*Занятие № 9*

*Номер учебной группы:* П-16

*Фамилия, инициалы учащегося:* Язубец Е.В.

*Дата выполнения работы:* 21.11.2022

*Тема работы:* «Разработка модели «сущность-связь» в нотации Чена с использованием современных CASE технологий»

*Результат выполнения работы*

**1.Программа Dia**

**Назначение Dia** - создание схем и диаграмм.

Схема, диаграмма может располагаться на одном или нескольких листах.

Dia включает ряд стандартных векторных графических форм и объектов линии. При создании схем можно использовать такие элементы, как прямоугольник, эллипс, многоугольник, безье-угольник, линии, дуги, зигзаг, ломаные линии, кривые безье, эскиз, изображение, текст.

Имеются базовые объекты и специальные.

Объекты можно выделять. перемещать, изменять их размеры, менять цвет, связывать (используя линии и стрелки), копировать и вставлять.

**При двойном щелчке по объекту** - просмотр свойств объекта и его редактирование. Например, для прямоугольника можно определить толщину линий, цвет линий, цвет заполнения, стиль линии (например - штриховая), радиус закругления углов и др.

При вставке текста настраивается шрифт и выравнивание.

Для стандартных форм Вы можете настроить прозрачность, угловую форму и формат изображения.

**Применяются слои.**

При сохранении проекта для дальнейшего редактирования нужно добавить к названию файла расширение '.dia'.

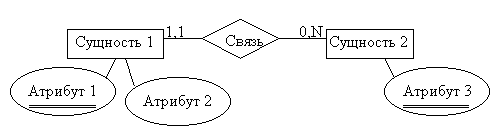
Можно загрузить и сохранить созданную схему или диаграмму в формате XML, экспорт в форматах EPS, SVG, XFIG, WMF и PNG, можно напечатать диаграммы (включая многократные страницы).

**2. Элементы используемые при создании модели сущность-связь в нотации Чена.**

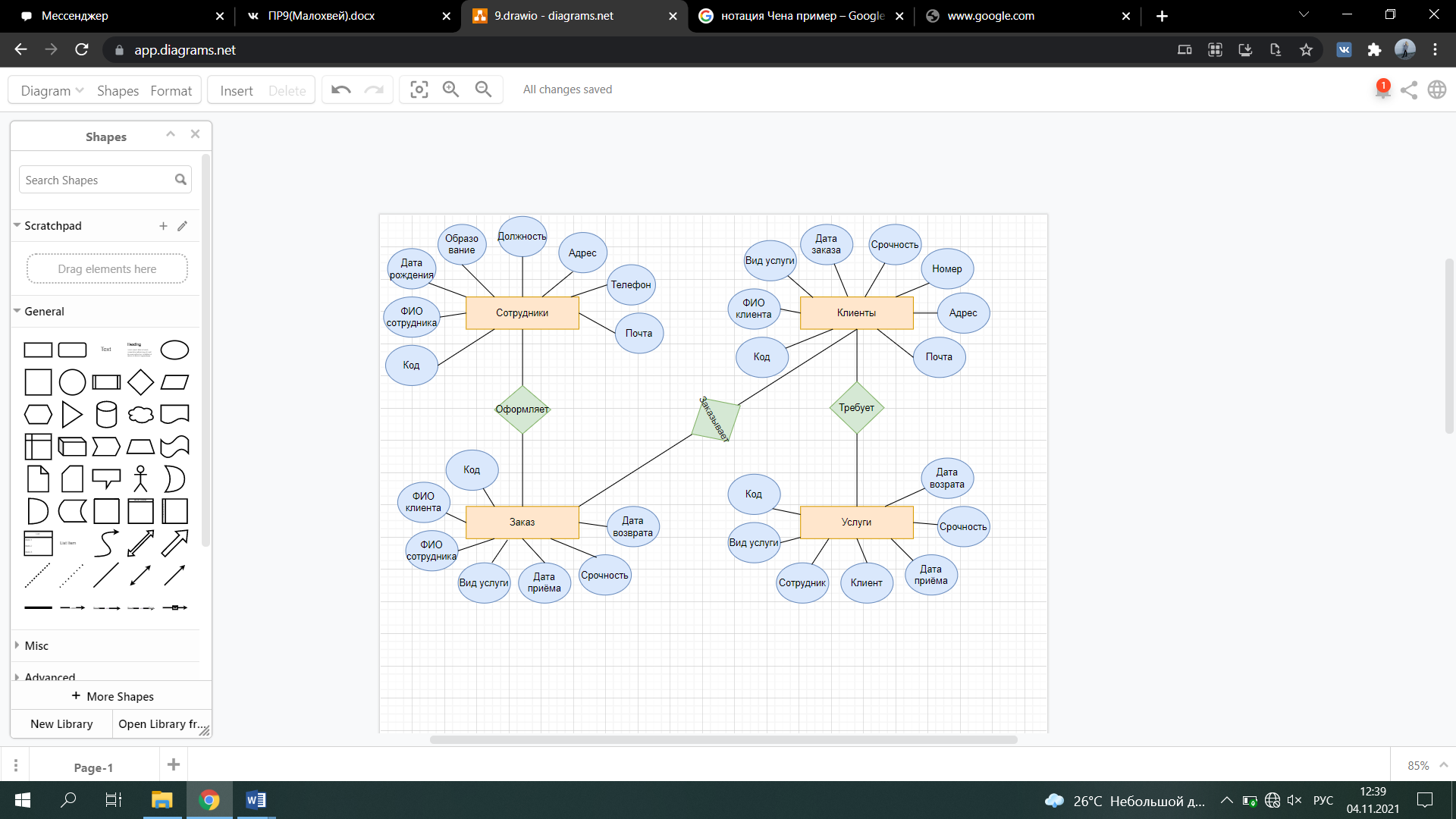
|  |  |
| --- | --- |
| **Элемент диаграммы** | **Обозначает** |
| http://www.mstu.edu.ru/study/materials/zelenkov/image144.gif | независимая сущность |
| http://www.mstu.edu.ru/study/materials/zelenkov/image41.gif | зависимая сущность |
| http://www.mstu.edu.ru/study/materials/zelenkov/image42.gif | родительская сущность в иерархической связи |
| http://www.mstu.edu.ru/study/materials/zelenkov/image145.gif | Связь |
| http://www.mstu.edu.ru/study/materials/zelenkov/image146.gif | идентифицирующая связь |
| http://www.mstu.edu.ru/study/materials/zelenkov/image123.gif | Атрибут |
| http://www.mstu.edu.ru/study/materials/zelenkov/image124.gif | первичный ключ |
| http://www.mstu.edu.ru/study/materials/zelenkov/image125.gif | внешний ключ (понятие внешнего ключа вводится в реляционной модели данных) |
| http://www.mstu.edu.ru/study/materials/zelenkov/image126.gif | многозначный атрибут |
| http://www.mstu.edu.ru/study/materials/zelenkov/image127.gif | получаемый (наследуемый) атрибут в иерархических связях |

Связь соединяется с ассоциируемыми сущностями линиями. Возле каждой сущности на линии, соединяющей ее со связью, цифрами указывается класс принадлежности**.**

**Пример:**



**3. модель «сущность-связь» в нотации Чена с использованием конструктора Dia.**



**4.Ответы на контрольные вопросы.**

1. Логический и физический уровни модели данных в ERwin.

**Ответ:** 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, "Постоянный клиент", "Отдел" или "Фамилия сотрудника". Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами (подробнее о сущностях и атрибутах будет рассказано ниже). **Логическая модель** данных может быть построена на основе другой логической модели, например, на основе модели процессов (см. гл. 1). 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 (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей. Если в логической модели не имеет значения, какой конкретно тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах - таблицах, колонках, индексах, процедурах и т. д. Разделение модели данных на логические и физические позволяет решить несколько важных задач.

2. Последовательность создания логической модели данных в ERwin.

**Ответ:** 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).

3. Последовательность создания физической модели данных в ERwin.

**Ответ:** 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. Например, можно перенести структуру данных с Oracle на Informix (или наоборот) или перенести структуру dbf-файлов в реляционную СУБД, тем самым облегчив решение по переходу от файл-серверной к клиент-серверной ИС. Заметим, однако, что формальный перенос структуры "плоских" таблиц на реляционную СУБД обычно неэффективен. Для того чтобы извлечь выгоды от перехода на клиент-серверную технологию, структуру данных следует модифицировать.

4. Альтернативные ключи.

**Ответ:** **Альтернативный ключ (Alternate Key)** - это потенциальный ключ, не ставший первичным. 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** - это атрибут (или группа атрибутов), несовпадающий с первичным ключом и уникально идентифицирующий экземпляр сущности. Атрибуты, составляющие альтернативный ключ, однозначно (уникально) идентифицируют экземпляры сущности. Например, для сущности служащий (идентификатор служащего, фамилия, имя, отчество) группа атрибутов «фамилия», «имя», «отчество» может являться альтернативным ключом (в предположении, что на предприятии не работают полные тезки). С помощью альтернативных ключей часто отображают различные индексы доступа к данным в конечной реализации реляционной базы. Одни и те же атрибуты сущности могут входить в несколько различных групп ключей.

5. Инвертированные индексы.

**Ответ:** **Инвертированный индекс** (также упоминается в качестве файла сообщений или инвертированного файла) - это индекс базы данных , в котором хранится отображение содержимого, такого как слова или числа, в его местоположения в таблица , либо в документе, либо в наборе документов (названном в отличие от прямого индекса , который сопоставляет документы с содержанием). Целью инвертированного индекса является обеспечение быстрого полнотекстового поиска за счет увеличения объема обработки при добавлении документа в базу данных. Инвертированный файл может быть самим файлом базы данных, а не ее индексом. Это самая популярная структура данных, используемая в системах поиска документов, широко используемых, например, в поисковых системах. Кроме того, несколько важных систем управления базами данных на базе мэйнфреймов общего назначения использовали архитектуры инвертированных списков, включая ADABAS, DATACOM / DB, и Модель 204.

Существует два основных варианта инвертированных индексов: A инвертированный индекс уровня записи (или инвертированный файловый индекс или просто инвертированный файл) содержит список ссылок на документы по каждому слову. Инвертированный индекс уровня слова (или полный инвертированный индекс или инвертированный список) дополнительно содержит позиции каждого слова в документе. Последняя форма предлагает больше функциональных возможностей (например, поиск фраз), но требует большей вычислительной мощности и места для создания.