НАЗНАЧЕНИЕ ТЕХНОЛОГИИ БАЗ ДАННЫХ. МЕТОДОЛОГИЯ ИНФОРМАЦИОННОГО МОДЕЛИРОВАНИЯ IDEF1X

Базы данных. Тема 1.

Бондаренко Виталий Иванович.
http://donnu.ru/phys/kt/bondarenko
https://vk.com/donnu_kkt_db

План

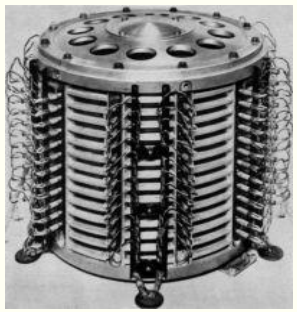
- Информационные системы и файловые системы
- Потребности информационных систем
- Понятие модели предметной области
- Основные понятия: сущность, атрибут, отношение
- Правила определения сущности, атрибута, отношения
- Основные правила формирования информационной модели
- Пример IDEF1X-модели

Информационные системы и устройства внешней памяти

- Информационная система (ИС) – программный комплекс, функции которого состоят:
  - в поддержке надежного долговременного хранения информации в памяти компьютера;
  - в выполнении требуемых для данного приложения преобразований информации и/или вычислений;
  - в предоставлении пользователям системы удобного и легко осваиваемого интерфейса.
- Объемы данных, с которыми приходится иметь дело ИС, достаточно велики, а сами данные обладают достаточно сложной структурой.
- Классическими примерами ИС являются банковские системы, системы резервирования авиационных или железнодорожных билетов, мест в гостиницах и т. д.

Информационные системы и устройства внешней памяти

- Надежное и долговременное хранение информации можно обеспечить только при наличии запоминающих устройств, сохраняющих информацию после выключения электропитания.
- Оперативная (основная) память (ОП) этим свойством обычно не обладает.
- В первые десятилетия развития вычислительной техники использовались два вида устройств внешней памяти: магнитные ленты и магнитные барабаны.



Информационные системы и устройства внешней памяти

В чем состоят реальные потребности разработчиков систем численных расчетов?

1. Для получения требуемых результатов серьезные вычислительные программы должны проработать достаточно долгое время (недели, месяцы, годы).
2. Требуется использовать программное сохранение частичных результатов вычислений, чтобы при возникновении непредвиденных сбоев аппаратуры можно было продолжить выполнение расчетов с некоторой контрольной точки.

- Для сохранения промежуточных результатов идеально подходят магнитные ленты:
  - при выполнении процедуры установки контрольной точки данные последовательно сбрасываются на ленту;
  - при необходимости перезапуска от сохраненной контрольной точки данные также последовательно с ленты считываются.

Информационные системы и устройства внешней памяти

- Вторая традиционная потребность численных программистов – максимально большой объем ОП.
- Большая ОП требуется, чтобы
  - обеспечить программе быстрый доступ к большому количеству обрабатываемых данных;
  - позволить выполнять сложные вычислительные программы большого объема.
- Поскольку объем реально доступной в ЭВМ ОП всегда являлся недостаточным для удовлетворения потребностей вычислений, требовалась быстрая внешняя память (ВП) для организации оверлеев и/или виртуальной памяти.
- Для этого идеально подходили магнитные барабаны.
- Обеспечивается быстрый доступ к внешней памяти.
- Для расширения ОП одной программы большой объем внешней памяти не требуется.

Информационные системы и устройства внешней памяти

- Требования к устройствам внешней памяти со стороны бизнес-приложений вызвали появление устройств внешней памяти со съемными пакетами магнитных дисков и подвижными головками чтения/записи, что явилось революцией в истории вычислительной техники.

- Эти устройства памяти
  - обладали существенно большей емкостью, чем магнитные барабаны (за счет наличия нескольких магнитных поверхностей);
  - обеспечивали удовлетворительную скорость доступа к данным в режиме произвольной выборки;
  - позволяли иметь архив данных практически неограниченного объема за счет возможности смены дискового пакета на устройстве.

Информационные системы и устройства внешней памяти

- С появлением магнитных дисков началась история систем управления данными во внешней памяти.
- До этого каждая прикладная программа, которой требовалось хранить данные во внешней памяти, сама определяла расположение каждой порции данных на магнитной ленте или барабане и выполняла обмены между оперативной и внешней памятью с помощью программно-аппаратных средств низкого уровня.
- Такой режим работы не позволял или очень затруднял поддержание на одном внешнем носителе нескольких архивов долговременно хранимой информации.
- Кроме того, каждой прикладной программе приходилось решать проблемы именования частей данных и структуризации данных во внешней памяти.

Файловые системы

- Историческим шагом явилось появление систем управления файлами.
- С точки зрения программиста приложений – это именованная область внешней памяти, в которую можно записывать и из которой можно считывать данные.
- Правила именования файлов, способ доступа к данным, хранящимся в файле, и структура этих данных зависят от конкретной системы управления файлами и, возможно, от типа файла.
- Система управления файлами берет на себя распределение внешней памяти, отображение имен файлов в соответствующие адреса внешней памяти и обеспечение доступа к данным.

Файловые системы

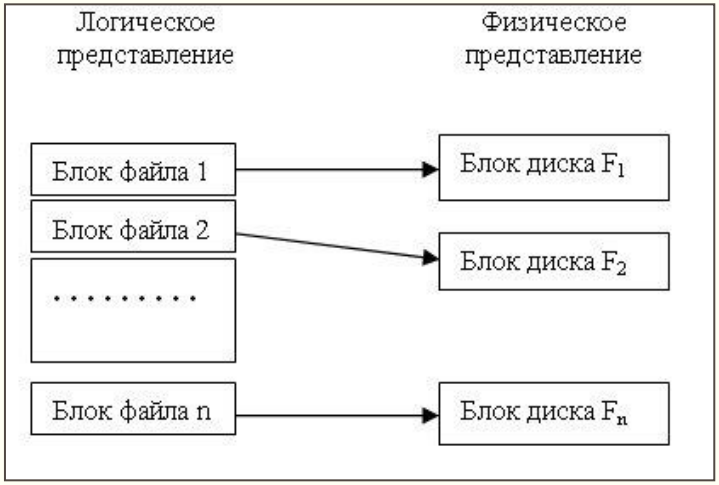
- Термин файловая система (file system) используется для обозначения как программной системы, управляющей файлами, так и архива файлов, хранящегося во внешней памяти.
- Было бы лучше в первом случае использовать термин система управления файлами, оставив за термином файловая система только второе значение.
- Однако принятая практика вынуждает использовать термин файловая система (ФС) в обоих смыслах.
- Точный смысл термина должен быть понятен из контекста.
- Аналогичная путаница возникает при некорректном использовании терминов база данных и система управления базами данных.
- Здесь эти термины строго различаются.

Файловые системы

- Обсудим историю ФС, их основные черты и области разумного применения:
  - структуры файлов;
  - логическая структура файловых систем и именование файлов;
  - авторизация доступа к файлам;
  - синхронизация многопользовательского доступа;
  - области разумного применения файлов.
- Ограничимся описанием основных свойств так называемых традиционных ФС, не затрагивая особенности современных систем с повышенной надежностью.

Файловые системы
Структуры файлов

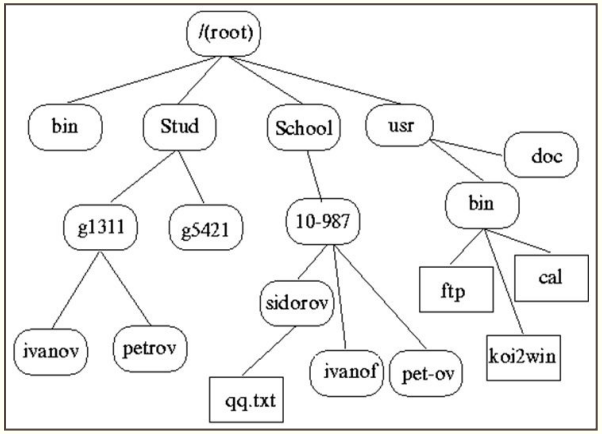
- Во всех современных ФС явно или неявно выделяется уровень, обеспечивающий работу с базовыми файлами, которые представляют собой наборы блоков, последовательно нумеруемых в адресном пространстве файла и отображаемых на физические блоки диска.
- Размер логического блока файла совпадает с размером физического блока диска или кратен ему;
  - обычно размер логического блока выбирается равным размеру страницы виртуальной памяти, поддерживаемой аппаратурой компьютера совместно с операционной системой.



- В некоторых ФС базовый уровень был доступен пользователю, но чаще он прикрывался некоторым более высоким уровнем, стандартным для пользователей.

Файловые системы
Логическая структура ФС и именование файлов

- Во всех современных файловых системах обеспечивается многоуровневое именование файлов за счет наличия во внешней памяти каталогов – дополнительных файлов со специальной структурой.



- Каждый каталог содержит имена каталогов и/или файлов, хранящихся в данном каталоге.
- Таким образом, полное имя файла состоит из списка имен каталогов плюс имя файла в каталоге, непосредственно содержащем данный файл.
- Поддержка многоуровневой схемы именования файлов обеспечивает несколько преимуществ, основным из которых является простая и удобная схема логической классификации файлов и генерации их имен.
- Можно сопоставить каталог или цепочку каталогов с пользователем, подразделением, проектом и т. д. и затем образовывать в этом каталоге файлы или каталоги, не опасаясь коллизий с именами других файлов или каталогов.

Файловые системы
Области разумного применения файлов

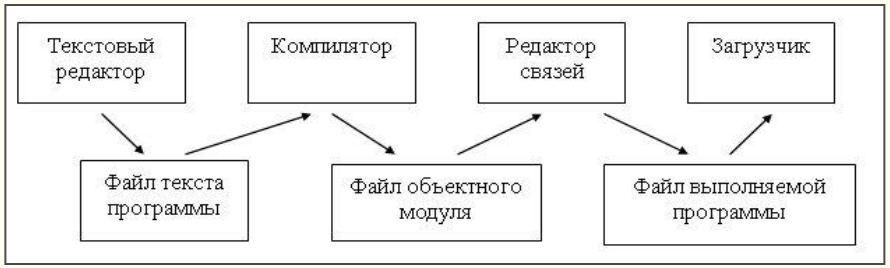
- Чаще всего файлы используются для хранения текстовых данных: документов, текстов программ и т. д.
- Такие файлы обычно создаются и модифицируются с помощью различных текстовых редакторов.
- Эти редакторы могут быть очень простыми, такими, как ed в мире UNIX или утилиты редактирования Far Manager, WordPad и других интерактивных сред Windows.
- Они могут быть сложными и многофункциональными, синтаксически ориентированными, как, например, GNU Emacs.
- Но обычно структура текстовых файлов очень проста (с точки зрения ФС):
  - либо последовательность записей, содержащих строки текста,
  - либо последовательность байтов, среди которых встречаются специальные символы (например, символы конца строки).
- Конечно же, сложность логической структуры текстового файла определяется текстовым редактором, но в любом случае ФС она не видна.

Файловые системы
Области разумного применения файлов

- Файлы, содержащие тексты программ, используются как входные файлы компиляторов (чтобы правильно воспринять текст программы, компилятор должен понимать логическую структуру текстового файла), которые, в свою очередь, формируют файлы, содержащие объектные модули.
- С точки зрения ФС объектные файлы также обладают очень простой структурой – последовательность записей или байтов.
- Система программирования накладывает на такую структуру более сложную и специфичную для этой системы логическую структуру объектного модуля.
- Логическая структура объектного модуля ФС неизвестна; эта структура поддерживается инструментами системы программирования.

Файловые системы
Области разумного применения файлов

- Аналогично обстоит дело с файлами, формируемыми редакторами связей (редактор связей должен понимать логическую структуру файлов объектных модулей) и содержащими образы выполняемых программ.
- Логическая структура таких файлов остается известной только редактору связей и загрузчику – программе операционной системы.



- Ситуация аналогична и в других случаях: например, при создании и использовании файлов, содержащих графическую, аудио- и видеоинформацию.

Файловые системы
Области разумного применения файлов

- Одним словом, файловые системы обычно обеспечивают хранение слабо структурированной информации, оставляя дальнейшую структуризацию прикладным программам.
- В перечисленных выше случаях использования файлов это даже хорошо, потому что
  - при разработке любой новой прикладной системы,
  - опираясь на простые, стандартные и сравнительно дешевые средства файловой системы,
  - можно реализовать те структуры хранения, которые наиболее точно соответствуют специфике данной прикладной области.

Потребности информационных систем

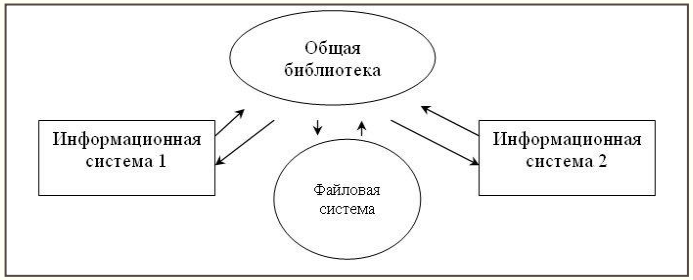
- Удовлетворяют ли рассмотренные выше базовые возможности файловых систем потребности информационных систем?
- Типовая информационная система, главным образом, ориентирована на хранение, выбор и модификацию данных соответствующей прикладной области.
- Структура таких данных зачастую очень сложна, и, хотя структуры данных различны в разных информационных системах, между ними часто бывает много общего.

Потребности информационных систем

- На начальном этапе использования вычислительной техники для построения ИС проблемы структуризации данных решались индивидуально в каждой ИС.
- Производились необходимые надстройки над файловыми системами (библиотеки программ), подобно тому, как это делается в компиляторах, редакторах и т. д.

Потребности информационных систем

- Но поскольку для функционирования информационных систем требуются сложные структуры данных, эти дополнительные индивидуальные средства управления данными являлись существенной частью информационных систем и практически повторялись от одной системы к другой.
- Стремление выделить общую часть информационных систем, ответственную за управление сложно структурированными данными, явилось первой побудительной причиной создания СУБД.
- Очень скоро стало понятно, что невозможно обойтись общей библиотекой программ, реализующей над стандартной базовой ФС более сложные методы
  хранения данных.



Потребности информационных систем

- Поясним это на примере.
- Пусть требуется реализовать ИС, поддерживающую учет служащих некоторой организации.
- Система должна выполнять следующие действия:
  - выдавать списки служащих по отделам;
  - поддерживать возможность перевода служащего из одного отдела в другой;
  - обеспечивать средства поддержки приема на работу новых служащих и увольнения работающих служащих.
- Кроме того, для каждого отдела должна поддерживаться возможность получения:
  - имени руководителя отдела;
  - общей численности отдела;
  - общей суммы заработной платы служащих отдела, среднего размера заработной платы и т. д.
- Для каждого служащего должна поддерживаться возможность получения:
  - номера удостоверения по полному имени служащего (для простоты допустим, что имена всех служащих различны);
  - полного имени по номеру удостоверения;
  - информации о соответствии служащего занимаемой должности и о размере его заработной платы.

Потребности информационных систем. Структуры данных

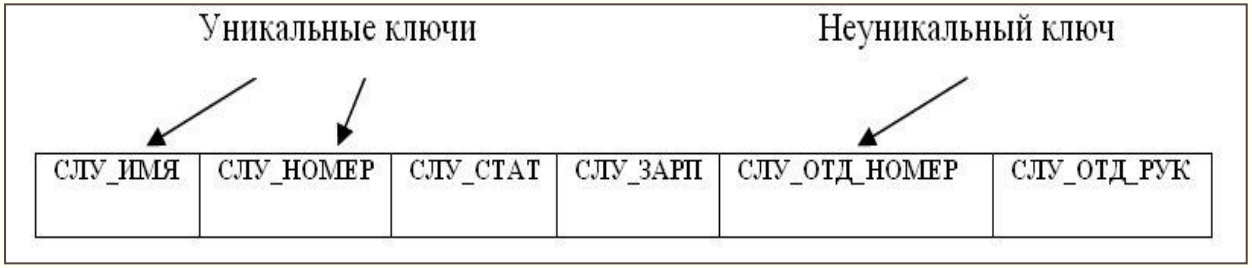
- Предположим, что мы решили основывать эту ИС на файловой системе и пользоваться одним файлом СЛУЖАЩИЕ, расширив базовые возможности файловой системы за счет специальной библиотеки функций.
- Поскольку минимальной информационной единицей является служащий, в этом файле должна содержаться одна запись для каждого служащего.
- Чтобы можно было удовлетворить указанные выше требования, запись о служащем должна иметь следующие поля:
  - полное имя служащего (СЛУ_ИМЯ);
  - номер его удостоверения (СЛУ_НОМЕР);
  - данные о соответствии служащего занимаемой должности (СЛУ_СТАТ; для простоты «да» или «нет», соответствует или не соответствует должности);
  - размер заработной платы (СЛУ_ЗАРП);
  - номер отдела (СЛУ_ОТД_НОМЕР).

Потребности информационных систем. Структуры данных

- Поскольку мы решили ограничиться одним файлом СЛУЖАЩИЕ, та же запись должна содержать имя руководителя отдела (СЛУ_ОТД_РУК).
- Иначе было бы невозможно, например, получить имя руководителя отдела с известным номером.
- Чтобы ИС могла эффективно выполнять свои базовые функции, необходимо обеспечить многоключевой доступ к файлу СЛУЖАЩИЕ по уникальным ключам СЛУ_ИМЯ и СЛУ_НОМЕР.
- В противном случае для выполнения наиболее часто используемых операций получения данных о конкретном служащем понадобится последовательный просмотр в среднем половины записей файла.

Потребности информационных систем. Структуры данных

- Кроме того, должна обеспечиваться возможность эффективного выбора всех записей с общим значением СЛУ_ОТД_НОМЕР, т. е. доступ по неуникальному ключу.
- Если не поддерживать специальный механизм доступа, то для получения данных об отделе в целом в общем случае потребуется полный просмотр файла.



- Но даже в этом случае, чтобы получить численность отдела или общий размер зарплаты, система должна будет выбрать все записи о служащих указанного отдела и посчитать соответствующие общие значения.

Потребности информационных систем. Структуры данных

- Таким образом, мы видим, что при реализации даже такой простой ИС на базе ФС возникают следующие затруднения:
  - требуется создание достаточно сложной надстройки для многоключевого доступа к файлам;
  - возникает существенная избыточность данных (для каждого служащего повторяется имя руководителя его отдела);
  - требуется выполнение массовой выборки и вычислений для получения суммарной информации об отделах.
- Кроме того, если в ходе эксплуатации системы потребуется, например, обеспечить операцию выдачи списков служащих, получающих указанную зарплату, то либо придется при выполнении каждой такой операции полностью просматривать файл, либо нужно будет реструктурировать файл СЛУЖАЩИЕ, объявляя ключевым и поле СЛУ_ЗАРП.

Потребности информационных систем. Структуры данных

- Для улучшения ситуации можно было бы поддерживать два многоключевых файла: СЛУЖАЩИЕ и ОТДЕЛЫ.
- Первый файл должен был бы содержать поля СЛУ_ИМЯ, СЛУ_НОМЕР, СЛУ_СТАТ, СЛУ_ЗАРП и СЛУ_ОТД_НОМЕР
- Второй – ОТД_НОМЕР, ОТД_РУК (номер удостоверения служащего, являющегося руководителем отдела), ОТД_СЛУ_ЗАРП (общий размер зарплаты служащих данного отдела) и ОТД_РАЗМЕР (общее число служащих в отделе)

Потребности информационных систем. Структуры данных

- Введение этих двух файлов позволило бы преодолеть большинство неудобств, перечисленных ранее.
- Каждый из файлов содержал бы только не дублируемую информацию, не возникала бы необходимость в динамических вычислениях суммарной информации по отделам.
- Но заметим, что при таком переходе наша ИС должна обладать некоторыми новыми особенностями, сближающими ее с СУБД.

Потребности информационных систем. Целостность данных.

- Теперь система должна «знать», что она работает с двумя информационно связанными файлами (это шаг в сторону схемы базы данных), должна иметь информацию о структуре и смысле каждого поля.
- Например, системе должно быть известно, что у полей СЛУ_ОТД_НОМЕР в файле СЛУЖАЩИЕ и ОТД_НОМЕР в файле ОТДЕЛЫ один и тот же смысл – номер отдела.
- Кроме того, система должна учитывать, что в ряде случаев изменение данных в одном файле должно автоматически вызывать модификацию второго файла, чтобы общее содержимое файлов было согласованным.
- Например, если на работу принимается новый служащий, то нужно добавить запись в файл СЛУЖАЩИЕ, а также должным образом изменить поля ОТД_СЛУ_ЗАРП и ОТД_РАЗМЕР в записи файла ОТДЕЛЫ, соответствующей отделу этого служащего.

Потребности информационных систем. Целостность данных.

- Более точно, система должна руководствоваться следующими правилами:
  - если в файле СЛУЖАЩИЕ содержится запись со значением поля СЛУ_ОТД_НОМЕР, равным n, то и в файле ОТДЕЛЫ должна содержаться запись со значением поля ОТД_НОМЕР, также равным n;
  - если в файле ОТДЕЛЫ содержится запись со значением поля ОТД_РУК, равным m, то и в файле СЛУЖАЩИЕ должна содержаться запись со значением поля СЛУ_НОМЕР, также равным m;
  - далее мы увидим, что эти правила являются частными случаями общего правила ссылочной целостности: поле СЛУ_ОТД_НОМЕР содержит «ссылки» на записи таблицы ОТДЕЛЫ, и поле ОТД_РУК содержит «ссылки» на записи таблицы СЛУЖАЩИЕ;

Потребности информационных систем. Целостность данных.

- при любом корректном состоянии ИС значение поля ОТД_СЛУ_ЗАРП любой записи отд_k файла ОТДЕЛЫ должно быть равно сумме значений поля СЛУ_ЗАРП всех тех записей файла СЛУЖАЩИЕ, в которых значение поля СЛУ_ОТД_НОМЕР совпадает со значением поля ОТД_НОМЕР записи отд_k;
- при любом корректном состоянии ИС значение поля ОТД_РАЗМЕР любой записи отд_k файла ОТДЕЛЫ должно быть равно числу всех тех записей файла СЛУЖАЩИЕ, в которых значение поля СЛУ_ОТД_НОМЕР совпадает со значением поля ОТД_НОМЕР записи отд_k;
- далее мы увидим, что эти правила представляют собой примеры общих ограничений целостности базы данных.

Потребности информационных систем. Целостность данных.

- Понятие согласованности, или целостности, данных является ключевым понятием баз данных.
- Фактически, если в ИС поддерживается согласованное хранение данных в нескольких файлах, можно говорить о том, что в ней поддерживается база данных (БД).
- Если же некоторая вспомогательная система управления данными позволяет работать с несколькими файлами, обеспечивая их согласованность, можно назвать ее системой управления базами данных (СУБД).
- Требование поддержки согласованности данных в нескольких файлах не позволяет при построении ИС обойтись библиотекой функций: такая система должна обладать некоторыми собственными данными (их принято называть метаданными), определяющими целостность данных.
- В нашем примере ИС должна отдельно сохранять метаданные о структуре файлов СЛУЖАЩИЕ и ОТДЕЛЫ, а также правила, определяющие условия целостности данных в этих файлах (принято считать, что правила также составляют часть метаданных).

Потребности информационных систем. Языки запросов.

- Но обеспечение целостности данных – это далеко не все, что обычно требуется от СУБД.
- Начнем с того, что даже в нашем примере пользователю ИС будет не слишком просто получить, например, общую численность отдела, в котором работает Петр Иванович Сидоров.
- Придется сначала узнать номер отдела, в котором работает указанный служащий, а затем установить численность этого отдела.
- Было бы гораздо проще, если бы СУБД позволяла сформулировать такой запрос на языке, более близком пользователям.
- Такие языки называются языками запросов к базам данных.

Потребности информационных систем. Языки запросов.

- На языке запросов SQL наш запрос можно было бы выразить в следующей форме (запрос 1):

```
SELECT ОТД_РАЗМЕР
  FROM СЛУЖАЩИЕ, ОТДЕЛЫ
    WHERE СЛУ_ИМЯ = 'ПЕТР ИВАНОВИЧ СИДОРОВ' AND
                СЛУ_ОТД_НОМЕР = ОТД_НОМЕР;
```

- Это пример запроса на языке SQL с «полусоединением»: с одной стороны, запрос адресуется к двум файлам – СЛУЖАЩИЕ и ОТДЕЛЫ, но с другой стороны, данные выбираются только из файла ОТДЕЛЫ.
- Условие СЛУ_ОТД_НОМЕР = ОТД_НОМЕР всего лишь «ограничивает» интересующий нас набор записей об отделах до одной записи, если Петр Иванович Сидоров действительно работает на данном предприятии.
- Если же Петр Иванович Сидоров не работает на предприятии, то условие СЛУ_ИМЯ = 'ПЕТР ИВАНОВИЧ СИДОРОВ' не будет удовлетворяться ни для одной записи файла СЛУЖАЩИЕ, и поэтому запрос выдаст пустой результат.

Потребности информационных систем. Языки запросов

- Возможна и другая формулировка того же запроса (запрос 2):

```
SELECT ОТД_РАЗМЕР
  FROM ОТДЕЛЫ
    WHERE ОТД_НОМЕР =
        (SELECT СЛУ_ОТД_НОМЕР
          FROM СЛУЖАЩИЕ
            WHERE СЛУ_ИМЯ = ‘ПЕТР ИВАНОВИЧ СИДОРОВ');
```

- Это пример запроса на языке SQL с вложенным подзапросом.
- Во вложенном подзапросе выбирается значение поля СЛУ_ОТД_НОМЕР из записи файла СЛУЖАЩИЕ, в которой значение поля СЛУ_ИМЯ равняется строковой константе ‘ПЕТР ИВАНОВИЧ СИДОРОВ’.
- Если такая запись существует, то она единственная, поскольку поле СЛУ_ИМЯ является уникальным ключом файла СЛУЖАЩИЕ.
- Тогда результатом выполнения подзапроса будет единственное значение – номер отдела, в котором работает Петр Иванович Сидоров.
- Во внешнем запросе это значение будет ключом доступа к файлу ОТДЕЛЫ, и снова будет выбрана только одна запись, поскольку поле ОТД_НОМЕР является уникальным ключом файла ОТДЕЛЫ.
- Если же на данном предприятии Сидоров не работает, то подзапрос выдаст пустой результат, и внешний запрос тоже выдаст пустой результат.

Потребности информационных систем. Языки запросов

- Приведенные примеры показывают, что при формулировке запроса с использованием SQL можно не задумываться о том, как будет выполняться этот запрос.
- Среди метаданных базы данных будет содержаться информация о том, что поле СЛУ_ИМЯ является ключевым для файла СЛУЖАЩИЕ (т. е. по заданному значению имени служащего можно быстро найти соответствующую запись или убедиться в том, что запись с таким значением поля СЛУ_ИМЯ в файле отсутствует), а поле ОТД_НОМЕР – ключевое для файла ОТДЕЛЫ (и более того, оба ключа в соответствующих файлах являются уникальными), и система сама воспользуется этим.
- Можно формально доказать, что формулировки запрос 1 и запрос 2 эквивалентны, т. е. вне зависимости от состояния данных всегда производят один и тот же результат.

Потребности информационных систем. Языки запросов

- Наиболее вероятным способом выполнения запроса в обеих формулировках будет выборка записи из файла СЛУЖАЩИЕ со значением поля СЛУ_ИМЯ, равным строке ‘ПЕТР ИВАНОВИЧ СИДОРОВ’, взятие из этой записи значения поля СЛУ_ОТД_НОМЕР и выборка из таблицы ОТДЕЛЫ записи с таким же значением поля ОТД_НОМ.
- Если же, например, возникнет потребность в получении списка сотрудников, не соответствующих занимаемой должности, то достаточно обратиться к системе с запросом (запрос 3):
- SELECT СЛУ_ИМЯ, СЛУ_НОМЕР
  FROM СЛУЖАЩИЕ
  WHERE СЛУ_СТАТ = ‘НЕТ’;
- Тогда система сама выполнит необходимый полный просмотр файла СЛУЖАЩИЕ, поскольку поле СЛУ_СТАТ не является ключевым, и другого способа выполнения не существует.

Потребности информационных систем. Транзакции, журнализация и многопользовательский режим

- Далее, представим себе, что в первоначальной реализации информационной системы, основанной на использовании библиотек расширенных методов доступа к файлам, обрабатывается операция принятия на работу нового служащего.
- Следуя требованиям согласованного изменения файлов, информационная система вставляет новую запись в файл СЛУЖАЩИЕ и собирается модифицировать соответствующую запись файла ОТДЕЛЫ (или вставлять в этот файл новую запись, если служащий является первым в своем отделе), но именно в этот момент происходит (например) аварийное выключение питания компьютера.
- Очевидно, что после перезапуска системы ее база данных будет находиться в рассогласованном состоянии (точно будут нарушены два последние правила), а может быть, и оба первые правила.
- Потребуется выяснить это (а для этого нужно явно проверить соответствие данных в файлах СЛУЖАЩИЕ и ОТДЕЛЫ) и привести данные в согласованное состояние.

Потребности информационных систем. Транзакции, журнализация и многопользовательский режим

- Проверку и коррекцию можно выполнить, например, следующим образом.
- Сгруппировать записи файла СЛУЖАЩИЕ по значениям поля СЛУ_ОТД_НОМЕР.
- Для каждой группы
  - проверить, существует ли в файле ОТДЕЛЫ запись, значение поля ОТД_НОМ которой равняется значению поля СЛУ_ОТД_НОМЕР записей данной группы;
  - если такой записи в файле ОТДЕЛЫ нет, то исключить группу из файла СЛУЖАЩИЕ и перейти к обработке следующей группы;
  - иначе посчитать число записей в группе и вычислить суммарное значение заработной платы;
  - обновить полученными значениями поля ОТД_РАЗМЕР и ОТД_СЛУ_ЗАРП соответствующей записи файла ОТДЕЛЫ и перейти к обработке следующей группы.

Потребности информационных систем. Транзакции, журнализация и многопользовательский режим

- Настоящие СУБД берут такую работу на себя, поддерживая транзакционное управление и журнализацию изменений базы данных.
- Прикладная система не обязана заботиться о поддержке корректности состояния базы данных, хотя и должна знать, какие цепочки операций изменения данных являются допустимыми.
- Представим теперь, что в информационной системе требуется обеспечить параллельную (например, многотерминальную) работу с базой данных служащих и отделов.
- Если опираться только на использование файлов, то для обеспечения корректности на все время модификации любого из двух файлов доступ других пользователей к этому файлу будет блокирован.
- Таким образом, зачисление на работу Петра Ивановича Сидорова существенно затормозит получение информации о служащем Иване Сидоровиче Петрове, даже если они работают в разных отделах.
- Настоящие СУБД обеспечивают гораздо более тонкую синхронизацию параллельного доступа к данным.

МЕТОДОЛОГИЯ ИНФОРМАЦИОННОГО МОДЕЛИРОВАНИЯ IDEF1X

Основные вопросы

- Понятие модели предметной области
- Основные понятия: сущность, атрибут, отношение
- Правила определения сущности, атрибута, отношения
- Основные правила формирования информационной модели
- Пример IDEF1X-модели на примере процесса постройки садового домика

Модель предметной области

- Под моделью предметной области понимается некоторая система, имитирующая структуру или функционирование исследуемой предметной области и отвечающая основному требованию – быть адекватной этой области.

Требования к моделям предметных областей

- формализация, обеспечивающая однозначное описание структуры предметной области;
- понятность для заказчиков и разработчиков на основе применения графических средств отображения модели;
- обеспечение оценки эффективности реализации модели предметной области на основе определенных методов и вычисляемых показателей.

Некоторые определения

- Язык моделирования – это нотация, в основном графическая, которая используется для описания проектов.

- Нотация представляет собой совокупность графических объектов, используемых в модели.

Что такое IDEF1X?

- Методология IDEF1X (IDEF1 Extended) – язык для семантического моделирования данных, основанных на концепции «сущность-связь». Является расширением стандарта IDEF1.
- Диаграмма «сущность-связь» ERD (Entity-Relationship Diagram) предназначена для разработки модели данных и обеспечивает стандартный способ определения данных и отношений между ними.
- Теоретической базой построения информационной модели является теория баз данных типа «сущность-связь».

Что такое IDEF1X?

- Согласно стандарту , основными составляющими модели IDEF1X являются:
- 1) люди, предметы, явления, о которых хранится информация (далее – сущности)
- 2) связи между этими элементами (далее – отношения)
- 3) характеристики этих элементов (далее – атрибуты)

Определение сущности

- Сущность – это множество реальных или абстрактных объектов (людей, мест, событий), обладающих общими атрибутами или характеристиками.
- Любой объект системы может быть представлен только одной сущностью, которая должна быть уникально идентифицирована.
- Пример. Сущность – Студент. Экземпляр сущности – студент Иванов И.И.

Понятие атрибута

- Атрибут – характеристика сущности.
- Пример. Сущность «Студент» имеет атрибут «ФИО».
- Экземпляр сущности «студент» (конкретный человек) будет иметь экземпляр атрибута «ФИО» (например, Иванов И.И.)

Понятие отношения

- Отношения – связь между двумя и более сущностями. Именование отношения осуществляется с помощью грамматического оборота глагола (имеет, определяет, …).

Таким образом…

- Сущности представляют собой базовый тип информации, хранимый в БД, а отношения показывают, как эти типы данных взаимосвязаны друг с другом.

Правила определения сущности

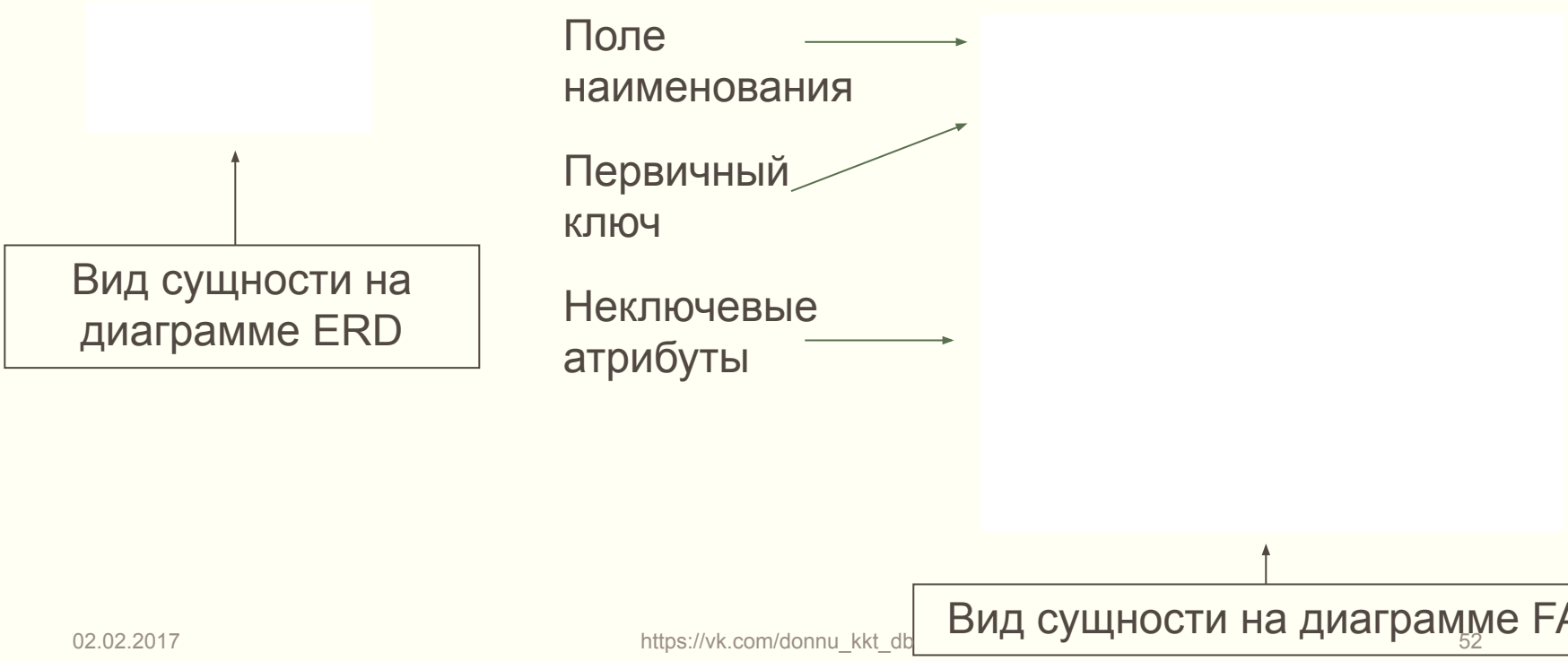
- Сущность должна иметь уникальное имя и именоваться существительным в единственном числе.
- Пример: Студент, Кредитная карта, Договор,…
- Сущность обладает одним или несколькими атрибутами, которые ей либо принадлежат, либо наследуются через отношения.
- Сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый образец сущности и называются ключом (составным ключом).

Правила определения сущности

- Каждая сущность может обладать любым количеством отношений с другими сущностями.
- Если внешний ключ целиком используется в составе первичного ключа, то сущность является зависимой от идентификатора.
- В нотации IDEF1X сущность изображается в виде прямоугольника, в зависимости от уровня представления данных могут быть некоторые различия

Графическое представление сущности

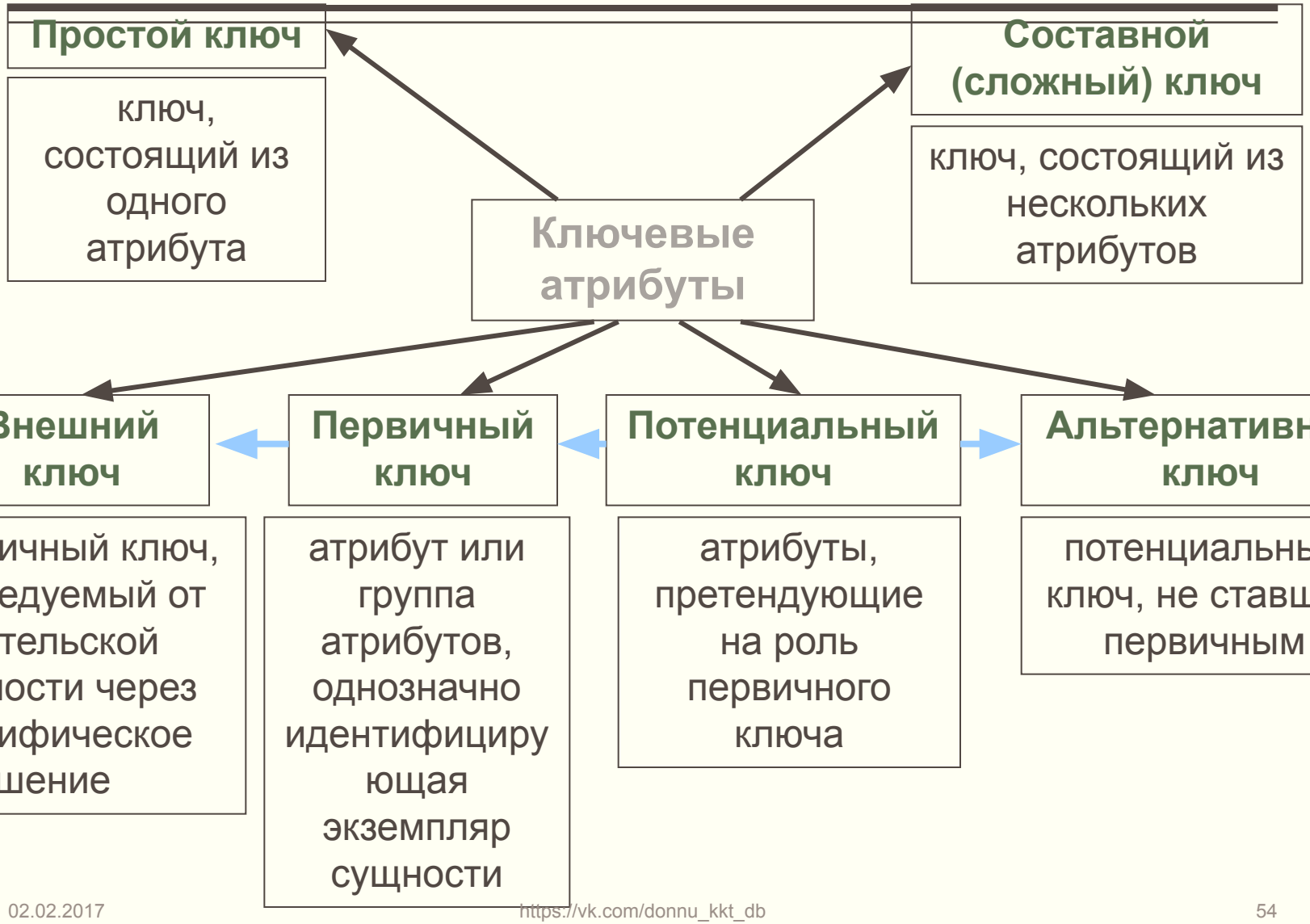
- Различают следующие уровни представления сущности: диаграмма «сущность-связь» (ERD), модель данных, основанная на ключах (KB), полная атрибутивная модель (FA)



Правила определения атрибутов

- Каждый атрибут каждой сущности обладает уникальным именем.
- Сущность может обладать любым количеством атрибутов.
- Различают собственные и наследуемые атрибуты. Собственные атрибуты являются уникальными в рамках модели. Наследуемые передаются от сущности-родителя при определении идентифицирующей связи.

Ключевые атрибуты



Примеры ключевых атрибутов

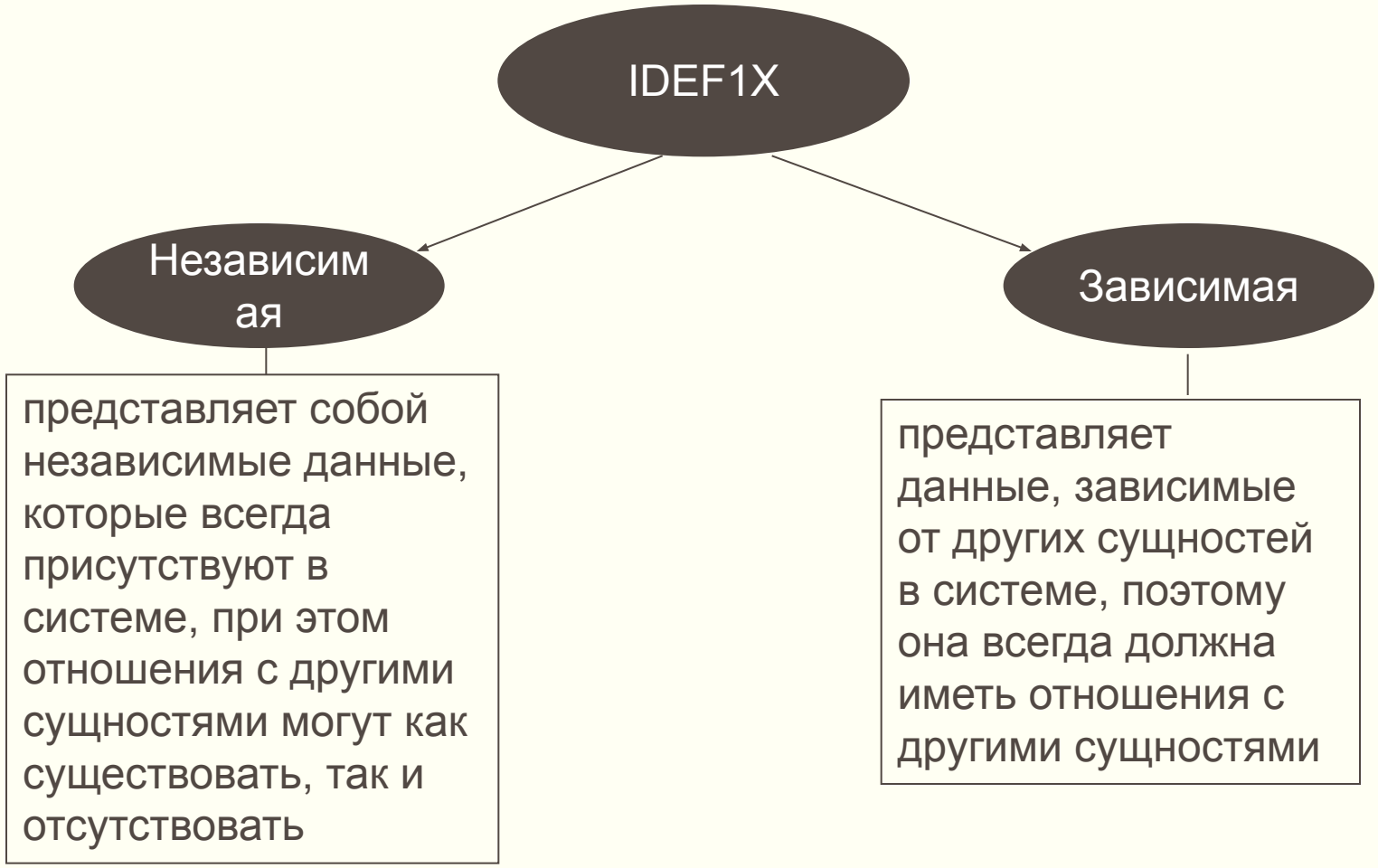
№_зачетнойКнижки – первичный простой ключ;

ФИО+дата_рождения – альтернативный ключ

ФИО+дата_рождения – первичный составной ключ;

№_зачетнойКнижки – альтернативынй ключ

Типы сущностей в IDEF1X



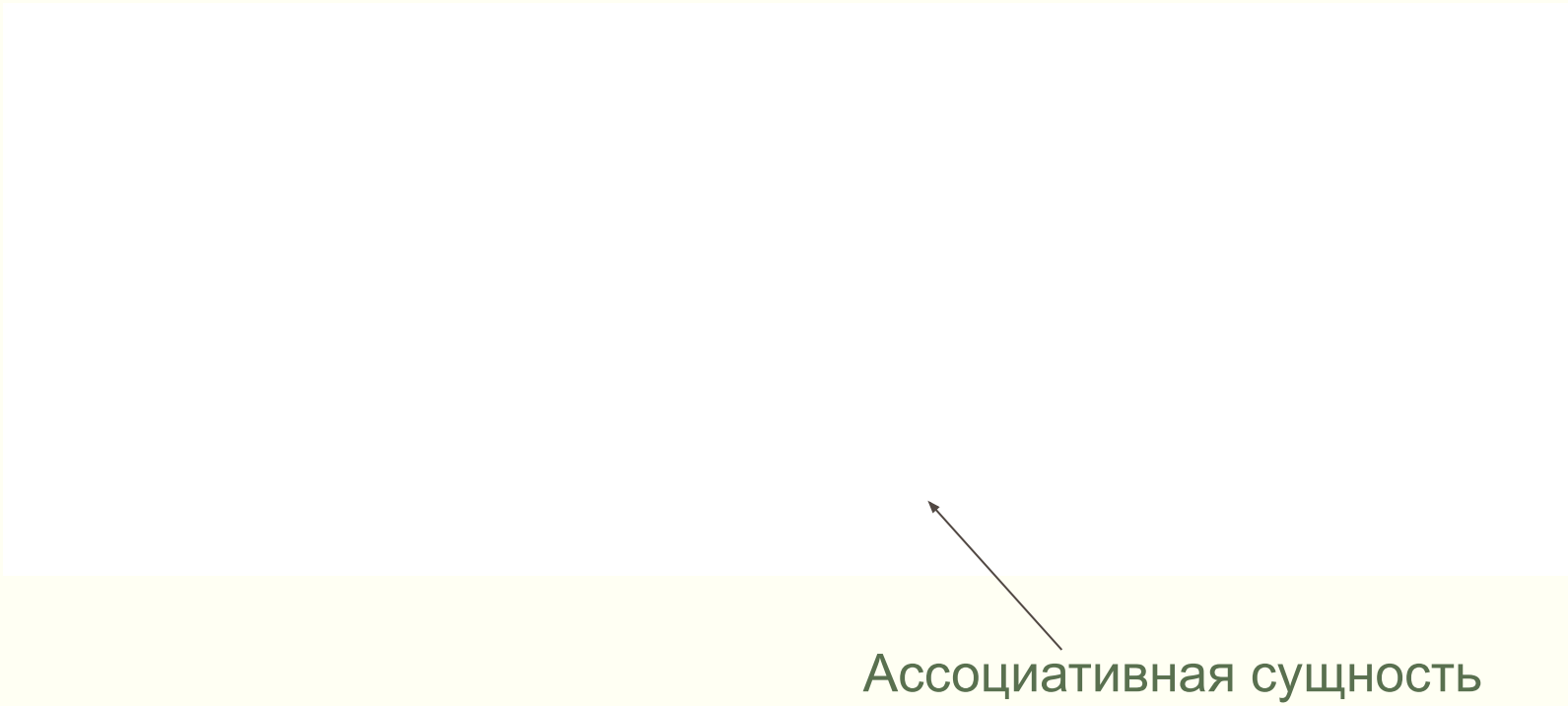
Типы зависимых сущностей

1. Характеристическая - это зависимая дочерняя сущность, которая связана только с одной родительской сущностью и по смыслу хранит информацию о характеристиках родительской сущности

2. Категориальная – дочерняя сущность в иерархии наследования

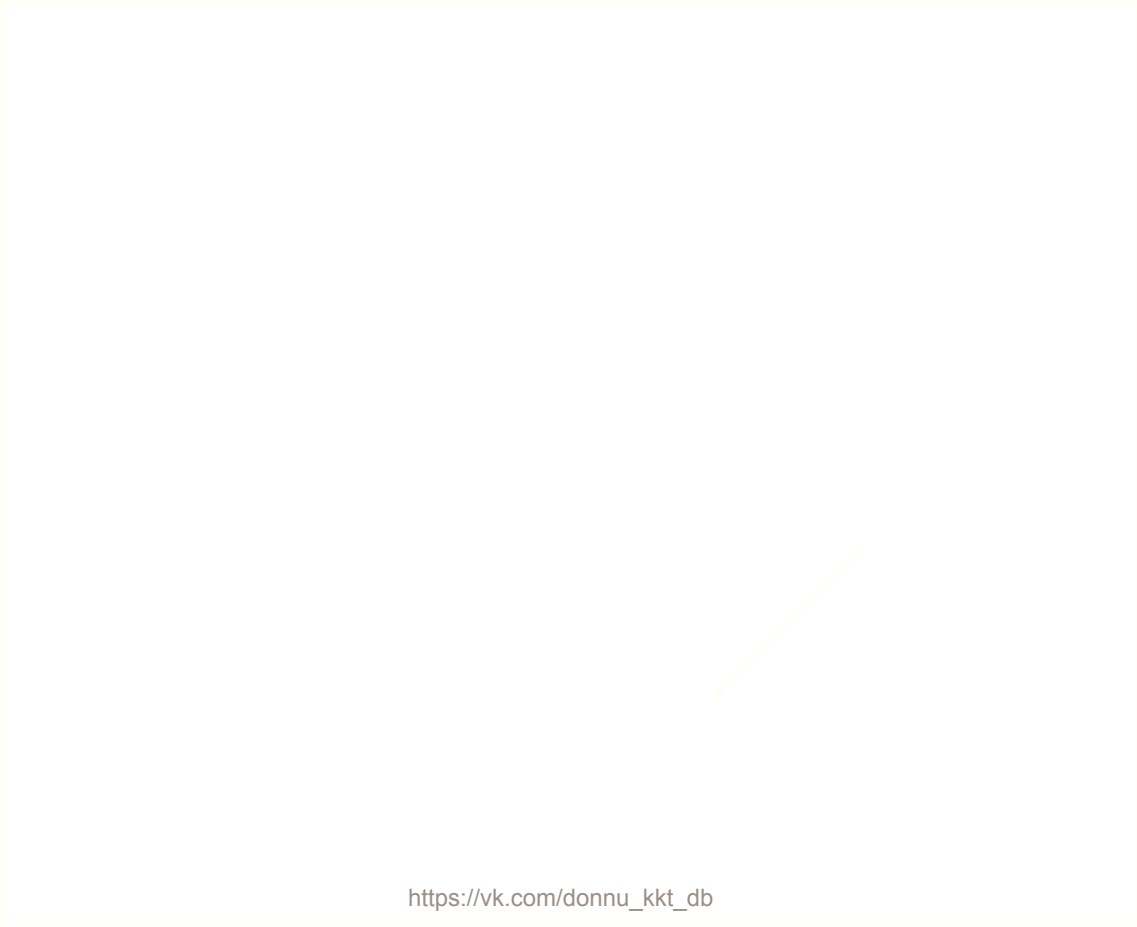
Типы зависимых сущностей

- Ассоциативная - сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущности



Типы зависимых сущностей

- Именующая - частный случай ассоциативной сущности, не имеет собственных атрибутов, только атрибуты родительской сущности



Правила отношений

- 1) При определении отношения типа «родитель-потомок»:
- 1.1. Экземпляр потомка связан с одним родителем
- 1.2. Экземпляр-родитель может быть связан с несколькими экземплярами потомков.
- 2) В идентифицирующем отношении сущность-потомок всегда является зависимой от идентифицирующей сущности.

Виды отношений

а) *идентифицирующее отношение* Сущность А1 однозначно определяет сущность А2. Ее первичный ключ наследуется в качестве первичного ключа сущностью А2 (внешний ключ)

б) *неидентифицирующее отношение* Сущность А1 связана с сущностью А2, но однозначно не определяет ее. Первичный ключ сущности А1 наследуется в качестве неключевого атрибута сущности А2

в) *отношение «многие-ко-многим» (неспецифическое).* Сущности А1 и А2 имеют формальную связь, но наследования атрибутов не происходит.

г) *отношение категоризации* (см. далее)

Правила отношений

- 3) Сущность может быть связана с любым количеством других сущностей как в качестве родителя, так и в качестве потомка.
- 4) Отношение определяется мощностью. Мощность связи служит для обозначения отношения количества экземпляров родительской сущности к числу экземпляров дочерней.

4 типа мощности отношений

- а) общий случай, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности

- б) когда одному экземпляру родительской сущности соответствует 1 или много экземпляров дочерней (0 исключается).

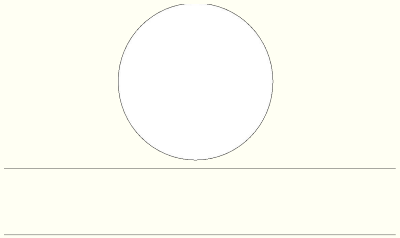
4 типа мощности отношений

- в) когда одному экземпляру родительской сущности соответствует 0 или 1 экземпляр дочерней сущности.

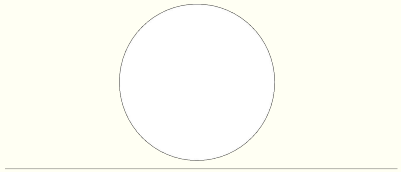
- г) когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.

Отношения категоризации

- Отношения категоризации – отношения между двумя и более сущностями, в которых каждый экземпляр одной сущности, называемой общей, связан в точности с одним экземпляром сущности, называемой сущностью-категорией.
- Категория выделяется из общей сущности по определенному признаку.
- Различают полную и неполную категоризацию



А) Дискриминатор – символ полной категоризации



Б) Дискриминатор – символ неполной категоризации

Пример отношений категоризации

Описание: Могут быть выделены следующие типы сотрудников: постоянный и совместитель. Категоризация неполная, т.к. могут быть и другие типы, например, консультанты. Тип – признак категоризации

Правила отношений категоризации

- 1. Сущность типа «категория» может иметь только одну общую сущность.
- 2. Сущность-категория, принадлежащая одному отношению категоризации, может быть общей сущностью в другом отношении категоризации

Пример иерархии категорий

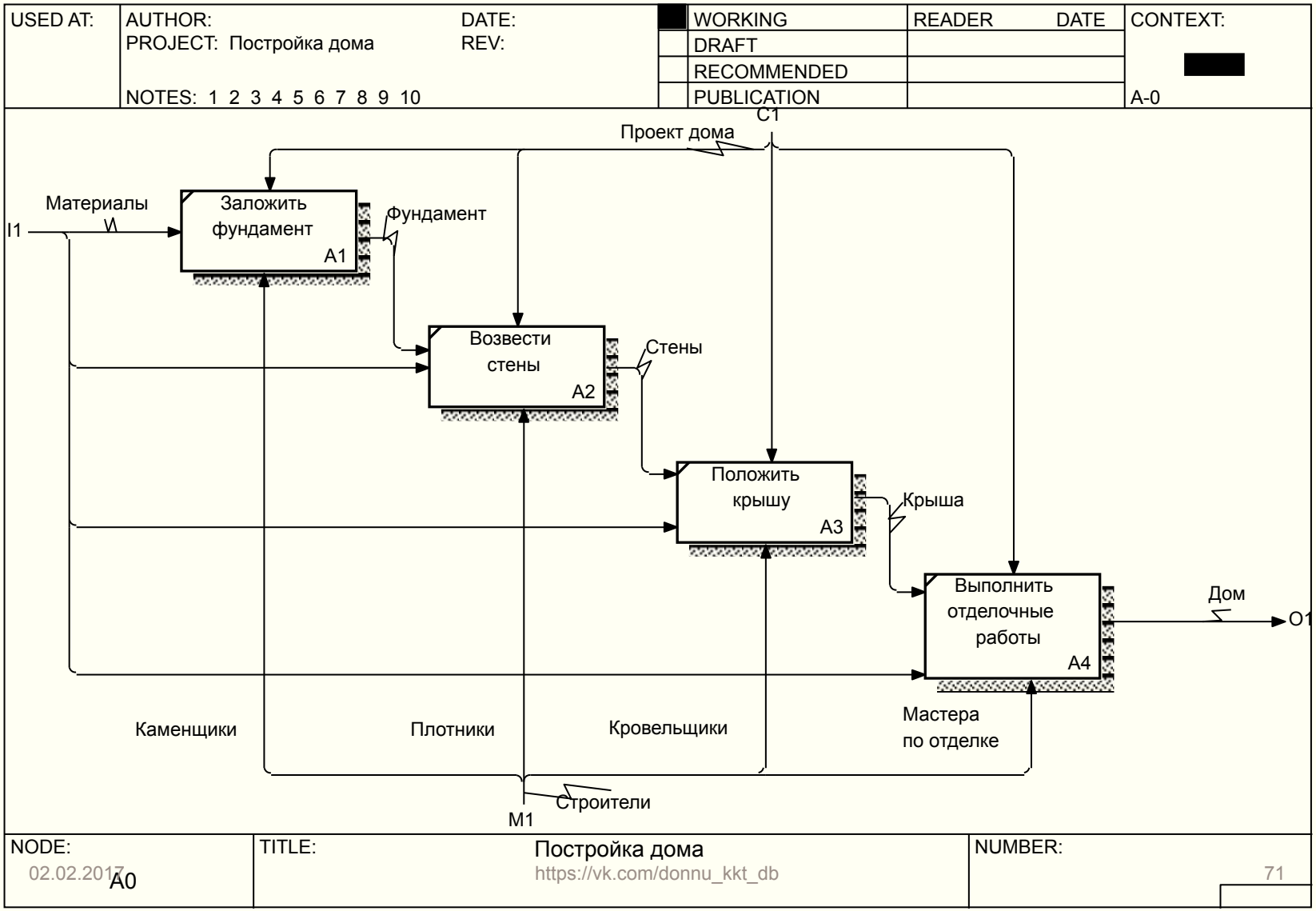
Правила отношений категоризации

- 3. Сущность может являться общей в любом количестве отношений категоризации.
- 4. Атрибуты первичного ключа сущности-категории должны совпадать с атрибутами первичного ключа общей сущности.
- 5. Все экземпляры сущности-категории имеют одно и то же значение дискриминатора, следовательно, все экземпляры других категорий должны иметь другое значение дискриминатора.

Основные правила построения информационной модели

- 1. Все стрелки (вход, выход, управление, механизм) функциональной модели IDEF0 становятся потенциальными сущностями, а функции, связывающие их, трансформируются в отношения между этими сущностями. Для этого составляется пул – список потенциальных сущностей.
- 2. Число сущностей и связей в IDEF1X-модели считается необозримым, если их количество превышает 25-30. Поэтому далее рассматривается совокупность сущностей и отношений для каждой функции.
- 3. Информационная модель функции должна позволять воспроизвести структуру документа и часть информации в нем, а также воспроизвести информацию порождаемого документа.
- 4. Текстовые пояснения заносятся в глоссарий или оформляются гипертекстом.
- 5. На основании определения типов отношений, анализа функций и дальнейшего изучения предметной области определяются атрибуты.

Пример функциональной модели, построенной с использованием CASE-средства BPWin



Построение информационной модели процесса постройки садового домика

- 1. На основе функциональной модели IDEF0 составим пул – список потенциальных сущностей.

Пул:

1. Дом
2. Крыша
3. Материалы
4. Проект дома
5. Стены
6. Строители
7. Фундамент
8. Каменщики
9. Плотники
10. Кровельщики
11. Мастера по отделке

Построение информационной модели процесса постройки садового домика

- 2. Определим сущности

Построение информационной модели процесса постройки садового домика

Заключение

- Развитие аппаратных и программных средств управления внешней памятью диктовалось потребностями ИС, для построения которых требовалась возможность надежного долговременного хранения больших объемов данных, а также обеспечение достаточно быстрого доступа к этим данным.
- Системы управления файлами во внешней памяти обеспечивают минимальные потребности ИС, предоставляя средства распределения и структуризации дисковой памяти, именования файлов, авторизации доступа и поддержки многопользовательского режима.
- На примере тривиальной ИС были показаны ситуации, в которых возможности ФС явно недостаточны.
- Более того, попытки расширения возможностей ФС путем включения в приложение дополнительных программных компонентов во многих случаях не приводят к успеху.
- В пределе такие попытки могут привести к появлению самостоятельного программного продукта, обладающего некоторыми чертами СУБД.
- Однако настоящие СУБД являются настолько большими и сложными программными системами, что вероятность успешного создания «самодельной» СУБД ничтожно мала.