Московский государственный университет экономики, статистики и информатики (МЭСИ)

Методология проектирования БД (Этап - Физическое проектирование).

Начальник отдела НИЧ, к.э.н., доцент Д.Г. Корнеев

2010 год

Общие положения

Методология проектирования - структурированный подход, предусматривающий использование специализированных процедур, технических приемов, инструментов, документации и нацеленный на поддержку и упрощение процесса проектирования.

Методология проектирования предусматривает разбиение всего процесса на несколько фаз, каждая из которых, в свою очередь, состоит из нескольких этапов. На каждом этапе разработчику предлагается набор технических приемов, позволяющих решать задачи, стоящие перед ним на данной стадии разработки. *Кроме того, методология предлагает методы планирования, координации, управления, оценки хода разработки проекта, а также структурированный подход к анализу и моделированию всего набора предъявляемых к базе данных требований и позволяет выполнить эти действия стандартизированными организованным образом.*

Концептуальное проектирование

Этап 1 Концептуальное проектирование базы данных (инфологическое) **-** процедура конструирования информационной модели предприятия, *не зависящей от каких-либо физических условий реализации.*

Фаза концептуального проектирования базы данных начинается с создания концептуальной модели данных предприятия, полностью независимой от любых деталей реализации. *К последним относятся: выбранный тип СУБД, состав программ приложения, используемый язык программирования, конкретная вычислительная платформа и любые другие физические особенности реализации*.

Логическое проектирование

Этап 2 Логическое проектирование базы данных - процесс конструирования информационной модели предприятия на основе существующих конкретных моделей данных, не зависимой от используемой СУБД и прочих физических условий реализации.

Фаза логического проектирования базы данных заключается в преобразовании концептуальной модели данных в логическую модель данных предприятия с учетом выбранного ***типа*** СУБД (например, предполагается использование некоторой реляционной СУБД). *Логическая модель данных является источником информации для фазы физического проектирования.* Она предоставляет разработчику физической модели данных средства проведения всестороннего анализа различных аспектов работы с данными, что имеет исключительно важное значение для выбора действительно эффективного проектного решения.

Физическое проектирование

Этап 3 - Физическое проектирование базы данных - *процесс создания описания конкретной реализации базы данных, размещаемой во вторичной памяти.* Предусматривает описание структуры хранения данных и методов доступа, предназначенных для осуществления наиболее эффективного доступа к информации.

Фаза физического проектирования базы данных предусматривает принятие разработчиком окончательного решения о способах реализации создаваемой базы. Поэтому физическое проектирование обязательно производится с учетом всех особенностей используемой СУБД. Между фазами физического и логического проектирования всегда имеется определенная обратная связь, поскольку решения, принятые на этапе физического проектирования с целью повышения производительности разрабатываемой системы, могут потребовать некоторого пересмотра логической модели данных.

Физическое проектирование БД (общие положения)

Образно говоря, при логическом проектировании разработчик сосредоточивается на том, ***что надо сделать***, тогда как при физическом проектировании он ищет способ, ***как это сделать***. В каждом случае требуется наличие различных навыков. Так, специалист по физическому проектированию баз данных должен ясно представлять, как та или иная СУБД функционирует в компьютерной системе, а также хорошо знать все функциональные возможности целевой СУБД. Поскольку функциональные возможности различных СУБД достаточно сильно отличаются друг от друга, физическое проектирование всегда тесно связано с особенностями конкретной выбранной системы.

Физическое проектирование БД (общие положения)

Однако, этап физического проектирования базы данных *не является совершенно изолированным* от других - как правило, между логическим и физическим проектированием имеется постоянная обратная связь, часто охватывающая и разработку пользовательских приложений. *Например, решения, принятые на этапе физического проектирования с целью повышения производительности системы, могут влиять на структуру ее логической схемы.*

Этапы физического проектирования

Этап 4. Перенос глобальной логической модели данных в среду целевой СУБД.

Этап 4.1. Проектирование таблиц базы данных в среде целевой СУБД.

Этап 4.2. Реализация бизнес-правил предприятия в среде целевой СУБД.

Этап 5. Проектирование физического представления базы данных.

Этап 5.1. Анализ транзакций.

Этап 5.2. Выбор файловой структуры.

Этап 5.3. Определение вторичных индексов.

Этап 5.4. Анализ необходимости введения контролируемой избыточности данных.

Этап 5.5. Определение требований к дисковой памяти.

Этап 6. Разработка механизмов защиты.

Этап 6.1. Разработка пользовательских представлений (видов).

Этап 6.2. Определение прав доступа.

Этап 7. Организация мониторинга и настройка функционирования системы.

Этап 4. Перенос глобальной логической модели данных в среду целевой СУБД.

Теперь необходимо принять решение о способе реализации таблиц базы данных.

Это решение зависит от типа выбранной целевой СУБД - при определении таблиц базы данных и ограничений целостности одни СИСТЕМЫ предоставляют больше возможностей, другие - меньше. Чтобы проиллюстрировать этот процесс, рассмотрим три различных способа реализации таблиц и требований ссылочной целостности.

1. Описание на языке SQL стандарта 1992 (SQL2).

2. Реализация с использованием триггеров.

3. Реализация с использованием уникальных индексов..

Физическое проектирование
Механизм создания доменов

Создание доменов:

CREATE DOMAIN ***property_type*** AS CHAR(1)
CHECK(VALUE IN('B', 'C', 'O', 'E', 'F', 'M', 'S'));

CREATE DOMAIN ***property_rooms*** AS INTEGER
CHECK (VALUE BETWEEN 1 ANO 15);

CREATE DOMAIN ***property_rent*** AS DECIMAL(6,2)
CHECK(VALUE BETWEEN 0 AND 9999);

Физическое проектирование
Механизм создания доменов

```
CREATE TABLE property for rent(
pno     PROPERTY_NUMBER          NOT NULL,
street  STREET                   NOT NULL,
area    AREA,
city    CITY                     NOT NULL,
pcode POST COOE,
type    PROPERTY_TYPE             DEFAULT ‘F’,
rooms PROPERTY_ROOMS             DEFAULT 4,
rent    PROPERTY_RENT             DEFAULT 600,
ono     OWNER_NUMBER              NOT NULL,
sno     STAFF_NUMBER,
bno     BRANCH_NUMBER             NOT NULL,
PRIMARY KEY  (pno)
FOREIGN KEY (sno) REFERENCES staff оп delete SET NULL on update
CASCADE
FOREIGN KEY (ono) REFERENCES owner on delete NO ACTION on update
CASCADE
FOREIGN KEY (bno) REFERENCES branch оп delete NO ACTION on update
CASCADE);
```

Физическое проектирование
Механизм создания триггеров

CREATE TRIGGER *property before update*
BEFORE UPDATE ON property_for_rent
FOR EACH ROW
WHEN (:NEW.rent / :OLD.rent > 1.1)
BEGIN
INSERT INTO ***property_for_rent_audit*** *VALUES*
(:OLD.pno, :OLD.street, :OLD.area, :OLD.city, :OLD.pcode,
:OLD.ono, :OLD.sno, :OLD.bno, :OLD.type, :OLD.rooms,
:OLD.rent)
END;

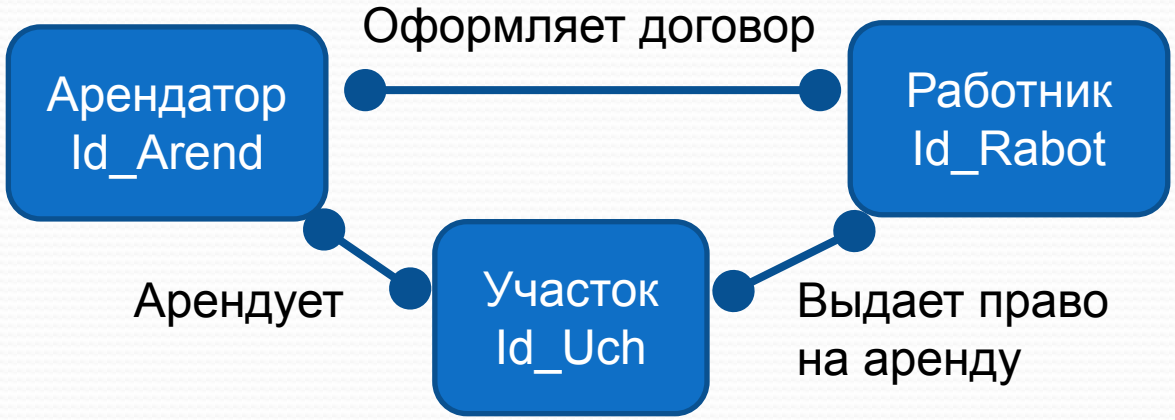
Физическое проектирование
Механизм создания индексов

Индекс представляет собой механизм доступа, усоряющий выборку данных из таблицы - он действует подобно помещенному в книгу указателю.
Уникальным называют такой ***индекс***, в котором двум различным кортежам таблицы запрещено иметь одинаковое индексируемое значение (является ключом отношения).

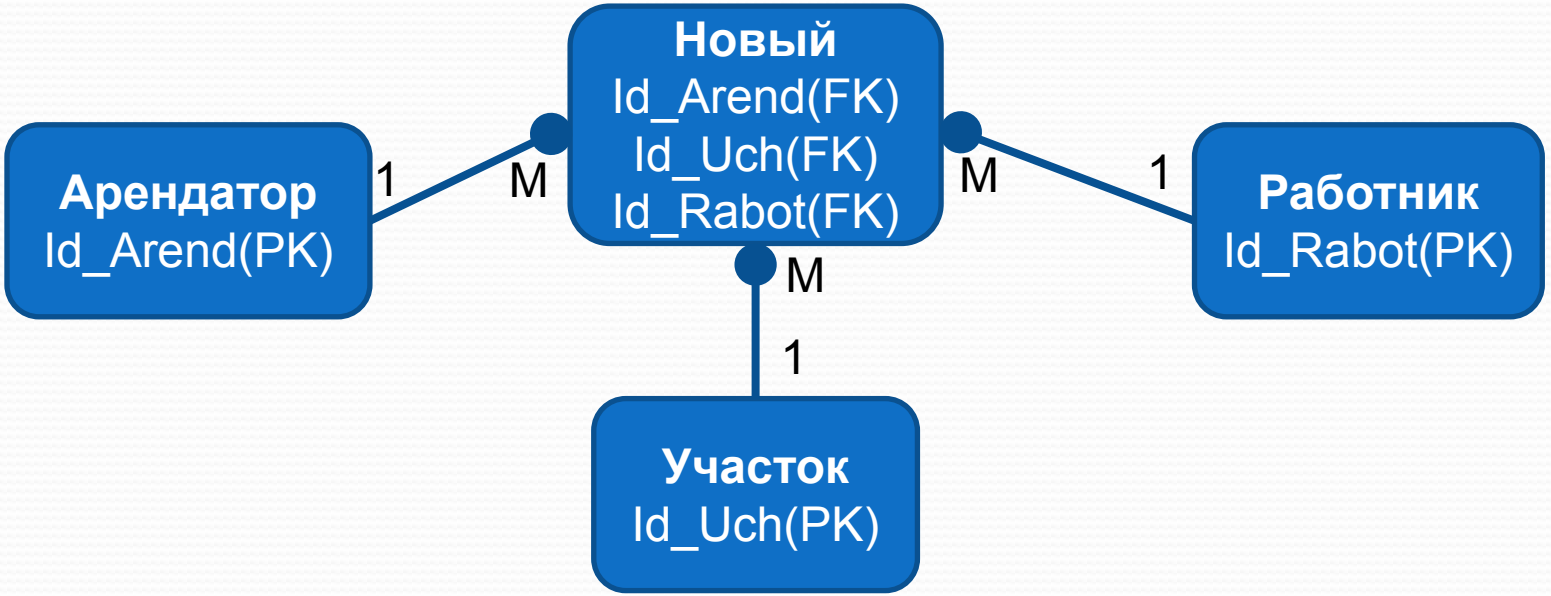
CREATE UNIQUE INDEX ***property_no_index***
ON ***property_for_rent*** (pno,rno);

Удаление сложных связей

Было (Концептуальная модель):



Стало (Логическая модель):



Удаление рекурсивных связей

Рекурсивными *называются такие связи, в которых сущность некоторого типа взаимодействует сама с собой.*

Если концептуальная модель содержит рекурсивные связи, они должны быть устранены посредством определения некоторой промежуточной сущности. Например, для отображения ситуации, когда один из работников руководит группой других работников, может быть установлена рекурсивная связь типа "один ко многим" (1:M) *«Руководит»* - как показано на рис. Рекурсивная природа этой связи требует особого подхода при работе с ней как на этапе логического проектирования, так и на этапе физической реализации базы данных.

Концептуальное проектирование. Этап 1.

Для упрощения данной рекурсивной связи типа 1:M мы заменим ее вновь созданной сущностью *Подчиненные* и дополнительной связью типа 1:1 с именем У*правляется* - как показано на рис.

Id_Sotrud	*Id_Rukov*
2	1
3	2
4	2

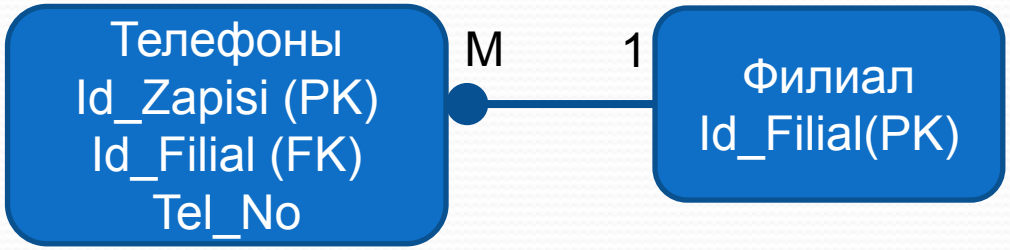
Id_Sotrud	*Name*
1	Иванов
2	Петров
3	Сидоров
4	Кузнецов

Удаление связей с атрибутами

Если в концептуальной модели присутствуют связи, имеющие собственные атрибуты, они должны быть преобразованы путем создания новой сущности. Например, рассмотрим ситуацию, когда требуется фиксировать количество рабочих часов, отработанных временным персоналом каждого из отделений предприятия. Связь **(M:N) Сотрудник (сущ.)** – ***Работает*** – **Филиал (сущ.)** имеет атрибут с именем *Отработано часов*. Преобразуем связь ***Работает*** в дополнительную сущность с именем ***Сотр_Фил***, которой назначим атрибут ***Kol_Chas (Отработано часов)***, после чего создадим две новых-связи типа 1:M, как показано на рис.

Удаление множественных атрибутов

Множественными называют атрибуты, которые могут иметь одновременно несколько значений для одного и того же экземпляра сущности. Если в концептуальной модели присутствует множественный атрибут, его следует преобразовать путем определения новой сущности. Например, для отображения ситуации, когда одно и то же отделение компании имеет несколько телефонных номеров, в концептуальной модели был определен множественный атрибут Tel_No, относящийся к сущности Branch. Этот множественный атрибут Telephone, имеющую единственный простой атрибут Tel_No, и создав новую связь типа 1:M.



Проверка связей типа 1:1

В процесс определения сущностей могли быть созданы две различные сущности, которые на самом деле представляют один и тот же объект в предметной области приложения. Например, могли быть созданы две сущности, Branch и Department, которые на самом деле представляют один и тот же тип объекта. Другими словами, имя Branch является синонимом имени Department. В подобном случае следует объединить эти две сущности в одну. Если первичные ключи объединяемых сущностей различны, выберите один из них в качестве первичного, а другой укажите как альтернативный ключ.

Удаление избыточных связей

Связь является ***избыточной****, если одна и та же информация может быть получена не только через нее, но и с помощью другой связи.*

Всегда следует стремиться создавать минимальные модели данных, и поэтому, если избыточная связь не является очевидно необходимой, ее следует удалять. Установить, что между двумя сущностями имеется больше одной связи, довольно просто. Однако из этого еще не следует, что одна из двух связей обязательно является избыточной, поскольку обе они могут

Представлять различные объединения, реально существующие в организации.

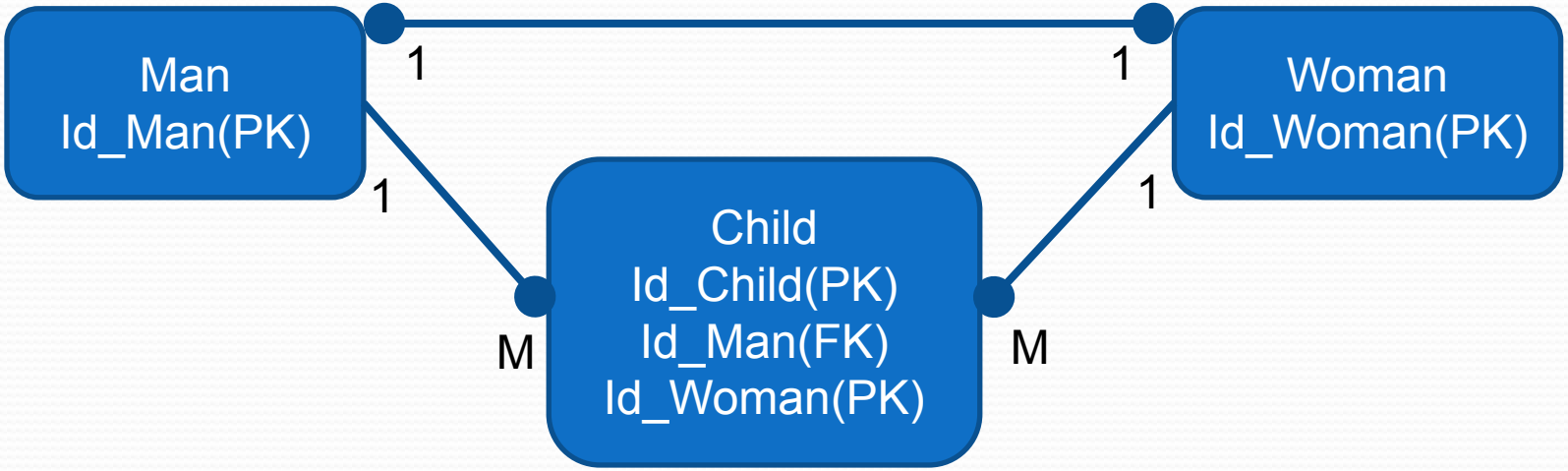
При устранении избыточности доступа большое значение имеют временные показатели.

Удаление избыточных связей

Например, рассмотрим ситуацию, когда необходимо смоделировать связи между сущностями Man (Мужчина), Woman (Женщина) и Child (Ребенок), как показано на рис. Очевидно, что между сущностями *Man* и *Child* имеется два пути доступа:

- один -- через непосредственную связь *FatherOf* (Является отцом);
- другой – через связи MarredTo (Женат на) и *MotherOf* (Является матерью).

На первый взгляд кажется, что связь *FatherOf* является избыточной. Однако, это утверждение может оказаться ошибочным. Например, отец может иметь детей от предыдущего брака, а мы моделируем только текущий брак отца (через связь **1:1).**



Этап 1.2. Определение типов связей.

Поэтому все существующие взаимоотношения примера не могут быть смоделированы без использования связи типа FatherOf.

Суть состоит в том, что при устранении избыточности очень важно исследовать значение каждой из связей, существующих между сущностями.

По завершении данного этапа мы получили упрощенную локальную концептуальную модель данных, из которой удалены все структуры, реализация которых в среде реляционных СУБД затруднительна.

Поэтому на данном этапе правильнее будет называть улучшенную *локальную концептуальную модель* - **локальной логической моделью** данных.

Этап «Проверка модели с помощью правил нормализации».

Цель *Проверка локальной логической модели данных с использованием технологии нормализации.*

Процедуры нормализации достаточно подробно были рассмотрены ранее. Нормализация используется для улучшения модели данных, для того чтобы она удовлетворяла различным ограничениям, позволяющим исключить нежелательное дублирование данных.

Нормализация гарантирует, что полученная в результате ее применения модель данных будет наилучшим образом отображать особенности использования информации на предприятии, не содержать противоречий, иметь минимальную избыточность и максимальную устойчивость к изменениям.

Этап «Проверка модели с помощью правил нормализации».

Нормализация представляет собой процедуру принятия решений о том, какие именно атрибуты должны быть объединены для представления сущностей каждого типа.

В одной из фундаментальных концепций теории нормализации утверждается, что атрибуты должны быть сгруппированы в отношения в соответствии с существующими между ними логическими связями.

В некоторых случаях имеют место утверждения, что нормализация разрабатываемых баз данных не позволяет достичь максимальной производительности при их обработке.

На это можно ответить следующими замечаниями.

Этап Нормализация

- Нормализация проекта позволяет организовать размещение данных в соответствии с их функциональными зависимостями. Поэтому данная процедура должна выполняться между этапами концептуального и физического проектирования.
- На этапе логического проектирования не ставится задача достичь окончательного вида проекта. Цель этого этапа заключается в предоставлении проектировщику более углубленного понимания природы и назначения данных, используемых на предприятии. Если к приложению предъявляются специфические требования в отношении его производительности, то они должны учитываться на этапе физического проектирования. В этом случае одним из подходов является *денормализация* определенных таблиц. Однако это не означает, что время на их нормализацию было затрачено впустую, поскольку для правильного выполнения нормализации проектировщик должен глубоко изучить семантику и особенности использования данных. *Подробнее о денормализации речь пойдет в последующих лекциях*.

Этап: Нормализация

-Нормализация проекта позволяет повысить его устойчивость и избавиться от аномалий обновления (изменения, добавления, корректировки записей).

-За последних несколько лет мощность компьютеров существенно возросла. Поэтому в некоторых случаях может быть вполне обоснованным решение реализовать проект, позволяющий упростить работу с данными за счет выполнения некоторого объема дополнительной обработки.

• Выполнение нормализации требует от разработчика полного понимания назначения каждого атрибута, присутствующего в создаваемой базе данных. Достигнутые за счет этого преимущества могут оказаться весьма существенными.

• В результате применения нормализации разработчик получает весьма гибкий проект базы данных, позволяющий легко вносить в нее необходимые расширения.

Этап Нормализация

Другими словами, наша задача состоит в проверке корректности состава каждого из созданных отношений посредством применения к ним процедуры нормализации. Процесс нормализации включает следующих три основных этапа:

- приведение к первой нормальной форме (lНФ), позволяющее удалить из отношений повторяющиеся группы атрибутов;
- приведение ко второй нормальной форме (2НФ), позволяющее устранить частичную зависимость атрибутов от первичного ключа;
- приведение к третьей нормальной форме (3НФ), позволяющее устранить транзитивную зависимость атрибутов.

Этап Проверка модели в отношении транзакций пользователей

Цель - *Убедиться в том, что локальная логическая модель данных позволяет выполнить все транзакции, предусмотренные данным представлением пользователя.*

Целью выполнения данного этапа является проверка локальной логической модели данных на возможность выполнения всех транзакций, предусмотренных данным представлением пользователя. *Перечень транзакций определяется в соответствии со спецификациями, описывающими действия, выполняемые данным пользователем.*

Этап Проверка модели в отношении транзакций пользователей

Используя ER-диаграммы, словарь данных и установленные связи между первичными и внешними ключами, указанные в описании отношений, мы попытаемся выполнить все необходимые операции доступа к данным вручную.

Если нам удастся подобным образом найти способ выполнения всех требуемых транзакций, то на этом проверка логической модели данных будет завершена. Однако, если какую-либо из транзакций выполнить вручную не удастся, значит, составленная модель данных является неадекватной и содержит ошибки, которые потребуется устранить.

Вероятнее всего, ошибка будет связана с пропуском в модели данных сущности, связи или атрибута.

Этап Проверка модели в отношении транзакций пользователей

Рассмотрим возможный подхода, благодаря которому мы сможем убедиться, что локальная логическая модель данных позволяет выполнить все необходимые транзакции.

Подход предусматривает выполнение проверки того, что данная логическая модель предоставляет всю информацию (сущности, связи и их атрибуты), необходимую для выполнения каждой из транзакций.

Например, транзакции, необходимые предприятия, могут включать следующие операции:

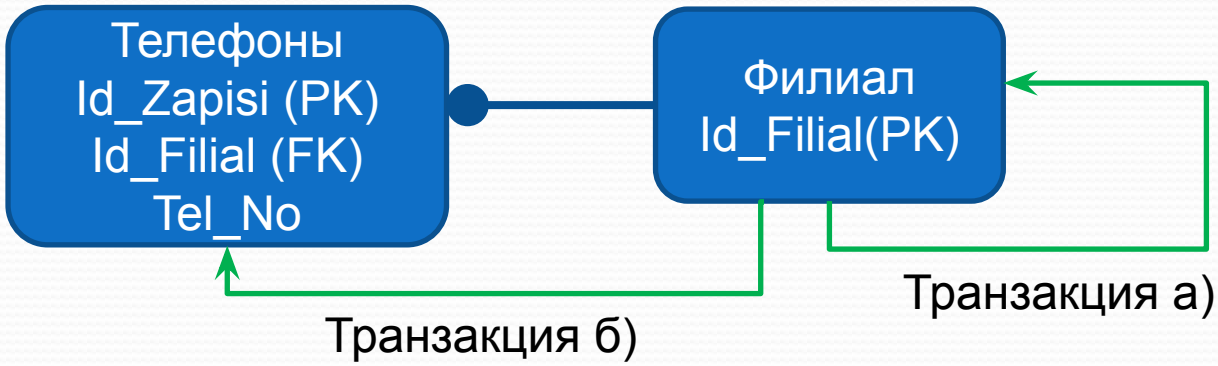
Этап Проверка модели в отношении транзакций пользователей

а) ***ввод сведений о новом филиале.*** Первичным ключом отношения *Филиал* является атрибут Id_Filial. Прежде всего следует убедиться, что присвоенный новому филиалу номер является уникальным. Если это не так, то ввод данных следует запретить, а выполнение операции завершить. В противном случае ввод данных о новом сотруднике разрешается.

б) *удаление сведений о филиале*, заданном его номером. Выполняется поиск заданного личного номера в соответствующем столбце отношения *Филиал.* Если он не будет найден, то фиксируется ошибка и дальнейшие действия не выполняются. В противном случае из отношения *Филиал* удаляется весь найденный кортеж и обновляется внешний КЛЮЧ всех кортежей отношения *Телефоны*, с которыми был связан "удаляемый» филиал.

Этап Проверка модели в отношении транзакций пользователей

Подход к проверке модели данных на соответствие требуемым транзакциям заключается в нанесении непосредственно на ER-диаграммы всех путей, которые потребуются для выполнения каждой из транзакций. Простой пример использования этого подхода для контроля за выполнением упомянутых выше транзакций (а и б) показан на рис.



Этап Проверка модели в отношении транзакций пользователей.

Этот подход позволяет визуально выделить те области модели, которые не используются для выполнения транзакций, а также те области, которые наиболее существенны с точки зрения выполнения транзакций.

Если на диаграмме имеются области, которые *не используются* ни в одной из транзакций, возникает вопрос о целесообразности представления этой информации в модели данных.

В то же время, если в модели присутствуют области, которые не позволяют найти подходящий метод выполнения не которой транзакции, потребуется провести анализ того, какая из обязательных для выполнения транзакции сущностей или связей была пропущена при составлении модели.

Этап 2.5. Создание диаграмм "сущность-связь"

Цель Создание окончательного варианта диаграмм "сущность-связь" (ER-диаграмм), являющихся локальным логическим представлением данных, используемых отдельными пользователями приложения.

Теперь все готово для того, чтобы создать окончательные варианты ER·диаграмм для отдельных представлений каждого из пользователей предприятия. *Данные на этих диаграммах были проверены с применением методов нормализации, а также проконтролированы на предмет возможности выполнения всех требуемых транзакций*.

Этап Определение требований поддержки целостности данных

Цель *Определение ограничений, налагаемых в представлениях пользователей требованием сохранения целостности данных*.

Ограничения целостности данных представляют собой такие ограничения, которые вводятся с целью предотвратить помещение в базу противоречивых данных. Отметим, что, хотя в конкретных СУБД *ФУНКЦИИ КОНТРОЛЯ целостности могут как поддерживаться, так и не поддерживаться*, в данном случае это не будет нас интересовать.

На этом этапе мы занимаемся проектированием только на верхнем уровне, где рассмотрение вопросов целостности данных является обязательным условием, не связанным с конкретными аспектами реализации.

Этап Определение требований поддержки целостности данных

Полное и точное отражение представления пользователя мы сможем получить ТОЛЬКО после определения ограничений, необходимых с ТОЧКИ зрения сохранения целостности данных.

Здесь мы обсудим пять типов ограничений целостности данных:

- *обязательные данные;*
- *ограничения для доменов атрибутов;*
- *целостность сущностей;*
- *ссылочная целостность;*
- *требования данного предприятия*.

Этап Определение требований поддержки целостности данных

Обязательные ***данные:***

Некоторые атрибуты всегда должны содержать одно из допустимых значений. Другими словами, эти атрибуты не могут иметь пустого (NULL) значения. Так, каждый работник должен: занимать ту или иную должность (например, «руководитель», «начальник отдела» или «бухгалтер» и т.д.).

Эти ограничения должны фиксироваться при занесении сведений об атрибуте в словарь данных.

Этап Определение требований поддержки целостности данных

Ограничения для доменов атрибутов.

Каждый атрибут имеет домен, представляющий собой набор его допустимых значений.

Например, атрибут "пол" может содержать одно из двух допустимых значений - "М" или "Ж", поэтому его домен состоит из двух символьных строк длиной в один символ, содержащих указанные значения.

Данные ограничения устанавливаются при определении доменов атрибутов, присутствующих в модели данных.

Этап Определение требований поддержки целостности данных

Целостность сущностей:

Первичный ключ любой сущности не может содержать пустого значения.

Например, каждая строка отношения Staff должна содержать уникальное значение атрибута первичного ключа; в данном случае это - атрибут Staf_No.

Подобные ограничения должны учитываться при определении первичных ключей для сущностей каждого типа.

Этап Определение требований поддержки целостности данных

Ссылочная целостность

Внешний ключ связывает каждую строку дочернего отношения с той строкой родительского отношения, которая содержит это же значение соответствующего потенциального ключа. Понятие ссылочной целостности означает, что если внешний ключ содержит некоторое значение, то оно обязательно должно присутствовать в потенциальном ключе одной из строк родительского отношения.

Этап Определение требований поддержки целостности данных

Например, атрибут Branch_No отношения Staff связывает данные о каждом из работников со строкой в отношении Branch, соответствующей тому отделению предприятия, в котором он работает.

Если поле Branch_No не пусто, оно должно содержать допустимое значение, присутствующее в атрибуте Branch_No одной из строк отношения Branch. В противном случае окажется, что работник трудится в несуществующем отделении.

Этап Определение требований поддержки целостности данных

Существует несколько важных моментов, связанных с использованием внешних ключей. Во-первых, следует проанализировать, допустимо ли использование во внешних ключах пустых значений. Например, имеет ли смысл сохранять сведения о работнике, если не задан номер отделения предприятия, в котором он работает?

Вопрос не в проверке существования заданного номера отделения, а в том, обязательно ли этот номер должен указываться. В общем случае, если участие дочернего отношения в связи является тотальным, то рекомендуется *запрещать использование пустых значений в соответствующем внешнем ключе*. В то же время, если участие дочернего отношения в связи является частичным, *то помещение пустых значений в атрибут внешнего ключа должно быть разрешено.*

Этап Определение требований поддержки целостности данных

Следующая проблема связана с организацией поддержки ссылочной целостности.

Реализация этой поддержки осуществляется посредством задания ограничений существования, определяющих условия, при которых может вставляться, обновляться или удаляться каждое значение потенциального или внешнего ключа. Рассмотрим связь Staff manages Property типа 1:M. Первичный ключ отношения Staff (атрибут Staff_No) является внешним ключом отношения Property.

Этап Определение требований поддержки целостности данных

Рассмотрим следующие ситуации:

Случай 1. Вставка новой строки в дочернее отношение *(Property).* Для обеспечения ссылочной целостности необходимо убедиться, что значение атрибута внешнего ключа Staff_No новой строки отношения Property равно пустому значению либо некоторому конкретному значению, присутствующему в одной из строк отношения Staff.

Случай 2. Удаление строки из дочернего отношения *(Property).* При удалении строки из дочернего отношения никаких нарушений ссылочной целостности не происходит.

Этап Определение требований поддержки целостности данных

Случай 3. Обновление внешнего ключа в строке дочернего отношения *(Property).* Этот случай подобен случаю 1. Для сохранения ссылочной целостности необходимо убедиться, что атрибут ***Staff_No*** в обновленной строке отношения Property содержит либо пустое значение, либо некоторое конкретное значение, присутствующее в одной из строк отношения Staff.

Случай 4. Вставка строки в родительское отношение *(Staff).* Вставка строки в родительское отношение (Staff) не может вызвать нарушения ссылочной целостности. Добавленная строка просто становится родительским объектом, не имеющим дочерних объектов. В данном случае это означает, что новый работник еще не отвечает ни за какие объекты недвижимости.

Этап Определение требований поддержки целостности данных

Случай 5. Удаление строки из родительского отношения *(Staff).* При удалении строки из родительского отношения ссылочная целостность будет нарушена в том случае, если в дочернем отношении будут существовать строки, ссылающиеся на удаленную строку родительского отношения. Другими словами, ссылочная целостность будет нарушена, если удаленный работник отвечал за один или больше объектов недвижимости.

В этом случае может быть использована однэ. из следующих стратегий.

Этап Определение требований поддержки целостности данных

NO ACTION. Удаление строки из родительского отношения запрещается, если в дочернем отношении существует хотя бы одна ссылающаяся на нее строка. В нашем случае это звучит так: "Нельзя удалить сведения о работнике, отвечающем в настоящий момент хотя бы за один объект недвижимости" .

CASCADE. При удалении строки из родительского отношения автоматически удаляются все ссылающиеся на нее строки дочернего отношения. Если любая из удаляемых строк дочернего отношения выступает в качестве родительской стороны в некоторой другой связи, то операция удаления применяется ко всем строкам дочернего отношения этой СВЯЗИ. *Другими словами, удаление строки родительского отношения автоматически распространяется на любые дочерние отношения.* В нашем случае это звучит так: "Удаление работника автоматически влечет за собой удаление сведений обо всех объектах недвижимости, которыми он занимался". *Очевидно, что в данном примере подобная стратегия неприемлема.*

Этап Определение требований поддержки целостности данных

SET NULL. При удалении строки из родительского отношения во всех ссылающихся на нее строках дочернего отношения в атрибут внешнего ключа записывается пустое значение. Следовательно, удаление строк из родительского отношения вызовет занесение пустого значения. в соответствующий атрибут строк дочернего отношения. В нашем случае это звучит так: "При удалении работника все объекты недвижимости, которыми он занимался, остаются без отвечающего за них работника".

Эта стратегия может использоваться только в тех случаях, когда в атрибут внешнего ключа дочернего отношения разрешается помещать пустые значения.

Этап Определение требований поддержки целостности данных

SET DEFAULT. При удалении строки из родительского отношения в атрибут внешнего ключа всех ссылающихся на нее строк дочернего отношения автоматически помещается значение, указанное для этого атрибута как значение по умолчанию. Таким образом, удаление строки из родительского отношения вызывает помещение принимаемого по умолчанию значения в атрибут внешнего ключа всех строк дочернего отношения, ссылающихся на удаленную строку.

Этап Определение требований поддержки целостности данных

В нашем случае это будет звучать так: "При удалении работника все объекты недвижимости, которыми он занимался, передаются некоторому другому работнику (например, руководителю отделения)". Эта стратегия применима только в тех случаях, когда атрибуту внешнего ключа дочернего отношения назначено некоторое значение, принимаемое по умолчанию.

• ***NO CHECK***. При удалении строки из родительского отношения никаких действий по сохранению ссылочной целостности данных не предпринимается.

Этап Определение требований поддержки целостности данных

Случай 6. Обновление первичного ключа в строке родительского отношения(Statt).

Если значение первичного ключа некоторой строки родительского отношения будет обновлено, нарушение ссылочной целостности будет иметь место в том случае, если в дочернем отношении существуют строки, ссылающиеся на исходное значение первичного ключа. В нашем случае это значит, что работник, для которого было выполнено обновление, *в данный момент отвечал за один или более объектов недвижимости.* Для сохранения ссылочной целостности может использоваться любая из описанных выше стратегий. *При использовании стратегии CASCADE обновление значения первичного ключа в строке родительского отношения будет отображено в любой строке дочернего отношения, ссылающейся на данную строку (каскадным образом).*

Этап Учет требования данного предприятия

В заключение требуется проанализировать ограничения, называемые ***ограничениями предприятия*** *(или бизнес-правилами).*

Например, обновление сущностей может регламентироваться принятыми на предприятии правилами, описывающими методы выполнения транзакций, связанных с подобными обновлениями. В нашем примере, в компании *DreamHome* может быть принято правило, *запрещающее одному работнику одновременно заниматься более чем десятью объектами недвижимости*.

Документирование всех ограничений целостности данных.

Поместите сведения обо всех установленных ограничениях целостности данных в словарь данных. Они потребуются на этапе физической реализации базы БД.

Взаимосвязь между логическими моделями данных и диаграммами потоков данных

Логическая модель данных отображает структуру сохраняемых данных предприятия. Диаграмма потоков данных *(Data Flow Diagram - DFD)* отображает перемещение данных в пределах предприятия и помещение их в хранилища информации. *Все атрибуты, которые сохраняются на предприятии, должны быть объявлены в одном из типов сущностей и, вероятно, найти свое место в потоках данных, перемещающихся в пределах предприятия.* Если обе эти технологии используются для моделирования требований пользователей, каждая из них может применяться для контроля согласованности и полноты другой.

Правила, которые определяют отношения между этими двумя технологиями, следующие:

- каждое хранилище данных должно представлять все множество типов сущностей;
- атрибуты в потоках данных должны принадлежать сущности того или иного типа.

Этап. Создание и проверка глобальной логической модели данных

Цель *Объединение отдельных локальных логических моделей данных в единую глобальную логическую модель данных, представляющую ту часть предприятия, которая охватывается данным приложением.*

На данном этапе фазы логического проектирования баз данных строится глобальная логическая модель данных, создаваемая посредством слияния отдельных логических моделей данных, отражающих представления каждого пользователя. *По завершении объединения локальных моделей необходимо проверить правильность полученной глобальной модели, как в отношении правил нормализации, так и в отношении возможности выполнения транзакций, предусмотренных спецификациями на функции отдельных пользователей.*

Этап. Создание и проверка глобальной логической модели данных

Проверка выполняется с использованием тех же методов, которые применялись при выполнении предыдущих этапов логического проектирования локальных моделей. Однако проведение нормализации потребуется только в том случае, если в процессе слияния были внесены изменения в состав отдельных типов сущностей.

Аналогично, проверка возможности выполнения транзакций выполняется только для тех областей модели, которые были подвергнуты изменениям в ходе слияния. *В больших системах подобный подход позволяет существенно сократить объем требуемых повторных проверок.*

Этап. Создание и проверка глобальной логической модели данных

Хотя каждая отдельная логическая модель данных предполагается корректной, полной и непротиворечивой, *любая из них отражает лишь восприятие системы отдельным пользователем или группой пользователей.* Другими словами, каждая модель отражает не функции предприятия, а лишь представление о функциях предприятия отдельных его работников. *Поэтому любая из этих моделей является неполной.*

А это означает, что между отдельными моделями в полном наборе представлений могут существовать несовместимость и взаимное перекрытие. Следовательно, при слиянии локальных моделей данных в единую глобальную модель придется прилагать усилия *для устранения конфликтов между отдельными представлениями и принимать во внимание их возможное перекрытие.*

Этап. Создание и проверка глобальной логической модели данных

Процедуру слияния можно считать самой важной в ходе логического проектирования базы данных, поскольку с ее помощью создается представление о предприятии, не зависящее от любых конкретных пользователей, бизнес-процессов или приложений.

Данный этап предусматривает выполнение следующих действий.

- Этап 3.1. Слияние локальных логических моделей данных в единую глобальную модель данных.
- Этап 3.2. Проверка глобальной логической модели данных.
- Этап 3.3. Проверка возможностей расширения модели в будущем.
- Этап 3.4. Создание окончательного варианта диаграммы "сущность-связь".
- Этап 3.5. Обсуждение глобальной логической модели данных с пользователями.

Этап 3.1. Слияние локальных логических моделей данных

Цель *Объединить отдельные локальные логические модели данных в единую глобальную логическую модель данных предприятия.*

В небольших системах, насчитывающих два-три пользовательских представления с незначительным количеством типов сущностей и связей, задача сравнения локальных моделей с последующим слиянием и устранением любых возможных противоречий является относительно несложной. *Однако, в крупных системах потребуется использовать более систематический подход*.

Ниже описывается один из подобных подходов, который можно использовать для выполнения слияния локальных моделей и разрешения любых возникающих при этом проблем.

Этап 3.1. Слияние локальных логических моделей данных

Предлагаемый подход предусматривает выполнение следующих действий:

1. Анализ имен сущностей и их первичных ключей.
2. Анализ имен связей.
3. Слияние общих сущностей из отдельных локальных моделей.
4. Включение (без слияния) сущностей, уникальных для каждого локального представления.
5. Слияние общих связей из отдельных локальных моделей.
6. Включение (без слияния) связей, уникальных для каждого локального представления.
7. Проверка на наличие пропущенных сущностей и связей.
8. Проверка корректности внешних ключей.
9. Проверка соблюдения ограничений целостности.
10. Выполнение чертежа глобальной логической модели данных.
11. Обновление документации

Этап 3.1. Слияние локальных логических моделей данных

Вероятно, самый простой метод слияния нескольких локальных моделей данных в единую модель состоит *в слиянии двух локальных моделей в одну общую модель*, с последующим добавлением к ней третьей локальной модели (и т.д.). Процесс добавления к предыдущему результату очередной локальной модели будет продолжаться то тех пор, пока все модели не будут слиты в единую глобальную модель. *Этот подход можно считать более простым, чем попытка слить все локальные модели за одну операцию.*

Очень важно отметить, что, прежде чем приступать к созданию глобального представления информационной модели предприятия, следует убедиться, что каждая из сливаемых локальных логических моделей была создана в полном соответствии с последовательностью действий, предусмотренных первым и вторым этапами разработки, определенными в обсуждаемой методологии проектирования баз данных.

Этап 3.1. Слияние локальных логических моделей данных

1. Анализ имен сущностей и их первичных ключей.

Может оказаться полезным предварительно проанализировать имена сущностей, присутствующих в локальных моделях данных, - эти сведения можно найти в словаре данных. Проблемы имеют место в следующих случаях:

- *если две или более сущностей имеют одно и то же имя, но на самом деле отличаются одна от другой;*
- *если две или более сущностей идентичны, но имеют различные имена.*

Для выявления возможных проблем следует сравнить между собой составы данных сущностей каждого типа. *В частности, обнаружить эквивалентные сущности с различными именами поможет сравнение их первичных ключей.*

2. Анализ имен связей.

Выполняемые действия аналогичны описанным на предыдущем этапе.

Этап 3.1. Слияние локальных логических моделей данных

Слияние общих сущностей из отдельных локальных моделей:

Следует проанализировать имена и содержимое сущностей каждого типа, присутствующих в сливаемых логических моделях. Обычно эта процедура включает следующие действия:

- слияние сущностей с одинаковыми именами и первичными ключами;
- слияние сущностей с одинаковыми именами, но с различными первичными

ключами;

- слияние сущностей с различными именами, имеющих одинаковые или различные первичные ключи.

Этап 3.1. Слияние локальных логических моделей данных

Слияние сущностей с одинаковыми именами и первичными ключами.

Как правило, сущности с одним и тем же первичным ключом представляют один и тот же объект реального мира и, следовательно, должны быть слиты. Объединенная сущность будет включать все атрибуты сливаемых сущностей, за исключением дублирующихся. Далее в примере перечислены атрибуты, связанные с двумя сущностями, носящими имя ***Staff***, которые определены в двух различных представлениях, названных ***Viewl*** и ***View2***.

В обоих случаях первичным ключом является атрибут ***Staff_No.*** Слияние этих сущностей осуществляется путем объединения их атрибутов, поэтому в *Сводной сущности* ***Staff*** *будут присутствовать все атрибуты исходных сущностей.*

Слияние сущностей с одинаковыми именами и первичными ключами.

- ***Представление View1***

Staff (Staff No, Name, Position, Sex, Salary, Branch_No)

Primary Key Staff No

Foreign Key Branch_No references Branch(Branch_No)

- ***Представление View2***

Staff (Staff No, FName, LName, Address, Branch_No)

Primary Key Staff No

Foreign Key Branch_No references Branch(Branch_No)

- ***Итоговая сущность Staff***

Staff (Staff No, FName, LName, Address, Position, Sex, Salary, Branch_No)

Primary Key Staff No

Foreign Key Branch_No references Branch(Branch_No)

Слияние сущностей с одинаковыми именами, но с различными первичными ключами.

В некоторых ситуациях могут быть обнаружены две сущности с одним и тем же именем, в которых используются различные первичные ключи, однако имеются одинаковые потенциальные ключи. В этом случае сущности сливаются аналогично тому, как это было сделано в предыдущем варианте. *Дополнительно потребуется выбрать в результирующей сущности первичный ключ, объявив все остальные ключи альтернативными.*

- ***Представление View1***

Staff (Staff_No, Name, Position, Sex, Salary, Branch_No)

Primary Key Name

Alternate Key Staff No

Foreign Key Branch_No references Branch(Branch_No)

- ***Представление View2***

Staff (Staff No, FName, LNarne, Address, Branch_No)

Primary Key Staff No

Alternate Key FName, LNarne

Foreign Key Branch_No references Branch(Branch_No)

- ***Глобальное представление***

Staff (Staff No, FNarne, LNarne, Address, Position, Sex, Salary, Branch_No)

Primary Key Staff No

Alternate Key FName, LNarne

Foreign Key Branch_No references Branch(Branch_No)

Слияние сущностей с различными именами, имеющих одинаковые или различные первичные ключи.

В некоторых случаях можно обнаружить сущности, которые имеют различные имена, но предназначены для одной и той же цели.

Подобные эквивалентные сущности можно распознать по их именам, *которые будут указывать на их сходное назначение*, по их содержанию и по их первичным ключам.

Кроме того, их можно распознать по участию в определенных связях.

Типичным примером подобной ситуации является наличие в моделях сущностей с названиями Staff (Персонал) и Employee (Работник), *которые, по сути, являются эквивалентными и должны быть слиты в единую сущность*.

Слияние локальных логических моделей данных

4 Включение (без слияния) сущностей, уникальных для каждого локального представления

На предыдущем этапе были выделены все сущности, описывающие подобные объекты. *Все остальные сущности просто включаются в глобальную модель без внесения каких-либо изменений.*

5. Слияние общих связей из отдельных локальных моделей

На этом этапе анализируются имена и назначение каждой из связей во всех представлениях отдельных пользователей. Прежде чем объединять связи, очень важно разрешить любые конфликты, которые могут иметь место между ними, - например, в отношении ограничений участия или кардинальности. *Выполняемые на этом этапе действия включают слияние связей с одинаковыми именами и назначением, после чего может потребоваться выполнить слияние связей с различными именами, но имеющих одно и то же назначение.*

Включение (без слияния) связей, уникальных для каждого локального представления

6. Включение (без слияния) связей, уникальных для каждого локального представления

На предыдущем этапе были выявлены и слиты все связи, имевшие сходное назначение (по определению, эти связи должны существовать между одними и теми же сущностями, которые также должны быть слиты друг с другом). Все оставшиеся связи включаются в глобальную модель без каких-либо изменений.

Проверка на наличие пропущенных сущностей и связей

7. Проверка на наличие пропущенных сущностей и связей

Вероятно, одной из самых трудных задач при создании глобальной модели данных является задача выявления пропущенных сущностей и связей между элементами представлений различных пользователей. Если на предприятии существует корпоративная модель данных, она может использоваться для обнаружения сущностей и связей между элементами представлений различных пользователей, которых нет ни в одном из локальных представлений.

Проверка на наличие пропущенных сущностей и связей

В то же время, при проведении опросов пользователей конкретного представления в качестве превентивной меры *следует попросить их уделить некоторое внимание сущностям и связям, которые, по их мнению, могут существовать в других представлениях.* Кроме того, при анализе атрибутов сущностей каждого типа *можно попробовать* выделить ссылки на сущности, принадлежащие другим пользовательским представлениям. *Достаточно часто оказывается, что атрибут, связанный с той или иной сущностью в представлении одного пользователя, соответствует первичному ключу, альтернативному ключу или даже простому, неключевому атрибуту некоторой сущности из другого представления.*

Слияние локальных логических моделей данных

8. Проверка корректности внешних ключей

На этом этапе может осуществляться слияние различных сущностей и связей, изменение первичных ключей и установка новых связей. Убедитесь, что внешние ключи в дочерних сущностях по-прежнему являются корректными, и в случае необходимости внесите в модель все требуемые изменения.

9. Проверка соблюдения ограничений целостности

Убедитесь, что установленные для глобальной логической модели ограничения целостности данных не вступают в противоречие с теми ограничениями, которые были установлены для каждого из пользовательских представлений. *Любые конфликты следует устранять посредством проведения консультаций с пользователями*.

Этап 3. Проверка возможностей расширения модели в будущем

Цель *Определение вероятности внесения каких-либо существенных изменений в созданную модель данных в обозримом будущем и оценка того, насколько данная модель приспособлена для этого.*

Очень важно, чтобы созданная глобальная модель *была легко расширяема.* Если модель сможет поддерживать только текущие требования, то время ее существования будет весьма ограниченным, а для реализации поддержки новых или изменяющихся требований потребуется прилагать значительные усилия. Необходимо так построить модель, чтобы она была легко расширяема и позволяла реализовать поддержку новых требований с минимальным изменениями в работе уже существующих пользователей. Следовательно, имеет смысл выполнить *проверку созданной модели с точки зрения эффективности реализации новых требований, которые могут иметь место в будущем.* Однако не следует вносить в модель каких-либо изменений, пока они не будут одобрены пользователями.