

ГОУ ВПО Уральский государственный технический университет УГТУ-УПИ  
Кафедра Автоматизированные системы управления

## **Работа с CASE-средствами BPwin, ERwin**

**К.А. Аксенов**  
**Б.И. Клебанов**

## CASE-средство BPwin

### Запуск программы

После запуска программы на экране появиться диалоговое окно, в котором следует выбрать режим работы: либо создать новую модель (Create model), либо открыть существующую модель (Open model) см. рис.1.

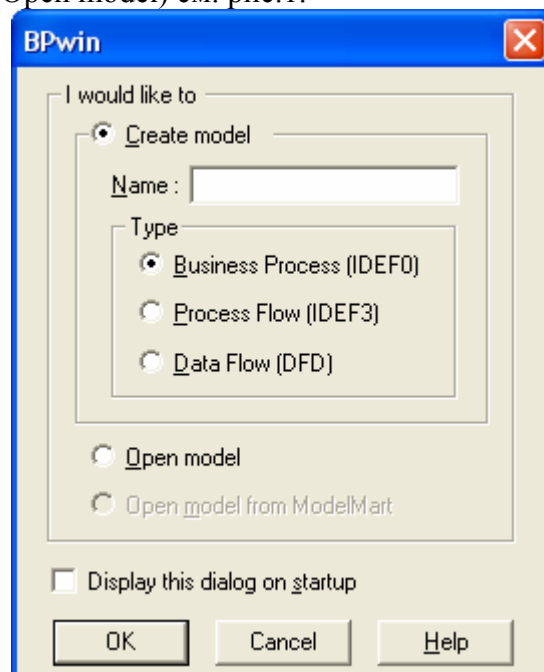


Рис.1.

При первом открытии программы (при создании новой модели) область построения содержит диаграмму IDEF-0:

### Основные инструменты

На основной панели инструментов расположены элементы управления, в основном знакомые по другим Windows-интерфейсам (рис.2.):

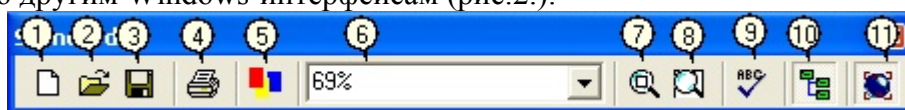


Рис.2.

1. Создать новую модель.
2. Открыть модель.
3. Сохранить модель.
4. Печать модели
5. Мастер создания отчетов.
6. Выбор масштаба.
7. Масштабирование.
8. Увеличение участка
9. Проверка ошибок
10. Включение и выключение навигатора модели

На основной панели инструментов (либо в любом желаемом месте экрана) расположены инструменты редактора BPWin:



Рис.3.

1. Pointer Tool – используется для выбора и определения позиции объектов добавленных в диаграмму.
2. Activity Box Tool – используется для установки блоков в диаграмме.
3. Arrow Tool – используется, чтобы устанавливать дуги в диаграмме.
4. Squiggle Tool – используется для создания тильды (squiggle, \_\_\_), которая соединяет дугу с ее названием.
5. Text Block Tool – используется для создания текстовых блоков.
6. Diagram Dictionary Editor – открывает диалоговое окно Diagram Dictionary Editor, где можно перейти на какую-либо диаграмму или создать новую диаграмму.
7. Go to Sibling Diagram – используется для отображения следующей диаграммы того же уровня.
8. Go to Parent Diagram – переход на родительскую диаграмму.
9. Go to Child Diagram – используется, чтобы отобразить диаграмму потомка или разложить выделенный блок на диаграмму потомка.

Любая диаграмма состоит из совокупности следующих объектов:

- Блоков;
- Дуг;
- Текстовых блоков.

Для работы с любым из этих объектов можно использовать либо основное меню (рис.4.):

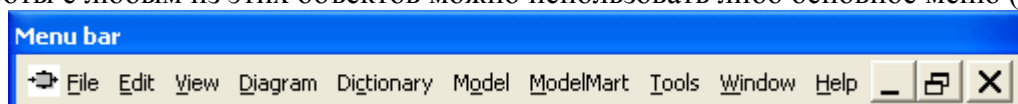


Рис.4.

либо контекстно-зависимое меню (меню, появляющееся при нажатии правой кнопки мыши). Принципы работы с меню являются стандартными для среды Windows. Объект сначала делается активным, затем над ним осуществляются необходимые действия.

### Описание полей бланка диаграммы

Каждая диаграмма располагается внутри бланка имеющего несколько информационных полей:

#### Поля верхней части рамки

**Used At** (Используется в) – используется для указания на родительский блок в случае, если на текущую диаграмму ссылались посредством стрелки вызова.

**Author** (Автор) – имя создателя диаграммы.

**Date** (Дата) – дата создания и имя проекта.

**Project** (Проект) – имя проекта.

**Rev** (Пересмотрено) – дата последнего редактирования диаграммы.

**Notes 12345678910** (Замечания) – используется при проведении сеанса экспертизы. Эксперт должен (на бумажной копии диаграммы) указать число замечаний, вычеркивая цифру из списка каждый раз при внесении нового замечания.

**Status (Статус)** – статус отображает стадию создания диаграммы, отображая все этапы публикации:

**Working** (Рабочая версия) – новая диаграмма, кардинально обновленная диаграмма или новый автор диаграммы;

**Draft** (Эскиз) – диаграмма прошла первичную экспертизу и готова к дальнейшему обсуждению;

**Recommended** (Рекомендовано) – диаграмма и все ее сопровождающие документы прошли экспертизу. Новых изменений не ожидается;

**Publication** (Публикация) – диаграмма готова к окончательной печати и публикации.

**Reader** (Читатель) – имя читателя (эксперта).

**Date** (Дата) – дата прочтения (экспертизы).

**Context** (Контекст) – схема расположения работ в диаграмме верхнего уровня. Работа, являющаяся родительской, показана темным прямоугольником, остальные – светлым. На контекстной диаграмме (A-0) показывается надпись TOP. В левом нижнем углу показывается номер по узлу родительской диаграммы:

### Поля нижней части рамки

**Node** (Узел) – номер узла диаграммы (номер родительского блока).

**Title** (Название) – имя диаграммы. По умолчанию – имя родительского блока.

**Number** (Номер) – С-номер, уникальный номер версии диаграммы.

**Page** (Страница) – номер страницы, может использоваться как номер страницы при формировании папки.

### Описание модели

IDEF0-модель предполагает наличие четко сформулированной цели, единственного субъекта моделирования и одной точки зрения. Для внесения области, цели и точки зрения в модели IDEF0 в BPwin следует выбрать пункт меню **Model/Model Properties**, вызывающий диалог **Model Properties** (Рис.5):

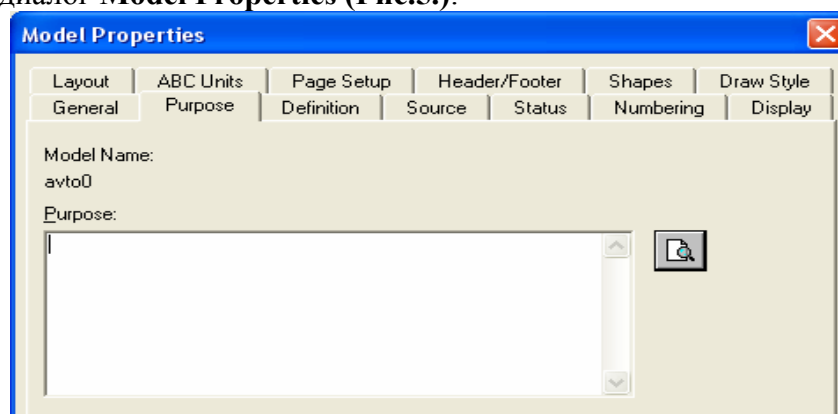


Рис.5.

В закладке **Purpose** следует внести цель и точку зрения, а в закладку **Definition** – определение модели и описание области. В закладке **Status** того же диалога можно описать статус модели (черновой вариант, рабочий, окончательный и т.д.), время создания и последнего редактирования (отслеживается в дальнейшем автоматически по системной дате). В закладке **Source** описываются источники информации для построения модели (например, "Опрос экспертов предметной области и анализ документации"). Закладка **General** служит для внесения имени проекта и модели, имени и инициалов автора и временных рамок, модели – AS-IS и TO-BE.

Результат описания модели можно получить в отчете Model Report. Диалог настройки отчета по модели вызывается из пункта меню **Tools/Reports/ModelReport**. В диалоге настройки следует выбрать необходимые поля (при этом автоматически отображается очередность вывода информации в отчет) рис.6.:

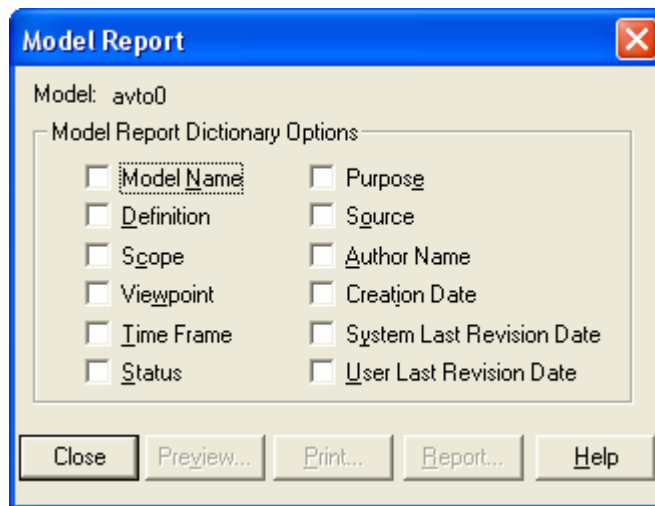


Рис.6.

Принцип работы в пакете BPwin рассмотрим на примере задачи обслуживания клиентов в банке для автомобилистов.

Задача: В банке для автомобилистов имеется 2 окошечка, каждое из которых обслуживается одним кассиром и имеет отдельную подъездную полосу. Обе полосы расположены рядом. Из предыдущих наблюдений известно, что интервалы времени между прибытием клиентов в час пик распределены экспоненциально с математическим ожиданием равным 0,5 единицы времени. Так как банк перегружен только в часы пик, то анализируется только этот период. Продолжительность обслуживания у обоих кассиров одинакова и распределена экспоненциально с математическим ожиданием, равным 0,3 единицы времени. Известно также, что при равной длине очереди, а так же при отсутствии очередей клиенты отдают предпочтение первой полосе. Во всех других случаях клиенты выбирают более короткую очередь. После того как клиент въехал в банк, он не может покинуть его, пока не будет обслужен. Однако он может сменить очередь, если стоит последним и разница в длине очередей при этом составляет не менее двух автомобилей. Из-за ограниченного места на каждой полосе может находиться не более трех автомобилей. В банке, таким образом, не может находиться более восьми автомобилей, включая автомобили двух клиентов, обслуживаемых в текущий момент кассиром. Если место перед банком заполнено до отказа, прибывший клиент считается потерянным, так как сразу уезжает.

Начальные условия имитации:

1. Оба кассира заняты. Продолжительность обслуживания для каждого кассира нормально распределена с математическим ожиданием, равным 1 единице времени, и среднеквадратическим отклонением, равным 0,3 единицы времени.
2. Прибытие первого клиента запланировано на момент времени 0,1.
3. В каждой очереди ожидают по два автомобиля.

Необходимо оценить следующие характеристики:

1. загрузку по каждому кассиру
2. число обслуженных клиентов
3. среднее время пребывания клиента в банке
4. среднее число клиентов в каждой очереди
5. процент клиентов, которым отказано в обслуживании
6. число смен подъездных полос

Имитация системы проводится в течении 1000 единиц времени.

### Работа с блоками и дугами

Методология IDEF0 предписывает построение иерархической системы диаграмм – единичных описаний фрагментов системы. Сначала проводится описание системы в

целом (контекстная диаграмма), после чего проводится декомпозиция – система разбивается на подсистемы, и каждая подсистема описывается отдельно.

### Контекстная диаграмма.

После создания проекта мы видим окно с единственным блоком. Назовем данный блок «Банк автомобилистов». Для этого необходимо щелкнуть правой клавишей мыши по блоку и выбрать команду **Name** и в диалоговом окне ввести название (рис.7).

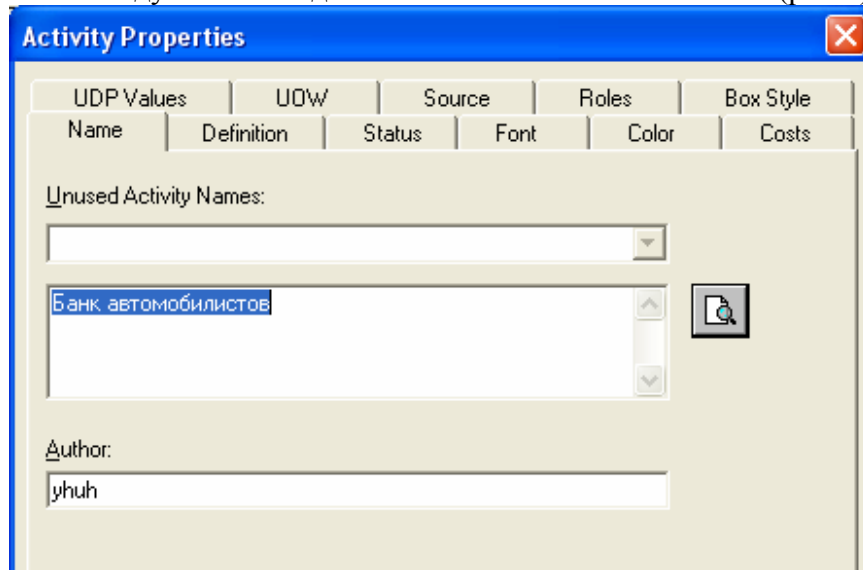



Рис.7.

По отношению к тексту применимы стандартные способы форматирования, для их использования следует выполнить команду контекстно-зависимого меню **Font**.

Вы можете изменять размеры блока

Для изменения высоты необходимо перетащить мышью верхнюю или нижнюю стороны блока, аналогично меняется размер по горизонтали.

После создания объекта «Банк автомобилистов» необходимо обозначить его основные функции и элементы взаимодействия. В Vrpwin этими элементами являются дуги

Для построения дуг управления, входа, выхода и механизмов необходимо выбрать инструмент  (Arrow Tool), затем щелкнуть мышью со стороны периметра и второй щелчок с соответствующей стороны блока. Для построения дуги выхода щелкнуть первоначально с правой стороны блока, затем со стороны периметра.

То с какой стороны дуга подходит к блоку является своего рода значением данной дуги.

Слева – вход в блок


Справа – выход в блок

Сверху – управляющая информация

Снизу – механизмы (средства производства)

Дугам, как и блокам можно придавать свои имена. Для этого необходимо: щелкнуть правой клавишей мыши по блоку и выбрать команду **Name** и в диалоговом окне ввести название дуги.

Определите наименования для созданных ранее дуг, соответственно типу дуги: «вход клиента», «выход клиента», «количество клиентов в очереди 1», «количество клиентов в очереди 2», «кассир 1», «кассир 2».

Название дуги является независимым объектом, который можно перемещать относительно дуги. Текст может располагаться по отношению к дуге в свободной форме, либо соединен с дугой символом тильды. Чтобы установить тильду следует нажать инструмент  (Squiggle Tool), а затем выбрать дугу, либо использовать команду контекстно-зависимого меню **Squiggle**.

**Изменение стиля** – команда контекстно-зависимого меню **Style**

**Изменение цвета** – команда контекстно-зависимого меню **Color**

**Изменение размера** – команда контекстно-зависимого меню **Trim**

**Редактирование формы.** Дуга представляет собой совокупность отдельных графических объектов: прямые участки, изогнутые участки, изображение наконечника. Отдельные элементы можно передвигать независимо друг от друга, меняя форму дуги, также дугу можно перемещать как единый неделимый элемент

### **Работа с текстовым блоком**

Для набора текста следует нажать инструмент **T** (Text Block Tool), после чего щелкнуть мышью в позиции предполагаемого ввода текста. Затем в появившемся диалоговом окне (рис.8).

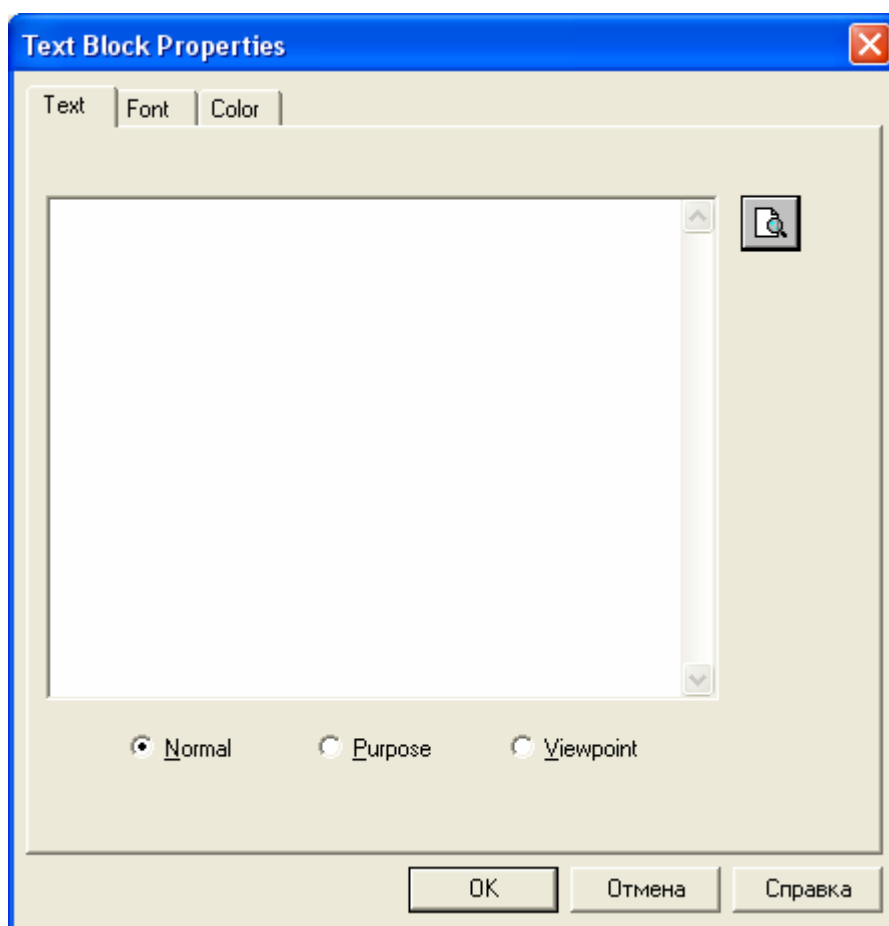
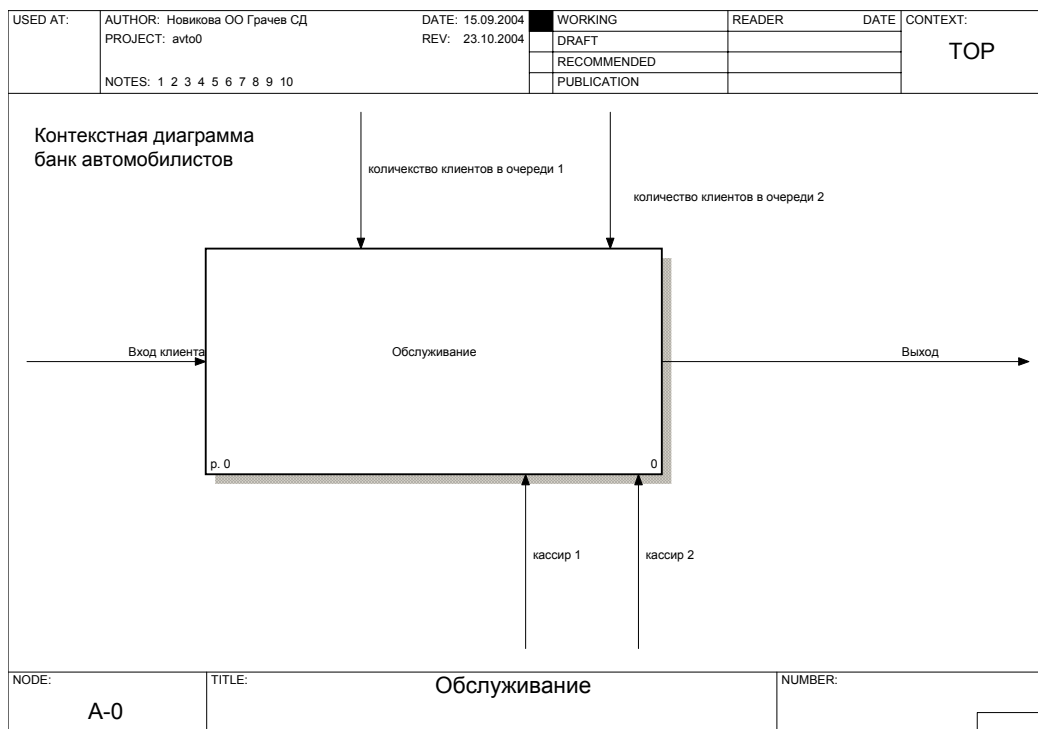


Рис.8.

набрать нужный текст и установить опцию значимости (обычный текст, цель, точка зрения).

### **Удаление блоков, дуг или текста**

Для удаления блока и дуги или текста необходимо их выделить щелчком левой кнопки мыши и нажать клавишу **Delete**, а затем подтвердить намерения по поводу удаления.



### Декомпозиция

После создания контекстной диаграммы необходимо расписать работу отдельных участков банка автомобилистов. Для этого декомпозируем эту диаграмму.

Для декомпозиции необходимо в браузере щелкнуть левой кнопкой мыши на имени диаграммы, а затем нажать кнопку ▼ (Go to Child Diagram), затем в диалоговом окне (Рис.9.):

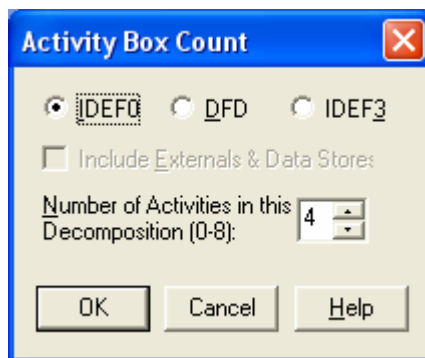


Рис.9.

ввести необходимое количество блоков и тип диаграммы:

Выполним декомпозицию блока диаграммы A-0, создав диаграмму A0, состоящую из 4 блоков: «Выбор очереди», «Обслужить касса 1», «Обслужить касса 2», «Выход».

Если в дальнейшем необходимо добавить блоки на диаграмме, то необходимо выбрать инструмент (**Activity Box Tool**) и щелкнуть мышью в нужном месте диаграммы.

После декомпозиции необходимо соединить получившиеся блоки дугами. Для этого необходимо выбрать инструмент , щелкнуть мышью по исходной стороне блока затем по конечной стороне следующего блока. Аналогично строятся разветвления и слияния дуг.

В результате получаем следующую диаграмму (рис.10.):



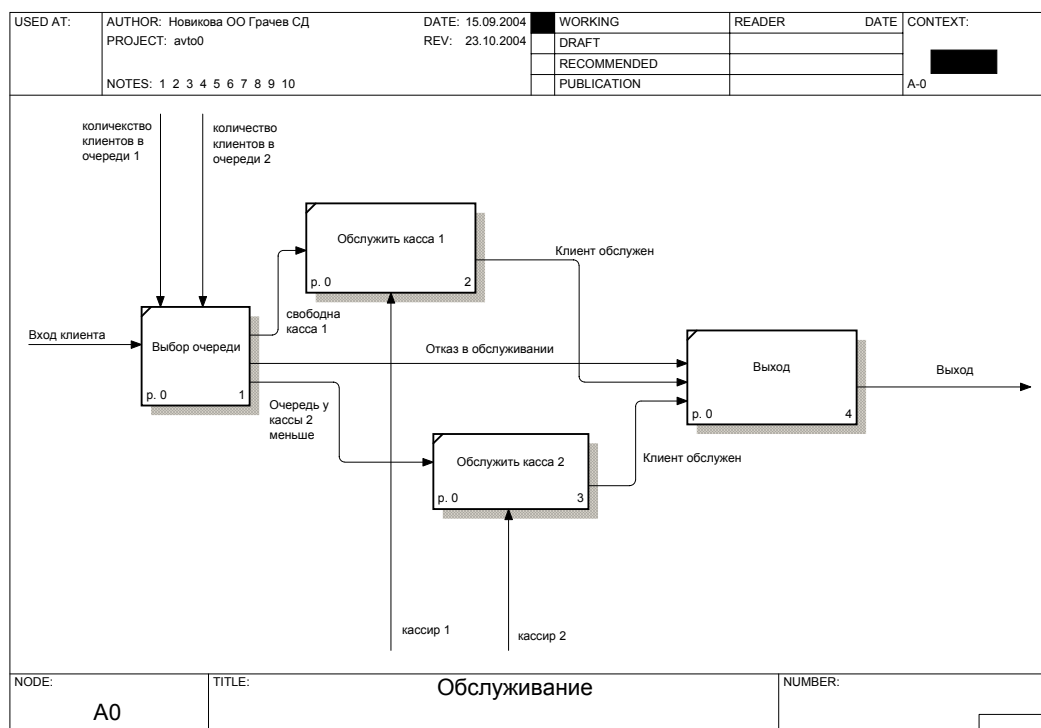


Рис.10.

Как видно в данной диаграмме используются только связи по выходу.

## Стоимостной анализ (ABC) и свойства, определяемые пользователем (UDP)

### Область применения

Стоимостной анализ используется для оценки модели. Он основан на работах (**Activity Based Costing, ABC**) и представляет собой соглашение об учете, используемое для сбора затрат, связанных с работами, с целью определить общую стоимость процесса. Обычно ABC применяется для того, чтобы понять происхождение выходных затрат и облегчить выбор нужной модели работ при реорганизации деятельности предприятия (Business Process Reengineering, BPR). ABC может проводиться только тогда, когда создание модели работы закончено.

### Основные элементы и их графическое изображение

ABC включает следующие основные понятия:

- объект затрат – причина, по которой работа выполняется, обычно, основной выход работы, стоимость работ есть суммарная стоимость объектов затрат.
- движитель затрат – характеристики входов и управлений работы, которые влияют на то, как выполняется и как долго длится работа;
- центры затрат, которые можно трактовать как статьи расхода.

При проведении стоимостного анализа в **BPwin** сначала задаются единицы измерения денег. Для задания единиц измерения следует вызвать диалог **Model Properties** (меню **Model/Model Properties**), закладка **ABC Units** (рис.11.).

**Model Properties**

General Purpose Definition Source Status Numbering Display  
Layout ABC Units Page Setup Header/Footer Shapes Draw Style

Model Name:  
avto0

**Cost**

Currency description: \$ U.S. Symbol placement: p. 1

Symbol: p. Number of decimals in diagrams: 0 Number of decimals in reports: 2

**Time**

Time Unit: Minutes Decimals in frequency values: 2 Decimals in duration values: 2

OK Отмена Применить Справка

Рис.11.

Если в списке выбора отсутствует необходимая валюта, ее можно добавить.

Затем описываются центры затрат (**cost centers**). Для внесения центров затрат необходимо вызвать диалог **Cost Center Editor** (меню **Model/Cost Center Editor**) рис.12.

**Cost Center Editor**

Cost center name (to be added after selected cost center):  
Касса 1

Cost centers:  
Касса 1  
Касса 2

Definition  
Обслуживание в кассе 1

Add Update Delete Close Help

Рис.12.

Каждому центру затрат следует дать подробное описание в окне **Definition**. Список центров затрат упорядочен. Порядок в списке можно менять при помощи стрелок, расположенных справа от списка. Задание определенной последовательности центров затрат в списке, во-первых, облегчает последующую работу при присвоении стоимости

работам, во-вторых, имеет значение при использовании единых стандартных отчетов в разных моделях.

Для задания стоимости работы (для каждой работы на диаграмме декомпозиции) следует щелкнуть правой кнопкой мыши по работе и на всплывающем меню выбрать **Costs** (рис.13).

UDP Values	UOW	Source	Roles	Box Style	
Name	Definition	Status	Font	Color	Costs
Activity Name: Обслужить касса 1					
Cost Center					\$ U.S.
Kassa 1					50.00
Kassa 2					0.00

This Activity has NO Decomposition. Total cost: 50,00

☐ Override decompositions Total cost x Frequency: 750,00

☐ Compute from decompositions

Frequency: 15.00

Duration: 0.30 Minutes

Duration x Frequency 4.50 Minutes

Cost Center Editor...

OK Отмена Применить Справка

Рис.13.

В диалоге **Activity Cost** указывается частота проведения данной работы в рамках общего процесса (окно **Frequency**) и продолжительность (**Duration**). Затем следует выбрать в списке один из центров затрат и в окне **Cost** задать его стоимость. Аналогично назначаются суммы по каждому центру затрат, т. е. задается стоимость каждой работы по каждой статье расхода. Если в процессе назначения стоимости возникает необходимость внесения дополнительных центров затрат, диалог **Cost Center Editor** вызывается прямо из диалога **Activity Cost** соответствующей кнопкой.

Общие затраты по работе рассчитываются как сумма по всем центрам затрат. При вычислении затрат вышестоящей (родительской) работы сначала вычисляется произведение затрат дочерней работы на частоту работы (число раз, которое работа выполняется в рамках проведения родительской работы), затем результаты складываются. Если во всех работах модели включен режим **Compute from Decompositions**, подобные вычисления автоматически проводятся по всей иерархии работ снизу вверх.

Если схема выполнения сложная (например, работы производятся альтернативно), можно отказаться от подсчета и задать итоговые суммы для каждой работы вручную (**Override Decompositions**).

Для проведения более тонкого анализа можно воспользоваться специализированным средством стоимостного анализа **EasyABC**. **BPwin** имеет двунаправленный интерфейс с **EasyABC**. Для экспорта данных в **EasyABC** следует выбрать пункт меню **File/Export/Node Tree**, задать в диалоге **Export node Tree** необходимые настройки и экспортировать дерево узлов в текстовый файл (.txt) рис.14.

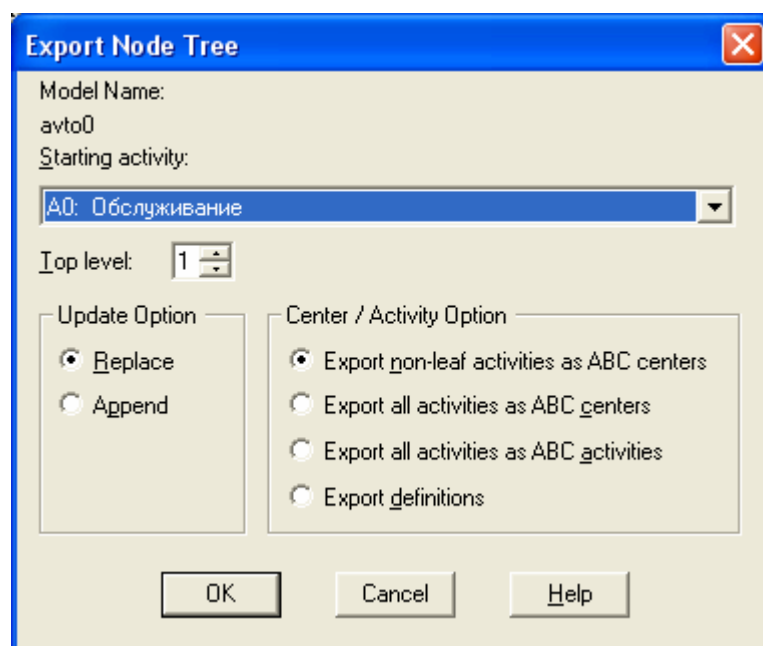


Рис.14.

Файл экспорта можно импортировать в **EasyABC**. После проведения необходимых расчетов результирующие данные можно импортировать из **EasyABC** в **BPwin**. Для импорта нужно выбрать меню **File/Import/Costs** и в диалоге **Import Activity Costs** выбрать необходимые установки (рис.15).

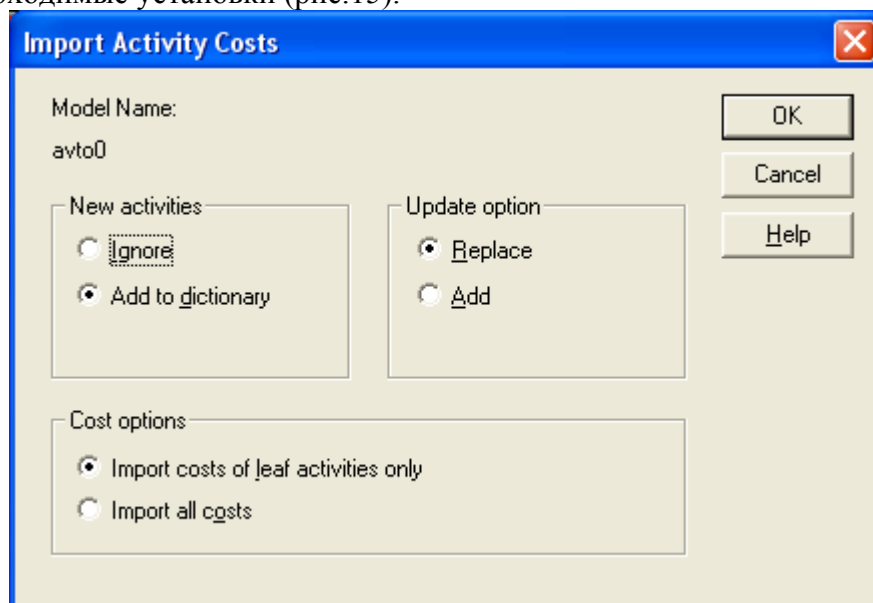


Рис.15.

Результаты стоимостного анализа наглядно представляются на специальном отчете **BPwin – Activity Cost Report** (меню **Tools/Report/Activity Cost Report...**).

Результаты отображаются и непосредственно на диаграммах. В левом нижнем углу прямоугольника работы может показываться либо стоимость (по умолчанию), либо продолжительность, либо частота проведения работы. Настройка отображения осуществляется в диалоге **Model Properties** (меню **Model/Model Properties**), закладка **Display, ABC Data, ABC Units**.

Если стоимостных показателей недостаточно, имеется возможность внесения собственных метрик – свойств, определенных пользователем (**User Defined Properties, UDP**) [1]. **UDP** позволяет провести дополнительный анализ, хотя и без суммирующих подсчетов.

Для описания **UDP** служит диалог **User-Defined Property Name Editor** (меню **Model/UDP Definition**) рис.16.

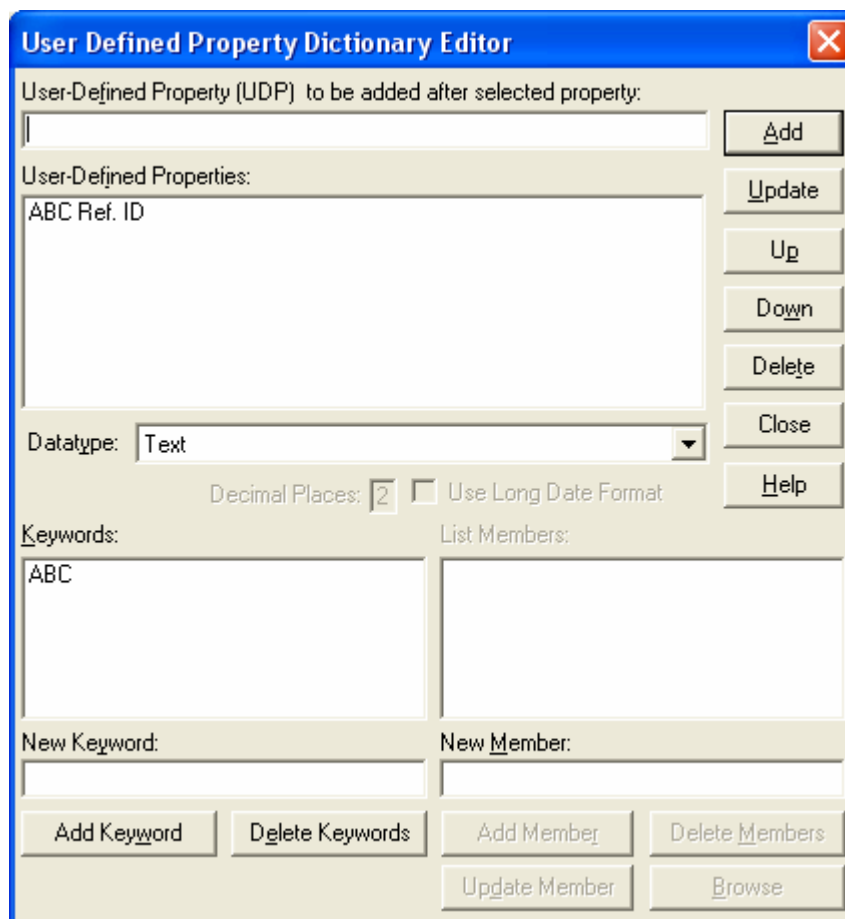


Рис.16.

В верхнем окне диалога вносится имя **UDP**, в списке выбора **Datatype** описывает тип свойства. Имеется возможность задания 18 различных типов **UDP**, в том числе управляющих команд и массивов, объединенных по категориям. Для внесения категории следует задать имя категории в окне **New Category/Member** и щелкнуть по кнопке **Add Category**. Для присвоения свойства категории необходимо выбрать **UDP** из списка, затем категорию из списка категорий и щелкнуть по кнопке **Update**. Одна категория может объединять несколько свойств, в то же время одно свойство может входить в несколько категорий. Свойство типа **List** может содержать массив предварительно определенных значений. Для определения области значений **UDP** типа **List** следует задать значение свойства в окне **New Category/Member** и щелкнуть по кнопке **Add Member**. Значения из списка можно редактировать и удалять.

Каждой работе можно поставить в соответствие набор **UDP**. Для этого следует щелкнуть правой кнопкой мыши по работе и выбрать пункт меню **UDP Editor**. В закладке **UDP Values** диалога **IDEF0 Activity Properties** можно задать значения **UDP**. Свойства типа **List** отображаются списком выбора, который заполнен предварительно определенными значениями.

Кнопка **Categories** служит для задания фильтра по категориям **UDP**. По умолчанию в списке показываются свойства всех категорий.

Результат задания проанализировать в отчете **Diagram Object Report** (меню **Report/Diagram Object Report...**).

В левом нижнем углу диалога настройки отчета показывается список **UDP**. С помощью кнопки **Activity Categories** можно установить фильтр по категориям.

Проведем стоимостной анализ нашего примера.

Нами были заданы центры затрат – касса 1 и касса 2. каждому центру затрат задали описание в окне (рис.17.):

**Activity Properties**

UDP Values | UOW | Source | Roles | Box Style

Name | Definition | Status | Font | Color | Costs

Activity Name:  
Обслужить касса 1

Cost Center	\$ U.S.
Касса 1	50,00
Касса 2	0,00

This Activity has NO Decomposition. Total cost: 50,00

☐ Override decompositions Total cost x Frequency: 750,00

☐ Compute from decompositions

Frequency: 15,00

Duration: 0,30 Minutes

Duration x Frequency 4,50 Minutes

Cost Center Editor...

OK Отмена Применить Справка

Рис.17.

Далее был проведен общий расчет как сумма по всем центрам затрат (рис.18).

Стоимостной анализ:

Activity Number: 0

Activity Name: Банк

Activity Cost (\$ U.S.): 24,00

Cost Center: Стоимость обслуживания

Cost Center Cost (\$ U.S.): 24,00

Activity Number: 1

Activity Name: Выбор очереди

Activity Cost (\$ U.S.): 0,00

Activity Number: 2

Activity Name: Касса1

Activity Cost (\$ U.S.): 12,00

Cost Center: Стоимость обслуживания

Cost Center Cost (\$ U.S.): 12,00

Activity Number: 3

Activity Name: Касса2

Activity Cost (\$ U.S.): 12,00

Cost Center: Стоимость обслуживания

Cost Center Cost (\$ U.S.): 12,00

Activity Number: 4

Activity Name: Выход

Activity Cost (\$ U.S.): 0,00

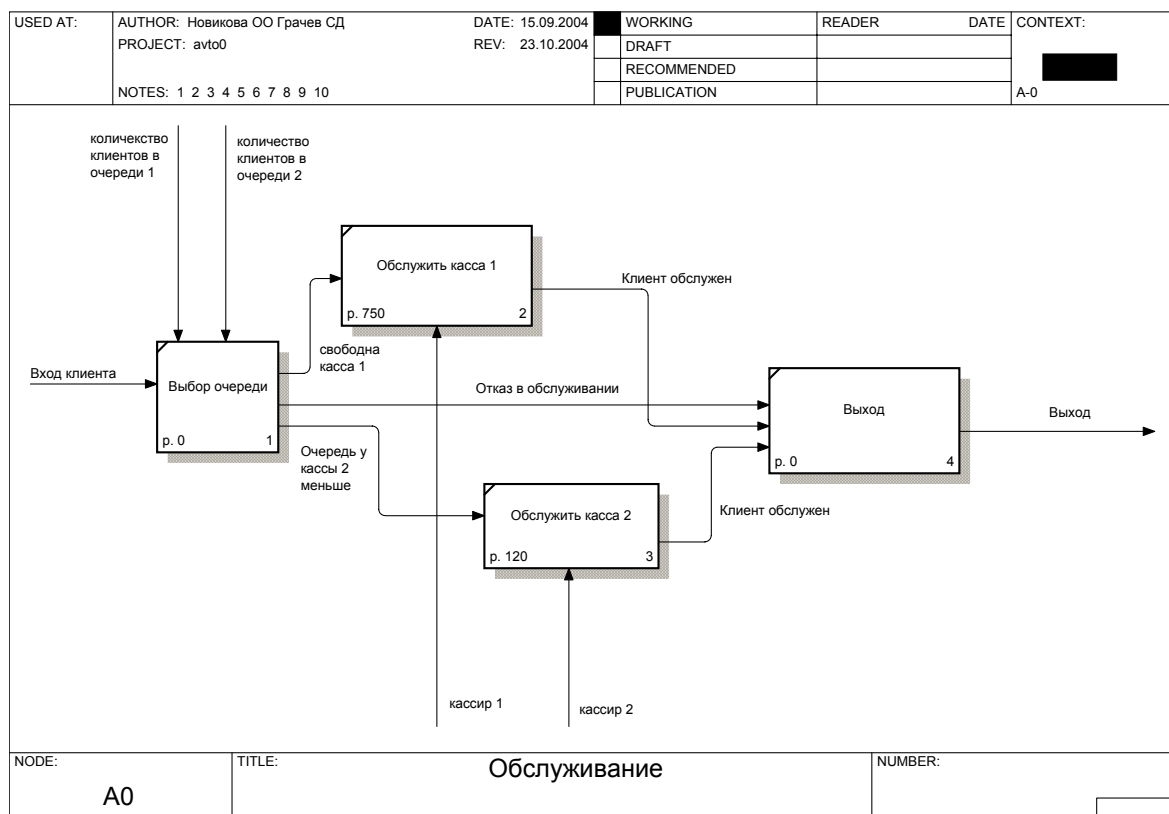


Рис.18.

## Диаграммы Workflow (IDEF3)

### Область применения

**IDEF3** – используется для описания логики взаимодействия информационных потоков. Эта методология моделирования использует графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов. Диаграммы Workflow могут быть использованы в моделировании бизнес-процессов для анализа завершенности процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации, например, последовательность обработки заказа или события, которые необходимо обработать за конечное время.

### Основные элементы

Диаграммы	Основная единица описания в IDEF3.
Единица работы (UOW)	Центральный компонент модели. Изображаются прямоугольниками с прямыми углами и имеют имя, выраженное отглагольным существительным, обозначающим процесс действия, одиночным или в составе фразы, и номер; другое имя существительное в составе той же фразы обычно отображает основной выход (результат работы).
Связи	Показывают взаимоотношение работ. Все связи в IDEF3 однонаправлены и могут быть направлены куда угодно, но обычно диаграммы IDEF3 стараются построить так, чтобы связи были направлены слева направо.

В **IDEF3** различают три типа стрелок (рис.19), изображающих связи, стиль которых устанавливается через контекстное меню **Style**:

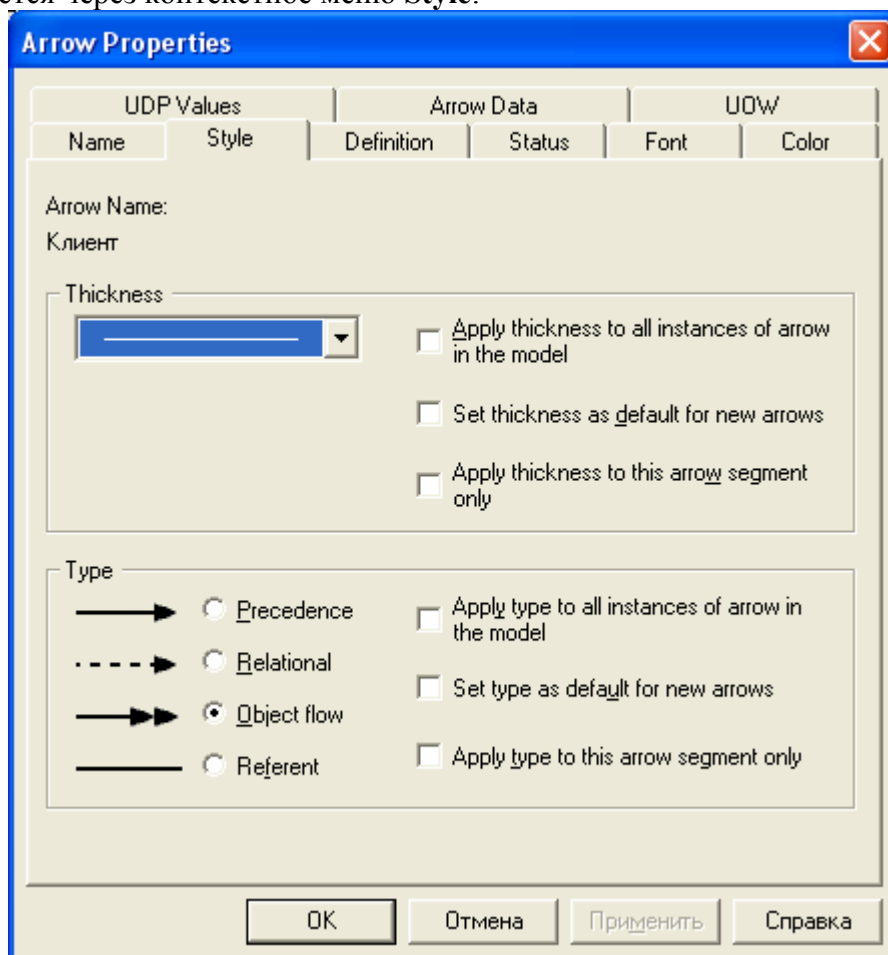

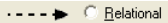




Рис.19.



<b>Старшая</b> 	Сплошная линия, связывающая единицы работ. Рисуется слева направо или сверху вниз. Показывает, что работа-источник должна закончиться прежде, чем работа-цель начнется.
<b>Отношения</b> 	Пунктирная линия, используемая для изображения связей между единицами работ, а также между единицами работ и объектами ссылок.
<b>Потоки объектов</b> 	Стрелка с двумя наконечниками, применяется для описания того факта, что объект используется в двух или более единицах работы, например когда объект порождается в одной работе и используется в другой.

**Старшая связь и поток объектов.** Старшая связь показывает, что работа-источник заканчивается ранее, чем начинается работа-цель. Часто результатом работы-источника становится объект, необходимый для запуска работы-цели. В этом случае стрелку, обозначающую объект, изображают с двойным наконечником.

**Перекрестки.** Окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончания нескольких работ. Перекрестки используются для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Различают перекрестки для слияния и разветвления стрелок. Перекресток не может использоваться одновременно для слияния и

для разветвления. Для внесения перекрестка служит кнопка  в палитре инструментов . В диалоге **Select Junction Type** (рис.20).

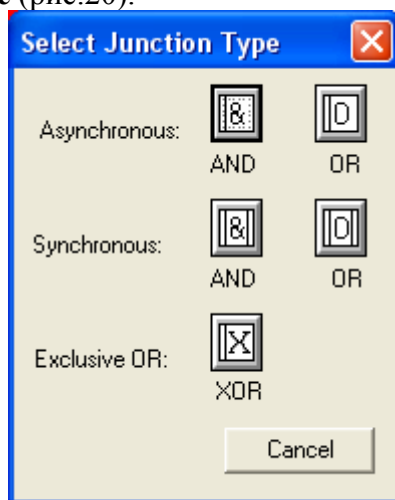



Рис.20.

необходимо указать тип перекрестка.

Наименование	Смысл в случае слияния стрелок	Смысл в случае разветвления стрелок
Asynchronous AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
Synchronous AND	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
Asynchronous OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
Synchronous OR	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются одновременно
XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс **J**. Можно редактировать свойства перекрестка окне его свойств. В **IDEF3** стрелки могут сливаться и разветвляться только через перекрестки.

**Объект ссылки.** Выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, перекрестком или работой. Для внесения объекта ссылки служит кнопка  в палитре инструментов. Объект ссылки изображается в виде прямоугольника, похожего на прямоугольник работы. Имя объекта ссылки задается в диалоге **Referent** (Рис.21.).

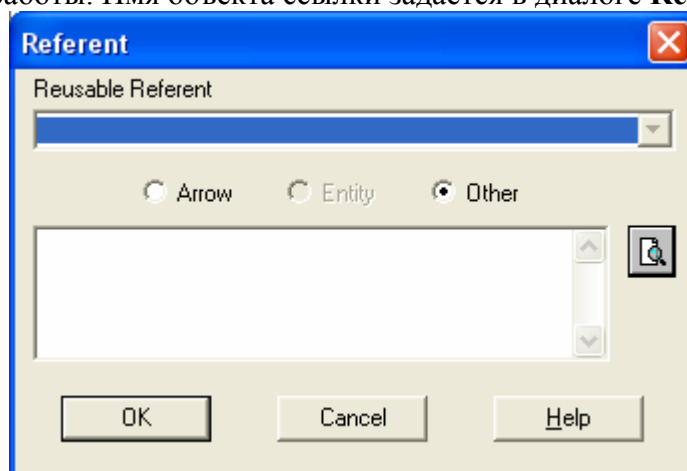


Рис.21.

В качестве имени можно использовать имя какой-либо стрелки с других диаграмм или имя сущности из модели данных. Объекты ссылки должны быть связаны с единицами работ или перекрестками пунктирными линиями. Официальная спецификация **IDEF3** различает три стиля объектов ссылок – **безусловные**, **синхронные** и **асинхронные**.

При внесении объектов ссылок помимо имени следует указывать тип объекта ссылки:

Тип объекта ссылки	Цель описания
OBJECT	Описывает участие важного объекта в работе
GOTO	Инструмент циклического перехода (в повторяющейся последовательности работ), возможно на текущей диаграмме, но не обязательно. GOTO может ссылаться на перекресток.
UOB	Применяется, когда необходимо подчеркнуть множественное использование какой-либо работы, но без цикла.
NOTE	Используется для документирования важной информации, относящейся к каким-либо графическим объектам на диаграмме.
ELAB	Используется для усовершенствования графиков или их более детального описания. Обычно употребляется для детального описания разветвления и слияния стрелок на перекрестках.

**Декомпозиция работ.** В IDEF3 декомпозиция используется для детализации работ. Можно многократно декомпозировать работу, т.е. работа может иметь множество дочерних работ. При этом номер работы состоит из номера родительской работы, версии декомпозиции и собственного номера работы на текущей диаграмме.

Построим диаграмму детализации работ банка автомобилистов в стандарте IDEF3. Начнем с построения контекстной диаграммы.

Для этого поместим 1 блок на диаграмму, и назовем его «Обслужить». После этого проведем связи между данной работой и внешними работами (рис.22).

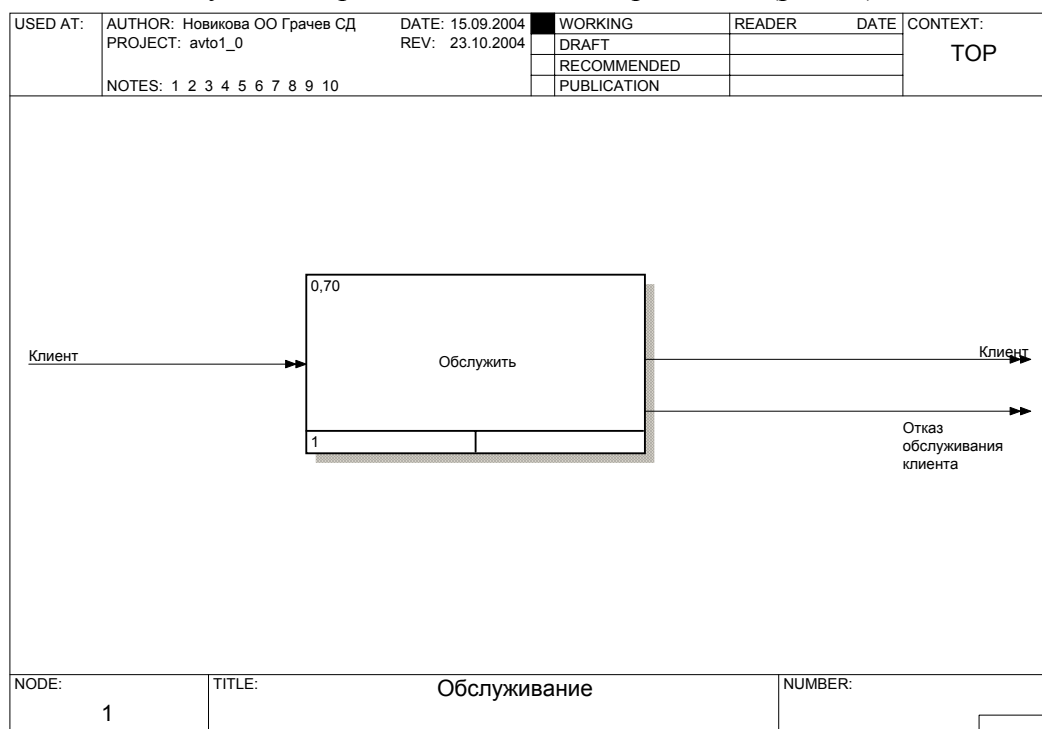


Рис.22.

Как видно, в банк поступает клиент, а из банка клиент либо выходит обслуженным, либо ему отказывается в обслуживании по условиям задачи. Для более детального рассмотрения работ декомпозируем данную работу на 3 вида работ:

1. Появление клиента
2. Обслужить клиента касса 1
3. Обслужить клиента касса 2

После декомпозиции получили следующую диаграмму (рис.23).

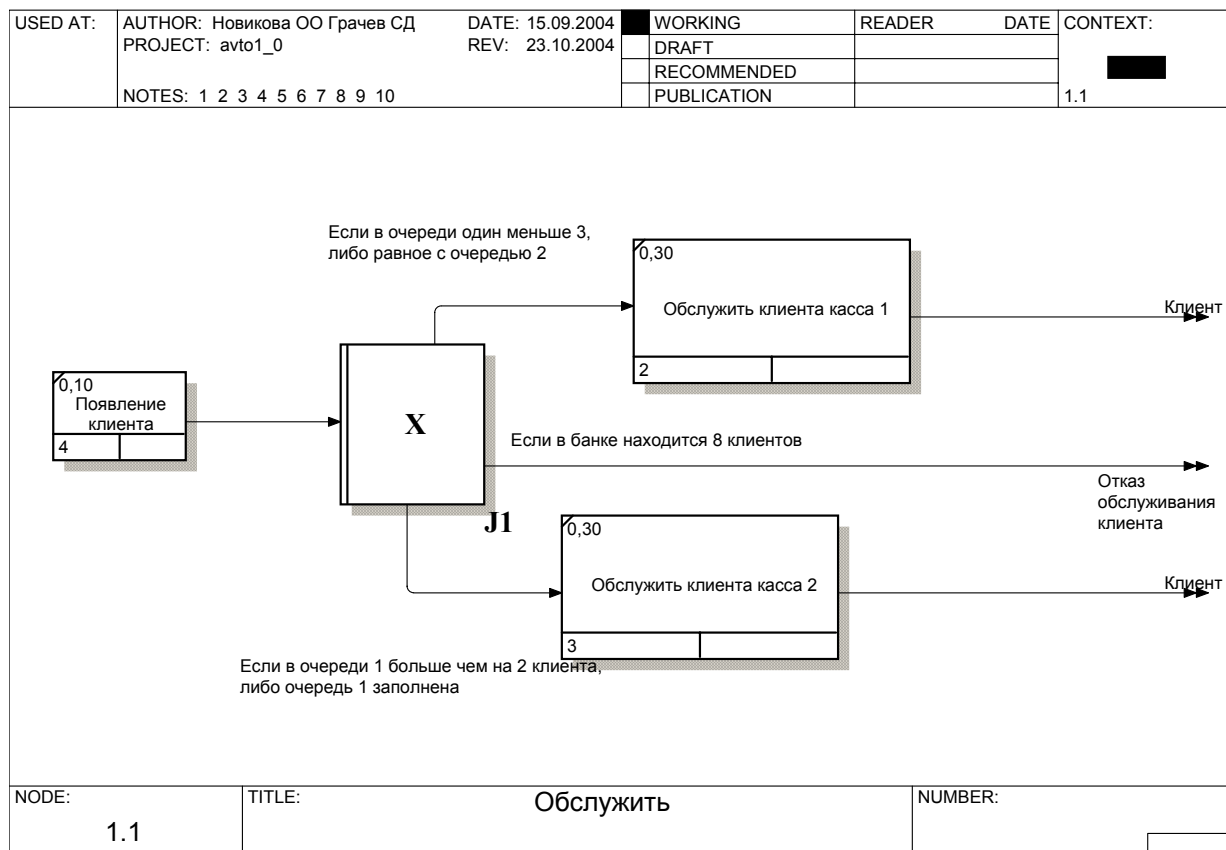


Рис.23.

На декомпозированной диаграмме видно, какие виды работ необходимо реализовывать для выполнения основной задачи. При необходимости каждую из этих работ можно декомпозировать на более мелкие для описания конкретной реализации.

**Перейдем к диаграмме DFD.**

### Область применения

Диаграммы потоков данных (**Data flow diagramming, DFD**) используются для описания документооборота и обработки информации. Подобно **IDEF0**, **DFD** представляет модельную систему как сеть связанных между собой работ.

### Создание диаграммы DFD.

Для создания диаграммы DFD необходимо в диалоге New Model (рис.24.) выбрать кнопку DFD

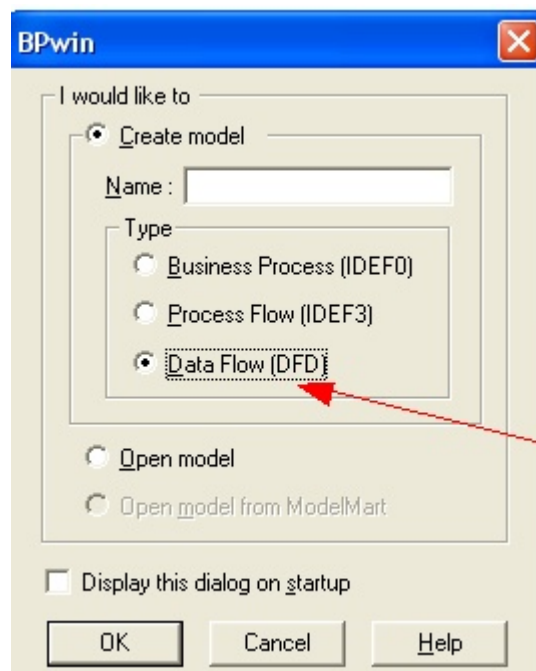


Рис.24.

**Декомпозиция работы IDEF0 в диаграмму DFD.** Для создания дочерней диаграммы **DFD** следует при декомпозиции в диалоге (рис.25) **Activity Box Count** выбрать кнопку **DFD**.

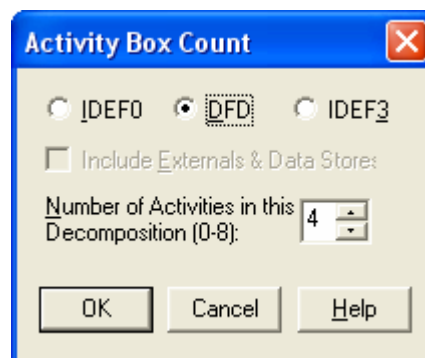


Рис.25.

Создается новая диаграмма **DFD**, и стрелки, которые касаются родительской работы, мигрируют на диаграмму нижнего уровня так, как если бы это была диаграмма **IDEF0**.

Стрелки входа родительской работы на дочерней диаграмме **DFD** показываются входящими стрелками с левой стороны диаграммы **DFD**, стрелки управления – входящими стрелками с верхней стороны диаграммы и т. д.

Согласно нотации **DFD** диаграмма не должна иметь граничных стрелок – все стрелки должны начинаться и заканчиваться на работах, хранилищах данных или внешних сущностях. Поэтому, если строго следовать правилам нотации, надо:

1. Удалить все граничные стрелки на диаграмме **DFD**.
2. Создать соответствующие внешние сущности и хранилища данных.
3. Создать внутренние стрелки, начинающиеся с внешних сущностей вместо граничных стрелок.

## Основные элементы

**DFD** описывает:

- функции обработки информации (работы);
- документы (стрелки, **arrow**), объекты, сотрудников или отделы, которые участвуют в обработке информации;
- внешние ссылки (**external references**), которые обеспечивают интерфейс с внешними объектами, находящимися за границами моделируемой системы;

- таблицы для хранения документов (хранилище данных, **data store**).

В **BPwin** для построения диаграмм потоков данных используется нотация Гейна-Сарсона.

Для того чтобы дополнить модель **IDEF0** диаграммой **DFD**, нужно в процессе декомпозиции в диалоге **Activity Box Count** “кликнуть” по радиокнопке **DFD**. В палитре инструментов на новой диаграмме **DFD** появляются новые кнопки (Рис. 26):

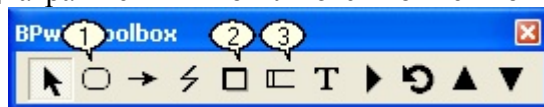


Рис.26.

- 1) **добавить в диаграмму работу Activity Box Tool** – используется для установки блоков в диаграмме.
- 2) **добавить в диаграмму внешнюю ссылку (External Reference)**. Внешняя ссылка является источником или приемником данных извне модели;
- 3) **добавить в диаграмму хранилище данных (Data store)**. Хранилище данных позволяет описать данные, которые необходимо сохранить в памяти прежде, чем использовать в работах;

Стрелки **DFD** показывают, как объекты (включая данные) двигаются от одной работы к другой. Это представление потоков совместно с хранилищами данных и внешними сущностями делает модели DFD более похожими на физические характеристики системы – движение объектов (data flow), хранение объектов (data stores), поставка и распространение объектов (external entities) рис.27.

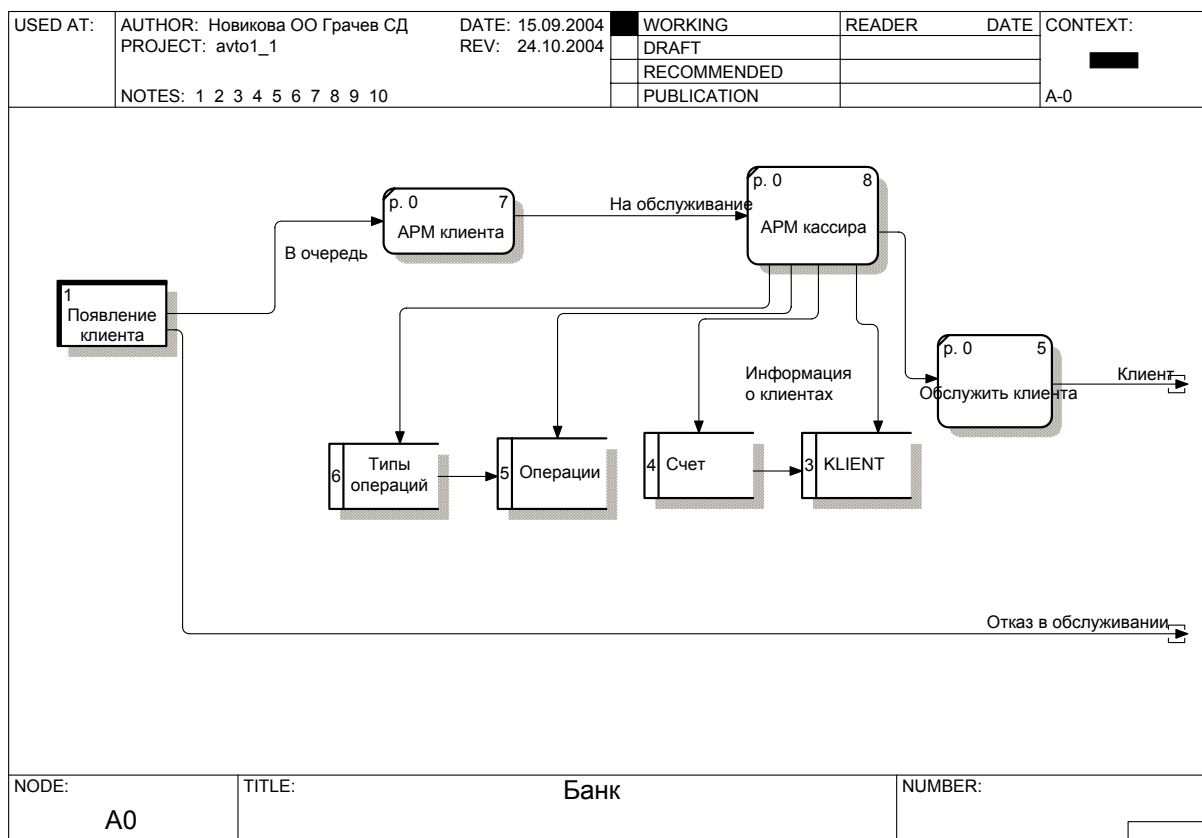


Рис.27.

**DFD** рассматривает систему как совокупность предметов. Контекстная диаграмма часто включает работы и внешние ссылки. Работы обычно именуются по названию системы, например «*АРМ клиента*».

<b>Работы</b>	Представляют собой функции системы, преобразующие входы в выходы. Изображаются прямоугольниками со скругленными углами.
<b>Внешние сущности</b>	Изображают входы в систему и/или выходы из системы. Изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы.
<b>Стрелки (Потоки данных)</b>	Описывают движение объектов из одной части системы в другую. Стрелки могут подходить и выходить из любой грани прямоугольника работы. Также применяются двунаправленные стрелки для описания диалогов типа «команда-ответ» между работами, между работой и внешней сущностью и между внешними сущностями.
<b>Хранилище данных</b>	Изображают объекты в покое, в отличие от стрелок, описывающих объекты в движении. Это очереди и т. п.

**Слияние и разветвление стрелок.** В DFD стрелки могут сливаться и разветвляться, что позволяет описать декомпозицию стрелок. Каждый новый сегмент сливающейся или разветвляющейся стрелки может иметь собственное имя.

**Нумерация объектов.** В DFD номер каждой работы может включать префикс, номер родительской работы (**A**) и номер объекта. Номер объекта – это уникальный номер работы на диаграмме. Уникальный номер имеют хранилища данных и внешние сущности независимо от их расположения на диаграмме. Каждое хранилище данных имеет префикс **D** и уникальный номер, например **D5**. Каждая внешняя сущность имеет префикс **E** и уникальный номер.

А теперь рассмотрим DFD диаграмму нашего примера.

Сначала построим контекстную диаграмму для определения внешних связей нашего объекта. Для этого поместим в область диаграммы блок работы (**Activity Box Tool**). В нашем примере внешними сущностями являются события «Появление клиента» и «Уход клиента». Для отображения данных сущностей на диаграмме добавим на нее два блока **External Reference** слева и справа от основной работы. После этого соединим внешние сущности с работой. Получили следующее см. рис.28.

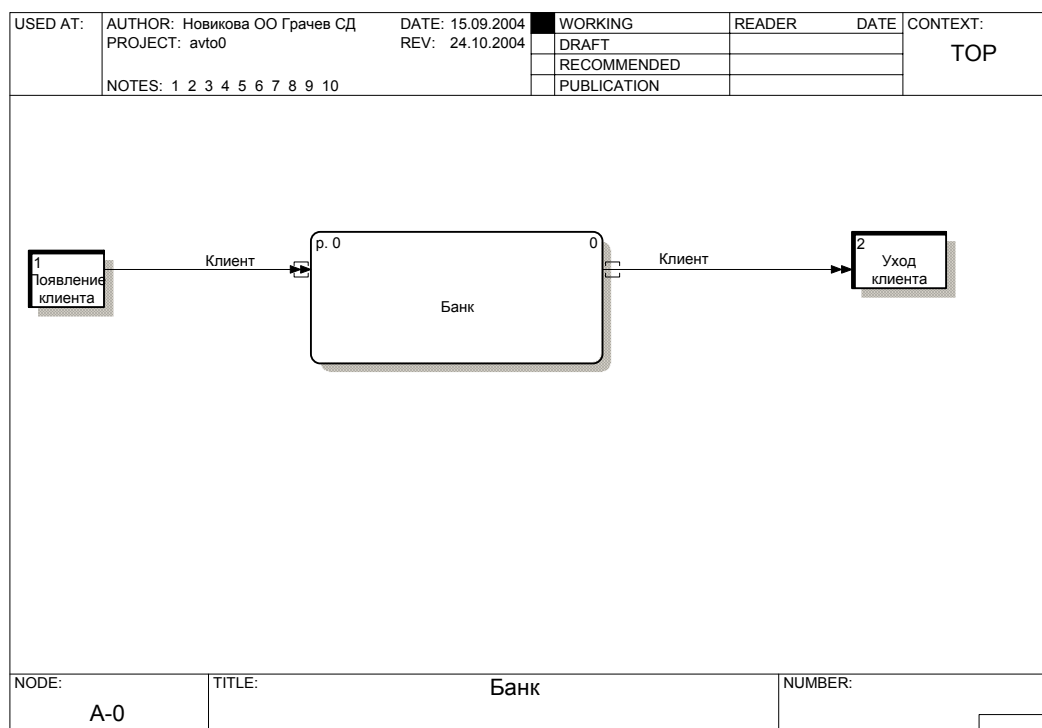


Рис.28.

Для более детального анализа передвижений внутри банка декомпозируем диаграмму (рис.29).

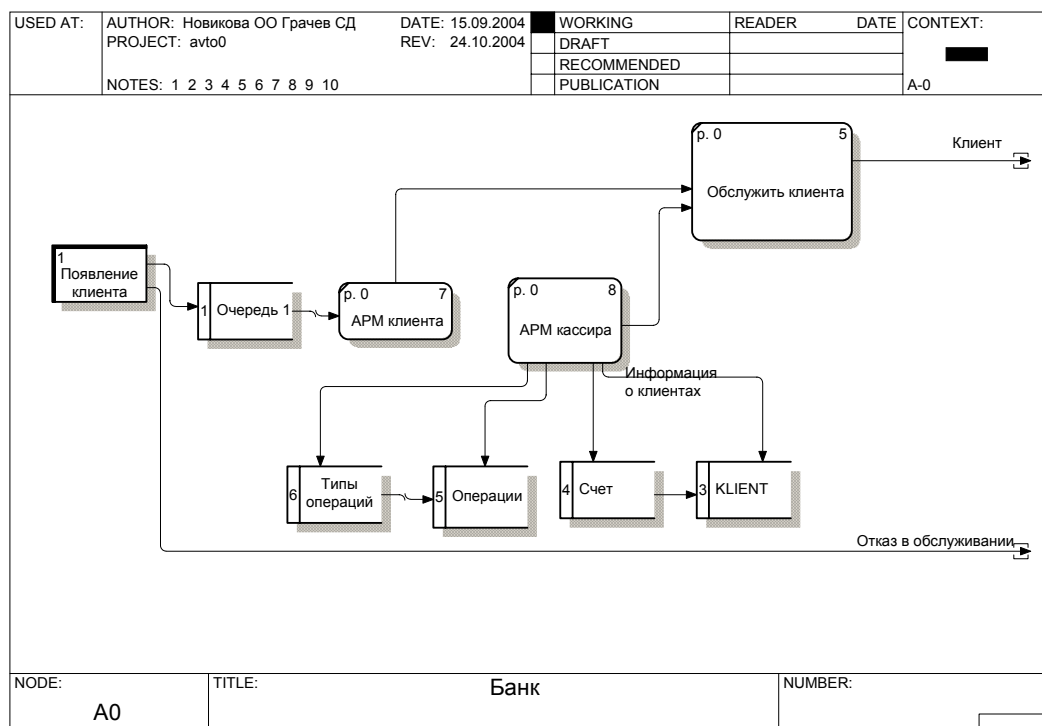


Рис.29.

После декомпозиции добавляем на диаграмму необходимые хранилища и соединяем блоки между собой.

Хранилище данных позволяет на определенных участках определить данные, которые будут сохраняться в памяти между процессами.

На DFD диаграмме можно увидеть саму структуру будущей работы банка, со всеми используемыми сущностями данных.



Данная декомпозированная DFD диаграмма будет являться основой для создания структуры БД хранения данных, поэтому тщательным образом необходимо продумывать каждое хранилище.

## ЧАСТЬ II: Использование ERwin для составления информационной модели

### Область применения

Erwin используется для построения модели данных. **ERwin** имеет два уровня представления модели – **логический** и **физический**. На логическом уровне данные не связаны с конкретной СУБД. Физический уровень данных – это по существу отображение системного каталога, который зависит от конкретной реализации СУБД. **ERwin** позволяет проводить процессы прямого и обратного проектирования БД. Это означает, что по модели данных можно сгенерировать схему БД или автоматически создать модель данных на основе информации системного каталога. Для создания моделей данных в Erwin используются две методологии: IDEF1X и IE. В данной работе рассматривается методология IDEF1X.

### Отображение модели данных в ERwin.

**ERwin** имеет два уровня представления модели – **логический** и **физический**. **Логический уровень** – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире, например, «*Фамилия сотрудника*», «*Отдел*». Объекты модели, представляемые на логическом уровне, называются **сущностями** и **атрибутами**. Логическая модель может быть построена на основе другой логической модели, например на основе модели процессов. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

**Физическая модель данных**, напротив, зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация о всех объектах БД. Поскольку стандартов на объекты БД не существует, физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей. Разделение модели данных на логические и физические позволяет решить несколько важных задач.

**Документирование модели.** На физическом уровне объекты БД могут называться так, как того требуют ограничения СУБД. На логическом уровне можно этим объектам дать синонимы – имена более понятные неспециалистам, в том числе на кириллице и с использованием специальных символов.

**Масштабирование.** Создание модели данных, как правило, начинается с создания логической модели. После описания логической модели, проектировщик может выбрать необходимую СУБД, и **ERwin** автоматически создаст соответствующую физическую модель. На основе физической модели **ERwin** может сгенерировать системный каталог СУБД или соответствующий **SQL-скрипт**. Этот процесс называется **прямым проектированием (Forward Engineering)**. Тем самым достигается масштабируемость – создав одну логическую модель данных, можно сгенерировать физические модели под любую поддерживаемую **ERwin** СУБД. С другой стороны, **ERwin** способен по содержимому системного каталога или **SQL-скрипту** воссоздать физическую и логическую модель данных (**Reverse Engineering**). На основе полученной логической модели данных можно сгенерировать физическую модель для другой СУБД и затем сгенерировать ее системный каталог. Следовательно, **ERwin** позволяет решить задачу по переносу структуры данных с одного сервера на другой [1].

## 2. Основные элементы

Для переключения между логической и физической моделью данных служит список выбора в левой части панели инструментов **ERwin**.

При переключении, если физической модели еще не существует, то она будет создана автоматически.

Палитра инструментов выглядит различно на разных уровнях отображения модели.

На логическом уровне палитра инструментов имеет (рис.30):

ErWin ToolBox

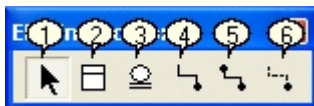


Рис.30.

1. **кнопка указателя** (режим мыши) – в этом режиме можно установить фокус на каком-либо объекте модели;
2. **кнопка внесения сущности** – для внесения сущности нужно щелкнуть левой кнопкой мыши по кнопке внесения сущности и один раз по свободному пространству на модели. Повторный щелчок приведет к внесению в модель еще одной новой сущности. Для редактирования сущностей или других объектов модели необходимо перейти в режим указателя;
3. **кнопка категории**. Категория, или категориальная связь, - специальный тип связи между сущностями. Для установления категориальной связи нужно щелкнуть левой кнопкой мыши по кнопке категории, затем один раз щелкнуть по сущности-родовому предку, затем – по сущности-потомку;
- 4 – 6. **кнопки создания связей**: идентифицирующую, «многие-ко-многим» и неидентифицирующую.

Drawing Object (рис.31):



Рис.31.

На этой панели расположены различные элементы для рисования (прямоугольник, круг, линия). Предназначены для создания вспомогательных элементов на схеме.

На физическом уровне палитра инструментов имеет (рис.32):

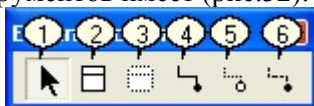


Рис.32.

1. **кнопка указателя** (режим мыши) – в этом режиме можно установить фокус на каком-либо объекте модели;
2. **кнопка внесения сущности** – для внесения сущности нужно щелкнуть левой кнопкой мыши по кнопке внесения сущности и один раз по свободному пространству на модели. Повторный щелчок приведет к внесению в модель еще одной новой сущности. Для редактирования сущностей или других объектов модели необходимо перейти в режим указателя;
3. **кнопка внесения представлений (view)**;
4. **кнопки создания связей**: идентифицирующая
5. **кнопки создания связей**: связь представлений.
6. **кнопки создания связей**: неидентифицирующая

Для создания моделей данных в **ERwin** можно использовать две нотации: **IDEF1X** и **IE (Information Engineering)**. Переключение между нотациями можно сделать в закладке **Notation** диалога **Model Properties** (меню **Model/ Model Properties ...**) рис.33.

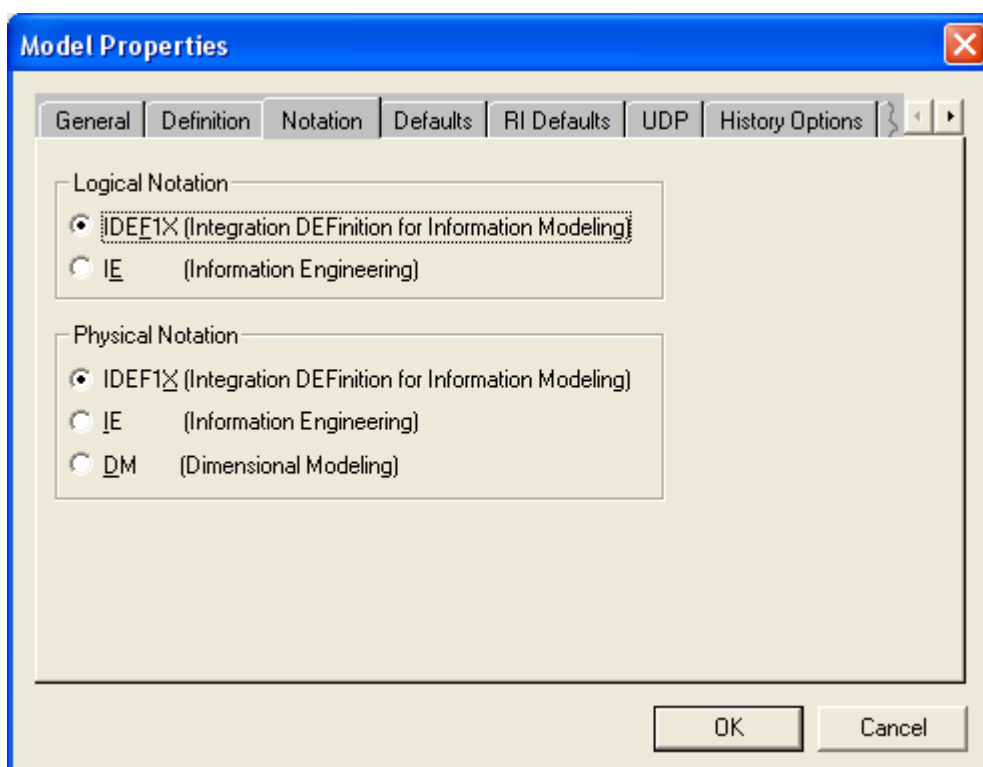


Рис.33.

В данной работе используется нотация **IDEF1X**.

**ERwin** имеет несколько уровней отображения диаграммы: уровень сущностей, уровень атрибутов, уровень определений, уровень первичных ключей и уровень иконок. Переключиться между первыми тремя уровнями можно с использованием кнопок панели инструментов. Переключиться на другие уровни отображения можно при помощи контекстного меню, которое появляется, если “кликнуть” по любому месту диаграммы, не занятому объектами модели. В контекстном меню следует выбрать пункт **Display Level** и затем необходимый уровень отображения (рис.34).

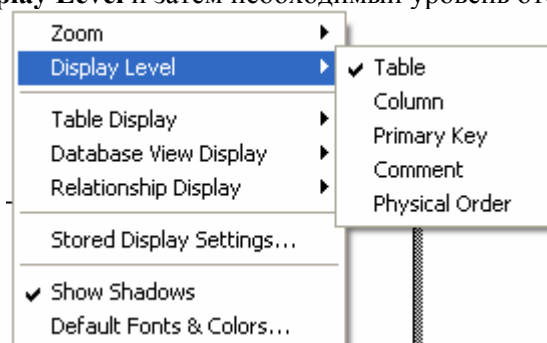


Рис.34.

## Создание логической модели данных

### Уровни логической модели

Различают три уровня логической модели, отличающихся по глубине представления информации о данных:

- \* диаграмма сущность-связь (**Entity Relationship Diagram (ERD)**);
- \* модель данных, основанная на ключах (**Key Based model (KB)**);
- \* полная атрибутивная модель (**Fully Attributed model (FA)**).

<b>Диаграмма сущность-связь</b>	- представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Диаграмма сущность-связь может включать связи многие-ко-многим и не включать описание ключей.
---------------------------------	---

<b>Модель данных, основанная на ключах</b>	- более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.
<b>Полная атрибутивная модель</b>	– наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

### Сущности и атрибуты

Основные компоненты диаграммы **ERwin** – это сущности, атрибуты и связи. Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. Атрибут выражает определенное свойство объекта. С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы.

Построение модели данных предполагает определение сущностей и атрибутов, т.е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте. Сущность можно определить как объект, событие или концепцию, информация о которой должна сохраняться. Сущности должны иметь наименование с четким смысловым значением. Фактически имя сущности дается по имени ее экземпляра. Примером может быть сущность **Клиент** (но не **Клиенты**!) с атрибутами **Номер Клиента**, **Фамилия Клиента** и **Адрес Клиента**. На уровне физической модели ей может соответствовать таблица **Client** с колонками **Client\_number**, **Client\_name** и **Client\_address**.

Для внесения сущности в модель необходимо убедиться, что вы находитесь на уровне логической модели, и «кликнуть» по кнопке сущности на панели инструментов (**ERwin Toolbox**), затем «кликнуть» по тому месту на диаграмме, где необходимо расположить новую сущность. Щелкнув правой кнопкой мыши по сущности и выбрав из всплывающего меню пункт **Entity Properties**, можно вызвать диалог **Entities**, в котором определяются имя, описание и комментарии сущности. Каждая сущность должна быть полностью определена с помощью текстового описания в закладке **Definition** (рис.35).

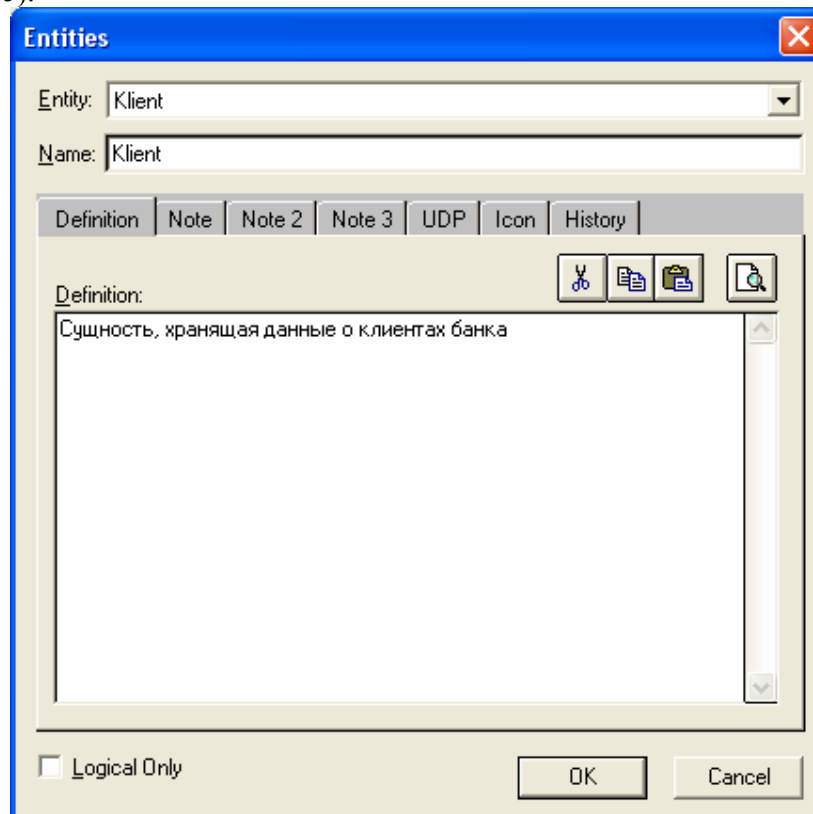


Рис.35.

Закладки **Note**, **Note 2**, **Note 3** (в прежних версиях **Query** и **Sample**), **UDP (User Defined Properties)** служат для внесения дополнительных комментариев и определений к сущности:

<b>Definition</b>	- используется для ввода определения сущности.
<b>Note</b>	- можно ввести полезное замечание, описывающее какое-либо бизнес-правило или соглашение по организации диаграммы.
<b>Note 2</b>	- можно задокументировать некоторые возможные запросы, которые, как ожидается, будут использоваться по отношению к сущности в БД.
<b>Note 3</b>	- позволяет вводить примеры данных для сущности (в произвольной форме).
<b>Icon</b>	- каждой сущности можно поставить в соответствие изображение, которое будет отображаться в режиме просмотра модели на уровне иконок.

Для определения **UDP** служит диалог **User-Defined Properties**(меню **Model/UDP dictionary...**)  
рис.36.

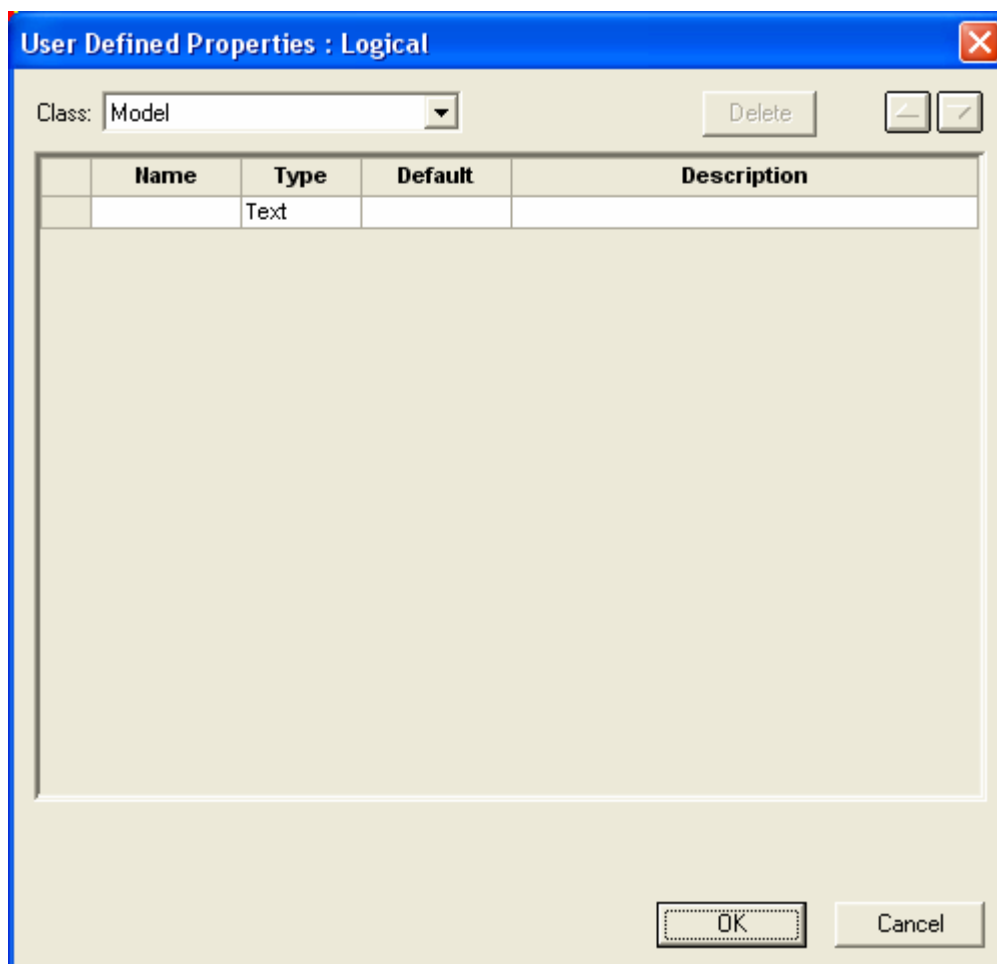


Рис.36.

В нем необходимо указать вид объекта, для которого заводится **UDP** (диаграмма в целом, сущность, атрибут и т. д.) и тип данных. **ERwin** поддерживает для **UDP** шесть типов данных, в том числе:

**List** – список - при задании списка значения следует разделять запятой, значение по умолчанию выделяется символом «~»;

**Command** – команда – выполняемая строка.

Значение свойств, определяемых пользователем, задается в закладке **UDP** диалога **Entity Properties**.

Атрибут или группа атрибутов, которые идентифицируют сущность, называется **первичным ключом**. Для описания атрибутов следует, «кликнув» правой кнопкой по сущности, выбрать в появившемся меню пункт **Attributes**. Появляется диалог **Attributes** (рис.37).

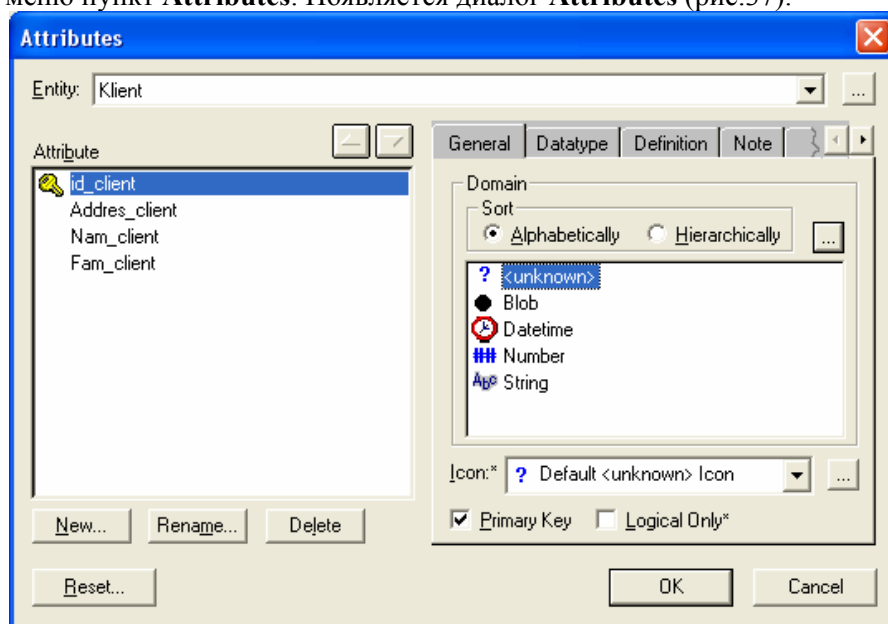


Рис.37.

Если щелкнуть по кнопке **New...**, то в появившемся диалоге **New Attribute** можно указать имя атрибута, имя соответствующей ему в физической модели колонки и домен. Домен атрибута будет использоваться при определении типа колонки на уровне физической модели (рис.38).

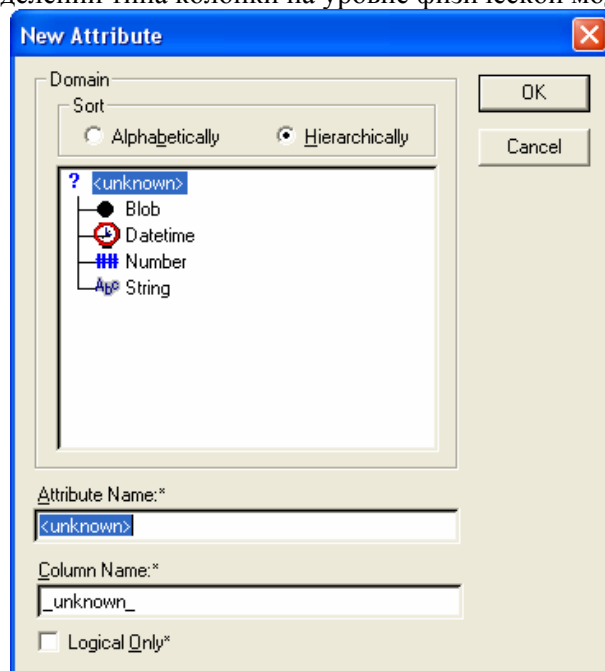


Рис.38.

Для атрибутов первичного ключа в закладке **General** диалога **Attributes** необходимо сделать пометку в окне выбора **Primary Key**.

Закладка **Datatype** позволяет выбрать тип атрибута.

Закладка **Definition** позволяет записывать определения отдельных атрибутов. Определения атрибутов можно также сгенерировать как часть схемы (**CREATE COMMENT on entity\_name.attribute\_name**). Закладка **Note** позволяет добавлять замечания об одном или нескольких атрибутах сущности, которые не вошли в определения. Закладка **UDP** служит для задания значений свойств, определяемых пользователем. Предварительно эти свойства должны быть внесены в диалог **User-Defined Property dictionary** как свойства атрибутов.

Для большей наглядности диаграммы каждый атрибут можно связать с иконкой. При помощи списка выбора **Icon** в закладке **General** можно связать иконку с атрибутом.

Очень важно дать атрибуту правильное имя. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Соблюдение этого правила позволяет частично решить проблему нормализации данных уже на этапе определения атрибутов. Согласно синтаксису **IDEF1X** имя атрибута должно быть уникально в рамках модели (а не только в рамках сущности!). Каждый атрибут должен быть определен (закладка **Definition**), при этом следует избегать циклических определений, например, когда термин 1 определяется через термин 2, термин 2 – через термин 3, а термин 3 в свою очередь – через термин 1.

Иногда определение атрибута легче дать через описание области значения. Например, оценка школьника – это число, принимающее значения 2, 3, 4 и 5.

Часто приходится создавать производные атрибуты, то есть атрибуты, значение которых можно вычислять из других атрибутов. Примером производного атрибута может служить **Возраст клиента**, который может быть вычислен из атрибута **Дата рождения клиента**. Такой атрибут может привести к конфликтам; действительно, если вовремя не обновить значение атрибута **Возраст клиента**, он может противоречить значению атрибута **Дата рождения клиента**. Производные атрибуты – ошибка нормализации, однако, их вводят для повышения производительности системы.

### Связи

**Связь** является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой (рис.39).



Рис.39.

Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы, например:

Каждый КЛИЕНТ <вносит/снимает> средства.

Связь показывает, какие именно действия делает клиент. По умолчанию имя связи на диаграмме не показывается. Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт **Relationship Display** и затем включить опцию **verb Phrase**.

На логическом уровне можно установить идентифицирующую связь один-ко-многим, связь многие-ко-многим и неидентифицирующую связь один-ко-многим (соответственно это кнопки слева направо в палитре инструментов).

В **IDEF1X** различают **зависимые** и **независимые сущности**. Тип сущности определяется ее связью с другими сущностями. **Идентифицирующая** связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Когда рисуется идентифицирующая связь, **ERwin** автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами (рис.40).

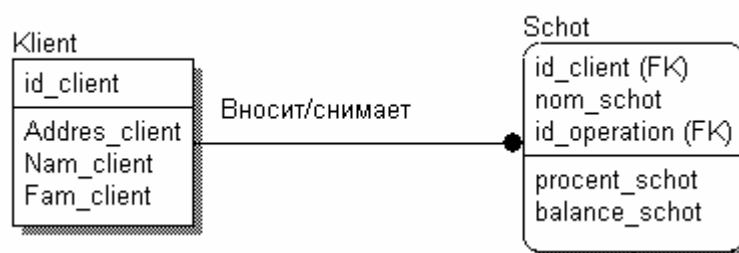


Рис.40.

Экземпляр зависимой сущности определяется только через отношение к родительской сущности. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности



автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ – **(FK)**.

В дальнейшем, при генерации схемы БД, атрибуты первичного ключа получают признак **NOT NULL**, что означает невозможность внесения записи в таблицу заказов без информации о номере клиента.

При установлении **неидентифицирующей** связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь (Рис. 41) служит для связывания независимых сущностей.

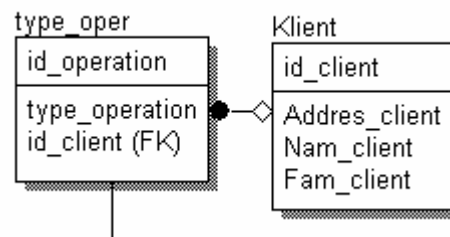


Рис.41.

Идентифицирующая связь показывается на диаграмме сплошной линией с жирной точкой на дочернем конце связи, неидентифицирующая – пунктирной.

Для создания новой связи следует:

- \* установить курсор на нужной кнопке в палитре инструментов (идентифицирующая или неидентифицирующая связь) и нажать левую кнопку мыши;
- \* щелкнуть сначала по родительской, а затем по дочерней сущности.

Для редактирования свойств связи следует “кликнуть” правой кнопкой мыши по связи и выбрать на контекстном меню пункт **Relationship Properties** (рис.42).

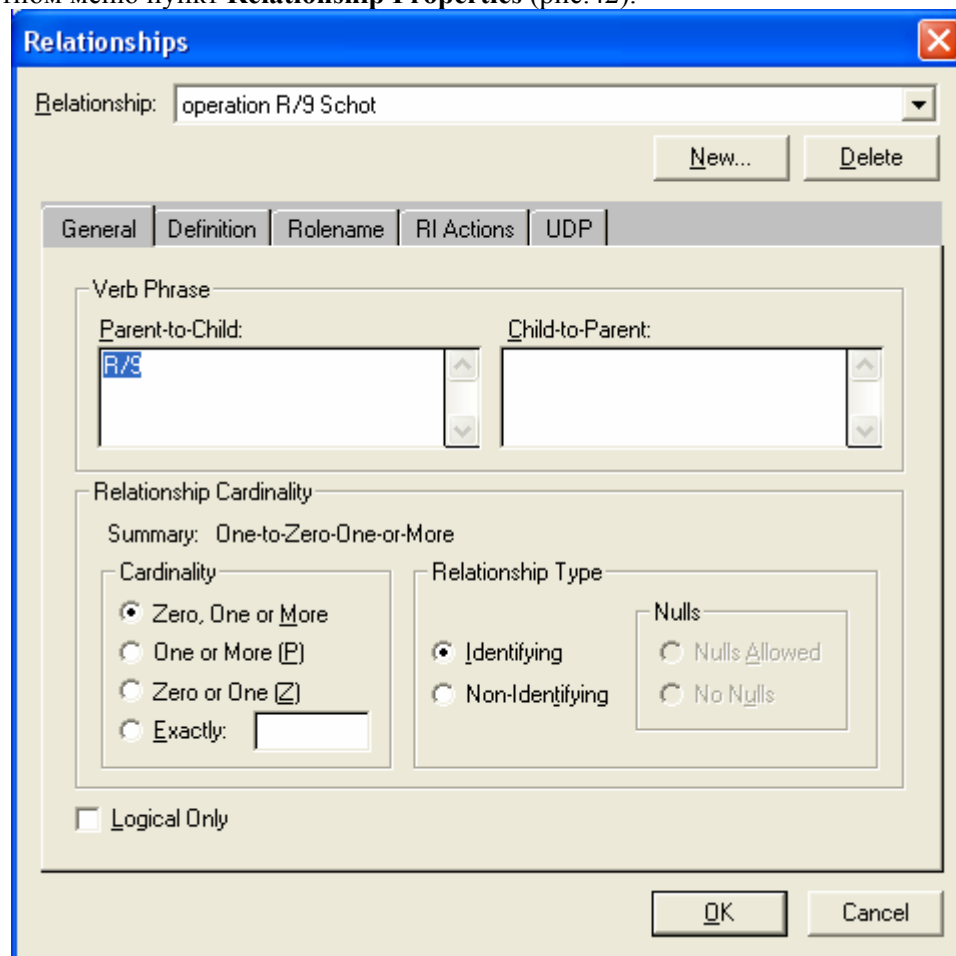


Рис.42.

В закладке **General** появившегося диалога можно задать мощность, имя и тип связи.

**Мощность связи (Cardinality)** – служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней.

Различают четыре типа мощности:

Общий случай, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности, не помечается каким-либо символом;
Символом Р помечается случай, когда одному экземпляру родительской сущности соответствуют 1 или много экземпляров дочерней сущности (исключено нулевое значение);
Символом Z помечается случай, когда одному экземпляру родительской сущности соответствуют 0 или 1 экземпляр дочерней сущности (исключены множественные значения);
Цифрой помечается случай точного соответствия, когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.

По умолчанию символ, обозначающий мощность связи, не показывается на диаграмме. Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт **Relationship Display** и затем включить опцию **Cardinality**.

**Имя связи (Verb Phrase)** – фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи один-ко-многим идентифицирующей или неидентифицирующей достаточно указать имя, характеризующее отношение от родительской к дочерней сущности (**Parent-to-Child**). Для связи многие-ко-многим следует указывать имена как **Parent-to-Child** так и **Child-to-Parent**.

**Тип связи (идентифицирующая/неидентифицирующая).** Для неидентифицирующей связи можно указать обязательность (**Nulls**). В случае обязательной связи (**No Nulls**) при генерации схемы БД атрибут внешнего ключа получит признак **NOT NULL**, несмотря на то, что внешний ключ не войдет в состав первичного ключа дочерней сущности. В случае необязательной связи (**Nulls Allowed**) внешний ключ может принимать значение **NULL**. Необязательная неидентифицирующая связь помечается прозрачным ромбом со стороны родительской сущности. В закладке **Definition** можно дать более полное определение связи для того, чтобы в дальнейшем иметь возможность на него ссылаться.

В закладке **Rolename** можно задать имя роли.

В закладке **RI Actions** правила ссылочной целостности.

**Имя роли (функциональное имя)** – это синоним атрибута внешнего ключа, который показывает, какую роль играет атрибут в дочерней сущности. По умолчанию в списке атрибутов показывается только имя роли. Для отображения полного имени атрибута (как функционального имени, так и имени роли) следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт **Entities Display** и затем включить опцию **Attribute**. Полное имя показывается как функциональное имя и базовое имя, разделенные точкой.

Обязательным является применение имен ролей в том случае, когда два или более атрибутов одной сущности определены по одной и той же области, т. е. Они имеют одинаковую область значений, но разный смысл.

Другим примером обязательности присвоения имен ролей являются **рекурсивные связи** (иногда их называют «рыболовный крючок» – **fish hook**), когда одна и та же сущность является и родительской и дочерней одновременно. При задании рекурсивной связи атрибут должен мигрировать в качестве внешнего ключа в состав неключевых атрибутов той же сущности. Атрибут не может появиться дважды в одной сущности под одним именем, поэтому обязательно должен получить имя роли. Рекурсивная связь может быть только неидентифицирующей.

Вид рекурсивной связи, называющийся **иерархической рекурсией**, задает связь, когда экземпляр родительской сущности может иметь множество экземпляров дочерней сущности, но экземпляр дочерней сущности может иметь только один экземпляр родительской сущности.

Другим видом рекурсии является **сетевая рекурсия**, когда экземпляр родительской сущности может иметь множество экземпляров дочерней сущности и, наоборот, экземпляр дочерней сущности может иметь множество экземпляров родительской сущности. Сетевая рекурсия задает паутину отношений между экземплярами родительской и дочерней сущностей. Это случай, когда сущность находится сама с собой в связи многие-ко-многим. Для разрешения связи многие-ко-многим необходимо создать новую сущность.

Если атрибут мигрирует в качестве внешнего ключа более, чем на один уровень, то на первом уровне отображается полное имя внешнего ключа (имя роли + базовое имя атрибута), на втором и более – только имя роли.

**Правила ссылочной целостности (referential integrity (RI))** – логические конструкции, которые выражают бизнес-правила использования данных и представляют собой правила вставки, замены и удаления. При генерации схемы БД на основе опций логической модели, задаваемых в закладке **Rolename/RI Actions**, будут сгенерированы правила декларативной ссылочной целостности, которые должны быть предписаны для каждой связи, и триггеры, обеспечивающие ссылочную целостность. **Триггеры** представляют собой программы, выполняемые всякий раз при выполнении команд вставки, замены или удаления (**INSERT**, **UPDATE** или **DELETE**).

Правила удаления управляют тем, что будет происходить в БД при удалении строки. Аналогично правила вставки и обновления управляют тем, что будет происходить с БД, если строки изменяются или добавляются.

**Erwin** автоматически присваивает каждой связи значение ссылочной целостности, устанавливаемой по умолчанию, прежде чем добавить ее в диаграмму. Режимы **RI**, присваиваемые **Erwin** по умолчанию, могут быть изменены в редакторе **Triggers**, который вызывается, если щелкнуть по кнопке **Table Triggers...** диалога **RI Triggers** (меню **Database/RI Triggers...**) рис.43.

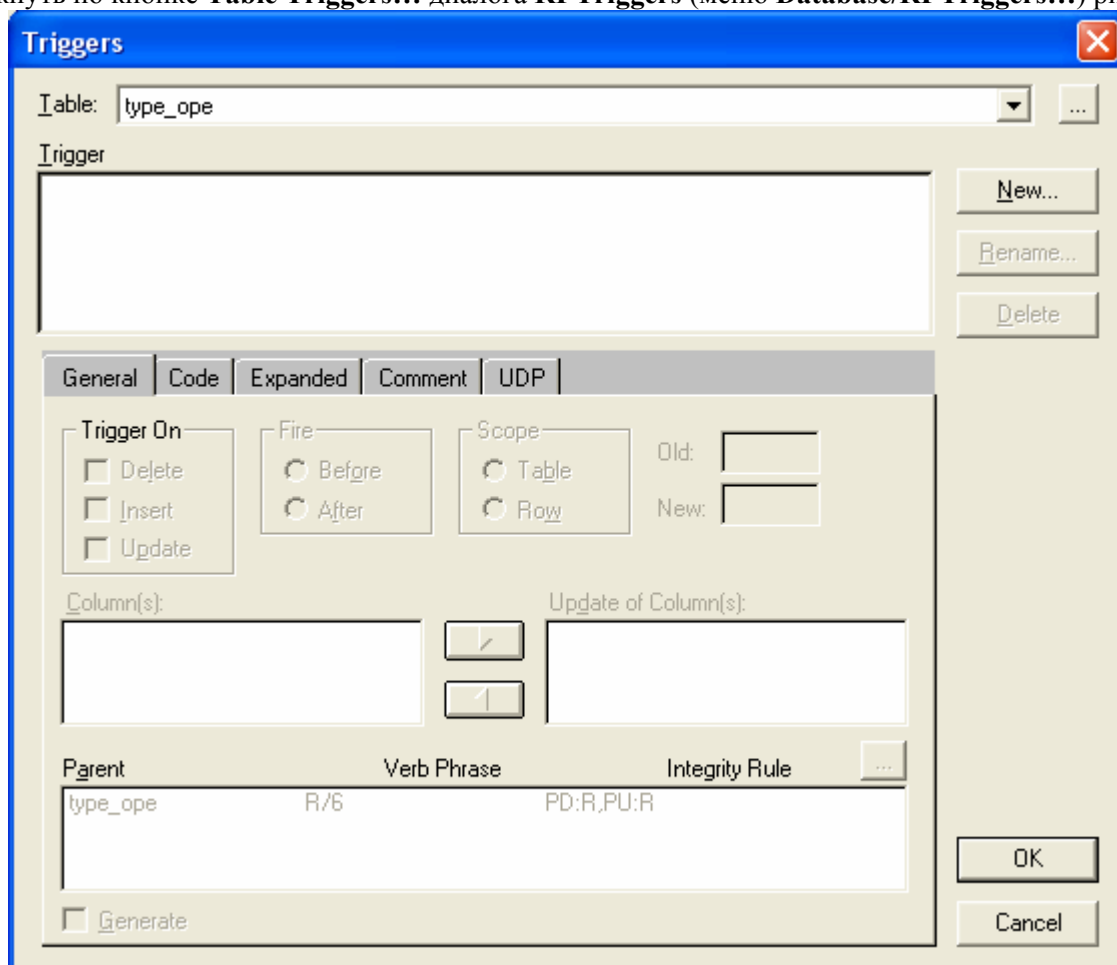


Рис.43.

**Связь многие-ко-многим** возможно только на уровне логической модели данных. Такая связь обозначается сплошной линией с двумя точками на концах (рис.44).

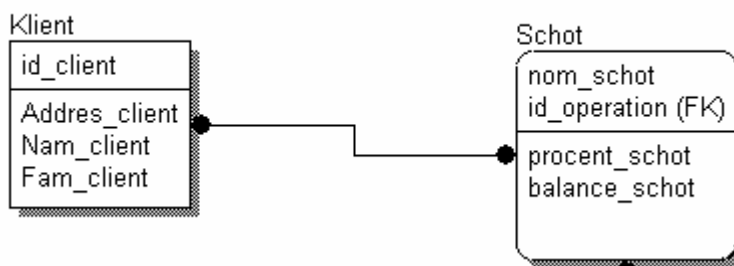


Рис.44.

Для внесения связи следует установить курсор на кнопке с изображением сплошной линии с двумя точками на концах в палитре инструментов, щелкнуть сначала по одной, а затем по другой сущности.

Связь многие-ко-многим должна именоваться двумя фразами – в обе стороны. Это облегчает чтение диаграммы.

При переходе к физическому уровню **ERwin** автоматически преобразует связь многие-ко-многим, добавляя новую таблицу и устанавливая две новые связи один-ко-многим от старых к новой таблице. При этом имя новой таблице автоматически присваивается как «Имя1\_Имя2».

### Типы сущностей и иерархия наследования

Как было указано выше, связи определяют, является ли сущность независимой или зависимой. Различают несколько типов зависимых сущностей:

<b>Характеристическая</b>	– зависимая дочерняя сущность, которая связана только с одной родительской и по смыслу хранит информацию о характеристиках родительской сущности.
<b>Ассоциативная</b>	– сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущностей.
<b>Именующая</b>	– частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрировавших в качестве внешнего ключа).
<b>Категориальная</b>	– дочерняя сущность в иерархии наследования.

**Иерархия наследования** (или **иерархия категорий**) представляет собой особый тип объединения сущностей, которые разделяют общие характеристики.

Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи, либо когда это диктуется бизнес-правилами.

Для каждой категории можно указать **дискриминатор** – атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой.

Иерархии категорий делятся на два типа – **полные** и **неполные**. В полной категории (Рис. 26) одному экземпляру родового предка обязательно соответствует экземпляр в каком-либо потомке.

Если категория еще не выстроена полностью и в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках, то такая категория будет неполной. Полная категория помечается кружком с двумя горизонтальными чертами, неполная – кружком с одной чертой. Возможна комбинация полной и неполной категорий (рис.45).

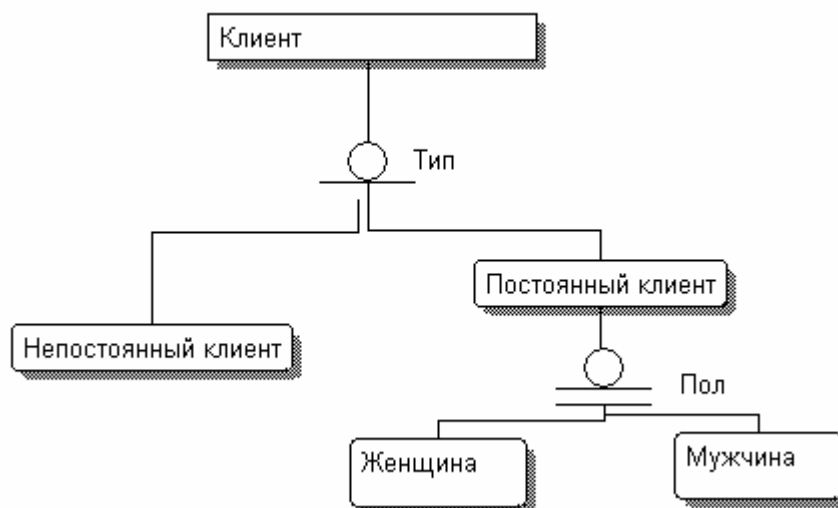


Рис.45.

Для создания категориальной связи следует:

1. установить курсор на кнопке с изображением кружка и двух горизонтальных черт в палитре инструментов и нажать левую кнопку мыши;
2. щелкнуть сначала по родовому предку, а затем по потомку;
3. для установления второй связи в иерархии категории следует сначала щелкнуть по символу категории, затем по второму потомку.

Для редактирования категорий нужно щелкнуть правой кнопкой мыши по символу категории и выбрать в контекстном меню пункт **Subtype Relationship...**. В диалоге **Subtype Relationship** можно указать атрибут – дискриминатор категории (список **Discriminator Attribute Choice**) и тип категории – полная/неполная (радиокнопки **Complete/Incomplete**) рис.46.

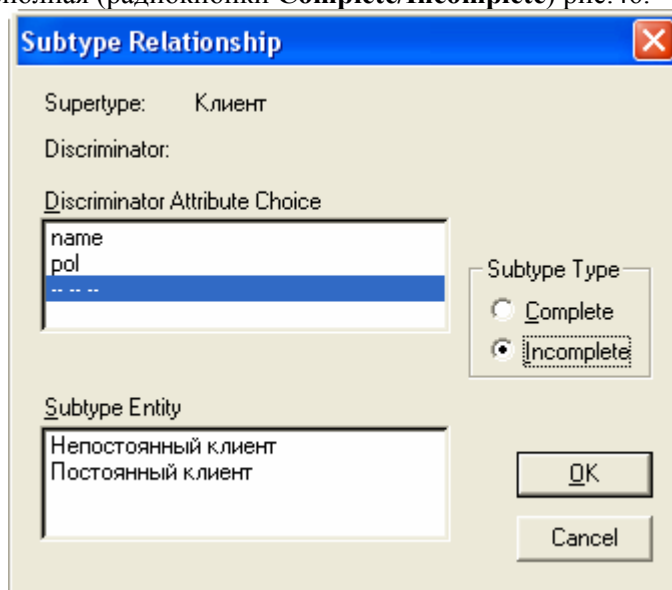


Рис.46.

Рассмотрим возможные стадии построения иерархии наследования.

Определение сущностей с общими (по определению) атрибутами:

<b>Перенос общих атрибутов в сущность – родовой предок.</b>	В случае обнаружения совпадающих по смыслу атрибутов следует создать новую сущность – родового предка и перенести в нее общие атрибуты.
<b>Создание неполной структуры категорий.</b>	Создается категориальная связь от новой сущности – родового предка к старым сущностям – потомкам. Новая сущность дополняется атрибутом-дискриминатором категории.

<b>Создание полной структуры категорий.</b>	Проводится дополнительный поиск сущностей, имеющих общие по смыслу атрибуты с родовым предком. Общие атрибуты переносятся в родового предка и категория преобразуется в полную (признак полной категории устанавливается в диалоге <b>Subtype Relationship</b> ).
<b>Комбинации полной и неполной структур категорий.</b>	При необходимости создание иерархии категорий можно продолжить. Для каждого потомка может найтись сущность с общими атрибутами, тогда сущность – потомок становится родовым предком для новых потомков, и т.д.

## Ключи

Каждый экземпляр должен быть уникален и отличаться от других атрибутов.

**Первичный ключ (primary key)** – это атрибут или группа атрибутов, однозначно идентифицирующая экземпляр сущности. Атрибуты первичного ключа на диаграмме не требуют специального обозначения – это те атрибуты, которые находятся в списке атрибутов выше горизонтальной линии. При внесении нового атрибута в диалоге **Attribute Editor** для того, чтобы сделать его атрибутом первичного ключа, нужно включить флажок Primary Key в нижней части закладки **General**. На диаграмме неключевой атрибут можно внести в состав первичного ключа, воспользовавшись режимом переноса атрибутов (кнопка в палитре инструментов).

Выбор первичного ключа может оказаться непростой задачей, решение которой может повлиять на эффективность будущей ИС. В одной сущности могут оказаться несколько атрибутов или набор атрибутов, претендующих на роль первичного ключа. Такие претенденты называются **потенциальными ключами (candidate key)**.

Ключи могут быть **сложными**, т.е. содержащими несколько атрибутов. Сложные первичные ключи не требуют специального обозначения – это список атрибутов выше горизонтальной линии. Для того, чтобы стать первичным, потенциальный ключ должен удовлетворять ряду требований:

**Уникальность.** Два экземпляра не должны иметь одинаковых значений возможного ключа.

**Компактность.** Сложный возможный ключ не должен содержать ни одного атрибута, удаление которого не приводило бы к утрате уникальности.

При выборе первичного ключа предпочтение должно отдаваться более простым ключам, т.е. ключам, содержащим меньшее количество атрибутов.

Атрибуты ключа не должны содержать нулевых значений. Если для обеспечения уникальности необходимо дополнить потенциальный ключ дополнительными атрибутами, то они не должны содержать нулевых значений.

Значение атрибутов ключа не должно меняться в течение всего времени существования экземпляра сущности.

Каждая сущность должна иметь, по крайней мере, один потенциальный ключ. Многие сущности имеют только один потенциальный ключ. Такой ключ становится первичным. Некоторые сущности могут иметь более одного возможного ключа. Тогда один из них становится первичным, а остальные альтернативными ключами. **Альтернативный ключ (Alternate Key)** – это потенциальный ключ, не ставший первичным. **ERwin** позволяет выделить атрибуты альтернативных ключей, и по умолчанию в дальнейшем при генерации схемы БД по этим атрибутам будет генерироваться уникальный индекс.

При работе ИС часто бывает необходимо обеспечить доступ к нескольким экземплярам сущности, объединенных каким-либо одним признаком. Для повышения производительности в этом случае используются неуникальные индексы. **ERwin** позволяет на уровне логической модели назначить атрибуты, которые будут участвовать в неуникальных индексах. Атрибуты, участвующие в неуникальных индексах, называются **Inversion Entries (инверсионные входы)**. **Inversion Entry** – это атрибут или группа атрибутов, которые не определяют экземпляр сущности уникальным образом, но часто используются для обращения к экземплярам сущности. **ERwin** генерирует неуникальный индекс для каждого **Inversion Entry**.

Создать альтернативные ключи и инверсионные входы можно в закладке **Key Group** диалога **Attribute Editor**. Если щелкнуть по кнопке со знаком (...), расположенной в правой верхней части закладки, вызывается диалог **Key Group Editor**. В верхней части диалога находится список ключей, в нижней – список атрибутов, доступных для включения в состав ключа (слева), и список ключевых атрибутов. Каждый вновь созданный ключ должен иметь хотя бы один атрибут. Для включения атрибута в состав ключа следует выделить его в левом списке и щелкнуть по кнопке со стрелкой.

Для создания нового ключа следует щелкнуть по кнопке **New....** Появляется диалог **New Key Group**. Имя нового ключа присваивается автоматически (**“Alternate Key N”** для альтернативного ключа и **“Inversion Entry N”** для инверсионного входа, где N – порядковый номер ключа).

Каждому ключу соответствует индекс, имя которого также присваивается автоматически (**“XAKNENTITY”** для альтернативного ключа и **“XIENENTITY”** для инверсионного входа, где N – порядковый номер ключа, ENTITY – имя сущности). Имена ключа и индекса при желании можно изменить вручную (рис.48).

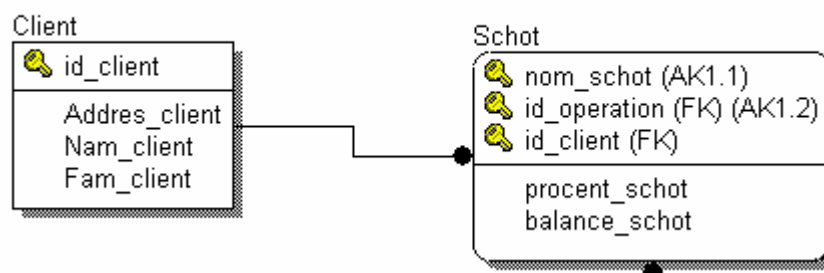


Рис.48.

На диаграмме (Рис. №) атрибуты альтернативных ключей обозначаются как **(FK)**. Когда альтернативный ключ содержит несколько атрибутов, **(AKn.m)** ставится после каждого. На Рис. Атрибуты **nom\_schot** – основной ключ и входит в альтернативный ключ 1, **id\_operation** - альтернативный ключ №1 и внешний ключ, **id\_client** - внешний ключ №2. Если один атрибут входит в состав нескольких ключей, ключи перечисляются в скобках через запятую (атрибут **id\_operation** входит в состав AK1 и FK1). По умолчанию номера альтернативных ключей рядом с именем атрибута на диаграмме не показываются. Для отображения номера следует в контекстном меню, которое появляется, если щелкнуть правой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт контекстного меню **Entity Display** и затем включить опцию **Alternate Key Designator (AK)**.

**Внешние ключи (Foreign Key)** создаются автоматически, когда связь соединяет сущности: связь образует ссылку на атрибуты первичного ключа в дочерней сущности, и эти атрибуты образуют внешний ключ в дочерней сущности (миграция ключа). Атрибуты внешнего ключа обозначаются символом **(FK)** после своего имени.

Зависимая сущность может иметь один и тот же внешний ключ из нескольких родительских сущностей. Сущность может также получить один и тот же внешний ключ несколько раз от одного и того же родителя через несколько разных связей. Когда **ERwin** обнаруживает одно из этих событий, он распознает, что два атрибута одинаковы, и помещает атрибут внешнего ключа в зависимой сущности только один раз. Хотя в закладке **Key Group** диалога **Attribute** этот атрибут будет входить в два внешних ключа, на диаграмме он показывается только один раз. Это комбинирование или объединение идентичных атрибутов называется **унификацией**.

Унификация производится, поскольку правила нормализации запрещают существование в одной сущности двух атрибутов с одинаковыми именами.

Когда унификация нежелательна (например, когда два атрибута имеют одинаковые имена, но на самом деле они отличаются по смыслу и необходимо, чтобы это отличие отражалось в диаграмме), нужно использовать имена ролей атрибутов внешнего ключа.

## Нормализация данных

**Нормализация** – процесс проверки и реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных. Нормализация позволяет быть уверенным, что каждый атрибут определен для своей сущности, значительно сократить объем памяти для хранения информации и устранить аномалии в организации хранения данных. В

результате проведения нормализации должна быть создана структура данных, при которой информация о каждом факте хранится только в одном месте. Процесс нормализации сводится к последовательному приведению структуры данных к **нормальным формам** – формализованным требованиям к организации данных [1]. Известны шесть нормальных форм:

- \* первая нормальная форма (1NF);
- \* вторая нормальная форма (2NF);
- \* третья нормальная форма (3NF);
- \* нормальная форма Бойса-Кодда (усиленная 3NF);
- \* четвертая нормальная форма (4NF);
- \* пятая нормальная форма (5NF).

На практике обычно ограничиваются приведением данных к третьей нормальной форме (полная атрибутивная модель, 3NF).

Нормальные формы основаны на понятии функциональной зависимости (в дальнейшем будет использоваться термин «зависимость»).

**Функциональная зависимость (FD).** Атрибут В сущности Е функционально зависит от атрибута А сущности Е тогда и только тогда, когда каждое значение А и Е связало с ним точно одно значение В и Е, т.е. А однозначно определяет В.

**Полная функциональная зависимость.** Атрибут В сущности Е полностью функционально зависит от ряда атрибутов А сущности Е тогда и только тогда, когда В функционально зависит от А и не зависит ни от какого подряда А. Функциональные зависимости определяются бизнес правилами предметной области.

**Первая нормальная форма (1NF).** Сущность находится в первой нормальной форме тогда и только тогда, когда все атрибуты содержат атомарные значения. Среди атрибутов не должно встречаться повторяющихся групп, т.е. несколько значений для каждого экземпляра. Другой ошибкой нормализации является хранение в одном атрибуте разных по смыслу значений.

Для приведения сущности к первой нормальной форме следует:

- \* разделить сложные атрибуты на атомарные;
- \* создать новую сущность;
- \* перенести в нее все «повторяющиеся» атрибуты;
- \* выбрать возможный ключ для нового РК (или создать новый РК);
- \* установить идентифицирующую связь от прежней сущности к новой, РК прежней сущности станет внешним ключом (FK) для новой сущности.

**Вторая нормальная форма (2NF).** Сущность находится во второй нормальной форме, если она находится в первой нормальной форме, и каждый неключевой атрибут полностью зависит от первичного ключа (не должно быть зависимости от части ключа). Вторая нормальная форма имеет смысл только для сущностей, имеющих сложный первичный ключ.

Для приведения сущности ко второй нормальной форме следует:

- \* выделить атрибуты, которые зависят только от части первичного ключа, создать новую сущность;
- \* поместить атрибуты, зависящие от части ключа, в их собственную (новую) сущность;
- \* установить идентифицирующую связь от прежней сущности к новой.

**Третья нормальная форма (3NF).** Сущность находится в третьей нормальной форме, если она находится во второй нормальной форме и никакой неключевой атрибут не зависит от другого неключевого атрибута (не должно быть взаимозависимости между неключевыми атрибутами).

Для приведения сущности к третьей нормальной форме следует:

- \* создать новую сущность и перенести в нее атрибуты с одной и той же зависимостью от неключевого атрибута;
- \* использовать атрибут(ы), определяющий эту зависимость, в качестве первичного ключа новой сущности;
- \* установить неидентифицирующую связь от новой сущности к старой.

В третьей нормальной форме каждый атрибут сущности зависит от ключа, от всего ключа целиком и ни от чего другого, кроме как от ключа.

**Домены**



Домены можно определить как совокупность значений, из которых берутся значения атрибутов. Каждый атрибут может быть определен только на одном домене, но на каждом домене может быть определено множество атрибутов. В понятие домена входит не только тип данных, но и область значений данных.

В **ERwin** домен может быть определен только один раз и использоваться как в логической, так и в физической модели.

Для создания домена в логической модели служит диалог **Domain Dictionary** (рис.49).

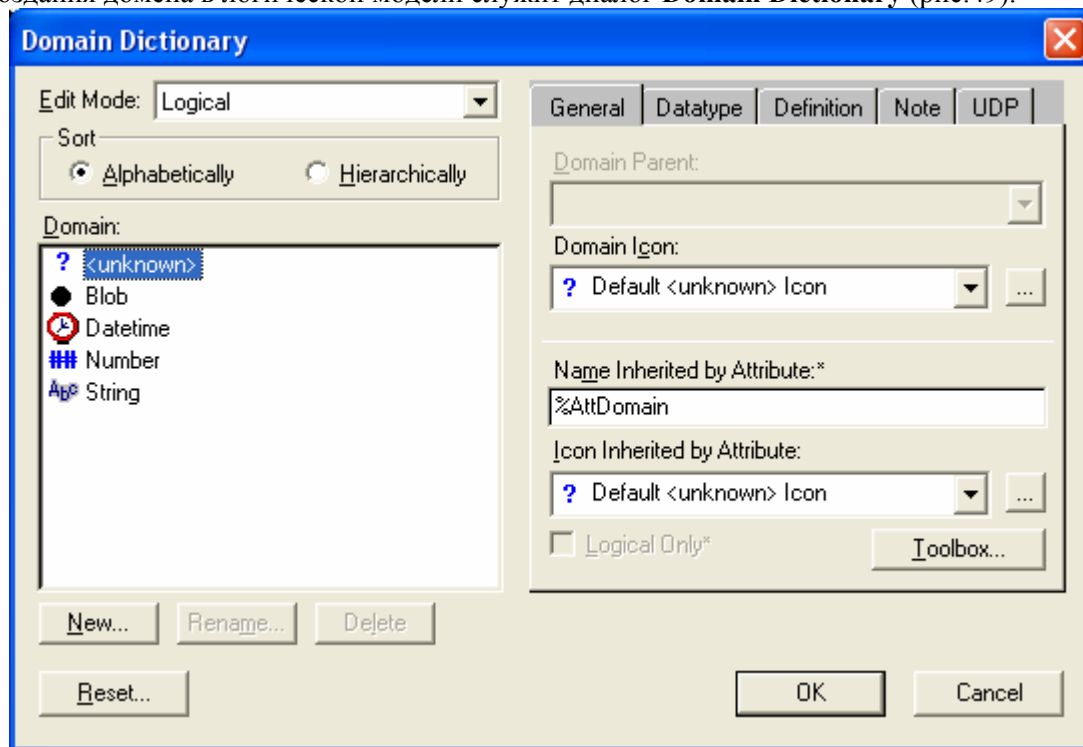


Рис.49.

Его можно вызвать из меню **Model/Domain Dictionary** либо нажав на кнопку, расположенную в верхней левой части закладки **General** диалога **Attribute**. Для создания нового домена в диалоге **Domain Dictionary** следует:

\* щелкнуть по кнопке **New**. Появляется диалог **New Domain** (рис.50);

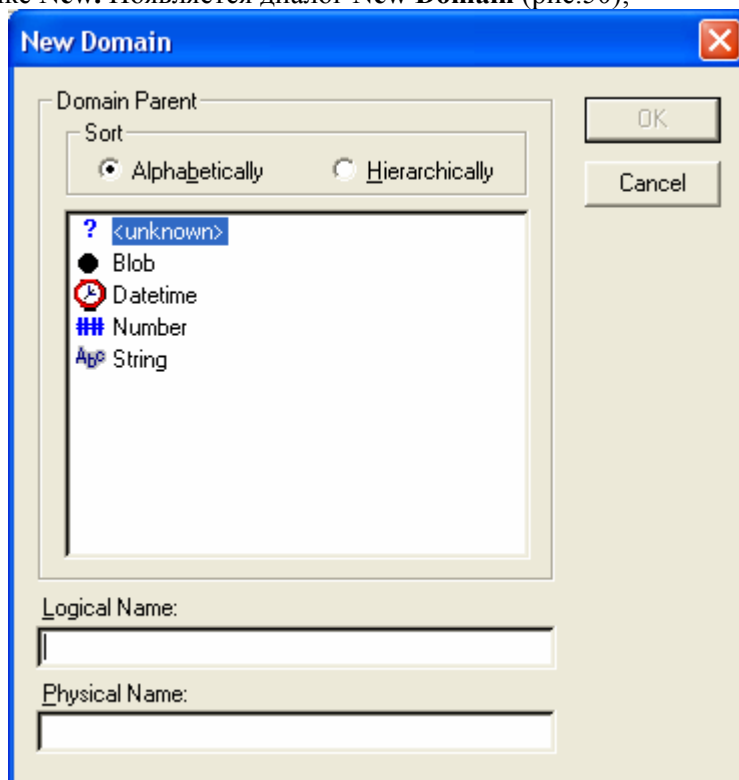


Рис.50.

- \* выбрать родительский домен из списка **Domain Parent**. Новый домен можно создать на основе уже созданного пользователем домена либо на основе изначально существующего. По умолчанию **ERwin** имеет четыре predefined домена (**String, Number, Blob, Datetime**). Новый домен наследует все свойства родительского домена. Эти свойства в дальнейшем можно переопределить;
- \* набрать имя домена в поле **Logical Name**. Можно также указать имя домена на физическом уровне в поле **Physical Name**. Если физическое имя не указано, по умолчанию оно принимает значение логического имени;
- \* щелкнуть по кнопке ОК.

В диалоге **Domain Dictionary** можно связать домен с иконкой, с которой он будет отображаться в списке доменов (**Domain Icon**), и иконкой, с которой атрибут, определенный на домене, будет отображаться в модели (**icon Inherited by Attribute**).

Домены расположены в окне браузера Model Explorer ErWin (рис.51).

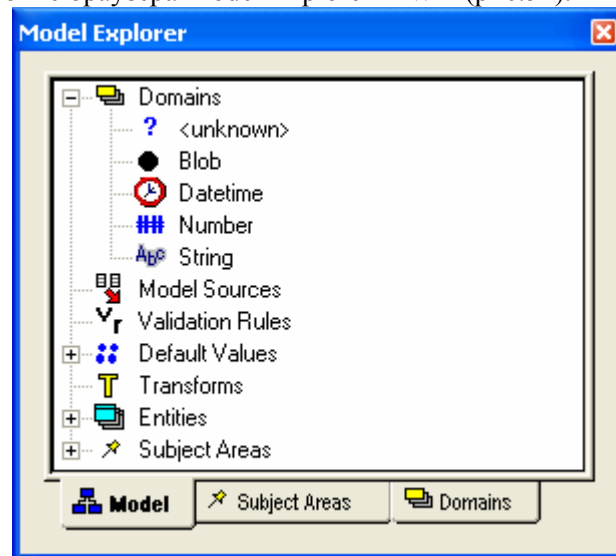


Рис.51.

С его помощью можно выбрать в списке домен и по методу **drag&drop** перенести его в какую-либо сущность. В ней будет создан новый атрибут с именем, которое следует задать в окне **Name Inherited by Attribute** диалога **Domain Dictionary**. Если значение поля не задано, по умолчанию принимается имя домена.

### Пример.

Рассмотрим нашу задачу про банк автомобилистов.

Для определения инфологической модели и определения структуры будущей БД воспользуемся DFD-моделью, созданной в программе BpWin в предыдущей лабораторной работе.

Для этого необходимо доопределить перечень таблиц и их полей будущей БД в BpWin.

Для доопределения перечня таблиц в модели DFD разработанной в Bpwin необходимо:

1. Открыть проект в Bpwin.
2. Выбрать в проекте DFD диаграмму
3. Открыть вкладку Entity and Attribute Dictionary Editor в меню **Model -> Entity/Attribute Editor** (рис.52).

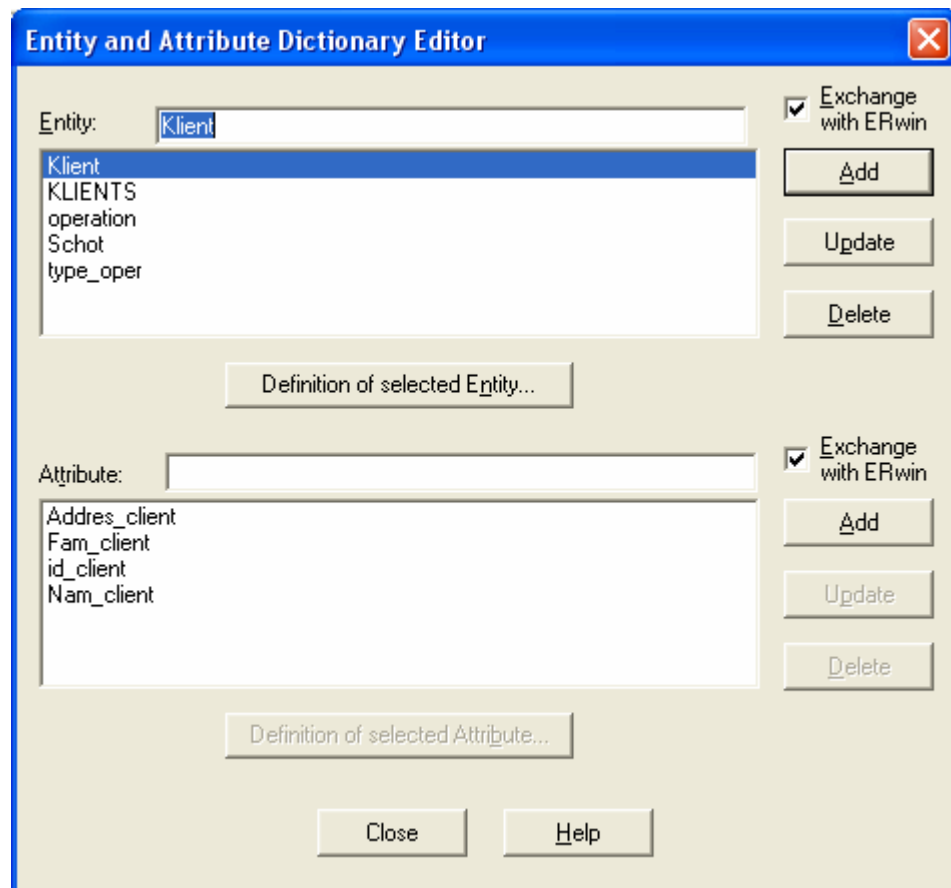


Рис.52.

После определения таблиц и их структур сделали экспорт из BPWin в ErWin. Для экспорта из BPWin в ErWin необходимо в меню File выбрать пункт экспорт в ErWin (рис.53).

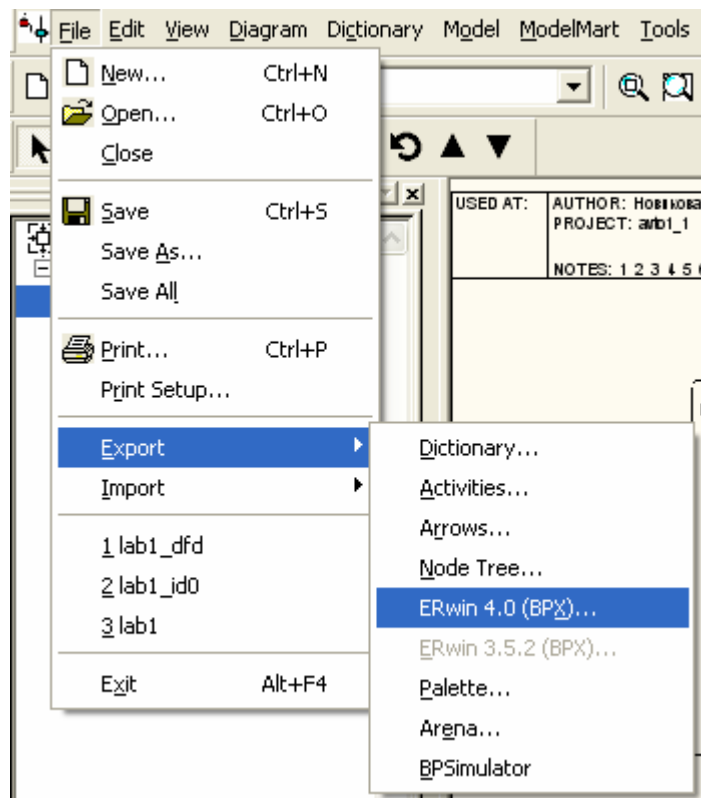


Рис.53.

Далее создаем новый проект в ErWin и производим Import из экспортированных файлов (рис.54-55).

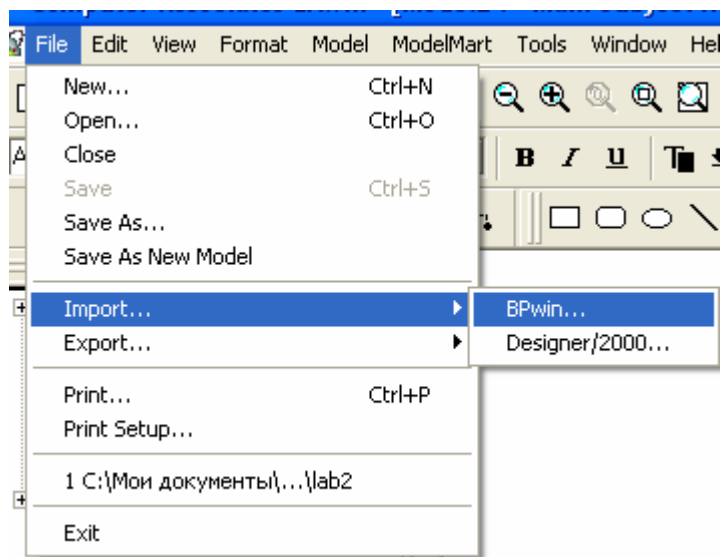


Рис.54.

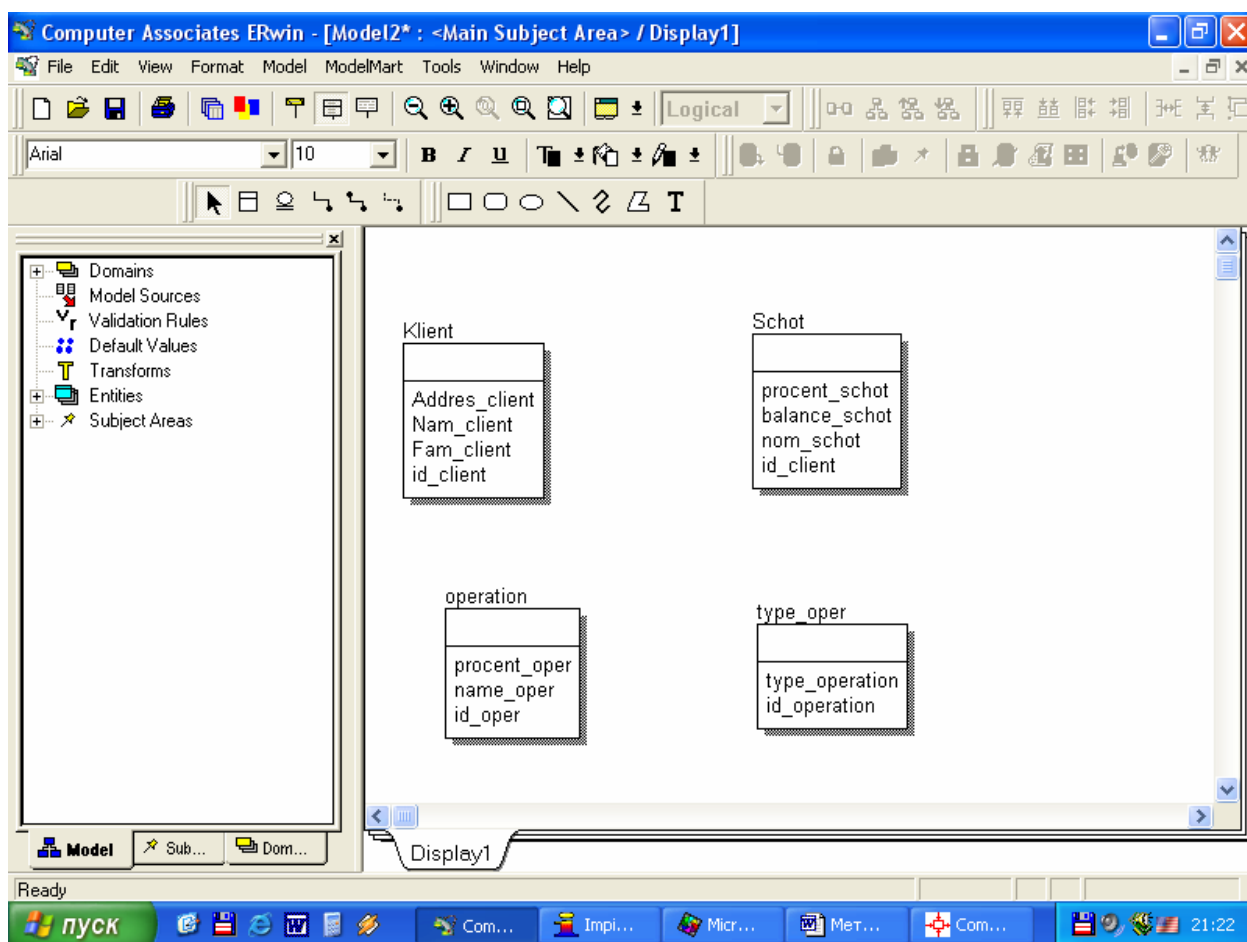


Рис.55.

Укажем типы полей каждой таблицы. Для этого нажмем правой клавишей мыши на представлении таблицы и выберем пункт Attributes (Рис.56):

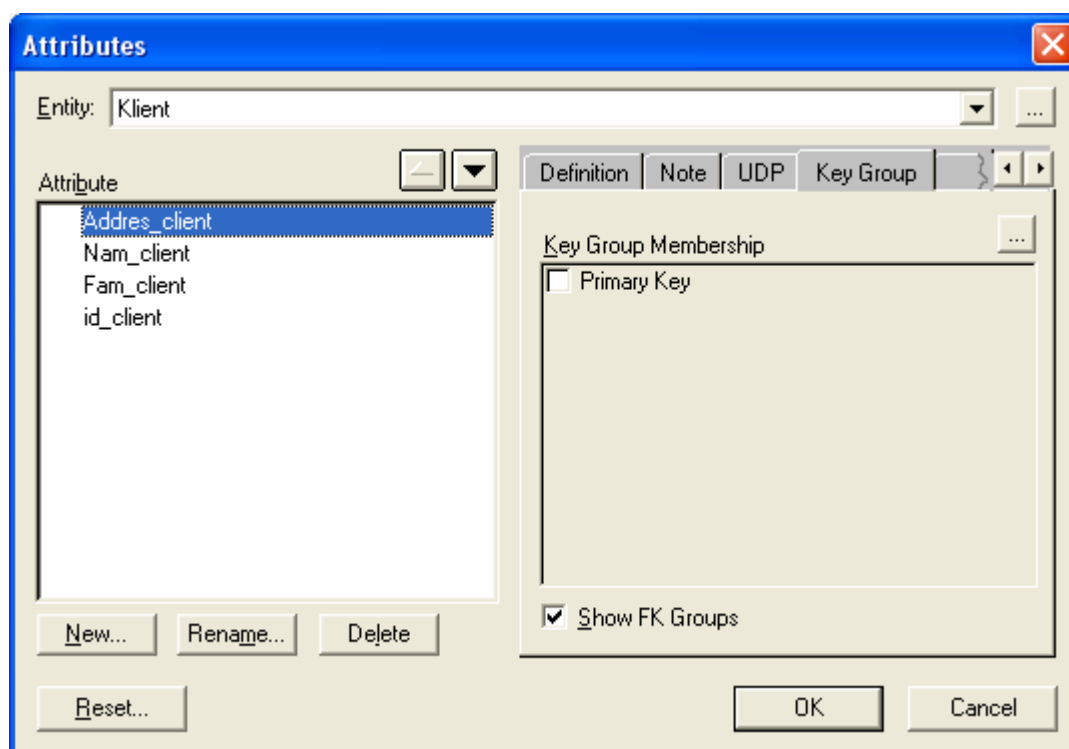


Рис.56.

После определения типов полей необходимо создать ключевые поля и определить связи между таблицами.

Ключевые поля создаются в окне «Attributes» на вкладке «Key Group» рис.57.

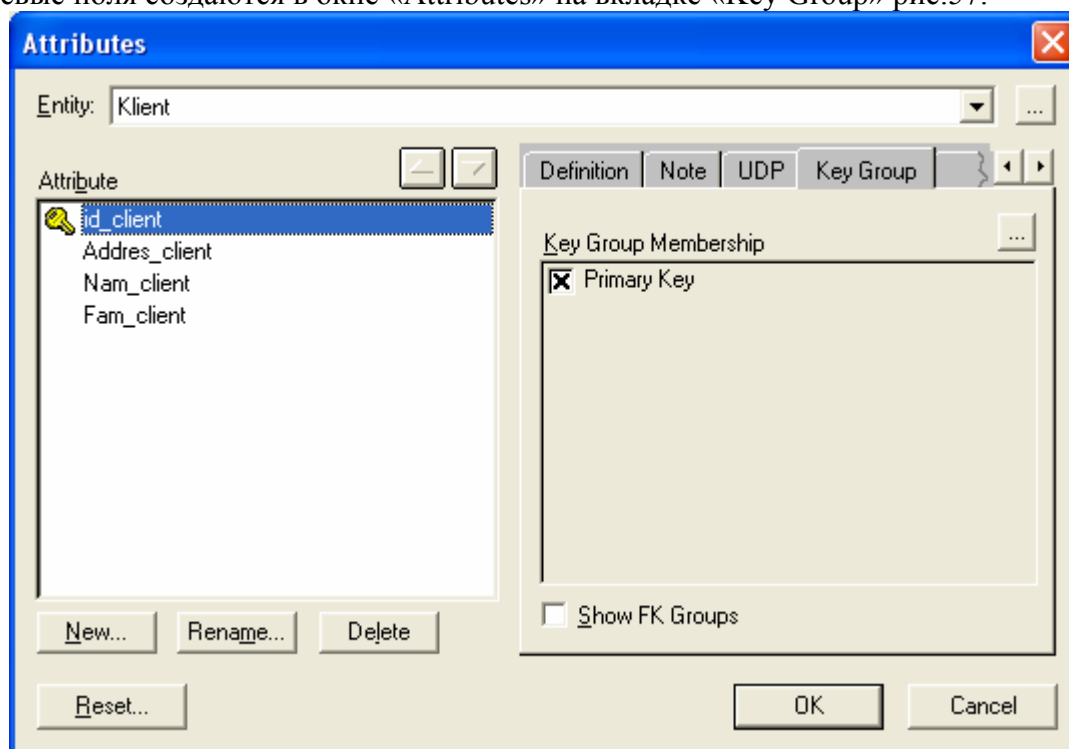


Рис.57.

Для создания связи одна-ко-многим необходимо выбрать «Identifying relationship» и соединить две таблицы. Остальные типы связей расположены там же. Если необходимо добавить какую либо сущность таблицы, то на этой же панели расположен элемент «Entity» рис.58.

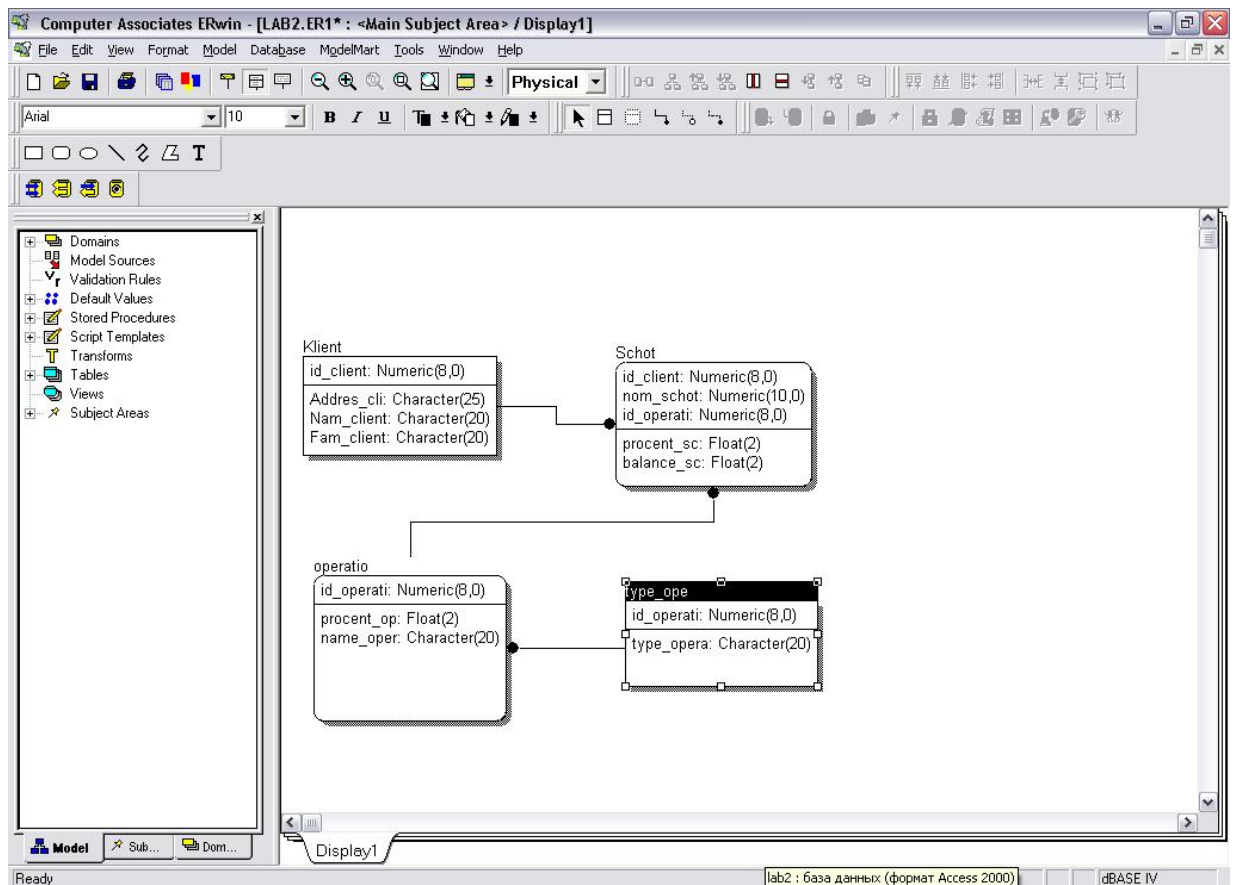


Рис.58.

После определения таблиц создали БД в Microsoft Access средствами ErWin на основе инфологической модели.

Для создания таблиц в БД на основе модели разработанной в ErWin необходимо:

1. Создать пустую БД
2. В ErWin выбрать пункт меню (рис.59)

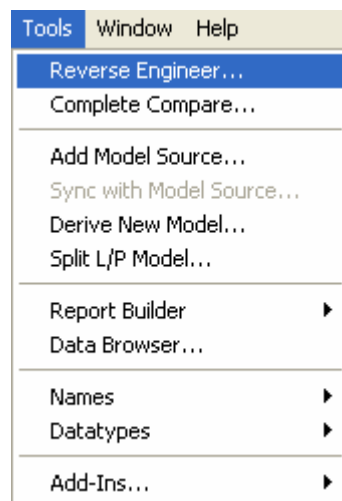


Рис.59.

3. В открывшемся окне необходимо указать тип СУБД, на котором расположена БД (рис.60).

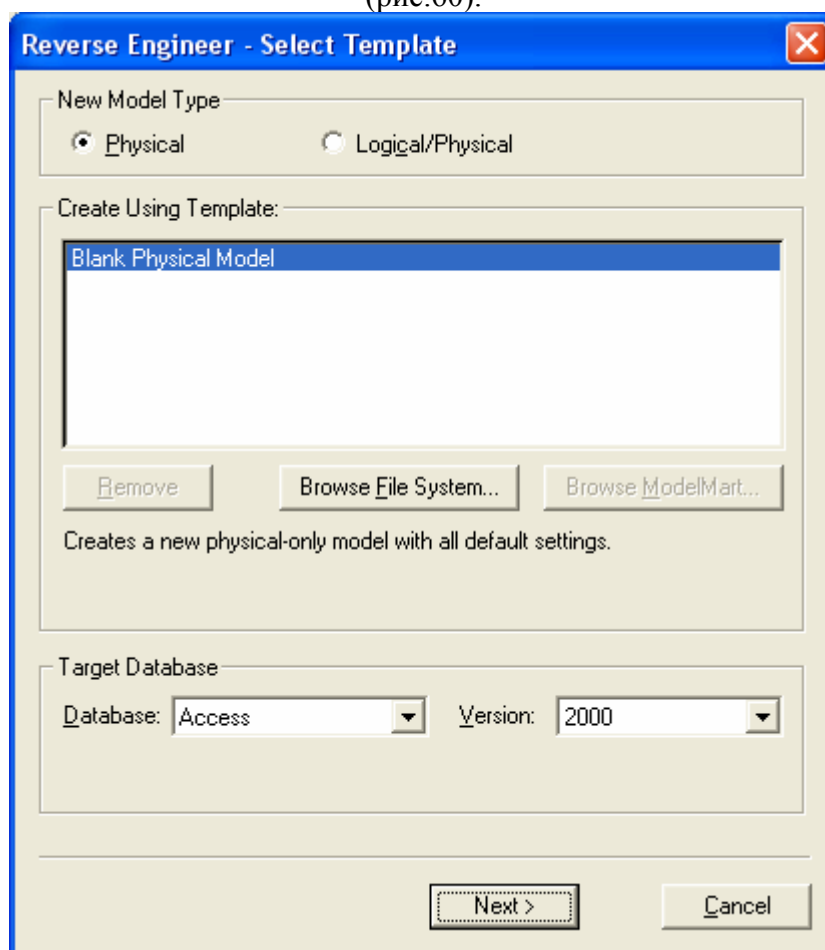


Рис.60.

4. После нажатия кнопки Next откроется окно выбора параметров создания или проверки структуры БД на сервере (рис.61).

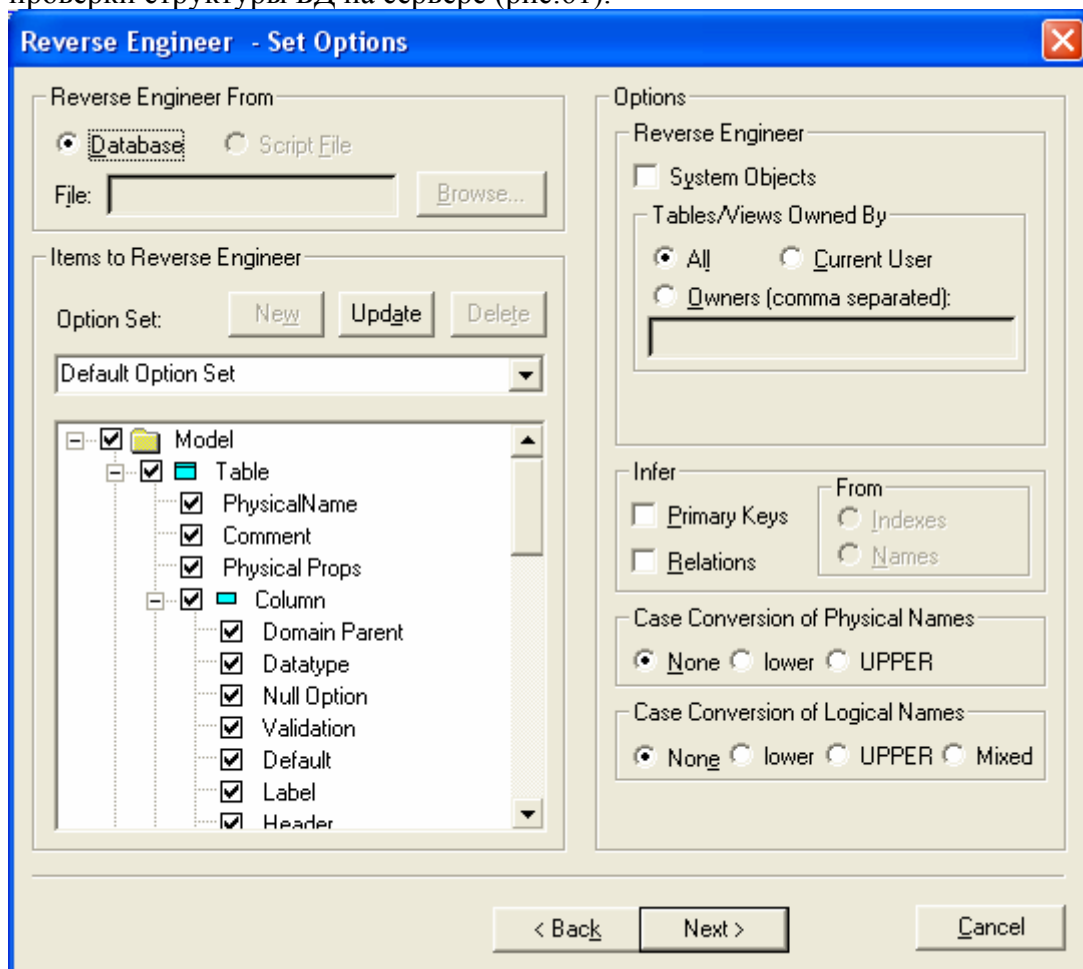


Рис.61.

5. После нажатия кнопки Next откроется окно ввода параметров соединения с сервером БД (рис.62).

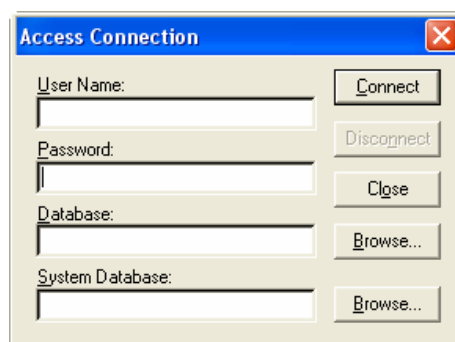


Рис.62.



В результате выполнения вышеописанных действий ErWin создал следующую структуру таблиц в БД Access (рис.63):

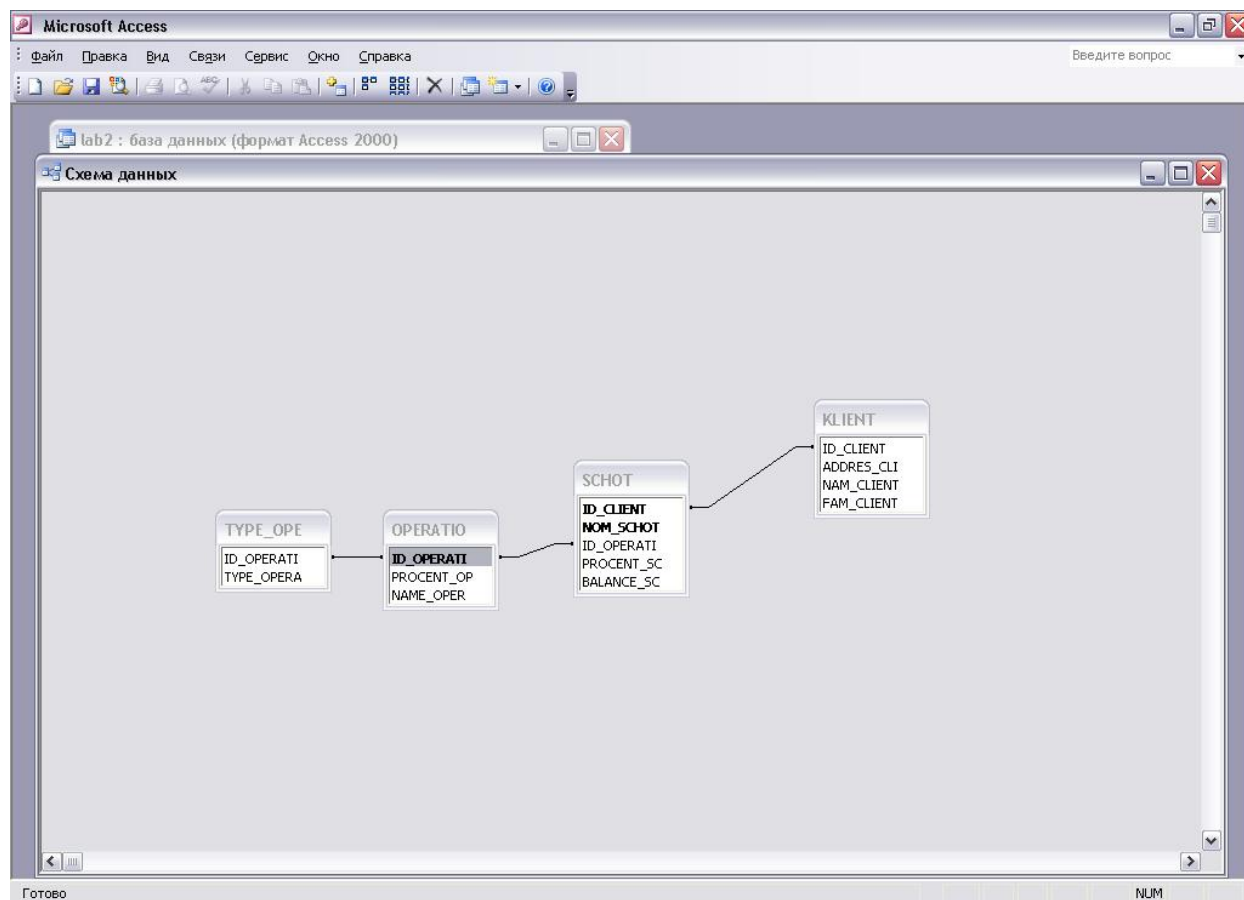


Рис.63.

### **Список литературы:**

1. С. В. Маклаков "ERwin и BPwin. CASE-средства разработки информационных систем" Москва «Диалг-МИФИ» 2001.
2. А. Вендров "CASE-технологии. Современные методы и средства проектирования информационных систем".
3. Программное обеспечение [www.interface.ru](http://www.interface.ru)
4. Описание стандартов семейства IDEF [www.idef.org](http://www.idef.org)