ТЕМА 3. ПРИНЦИПЫ ПОСТРОЕНИЯ БАЗ ДАННЫХ. ЖИЗНЕННЫЙ ЦИКЛ БАЗ ДАННЫХ

Классическая технология проектирования реляционных баз данных связана с *теорией нормализации*, основанной на анализе функциональных зависимостей между атрибутами отношений. Процесс нормализации имеет своей целью устранение избыточности данных. Нормализация позволяет существенно сократить объем хранимой информации и устранить аномалии в организации хранения данных. Степень нормализации данных может быть различной. Приведение модели к требуемому уровню нормальной формы является основой построения реляционной базы данных.

Нормализация достигается путем проверки соответствия таблиц ряду условий, определенных в трех уровнях нормализации: первой, второй и третьей нормальных формах (существуют также и другие уровни).

Первая нормальная форма требует, чтобы каждое поле таблицы БД было неделимым и не содержало повторяющихся групп.

Неделимость поля означает, что содержащиеся в нем значения не должны делиться на более мелкие. Например, если в поле «Подразделение» содержится название факультета и кафедры, требование неделимости не соблюдается и необходимо выделить название факультета или кафедры в отдельное поле.

Повторяющимися являются поля, содержащие одинаковые по смыслу значения. Например, если требуется получить статистику продаж четырех товаров по месяцам, можно создать поля для хранения данных о продаже по каждому товару. Однако что делать, если товаров не 4, а 104, и как быть, если количество товаров заранее не известно? Повторяющиеся группы следует устранить, сохранив в таблице единственное поле «Товар». В результате получим запись, содержащую информацию о статистике продаж по одному товару, но этот товар может быть любым.

Вторая нормальная форма требует, чтобы все поля таблицы зависели от первичного ключа, то есть, чтобы первичный ключ однозначно определял запись и не был избыточен. Если же в какой-либо таблице имеется зависимость каких-либо не ключевых полей от части первичного ключа,

следует выделить их в отдельную таблицу, сделав первичным ключом новой таблицы ту часть первичного ключа, от которой зависят данные поля, и установить связь "один ко многим" от новой таблицы к старой.

Третья нормальная форма требует, чтобы в таблицах не имелось транзитивных зависимостей между не ключевыми полями, то есть чтобы значение любого поля, не входящего в первичный ключ, не зависело от значения другого поля, также не входящего в первичный ключ.

Результатом нормализации является модель данных, которую легко поддерживать, не содержащая неопределенностей в данных и повторений данных.

После формальных определений трех уровней нормализации разберем конкретный пример и опишем возможные проблемы. В качестве примера будет рассматриваться база данных, содержащая сведения о посещаемых студентами курсах.

Таблицы базы данных до нормализации

В этом примере предполагается, что:

- студент может записаться на любое число курсов;
- лекторы могут вести несколько курсов;
- каждый лектор всегда проводит занятия в одной и той же аудитории;
- в каждой аудитории читается только один курс.

Пусть для хранения этих сведений используются следующие Таблицы.

Students (студенты)

Таблица 6

Name (имя)	Phoneno (телефон)	CourseRegistrations (посещаемые курсы)
Maijorie Green	415986	Basic Computing, Database Administration
Bun Gringelsby	707938	Database Administration, Advanced Hardware Support
Anico Yokamoto	415935	Advanced Hardware Support

Courses (курсы)

Таблица 7

Course (курс)	Lecturer (лектор)	Room (аудитория)
Basic Computing	Meander Smith	542 South
Database Administration	Dean Straight	221 East
Advanced Hardware Support	Dean Straight	221 East

В этом случае появляются следующие логические противоречия:

· если курс Basic Computing будет закрыт, из таблицы будет удален лектор Meander Smith и аудитория 542 South;

· число курсов, на которые может записаться студент, ограничено длиной записи которую допускает поле *Course Registrations;*

· трудно выполнять поиск значений в поле *Course Registrations, а* также использовать его в вычислениях;

· в каждой регистрационной записи повторяется полное название курса. В результате неэффективно используется пространство и растет вероятность появления несогласованных данных, если название курса введено с ошибками. Кроме того, при изменении названия курса потребуется проводить поиск и обновление всех регистрационных записей;

· таблицу *Students* невозможно индексировать по фамилии, так как в поле *name* хранятся полные имена студентов;

· если лектор сменит аудиторию, придется обновить сведения обо всех преподаваемых им курсах.

Проведем нормализацию

Таблицы базы данных после нормализации

Students (студенты)

Таблица 8

StudentsID(код студента)	Firstname (имя)	Lastname (фамилия)	Phoneno (телефон)

1001	Maijorie	Green	415986
1002	Bun	Gringelsby	707938
1003	Anico	Yokamoto	415935

Регистрационные записи (Registrations)

Таблица 9

RegID (код записи)	StudentsID(код студента	Courses (курсы)
1	1001	1
2	1002	2
3	1003	3
4	1004	4
5	1005	5

Courses (курсы)

Таблица 10

Course ID(курс)	Course (курс)	LecturerID (код лектора)
1	Basic Computing	1
2	Database Administration	2
3	Advanced Hardware Support	3

Lecturers (лектор)

Таблица 11

LecturerID (код лектора)	Firstname (имя)	Lastname (фамилия)	Room (аудитория)

1	Meander	Smith	542 South
2	Dean	Straight	221 East

Между таблицами существуют следующие связи:

Students (студенты) — *Courses* (курсы): отношение «многие ко многим» через промежуточную таблицу *Registrations* (регистрационные записи), другими словами это отношение сведено к двум отношениям «один ко многим»;

Students (студенты) — *Registrations* (регистрационные записи): отношения «один ко многим»;

Courses (курсы) — *Registrations* (регистрационные записи): отношение «один ко многим»;

Lecturers (лекторы) — *Courses* (курсы): отношение «один ко многим».

Очевидные преимущества нормализации этих таблиц:

- каждая таблица содержит только один набор связанных данных. Например, в таблице *Students* теперь нет сведений о посещаемых курсах;
- в каждой таблице имеется первичный ключ: в таблице *Students* — это поле *StudentID,* в таблице *Registrations* — *RegID,* в таблице *Courses* — *CourseID* и в таблице *Lecturers* — *LecturerW,*
- отсутствуют составные поля. Каждое поле описывает только один атрибут. *Например,* поле, содержавшее имя и фамилию студента, разбито на отдельные поля, которые содержат имя и фамилию студента;
- отсутствуют повторяющиеся данные. Так, теперь имена лекторов записываются только один раз;
- отсутствуют поля, содержащие несколько значений. Например, каждая регистрационная запись курса теперь расположена в отдельной строке таблицы *Registrations.* Для сравнения взгляните на поле *Course Registrations* (посещаемые курсы) предыдущего варианта таблицы *Students;*
- каждое поле полностью зависит от первичного ключа. Например, в таблице *Courses* нет поля *Room.* Это связано с тем, что аудитория зависит не от кода курса *(CourseID),* а от кода лектора *(LecturerID).*

Вот основные преимущества нормализации:

- облегчается сортировка и создание индекса, поскольку таблицы стали более компактными;
- индексы становятся более компактными;
- меньшее число индексов в одной таблице позволяет быстрее выполнять обновления записей;
- в таблицах содержится меньше значений NULL и избыточных данных, что повышает компактность базы данных;
- уменьшается вероятность конфликтов блокировок таблиц, поскольку блокировать приходится ограниченные наборы данных.

Проект реляционной базы данных - это набор взаимосвязанных отношений, для которых определены все атрибуты, заданы первичные ключи отношений и заданы еще некоторые дополнительные свойства отношений, которые относятся к принципам поддержки целостности. Фактически проект базы данных - это фундамент будущего программного комплекса, который будет использоваться достаточно долго и многими пользователями. Этапы жизненного цикла базы данных (см рис 1) аналогичны, в основном, развитию любой программной системы, однако в них есть определенная специфика, касающаяся только баз данных.

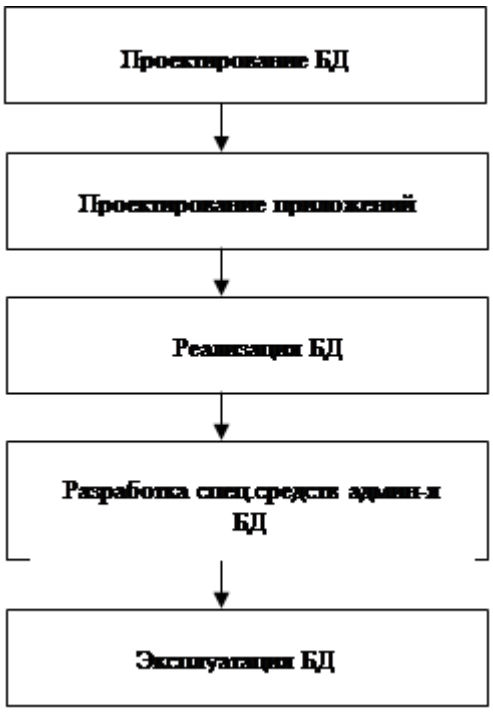

Рис. 1. Этапы жизненного цикла БД

Процесс проектирования БД представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. Можно выделить следующие *этапы проектирования*:

1. Системный анализ и словесное описание информационных объектов предметной области.

2. Проектирование инфологической модели предметной области - частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели.

3. Даталогическое или логическое проектирование БД, то есть описание БД в терминах принятой дата логической модели данных.

4. Физическое проектирование БД, то есть выбор эффективного размещения БД на внешних носителях для обеспечения наиболее эффективной работы приложения.

Если учесть, что между вторым и третьим этапами необходимо принять решение, с использованием какой стандартной СУБД будет реализовываться

наш проект, то условно процесс проектирования БД можно представить последовательностью выполнения пяти соответствующих этапов (см. рис. 2.)

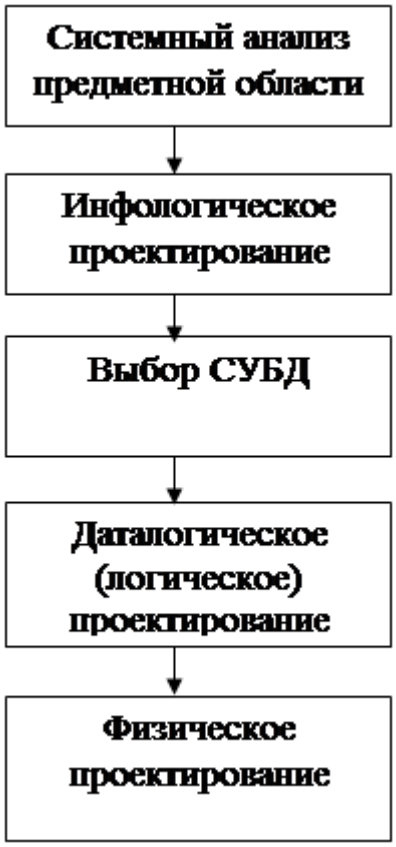

Рис. 2. Этапы проектирования БД