КОНСПЕКТ ЗАНЯТИЯ № 2 ЧЕТВЕРТОЙ НЕДЕЛИ КУРСА «БАЗЫ ДАННЫХ»

1.2. ВТОРАЯ НОРМАЛЬНАЯ ФОРМА[1]

Отношение R находится во ***второй нормальной форме*** (***2НФ***) тогда и только тогда, когда отношение находится в 1НФ и *нет неключевых атрибутов, зависящих от части сложного ключа*. ***Неключевой атрибут*** - это атрибут, не входящий в состав никакого потенциального ключа.

Замечание. Если потенциальный ключ отношения является простым, то отношение автоматически находится в 2НФ.

Отношение **СОТРУДНИКИ** не находится в 2НФ, т.к. есть атрибуты, зависящие от части сложного ключа {*Н_СОТР, Н_ПРО*}:

Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника является зависимостью от части *сложного ключа*:

Н_СОТР →**ФАМ**
Н_СОТР →**Н_ОТД**
Н_СОТР →**ТЕЛ**

Зависимость наименования проекта от номера проекта является зависимостью от части *сложного ключа*:

Н_ПРО →**ПРОЕКТ**

Для того, чтобы устранить зависимость атрибутов от части сложного ключа, нужно произвести ***декомпозицию*** отношения на несколько отношений. При этом *те атрибуты, которые зависят от части сложного ключа,* выносятся в отдельное отношение.

Формальное объяснение:

Процесс разбиения отношения R{A,B,C} на два отношения R1{A,B}, R2{A,C} называется ***проецированием****, а отношения R1 и R2 –* ***проекциями.*** *Здесь A, B и C – это некоторые непересекающиеся подмножества атрибутов исходного отношения, объединение которых*

[1] Раздел подготовлен и разработан в соавторстве со студентом 2 курса направления Информатика и вычислительная техника – Куницыной Софией.

даст все множество атрибутов. Если была ***произведена декомпозиция*** *без потерь, то соединение* ***проекций*** *R1 и R2 должно дать исходное отношение R.*

При проведении декомпозиции часто опираются на теорему Хеза (I.J. Heath):

“пусть R{A,B,C} является реляционным отношением, где A, B, C – атрибуты (возможно, составные) этого отношения. Если R удовлетворяет зависимости A → B, то R равно соединению своих проекций на {A,B} и {A,C}.”

Отношение **СОТРУДНИКИ** декомпозируем на три отношения:

1. **СОТРУДНИКИ_ОТДЕЛЫ**,
2. **ПРОЕКТЫ**,
3. **ЗАДАНИЯ**.

Отношение **СОТРУДНИКИ_ОТДЕЛЫ** (***Н_СОТР***, **ФАМ**, **Н_ОТД**, **ТЕЛ**)

Функциональные зависимости:

Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника:

Н_СОТР →**ФАМ**

Н_СОТР →**Н_ОТД**

Н_СОТР →**ТЕЛ**

Зависимость номера телефона от номера отдела:

Н_ОТД →**ТЕЛ**

Н_СОТР	**ФАМ**	**Н_ОТД**	**ТЕЛ**
1	Иванов	1	11-22-33
2	Петров	1	11-22-33
3	Сидоров	2	33-22-11

Таблица 2 Отношение СОТРУДНИКИ_ОТДЕЛЫ

Отношение **ПРОЕКТЫ** (***Н_ПРО***, **ПРОЕКТ**)

Функциональные зависимости:

Н_ПРО →**ПРОЕКТ**

Н_ПРО	**ПРОЕКТ**
1	Космос
2	Климат

Таблица 3 Отношение ПРОЕКТЫ

Отношение **ЗАДАНИЯ** (***Н_СОТР***, ***Н_ПРО***, **Н_ЗАДАН**):

Функциональные зависимости:

{*Н_СОТР*, *Н_ПРО*} →**Н_ЗАДАН**

Н_СОТР	***Н_ПРО***	**Н_ЗАДАН**
1	*1*	1
1	*2*	1
2	*1*	2
3	*1*	3
3	*2*	2

Таблица 4 Отношения ЗАДАНИЯ

Анализ декомпозированных отношений

Отношения, полученные в результате декомпозиции, находятся в 2НФ. Действительно, отношения **СОТРУДНИКИ_ОТДЕЛЫ** и **ПРОЕКТЫ** имеют простые ключи, следовательно автоматически находятся в 2НФ, отношение **ЗАДАНИЯ** имеет сложный ключ, но единственный неключевой атрибут **Н_ЗАДАН** функционально зависит от всего ключа **{*Н_СОТР*, *Н_ПРО*}**, а не от его части.

Часть аномалий обновления устранена. Так, данные о сотрудниках и проектах теперь хранятся в различных отношениях, поэтому при появлении сотрудников, не участвующих ни в одном проекте просто добавляются кортежи в отношение **СОТРУДНИКИ_ОТДЕЛЫ**. Точно также, при появлении проекта, над которым не работает ни один сотрудник, просто вставляется кортеж в отношение **ПРОЕКТЫ**.

Фамилии сотрудников и наименования проектов теперь хранятся без избыточности. Если сотрудник сменит фамилию или проект сменит наименование, то такое обновление будет произведено в одном месте.

Если по проекту временно прекращены работы, но требуется, чтобы сам проект сохранился, то для этого проекта удаляются соответствующие кортежи в отношении **ЗАДАНИЯ**, а данные о самом проекте и данные о сотрудниках, участвовавших в проекте, остаются в отношениях **ПРОЕКТЫ** и **СОТРУДНИКИ_ОТДЕЛЫ**.

Тем не менее, часть аномалий разрешить не удалось.

Оставшиеся аномалии вставки

В отношение **СОТРУДНИКИ_ОТДЕЛЫ** нельзя вставить кортеж (4, Пушников, 1, 33-22-11), т.к. при этом получится, что два сотрудника из 1-го отдела (Иванов и Пушников) имеют разные номера телефонов, а это противоречит модели предметной области. В этой ситуации можно предложить два решения, в зависимости от того, что реально произошло в предметной области. Другой номер телефона может быть введен по двум причинам - по ошибке человека, вводящего данные о новом сотруднике, или потому что номер в отделе действительно изменился. Тогда можно написать триггер, который при вставке записи о сотруднике проверяет, совпадает ли телефон с уже имеющимся телефоном у другого сотрудника этого же отдела. Если номера отличаются, то система должна задать вопрос, оставить ли старый номер в отделе или заменить его новым. Если нужно оставить старый номер (новый номер введен ошибочно), то кортеж с данными о новом сотруднике будет вставлен, но номер телефона у него будет тот, который уже есть в отделе (в данном случае, 11-22-33). Если же номер в отделе действительно изменился, то кортеж будет вставлен с новым номером, и одновременно будут изменены номера телефонов у всех сотрудников этого же отдела. И в том и в другом случае не обойтись без разработки громоздкого триггера (о триггерах еще поговорим).

Причина аномалии - избыточность данных, порожденная тем, что в одном отношении хранится разнородная информация (о сотрудниках и об отделах).

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

Оставшиеся аномалии обновления

Одни и те же номера телефонов повторяются во многих кортежах отношения. Поэтому если в отделе меняется номер телефона, то такие изменения необходимо одновременно выполнить во всех местах, где этот номер телефона встречаются, иначе отношение станет некорректным. Таким образом, обновление базы данных одним действием реализовать невозможно. Необходимо написать триггер, который при обновлении одной записи корректно исправляет номера телефонов 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.

Оставшиеся аномалии удаления

При удалении некоторых данных по-прежнему может произойти потеря другой информации. Например, если удалить сотрудника Сидорова, то будет потеряна информация о том, что в отделе номер 2 находится телефон 33-22-11.

Причина аномалии - хранение в одном отношении разнородной информации (и о сотрудниках, и об отделах).

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

Заметим, что при переходе ко второй нормальной форме отношения стали *почти* адекватными предметной области. Остались также трудности в разработке базы данных, связанные с необходимостью написания триггеров, поддерживающих целостность базы данных. Эти трудности теперь связаны только с одним отношением **СОТРУДНИКИ_ОТДЕЛЫ**.

1.3. ТРЕТЬЯ НОРМАЛЬНАЯ ФОРМА[2]

Атрибуты называются ***взаимно независимыми***, если ни один из них не является функционально зависимым от другого.

Отношение R находится в ***третьей нормальной форме*** (***3НФ***) тогда и только тогда, когда отношение находится в 2НФ и *все неключевые атрибуты взаимно независимы*.

Нужно отметить, что если существует ФЗ между неключевыми атрибутами, а детерминант, в свою очередь, будет зависеть от первичного ключа, мы получим транзитивную ФЗ. Иными словами, если в отношении есть только потенциальный ключ и можно выделить транзитивные ФЗ, то это указывает, что отношение не соответствует 3НФ.
Напомним запись транзитивности: $A \rightarrow B$ и $B \rightarrow C$, то $A \rightarrow C$.

Отношение **СОТРУДНИКИ_ОТДЕЛЫ** не находится в 3НФ, т.к. имеется функциональная зависимость неключевых атрибутов (зависимость номера телефона от номера отдела): **Н_ОТД →ТЕЛ**

Для того чтобы устранить зависимость неключевых атрибутов, нужно произвести декомпозицию отношения на несколько отношений. При этом *те неключевые атрибуты, которые являются зависимыми (транзитивными)*, выносятся в отдельное отношение.

Отношение **СОТРУДНИКИ_ОТДЕЛЫ** декомпозируем на два отношения - **СОТРУДНИКИ**, **ОТДЕЛЫ**.

Отношение **СОТРУДНИКИ** (***Н_СОТР***, **ФАМ**, **Н_ОТД**):

Функциональные зависимости:

Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника:

Н_СОТР →**ФАМ**
Н_СОТР →**Н_ОТД**
Н_СОТР →**ТЕЛ**

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

Н_СОТР	**ФАМ**	**Н_ОТД**
1	Иванов	1
2	Петров	1
3	Сидоров	2

Таблица 5 Отношение СОТРУДНИКИ

Отношение **ОТДЕЛЫ** (***Н_ОТД***, **ТЕЛ**):

Функциональные зависимости:

Зависимость номера телефона от номера отдела:

Н_ОТД $\rightarrow$**ТЕЛ**

Н_ОТД	**ТЕЛ**
1	11-22-33
2	33-22-11

Таблица 6 Отношение ОТДЕЛЫ

Обратим внимание на то, что атрибут **Н_ОТД**, *не являвшийся ключевым* в отношении **СОТРУДНИКИ_ОТДЕЛЫ,** *становится потенциальным ключом* в отношении **ОТДЕЛЫ**. Именно за счет этого устраняется избыточность, связанная с многократным хранением одних и тех же номеров телефонов.

Вывод. Таким образом, все обнаруженные аномалии обновления устранены. Реляционная модель, состоящая из четырех отношений **СОТРУДНИКИ**, **ОТДЕЛЫ**, **ПРОЕКТЫ**, **ЗАДАНИЯ**, находящихся в третьей нормальной форме, является адекватной описанной модели предметной области, и требует наличия только тех триггеров, которые поддерживают ссылочную целостность. Такие триггеры являются стандартными и не требуют больших усилий в разработке.

Итак, представим **алгоритм нормализации** (т.е. алгоритм приведения отношений к 3НФ):

- *Шаг 1 (Приведение к 1НФ).* На первом шаге задается одно или несколько отношений, отображающих понятия предметной области. По модели предметной области выписываются обнаруженные функциональные зависимости. Все отношения автоматически находятся в 1НФ.

- *Шаг 2 (Приведение к 2НФ).* Если в некоторых отношениях обнаружена зависимость атрибутов от части сложного ключа, то проводим декомпозицию этих отношений на несколько отношений следующим образом: те атрибуты, которые зависят от части сложного ключа выносятся в отдельное отношение вместе с этой частью ключа. В исходном отношении остаются все ключевые атрибуты:

Исходное отношение: $R(K_1, K_2, A_1, .., A_n, B_1, ..., B_m)$.

Ключ: (K_1, K_2) - сложный.

Функциональные зависимости:

$(K_1, K_2) \to (A_1, .., A_n, B_1, ... B_m)$ - зависимость всех атрибутов от ключа отношения.

$(K_1) \to (A_1, .., A_n)$ - зависимость некоторых атрибутов от части сложного ключа.

Декомпозированные отношения:

$R_1(K_1, K_2, B_1, ..., B_m)$ - остаток от исходного отношения. Ключ (K_1, K_2).

$R_2(K_1, A_1, .., A_n)$ - атрибуты, вынесенные из исходного отношения вместе с частью сложного ключа. **Ключ** K_1.

- *Шаг 3 (Приведение к 3НФ).* Если в некоторых отношениях обнаружена зависимость некоторых неключевых атрибутов от других неключевых атрибутов, то проводим декомпозицию этих отношений следующим образом: те неключевые атрибуты, которые зависят от других неключевых атрибутов выносятся в отдельное отношение. В новом отношении ключом становится детерминант функциональной зависимости:

Исходное отношение: $R(K, A_1, .., A_n, B_1, ..., B_m)$.

Ключ: K.

Функциональные зависимости:

$K \to (A_1, .., A_n, B_1, ..., B_m)$ - зависимость всех атрибутов от ключа отношения.

$(A_1, .., A_n) \to (B_1, ..., B_m)$ - зависимость некоторых неключевых атрибутов других неключевых атрибутов.

Декомпозированные отношения:

$R_1(K, A_1, .., A_n)$ - остаток от исходного отношения. Ключ K.

$R_2(A_1, .., A_n, B_1, ..., B_m)$ - атрибуты, вынесенные из исходного отношения вместе с детерминантом функциональной зависимости**. Ключ** $(A_1, .., A_n)$.

Замечание. На практике, при создании логической модели данных, как правило, не следуют прямо приведенному алгоритму нормализации. Опытные разработчики обычно сразу строят отношения в 3НФ. Кроме того, основным средством разработки логических моделей данных являются различные варианты ER-диаграмм. Особенность этих диаграмм в том, что они сразу позволяют создавать отношения в 3НФ. Тем не менее, приведенный алгоритм важен по двум причинам. *Во-первых*, этот алгоритм показывает, какие проблемы возникают при разработке слабо нормализованных отношений. *Во-вторых*, как правило, модель предметной области никогда не бывает правильно разработана с первого шага. Эксперты предметной области могут забыть о чем-либо упомянуть, разработчик может неправильно понять эксперта, во время разработки могут измениться правила, принятые в предметной области, и т.д. Все это может привести к появлению новых зависимостей, которые отсутствовали в первоначальной модели предметной области. Тут как раз и необходимо использовать алгоритм нормализации хотя бы для того, чтобы убедиться, что отношения остались в 3НФ и логическая модель не ухудшилась.