КР 2 ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БД

Курсовая работа выполняется для базы данных, самостоятельно созданной студентом по описанию предметной области.

**Цель:** получить навыки проектирования реляционных баз данных (БД)

**Ожидаемый результат**: построение схемы реляционной базы данных для выбранной предметной области

**Порядок выполнения работы**:

1. преобразовать ER-модель, созданную в КР 1 в схему реляционной БД
2. определить требования к объектам реляционной модели
3. оформить отчет о работе

Методические указания к выполнению работы

## 3.1 Построение реляционной модели БД

Цель данного этапа – преобразование концептуальной модели в структуры хранения информации. Результатом этого должна стать логическая, в нашем случае *реляционная модель*, не зависящая от особенностей используемой СУБД. Последовательность действий следующая.

1. **Каждая сущность ER-модели преобразуется в таблицу** (*отношение*). Каждая таблица должна иметь *первичный ключ* – уникальное поле, значение которого не может быть NULL. Это свойство первичного ключа обеспечивает *целостность сущностей*. В качестве первичного ключа выбираем обязательный уникальный атрибут сущности или создаем суррогатный автоинкрементный первичный ключ (identity). Первичный ключ может быть составным. Для таблиц, содержащих персональную информацию, рекомндуется в качестве первичного ключа использовать поле типа unique identifier/
2. **Устраняем составные и многозначные атрибуты**. Подчиненные атрибуты можно разместить отдельно в соответствующих полях, либо объединить в одно поле, если бизнес-логика не требует отдельного использования этих атрибутов. На рисунке 1а все поля адреса объединены в одну строку. Возможны два варианта преобразования многозначных атрибутов: а) вынести многозначный атрибут в отдельную таблицу; б) создать несколько полей для нескольких значений атрибута. На рис.1а показан второй вариант. Обратите внимание, что только поле Тел.1 является обязательным.

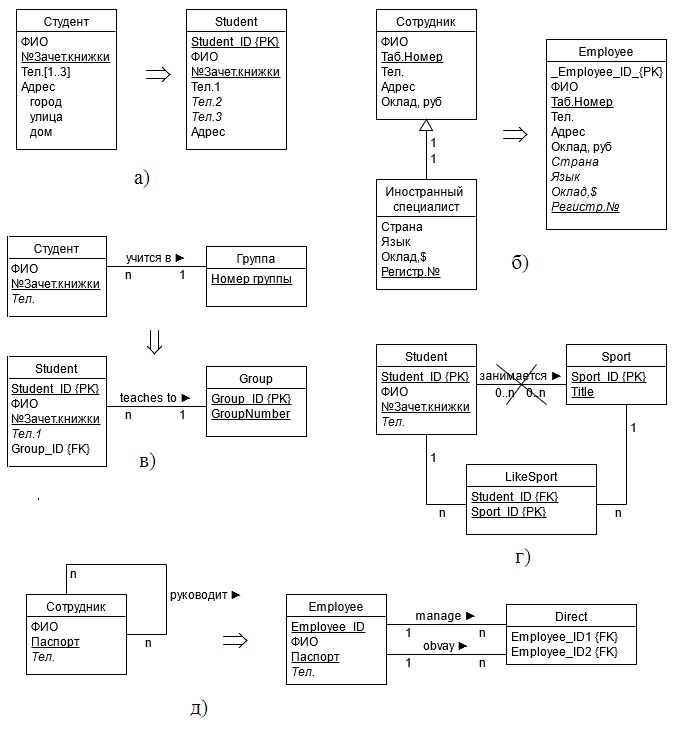


Рисунок 1 – Преобразование концептуальной модели в реляционную

(а – устранение многозначных и составных атрибутов; б – преобразование   
связи 1:1; в – реализация связи 1:n; г – преобразование связи n:n;  
д – преобразование рекурсивной связи n:n)

1. **Создаем связи между таблицами**. Связь реализуется парой ключей: первичный ключ (PK–primary key) в родительской таблице и внешний ключ (FK – foreign key) в дочерней (подчиненной) таблице. Значение внешнего ключа должно указывать на одно из значений первичного ключа или быть NULL. Это свойство внешнего ключа обеспечивает *ссылочную целостность* данных.
2. «*Один-к-одному*» – таблицы можно объединить в одну. Имеет смысл выносить в отдельную таблицу данные, которые необязательны и редко используются (рис.1б). Таблицы связываются по первичным ключам, либо нужно позаботиться о том, чтобы значение внешнего ключа было уникальным.
3. «*Один-ко-многим*» – в дочерней таблице создается поле внешнего ключа {FK} которое ссылается на поле первичного ключа {PK} родительской таблицы. Если на стороне родительской таблицы стоит кратность связи «0..1», то внешнему ключу следует разрешить NULL (рис.1в).
4. «*Многие-ко-многим*» – связь разрывается (рис.1г), создается дополнительная таблица, в которую в качестве внешних ключей помещаются ссылки на первичные ключи связываемых таблиц.
5. . Аналогично поступаем с рекурсивной связью «многие-ко-многим» (рис. 4д).
6. **Реализация пользовательских ограничений**. Современные СУБД предоставляют пользователю средства реализации прочих ограничений, накладываемых предметной областью. Важно, что эти средства работают на стороне сервера. Это – триггеры, и constraints (ограничения). Триггеры и constraints проверяют непротиворечивость данных при вставке, удалении или изменении их в таблице. Перечень необходимых объектов оформляется в виде таблиц, как показано в приложении 1.

## 3. 2 Проверка нормализации БД

Создание любой базы данных преследует две основные цели: понижение избыточности хранимых данных и повышение их надежности [2].

Удовлетворение этих требований достигается нормализацией отношений БД. Процесс нормализации заключается в разложении (декомпозиции) исходных отношений БД на более мелкие и простые. При этом на каждой ступени этого процесса схемы отношений приводятся к нормальным формам (НФ).

В теории реляционных БД разработано несколько нормальных форм, которые подчиняются правилу вложенности.

Отношение находится в первой нормальной форме (1НФ), если значения всех его атрибутов атомарны. Все повторяющиеся группы должны быть удалены в новое отношение.

Отношение находится во второй нормальной форме (2НФ), если оно находится в 1НФ, а каждый его не ключевой атрибут функционально полностью зависит от ключа, т.е. 2НФ требует, чтобы не было не ключевых атрибутов, которые зависят от части первичного ключа.

Отношение находится в третьей нормальной форме (3НФ), если оно находится во 2НФ и все его не ключевые атрибуты зависят только от первичного ключа. 3НФ – это отсутствие транзитивной зависимости, т.е. ни один не ключевой столбец не должен зависеть от другого не ключевого столбца таблицы.

Отношение находится в НФБК, если отсутствуют функциональные зависимости части первичного ключа от не ключевых атрибутов.

При реализации реляционной БД обязательным условием является удовлетворение требований 1НФ. Нормализация до 3НФ сохраняет все функциональные зависимости, т.е. семантику предметной области. НФБК разрушает некоторые функциональные зависимости. Рекомендуется проводить нормализацию до 3НФ. Проверка условий нормализации выполняется отдельно для каждой таблицы

Контрольные вопросы

1. Каковы задачи, решаемые на этапе логического проектирования?
2. Каковы базовые свойства реляционной модели данных?
3. В чем состоят требования целостной части реляционной модели данных?
4. Каковы общие свойства нормальных форм?
5. Каковы условия нахождения отношений в первой нормальной форме?
6. Каковы условия нахождения отношений во второй нормальной форме?
7. Каковы условия нахождения отношений в третьей нормальной форме?
8. Каковы условия нахождения отношений в третьей усиленной нормальной форме?
9. В чем состоят общие требования обеспечения ограничений целостности?
10. Каковы средства задания ограничений целостности в реляционной модели?

ПРИЛОЖЕНИЕ 1

ПРИМЕР ОТЧЕТА

3 ПОСТРОЕНИЕ ЛОГИЧЕСКОЙ МОДЕЛИ

3.1 Перевод ER-модели в реляционную форму

В ER-модели между сущностями нет связей «многие-ко-многим», нет многозначных и составных атрибутов, поэтому дополнительные таблицы не требуются. В реляционной модели каждой сущности соответствует таблица. Было решено для сущностей «Ученик» и «Учитель» использовать отдельные таблицы, при этом логин должен быть уникальным в пределах обеих таблиц.

Обеспечение целостности модели достигается включением триггеров, проверяющих логин пользователя. Эти триггеры, типа AFTER INSERT, AFTER UPDATE, необходимо добавить к таблицам «Ученик» и «Учитель» (проверяем уникальность логина). Кроме того (см. п. 3.1.1) необходим триггеры в таблице Marks («Оценки») для поддержания в непротиворечивом состоянии внешних ключей.

В разработанной реляционной форме глобальной модели используется 8 таблиц. Рассмотрим каждую из них более подробно.

3.1.1 Таблица Puples («Ученики»)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Название | Тип данных | Значение по умолчанию | Уникальность | Ограничения | Обязательность | Первичный ключ | Внешний ключ |
| PipleID | uniqueidentifier | NewGuid | + |  | + | + | - |
| Surname | varchar(20) | - | - |  | + | - | - |
| Name | varchar(20) | - | - |  | + | - | - |
| RoleID | int | - | - |  | + | - | + |
| Login | varchar(20) | - | + | >8 симв | + | - | - |
| Password | varchar(20) | - | >8 буквы, цифры | + | - | - |
| ClassID | int | - | - |  | + | - | + |

Для идентификации записей введен суррогатный первичный ключ со спецификацией uniqueidentifier. Тем не менее, уникальность поля Login должна обеспечиваться согласно модели. Внешний ключ ClassID используtтся для связи с таблицей Class.

3.1.2 Таблица Roles («Роль»)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Название | Тип данных | Значение по умолчанию | Уникальность | Ограничения | Обязательность | Первичный ключ | Внешний ключ |
| RoleID | int | identity | + |  | + | + | - |
| Name | varchar(20) | - | - |  | + | - | - |
| LoginServ | varchar(20) | - | - | + | + | - | - |
| PasswordServ | varchar(20) | - | - |  | + | - | - |

Данная таблица соответствует ранее описанному объекту «Роль».

Для идентификации записей введен суррогатный первичный ключ со спецификацией identity (автоинкремент). Кандидаты на первичный ключ уникальные поля «Название» и «ЛогинСерв» не используются из-за их символьного формата. За обеспечение уникальности полей «Название» и «ЛогинСерв» отвечают соответствующие уникальные индексы.

Аналогичные таблицы указываем для всех таблиц реляционной БД

3.2 Структура связей БД

Структура связей между таблицами реляционной БД представлена на рисунке 3.1.

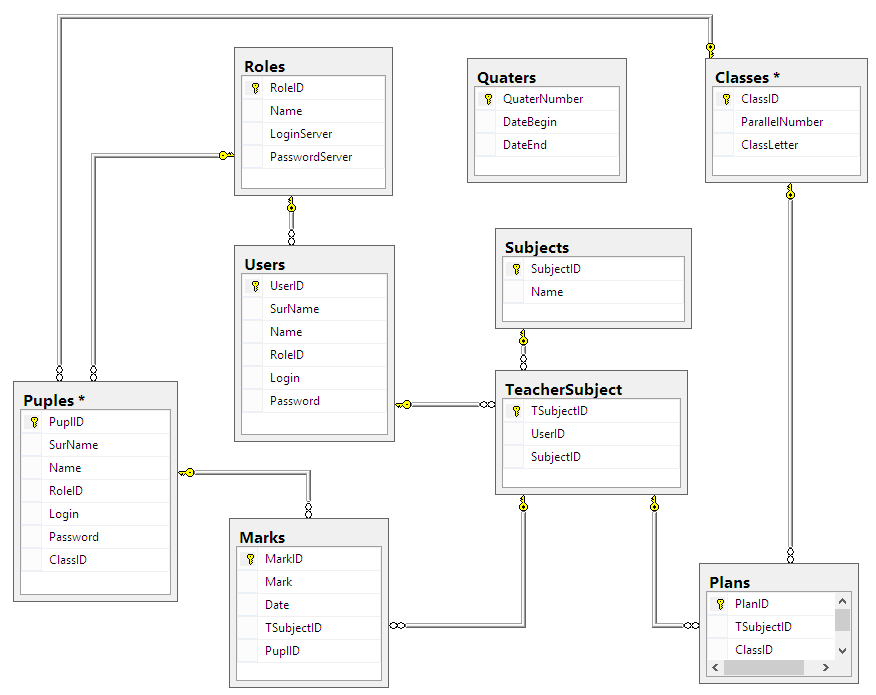


Рисунок 3.1 – Схема реляционной базы данных

3.3 Спецификация ограничений и правил поддержания целостности

В данном подразделе учитываются ограничения и правила поддержания целостности, выявленные на предыдущих этапах в п. 3.1. Новые ограничения и правила обнаружены не были.

3.4 Проверка условий нормализации

Проверка проводится отдельно для каждой таблицы на соответствие нормальным формам.

Все таблицы БД находятся в НФБК тат как:

1НФ – значения всех атрибутов атомарны;

2НФ – отсутствуют функциональные зависимости от части первичного ключа;

3НФ – отсутствуют транзитивные зависимости от первичного ключа

НФБК – отсутствуют функциональные зависимости части первичного ключа от не ключевых атрибутов.

Дальнейшая нормализация БД не требуется.