ТЕМА 8. НОРМАЛЬНЫЕ ФОРМЫ ВЫСОКИХ ПОРЯДКОВ

Многозначные зависимости и четвертая нормальная форма

Чтобы перейти к вопросам дальнейшей нормализации, рассмотрим еще одну возможную (четвертую) интерпретацию переменной отношения СЛУЖ_ПРО_ЗАДАН. Предположим, что каждый служащий может участвовать в нескольких проектах, но в каждом проекте, в котором он участвует, им должны выполняться одни и те же задания. Возможное значение четвертого варианта переменной отношения СЛУЖ_ПРО_ЗАДАН показано на рис. 1.

СЛУ_НОМ	ПРО_НОМ	СЛУ_ЗАДАН
2934	1	A
2934	1	B
2934	2	A
2934	2	B
....	....	....
2941	1	A
2941	1	D

Рис. 1. Возможное значение переменной отношения СЛУЖ_ПРО_ЗАДАН

Аномалии обновлений при наличии многозначных зависимостей и возможная декомпозиция

В новом варианте переменной отношения единственным возможным ключом является заголовок отношения {СЛУ_НОМ, ПРО_НОМ, СЛУ_ЗАДАН}. Кортеж <сн, пн, сз> входит в тело отношения в том и только в том случае, когда служащий с номером сн выполняет в проекте пн задание сз. Поскольку для каждого служащего указываются все проекты, в которых он участвует, и все задания, которые он должен выполнять в этих проектах, для каждого допустимого значения переменной отношения СЛУЖ_ПРО_ЗАДАН должно выполняться следующее ограничение (ТСПЗобозначает тело отношения):

IF (<сн, $пн_1$, $сз_1$> ТСПЗ AND <сн, $пн_2$, $сз_2$> ТСПЗ)
THEN (<сн, $пн_1$, $сз_2$> ТСПЗ AND <сн, $пн_2$, $сз_1$> ТСПЗ)

Наличие такого ограничения приводит к тому, что при работе с отношением СЛУЖ_ПРО_ЗАДАНпроявляются аномалии обновления.

- **Добавление кортежа.** Если уже участвующий в проектах служащий присоединяется к новому проекту, то к телу значения переменной отношения СЛУЖ_ПРО_ЗАДАН требуется добавить столько кортежей, сколько заданий выполняет этот служащий.
- **Удаление кортежей.** Если служащий прекращает участие в проектах, то отсутствует возможность сохранить данные о заданиях, которые он может выполнять.
- **Модификация кортежей.** При изменении одного из заданий служащего необходимо изменить значение атрибута СЛУ_ЗАДАН в стольких кортежах, в скольких проектах участвует служащий.

Трудности, связанные с обновлением переменной отношения СЛУЖ_ПРО_ЗАДАН, решаются путем его декомпозиции на две переменных отношений: СЛУЖ_ПРО_НОМ {СЛУ_НОМ, ПРО_НОМ} и СЛУЖ_ЗАДАНИЕ {СЛУ_НОМ, СЛУ_ЗАДАН}. Значения этих переменных отношений, соответствующие значению переменной отношения СЛУЖ_ПРО_ЗАДАН с рис. 1, показаны на рис. 2.

Легко видеть, что декомпозиция, представленная на рис. 2, является декомпозицией без потерь и что эта декомпозиция решает перечисленные выше проблемы с обновлением переменной отношения СЛУЖ_ПРО_ЗАДАН.

Значение переменной отношения **СЛУЖ_ПРО_НОМ**

СЛУ_НОМ	ПРО_НОМ
2934	1
2934	2
....	....
2941	1

Значение переменной отношения **СЛУЖ_ЗАДАНИЕ**

СЛУ_НОМ	СЛУ_ЗАДАН
2934	A
2934	B
....	....
2941	A
2941	D

Рис. 2. Значения переменных отношений СЛУЖ_ПРО_НОМ и СЛУЖ_ЗАДАНИЕ

- **Добавление кортежа.** Если некоторый уже участвующий в проектах служащий присоединяется к новому проекту, то к телу значения переменной отношения СЛУЖ_ПРО_НОМ требуется добавить один кортеж, соответствующий новому проекту.
- **Удаление кортежей.** Если служащий прекращает участие в проектах, то данные о заданиях, которые он может выполнять, остаются в отношении СЛУЖ_ЗАДАНИЕ.
- **Модификация кортежей.** При изменении одного из заданий служащего необходимо изменить значение атрибута СЛУ_ЗАДАН в одном кортеже отношения СЛУЖ_ЗАДАНИЕ.

Многозначные зависимости. Теорема Фейджина. Четвертая нормальная форма

Заметим, что последний вариант переменной отношения СЛУЖ_ПРО_ЗАДАН находится в BCNF, поскольку все атрибуты заголовка отношения входят в состав единственно возможного ключа. В этом отношении вообще отсутствуют нетривиальные FD. Поэтому ранее обсуждавшиеся принципы нормализации здесь неприменимы, но, тем не менее, мы получили полезную декомпозицию. Все дело в том, что в случае четвертого варианта отношения СЛУЖ_ПРО_ЗАДАН мы имеем дело с новым видом зависимости, впервые обнаруженным Роном Фейджином в 1971 г. Фейджин назвал зависимости этого вида многозначными (multi-valued dependency – MVD).

В отношении СЛУЖ_ПРО_ЗАДАН выполняются две MVD: СЛУ_НОМ$\rightarrow\!\!\!\rightarrow$ПРО_НОМ и СЛУ_НОМ$\rightarrow\!\!\!\rightarrow$СЛУ_ЗАДАН. Первая MVD означает, что каждому значению атрибута СЛУ_НОМ соответствует определяемое только этим значением множество значений атрибута ПРО_НОМ. Другими словами, в результате вычисления алгебраического выражения

```
(СЛУЖ_ПРО_НОМ WHERE (СЛУ_НОМ = сн AND СЛУ_ЗАДАН = сз)) PROJECT {ПРО_НОМ}
```

для фиксированного допустимого значения сн и любого допустимого значения сз мы всегда получим одно и то же множество значений атрибута ПРО_НОМ. Аналогично трактуется вторая MVD.

В переменной отношения r с атрибутами A, B, C (в общем случае, составными) имеется многозначная зависимость B от A (A$\rightarrow\!\!\!\rightarrow$B) в том и только в том случае, когда множество значений атрибута B, соответствующее паре значений атрибутов A и C, зависит от значения A и не зависит от значения C.

Многозначные зависимости обладают интересным свойством "двойственности", которое демонстрирует следующая лемма.

Теорема Фейджина

Пусть имеется переменная отношения `r` с атрибутами `A`, `B`, `C` (в общем случае, составными). Отношение `r` декомпозируется без потерь на проекции `{A, B}` и `{A, C}` тогда и только тогда, когда для него выполняется MVD $A \twoheadrightarrow B \mid C$.

Теорема Фейджина обеспечивает основу для декомпозиции отношений, удаляющей "аномальные" многозначные зависимости, с приведением отношений в четвертую нормальную форму.

Переменная отношения r находится в четвертой нормальной форме (4NF) в том и только в том случае, когда она находится в BCNF, и все MVD r являются FD с детерминантами – возможными ключами отношения r.

Вопрос о необходимости нормализации

Процесс проектирования реляционной базы на основе метода нормализации преследует две основных цели:

- избежать избыточности хранения данных;
- устранить аномалии обновления отношений.

Рассмотрим, насколько эти цели актуальны в современных условиях, когда объемы доступных носителей внешней памяти непрерывно возрастают, стоимость их падает, а современные серверы реляционных баз данных способны автоматически поддерживать целостность баз данных. Здесь следует отметить два важных обстоятельства.

Во-первых, теория реляционных баз данных и методы их проектирования активно развивались уже более 25 лет тому назад. Ситуация в области технологии аппаратуры и программного обеспечения тогда была совсем иной, чем сегодня, и хорошо нормализованные реляционные базы данных в значительной степени способствовали росту эффективности приложений.

Во-вторых, в то время реляционные базы преимущественно использовались в информационных системах оперативной обработки транзакций (On-Line Transaction Processing – OLTP). Системам категории OLTP свойственны частые обновления базы данных, поэтому аномалии обновлений, даже если их корректировка производится СУБД автоматически, могут заметно снижать эффективность приложения.

Сегодня на переднем крае приложений баз данных находятся системы категории оперативной аналитической обработки (On-Line Analytical Processing – OLAP). В подобных системах, в частности, системах поддержки принятия решений, базы данных в основном используются для выборки данных, поэтому аномалиями обновлений можно пренебречь, а объем этих баз настолько огромен, что можно пренебречь и избыточностью хранения.

Значит ли это, что подход к проектированию реляционных баз данных методом нормализации утратил свою роль? Нет!

Мир приложений баз данных в настоящее время огромен. Сегодня любое мало-мальски приличное предприятие использует хотя бы одно приложение баз данных – бухгалтерские, складские, кадровые системы. Это системы категории OLTP с частым обновлением данных и умеренными запросами к базе данных, не вызывающими соединений многих отношений. Для небольших компаний равно важны как эффективность информационных систем, так и стоимость используемых аппаратно-программных средств. Правильно спроектированные, хорошо нормализованные реляционные базы данных помогают решению корпоративных проблем.

Да, любое правильно развивающееся предприятие рано или поздно приходит к использованию систем категории OLAP, например, некоторой разновидности систем поддержки принятия решений (Decision Support System – DSS). В базах данных таких систем обновления очень редки, а запросы могут иметь произвольную сложность, включая соединения многих отношений. Но, во-первых, технологически правильно для системы OLAP поддерживать отдельную базу данных (обычно подобные базы данных называют **хранилищами данных – DataWarehouse**), а во-вторых, основными

источниками данных для построения таких хранилищ данных являются базы данных систем OLTP. Так что актуальность правильно спроектированных баз данных OLTP-систем не уменьшается, а постоянно возрастает.

Следует ли из этого, что принципы нормализации непригодны для проектирования баз данных OLAP-приложений? И снова в ответ категорическое НЕТ! Возможно, окончательная схема такой базы данных должна быть денормализована из соображений повышения эффективности выполнения запросов. Но чтобы получить правильную денормализованную схему, нужно сначала понять, как выглядит нормализованная схема.

Основной вывод можно сформулировать следующим образом: пока мы остаемся в мире реляционных баз данных, для правильного проектирования базы данных необходимо понимать принципы нормализации, воспринимая их не как догму, а как руководство к действию.