Нормализация отношений

Одно из назначений базы данных - предоставление информации пользователям. Информация извлекается из реляционной базы данных при помощи оператора SQL - SELECT. Одной из наиболее дорогостоящих операций при выполнении оператора SELECT является операция соединение таблиц. Таким образом, чем больше взаимосвязанных отношений было создано в ходе логического моделирования, тем больше вероятность того, что при выполнении запросов эти отношения будут соединяться, и, следовательно, тем медленнее будут выполняться запросы. Таким образом, увеличение количества отношений приводит к замедлению выполнения операций выборки данных, особенно, если запросы заранее неизвестны.

Пример

Рассмотрим в качестве предметной области некоторую организацию, выполняющую некоторые проекты. Модель предметной области опишем следующим неформальным текстом:

1. Сотрудники организации выполняют проекты.
2. Проекты состоят из нескольких заданий.
3. Каждый сотрудник может участвовать в одном или нескольких проектах, или временно не участвовать ни в каких проектах.
4. Над каждым проектом может работать несколько сотрудников, или временно проект может быть приостановлен, тогда над ним не работает ни один сотрудник.
5. Над каждым заданием в проекте работает ровно один сотрудник.
6. Каждый сотрудник числится в одном отделе.
7. Каждый сотрудник имеет телефон, находящийся в отделе сотрудника.

В ходе дополнительного уточнения того, какие данные необходимо учитывать, выяснилось следующее:

1. О каждом сотруднике необходимо хранить табельный номер и фамилию. Табельный номер является уникальным для каждого сотрудника.
2. Каждый отдел имеет уникальный номер.
3. Каждый проект имеет номер и наименование. Номер проекта является уникальным.
4. Каждая работа из проекта имеет номер, уникальный в пределах проекта. Работы в разных проектах могут иметь одинаковые номера.

Создадим, например, такую таблицу и назовем ее ФИРМА

Н_ сотрудника	**Фамилия**	**Должность**	**Н_ комнаты**	**Назв_отдела**	**Телефон**	**Н_ проекта**	**Назв_ проекта**	**Дата_ начала**	**Дата_ сдачи**	**Задание**
1	Иванов	Бухгалтер	1	Бухгалтерия	11-11-11	*1*	Сайт фирмы «Фотек»	1.04.06	1.07.06	Расчет затрат
1	Иванов	Бухгалтер	1	Бухгалтерия	11-11-11	*2*	ЭС «Спец»	20.06.06	2.12.06	Расчет затрат
2	Петров	Ведущий программист	2	Отдел разработки	22-22-22	*2*	ЭС «Спец»	1.04.06	1.07.06	Управление проектом
3	Сидоров	Тестировщик	3	Вычислительный центр	33-33-33	*1*	Сайт фирмы «Фотек»	1.04.06	1.07.06	Тестирование
3	Сидоров	Тестировщик	3	Вычислительный центр	33-33-33	*2*	ЭС «Спец»	20.06.06	2.12.06	Тестирование

Даже одного взгляда на таблицу отношения **Фирма** достаточно, чтобы увидеть, что данные хранятся в ней с большой избыточностью. Во многих строках повторяются фамилии сотрудников, номера 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.

Исторически эти проблемы получили название ***аномалии***. В общем, **аномалии БД** - противоречие между моделью предметной области и физической моделью данных, поддерживаемых средствами конкретной СУБД. Аномалии возникают в том случае, когда есть либо *неадекватность модели данных предметной области*, либо некоторые *дополнительные трудности в реализации ограничений предметной области* средствами СУБД. Таким образом, мы будем придерживаться понятия аномалии как неадекватности модели данных предметной области, (что говорит на самом деле о том, что логическая модель данных попросту неверна!) или как необходимости дополнительных усилий для реализации всех ограничений определенных в предметной области (дополнительный программный код в виде триггеров или хранимых процедур).

Т.к. аномалии проявляют себя при выполнении операций, изменяющих состояние базы данных, то различают следующие виды аномалий:

- аномалии вставки (INSERT);
- аномалии обновления (UPDATE);
- аномалии удаления (DELETE).

В отношении **Фирма** можно привести примеры следующих аномалий

Аномалии вставки (INSERT)

В отношение **Фирма** нельзя вставить данные о сотруднике, который пока не участвует ни в одном проекте. Действительно, если, например, во второй комнате появляется новый сотрудник, скажем, Сергеев, и он пока не участвует ни в одном проекте, то мы должны вставить в отношение кортеж *(4, Сергее, Программист, 2, Отдел разработки, 22-22-22, null, null, null, null, null)*. Это сделать невозможно, т.к. атрибут **Н_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.

Аномалии обновления (UPDATE)

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.

Аномалии удаления (DELETE)

При удалении некоторых данных может произойти потеря другой информации. Например, если закрыть проект «ЭС «Спец» и удалить все строки, в которых он встречается, то будут потеряны все данные о сотруднике Петрове. Если удалить сотрудника Петрова, то будет потеряна информация о комнате № 2 и о том, что в комнате номер 2 находится телефон 22-22-22. Если по проекту временно прекращены работы, то при удалении данных о работах по этому проекту будут удалены и данные о самом проекте (наименование проекта, даты его выполнения).

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.

1.1 **1НФ (Первая Нормальная Форма)**

Первая нормальная форма (***1НФ***) - это обычное отношение. Согласно нашему определению отношений, любое отношение автоматически уже находится в 1НФ. Напомним кратко свойства отношений (это и будут свойства 1НФ):

- В отношении нет одинаковых кортежей.
- Кортежи не упорядочены.
- Атрибуты не упорядочены и различаются по наименованию.
- Все значения атрибутов атомарны.

В ходе логического моделирования на первом шаге предложено хранить данные в одном отношении, имеющем следующие атрибуты:

СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ (***Н_СОТР***, **ФАМ**, **Н_ОТД**, **ТЕЛ**, ***Н_ПРО***, **ПРОЕКТ**, **Н_ЗАДАН**)

где

Н_СОТР - табельный номер сотрудника

ФАМ - фамилия сотрудника

Н_ОТД - номер отдела, в котором числится сотрудник

ТЕЛ - телефон сотрудника

Н_ПРО - номер проекта, над которым работает сотрудник

ПРОЕКТ - наименование проекта, над которым работает сотрудник

Н_ЗАДАН - номер задания, над которым работает сотрудник

Т.к. каждый сотрудник в каждом проекте выполняет ровно одно задание, то в качестве потенциального ключа отношения необходимо взять пару атрибутов {***Н_СОТР***, ***Н_ПРО***}.

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.

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

1.2 **2НФ (Вторая Нормальная Форма)**

Отношение R находится во *второй нормальной форме* (*2НФ*) тогда и только тогда, когда отношение находится в 1НФ и *нетнеключевых 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_ОТДЕЛЫ_ПРОЕКТЫ не находится в 2НФ, т.к. есть атрибуты, зависящие от части сложного ключа:

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.

Отношение СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ декомпозируем на три отношения - СОТРУДНИКИ_ОТДЕЛЫ, ПРОЕКТЫ, 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

Отношение ПРОЕКТЫ (*Н_ПРО*, ПРОЕКТ):

Функциональные зависимости:

Н_ПРО → ПРОЕКТ

Н_ПРО	ПРОЕКТ

1	Космос
2	Климат

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

1.3 **3НФ (Третья Нормальная Форма)**

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*.

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,* 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

Н_ОТД	ТЕЛ
1	11-22-33
2	33-22-11

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

Отношение ОТДЕЛЫ (*Н_ОТД*, ТЕЛ):
Функциональные зависимости:
Зависимость номера телефона от номера отдела:
Н_ОТД → ТЕЛ

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.

1.4 **OLTP и OLAP-системы**

Сильно нормализованные модели данных хорошо подходят для OLTP-приложений – *On-LineTransactionProcessing*(OLTP) – приложений оперативной обработки транзакций. Типичными примерами OLTP-приложений являются системы складского учета, заказов билетов, операционные банковские системы и другие. Основная функция подобных систем заключается в выполнении большого количества коротких транзакций. Сами транзакции являются достаточно простыми, но проблемы состоят в том, что таких транзакций очень много, выполняются они одновременно и при возникновении ошибок транзакция должна откатиться и вернуть систему в состояние, в котором та была до начала транзакции. Практически все запросы к базе данных в OLTP-приложениях состоят из команд вставки, обновления и удаления. Запросы на выборку, в основном, предназначены для предоставления пользователям выборки данных из различного рода справочников. Таким образом, большая часть запросов известна заранее ещё на этапе проектирования системы. Критическим для OLTP-приложений является скорость и надежность выполнения коротких операций обновления данных. Чем выше уровень нормализации данных в OLTP-приложениях, тем оно быстрее и надежней. Отступления от этого правила могут происходить тогда, когда уже на этапе разработки известны некоторые часто возникающие запросы, требующие соединения отношений и от скорости выполнения которых существенно зависит работа приложений.

Другим типом приложений являются OLAP-приложения – *On-LineAnaliticalProcessing*(OLAP) – приложения оперативной аналитической обработки данных. Это обобщенный термин, характеризующий принципы построения систем поддержки принятия решений – DecisionSupportSystem (DSS), хранилищ данных – DataWarehouse, систем интеллектуального анализа данных – DataMining. Такие системы предназначены для нахождения зависимостей между данными, для проведения динамического анализа по принципу «что если…» и тому подобных задач. OLAP-приложения оперируют с большими массивами данных, накопленными на предприятии или взятыми из других источников. Такие системы характеризуются следующими признаками:

- добавление в систему новых данных происходит относительно редко крупными блоками, например, один раз в месяц или квартал;
- данные, добавленные в систему, как правило, никогда не удаляются;
- перед загрузкой данные проходят различные подготовительные процедуры, связанные с приведением их к определенным форматам и тому подобное;
- запросы к системе являются нерегламентированными и достаточно сложными;
- скорость выполнения запросов важна, но не критична.

Характеристика	**OLTP**	**OLAP**
Назначение системы	Регистрация, оперативный поиск и обработка транзакций, регламентированный анализ	Работа с историческими данными, аналитическая обработка, прогнозирование, моделирование
Хранимые данные	Оперативные, детализированные	Охватывающие большой период времени, агрегированные
Тип данных	Структурированные	Разнотипные
"Возраст" данных	Текущие (несколько месяцев)	Исторические (за годы) и прогнозируемые
Частота обновления данных	Высокая, небольшими "порциями"	Малая, большими "порциями"
Уровень агрегации данных	Детализированные данные	В основном - агрегированные данные
Преобладающие операции	Ввод данных, поиск, обновление	Анализ данных
Способ использования данных	Предсказуемый	Непредсказуемый
Взаимодействие с пользователем	На уровне транзакции	На уровне всей базы данных
Вид деятельности	Оперативная, тактическая	Аналитическая, стратегическая
Приоритеты	Высокая производительность Высокая доступность	Гибкость Автономность пользователя
Категория пользователей	Большое количество работников исполнительного звена	Относительно малое количество работников руководящего звена