РАЗРАБОТКА БАЗ ДАННЫХ

ФИО преподавателя: Богомольная Г.В.

e-mail: bogomolnaya@mirea.ru

ТЕМА
МОДЕЛИРОВАНИЕ ДАННЫХ

План лекции

- Логическое моделирование данных:
  - ✓ метод Баркера;
  - ✓ метод IDEF1X;
  - ✓ Подход, используемый в CASE-средстве Silverrun.
- Алгоритм перехода от ER–модели к реляционной схеме данных.
- Физическое проектирование баз данных.

Моделирование данных

Цель моделирования данных - обеспечение разработчика ИС концептуальной схемой базы данных в форме одной или нескольких локальных моделей, которые могут быть отображены в любую систему баз данных.

Средство моделирования данных - диаграммы "сущность-связь".

Базовые понятия диаграммы «сущность-связь» (ERD):

Сущность (Entiry) - реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области.

Связь (Relationship) - поименованная ассоциация между двумя сущностями, значимая для предметной области, при которой каждый экземпляр одной сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, и наоборот.

Атрибут (Attribute) - любая характеристика сущности, значимая для предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности.

Моделирование данных

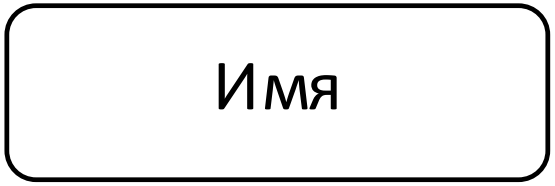
Свойства сущности:

- иметь уникальное имя:
  - к одному и тому же имени должна всегда применяться одна и та же интерпретация;
  - одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- обладать одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через связь;
- обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности.

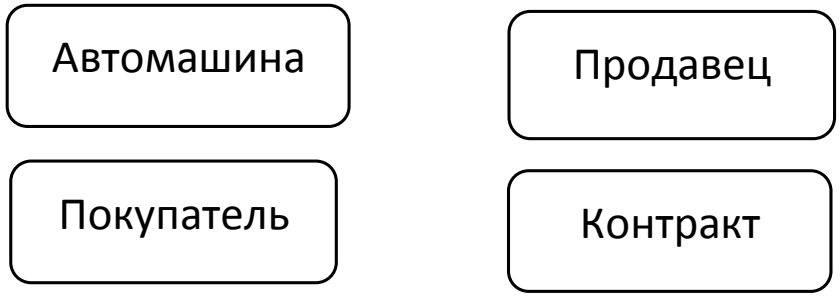
Метод Баркера

Первый шаг моделирования - извлечение информации из интервью и выделение сущностей.

Графическое изображение сущности



Выделение сущностей

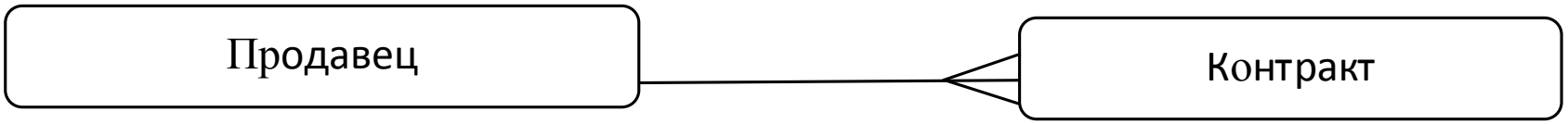


Метод Баркера

Второй шаг моделирования - идентификация связей.

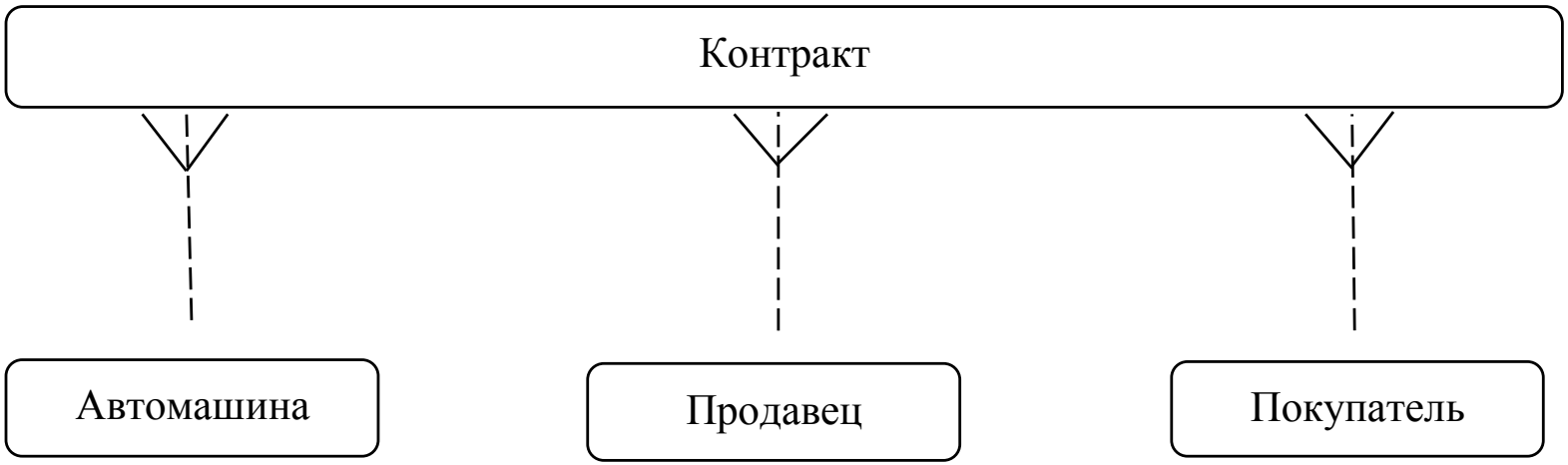
Графическое изображение степени и обязательности связи

Графическое изображение - связь продавца с контрактом



Метод Баркера

Диаграмма «сущность-связь» без атрибутов



Метод Баркера

Третий шаг моделирования - идентификация атрибутов.

Графическое изображение атрибутов:

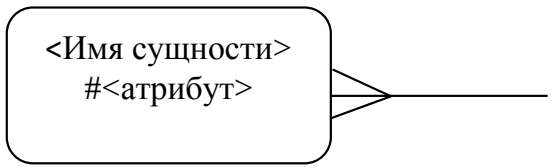
обязательный (помечен звездочкой), необязательный (помечен кружком)

<Имя сущности>
*<атрибут-1>
°<атрибут-2>

Виды идентификации:

а - полная идентификация;

б- идентификация посредством другой сущности

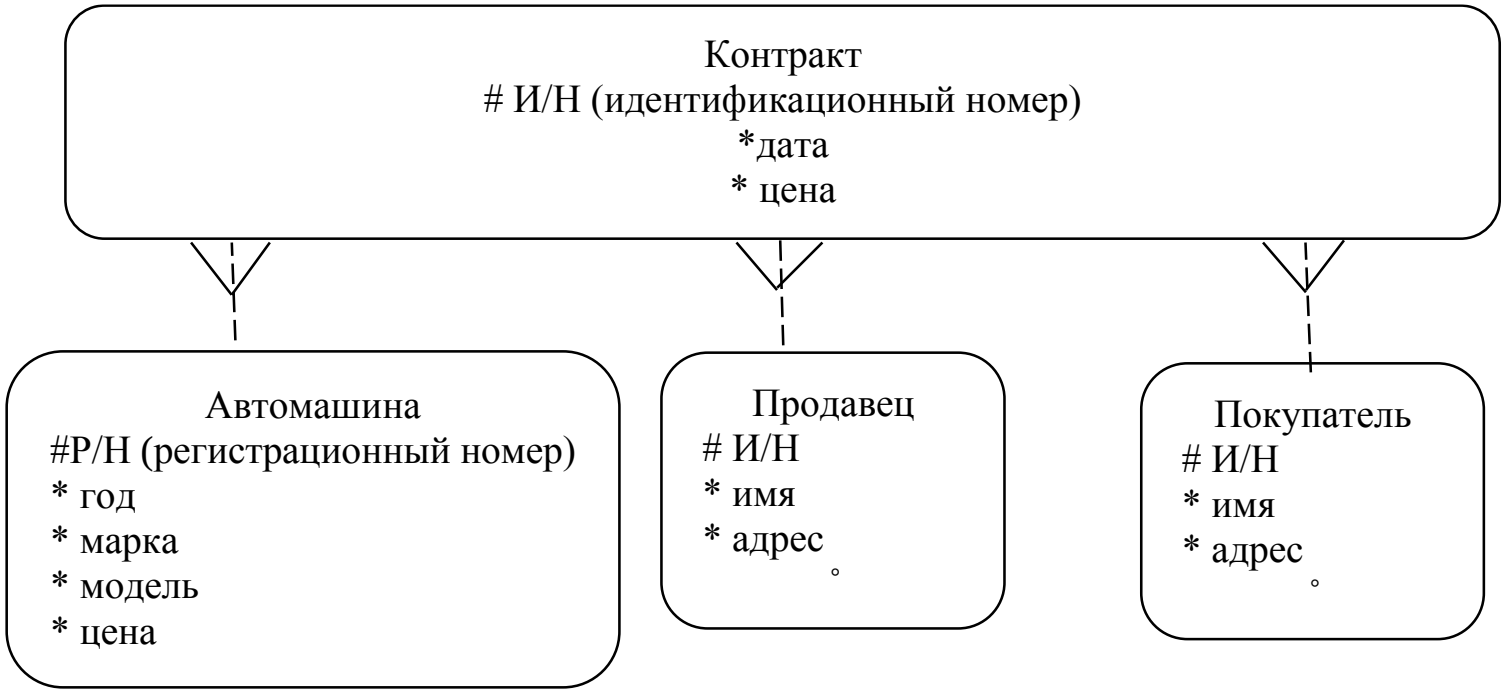


а

б

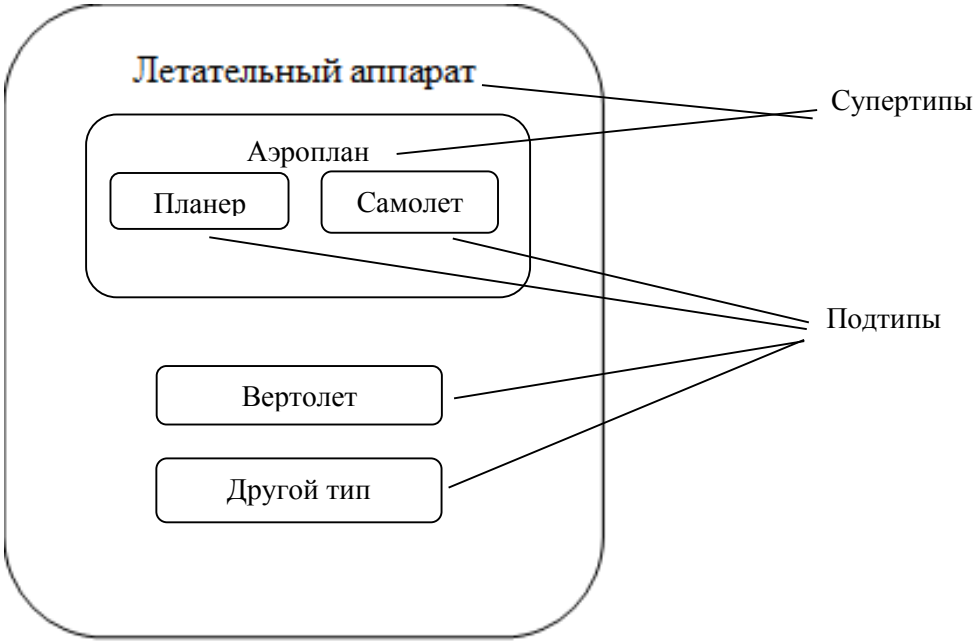
Метод Баркера

Диаграмма «сущность-связь» с атрибутами



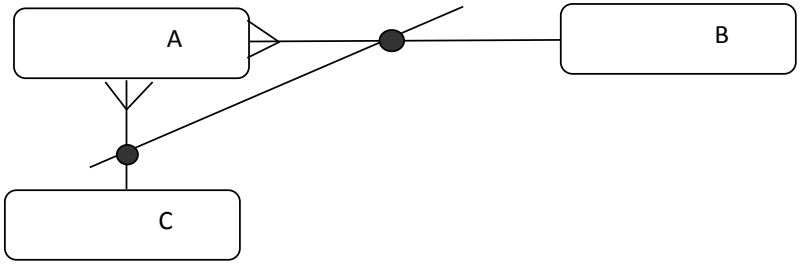
Метод Баркера

Супертипы и подтипы: одна сущность является обобщающим понятием для группы подобных сущностей

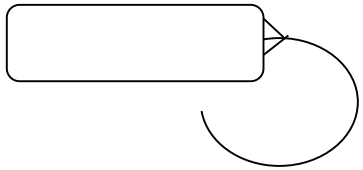


Метод Баркера

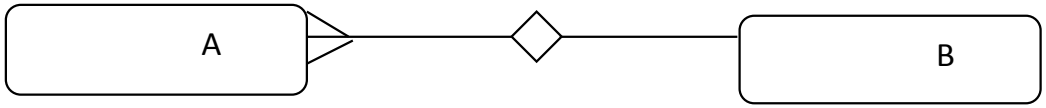
Взаимно исключающие связи: каждый экземпляр сущности участвует только в одной связи из группы взаимно исключающих связей



Рекурсивная связь: сущность может быть связана сама с собой



Неперемещаемые (non-transferrable) связи: экземпляр сущности не может быть перенесен из одного экземпляра связи в другой



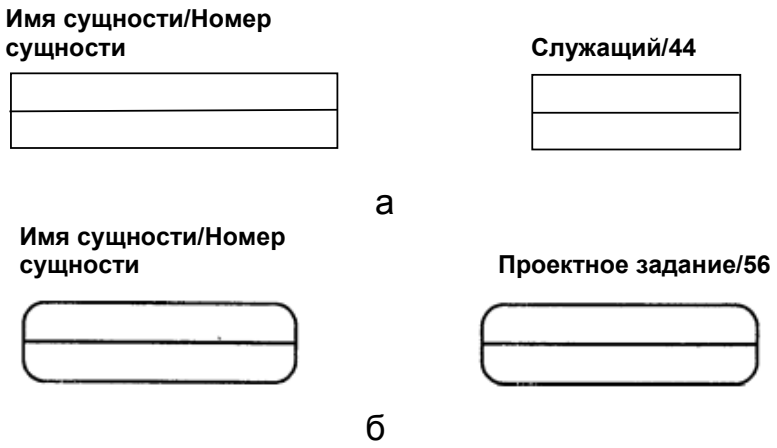
Метод IDEF1X

Метод IDEF1X основан на подходе Чена, позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме.

Сущность является не зависимой от идентификаторов, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями.

Сущность является зависимой от идентификаторов, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности.

Независимые** (а) **и зависимые** (б) **от идентификатора сущности



Метод IDEF1X

Степень/мощность связи - количество экземпляров сущности-потомка, которое может существовать для каждого экземпляра сущности-родителя.

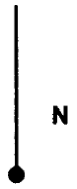
Мощность связи может принимать следующие значения:

N - ноль, один или более,

Z - ноль или один,

P - один или более,

фиксированное число.



По умолчанию мощность связи принимается равной N:

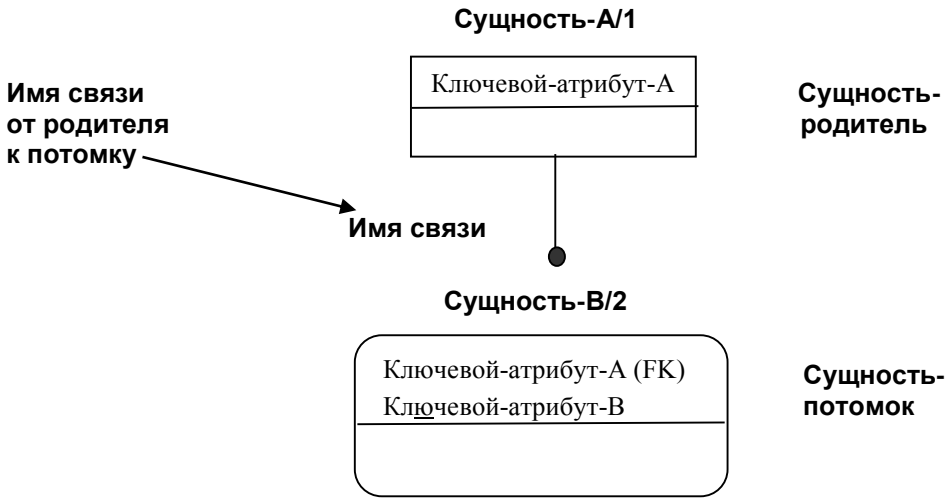
Метод IDEF1X

Идентифицирующая связь - если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем.

Сущность-потомок в идентифицирующей связи является зависимой от идентификатора сущностью.

Сущность-родитель в идентифицирующей связи может быть, как независимой, так и зависимой от идентификатора сущностью.

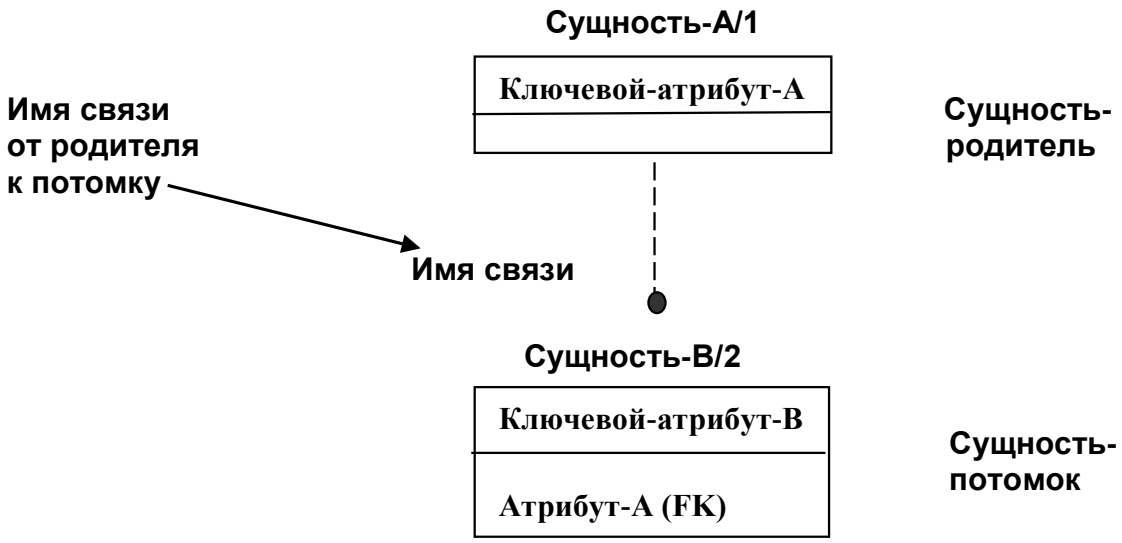
Идентифицирующая связь



Метод IDEF1X

Неидентифицирующая связь - если экземпляр сущности-потомка не определяется однозначно своей связью с сущностью-родителем

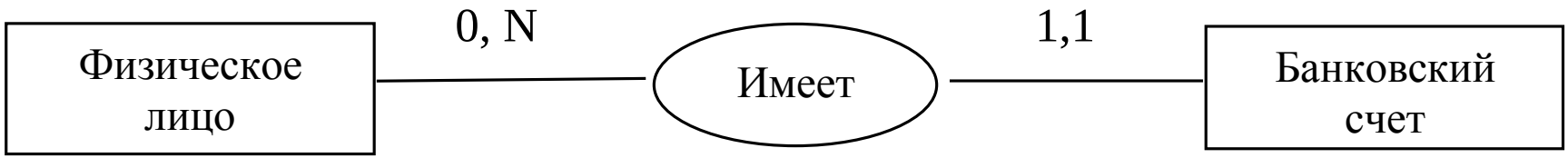
Неидентифицирующая связь



Подход, используемый в CASE-средстве Silverrun

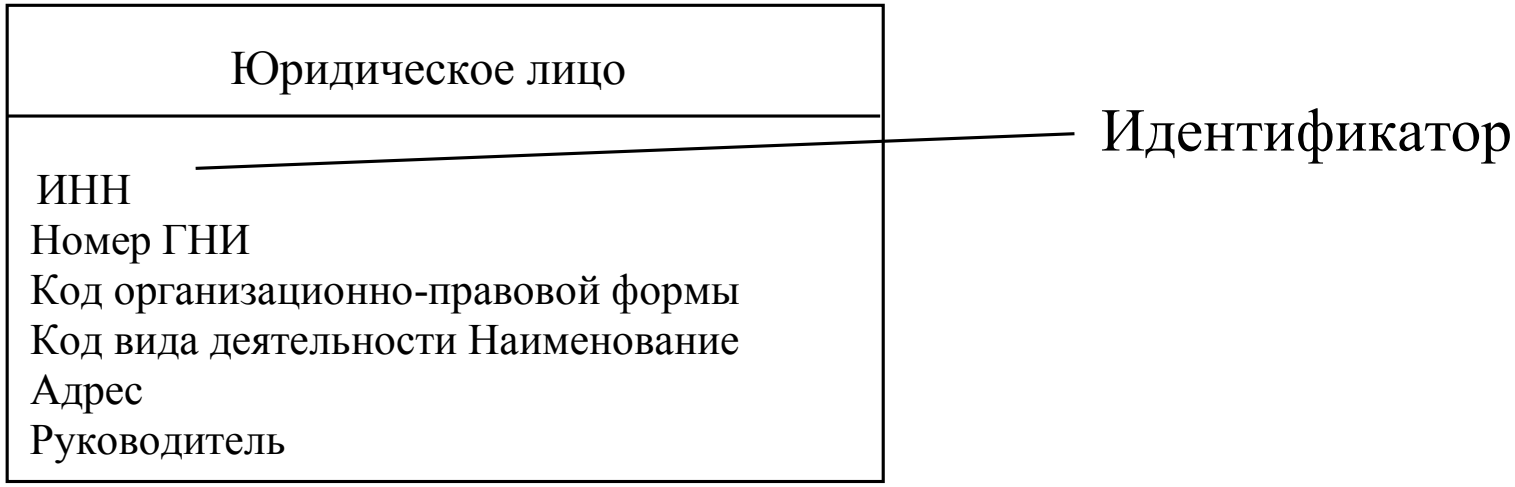
Вариант нотации Чена используется для концептуального моделирования данных на стадии формирования требований

ERD-диаграмма



Подход, используемый в CASE-средстве Silverrun

Графическое представление сущности



Подход, используемый в CASE-средстве Silverrun

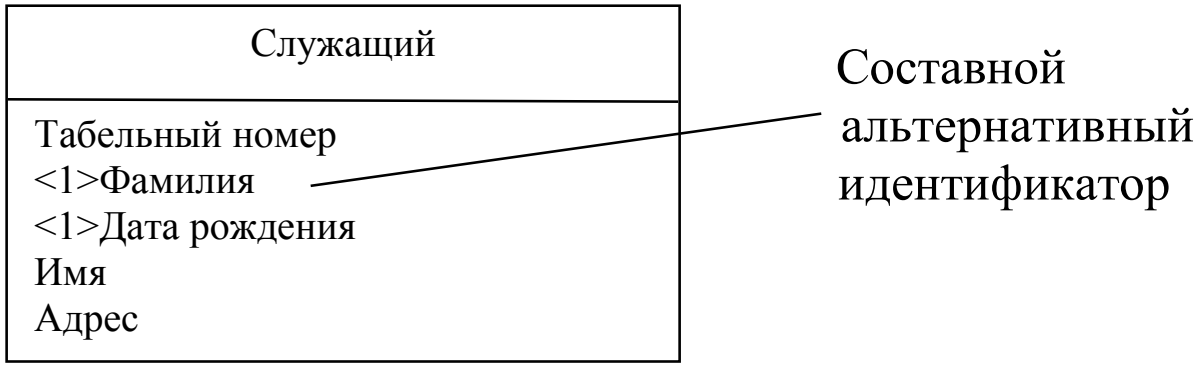
Виды идентификаторов:

- *первичный/альтернативный:*

Первичный (основной) идентификатор – один, на диаграмме подчеркивается. Альтернативные идентификаторы предваряются символами <1> для первого альтернативного идентификатора, <2> для второго и т. д.

- *простой/составной:* идентификатор, состоящий из одного атрибута, является простым, из нескольких атрибутов - составным;

Альтернативные идентификаторы



Подход, используемый в CASE-средстве Silverrun

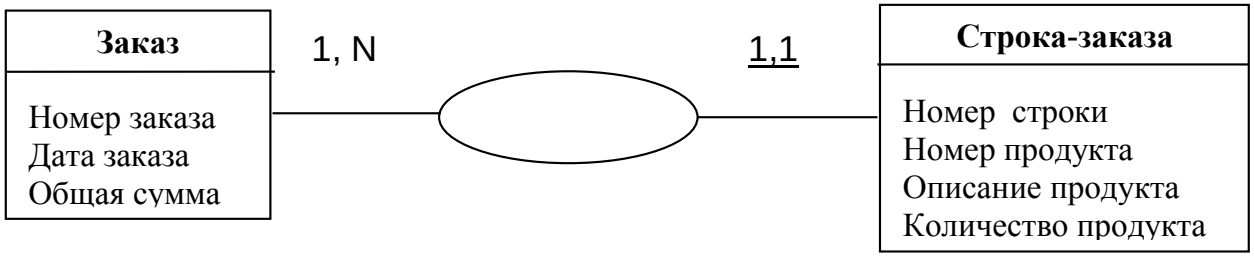
Виды идентификаторов:

- *абсолютный/относительный:*

  – абсолютный идентификатор - если все атрибуты, составляющие идентификатор, принадлежат сущности;

  – относительный идентификатор - если один или более атрибутов идентификатора принадлежат другой сущности.

Относительный идентификатор

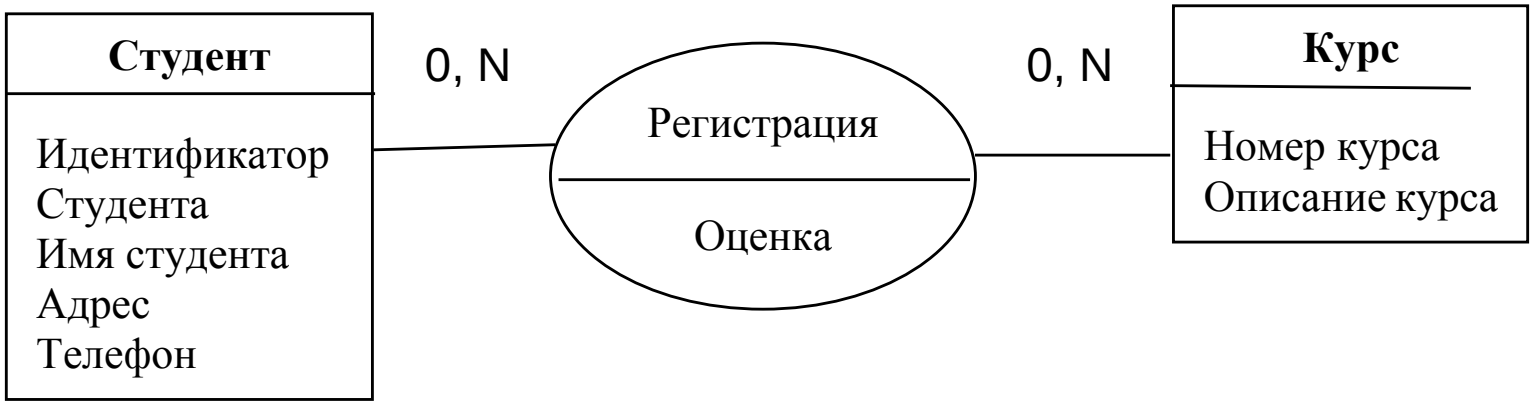


Подход, используемый в CASE-средстве Silverrun

Атрибуты связи

Связь между сущностями в концептуальной модели данных является типом, который представляет множество экземпляров связи между экземплярами сущностей.

Идентификатор связи

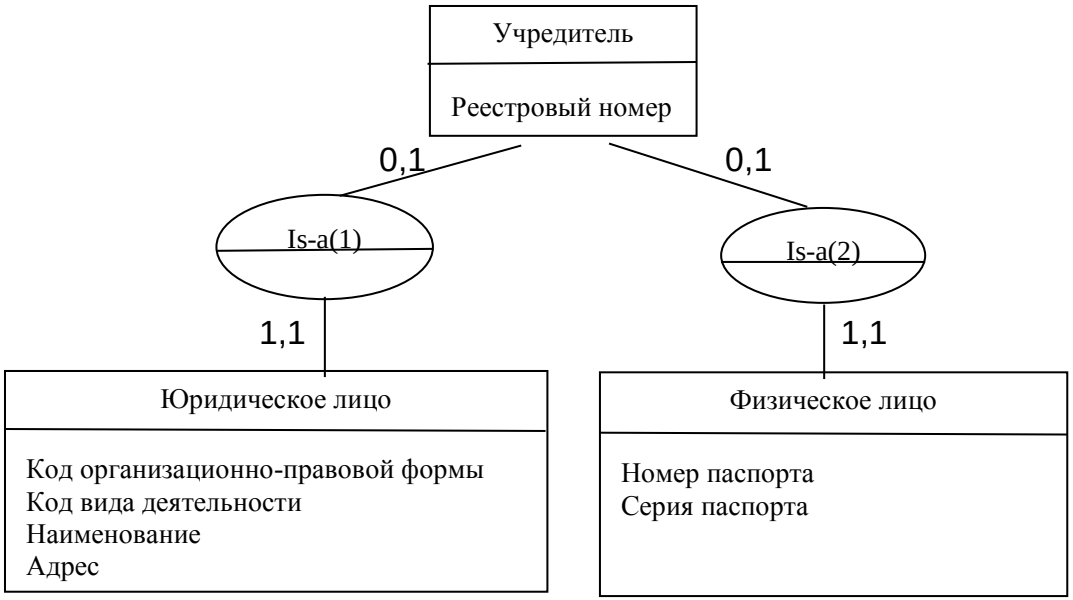


Подход, используемый в CASE-средстве Silverrun

Атрибуты связи

Связь "супертип - подтип" - общие атрибуты типа определяются в сущности - супертипе, сущность-подтип наследует все атрибуты супертипа

Связь "супертип-подтип"



Алгоритм перехода от ER–модели к реляционной схеме данных

Шаг 1. Каждая простая сущность превращается в таблицу. Имя сущности становится именем таблицы.

Шаг 2. Каждый атрибут становится возможным столбцом с тем же именем. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения; столбцы, соответствующие обязательным атрибутам, - не могут.

Шаг 3. Компоненты уникального идентификатора сущности превращаются в первичный ключ таблицы. Если в состав уникального идентификатора входят связи, к числу столбцов первичного ключа добавляется копия уникального идентификатора сущности, находящейся на дальнем конце связи.

Алгоритм перехода от ER–модели к реляционной схеме данных

Шаг 4. Связи многие-к-одному и один-к-одному становятся внешними ключами. Необязательные связи соответствуют столбцам, допускающим неопределенные значения; обязательные связи - столбцам, не допускающим неопределенные значения.

Шаг 5. Индексы создаются для первичного ключа (уникальный индекс), внешних ключей и тех атрибутов, на которых предполагается базировать запросы.

Шаг 6. Если в концептуальной схеме присутствовали подтипы, то возможны два способа:

- все подтипы в одной таблице (а);
- для каждого подтипа - отдельная таблица (б) **.**

Алгоритм перехода от ER–модели к реляционной схеме данных

Все в одной таблице	Таблица - на подтип
Преимущества	
Все хранится вместе Легкий доступ к супертипу и подтипам Требуется меньше таблиц	Более ясны правила подтипов Программы работают только с нужными таблицами
Недостатки	
Слишком общее решение Требуется дополнительная логика работы с разными наборами столбцов и разными ограничениями Потенциальное узкое место (в связи с блокировками) Столбцы подтипов должны быть необязательными В некоторых СУБД для хранения неопределенных значений требуется дополнительная память	Слишком много таблиц Смущающие столбцы в представлении UNION Потенциальная потеря производительности при работе через UNION Над супертипом невозможны модификации

Алгоритм перехода от ER–модели к реляционной схеме данных

Шаг 7. Имеется два способа работы при наличии исключающих связей:

- общий домен (а)
- явные внешние ключи (б)

Если остающиеся внешние ключи все в одном домене (способ (а)) - создаются два столбца:

- идентификатор связи
- и идентификатор сущности.

Если результирующие внешние ключи не относятся к одному домену - для каждой связи создаются явные столбцы внешних ключей.

ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

Особенности построения физической модели базы данных

Физический уровень –отображение логической модели на модель данных конкретной СУБД.

Физическое проектирование - преобразование логической схемы с учетом синтаксиса, семантики и возможностей выбранной целевой СУБД.

Проблемы проектирования базы данных

- *Проблема логического проектирования баз данных*: Каким образом отобразить объекты предметной области в абстрактные объекты модели данных?
- *Проблема физического проектирования баз данных:* Как обеспечить эффективность выполнения запросов к базе данных?

Особенности построения физической модели базы данных

Основные определения элементов физической модели

Физический тип данных – тип данных, характеризующий столбец с данными.

Уникальный индекс первичного ключа – индекс, передающий столбцу в таблице все свойства первичного ключа.

Хранимая процедура - объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

Триггер – хранимая процедура, запускаемая СУБД автоматически, при наступлении определенного в коде хранимой процедуры события.

Внешний ключ – подмножество столбцов некоторой переменной таблицы R2, значения которых должны совпадать со значениями некоторого первичного ключа некоторой переменной таблицы R1.

Особенности построения физической модели базы данных

Термины физической модели

Сущность (концептуальная или логическая модель)	Таблица (физическая модель)
Зависимая сущность	Первичный ключ родителя, как часть первичного ключа потомка
Независимая сущность	Первичный ключ родителя, как неключевой атрибут потомка
Атрибут	Столбец
Логический тип данных (text, number, clob)	Физический тип данных (зависит от СУБД)
Домен (логический)	Домен (физический)
Первичный ключ	Первичный ключ, уникальный кластеризованный индекс первичного ключа
Внешний ключ	Внешний ключ, уникальный некластеризованный индекс внешнего ключа
Альтернативный ключ	Альтернативный ключ, уникальный некластеризованный индекс альтернативного ключа
Бизнес правило	Триггер или хранимая процедура
Связь	Связь, поддерживаемая внешними ключами
Идентифицирующая связь	Первичный ключ родителя становится частью первичного ключа потомка
Неидентифицирующая связь	Первичный ключ родителя становится неключевым атрибутом потомка

Особенности построения физической модели базы данных

Этапы физического проектирования баз данных

1. Проектирование таблиц базы данных с учетом специфики выбранной СУБД.
2. Реализация бизнес-правил в выбранной СУБД.
3. Дальнейшая оптимизация физической модели базы данных.
4. Разработка стратегии обеспечения безопасности информации.
5. Осуществление постоянного мониторинга базы данных и СУБД.

Особенности построения физической модели базы данных

Анализ необходимости введения контролируемой избыточности

Денормализация – снижение требований к уровню нормализации отношений.

Виды денормализации, повышающие производительность системы

1. Использование производных данных:
   - дополнительная стоимость хранения производных данных и поддержки согласованности с текущими значениями исходных данных;
   - издержки на выполнение вычислений значений производных атрибутов при каждом обращении к ним.
2. Дублирование атрибутов.

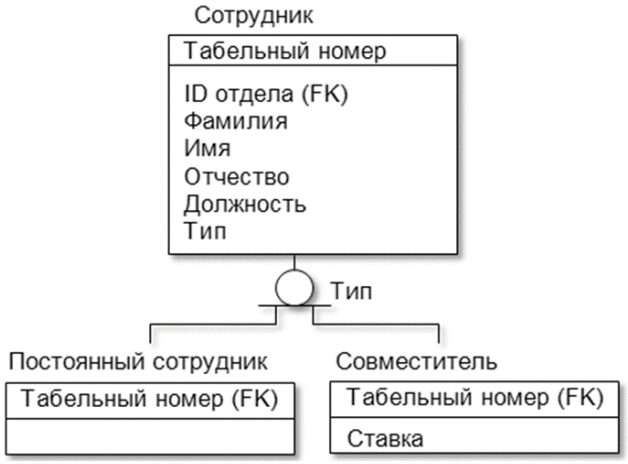
Особенности построения физической модели базы данных

Анализ необходимости введения контролируемой избыточности

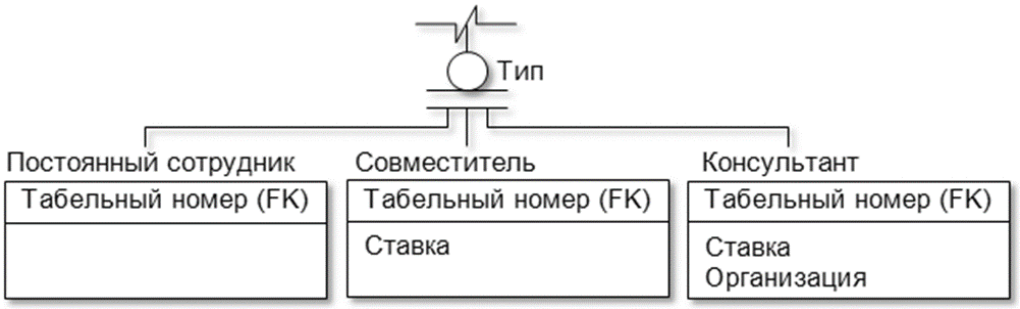
Дублирование атрибутов

2.1. Объединение отношений, связанных 1:1.

Иерархия наследования
(неполная категория)



Иерархия наследования
(полная категория)



Особенности построения физической модели базы данных

Анализ необходимости введения контролируемой избыточности

Дублирование атрибутов

2.2. Дублирование атрибутов в связях типа 1:M:

- возможность включения атрибута одной таблицы в другую таблицу.

2.3. Использование служебных таблиц:

- значительно снижается вероятность ошибки при указании значений для атрибутов;
- уменьшается размер исходной таблицы;
- при изменении описания параметра значительно проще изменить одно значение в служебной таблице, чем корректировать множество записей в исходной.

2.4. Введение повторяющихся (многозначных) атрибутов.

2.5. Создание сводных таблиц.

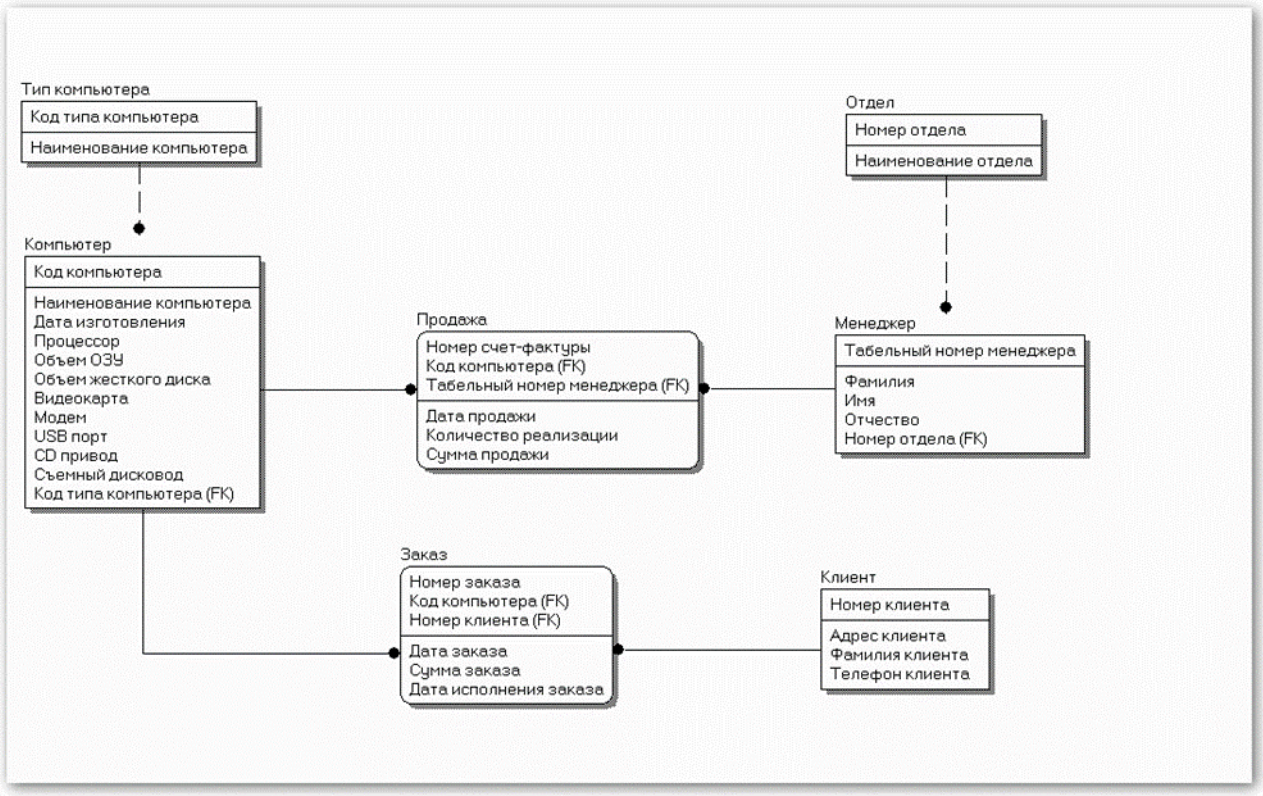
Особенности построения физической модели базы данных

Перенос логической схемы данных в среду целевой СУБД

1. Проектирование таблиц и связей.
2. Задание:
   - доменов;
   - первичных, альтернативных и внешних ключей;
   - неопределенных (NULL) и обязательных (NOT NULL) значений;
   - значений по умолчанию (DEFAULT);
   - правил контроля целостности;
   - хранимых процедур и триггеров.
3. Модификация логической схемы с учетом семантики и синтаксиса, принятой в целевой СУБД.

Особенности построения физической модели базы данных

Логическая модель данных системы "Реализация средств вычислительной техники"



Особенности построения физической модели базы данных

Перенос логической схемы данных в среду целевой СУБД

Правила ссылочной целостности

Правило целостности внешних ключей:

- для каждого значения внешнего ключа должно существовать соответствующее значение первичного ключа в родительском отношении.

Ссылочная целостность может быть нарушена при выполнении операций:

1) обновление кортежа в родительском отношении;
2) удаление кортежа в родительском отношении;
3) вставка кортежа в дочернее отношение;
4) обновление кортежа в дочернем отношении.

Особенности построения физической модели базы данных

Перенос логической схемы данных в среду целевой СУБД

Основные стратегии поддержания ссылочной целостности:

1. RESTRICT – не разрешать выполнение операции, приводящей к нарушению ссылочной целостности.
2. CASCADE – разрешить выполнение требуемой операции, но внести при этом необходимые поправки в других кортежах отношений так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи.

Дополнительные стратегии поддержания ссылочной целостности:

1. NONE – никаких операций по поддержке ссылочной целостности не выполняется.
2. SET NULL – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей заменять на неопределенные значения (null-значения).
3. SET DEFAULT – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на некоторое значение, принятое по умолчанию.

Особенности построения физической модели базы данных

Реализация бизнес-правил и анализ транзакций

После реализации бизнес-правил необходимо проверить выполнимость и эффективность всех транзакций.

Разработка механизмов защиты

Разработка пользовательских представлений

Представление в БД – динамический результат одной или более операций, выполненных над таблицами БД с целью получения новой сводной таблицы. Представление является виртуальной таблицей, которая реально в БД не существует, но создается по запросу (SELECT) определенного пользователя в результате выполнения этого запроса.

Определение прав доступа

Каждый пользователь обладает строго определенным набором прав (привилегий) в отношении конкретной таблицы или представления.

Особенности построения физической модели базы данных

Организация мониторинга и настройка функционирования системы

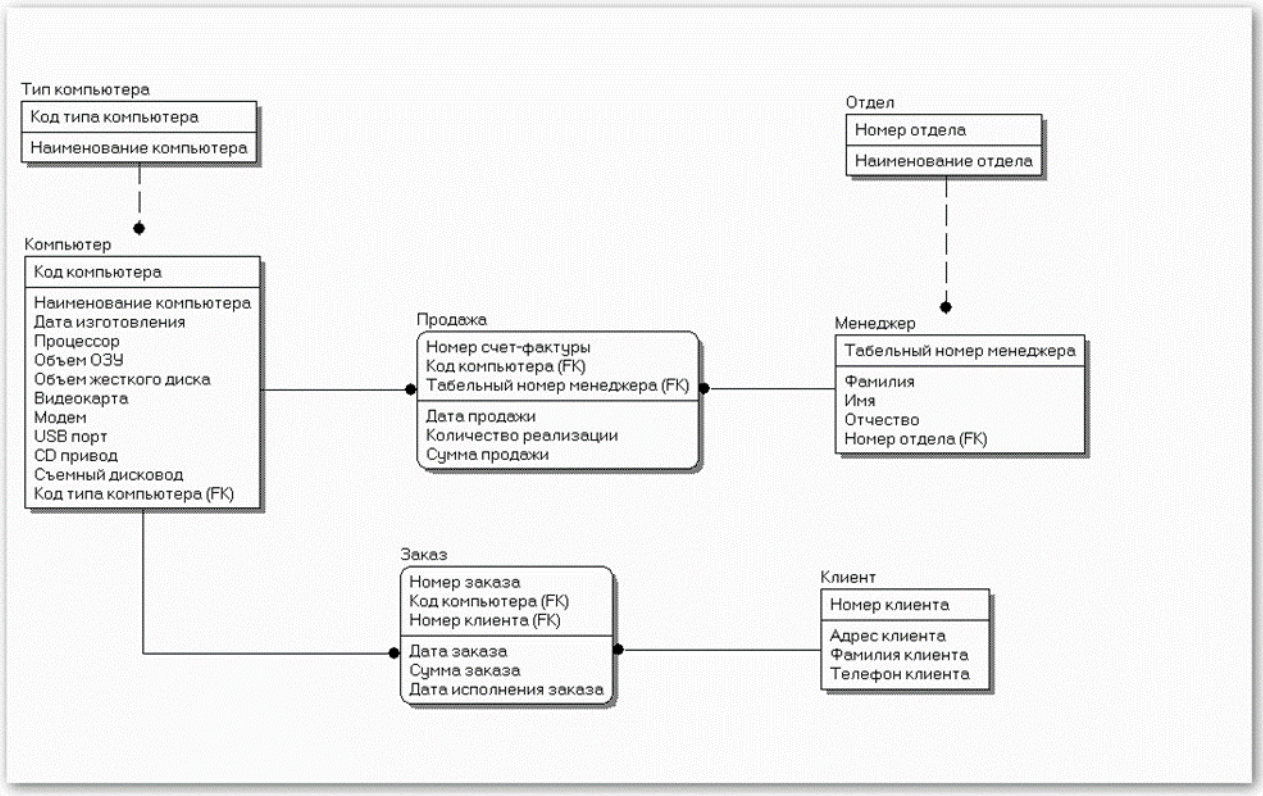
Мониторинг необходим с целью устранения ошибочных проектных решений или изменения требований к системе.

На протяжении всего жизненного цикла системы необходимо постоянно вести наблюдение за уровнем ее производительности, что позволит своевременно реагировать на изменения, происходящие в окружающей среде.

Внесение любых изменений в БД должно проводиться с обязательным их тестированием.

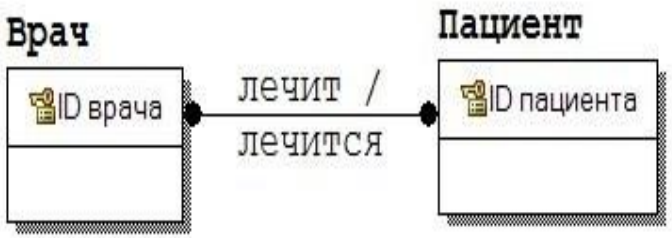
Особенности построения физической модели базы данных

Логическая модель данных системы "Реализация средств вычислительной техники"



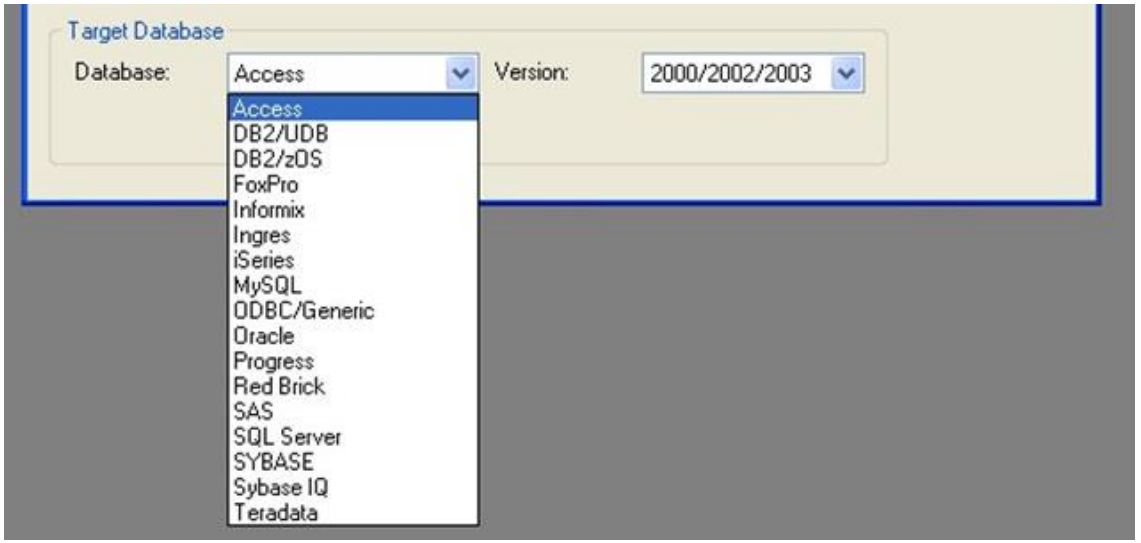
Особенности построения физической модели базы данных

Логическая модель данных системы



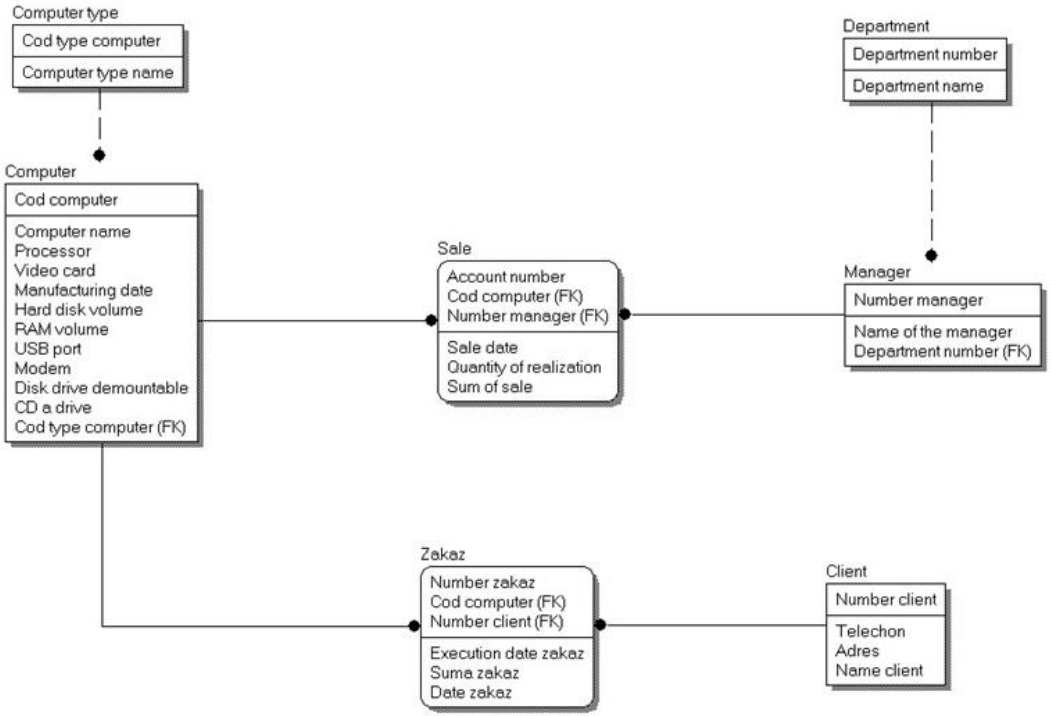
Особенности построения физической модели базы данных

Выбор СУБД Target Database



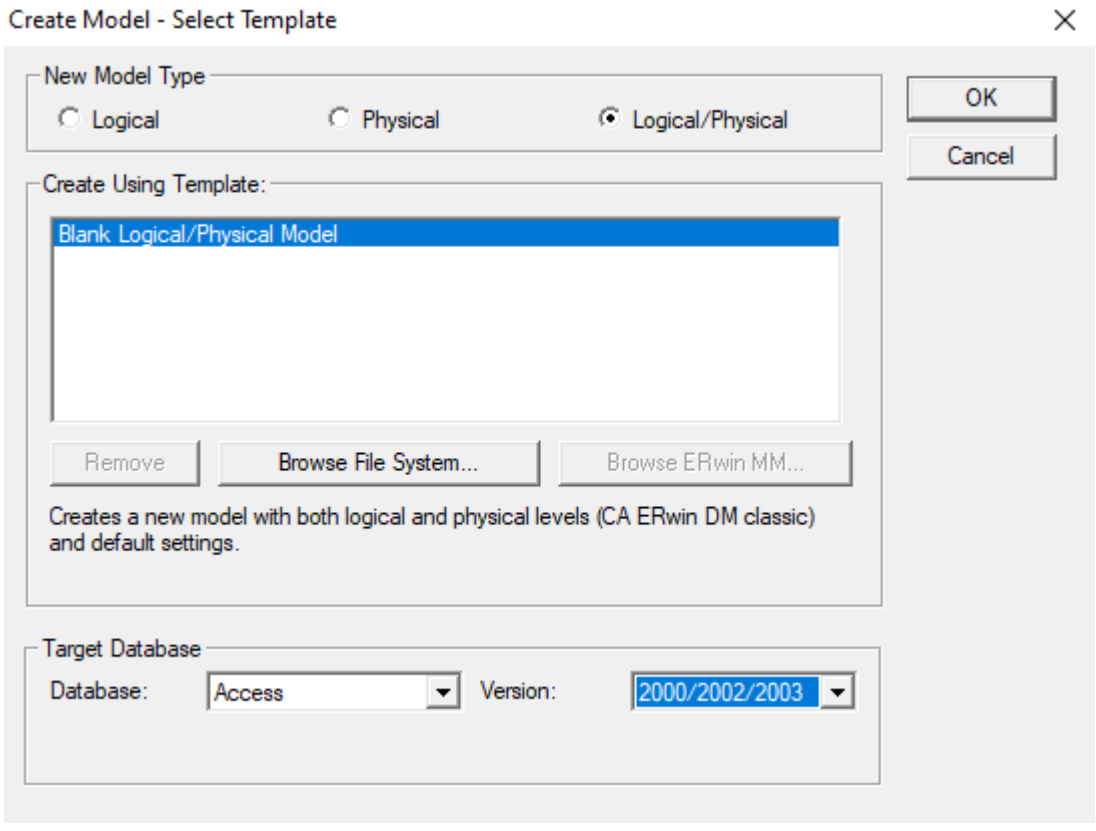
Особенности построения физической модели базы данных

Физическая модель данных системы "Реализация средств вычислительной техники"



Пример построения модели базы данных

Логическая модель данных системы «Деятельность автосалона"



Пример построения модели базы данных

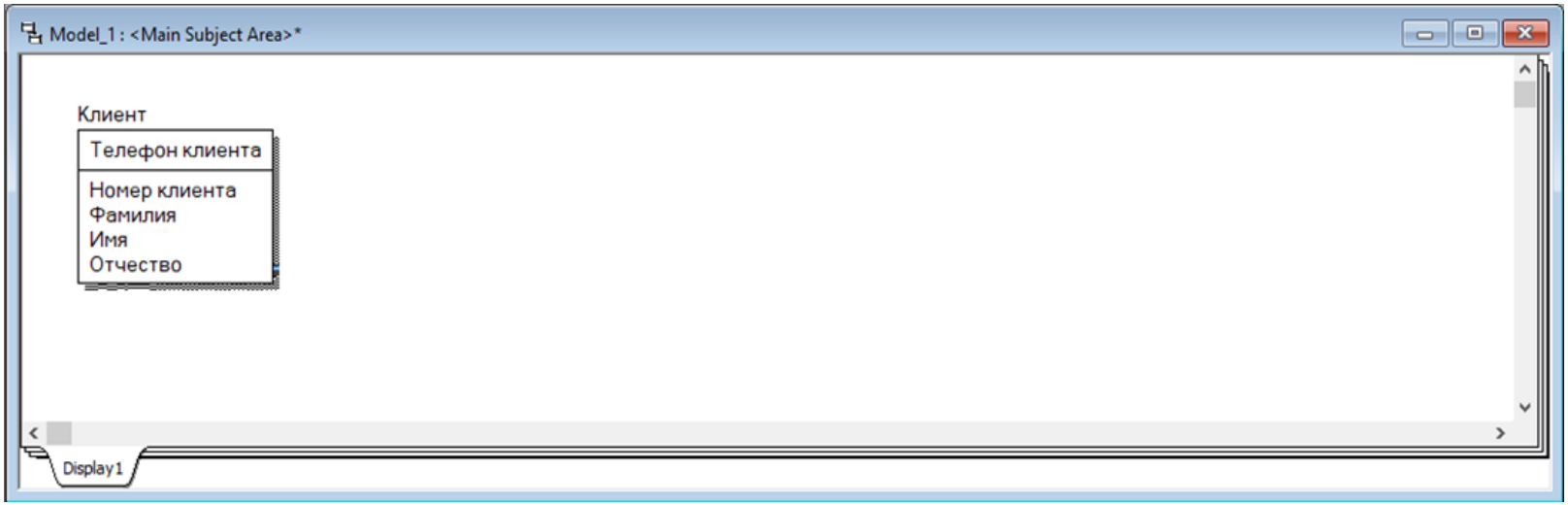
Логическая модель данных системы «Деятельность автосалона"

Создадим новую сущность «Клиент» со следующими атрибутами:

- Номер клиента (Primary Key, Number);
- Фамилия (String);
- Имя (String);
- Отчество (String);
- Телефон клиента (String).

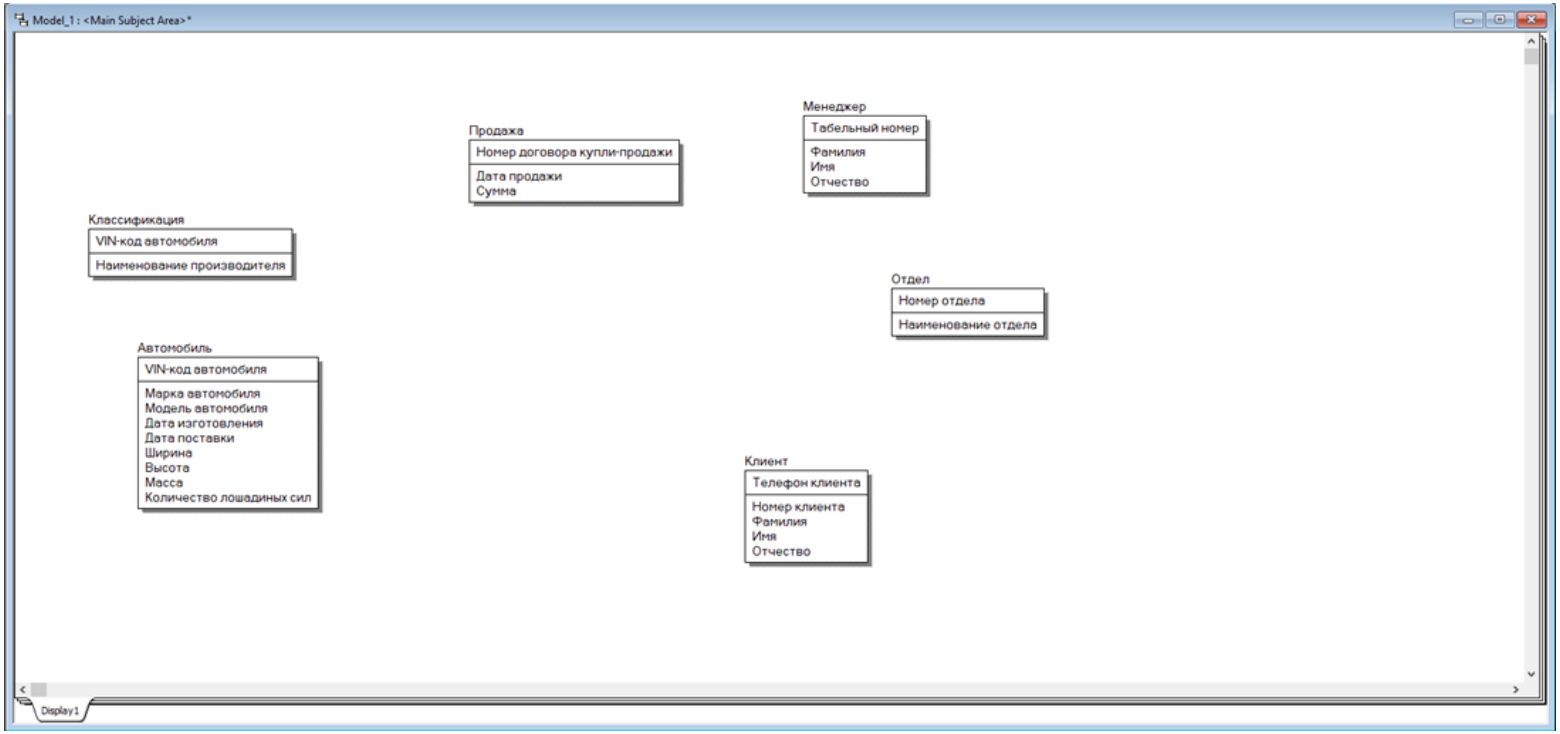
Пример построения модели базы данных

Логическая модель данных системы «Деятельность автосалона"



Пример построения модели базы данных

Логическая модель данных системы «Деятельность автосалона"



Пример построения модели базы данных

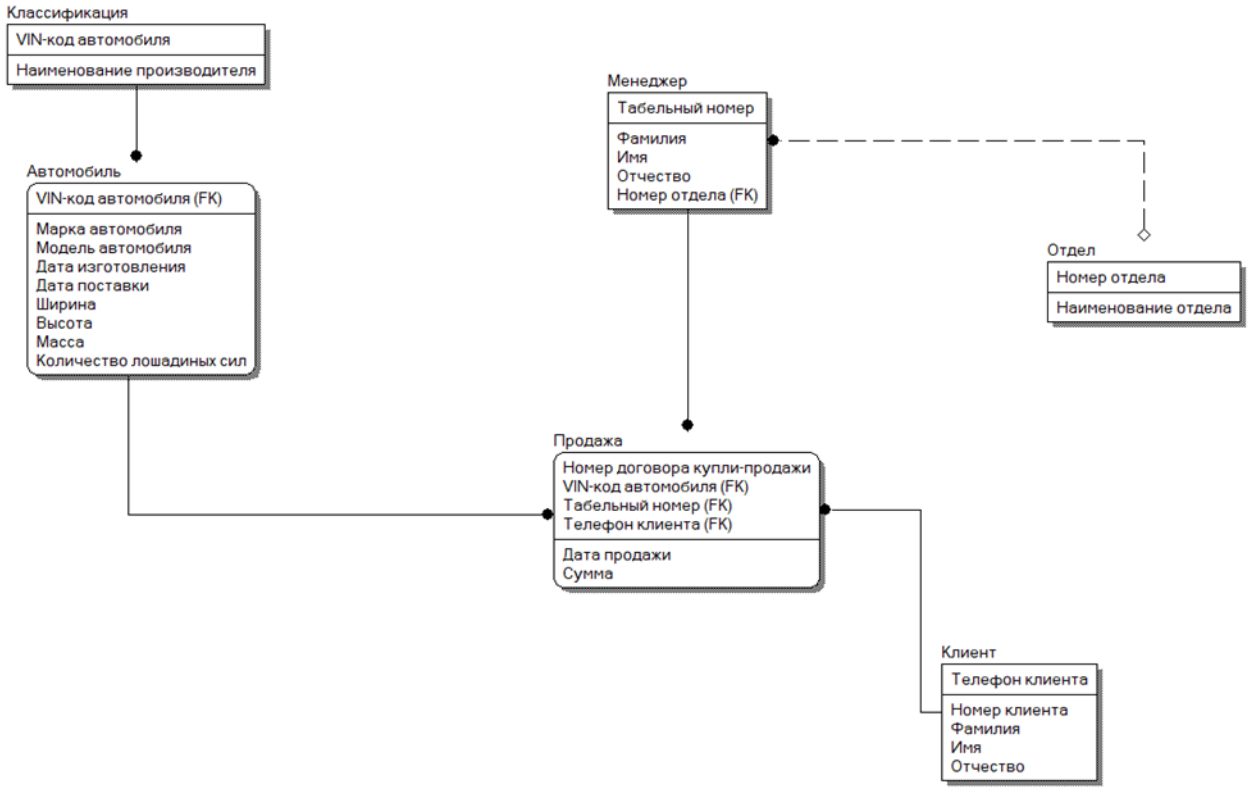
Логическая модель данных системы «Деятельность автосалона"

Установим связи между сущностями:

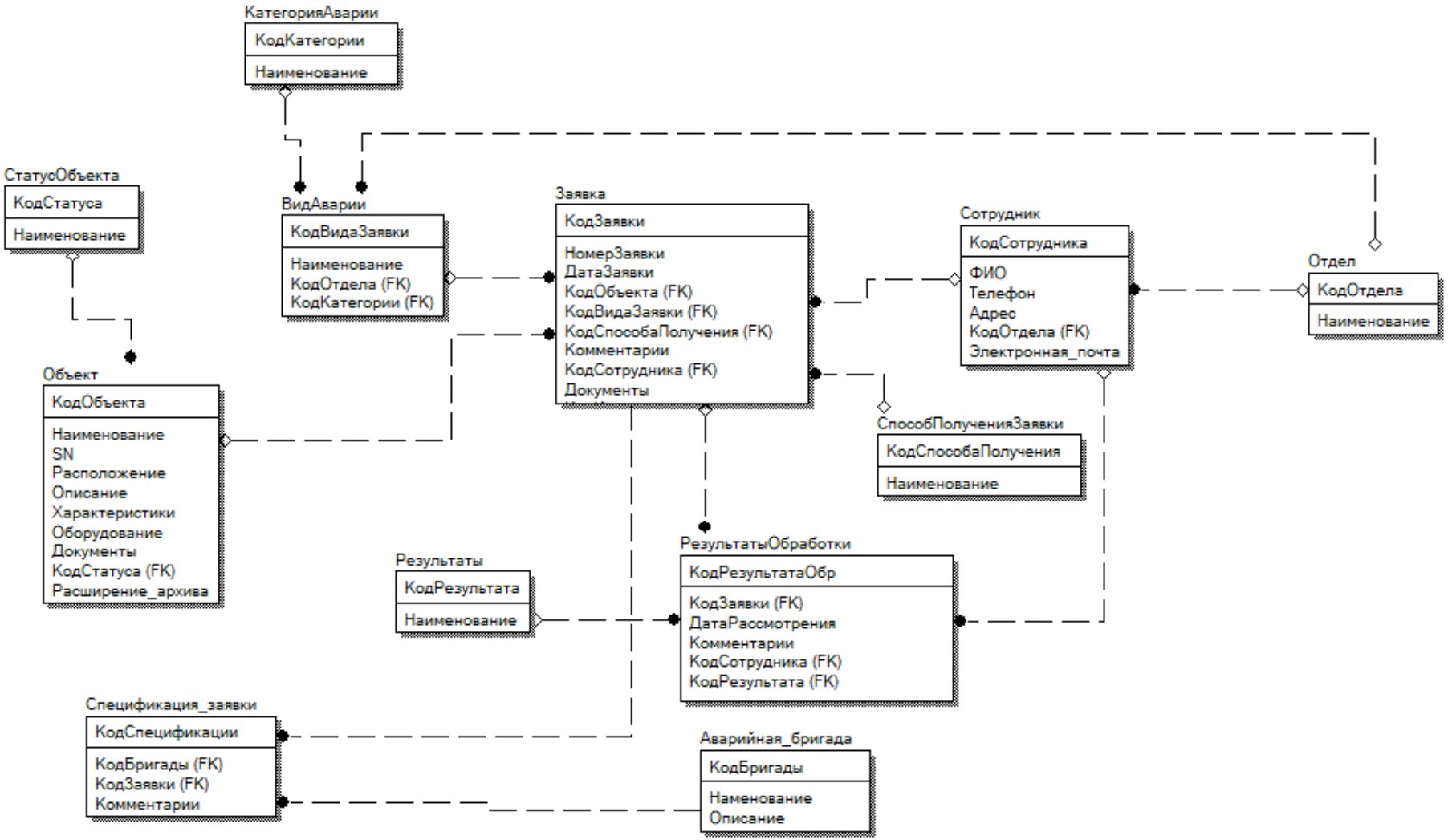
- классификация → автомобиль (идентифицирующая связь);
- автомобиль → продажа (идентифицирующая связь);
- отдел → менеджер (неидентифицирующая связь);
- менеджер → продажа (идентифицирующая связь);
- клиент → продажа (идентифицирующая связь).

Пример построения модели базы данных

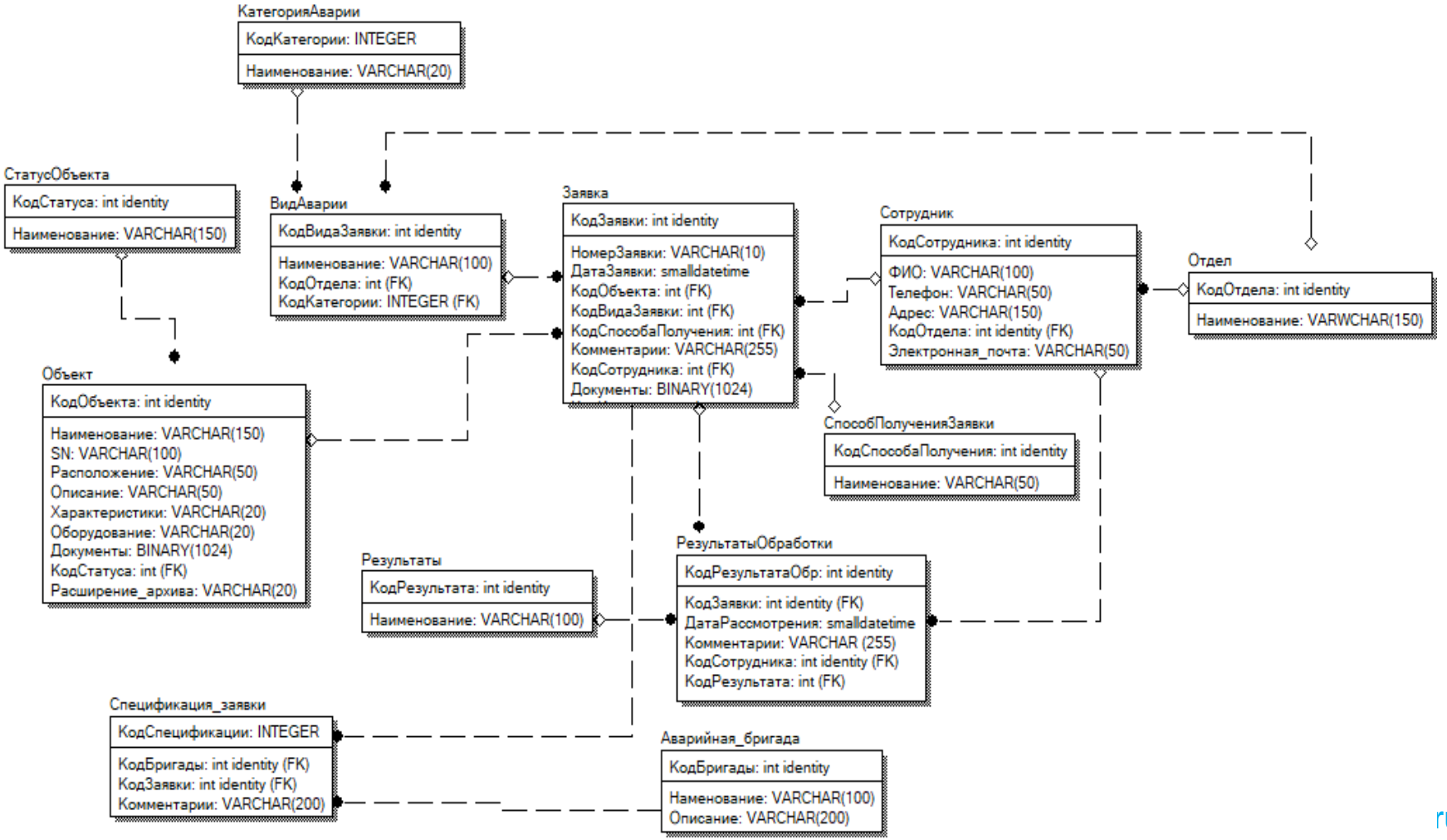
Логическая модель данных системы «Деятельность автосалона"



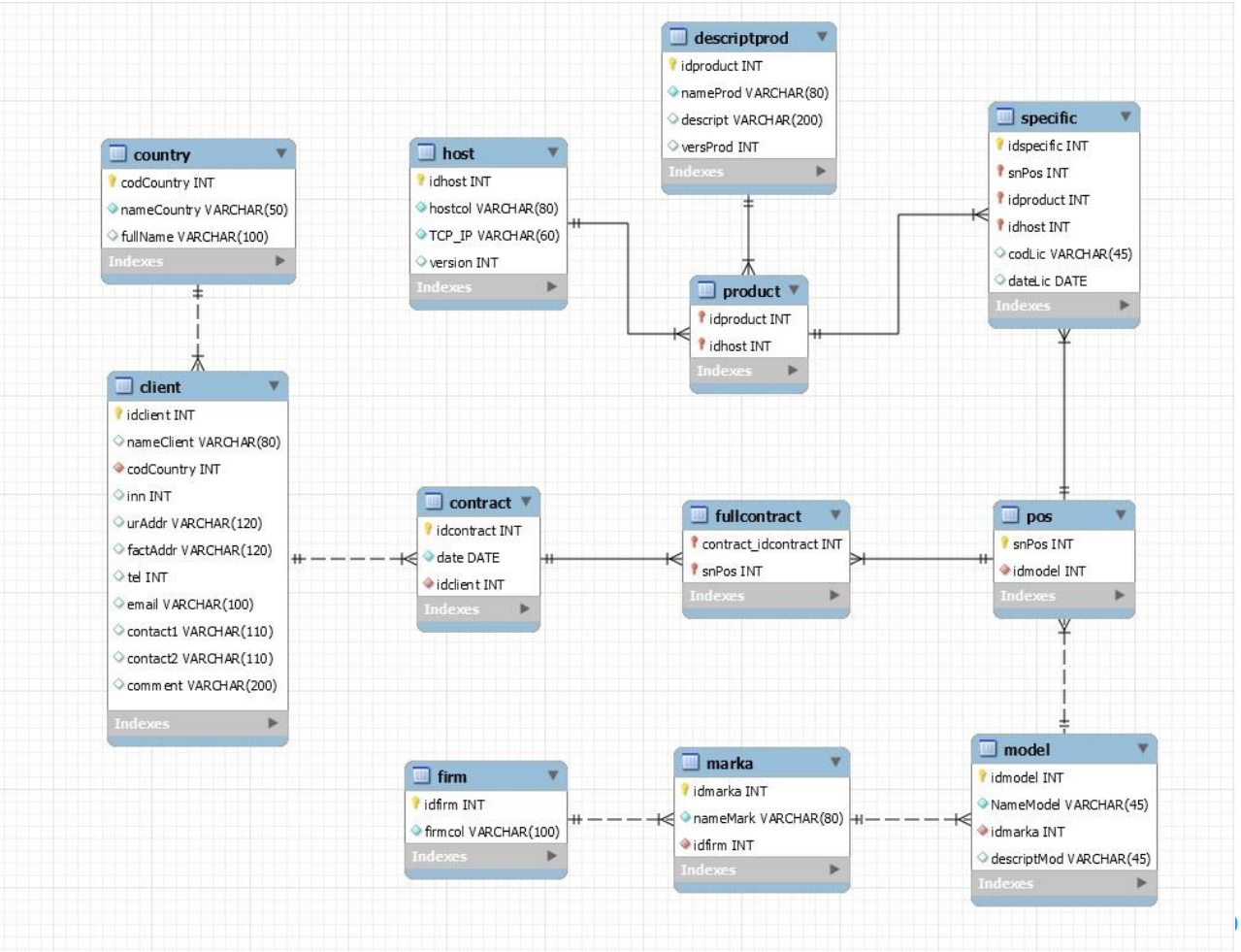
Пример: *логическая модель данных системы «Обработка и контроль заявок аварийной службы»*



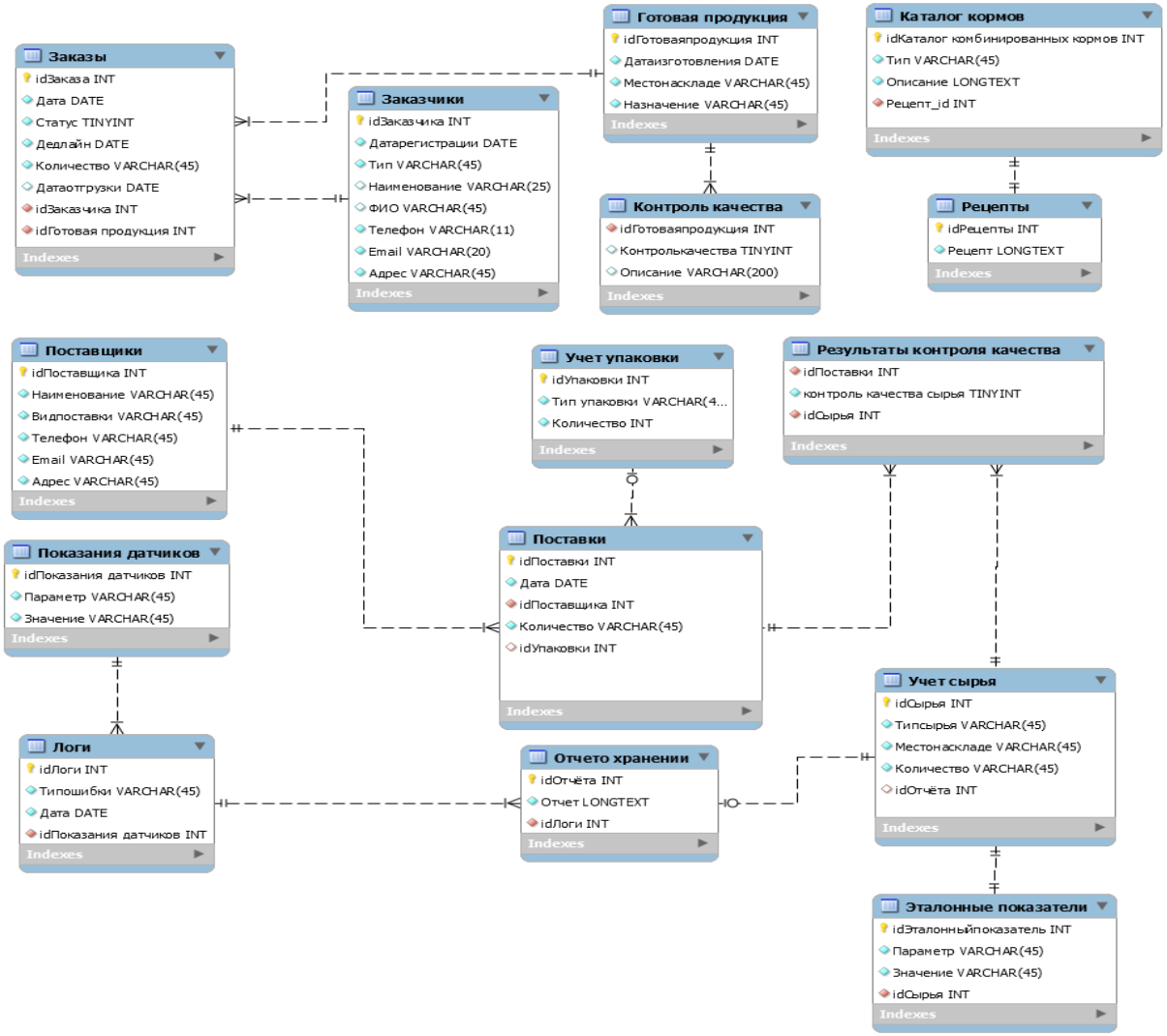
Пример: *физическая модель данных системы «Обработка и контроль заявок аварийной службы»*



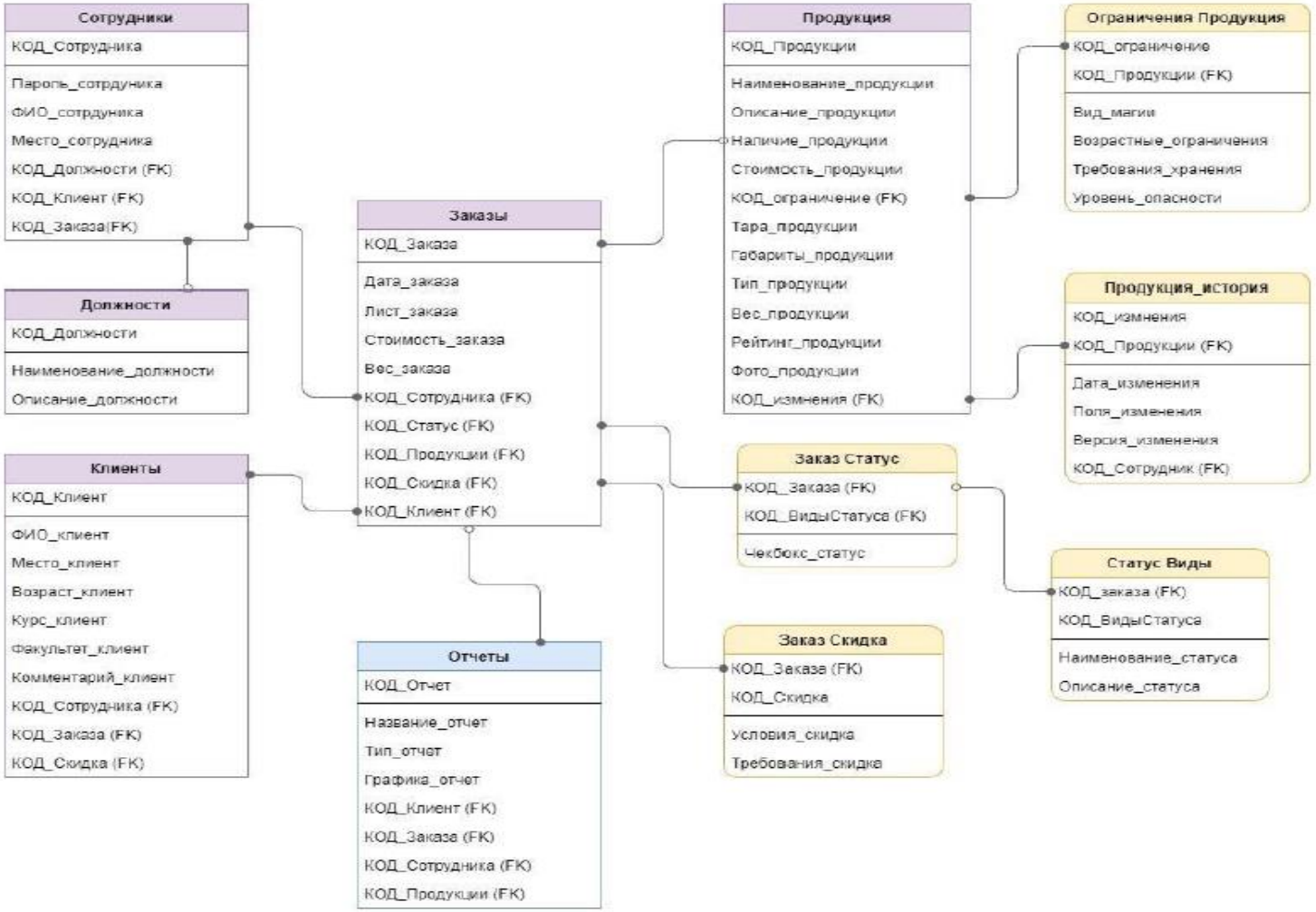
Пример: *физическая модель данных системы «Отслеживание и выдача лицензий на ПО»*



Пример: *физическая модель данных системы «Производство комбинированного корма»*



Пример: *физическая модель данных системы «Обработка почтовых заказов»*



Спасибо за внимание!

РАЗРАБОТКА БАЗ ДАННЫХ

ФИО преподавателя: Богомольная Г.В.

e-mail: bogomolnaya@mirea.ru

ТЕМА
МОДЕЛИРОВАНИЕ ДАННЫХ

План лекции

- Функциональные зависимости
- Основные аксиомы Армстронга
- Алгоритм процесса последовательной нормализации

Нормализация

Уровни моделирования:

- модель предметной области;
- логическая модель данных;
- физическая модель данных;
- база данных и приложения.

Критерии логической модели данных:

- адекватность БД предметной области;
- легкость разработки и сопровождения БД;
- скорость выполнения операций модификации данных (вставка, обновление, удаление);
- скорость выполнения операций выборки данных.

Нормализация

Уровень логического моделирования включает:

- описание концептуальной схемы БД в терминах выбранной СУБД;
- описание внешних моделей в терминах выбранной СУБД;
- описание декларативных правил поддержки целостности базы данных;
- разработку процедур поддержки семантической целостности БД.

Создание логической модели - процесс разработки корректной схемы реляционной БД.

Корректная схема БД - в которой отсутствуют нежелательные зависимости между атрибутами отношений.

Нормализация

Разработка корректной схемы реляционной БД может быть выполнена:

- ***путём декомпозиции*** (разбиения), когда исходное множество отношений, входящих в схему БД заменяется другим множеством отношений, являющихся проекциями исходных отношений;
- ***путем синтеза***, т.е. путем компоновки из заданных исходных элементарных зависимостей между объектами предметной области схемы БД.

Декомпозиция должна сохранять **эквивалентность** схем БД.
Схемы БД называются эквивалентными, если содержание исходной БД может быть получено путем естественного соединения отношений, входящих в результирующую схему, и при этом не появляется новых кортежей в исходной БД.

Нормализация

При создании логической модели (с использованием декомпозиции) ***требуется реализовать алгоритм процесса последовательной нормализации схем отношений.***

В теории реляционных БД выделяется последовательность шагов:
- приведение к первой нормальной форме (**1NF**);
- приведение ко второй нормальной форме (**2NF**);
- приведение к третьей нормальной форме (3NF);
- приведение к нормальной форме Бойса-Кодда (**BCNF**);
- приведение к четвертой нормальной форме (**4NF**);
- приведение к пятой нормальной форме / форме проекции-соединения (**5NF**).

При этом каждая последующая итерация соответствует нормальной форме более высокого уровня и обладает лучшими свойствами по сравнению с предыдущей.

Нормализация

Шаг 1 (Приведение к 1NF)

- задается одно / несколько отношений, отображающих понятия предметной области;
- по модели предметной области выписываются обнаруженные ***функциональные зависимости (ФЗ)***;
- все отношения автоматически находятся в 1NF.

Отношение в 1NF обладает следующими свойствами:

- в отношении нет одинаковых кортежей;
- кортежи не упорядочены;
- атрибуты не упорядочены;
- все значения атрибутов атомарны (на пересечении каждого столбца и каждой строки находятся только элементарные значения атрибутов).

Ненормализованное отношение

Преподаватель	День недели	Номер пары	Название дисциплины	Тип занятий	Группа
Петров В.И.	Пон.	1	БД	Лекция	ИП3
	Вт.	1	ВС	Лаб. р.	ИП2
	ВТ.	2	БД	Лаб. р.	ИП3

Отношение в 1NF

Преподаватель	День недели	Номер пары	Название дисциплины	Тип занятий	Группа
Петров В.И.	Пон.	1	БД	Лекция	ИП3
Петров В.И.	Вт.	1	ВС	Лаб. р.	ИП2
Петров В.И.	ВТ.	2	БД	Лаб. р.	ИП3

Нормализация

Шаг 2 (Приведение к 2NF)

*Отношение находится в **2NF** тогда и только тогда, когда оно находится в первой нормальной форме и не содержит неполных функциональных зависимостей непервичных атрибутов от атрибутов первичного ключа.*

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

(K_1, K_2) - сложный (составной) ключ.

Функциональные зависимости:

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

$(K_1) \rightarrow (A_1, .., A_n)$ - 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

Приведение исходного отношения к 2NF

Структура исходного отношения

ФИО	**Номер зач._кн.**	Группа	**Дисциплина**	Оценка

Разбиение исходного отношения на проекции

ФИО	**Номер зач._кн.**	Группа

Номер_зач.кн.	Дисциплина	Оценка

Нормализация

Шаг 2 (Приведение к 2NF)

Функциональной зависимостью *набора атрибутов* ***B*** *отношения* ***R*** *от набора атрибутов* ***A*** *того же отношения, обозначаемой как* ***R.A -> R.B*** *или* ***A -> B*** *называется такое соотношение проекций* ***R[A]*** *и* ***R[B]****, при котором в каждый момент времени любому элементу проекции* ***R[A]*** *соответствует только один элемент проекции* ***R[B]*** *, входящий вместе с ним в какой-либо кортеж отношения* ***R****.*

Полная функциональная зависимость R.A -> R.B - если набор атрибутов ***B*** функционально зависит от ***A*** и не зависит от любого подмножества ***A***, т.е. для любого ***A1***, являющегося подмножеством ***A***, ***R.B*** функционально не зависит от ***R.A1***, в противном случае зависимость ***R.A -> R.B*** называется неполной.

Нормализация

Шаг 3 (Приведение к 3NF)

*Отношение находится **в 3NF** тогда и только тогда, когда оно находится во второй нормальной форме и не содержит транзитивных зависимостей.*

***Транзитивной** называется функциональная зависимость **R.A -> R.B**, если существует набор атрибутов **C** такой, что:*

1. ***C** не является подмножеством **A.***
2. ***C** не включает в себя **B.***
3. *Существует функциональная зависимость **R.A -> R.C.***
4. *Не существует функциональной зависимо**сти R.C -> R.A.***
5. *Существует функциональная зависимость **R.C -> R.B.***

Детерминант отношения - атрибут или набор атрибутов, от которых зависит другой атрибут, если в отношении существует несколько функциональных зависимостей.

Первичный ключ отношения - среди всех возможных ключей отношения обычно выбирают один, который считается главным.

Неключевой атрибут - любой атрибут отношения, не входящий в состав ни одного возможного ключа отношения.

Взаимно-независимые атрибуты - не зависят функционально один от другого.

Нормализация

Шаг 3 (Приведение к 3NF)

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

Функциональные зависимости:

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

$(A_1, .., A_n) \rightarrow (B_1, ..., B_m)$ - зависимость одних неключевых атрибутов от 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 - $(A_1, .., A_n)$

Структура исходного отношения

ФИО	**Номер зач. кн.**	Группа	Факультет	Специальность	Выпускающая кафедра

есть следующие функциональные зависимости, образующие транзитивные группы:

Группа -> Факультет
Группа -> Специальность
Группа -> Выпускавшая кафедра
Выпускавшая кафедра -> Факультет

Номер зач.кн. -> ФИО
Номер зач.кн. -> Группа
Номер зач.кн. -> Факультет
Номер зач.кн. -> Специальность
Номер зач.кн. -> Выпускающая кафедра

Приведение исходного отношения к 3NF

Номер зач.кн.	ФИО	Специальность	Группа

Группа	Выпускающая кафедра

Выпускающая кафедра	Факультет

Нормализация

Основные аксиомы Армстронга:

1. *Рефлективность:* если B является подмножеством A, то A->B.
2. *Дополнение:* если A->B, то AC->BC.
3. *Транзитивность:* если A->B и B->C, то A->C.

Замыканием называется множество всех возможных функциональных зависимостей, выводимое из заданного набора исходных функциональных зависимостей.

Нормализация

Шаг 4 (Приведение к BCNF)

*Отношение находится **в нормальной форме Бойса-Кодда**, если оно находится в третьей нормальной форме и каждый детерминант отношения является возможным ключом отношения.*

Структура исходного отношения

Номер зач. кн.	**Идентификатор студента**	**Дисциплина**	**Дата**	Оценка

Имеются функциональные зависимости:

Номер зач. кн. Дисциплина. Дата -> Оценка;
Идентификатор студента. Дисциплина. Дата -> Оценка;
Номер зач. кн. -> Идентификатор студента;
Идентификатор студента -> Номер зач.кн.

Приведение исходного отношения к форме Бойса-Кодда

Идентификатор студента	**Дисциплина**	**Дата**	Оценка

Номер зач. кн.	Идентификатор студента

или

Номер зач. кн.	**Дисциплина**	**Дата**	Оценка

Номер зач. кн.	Идентификатор студента

Нормализация

Шаг 5 (Приведение к 4NF)

Нормальные формы высших порядков

В отношении R(A, B, C) существует многозначная зависимость (multi valid dependence, MVD) R.A -» R.B в том и только и том случае, если, множество значений B, соответствующее паре значений A и C, зависит только от A и не зависит от C.

Структура исходного отношения с MVD

Номер зач._кн.	Группа	Дисциплина

Существуют две многозначные зависимости:

Группа -» Дисциплина

Группа -» Номер_зач.кн.

Теорема Фейджина

Отношение R(A, B, C) можно спроецировать без потерь в отношения R1 (A, B) и R2 (A, C) в том и только в том случае, когда существует MVD A-» B / C (что равнозначно наличию двух зависимостей A-» B и A-» C).

Нормализация

Шаг 5 (Приведение к 4NF)

*Отношение R находится **в 4NF** в том и только в том случае, если в случае существования многозначной зависимости A-» B все остальные атрибуты R функционально зависят от A.*

Структура исходного отношения с MVD

Номер зач.кн.	Группа	Дисциплина

Приведение исходного отношения к 4NF

Разбиение исходного отношения на проекции

Номер зач.кн.	Группа

Группа	Дисциплина

Нормализация

Шаг 6 (Приведение к 5NF)

*Отношение R находится **в пятой нормальной форме** (нормальной форме проекции соединения - PJ/NF) в том и только в том случае, когда любая зависимость соединения в R следует из существования некоторого возможного ключа в R.*

Отношение R(X,У, .., Z) удовлетворяет зависимости соединения (X, У,..., Z) в том и только в том случае, когда R восстанавливается без потерь путем соединения своих проекций на X, У, ..., Z. Здесь X, У, ..., Z - наборы атрибутов отношения R.

Нормализация

Шаг 6 (Приведение к 5NF)

Структура исходного отношения R1

Преподаватель	Кафедра	Дисциплина

Обозначим наборы атрибутов:
ПК (Преподаватель. Кафедра)
ПД (Преподаватель. Дисциплина)
КД (Кафедра. Дисциплина)

Допустим, R1 удовлетворяет проекции соединения (ПК, ПД, КД).
Тогда отношение R1 не находится в NF/PJ, т. к. его единственный ключ
- полный набор атрибутов

Приведение исходного отношения R1 к 5NF = форме PJ/NF

R2

Преподаватель	Кафедра

R3

Преподаватель	Дисциплина

R4

Кафедра	Дисциплина

Нормализация

Сравнение нормализованных и ненормализованных моделей

Критерий	Отношения слабо нормализованы (1НФ, 2НФ)	Отношения сильно нормализованы (3НФ)
Адекватность БД предметной области	ХУЖЕ (-)	ЛУЧШЕ (+)
Легкость разработки и сопровождения БД	СЛОЖНЕЕ (-)	ЛЕГЧЕ (+)
Скорость выполнения вставки, обновления, удаления	МЕДЛЕННЕЕ (-)	БЫСТРЕЕ (+)
Скорость выполнения выборки данных	БЫСТРЕЕ (+)	МЕДЛЕННЕЕ (-)

Спасибо за внимание!

РАЗРАБОТКА БАЗ ДАННЫХ

ФИО преподавателя: Богомольная Г.В.

e-mail: bogomolnaya@mirea.ru

ТЕМА
СТРУКТУРА SQL

План лекции

- Основные объекты структуры базы данных SQL-сервера.
- Синтаксис оператора создания таблиц.
- Синтаксис операторов обновления и удаления таблиц.
- Создание индекса.

Структура SQL

Основные объекты структуры базы данных SQL-сервера

Объекты	Смысл
Tables	Таблицы базы данных, в которых хранятся собственно данные
Views	Представления (виртуальные таблицы) для отображения данных из таблиц
Stored Procedures	Хранимые процедуры
Triggers	Триггеры – специальные хранимые процедуры, вызываемые при изменении данных в таблице
User Defined function	Создаваемые пользователем функции
Indexes	Индексы – дополнительные структуры, призванные повысить производительность работы с данными
User Defined Data Types	Определяемые пользователем типы данных
Keys	Ключи – один из видов ограничений целостности данных
Constraints	Ограничение целостности – объекты для обеспечения логической целостности данных
Users	Пользователи, обладающие доступом к базе данных
Roles	Роли, позволяющие объединять пользователей в группы
Rules	Правила базы данных, позволяющие контролировать логическую целостность данных
Defaults	Умолчания или стандартные установки базы данных

Структура SQL

Операторы определения данных DDL (Data Definition Language)

Оператор	Смысл	Действие
CREATE TABLE	Создать таблицу	Создает новую таблицу в БД
DROP TABLE	Удалить таблицу	Удаляет таблицу из БД
ALTER TABLE	Изменить таблицу	Изменяет структуру существующей таблицы или ограничения целостности, задаваемые для данной таблицы
CREATE VIEW	Создать представление	Создает виртуальную таблицу, соответствующую некоторому SQL-запросу
ALTER VIEW	Изменить представление	Изменяет ранее созданное представление
DROP VIEW	Удалить представление	Удаляет ранее созданное представление
CREATE INDEX	Создать индекс	Создает индекс для таблицы для обеспечения быстрого доступа по атрибутам, входящим в индекс
DROP INDEX	Удалить индекс	Удаляет ранее созданный индекс

Операторы определения данных

Создание базы данных в среде MS SQL Server

<определение_базы_данных> ::=
CREATE DATABASE имя_базы_данных
[ON [PRIMARY]
[<определение_файла> [,...n]]
[,<определение_группы> [,...n]]]
[LOG ON {<определение_файла>[,...n] }]
[FOR LOAD | FOR ATTACH]

Операторы определения данных

Создание таблиц

Базовый упрощенный синтаксис оператора создания таблицы ***CREATE TABLE***

<определение_таблицы> ::=
  CREATE TABLE имя_таблицы
  (имя_столбца тип_данных
    [NULL | NOT NULL] [,...n])

Пример оператора создания таблицы:

```
CREATE TABLE s1
(ФИО VARCHAR (20) NOT NULL,
Дисциплина VARCHAR (20) NOT NULL,
Оценка SMALLINT NOT NULL);
```

Операторы определения данных

Создание таблиц

Базовое полное определение оператора ***CREATE TABLE***

CREATE TABLE имя_таблицы
({ имя_столбца тип_даных [NOT NULL] [UNIQUE]
[DEFAULT значение по умолчанию]
[CHECK (условие проверки на допустимость) [,...]}
[PRIMARY KEY (список столбцов),]
{[UNIQUE (список столбцов),] [,...]}
{[FOREIGN KEY {список столбцов внешних ключей)
REFERENCES имя родительской таблицы [(список столбцов ключей-кандидатов)]}
[ON UPDATE правило ссылочной целостности]
[ON DELETE правило ссылочной целостности]] [,...]}
{[CHECK (условие проверки на допустимость)] [,...]})

Пример оператора создания таблицы:

CREATE TABLE s1 (ФИО VARCHAR (20) NOT NULL, Дисциплина VARCHAR (20) NOT NULL, Оценка SMALLINT NOT NULL);
PRIMARY KEY (ФИО, Дисциплина),
FOREIGN KEY ФИО REFERENCES S2
ON UPDATE CASCADE
ON DELETE CASCADE);

Правила ссылочной целостности

Правило целостности внешних ключей:

- для каждого значения внешнего ключа должно существовать соответствующее значение первичного ключа в родительском отношении.

Ссылочная целостность может быть нарушена при выполнении операций:

1) обновление кортежа в родительском отношении;
2) удаление кортежа в родительском отношении;
3) вставка кортежа в дочернее отношение;
4) обновление кортежа в дочернем отношении.

Стратегии поддержания ссылочной целостности

Основные стратегии поддержания ссылочной целостности:

1. RESTRICT – не разрешать выполнение операции, приводящей к нарушению ссылочной целостности.
2. CASCADE – разрешить выполнение требуемой операции, но внести при этом необходимые поправки в других кортежах отношений так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи.

Дополнительные стратегии поддержания ссылочной целостности:

1. NONE – никаких операций по поддержке ссылочной целостности не выполняется.
2. SET NULL – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей заменять на неопределенные значения (null-значения).
3. SET DEFAULT – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на некоторое значение, принятое по умолчанию.

Операторы определения данных

Обновление таблиц

*Обобщенный формат оператора **ALTER TABLE***

ALTER TABLE имя_таблицы
[ADD [COLUMN] имя столбца тип данных [NOT NULL] [UNIQUE]
[DEFAULT значение по умолчанию] [CHECK (условие проверки на допустимость)]]
[DROP [COLUMN]] имя_столбца [RISTRICT | CASCADE]]
[ADD [CONSTRAINT [(имя ограничения)] ограничение]
[DROP CONSTRAINT имя ограничения [RISTRICT I CASCADE]]
[ALTER [COLUMN] SET DEFAULT значение по умолчанию]
[ALTER (COLUMN] DROP DEFAULT]

Пример оператора обновления таблицы:

ALTER TABLE s1
ADD Группа varchar (7) NOT NULL;

Операторы определения данных

Обновление таблиц

Модификация структуры таблицы

```
ALTER TABLE имя_таблицы
  {[ADD [COLUMN] имя_столбца тип_данных [NULL | NOT NULL]]
  | [DROP [COLUMN] имя_столбца]}
```

Модификация таблицы

```
ALTER TABLE имя_таблицы
  {[ALTER COLUMN имя_столбца
  {новый_тип_данных [(точность[,масштаб])] [ NULL | NOT NULL ]}]
  | ADD { [имя_столбца тип_данных]
  | имя_столбца AS выражение } [,...n]
  | DROP {COLUMN имя_столбца}[,...n]
  }
```

Операторы определения данных

Удаление таблиц

DROP TABLE имя_таблицы [RISTRICT I CASCADE]

Пример оператора удаления таблицы:

DROP TABLE s1;

Операторы создания и удаления индексов

Создать индекс:

CREATE [UNIQUE] INDEX имя_индекса
ON имя_таблицы (столбец [ASC| DESC] [,_.])

Удалить индекс:

DROP INDEX имя_индекса

Способы определения индекса

- автоматическое создание индекса при создании первичного ключа;
- автоматическое создание индекса при определении ограничения целостности UNIQUE ;
- создание индекса с помощью команды CREATE INDEX.

Индексы и методы доступа

Индексы – это механизмы быстрого доступа к данным в таблицах БД.

Физическая структура таблицы

Порядковый № записи	Дата прихода товара	Наименование товара	Количество
1	10.01.2020	Сахар	10
2	12.01.2020	Картофель	50
3	12.01.2020	Свекла	20
4	14.01.2020	Сахар	50
5	14.01.2020	Свекла	10
6	16.01.2020	Сливы	4

Логическая структура индексов

По дате прихода товара		По наименованию товара		По количеству	
Дата прихода	№ записи	Товар	№ записи	Количество	№ записи
10.01.2020	1	Картофель	2	4	6
12.01.2020	2	Сахар	1	10	1
12.01.2020	4	Сахар	4	10	5
14.01.2020	3	Свекла	3	20	3
14.01.2020	5	Свекла	5	50	2
16,01.2020	6	Сливы	6	50	4

Индексы и методы доступа

Последовательный метод доступа к данным в таблицах БД:

- просматриваются все записи таблицы, от первой к последней.

Индексно-последовательный метод доступа к данным в таблицах БД:

- поиск ведется по индексу, а не по самой таблице;
- поиск в индексе начинается только с первой строки, удовлетворяющей, условию запроса или его части («прямой доступ»);
- строки в индексе, начиная с такой записи, просматриваются последовательно.

Спасибо за внимание!

РАЗРАБОТКА БАЗ ДАННЫХ

ФИО преподавателя: Богомольная Г.В.

e-mail: bogomolnaya@mirea.ru

ТЕМА
СТРУКТУРА SQL

План лекции

- Создание индекса.
- Язык запросов DQL.

Операторы определения данных

Операторы создания и удаления индексов

Создать индекс:

CREATE [UNIQUE] INDEX имя_индекса
ON имя_таблицы (столбец [ASC| DESC] [,_.])

Удалить индекс:

DROP INDEX имя_индекса

Способы определения индекса

- автоматическое создание индекса при создании первичного ключа;
- автоматическое создание индекса при определении ограничения целостности UNIQUE ;
- создание индекса с помощью команды CREATE INDEX.

Индексы и методы доступа

Индексы – это механизмы быстрого доступа к данным в таблицах БД.

Физическая структура таблицы

Порядковый № записи	Дата прихода товара	Наименование товара	Количество
1	10.01.2020	Сахар	10
2	12.01.2020	Картофель	50
3	12.01.2020	Свекла	20
4	14.01.2020	Сахар	50
5	14.01.2020	Свекла	10
6	16.01.2020	Сливы	4

Логическая структура индексов

По дате прихода товара		По наименованию товара		По количеству	
Дата прихода	№ записи	Товар	№ записи	Количество	№ записи
10.01.2020	1	Картофель	2	4	6
12.01.2020	2	Сахар	1	10	1
12.01.2020	4	Сахар	4	10	5
14.01.2020	3	Свекла	3	20	3
14.01.2020	5	Свекла	5	50	2
16,01.2020	6	Сливы	6	50	4

Индексы и методы доступа

Последовательный метод доступа к данным в таблицах БД:

- просматриваются все записи таблицы, от первой к последней.

Индексно-последовательный метод доступа к данным в таблицах БД:

- поиск ведется по индексу, а не по самой таблице;
- поиск в индексе начинается только с первой строки, удовлетворяющей, условию запроса или его части («прямой доступ»);
- строки в индексе, начиная с такой записи, просматриваются последовательно.

Язык запросов DQL (Data Query Language)

Синтаксис оператора SELECT:

SELECT [ALL | DISTINCT] {*|[имя_столбца]} [,...n]
FROM имя_таблицы [,...n]
[WHERE <условие_поиска_предикат-условие_выборки_или_соединения>]
[GROUP BY имя_столбца [,...n]]
[HAVING <критерии выбора групп>]
[ORDER BY имя_столбца [,...n]]

Пример простого запроса

```
SELECT Name_kaf, Nom_telef
FROM kafedra;
```

Результирующая таблица

Name_kaf	Nom_telef
Физики	99-77
Прикладной математики	23-43

Язык запросов DQL (Data Query Language)

Фраза WHERE

Основные типы условий поиска (предикатов):

- сравнения " =, <>,>, <,> =, <=" - для сравнения результатов вычисления двух выражений; более сложные выражения строятся с помощью логических операторов AND, OR, NOT;
- BETWEEN A AND B - предикат истинен, когда вычисленное значение выражения попадает в заданный диапазон;
- IN - предикат истинен тогда, когда сравниваемое значение входит в множество заданных значений;
- LIKE и NOT LIKE - предикаты, смысл которых противоположен, требуют задания шаблона, с которым сравнивается заданное значение;
- IS NULL - предикат, применяющийся для выявления равенства значения некоторого атрибута неопределенному значению:

Язык запросов DQL (Data Query Language)

Пример запроса с предикатом сравнения

```
SELECT *
FROM kafedra
WHERE Name_kaf = 'Физики';
```

Результат запроса

Kod_kaf	Name_kaf	Nom_telef	Nom_Auditoria	Col_sotr	Zav_kaf
004	Физики	99-77	385	18	Петров И.С.

Пример запроса с предикатом диапазона

```
SELECT *
FROM kafedra
WHERE Nom_Auditoria BETWEEN 1 AND 99;
```

Результат запроса

Kod_kaf	Name_kaf	Nom_telef	Nom_Auditoria	Col_sotr	Zav_kaf
008	Математики	65-43	003	15	Иванов И.И.
004	Физики	99-77	085	18	Петров И.С.

Язык запросов DQL (Data Query Language)

Примеры запросов с предикатом принадлежности множеству

```
SELECT Фамилия, Город
  FROM Клиент
  WHERE Город IN ("Москва", "Самара");
```

ИЛИ

```
SELECT Фамилия, Город
  FROM Клиент
  WHERE Город NOT IN ("Москва", "Самара");
```

Соответствие шаблону

% – вместо этого символа может быть подставлено любое количество произвольных символов.

_ – заменяет один символ строки.

[] – вместо символа строки будет подставлен один из возможных символов, указанный в этих ограничителях.

[^] – вместо соответствующего символа строки будут подставлены все символы, кроме указанных в ограничителях.

Язык запросов DQL (Data Query Language)

Примеры запросов с предикатом соответствия шаблону

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон LIKE '_4%‘;
```

или

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон LIKE '_[2,4]%‘;
```

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон LIKE '_[2-4]%‘;
```

или

```
SELECT Клиент.Фамилия
   FROM Клиент
   WHERE Клиент.Фамилия LIKE "%ро%«;
```

Примеры запросов с предикатом неопределенного значения

```
SELECT Фамилия, Телефон
   FROM Клиент
   WHERE Телефон IS NULL;
```

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон Is Not Null;
```

Язык запросов DQL (Data Query Language)

Фраза ORDER BY

Пример запроса

```
SELECT *
FROM kafedra
ORDER BY Name_kaf ASC;
```

Результат запроса

Kod_kaf	Name_kaf	Nom_telef	Nom_Auditoria	Col_sotr	Zav_kaf
001	Графики	23-33	385	18	Орлов В.М.
003	Истории	78-72	465	16	Серов О.И.
008	Математики	65-43	003	15	Иванов И.И
004	Физики	99-77	085	18	Петров И.С.

Спасибо за внимание!

РАЗРАБОТКА БАЗ ДАННЫХ

ФИО преподавателя: Богомольная Г.В.

e-mail: bogomolnaya@mirea.ru

ТЕМА
СТРУКТУРА SQL

План лекции

- Язык запросов DQL.
- Теоретико-множественные и специальные операции над отношениями.

Язык запросов DQL (Data Query Language)

Синтаксис оператора SELECT:

SELECT [ALL | DISTINCT] {*|[имя_столбца]} [,...n]
FROM имя_таблицы [,...n]
[WHERE <условие_поиска_предикат-условие_выборки_или_соединения>]
[GROUP BY имя_столбца [,...n]]
[HAVING <критерии выбора групп>]
[ORDER BY имя_столбца [,...n]]

Пример простого запроса

```
SELECT Name_kaf, Nom_telef
FROM kafedra;
```

Результирующая таблица

Name_kaf	Nom_telef
Физики	99-77
Прикладной математики	23-43

Язык запросов DQL (Data Query Language)

Фраза WHERE

Основные типы условий поиска (предикатов):

- сравнения " =, <>,>, <,> =, <=" - для сравнения результатов вычисления двух выражений; более сложные выражения строятся с помощью логических операторов AND, OR, NOT;
- BETWEEN A AND B - предикат истинен, когда вычисленное значение выражения попадает в заданный диапазон;
- IN - предикат истинен тогда, когда сравниваемое значение входит в множество заданных значений;
- LIKE и NOT LIKE - предикаты, смысл которых противоположен, требуют задания шаблона, с которым сравнивается заданное значение;
- IS NULL - предикат, применяющийся для выявления равенства значения некоторого атрибута неопределенному значению:

Язык запросов DQL (Data Query Language)

Пример запроса с предикатом сравнения

```
SELECT *
FROM kafedra
WHERE Name_kaf = 'Физики';
```

Результат запроса

Kod_kaf	Name_kaf	Nom_telef	Nom_Auditoria	Col_sotr	Zav_kaf
004	Физики	99-77	385	18	Петров И.С.

Пример запроса с предикатом диапазона

```
SELECT *
FROM kafedra
WHERE Nom_Auditoria BETWEEN 1 AND 99;
```

Результат запроса

Kod_kaf	Name_kaf	Nom_telef	Nom_Auditoria	Col_sotr	Zav_kaf
008	Математики	65-43	003	15	Иванов И.И.
004	Физики	99-77	085	18	Петров И.С.

Язык запросов DQL (Data Query Language)

Примеры запросов с предикатом принадлежности множеству

```
SELECT Фамилия, Город
  FROM Клиент
  WHERE Город IN ("Москва", "Самара");
```

ИЛИ

```
SELECT Фамилия, Город
  FROM Клиент
  WHERE Город NOT IN ("Москва", "Самара");
```

Соответствие шаблону

% – вместо этого символа может быть подставлено любое количество произвольных символов.

_ – заменяет один символ строки.

[] – вместо символа строки будет подставлен один из возможных символов, указанный в этих ограничителях.

[^] – вместо соответствующего символа строки будут подставлены все символы, кроме указанных в ограничителях.

Язык запросов DQL (Data Query Language)

Примеры запросов с предикатом соответствия шаблону

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон LIKE '_4%‘;
```

или

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон LIKE '_[2,4]%‘;
```

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон LIKE '_[2-4]%‘;
```

или

```
SELECT Клиент.Фамилия
   FROM Клиент
   WHERE Клиент.Фамилия LIKE “%ро%”;
```

Примеры запросов с предикатом неопределенного значения

```
SELECT Фамилия, Телефон
   FROM Клиент
   WHERE Телефон IS NULL;
```

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон Is Not Null;
```

Язык запросов DQL (Data Query Language)

Фраза ORDER BY

Пример запроса

```
SELECT *
FROM kafedra
ORDER BY Name_kaf ASC;
```

Результат запроса

Kod_kaf	Name_kaf	Nom_telef	Nom_Auditoria	Col_sotr	Zav_kaf
001	Графики	23-33	385	18	Орлов В.М.
003	Истории	78-72	465	16	Серов О.И.
008	Математики	65-43	003	15	Иванов И.И
004	Физики	99-77	085	18	Петров И.С.

Язык запросов DQL (Data Query Language)

Агрегатные функции языка

Функция	Результат
COUNT	Количество строк или непустых значений полей, которые выбрал запрос
SUM	Сумма всех выбранных значений данного поля
AVG	Среднеарифметическое значение всех выбранных значений данного поля
M1N	Наименьшее из всех выбранных значений данного поля
MAX	Наибольшее из всех выбранных значений данного поля

Пример запроса

```
SELECT COUNT (*) AS count
FROM kafedra;
```

Результат запроса

count
4

Пример запроса

```
SELECT AVG(Col_sotr) AS avg
FROM kafedra;
```

Результат запроса

avg
17

Язык запросов DQL (Data Query Language)

Группирование результатов

```
SELECT ФИО, COUNT (Начисления) AS count, SUM (Начисления) AS sum
FROM r
GROUP BY ФИО
ORDER BY ФИО;
```

r

ФИО	Этап	Начисления (руб)
Семенов Т.Т.	Этап 1	1000
Просов С.М.	Этап 1	2000
Мехова И.И.	Этап 1	500
Семенов Т.Т.	Этап 2	500
Просов С.М.	Этап 2	500
Мехова И.И.	Этап 2	1000
Просов С.М.	Этап 3	1000
Мехова И.И.	Этап 3	1000
Чемцов Я.Ю.	Этап 3	2000
Чемцов Я.Ю.	Этап 4	2000
Яров И.М.	Этап 4	3000

Результат запроса

ФИО	count	sum
Мехова И.И.	3	2500
Просов С.М.	3	3500
Семенова Т.Т.	2	1500
Чемцов Я.Ю.	2	4000
Яров И.М.	1	3000

Язык запросов DQL (Data Query Language)

Группирование результатов

s

ФИО	Дисциплина	Оценка
Муров С.М.	Физика	4
Цуканов Т.Т.	Физика	5
Думская М.Т.	Физика	3
Дрозд Г.Р.	Физика	4
Муров С.М.	История	4
Цуканов Т.Т.	История	5
Думская М.Т.	История	3
Цуканов Т.Т.	Математика	5
Думская М.Т.	Математика	4
Дрозд Г.Р.	Математика	5
Петрова С.О.	Электротехника	5
Часов И.И.	Электротехника	4
Иванова Я.С.	Электротехника	5
Крисс Р.О.	Электротехника	3
Часов И.И.	Иностр. язык	5
Иванова Я.С.	Иностр. язык	4
Часов И.И.	Экономика	4
Иванова Я.С.	Экономика	4
Крисс Р.О.	Экономика	5
Фирсова Л.Р.	Экономика	3

```
SELECT Дисциплина, COUNT (*) AS count
FROM s
GROUP BY Дисциплина
ORDER BY Дисциплина;
```

Результат запроса

Дисциплина	count
Иностр. язык	2
История	3
Математика	3
Физика	4
Экономика	4
Электротехника	4

Язык запросов DQL (Data Query Language)

Группирование результатов

Пример запроса с предикатом

```
SELECT ФИО, COUNT (Начисления) AS count, SUM (Начисления) AS sum
FROM r
GROUP BY ФИО
HAVING COUNT (Начисления) > 1
ORDER BY ФИО;
```

r

ФИО	**Этап**	**Начисления** (руб)
Семенов Т.Т.	Этап 1	1000
Просов С.М.	Этап 1	2000
Мехова И.И.	Этап 1	500
Семенов Т.Т.	Этап 2	500
Просов С.М.	Этап 2	500
Мехова И.И.	Этап 2	1000
Просов С.М.	Этап 3	1000
Мехова И.И.	Этап 3	1000
Чемцов Я.Ю.	Этап 3	2000
Чемцов Я.Ю.	Этап 4	2000
Яров И.М.	Этап 4	3000

Результат запроса

ФИО	count	sum
Мехова И.И.	3	2500
Просов С.М.	3	3500
Семенов Т.Т.	2	1500
Чемцов Я.Ю.	2	4000

Язык запросов DQL (Data Query Language)

Вложенные запросы

Пример запроса

```
SELECT ФИО, Этап, Начисления
FROM r
WHERE Начисления> (SELECT AVG(Начисления) FROM r);
```

Результат запроса

ФИО	Этап	Начисления (руб)
Просов С. М.	Этап 1	2000
Чемцов Я.Ю.	Этап 3	2000
Чемцов Я.Ю.	Этап 4	2000
Яров И.М.	Этап 4	3200

Язык запросов DQL (Data Query Language)

Многотабличные запросы

Примеры простых запросов

```
SELECT *
FROM rl, r2;
```

```
SELECT rl.A, r2.B
FROM rl, r2;
```

Пример многотабличного запроса с предикатом

r1

ФИО	Отдел
Семенов Т.Т.	03
Просов С.М.	03
Мехова И.И.	03
Чемцов Я.Ю.	04
Яров И.М.	04

r2

Отдел	Этап
03	Этап 1
03	Этап 2
03	Этап 3
04	Этап 3
04	Этап 4

r3

ФИО	Этап	Начисления
Семенов Т.Т.	Этап 1	1000
Просов С.М.	Этап 1	2000
Мехова И.И.	Этап 1	500
Семенов Т.Т.	Этап 2	500
Просов С.М.	Этап 2	500
Мехова И.И.	Этап 2	1000
Просов С.М.	Этап 3	1000
Мехова И.И.	Этап 3	1000
Чемцов Я.Ю.	Этап 3	2000
Чемцов Я.Ю.	Этап 4	2000
Яров И.М.	Этап 4	3000

```
SELECT r3.ФИО, r3.Этап
FROM r1, r3
WHERE r1.Отдел = '03' AND
r1.ФИО = r3.ФИО AND
r3.Этап = 'Этап_3';
```

Результат запроса

ФИО	Этап
ПросовС.М.	Этап_3
Мехова И.И.	Этап_3

Пример многотабличных запросов с предикатом

s1

ФИО	Дисциплина	Оценка
Муров С.М.	Физика	4
Цуканов Т.Т.	Физика	5
Думская М.Т.	Физика	3
Дрозд Г.Р.	Физика	4
Муров С.М.	История	4
Цуканов Т.Т.	История	5
Думская М.Т.	История	3
Цуканов Т.Т.	Математика	5
Думская М.Т.	Математика	4
Дрозд Г.Р.	Математика	5
Петрова С.О.	Электротехника	5
Часов И.И.	Электротехника	4
Иванова Я.С.	Электротехника	5
Крисс Р.О.	Электротехника	3
Часов И.И.	Иностр. язык	5
Иванова Я.С.	Иностр. язык	4
Часов И.И.	Экономика	4
Иванова Я.С.	Экономика	4
Крисс Р.О.	Экономика	5
Фирсова Л.Р.	Экономика	3

s2

ФИО	Группа
Мур С.М.	02-КТ-21
Цуканов Т.Т.	02-КТ-21
Думская М.Т.	02-КТ-21
Дрозд Г.Р.	02-КТ-21
Петров С.О.	02-КТ-12
Часв И.И.	02-КТ-12
Иванова Я.С.	02-КТ-12
Крисс Р.О.	02-КТ-12
Фирсова Л.Р.	02-КТ-12

s3

Группа	Дисциплина
02-КТ-21	Физика
02-КТ-21	История
02-КТ-21	Математика
02-КТ-12	Экономика
02-КТ-12	Электротехника
02-КТ-12	Иностр. язык

```
SELECT s2.Группа
FROM s1, s2
WHERE s1.ФИО = s2.ФИО AND
s1.Оценка = 5
GROUP BY s2.Группа, s1.Дисциплина
HAVING count (*)> 1;
```

Результат запроса

Группа
02-КТ-21
02-КТ-12

```
SELECT ФИО
FROM s2,s3
WHERE s2.Группа=s3.Группа AND
Дисциплина = 'История' AND NOT EXISTS (SELECT ФИО
FROM s1
WHERE ФИО = s1.ФИО AND
Дисциплина = 'История');
```

Результат запроса

ФИО
Дрозд Г. Р.

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

```
CREATE TABLE R
(a1 CHAR(1), a2 INT, PRIMARY KEY(a1,a2))
```

R

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

```
CREATE TABLE S
(b1 INT PRIMARY KEY, b2 CHAR(1))
```

S

b1	b2
1	h
2	g
3	h

Операция выборки

```
SELECT a1, a2
FROM R
WHERE a2=1
```

Операция проекции

```
SELECT DISTINCT b2
FROM S
```

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

R

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

S

b1	b2
1	h
2	g
3	h

Декартово произведение

RxS={(a, 1, 1, h), (a, 2, 1, h), (b, 1, 1, h), ... }

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R, S
```

Результат декартова произведения двух отношений

R x S			
R.a1	R.a2	S.b1	S.b2
A	1	1	h
A	1	2	g
A	1	3	h
A	2	1	h
A	2	2	g
A	2	3	h
B	1	1	h
B	1	2	g
B	1	3	h
B	3	1	h
B	3	2	g
B	3	3	h
B	4	1	h
B	4	2	g
B	4	3	h

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция соединения по двум отношениям

Формат операции

```
FROM  имя_таблицы_1  {INNER  |  LEFT  |
RIGHT}
   JOIN имя_таблицы_2
ON условие_соединения
```

Типы операций соединения:

- тета-соединение;
- соединение по эквивалентности ;
- естественное соединение;
- внешнее соединение;
- полусоединение.

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция тета-соединения -

определяет отношение, которое содержит кортежи из декартова произведения отношений R и S, удовлетворяющие предикату F.

F {R.aiΘS.bj}, где **Θ** - один из операторов сравнения (>, >=, <, <=, =, <>).

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R, S
WHERE R.a2=S.b1
```

или

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R INNER JOIN S ON  R.a2=S.b1
```

R

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

S

b1	b2
1	h
2	g
3	h

$$R \triangleright\triangleleft_F S, F = (R.a2 = S.b1)$$

R.a1	R.a2	S.b1	S.b2
A	1	1	h
A	2	2	g
B	3	3	h
B	1	1	h

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция естественное соединение -

соединение по эквивалентности двух отношений R и S, выполненное по всем общим атрибутам, из результатов которого исключается по одному экземпляру каждого общего атрибута.

```
SELECT R.a1, R.a2, S.b2
FROM R, S
WHERE R.a2=S.b1
```

или

```
SELECT R.a1, S.b1, S.b2
FROM R INNER JOIN S ON  R.a2=S.b1
```

R

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

S

b1	b2
1	h
2	g
3	h

$$R \triangleright\triangleleft S, F = (R.a2 = S.b1)$$

R.a1	R.a2 или S.b1	S.b2
A	1	h
A	2	g
B	3	h
B	1	h

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Примеры операций естественного соединения

```
SELECT *
FROM Сделка, Товар
WHERE Сделка.КодТовара=Товар.КодТовара
```

эквивалентно

```
SELECT *
FROM Товар INNER JOIN Сделка
  ON Товар.КодТовара=Сделка.КодТовара
```

Вложенные соединения

```
SELECT   Товар.Название,   Сделка.Количество,
Сделка.   Дата, Клиент.Фирма
FROM Клиент INNER JOIN
  (Товар INNER JOIN Сделка
ON Товар.КодТовара=Сделка.КодТовара)
ON Клиент.КодКлиента=Сделка.КодКлиента
```

Использование псевдонимов таблиц

```
SELECT T.Название, С.Количество,
    С.Дата, К.Фирма
FROM Клиент AS К INNER JOIN
  (Товар AS T INNER JOIN
  Сделка AS С
ON T.КодТовара=С.КодТовара)
ON К.КодКлиента=С.КодКлиента;
```

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция левое внешнее соединение -

соединение, при котором кортежи отношения R, не имеющие совпадающих значений в общих столбцах отношения S, также включаются в результирующее отношение.

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R LEFT JOIN S ON R.a2=S.b1
```

R

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

S

b1	b2
1	h
2	g
3	h

$R \supset \triangleleft S$

R.a1	R.a2	S.b1	S.b2
A	1	1	h
A	2	2	g
B	1	1	h
B	3	3	h
B	4	null	nul

Операция правое внешнее соединение $R \triangleleft \subset S$ **-**

в результирующем отношении содержатся все кортежи правого отношения

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R RIGHT JOIN S ON R.a2=S.b1
```

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция полусоединение -

определяет отношение, содержащее те кортежи отношения R, которые входят в соединение отношений R и S.

```
SELECT R.a1, R.a2
FROM R, S
WHERE R.a2=S.b1
```

или

```
SELECT R.a1,   R.a2
FROM R INNER JOIN S ON R.a2=S.b1
```

$R \triangleleft_F S, F = (R.a2 = S.b1)$

R.a1	R.a2
A	1
A	2
B	3
B	1

Пример операции внешнего соединения

```
SELECT Товар.*, Сделка.*
FROM Товар LEFT JOIN Сделка
ON Товар.КодТовара=Сделка.КодТовара;
```

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция объединения

Объединением двух таблиц R и S является таблица, содержащая все строки, которые имеются в первой таблице R, во второй таблице S или в обеих таблицах сразу.

```
SELECT R.a1, R.a2
FROM R
UNION
SELECT S.b2, S.b1
FROM S
```

Операция пересечения

Пересечением двух таблиц R и S является таблица, содержащая все строки, присутствующие в обеих исходных таблицах одновременно.

```
SELECT R.a1, R.a2
FROM R,S
WHERE R.a1=S.b1 AND R.a2=S.b2
```

или

```
SELECT R.a1, R.a2
FROM R
WHERE R.a1 IN
   (SELECT S.b1 FROM S
    WHERE S.b1=R.a1) AND R.a2 IN
      (SELECT S.b2
       FROM S
       WHERE S.b2=R.a2)
```

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция разности

Разностью двух таблиц R и S является таблица, содержащая все строки, которые присутствуют в таблице R, но отсутствуют в таблице S.

```
SELECT R.a1, R.a2
FROM R
WHERE NOT EXISTS
   (SELECT S.b1,S.b2
    FROM S
    WHERE   S.b1=R.a2   AND
S.b2=R.a1)
```

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция деления отношений

Результат деления R:S - набор кортежей отношения R, определенных на множестве атрибутов C, соответствующих комбинации всех кортежей отношения S .

Отношение R определено на множестве атрибутов A, а отношение S - на множестве атрибутов B, причем A>=B и C=A - B.

T1=ПС(R);
T2=ПС((S X T1) -R);
T=T1 - T2.

Пример

A = {имя, пол, рост, возраст, вес}; B = {имя, пол, возраст}; C = {рост, вес}

R

имя	пол	рост	возраст	вес
a	ж	160	20	60
b	м	180	30	70
c	ж	150	16	40

S

имя	пол	возраст
a	ж	20

T1=ПС(R)

рост	вес
160	60
180	70
150	40

TT=(S X T1)-R

имя	пол	возраст	рост	вес
a	ж	20	180	70
a	ж	20	150	40

T2=ПС((S X T1)-R)

рост	вес
180	70
150	40

T=T1-T2

рост	вес
160	60

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция деления отношений

```
CREATE TABLE R
(i    int primary key, имя    varchar(3), пол    varchar(3), рост   int, возраст int, вес    int)

CREATE TABLE S
(i    int primary key, имя    varchar(3), пол    varchar(3), возраст int)
```

1. Создание отношения T1

```
CREATE VIEW T1
AS
SELECT рост,вес
FROM R
```

2. Создание отношения TT

```
CREATE VIEW TT AS
SELECT S.имя, S.пол, S.возраст, T1.рост, T1.вес
FROM S, T1
```

4. Создание отношения T

```
SELECT T1.рост, T1.вес
FROM T1
WHERE NOT EXISTS
   (SELECT T2.рост,T2.вес
    FROM T2
    WHERE T1.рост=T2.рост AND T1.вес=T2.вес)
```

3. Создание отношения T2

```
CREATE VIEW T2
AS
SELECT TT.рост, TT.вес
FROM TT
WHERE NOT EXISTS
   (SELECT R.рост, R.вес
    FROM R
    WHERE TT.имя=R.имя AND TT.пол=R.пол
        AND TT.возраст=R.возраст
        AND TT.рост=R.рост
                AND TT.вес=R.вес)
```

Спасибо за внимание!

РАЗРАБОТКА БАЗ ДАННЫХ

ФИО преподавателя: Богомольная Г.В.

e-mail: bogomolnaya@mirea.ru

ТЕМА
СТРУКТУРА SQL

План лекции

- Представления
- Хранимые процедуры
- Триггеры

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

CREATE TABLE R
(a1 CHAR(1), a2 INT, PRIMARY KEY(a1,a2))

R

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

CREATE TABLE S
(b1 INT PRIMARY KEY, b2 CHAR(1))

S

b1	b2
1	h
2	g
3	h

Операция выборки

SELECT a1, a2
FROM R
WHERE a2=1

Операция проекции

SELECT DISTINCT b2
FROM S

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

R

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

S

b1	b2
1	h
2	g
3	h

Декартово произведение

RxS={(a, 1, 1, h), (a, 2, 1, h), (b, 1, 1, h), ... }

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R, S
```

Результат декартова произведения двух отношений

R x S			
R.a1	R.a2	S.b1	S.b2
A	1	1	h
A	1	2	g
A	1	3	h
A	2	1	h
A	2	2	g
A	2	3	h
B	1	1	h
B	1	2	g
B	1	3	h
B	3	1	h
B	3	2	g
B	3	3	h
B	4	1	h
B	4	2	g
B	4	3	h

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция соединения по двум отношениям

Формат операции

```
FROM  имя_таблицы_1  {INNER  |  LEFT  |
RIGHT}
   JOIN имя_таблицы_2
ON условие_соединения
```

Типы операций соединения:

- тета-соединение;
- соединение по эквивалентности ;
- естественное соединение;
- внешнее соединение;
- полусоединение.

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция тета-соединения -

определяет отношение, которое содержит кортежи из декартова произведения отношений R и S, удовлетворяющие предикату F.

F {R.aiΘS.bj}, где **Θ** - один из операторов сравнения (>, >=, <, <=, =, <>).

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R, S
WHERE R.a2=S.b1
```

или

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R INNER JOIN S ON  R.a2=S.b1
```

R

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

S

b1	b2
1	h
2	g
3	h

$$R \triangleright\triangleleft_F S, F = (R.a2 = S.b1)$$

R.a1	R.a2	S.b1	S.b2
A	1	1	h
A	2	2	g
B	3	3	h
B	1	1	h

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция естественное соединение -

соединение по эквивалентности двух отношений R и S, выполненное по всем общим атрибутам, из результатов которого исключается по одному экземпляру каждого общего атрибута.

```
SELECT R.a1, R.a2, S.b2
FROM R, S
WHERE R.a2=S.b1
```

или

```
SELECT R.a1, S.b1, S.b2
FROM R INNER JOIN S ON  R.a2=S.b1
```

R

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

S

b1	b2
1	h
2	g
3	h

$$R \triangleright\triangleleft S, F = (R.a2 = S.b1)$$

R.a1	R.a2 или S.b1	S.b2
A	1	h
A	2	g
B	3	h
B	1	h

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Примеры операций естественного соединения

```
SELECT *
FROM Сделка, Товар
WHERE Сделка.КодТовара=Товар.КодТовара
```

эквивалентно

```
SELECT *
FROM Товар INNER JOIN Сделка
  ON Товар.КодТовара=Сделка.КодТовара
```

Вложенные соединения

```
SELECT   Товар.Название,   Сделка.Количество,
Сделка.   Дата, Клиент.Фирма
FROM Клиент INNER JOIN
  (Товар INNER JOIN Сделка
ON Товар.КодТовара=Сделка.КодТовара)
ON Клиент.КодКлиента=Сделка.КодКлиента
```

Использование псевдонимов таблиц

```
SELECT Т.Название, С.Количество,
    С.Дата, К.Фирма
FROM Клиент AS К INNER JOIN
  (Товар AS Т INNER JOIN
  Сделка AS С
ON Т.КодТовара=С.КодТовара)
ON К.КодКлиента=С.КодКлиента;
```

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция левое внешнее соединение -

соединение, при котором кортежи отношения R, не имеющие совпадающих значений в общих столбцах отношения S, также включаются в результирующее отношение.

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R LEFT JOIN S ON R.a2=S.b1
```

R

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

S

b1	b2
1	h
2	g
3	h

$R \supset \triangleleft S$

R.a1	R.a2	S.b1	S.b2
A	1	1	h
A	2	2	g
B	1	1	h
B	3	3	h
B	4	null	nul

Операция правое внешнее соединение $R \triangleleft \subset S$ ***-***

в результирующем отношении содержатся все кортежи правого отношения

```
SELECT R.a1, R.a2, S.b1, S.b2
FROM R RIGHT JOIN S ON R.a2=S.b1
```

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция полусоединение -

определяет отношение, содержащее те кортежи отношения R, которые входят в соединение отношений R и S.

```
SELECT R.a1, R.a2
FROM R, S
WHERE R.a2=S.b1
```

или

```
SELECT R.a1,   R.a2
FROM R INNER JOIN S ON R.a2=S.b1
```

$R \triangleleft_F S, F = (R.a2 = S.b1)$

R.a1	R.a2
A	1
A	2
B	3
B	1

Пример операции внешнего соединения

```
SELECT Товар.*, Сделка.*
FROM Товар LEFT JOIN Сделка
ON Товар.КодТовара=Сделка.КодТовара;
```

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция объединения

Объединением двух таблиц R и S является таблица, содержащая все строки, которые имеются в первой таблице R, во второй таблице S или в обеих таблицах сразу.

```
SELECT R.a1, R.a2
FROM R
UNION
SELECT S.b2, S.b1
FROM S
```

Операция пересечения

Пересечением двух таблиц R и S является таблица, содержащая все строки, присутствующие в обеих исходных таблицах одновременно.

```
SELECT R.a1, R.a2
FROM R,S
WHERE R.a1=S.b1 AND R.a2=S.b2
```

или

```
SELECT R.a1, R.a2
FROM R
WHERE R.a1 IN
   (SELECT S.b1 FROM S
    WHERE S.b1=R.a1) AND R.a2 IN
      (SELECT S.b2
       FROM S
       WHERE S.b2=R.a2)
```

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция разности

Разностью двух таблиц R и S является таблица, содержащая все строки, которые присутствуют в таблице R, но отсутствуют в таблице S.

```
SELECT R.a1, R.a2
FROM R
WHERE NOT EXISTS
   (SELECT S.b1,S.b2
    FROM S
    WHERE   S.b1=R.a2   AND
S.b2=R.a1)
```

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция деления отношений

Результат деления R:S - набор кортежей отношения R, определенных на множестве атрибутов C, соответствующих комбинации всех кортежей отношения S .

Отношение R определено на множестве атрибутов A, а отношение S - на множестве атрибутов B, причем A>=B и C=A - B.

T1=ПС(R);
T2=ПС((S X T1) -R);
T=T1 - T2.

Пример

A = {имя, пол, рост, возраст, вес}; B = {имя, пол, возраст}; C = {рост, вес}

R

имя	пол	рост	возраст	вес
a	ж	160	20	60
b	м	180	30	70
c	ж	150	16	40

S

имя	пол	возраст
a	ж	20

T1=ПС(R)

рост	вес
160	60
180	70
150	40

TT=(S X T1)-R

имя	пол	возраст	рост	вес
a	ж	20	180	70
a	ж	20	150	40

T2=ПС((S X T1)-R)

рост	вес
180	70
150	40

T=T1-T2

рост	вес
160	60

Язык запросов DQL (Data Query Language)

Теоретико-множественные и специальные операции над отношениями

Операция деления отношений

```
CREATE TABLE R
(i    int primary key, имя    varchar(3), пол    varchar(3), рост   int, возраст int, вес    int)

CREATE TABLE S
(i    int primary key, имя    varchar(3), пол    varchar(3), возраст int)
```

1. Создание отношения T1

```
CREATE VIEW T1
AS
SELECT рост,вес
FROM R
```

2. Создание отношения TT

```
CREATE VIEW TT AS
SELECT S.имя, S.пол, S.возраст, T1.рост, T1.вес
FROM S, T1
```

4. Создание отношения T

```
SELECT T1.рост, T1.вес
FROM T1
WHERE NOT EXISTS
   (SELECT T2.рост,T2.вес
    FROM T2
    WHERE T1.рост=T2.рост AND T1.вес=T2.вес)
```

3. Создание отношения T2

```
CREATE VIEW T2
AS
SELECT TT.рост, TT.вес
FROM TT
WHERE NOT EXISTS
   (SELECT R.рост, R.вес
    FROM R
    WHERE TT.имя=R.имя AND TT.пол=R.пол
        AND TT.возраст=R.возраст
        AND TT.рост=R.рост
                AND TT.вес=R.вес)
```

Язык запросов DQL (Data Query Language)

Представления

Представление - предопределенный запрос, хранящийся в базе данных, который выглядит как обычная таблица и не требует для своего хранения дисковой памяти. Для хранения представления используется только оперативная память.

```
<определение_представления> ::=
        { CREATE| ALTER} VIEW имя_представления
        [(имя_столбца [,...n])]
        [WITH ENCRYPTION]
 AS SELECT_оператор
        [WITH CHECK OPTION]
```

Параметр WITH ENCRYPTION предписывает серверу шифровать SQL-код запроса для гарантии его защиты от несанкционированного просмотра и использования.

Параметр WITH CHECK OPTION предписывает серверу исполнять проверку изменений, производимых через представление, на соответствие критериям, определенным в операторе SELECT.

Язык запросов DQL (Data Query Language)

Представления

Пример создания представления

```
CREATE VIEW view1 AS
SELECT КодКлиента, Фамилия, ГородКлиента
FROM Клиент
WHERE ГородКлиента='Москва'
```

Пример выборки данных из представления

```
SELECT * FROM view1
```

Пример ввода данных в представления

```
INSERT INTO view1 VALUES (12,'Петров', 'Самара')
```

Пример создания представления с проверкой команд модификации

```
ALTER VIEW view1 AS
SELECT КодКлиента, Фамилия, ГородКлиента
FROM Клиент
WHERE ГородКлиента='Москва'
WITH CHECK OPTION
```

Удаление представления

```
DROP VIEW имя_представления [,...n]
```

Язык запросов DQL (Data Query Language)

Представления

Обновление данных в представлениях

Модифицируемое представление определяется критериями:
- основывается только на одной базовой таблице;
- содержит первичный ключ этой таблицы;
- не содержит DISTINCT в своем определении;
- не использует GROUP BY или HAVING в своем определении;
- по возможности не применяет в своем определении подзапросы;
- не использует константы или выражения значений среди выбранных полей вывода;
- в просмотр включается каждый столбец таблицы, имеющий атрибут NOT NULL;
- оператор SELECT просмотра не использует агрегирующие (итоговые) функции, соединения таблиц, хранимые процедуры и функции, определенные пользователем;
- основывается на одиночном запросе, поэтому объединение UNION не разрешено.

Язык запросов DQL (Data Query Language)

Представления

Обновление данных в представлениях

Пример немодифицируемого представления с данными из разных таблиц

```
CREATE VIEW view2 AS
SELECT Клиент.Фамилия, Клиент.Фирма,
        Сделка.Количество
FROM Клиент INNER JOIN Сделка
ON Клиент.КодКлиента=Сделка.КодКлиента
```

Пример немодифицируемого представления с группировкой и итоговыми функциями

```
CREATE VIEW view3(Тип, Общ_остаток) AS
SELECT Тип, Sum(Остаток)
FROM Товар
GROUP BY Тип
```

Пример модифицируемого представления с вычислениями

```
CREATE VIEW view4(Код, Название, Тип, Цена, Налог) AS
SELECT КодТовара, Название, Тип, Цена, Цена*0.05
FROM Товар
```

Язык запросов DQL (Data Query Language)

Представления

Преимущества применения представлений:

- независимость от данных;
- актуальность;
- повышение защищенности данных;
- снижение стоимости;
- дополнительные удобства;
- возможность настройки;
- обеспечение целостности данных.

Недостатки использования представлений:

- ограниченные возможности обновления;
- структурные ограничения;
- снижение производительности.

Язык запросов DQL (Data Query Language)

Хранимые процедуры

Хранимые процедуры - набор команд, состоящий из одного или нескольких операторов SQL или функций и сохраняемый в базе данных в откомпилированном виде.

Преимущества выполнения в базе данных хранимых процедур :

- необходимые операторы уже содержатся в базе данных;
- операторы прошли этап синтаксического анализа и находятся в исполняемом формате; перед выполнением хранимой процедуры SQL Server генерирует для нее план исполнения, выполняет ее оптимизацию и компиляцию;
- поддерживают модульное программирование, так как позволяют разбивать большие задачи на самостоятельные, более мелкие и удобные в управлении части;
- могут вызывать другие хранимые процедуры и функции;
- могут быть вызваны из прикладных программ других типов;
- выполняются быстрее, чем последовательность операторов;
- проще использовать: они могут состоять из десятков и сотен команд, но для их запуска достаточно указать всего лишь имя нужной хранимой процедуры-позволяет уменьшить размер запроса от клиента на сервер, а значит, и нагрузку на сеть.

Язык запросов DQL (Data Query Language)

Хранимые процедуры

Типы хранимых процедур

- Системные хранимые процедуры предназначены для выполнения различных административных действий.
- Пользовательские хранимые процедуры реализуют те или иные действия.
- Временные хранимые процедуры существуют некоторое время, после чего автоматически уничтожаются сервером. Они делятся на:
  - Локальные временные хранимые процедуры могут быть вызваны только из того соединения, в котором созданы. При создании такой процедуры ей необходимо дать имя, начинающееся с одного символа #. Автоматически удаляются при отключении пользователя, перезапуске или остановке сервера.
  - Глобальные временные хранимые процедуры доступны для любых соединений сервера, на котором имеется такая же процедура. Для ее определения достаточно дать ей имя, начинающееся с символов ##. Удаляются при перезапуске или остановке сервера.

Язык запросов DQL (Data Query Language)

Хранимые процедуры

<определение_процедуры>::=
{CREATE | ALTER } [PROCEDURE] имя_процедуры
[;номер]
[{@имя_параметра тип_данных } [VARYING]
[=default][OUTPUT]][,...n]
[WITH { RECOMPILE | ENCRYPTION | RECOMPILE,
ENCRYPTION }]
[FOR REPLICATION]
AS
sql_оператор [...n]

Выполнение хранимой процедуры

[[EXECUTE] имя_процедуры [;номер]
[[@имя_параметра=]{значение |
@имя_переменной}
[OUTPUT]|[DEFAULT]][,...n]

Язык запросов DQL (Data Query Language)

Хранимые процедуры

Пример процедуры без параметров

```
CREATE PROC my_proc1
AS
SELECT Товар.Название,
      Товар.Цена*Сделка.Количество
       AS Стоимость, Клиент.Фамилия
  FROM Клиент INNER JOIN
  (Товар INNER JOIN Сделка
  ON Товар.КодТовара=Сделка.КодТовара)
  ON Клиент.КодКлиента=Сделка.КодКлиента
  WHERE Клиент.Фамилия=’Иванов’
```

Для обращения к процедуре

```
EXEC my_proc1 или my_proc1
```

Язык запросов DQL (Data Query Language)

Триггеры

Триггер – откомпилированная SQL-процедура, исполнение которой обусловлено наступлением определенных событий внутри реляционной базы данных.

С помощью триггеров достигаются цели:

- проверка корректности введенных данных и выполнение сложных ограничений целостности данных, которые трудно, если вообще возможно, поддерживать с помощью ограничений целостности, установленных для таблицы;
- выдача предупреждений, напоминающих о необходимости выполнения некоторых действий при обновлении таблицы, реализованном определенным образом;
- накопление аудиторской информации посредством фиксации сведений о внесенных изменениях и тех лицах, которые их выполнили;
- поддержка репликации.

Язык запросов DQL (Data Query Language)

Триггеры

Основной формат команды CREATE TRIGGER:

<Определение_триггера>::=
CREATE TRIGGER имя_триггера
BEFORE | AFTER <триггерное_событие>
ON <имя_таблицы>
[REFERENCING
<список_старых_или_новых_псевдонимов>]
[FOR EACH { ROW | STATEMENT}]
[WHEN(условие_триггера)]
<тело_триггера>

BEFORE - триггер запускается до выполнения связанных с ним событий.
AFTER - триггер запускается после выполнения связанных с ним событий.
<список_старых_или_новых_псевдонимов> - старая / новая строка (OLD / NEW) либо старая / новая таблица (OLD TABLE / NEW TABLE).
FOR EACH ROW - выполняемые триггером действия задаются для каждой строки.
FOR EACH STATEMENT - действия задаются только один раз для каждого события.

Язык запросов DQL (Data Query Language)

Оператор создания или изменения триггера:

```
<Определение_триггера>::=
{CREATE | ALTER} TRIGGER имя_триггера
ON {имя_таблицы | имя_представления }
[WITH ENCRYPTION ]
{
{ { FOR | AFTER | INSTEAD OF }
{ [ DELETE] [,] [ INSERT] [,] [ UPDATE] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS
  sql_оператор[...n]
} |
{ {FOR | AFTER | INSTEAD OF } { [INSERT] [,]  [UPDATE] }
[ WITH APPEND]
[ NOT FOR REPLICATION]
AS
{ IF UPDATE(имя_столбца)
[ {AND | OR} UPDATE(имя_столбца)] [...n]
|
IF (COLUMNS_UPDATES(){оператор_бит_обработки}
 бит_маска_изменения)
{оператор_бит_сравнения }бит_маска [...n]}
sql_оператор [...n]
}
}
```

Оператор удаления триггера:

```
DROP TRIGGER {имя_триггера} [,...n]
```

Пример триггера для реализации ограничений на значение

Команда вставки записи в таблицу Сделка

```
INSERT INTO Сделка
 VALUES (3,-299,'01/08/2002')
```

```
CREATE TRIGGER Триггер_ins
ON Сделка FOR INSERT
AS
IF @@ROWCOUNT=1
BEGIN
 IF NOT EXISTS(SELECT *
   FROM inserted
   WHERE -inserted.количество<=ALL(SELECT
    Склад.Остаток
  FROM Склад,Сделка
  WHERE Склад.КодТовара=
   Сделка.КодТовара))
  BEGIN
   ROLLBACK TRAN
  PRINT
       'Отмена поставки: товара на складе нет'
  END
END
```

Язык запросов DQL (Data Query Language)

Типы триггеров

По параметрам поведения :

- *AFTER-триггер* выполняется после успешного выполнения вызвавших его команд. Если команды по какой-то причине не могут быть успешно завершены, триггер не выполняется. Можно определить несколько AFTER-триггеров для каждой операции (INSERT, UPDATE, DELETE). Если для таблицы предусмотрено выполнение нескольких AFTER-триггеров, то с помощью системной хранимой процедуры sp_settriggerorder можно указать, какой из них будет выполняться первым, а какой последним. По умолчанию в SQL Server все триггеры являются AFTER-триггерами.
- *INSTEAD OF-триггер* вызывается вместо выполнения команд, может быть определен для таблицы и для представления. Для каждой операции INSERT, UPDATE, DELETE можно определить только один INSTEAD OF -триггер.

По типу команд:

- *INSERT TRIGGER* – запускаются при попытке вставки данных с помощью команды INSERT.
- *UPDATE TRIGGER* – запускаются при попытке изменения данных с помощью команды UPDATE.
- *DELETE TRIGGER* – запускаются при попытке удаления данных с помощью команды DELETE.

Внутри триггера не допускается выполнение операций:

- ***создание, изменение и удаление базы данных;***
- ***восстановление резервной копии базы данных или журнала транзакций.***

Пример триггера для операции удаления записи из таблицы

Команда удаления записи из таблицы Сделка

```
DELETE FROM Сделка WHERE КодСделки=4
```

```
CREATE TRIGGER Триггер_del
ON Сделка FOR DELETE
AS
IF @@ROWCOUNT=1 -- удалена одна запись
BEGIN
  DECLARE @y INT,@x INT
--определяется код и количество товара из
--удаленной из таблицы Склад записи
  SELECT @y=КодТовара, @x=Количество
  FROM deleted
--в таблице Склад корректируется количество
--товара
    UPDATE Склад
    SET Остаток=Остаток-@x
    WHERE КодТовара=@y
END
```

Пример триггера для операции изменения записи в таблице

Команда

```
INSERT INTO Сделка
  VALUES (3,1,200,'01/08/2002')
```

```
ALTER TRIGGER Триггер_ins
ON Сделка FOR INSERT
AS
DECLARE @x INT, @y INT
IF @@ROWCOUNT=1
BEGIN
IF NOT EXISTS(SELECT *
        FROM inserted
        WHERE -inserted.количество<  =ALL(SELECT Склад.Остаток
       FROM Склад,Сделка
       WHERE Склад.КодТовара=Сделка.КодТовара))
   BEGIN
     ROLLBACK TRAN
      PRINT 'откат товара нет '
    END
IF NOT EXISTS ( SELECT *
        FROM Склад C, inserted i
        WHERE C.КодТовара=i.КодТовара )
    INSERT INTO Склад (КодТовара,Остаток)
 ELSE
BEGIN
   SELECT @y=i.КодТовара, @x=i.Количество
   FROM Сделка C, inserted i
   WHERE C.КодТовара=i.КодТовара
UPDATE Склад
       SET Остаток=остаток+@x
       WHERE КодТовара=@y
   END
END
```

Спасибо за внимание!