КОНСПЕКТ ЗАНЯТИЯ № 1 ЧЕТВЕРТОЙ НЕДЕЛИ КУРСА «БАЗЫ ДАННЫХ»

1. НОРМАЛИЗАЦИЯ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Перейдем к рассмотрению подходов к логическому проектированию реляционных баз данных. На этом этапе разрабатывается логическая структура БД: определяются реляционные отношения, их атрибуты, потенциальные и внешние ключи. Эта задача очень ответственная, так как ошибки, сделанные при разработке структуры данных, могут привести к нежелательным эффектам при сохранении, удалении и изменении данных, называемым *аномалиями модификации*.

Аномалия удаления проявляется в том, что удаляя факты, относящиеся к одной сущности (объекту, явлению и т. п.), мы непроизвольно удаляем факты, относящиеся к другой сущности.

Аномалия вставки приводит к тому, что нельзя поместить в БД некоторый факт об одной сущности, не указав дополнительно некоторый факт о другой сущности.

Аномалия обновления заключается в том, что для внесения изменения в данные об одном факте (событии, объекте), необходимо выполнить множественные изменения в БД.

Примеры аномалий будут приведены ниже.

Одним из существенных преимуществ реляционной модели является наличие формального механизма оценки качества логической структуры базы данных средствами теории нормализации.

Нормализация – это процесс приведения структуры реляционных отношений к форме, обладающей лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, т. е. исключена избыточность информации. Кроме задачи более эффективного использования памяти, нормализация позволяет снизить угрозу нарушения целостности базы данных из-за появления в ней внутренних противоречий.

Вводится понятие *нормальных форм* (НФ) отношений, каждой из которых соответствует определенный набор ограничений. Отношение находится в данной нормальной форме, если удовлетворяет указанным ограничениям. Каждая следующая НФ включает в себя требования всех

предыдущих, т. е. является более строгим ограничением. Прежде чем перейти к рассмотрению конкретных НФ, необходимо ввести ряд определений.

Пусть R – реляционное отношение, а X и Y – некоторые подмножества атрибутов этого отношения. Y *функционально зависимо* от X тогда и только тогда, когда для каждого значения множества X существует только одно значение множества Y [3]. Иначе говоря, если два кортежа отношения совпадают по значению X, то они обязательно будут совпадать и по значению Y. Записывается функциональная зависимость (ФЗ) как X→Y, читается как «X функционально определяет Y». Если существует ФЗ X→Y, то X называют *детерминантом*, а Y – *зависимой частью*.

Из определения ФЗ, в частности, следует, что любое подмножество атрибутов отношения функционально зависимо от любого из потенциальных ключей. Иными словами: если атрибуты X составляют потенциальный ключ отношения R, то *любой* атрибут отношения R функционально зависит от X.

Забегая вперед, можно заметить, что если отношение R удовлетворяет ФЗ A→B и A не является потенциальным ключом, то в R присутствует некоторая избыточность, которая должна быть устранена в ходе процесса нормализации.

Избыточность данных приводит к непродуктивному расходованию свободного места на диске и затрудняет обслуживание баз данных. Например, если данные, хранящиеся в нескольких местах, потребуется изменить, в них придется внести одни и те же изменения во всех местах.[1]

Рассмотрим пример. Пусть отношение STUDENTS имеет структуру, представленную в табл. 5.1 и атрибут StudID является первичным ключом. В представленном отношении есть следующие функциональные зависимости:

- {StudID} → {ФИО};
- {StudID, ФИО} → {Группа};
- {StudID, ФИО} → {StudID} и т.д.

[1] Информация взята из доклада студента второго курса направления Информатика и вычислительная техника – Веховой Ксении.

Таблица 5.1

Отношение STUDENTS

StudID	ФИО	Группа
123	Иванов И.И.	382
124	Петров П.П.	382
127	Сидоров С.С.	383

В то же время, подмножество атрибутов {Группа} в общем случае не является функционально зависимым от {ФИО}, так как в списке студентов могут оказаться однофамильцы с одинаковыми инициалами, но обучающиеся в разных группах.

Как видно из примера, множество ФЗ, даже для отношения с небольшой степенью, может быть достаточно велико. В то же время, ФЗ является ограничением целостности и его выполнение должно проверяться СУБД при таких операциях, как добавление, удаление или изменение данных. Поэтому желательно исключить из рассмотрения такие ФЗ, которые можно вывести из других ФЗ.

1.1. ПЕРВАЯ НОРМАЛЬНАЯ ФОРМА[2]

Любое базовое отношение находится в **первой нормальной форме** тогда и только тогда, когда схема этого отношения содержит только *простые* (состоящие из одного компонента) и только *однозначные* (содержащие одно значение для одной сущности) атрибуты и ни один из ключевых атрибутов не имеет значения NULL. *Ключевым* является атрибут, входящий в любой из потенциальных ключей.

Непервую нормальную форму можно получить, если допустить, что атрибуты отношения могут быть определены на сложных типах данных – массивах, структурах. Легко себе представить таблицу, у которой в некоторых в некоторых ячейках содержатся массивы, в других ячейках – определенные пользователями сложные структуры и тд.

Требование, что отношения должны содержать только данные простых типов, объясняет, почему отношения иногда называют *плоскими таблицами*.

[2] Раздел подготовлен и разработан в соавторстве со студентом 2 курса направления Информатика и вычислительная техника – Веховой Ксении.

Таким образом, любое отношение автоматически уже находится в 1НФ. Напомним кратко свойства отношений (это и будут свойства 1НФ):

1. *Кортежи не упорядочены* - кортежи отношений не должны зависеть друг от друга.
2. *В отношении нет одинаковый кортежей* – все кортежи должны быть уникальны.
3. *Атрибуты не упорядочены* – порядок атрибутов не должен влиять на понимание информации.
4. *Атрибуты уникальны* – в отношении не могут существовать несколько атрибутов с одним именем.
5. *Все значение атрибутов атомарны*. Элемент является атомарным, если его нельзя разделить на части, которые могут использоваться в таблице независимо друг от друга.

В ходе логического моделирования на первом шаге предложено хранить данные в одном ТАБЛИЦЕ (**СОТРУДНИКИ**), имеющем следующие СТОЛБЦЫ:

- **Н_СОТР** (табельный номер сотрудника)**,**
- **ФАМ** (фамилия сотрудника)**,**
- **Н_ОТД** (номер отдела, в котором работает сотрудник)**,**
- **ТЕЛ** (телефон отдела)**,**
- **Н_ПРО** (номер проекта, над которым работает сотрудник)**,**
- **ПРОЕКТ** (наименование проекта, над которым работает сотрудник)**,**
- **Н_ЗАДАН** (номер задания, над которым работает сотрудник).

Н_СОТР	ФАМ	Н_ОТД	ТЕЛ	Н_ПРО	ПРОЕКТ	Н_ЗАДАН
1	Иванов	1	11-22-33	1 2	Космос Климат	1 1
2	Петров	1	11-22-33	1	Космос	2
3	Сидоров	2	33-22-11	1 2	Космос Климат	3 2

Данная группированная ТАБЛИЦА нарушает требования 1НФ, то есть структура не является ОТНОШЕНИЕМ.

Чтобы привести ТАБЛИЦУ к требованиям 1НФ надо выполнить преобразования, результат которых представлен ниже. Обратите внимание, что в получившемся ОТНОШЕНИИ составной первичный

ключ = {Н_СОТР, Н_ПРО}, так как каждый сотрудник в каждом проекте выполняет ровно одно задание.

Н_СОТР	**ФАМ**	**Н_ОТД**	**ТЕЛ**	***Н_ПРО***	**ПРОЕКТ**	**Н_ЗАДАН**
1	Иванов	1	11-22-33	*1*	Космос	1
1	Иванов	1	11-22-33	*2*	Климат	1
2	Петров	1	11-22-33	*1*	Космос	2
3	Сидоров	2	33-22-11	*1*	Космос	3
3	Сидоров	2	33-22-11	*2*	Климат	2

В текущий момент состояние предметной области отражается следующими фактами:

- Сотрудник Иванов, работающий в 1 отделе, выполняет в первом проекте "Космос" задание 1 и во втором проекте "Климат" задание 1.
- Сотрудник Петров, работающий в 1 отделе, выполняет в первом проекте "Космос" задание 2.
- Сотрудник Сидоров, работающий во 2 отделе, выполняет в первом проекте "Космос" задание 3 и во втором проекте "Климат" задание 2.

Хотя данное преобразование и улучшило структуру отношения, оно все равно содержит избыточную информацию – повторяющиеся фамилии сотрудников, номер телефонов, наименования проектов. Кроме того, в данном отношении хранятся вместе независимые друг от друга данные - и данные о сотрудниках, и об отделах, и о проектах, и о работах по проектам. Пока никаких действий с отношением не производится, это не страшно. Но как только состояние предметной области изменяется, то, при попытках соответствующим образом изменить состояние базы данных, возникает большое количество проблем.

Рассмотрим аномалии модификации этой таблицы:

1. *Аномалия вставки.*

В отношение **СОТРУДНИКИ** нельзя вставить данные о сотруднике, который пока не участвует ни в одном проекте. Действительно, если, например, во втором отделе появляется новый сотрудник, скажем, Пушников, и он пока не участвует ни в одном проекте, то мы должны вставить в отношение кортеж (4, Пушников, 2, 33-22-11, null, null, null). Это сделать невозможно, т.к. атрибут ***Н_ПРО*** (номер проекта) входит в

состав потенциального ключа, и, следовательно, не может содержать null-значений.

Точно также нельзя вставить данные о проекте, над которым пока не работает ни один сотрудник.

Причина аномалии - хранение в одном отношении разнородной информации (и о сотрудниках, и о проектах, и о работах по проекту).

Вывод - логическая модель данных неадекватна модели предметной области. База данных, основанная на такой модели, будет работать неправильно.

2. *Аномалии обновления*

Фамилии сотрудников, наименования проектов, номера телефонов повторяются во многих кортежах отношения. Поэтому если сотрудник меняет фамилию, или проект меняет наименование, или меняется номер телефона, то такие изменения необходимо *одновременно* выполнить во всех местах, где эта фамилия, наименование или номер телефона встречаются, иначе отношение станет некорректным (например, один и тот же проект в разных кортежах будет называться по-разному). Таким образом, обновление базы данных одним действием реализовать невозможно. Для поддержания отношения в целостном состоянии необходимо написать триггер, который при обновлении одной записи корректно исправлял бы данные и в других местах.

Причина аномалии - избыточность данных, также порожденная тем, что в одном отношении хранится разнородная информация.

Вывод - увеличивается сложность разработки базы данных. База данных, основанная на такой модели, будет работать правильно только при наличии дополнительного программного кода в виде триггеров.

3. Аномалии удаления

При удалении некоторых данных может произойти потеря другой информации. Например, если закрыть проект "Космос" и удалить все строки, в которых он встречается, то будут потеряны все данные о сотруднике Петрове. Если удалить сотрудника Сидорова, то будет потеряна информация о том, что в отделе номер 2 находится телефон 33-22-11. Если по проекту временно прекращены работы, то при удалении данных о работах по этому проекту будут удалены и данные о самом проекте (наименование проекта). При этом если был сотрудник, который

работал только над этим проектом, то будут потеряны и данные об этом сотруднике.

Причина аномалии - хранение в одном отношении разнородной информации (и о сотрудниках, и о проектах, и о работах по проекту).

Вывод - логическая модель данных неадекватна модели предметной области. База данных, основанная на такой модели, будет работать неправильно.

Последовательное выполнение следующих шагов позволит таблицу привести в вид отношения, отвечающего требованиям 1НФ:

1. Устраните повторяющиеся группы в отдельных таблицах (одинаковые строки).
2. Создайте отдельную таблицу для каждого набора связанных данных.
3. Идентифицируйте каждый набор связанных данных с помощью первичного ключа.

Обозначим функциональные зависимости полученного отношения:

Зависимость атрибутов от ключа отношения:

{Н_СОТР, Н_ПРО} $\rightarrow$**ФАМ**

{Н_СОТР, Н_ПРО} $\rightarrow$**Н_ОТД**

{Н_СОТР, Н_ПРО} $\rightarrow$**ТЕЛ**

{Н_СОТР, Н_ПРО} $\rightarrow$**ПРОЕКТ**

{Н_СОТР, Н_ПРО} $\rightarrow$**Н_ЗАДАН**

Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника:

Н_СОТР $\rightarrow$**ФАМ**

Н_СОТР $\rightarrow$**Н_ОТД**

Н_СОТР $\rightarrow$**ТЕЛ**

Зависимость наименования проекта от номера проекта:

Н_ПРО $\rightarrow$**ПРОЕКТ**

Зависимость номера телефона от номера отдела:

Н_ОТД $\rightarrow$**ТЕЛ**

Замечание. Функциональная зависимость - *семантическое понятие*. Она возникает, когда по значениям одних данных в предметной области можно определить значения других данных. Например, зная табельный номер сотрудника, можно определить его фамилию, по номеру отдела можно определить телефона. Функциональная зависимость *задает дополнительные ограничения* на данные, которые могут храниться в отношениях. Для корректности базы данных (адекватности предметной области) необходимо при выполнении операций модификации базы данных проверять все ограничения, определенные функциональными зависимостями.

Функциональная зависимость атрибутов утверждает лишь то, что для каждого конкретного состояния базы данных по значению одного атрибута (детерминанта) можно однозначно определить значение другого атрибута (зависимой части). Но конкретные значение зависимой части могут быть различны в различных состояниях базы данных.