Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Модели данных и системы управления базами данных»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  ассистент  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.В. Давыдчик |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«ПРОГРАММНОЕ СРЕДСТВО УЧЁТА ТЕХНИКИ НА ПРЕДПРИЯТИИ»**

БГУИР КП 1-40 04 01 027 ПЗ

|  |  |
| --- | --- |
|  | Выполнила студент группы 053504  КУДРЯВЦЕВА Авелина Николаевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc149334887)

[1 МОДЕЛИРОВАНИЕ ДАННЫХ 6](#_Toc149334888)

[1.1 Типы моделей данных 6](#_Toc149334889)

[1.2 Методы моделирования данных 9](#_Toc149334890)

[1.3 Процесс моделирования данных 10](#_Toc149334891)

[1.4 Системы управления базами данных 11](#_Toc149334892)

[2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 12](#_Toc149334893)

[2.1 Язык программироавния *Python* 12](#_Toc149334894)

[2.2 *PyCharm* 13](#_Toc149334895)

2.3 *PostgreSQ*L..………………………………………………...……………..14

[3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ 15](#_Toc149334896)

[4 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 16](#_Toc149334897)

[5 АРХИТЕКТУРА ВЕБ-ПРИЛОЖЕНИЯ 17](#_Toc149334898)

[ЗАКЛЮЧЕНИЕ 18](#_Toc149334899)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc149334900)

# **ВВЕДЕНИЕ**

В современном цифровом мире, в котором доступность информации и скорость обработки данных становятся все более важными, внедрение эффективных баз данных становится ключевым компонентом многих веб-приложений. Базы данных служат фундаментальным инструментом для организации информации и обеспечения доступа к ней во всех сферах деятельности, начиная от предприятий и исследовательских организаций и заканчивая личным использованием.

Актуальность задачи автоматизации учета компьютерного и другого оборудования на предприятии возрастает при наличии большого парка компьютеров, офисной техники, торгового и другого оборудования. Наибольшая потребность знать, где и какая единица находится, оперативно отслеживать изменения, связанные с оборудованием, возникает у ИТ-подразделений. Особую значимость задача автоматизации учета оборудования приобретает на крупных предприятиях. Обработка все время растущих массивов информации стала возможна только с использованием современных компьютерных технологий. Организовать систему учёта техники на предприятии, вести учёт компьютеров и комплектующих, сейчас невозможно без дополнительно установленного на компьютер программного обеспечения.

Главной целью курсовой работы является разработка информационной системы для учета компьютерной техники предприятия, с использованием методологии функционального моделирования и графической нотации IDEF0, диаграмм потоков данных DFD и стандарта документирования процессов IDEF3.

# 

# **1 МОДЕЛИРОВАНИЕ ДАННЫХ**

Моделирование данных  – это процесс создания визуального представления или чертежа, определяющего системы сбора и управления информацией в любой организации. Этот план или модель данных помогает различным заинтересованным сторонам, таким как аналитики данных, ученые и инженеры, создать единое представление о данных организации. Модель описывает, какие данные собирает компания, взаимосвязь между различными наборами данных и методы, которые будут использоваться для хранения и анализа данных.

В современной практике моделирования управленческой и производственной деятельности для обозначения объектов моделирования принято использовать термин «бизнес - процесс». При моделировании бизнес - процессов следует уделить внимание ряду факторов:

– Корректная постановка целей;

– Грамотная информированность персонала организации относительно целей и результатов проекта;

– Эффективное применение инструментов моделирования;

– Наличие корпоративных стандартов описания и регламентации бизнес - процессов.

Для моделирования бизнес - процессов используется несколько различных методов. Их основой являются как структурный, так и объектно-ориентированный подходы к моделированию. Наиболее развитые методы используют элементы обоих подходов. К числу наиболее распространенных методов можно отнести:

– метод функционального моделирования SADT (IDEF0);

– метод моделирования процессов IDEF3;

– моделирование потоков данных DFD;

С точки зрения бизнес–моделирования каждый из представленных подходов обладает своим преимуществами. Объектный подход позволяет построить более устойчивую к изменениям систему, лучше соответствует

существующим структурам организации. Функциональное моделирование хорошо показывает себя в тех случаях, когда организационная структура находится в процессе изменения или вообще слабо оформлена. Подход от выполняемых функций интуитивно лучше понимается исполнителями при получении от них информации об их текущей работе.

## **Типы моделей данных**

Метод функционального моделирования IDEF0(Function Modeling) - совокупность правил и процедур, предназначенных для построения функциональной модели объекта какой–либо предметной области. Функциональная модель объекта отображает производимые им действия и связи между ними.

В соответствии с этим методом бизнес - модель должна выглядеть следующим образом:

Верхний уровень модели должен отражать только контекст системы, то есть её взаимодействие с внешним миром.

На втором уровне модели должны находиться все основные виды деятельности предприятия, другими словами тематически сгруппированные бизнес - процессы предприятия и их взаимосвязь.

Дальнейшая детализация бизнес–процессов осуществляется посредством бизнес–функций, то есть совокупностей операций, сгруппированных по определенным признакам.

Описание элементарной бизнес–операций осуществляется с помощью задания алгоритма ее выполнения. Пример диаграммы IDEF0 изображен на рисунке 1.



Рисунок 1 – Пример описания диаграммы IDEF0

Метод моделирования потоков данных DFD (Data Flow Diagrams)- диаграммы потоков данных. Основное средство моделирования функциональных требований к проектируемой системе.

Компоненты модели: диаграммы; словарь данных; спецификации процессов.

Элементы диаграмм: поток данных; хранилище; внешняя сущность.

Поток данных механизм, использующейся для моделирования и передачи информации из одной части системы в другую.

Внешняя сущность-объект/субъект вне контекста системы, который является.

Хранилище - срез потоков данных во времени, содержащий данные, которые нужно сохранить между процессами.

Основные преимущества:

– возможность однозначно определить внешние сущности, анализируя потоки информации внутри и вне системы;

– возможность проектирования сверху вниз, что облегчает построение модели «как должно быть»;

– наличие спецификаций процессов нижнего уровня, что позволяет преодолеть логическую незавершенность функциональной модели и построить полную функциональную спецификацию разрабатываемой системы;

– модели имеют очень богатый набор элементов, адекватно отражающих их специфику;

– существуют и поддерживаются рядом CASE-инструментов алгоритмы автоматического преобразования иерархии DFD в структурные карты, демонстрирующие межсистемные, внутрисистемные связи и иерархию систем

Недостатки:

– необходимость искусственного ввода управляющих процессов, поскольку управляющие воздействия (потоки) и управляющие процессы с точки зрения DFD ничем не отличаются от обычных;

– отсутствие понятия времени, т.е. отсутствие анализа временных промежутков при преобразовании данных (все ограничения по времени должны быть введены в спецификациях процессов).

Пример диаграммы DFD изображен на рисунке 2.

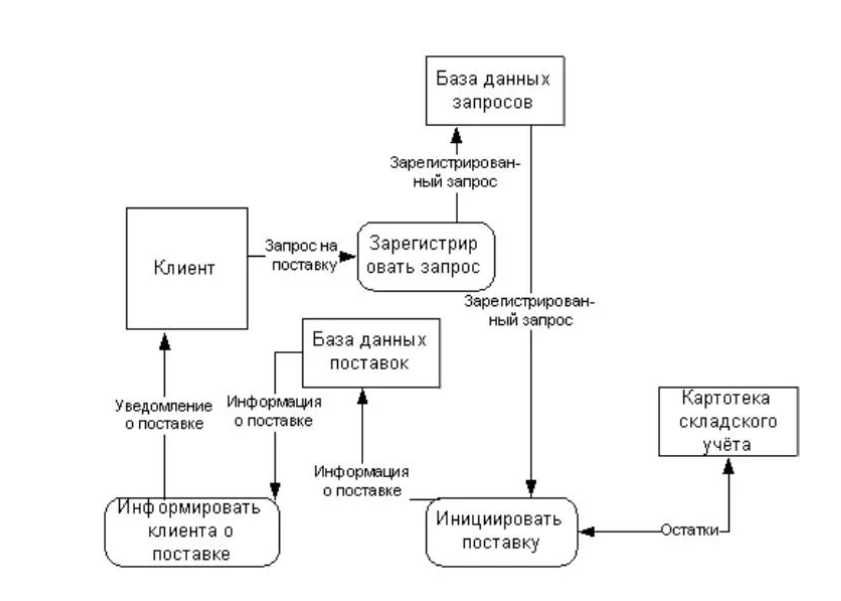


Рисунок 2 – Пример DFD диаграммы

Метод моделирования процессов IDEF3 (Integrated DEFinition for Process Description Capture Method) - методология моделирования и стандарт документирования процессов, происходящих в системе. Метод документирования технологических процессов предоставляет механизм документирования и сбора информации о процессах. IDEF3 показывает причинно-следственные связи между ситуациями и событиями в понятной эксперту форме, используя структурный метод выражения знаний о том, как функционирует система, процесс или предприятие.

Техника описания набора данных IDEF3 является частью структурного анализа. В отличие от некоторых методик описаний процессов IDEF3 не ограничивает аналитика чрезмерно жесткими рамками синтаксиса, что может привести к созданию неполных или противоречивых моделей может быть также использован как метод создания процессов. IDEF3 дополняет IDEFO и содержит все необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа. Пример IDEF3 диаграммы представлен на рисунке 3.

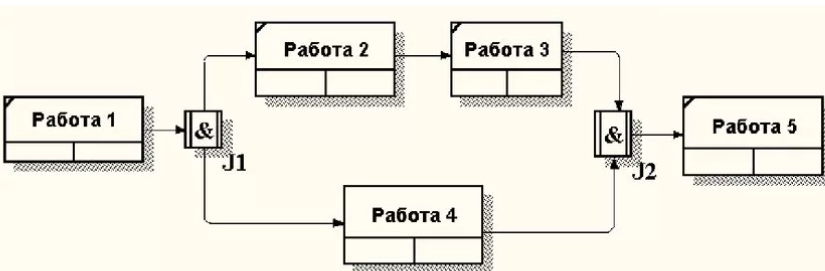


Рисунок 3 – Пример IDEF3 диаграммы

# **Методы моделирования данных**

Методы моделирования данных – это различные методы, которые можно использовать для создания различных моделей данных. Эти подходы развивались с течением времени в результате инноваций в концепциях баз данных и управления данными. Далее представлены основные типы моделирования данных.

Иерархические модели данных представляют отношения «один ко многим» в древовидном формате. В модели этого типа каждая запись имеет единственный корень или родительский элемент, который сопоставляется с одной или несколькими дочерними таблицами. Эта модель была реализована в IBM Information Management System (IMS) ​​в 1966 году и быстро нашла широкое применение, особенно в банковской сфере. Хотя этот подход менее эффективен, чем недавно разработанные модели баз данных, он все еще используется в системах расширяемого языка разметки (XML) и географических информационных системах (ГИС).

При иерархическом моделировании данных можно представить отношения между различными элементами данных в древовидном формате. Иерархические модели данных представляют собой отношения «один ко многим», когда родительские или корневые классы данных отображаются на несколько дочерних.

Иерархическое моделирование данных со временем превратилось в графовое. Графовые модели данных представляют отношения данных, в которых сущности рассматриваются одинаково. Сущности могут связываться друг с другом отношениями «один ко многим» или «многие ко многим» без понятия «родительский» или «дочерний».

Реляционные модели данных были предложены исследователем IBM Э. Ф. Коддом в 1970 году. Они до сих пор встречаются во многих реляционных базах данных, обычно используемых в корпоративных вычислениях. Реляционное моделирование не требует детального понимания физических свойств используемого хранилища данных. В нем сегменты данных объединяются с помощью таблиц, что упрощает базу данных.

Моделирование данных «сущность-связь» (ER) использует формальные диаграммы для представления отношений между сущностями в базе данных. Архитекторы данных используют несколько инструментов ER-моделирования для представления данных.

В объектно-ориентированном программировании для хранения данных используются структуры данных, называемые объектами. Эти объекты данных являются программными абстракциями объектов реального мира. Например, в объектно-ориентированной модели данных автосалон будет иметь такие объекты данных, как «Клиенты», с такими атрибутами, как имя, адрес и номер телефона. Вы будете хранить данные о клиентах таким образом, чтобы каждый реальный клиент был представлен в виде объекта данных о клиенте.

Современные корпоративные вычисления используют технологию хранилищ данных для хранения больших объемов данных для аналитики. Вы можете использовать проекты многомерного моделирования данных для их высокоскоростного хранения и извлечения из хранилища. Многомерные модели используют дублирование или избыточные данные и отдают приоритет производительности, а не использованию меньшего пространства для хранения данных.

# **Процесс моделирования данных**

Моделирование данных начинается с договоренности о том, какие символы используются для представления данных, как размещаются модели и как передаются бизнес-требования. Это формализованный рабочий процесс, включающий ряд задач, которые должны выполняться итеративно. Сам процесс обычно выглядят так:

1 Определите сущности. На этом этапе идентифицируем объекты, события или концепции, представленные в наборе данных, который необходимо смоделировать. Каждая сущность должна быть целостной и логически отделенной от всех остальных.

2 Определите ключевые свойства каждой сущности. Каждый тип сущности можно отличить от всех остальных, поскольку он имеет одно или несколько уникальных свойств, называемых атрибутами. Например, сущность «клиент» может обладать такими атрибутами, как имя, фамилия, номер телефона и т.д. Сущность «адрес» может включать название и номер улицы, город, страну и почтовый индекс.

3 Определите связи между сущностями. Самый ранний черновик модели данных будет определять характер отношений, которые каждая сущность имеет с другими. В приведенном выше примере каждый клиент «живет по» адресу. Если бы эта модель была расширена за счет включения сущности «заказы», ​​каждый заказ также был бы отправлен на адрес. Эти отношения обычно документируются с помощью унифицированного языка моделирования (UML).

4 Полностью сопоставьте атрибуты с сущностями. Это гарантирует, что модель отражает то, как бизнес будет использовать данные. Широко используются несколько формальных шаблонов (паттернов) моделирования данных. Объектно-ориентированные разработчики часто применяют шаблоны для анализа или шаблоны проектирования, в то время как заинтересованные стороны из других областей бизнеса могут обратиться к другим паттернам.

5 Назначьте ключи по мере необходимости и определите степень нормализации. Нормализация — это метод организации моделей данных, в которых числовые идентификаторы (ключи) назначаются группам данных для установления связей между ними без повторения данных. Например, если каждому клиенту назначен ключ, этот ключ можно связать как с его адресом, так и с историей заказов, без необходимости повторять эту информацию в таблице с именами клиентов. Нормализация помогает уменьшить объем дискового пространства, необходимого для базы данных, но может сказываться на производительности запросов.

6 Завершите и проверьте модель данных. Моделирование данных — это итеративный процесс, который следует повторять и совершенствовать под потребности бизнеса.

# **Системы управления базами данных**

Система управления базами данных (СУБД) — это программное обеспечение, предназначенное для хранения, извлечения и управления данными. Наиболее распространенной СУБД в системе баз данных предприятия является СУРБД. Полная форма СУБД — это система управления реляционными базами данных.

Согласно реляционной модели Э. Ф. Кодда, СУБД позволяет пользователям создавать, обновлять, управлять реляционной базой данных и взаимодействовать с ней, позволяя хранить данные в табличной форме.

Система управления реляционными базами данных организует данные в таблицах, которые могут быть связаны внутри в зависимости от общих данных. Это позволяет пользователю легко получить одну или несколько таблиц с помощью всего одного запроса. С другой стороны, в плоском файле данные хранятся в единой табличной структуре, что менее эффективно и требует больше места и памяти.

Наиболее коммерчески доступной и общекорпоративной системой управления базами данных или системой управления реляционными базами данных, используемой сегодня, является язык структурированных запросов (база данных *SQL*) для доступа к базе данных.  
Другие широко используемые системы управления реляционными базами данных в компаниях включают базу данных *Oracle*, *MySQL*, *PostgreSQL* (реляционная база данных с открытым исходным кодом) и *Microsoft* *SQL* *Server*. Структуры РСУБД обычно используются для выполнения четырех основных операций: *CRUD* (создание, чтение, обновление и удаление), которые имеют решающее значение для поддержки согласованного управления данными.

# **2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

# **2.1 Язык программироавния *Python***

Python – это высокоуровневый интерпретируемый язык программирования, который был создан в конце 1980-х годов в Нидерландах Гвидо ван Россумом. Он имеет простой и понятный синтаксис, что делает его очень популярным среди начинающих программистов и обеспечивает быстрое и легкое создание программ. Основные особенности языка представлены ниже.

Интерпретируемый язык: Python – это интерпретируемый язык, что означает, что он не компилируется в машинный код, а выполняется интерпретатором.

Кросс-платформенность: Python поддерживает большое количество операционных систем, таких как Windows, Linux, macOS и многие другие.

Простой синтаксис: Python имеет простой и читаемый синтаксис, что делает его легко изучаемым для начинающих программистов.

Обширная стандартная библиотека: Python поставляется с обширной стандартной библиотекой, которая содержит множество модулей, что позволяет программистам решать широкий спектр задач.

Динамическая типизация: В Python не нужно объявлять типы данных, что облегчает и ускоряет процесс написания кода.

Объектно-ориентированное программирование: Python поддерживает объектно-ориентированное программирование, что позволяет создавать классы и объекты, что упрощает разработку крупных проектов.

Основные возможности языка представлены на рисунке 2.2.



Рисунок 4 – Основные возможности использования Python

# **2.2 *PyCharm***

PyCharm – это интегрированная среда разработки (*IDE*), разработанная компанией *JetBrains*, которая специализируется на предоставлении инструментов для разработки на языке программирования *Python*. Эта *IDE* пользуется заслуженной популярностью среди разработчиков *Python* благодаря своей мощной функциональности и удобству использования.

Одной из ключевых особенностей *PyCharm* является его возможность поддержки веб-приложений. Это делает *PyCharm* актуальным инструментом для разработчиков, занимающихся веб-разработкой и взаимодействием с базами данных. Далее рассмотрены основные возможности данной среды разработки.

*PyCharm* обеспечивает удобную настройку проекта для веб-разработки. Разработчики могут создавать проекты *Python* с заранее настроенными параметрами, что значительно упрощает начало работы. Кроме того, *IDE* предоставляет интеграцию с популярными веб-фреймворками, такими как *Django* и *Flask*. Это облегчает процесс создания веб-приложений и их настройку, включая взаимодействие с базой данных.

Однако одним из наиболее значимых аспектов использования *PyCharm* для веб-приложений является его интеграция с системами управления базами данных (СУБД), включая *PostgreSQL*, *MySQL*, *SQLite* и другими. Это означает возможность легко управлять базами данных, создавать и изменять схему, выполнять *SQL*-запросы и многое другое, прямо из *IDE*. Эта интеграция значительно упрощает взаимодействие веб-приложения с базой данных и обеспечивает более эффективную разработку.

*PyCharm* также предоставляет инструменты для отладки и профилирования веб-приложений, что помогает выявлять и устранять ошибки и оптимизировать производительность приложений. Важно отметить, что *IDE* обладает автоматизированными средствами тестирования, что способствует обеспечению качества кода и приложения в целом.

*PyCharm* – это мощное и универсальное средство для разработки веб-приложений, особенно тех, которые используют проектированную базу данных. Его интеграция с веб-фреймворками, удобное управление базами данных и богатые инструменты разработки делают его незаменимым помощником для разработчиков, стремящихся создать эффективные и надежные веб-приложения.

**2.3 *PostgreSQL***

PostgreSQL, также известная как *Postgres*, является мощной и открытой системой управления реляционными базами данных, которая пользуется высокой популярностью в мире разработки приложений и анализа данных. Основанный на модели объектно-реляционной базы данных (*ORDBMS*), *PostgreSQL* обеспечивает надежное и гибкое хранение и обработку данных для разнообразных приложений.

*PostgreSQL* имеет открытый исходный код, что делает его привлекательным для широкого круга разработчиков и организаций. СУБД поддерживает мощный *SQL*, включая сложные запросы и агрегатные функции. Расширяемость *PostgreSQL* позволяет легко добавлять новые типы данных и функциональность.

СУБД обладает встроенной поддержкой *JSON*-данных, что полезно для работы с полуструктурированными данными. Она также гарантирует *ACID*-совместимость, обеспечивая надежность и целостность данных.

*PostgreSQL* поддерживает многопоточность и параллелизм для высокой производительности и масштабируемости. Он также предоставляет инструменты для настройки репликации и обеспечения отказоустойчивости.

Активное сообщество разработчиков и пользователей обеспечивает поддержку, обновления и регулярное выпуск новых версий. *PostgreSQL* – это мощная и гибкая СУБД, подходящая для различных задач от веб-приложений до анализа данных.

# **3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ**

# **4 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ**

# **5 АРХИТЕКТУРА ВЕБ-ПРИЛОЖЕНИЯ**

# **ЗАКЛЮЧЕНИЕ**

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**