же самые значения, которые присутствуют в столбце UNIQUE или PRIMARY KEY в таблице, на которую они ссылаются. Это ограничение на уровне связи.

1. Следующая инструкция создает одиночную таблицу с PRIMARY KEY:

```
CREATE TABLE COUNTRY
  (COUNTRYNAME NOT NULL PRIMARY KEY,
   CURRENCY VARCHAR(10) NOT NULL);
```

2. Следующая инструкция создает таблицу с UNIQUE ограничением и на уровне столбца MODEL и на уровне таблицы (столбцы MODELNAME, ITEMID):

```
CREATE TABLE STOCK
  (MODEL SMALLINT NOT NULL UNIQUE,
  MODELNAME CHAR(10) NOT NULL,
  ITEMID INTEGER NOT NULL,
  CONSTRAINT MOD_UNIQUE UNIQUE (MODELNAME, ITEMID));
```

3. Следующая инструкция создает таблицу с вычисляемым столбцом NEW_SALARY:

```
CREATE TABLE SALARY_HISTORY
  (EMP_NO EMPNO NOT NULL,
  CHANGE_DATE DATE DEFAULT "NOW" NOT NULL,
  UPDATER_ID VARCHAR(20) NOT NULL,
  OLD_SALARY SALARY NOT NULL,
  PERCENT_CHANGE DOUBLE PRECISION DEFAULT 0 NOT NULL
                 CHECK (PERCENT_CHANGE BETWEEN -50 AND 50),
  NEW_SALARY COMPUTED BY (OLD_SALARY + OLD_SALARY * PERCENT_CHANGE / 100),
  PRIMARY KEY (EMP_NO, CHANGE_DATE, UPDATER_ID),
  FOREIGN KEY (EMP_NO) REFERENCES EMPLOYEE (EMP_NO));
```

Пояснения к оператору. Объявлены поля:

- поле EMP_NO целочисленное, не может принимать значение NULL;

− поле CHANGE_DATE имеет тип DATE, значение по умолчанию – текущая дата, не может принимать значение NULL;

− поле UPDATER_ID – строка переменной длины с максимальной длиной 20 символов, не может принимать значение NULL;

− поле OLD_SALARY – содержит денежное значение, которое хранится с точностью до трех знаков после запятой, не может принимать значение NULL;

− поле PERCENT_CHANGE вещественное число, по умолчанию имеет значение 0, не может принимать значение NULL.

Поле **NEW_SALARY** вычисляется как:

```
OLD_SALARY + OLD_SALARY * PERCENT_CHANGE / 100
```

Объявлены ограничения:

− ограничение CHECK – поле **PERCENT_CHANGE** может принимать значение из диапазона [-50,50];

− объявляется составной первичный ключ, состоящий из трех полей: **EMP_NO, CHANGE_DATE, UPDATER_ID**;

− описывается внешний ключ (FOREIGN KEY) - поле EMP_NO ссылается (REFERENCES) на поле EMP_NO таблицы EMPLOYEE, обновление и удаление из таблицы EMPLOYEE каскадируется.