Министерство науки и высшего образования

Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана» (национальный исследовательский университет)»

Московский техникум космического приборостроения

|  |  |  |
| --- | --- | --- |
|  |  | УТВЕРЖДАЮ  Заместитель директора по УР  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.Н. Михайлова  (подпись, дата) |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(ДИПЛОМНЫЙ ПРОЕКТ)

Разработка и администрирование информационной системы «Учет компьютерной техники» в АО «Корпорация ВНИИЭМ»

Пояснительная записка

Группа ТБД-81

|  |  |  |
| --- | --- | --- |
| Председатель предметной (цикловой) комиссии | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | Е.С. Демина\_\_\_\_\_  (ФИО) |
| Руководитель разработки от техникума | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | Е.С. Демина\_\_\_\_\_  (ФИО) |
| Рецензент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    (ФИО) |
| Руководитель разработки от предприятия | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ФИО) |
| Консультант по экономической части | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | И.В. Протоерескул    (ФИО) |
| Разработчик | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    (ФИО) |

Москва 2024

СОДЕРЖАНИЕ

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 3](#_Toc160991245)

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ 4](#_Toc160991246)

[ВВЕДЕНИЕ 5](#_Toc160991247)

[1 Теоретическая часть 8](#_Toc160991248)

[1.1 Технико–экономическая характеристика процесса приема и обработки заявок 8](#_Toc160991249)

[1.1.1 Характеристика предприятия АО «Корпорация ВНИИЭМ» 8](#_Toc160991250)

[1.1.2 Краткая характеристика технического отдела «41», его видов деятельности 9](#_Toc160991251)

[1.2 Концептуальное моделирование процесса учета компьютерной техники 12](#_Toc160991252)

[1.3 Постановка задачи на разработку и администрирование информационной системы 14](#_Toc160991253)

[1.4 Анализ существующих разработок и обоснование выбора технологии разработки и администрирования информационной системы 15](#_Toc160991254)

[1.5 Выводы по первой главе 16](#_Toc160991255)

[ЛИТЕРАТУРА 17](#_Toc160991256)

# ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Таблица 1 – Термины и определения

|  |  |
| --- | --- |
| Термин | Определение |
| Программное обеспечение | Программа или множество программ, используемых для управления компьютером или техникой, для  которой оно предназначено |
| Программа | Последовательность действий по заданному  алгоритму |
| Система управления базами данных | Совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и  использованием баз данных |

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

Таблица 2 – Перечень сокращений

|  |  |
| --- | --- |
| Сокращение | Расшифровка |
| ГБ | Гигабайт |
| ИБП | Источник бесперебойного питания |
| СУБД | Система управления базами данных |
| RAID | Redundant Array of Independent Disks |
| SQL | Structured Query Language |
| YAML | Ain't Markup Language |
| Wi–Fi | Wireless Fidelity – беспроводное качество |

# ВВЕДЕНИЕ

В современном информационном обществе одним из ключевых направлений является эффективное управление и контроль за компьютерной техникой в организации. Именно поэтому разработка модуля «Учет компьютерной техники» информационной системы становится актуальной задачей для предприятий, стремящихся к оптимизации бизнес-процессов и повышению уровня безопасности и производительности.

В данном дипломном проекте будет проведен анализ необходимости и практической целесообразности внедрения указанного модуля в АО «Корпорация ВНИИЭМ», а также разработаны и реализованы соответствующие функциональные и технические требования. Особое внимание будет уделено разработке удобного механизма управления данными, позволяющих эффективно контролировать состояние и распределение компьютерной техники, проводить анализ ее использования и планировать необходимые ресурсы для обеспечения бесперебойной работы организации. К актуальным причинам разработки подобного модуля информационной системы можно отнести:

* Оптимизация процесса учета компьютерной техники. Ручной учет техники может быть неэффективным и времязатратным процессом, который подвержен ошибкам. Внедрение специализированного модуля позволит автоматизировать процесс учета и значительно упростит его для сотрудников организации;
* повышение безопасности. Контроль за распределением компьютерной техники и ее использованием позволит предотвратить утрату или несанкционированное использование ценного оборудования, а также обеспечить защиту конфиденциальных данных и информации;
* повышение производительности. Анализ использования компьютерной техники позволит выявить оборудование, которое не используется эффективно, и оптимизировать его размещение или замену. Это позволит улучшить производительность сотрудников и сократить издержки на обслуживание неиспользуемого оборудования;

Концепция разработки и администрирования баз данных исследовалась различными специалистами и исследовательскими группами в области информационных технологий. Ниже приведены некоторые известные исследования и их авторы: Совместный опыт четырех специалистов Techies Gokulkumar Radhakrishnan, Sundaravel Ramasubbu, Arun Kumar Kubendiran, и Mahendran Manickam описанный в книге «Just Relax DBA» описывает современные методы администрирования [1]. Научная статья по компьютерным и информационным наукам «Технология разработки баз данных информационных систем», авторы научной работы — Грачев В.М., Есин В.И., Полухина Н.Г., Гассомахин С.Г. [2]. В данной статье рассматривается проблема создания необходимой технологии, позволяющей осуществлять эффективную разработку корпоративных баз данных. В качестве решения предлагается новая технология, основывающаяся на семантической модели «объект-событие», использовании универсальной модели данных, языка модели данных, специального программного инструментария разработчика базы данных. Книга Джордана Краузе «Windows Server 2022: комплексное администрирование среды Windows Server, 4-е издание» Она предоставляет справочник по администрированию в Windows Server, который поможет спроектировать и внедрить Microsoft Server 2022 в корпоративной среде и использовать его для создания безопасных и эффективных сетей [3]. при написании работы будут использоваться данные источники и опираясь на них будут выведены некоторые решения.

Таким образом, разработка модуля «Учет компьютерной техники» в информационной системе представляет собой важную задачу для повышения эффективности работы организации и обеспечения надежного управления компьютерным оборудованием.

Объектом исследования ВКР является процесс учета компьютерной техники в АО «Корпорация ВНИИЭМ».

Предметом исследования ВКР является модуль информационной системы «Учет компьютерной техники».

Целью ВКР является проектирование и разработка модуля «Учет компьютерной техники» в АО «Корпорация ВНИИЭМ». Для достижения поставленной цели необходимо решить следующие задачи:

− Провести анализ предметной области с целью выявления процессов клиентов, требующих автоматизации;

− описать функциональные требования к проектируемой системе;

− провести анализ средств разработки приложения и базы данных;

− построить модель данных на логическом и физическом уровнях;

− разработать информационную систему (ИС) и описать принципы работы с ней;

− оценить себестоимость информационной системы и целесообразность ее разработки.

Структура ВКР представлена введением, пятью главами, заключением, списком использованной литературы и приложениями. В первой главе дано описание деятельности АО «Корпорация ВНИИЭМ», представлена ее организационная структура. Описан существующий процесс учета компьютерной техники и его недостатки. Разработана модель автоматизированного процесса учета компьютерной техники и сформулированы требования к проектируемой ИС по видам обеспечения. Во второй главе разработаны логическая и физическая модель данных исследуемой предметной области. Разработан и введен в эксплуатацию модуль «Учет компьютерной техники», а также описана его функциональность. Третья глава посвящена оценке себестоимости и экономической целесообразности разработки модуля системы.

# Теоретическая часть

## Технико–экономическая характеристика процесса приема и обработки заявок

### Характеристика предприятия АО «Корпорация ВНИИЭМ»

АО «Корпорация ВНИИЭМ» — это крупное научно-исследовательское предприятие, специализирующееся в области разработки электронных и микроволновых приборов, аппаратуры и систем для оборонно-промышленного комплекса. На рисунке 1.1 изображена организационная структура АО «Корпорация ВНИИЭМ».



Рисунок 1.1 - Организационная структура АО «Корпорация ВНИИЭМ»

Предприятие имеет богатый опыт и отличную репутацию на рынке за счет своего высокого профессионализма, инновационных технологий и высокого уровня качества продукции. «Корпорация ВНИИЭМ» постоянно совершенствует свои продукты и внедряет новейшие разработки, что позволяет быть на передовой позиции в своей отрасли.

Предприятие также активно сотрудничает с ведущими научными центрами и университетами, что способствует развитию науки и технологий в стране.

«Корпорация ВНИИЭМ» стремится к постоянному росту и развитию, сохраняя свои высокие стандарты производства и обслуживания клиентов.

### Краткая характеристика технического отдела «41», его видов деятельности

Технический отдел «41» является ключевым подразделением в организации, отвечающим за поддержание и развитие технической базы компании. В его компетенции входит широкий спектр работ, направленных на обеспечение бесперебойной работы оборудования, а также на совершенствование и модернизацию технических систем. На рисунке 1.2 изображена структура технического отдела.



Рисунок 1.2 – Структура технического отдела

Технический отдел «41» осуществляет широкий спектр деятельности, включающий в себя следующие виды работ:

* Разработка технических решений: создание новых технических концепций, проектирование систем, устройств, программ и других технических продуктов;
* тестирование: проведение испытаний и проверок оборудования, программного обеспечения и других технических решений для оценки их работоспособности, соответствия спецификациям и стандартам качества;
* внедрение технических решений: внедрение новых технологий, программных продуктов и оборудования в рабочие процессы организации;
* учет техники: ведение учета основных средств, оборудования и техники, контроль их состояния и эксплуатационных параметров;
* мониторинг работы оборудования: постоянный контроль за работой и производительностью технического оборудования с целью оперативного выявления неисправностей и проведения своевременного ремонта;
* обслуживание оборудования: проведение регулярных технических осмотров, предотвращение и устранение проблемных ситуаций, профилактическое обслуживание оборудования;
* проведение профилактических мероприятий: планомерное проведение мероприятий по предупреждению отказов и повышению надежности работы оборудования;
* ремонт оборудования: выполнение технических работ по восстановлению работоспособности оборудования, устранению неисправностей;
* контроль безопасности технических работ: осуществление проверок и обеспечение безопасности выполнения технических операций и процессов.

На рисунке 1.3 изображена диаграмма прецендентов работы технического отдела.

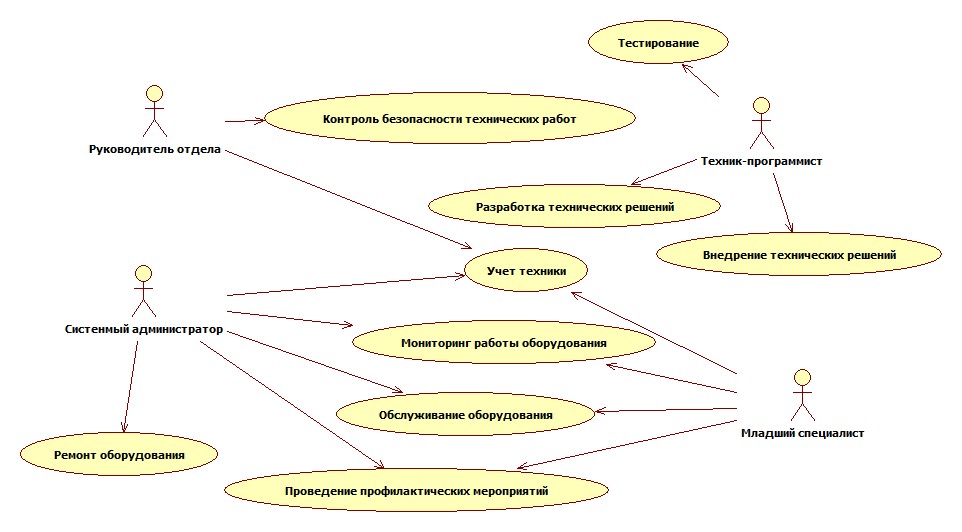


Рисунок 1.3 – Диаграмма прецендентов

Для обеспечения эффективной работы технического отдела необходимо проводить учет техники, который включает в себя следующие шаги:

* + - Идентификация и классификация техники - необходимо иметь данные, в которых будут указаны все виды техники, ее модели, серийные номера, даты покупки, срок службы и другие характеристики;
    - ведение журнала учета техники - важно вести документацию о перемещениях техники, ремонтах, проведенных технических обслуживаниях и заменах деталей. Это позволит оперативно отслеживать состояние оборудования и предотвращать возможные проблемы;
    - планирование технического обслуживания - на основе данных о сроках службы и рекомендаций производителей необходимо разработать план регулярного технического обслуживания оборудования. Это поможет предотвратить непредвиденные поломки и снизить риск простоев в производстве;
    - анализ эффективности использования техники - регулярно следует оценивать производительность оборудования, его загрузку и эффективность работы. По результатам анализа можно принимать меры по оптимизации производственных процессов и повышению производительности.

В настоящее время существует большая проблема при поиске и использовании данных, так как все процессы происходят изолировано друг от друга. Таким образом мы сталкиваемся с определенными сложностями: данные хранятся в разных местах, поэтому при их сборе приходится обращаться к разным источникам, что может привести к потере информации, ошибкам при анализе и сопоставлении, и, соответственно, такой подход занимает очень много времени. На рисунке 1.4 изображена диаграмма последовательности получения отчета руководителем.

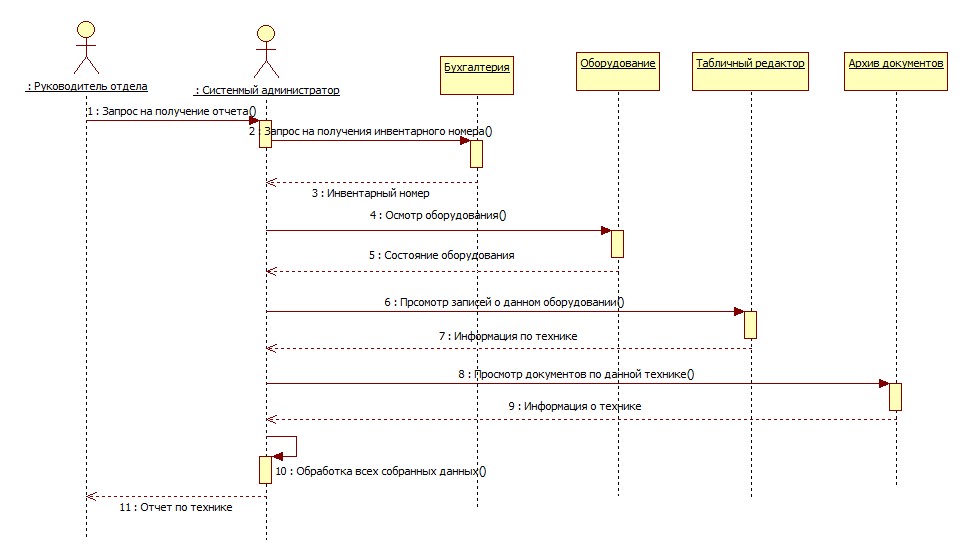


Рисунок 1.4 – Диаграмма последовательности

Процесс сбора данных требует большого количества времени. Обычно для этого необходимо посылать несколько запросов в разные места, и только потом объединять полученные данные. Для решения этой проблемы мы сделаем учет техники более эффективным путем автоматизации этих процессов. Теперь для сбора, сопоставления и анализа данных об оборудовании достаточно одного запроса, так как все данные о техническом оборудовании будут храниться в одном месте. Таким образом мы сократим время и минимизирует риск ошибок.

## Концептуальное моделирование процесса учета компьютерной техники

Чтобы вести учет технических средств, необходимо располагать данными о самой технике и работнике, к которому она закреплена. В корпоративных нормативах прописаны правила отчетности оборудования. Для осуществления этой процедуры требуется активное участие персонала и использование табличных приложений, содержащих всю необходимую информацию о технических устройствах. На рисунке 1.5 изображена контекстная диаграмма IDEF0 модели AS-IS.

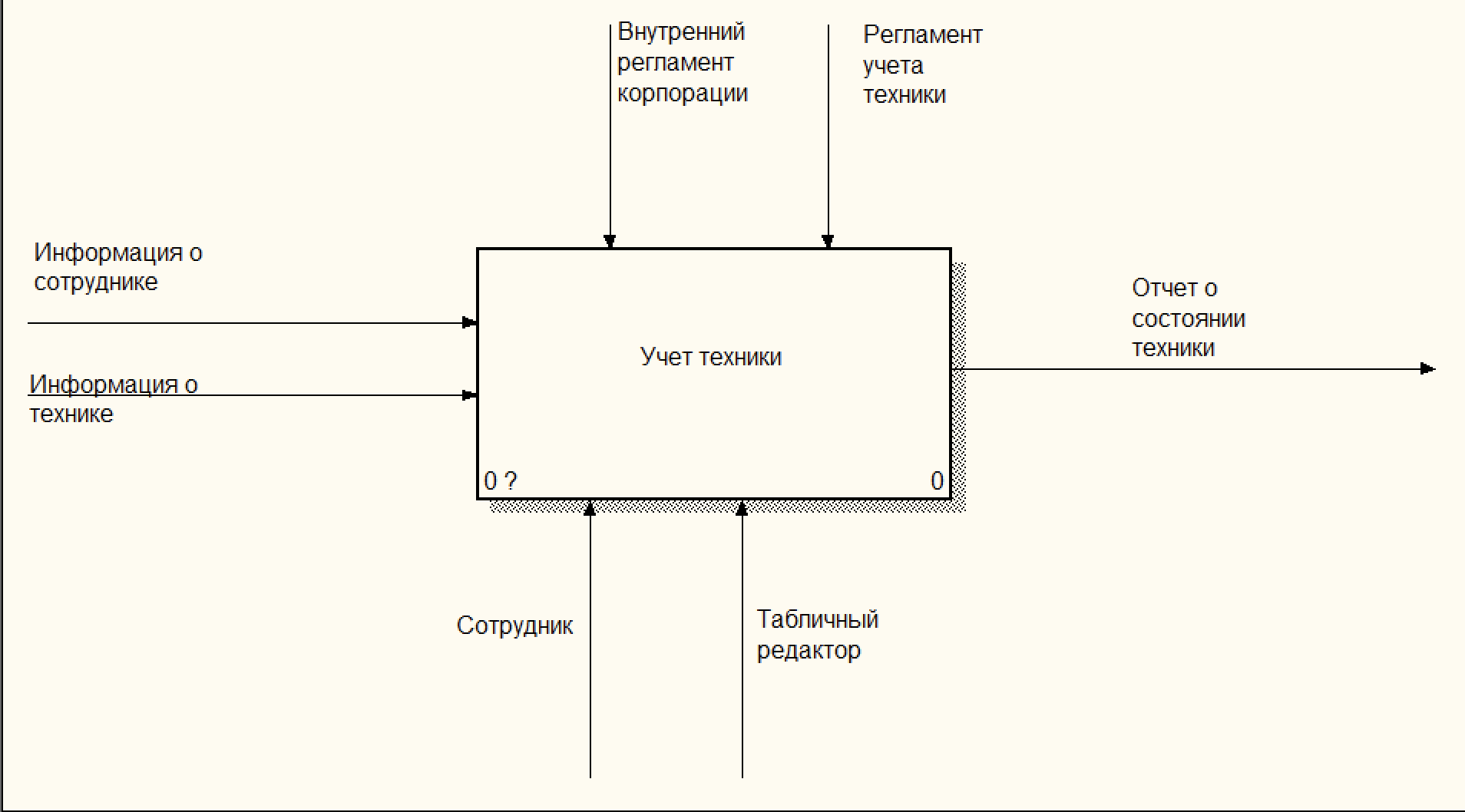


Рисунок 1.5 – Контекстная диаграмма IDEF0 модели AS-IS

В самом начале процесса учета производится тщательный осмотр оборудования, в ходе которого сверяются сведения о технике и её ответственном сотруднике, сравнивая их с данными, полученными во время осмотра. Затем информация вносится в соответствующую форму электронной таблицы, на основе которой составляется подробный отчет о текущем состоянии технического оборудования. Все этапы процедуры предполагают участие сотрудника компании. На рисунке 1.6 изображена декомпозиция контекстной диаграммы IDEF0 модели AS-IS.

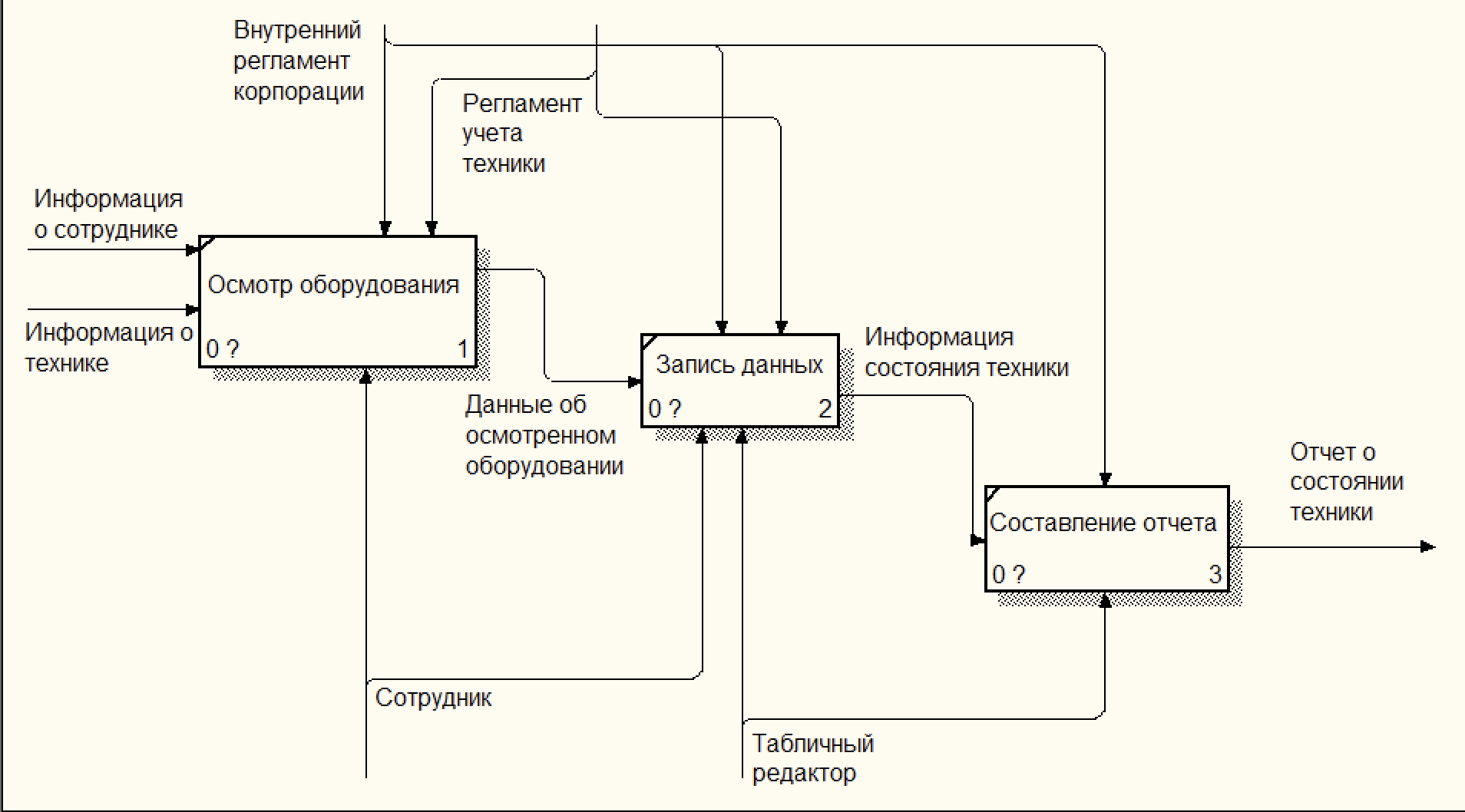


Рисунок 1.6 – Декомпозиция контекстной диаграммы IDEF0 модели AS-IS

После автоматизации предполагается использование информационной системы вместо табличного редактора. На рисунке 1.7 изображена контекстная диаграмма IDEF0 модели TO–BE.

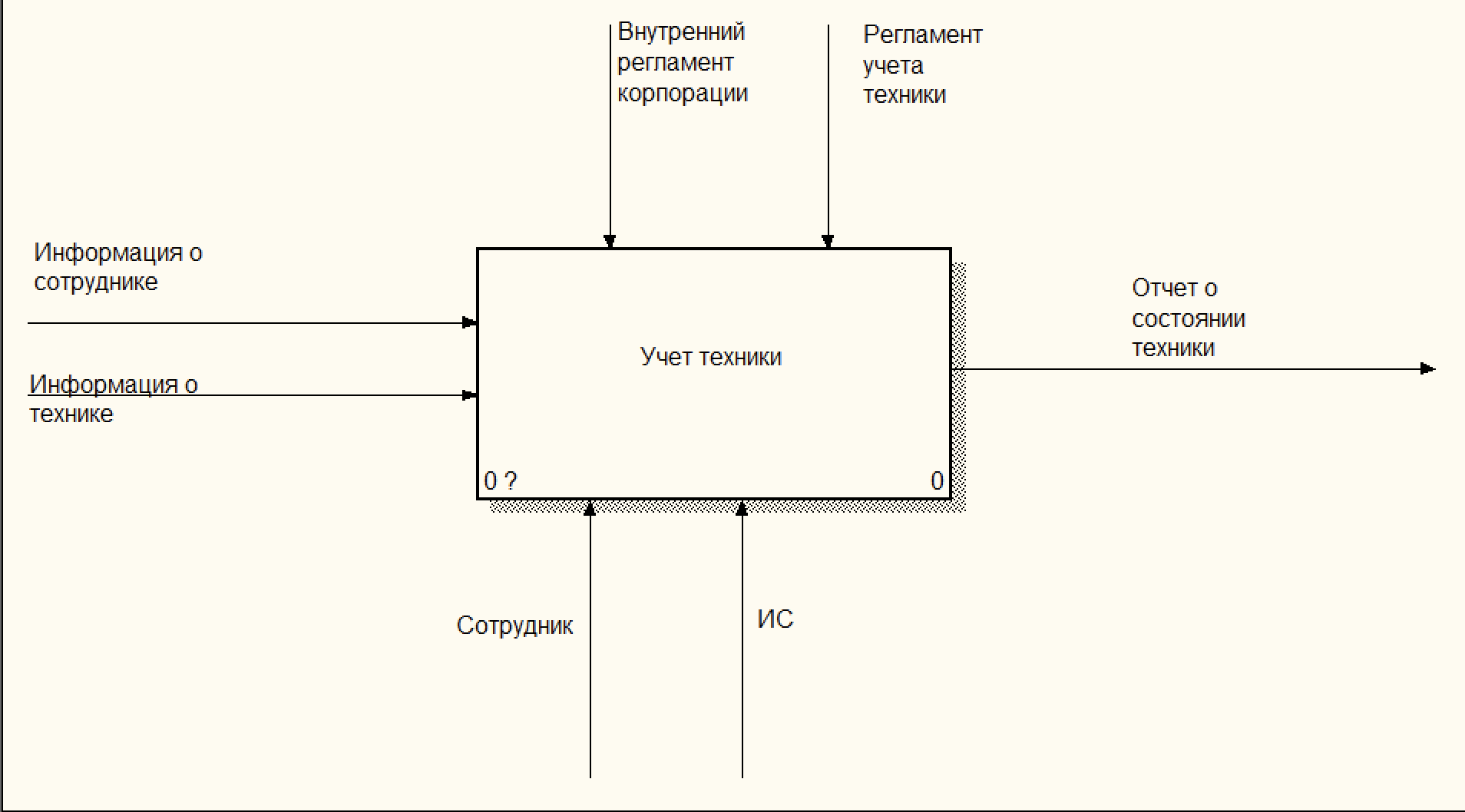


Рисунок 1.7 – Контекстная диаграмма IDEF0 модели TO–BE

После автоматизации процессов участие сотрудника в составлении отчета становится излишним, что благоприятно сказывается на эффективности работы отдела. На рисунке 1.8 изображена декомпозиция контекстной диаграммы IDEF0 модели TO–BE.

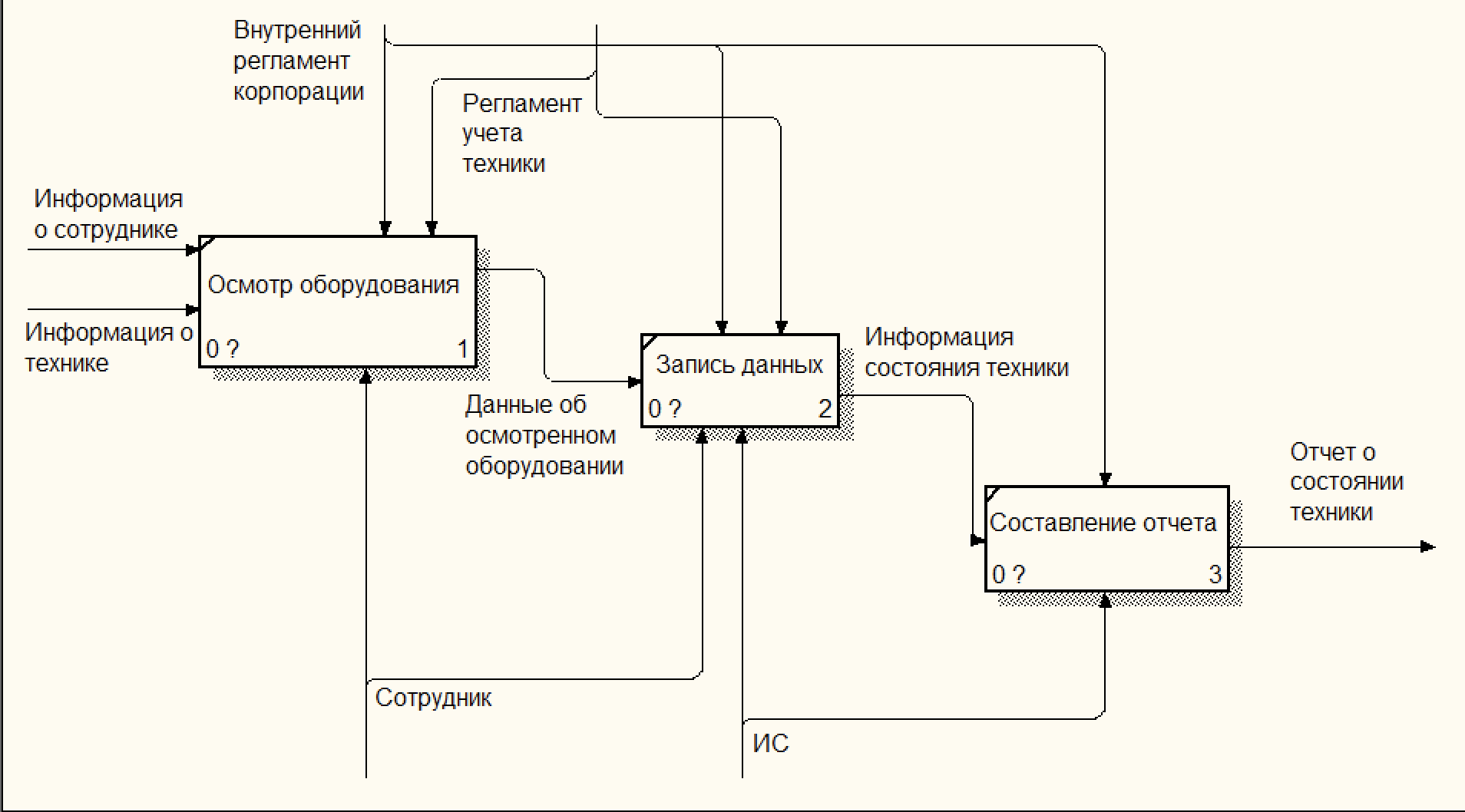


Рисунок 1.8 – Декомпозиция контекстной диаграммы IDEF0 модели TO–BE

Итак, эти диаграммы иллюстрируют, какие процессы будут затронуты внедрением информационной системы, и как она поможет оптимизировать работу отдела и учет техники.

## Постановка задачи на разработку и администрирование информационной системы

Постановка задачи на разработку и администрирование информационной системы «Учет компьютерной техники» в АО «Корпорация ВНИИЭМ».

Цели и задачи системы:

* Создание централизованной системы учета компьютерной техники в АО «Корпорация ВНИИЭМ»;
* автоматизация процессов инвентаризации, перемещения и списания компьютерной техники;
* обеспечение оперативного доступа к актуальной информации о состоянии компьютерного парка;
* улучшение контроля за использованием компьютерной техники.

Функциональные требования. Система должна обеспечивать следующие функции:

* Регистрация и учет компьютерной техники (инвентаризация);
* учет перемещений техники между подразделениями и сотрудниками;
* учет списания и утилизации техники;
* формирование отчетов по различным критериям (подразделения, сотрудники, модели техники);
* импорт и экспорт данных в различных форматах;
* интеграция с другими информационными системами (например, системой учета техники на складе).

Технические требования:

* Система должна быть разработана на современной технологической платформе;
* должна обеспечиваться высокая производительность и масштабируемость системы;
* система должна быть защищена от несанкционированного доступа;
* должен быть обеспечен удобный и интуитивно понятный пользовательский интерфейс.

Администрирование системы:

* Назначение ролей и прав доступа для пользователей;
* контроль работоспособности системы;
* резервное копирование и восстановление данных.

## Анализ существующих разработок и обоснование выбора технологии разработки и администрирования информационной системы

После определения целей проекта наступает момент выбора между веб-приложением и прикладным программным обеспечением. В данном случае выбор падает на веб-приложение по нескольким причинам.

Веб-приложение не требует использования ресурсов компьютера, так как запускается на удаленном сервере. Это значит, что пользователи могут получить доступ к интерфейсу через локальную сеть, не загружая свои компьютеры и не тратя ресурсы.

Кроме того, нет необходимости устанавливать приложение на каждый компьютер отдельно. Поскольку веб-приложение доступно через сеть, все пользователи могут просто зайти на сайт и начать пользоваться функционалом без дополнительных действий.

Также стоит отметить, что все пользователи подключены к одной локальной сети, что упрощает процесс взаимодействия между ними. Это позволяет обеспечить более гибкую и удобную работу для пользователей, так как они могут использовать приложение с любого компьютера, имеющего доступ к сети.

В итоге, выбор в пользу веб-приложения обусловлен удобством использования, экономией ресурсов и простотой распространения среди пользователей.

Ключевой аспект сравнительного анализа - используемые технологии веб-разработки. Рассмотрим популярные языки программирования, фреймворки и инструменты, доступные в каждой из рассматриваемых сред [4]. Необходимо провести краткое описание каждого из языков программирования.

JavaScript – один из самых востребованных языков программирования для разработки веб-приложений. Он используется для написания front-end части веб-приложений и взаимодействия с браузером. Благодаря обилию библиотек и фреймворков, таких как React, Angular и Vue, JavaScript становится мощным инструментом для создания интерактивных и динамических веб-приложений.

Python – язык программирования общего назначения, широко применяемый для веб-разработки. Его простой и понятный синтаксис пользуется популярностью среди новичков. Фреймворки для веб-разработки, такие как Django и Flask, делают Python доступным и удобным для создания веб-приложений.

Ruby – динамический язык программирования, также часто используемый для веб-разработки. Простой и интуитивно понятный синтаксис делает Ruby популярным среди разработчиков. Ruby on Rails – популярный фреймворк для создания веб-приложений, упрощающий процесс разработки.

PHP – специализированный язык программирования для создания веб-приложений. Большое количество библиотек и фреймворков, таких как Symfony и Laravel, делают PHP мощным инструментом для разработки веб-приложений.

Среди основных инструментов для веб-разработки наиболее популярными и конкурирующими являются Django [6], Node.js [7] и PHP [8]. Django — это веб-фреймворк на языке Python, в то время как Node.js — это среда выполнения JavaScript. Все три инструмента являются открытыми и созданы для обеспечения полнофункциональной разработки на одном программном языке.

Проведем сравнительную характеристику по следующим пунктам.

1. Производительность.

Django в зависимости от специфики проекта и оптимизации кода, производительность может меняться. Node.js благодаря асинхронной модели выполнения, может быть более производительным в сравнении с Django в некоторых случаях. PHP, с другой стороны, изначально создавался для обработки веб-страниц, поэтому он обладает высокой производительностью при работе с веб-сайтами и веб-приложениями. PHP имеет встроенный обработчик HTML, что ускоряет работу с контентом и уменьшает время отклика сервера. Однако, как и в случае с другими языками, производительность PHP зависит от оптимизации кода и архитектуры проекта.

1. Масштабируемость.

В Django разработчики создают каждую функцию автономно, а затем устанавливают связи между ними. Это позволяет легко реализовать и усовершенствовать новые функции, не беспокоясь о том, как их интегрировать в приложение. Фреймворк также может грамотно обрабатывать большой трафик и значительные объемы данных, что делает все приложения готовыми к масштабированию с самого начала.

Node.js также обладает хорошей масштабируемостью благодаря возможности создания микросервисов и использования кластеризации. Его однопоточная модель цикла событий способна обрабатывать даже миллион одновременных подключений. Благодаря мощным инструментам, таким как Cluster и PM2 process manager, Node.js упрощает оптимизацию веб-приложений.

В то время PHP, как правило, используется для создания динамических веб-страниц, его масштабируемость может быть ограничена. В отличие от Django и Node.js, PHP не предоставляет таких мощных инструментов для масштабирования и оптимизации. Высокий трафик и большие объемы данных могут привести к замедлению работы PHP-приложений, что может потребовать значительных усилий по оптимизации кода и настройке сервера.

1. Полнота.

Django включает в себя множество готовых компонентов и библиотек, что делает его довольно полным фреймворком для веб-разработки.

Node.js более гибкий и позволяет разработчику выбирать собственные библиотеки и инструменты, что может быть как плюсом, так и минусом в зависимости от потребностей проекта.

PHP также является довольно полным языком программирования для веб-разработки, благодаря большому количеству библиотек и расширений, доступных для него. Однако, в отличие от Django, PHP не является фреймворком, а является языком программирования, который может быть использован для создания веб-приложений. Это означает, что разработчик должен самостоятельно выбирать и настраивать библиотеки и инструменты для своего проекта, что может быть более трудоемким, чем использование полного фреймворка.

1. Скорость разработки.

Благодаря простоте языка процесс разработки приложения на Python происходит немного быстрее, чем при использовании JavaScript. Django ставит перед собой главную задачу — обеспечить разработчикам возможность быстрого создания приложений с минимальным объемом кода. Фреймворк предлагает целый набор инструментов для быстрой разработки MVP и дальнейшей масштабируемости. С использованием Node.js разработчики могут полагаться на пакеты npm для решения общих проблем, однако не все из них хорошо поддерживаются, некоторые имеют недостаточно документации или даже содержат ошибки. В целом, PHP является довольно быстрым языком программирования для разработки веб-приложений, но его гибкость может потребовать большего количества ручной настройки и поддержки по сравнению с полными фреймворками, такими как Django.

1. Безопасность.

Создатели Django придают большое значение безопасности, предоставляя встроенные инструменты, такие как предотвращение кликджекинга и защита от SQL-инъекций, которые позволяют опытным разработчикам предотвращать серьезные нарушения.

Node.js также обеспечивает инструменты для обеспечения безопасности благодаря своей обширной библиотеке пакетов, защищенной брандмауэром компании, и дополнительным расширенным функциям безопасности. Это позволяет корпоративным пользователям не беспокоиться о безопасности своего приложения.

В PHP отсутствуют встроенные инструменты безопасности, контроля доступа, защиты от XSS-атак, CSRF-атак и SQL-инъекций. Это означает, что разработчик должен самостоятельно выбирать и настраивать библиотеки и инструменты для своего проекта, что может быть более трудоемким.

1. Эффективность затрат.

Django позволяет разработчикам быстро создавать работающее, компетентное программное обеспечение, что делает его довольно рентабельным по сравнению Node.js. В то время PHP считается менее ресурсоемким языком, который позволяет создавать веб-приложения, требующий меньше вычислительных ресурсов на сервере.

1. База данных.

Django использует ORM (Object-Relational Mapping) для работы с базами данных, что позволяет разработчикам использовать объектно-ориентированный подход к работе с данными, вместо написания SQL-запросов и имеет встроенную административную панель, которая упрощает работу с базой данных, позволяя администраторам добавлять, редактировать и удалять данные без необходимости писать дополнительный код.

Node.js не имеет встроенной поддержки ORM, поэтому разработчики должны использовать сторонние библиотеки, такие как Sequelize или Mongoose, для работы с базами данных, но обладает преимуществом в работе с NoSQL базами данных, такими как MongoDB, благодаря асинхронной архитектуре и возможности работы с большими объемами данных.

PHP также имеет встроенную поддержку для работы с различными типами баз данных, такими как MySQL, PostgreSQL, SQLite и другими. Однако, в отличие от Django, PHP не имеет встроенного ORM, что может усложнять работу с базами данных и требует от разработчиков писать SQL-запросы вручную. В таблице 1.1 проведена сравнительная характеристика инструментов веб-разработки.

Таблица 1.1 – сравнительная характеристика инструментов веб-разработки

|  |  |  |  |
| --- | --- | --- | --- |
|  | Django | Node.js | PHP |
| Производительность | - | + | + |
| Масштабируемость | + | + | - |
| Полнота | + | - | +/- |
| Скорость разработки | + | - | + |
| Безопасность | + | + | - |
| Эффективность затрат | + | - | + |
| База данных | + | +/- | - |

Исходя из описанных характеристик, фреймворк Django является наилучшим выбором для создания веб-приложения по учету техники из-за необходимости обработки больших объемов данных. Простой синтаксис, развитая экосистема библиотек, быстрая разработка, низкие затраты и хорошая совместимость с реляционными базами данных делают его идеальным инструментом для разработки современного и масштабируемого веб-приложения.

Следующим шагом будет проведение сравнительного анализа СУБД и выбора конкретного из них. Сравнительный анализ СУБД MySQL, PostgreSQL и SQLite основывается на различных критериях, таких как производительность, функциональность, надежность и масштабируемость, простота использования. Рассмотрим каждый критерий подробнее в таблице 1.2.

Таблица 1.2 – Сравнительный анализ СУБД.

|  |  |  |  |
| --- | --- | --- | --- |
| СУБД  Критерии | MySQL | PostgreSQL | SQLite |
| Производительность | Высокая производительностью и низкая задержкой, особенно при работе со средними объемами данных. | Обеспечивает хорошую производительность, особенно при сложных запросах и транзакциях | SQLite имеет более низкую производительность по сравнению с MySQL и PostgreSQL, |
| Функциональность | Имеет более ограниченный функционал, но поддерживают основные операции и запросы. | Обладает более широким набором функциональности, включая поддержку географических и пространственных данных, полнотекстового поиска и временных рядов. | Легковесная, встроенная база данных, которая обладает ограниченной функциональностью по сравнению с MySQL и PostgreSQL |
| Надежность и масштабируемость | Имеет некоторые средства для обеспечения надежности и масштабируемости, но в меньшей степени. | Обеспечивает высокую надежность и масштабируемость. Поддерживает репликацию, кластеризацию и высокую доступность. | Отсутствует управление пользователями и функции безопасности. Трудно масштабируется |
| Простота использования | Имеет простой интерфейс и является довольно простой в использовании. | Более сложна в использовании, чем MySQL. требуется более глубокое понимание и опыт работы с СУБД. | Она очень проста в использовании и не требует установки или настройки. |

В результате сравнительного анализа СУБД MySQL, PostgreSQL и SQLite можно сделать следующие выводы.

MySQL – это широко используемая и распространенная СУБД, основным преимуществом которой является простота использования и отличная производительность при работе с данными. Однако MySQL имеет некоторые ограничения, особенно в отношении поддержки транзакций и сжатия данных.

PostgreSQL – это более продвинутая и мощная СУБД, которая поддерживает все основные функции и возможности SQL. PostgreSQL имеет отличную поддержку транзакций, полнотекстового поиска и географических данных. Однако PostgreSQL требует больше ресурсов и сложнее в настройке и управлении, поэтому может быть менее подходящим для небольших проектов.

SQLite – это компактная и быстрая база данных, которая отлично подходит для небольших проектов и мобильных приложений. Однако из-за своей архитектуры, SQLite не является подходящим выбором для крупных и высоконагруженных систем. База данных SQLite не предоставляет механизмов для масштабирования и репликации данных, что делает ее менее надежной для крупных проектов.

В заключение, учитывая отсутствие опытного специалиста по базам данных в компании и необходимость в простом и надежном решении, выбор MySQL кажется наиболее обоснованным. MySQL обладает достаточными возможностями для обработки данных и не требует значительных затрат на обучение и поддержку. Таким образом, использование MySQL для обслуживания данной задачи будет являться правильным решением с учетом потребностей и возможностей.

В данной работе будет использоваться инструмент Docker [9]. Докеризация — это процесс упаковки приложения и всех его зависимостей в контейнер, который можно легко перемещать из одной среды в другую. Docker compose [10] — это инструмент, который позволяет создавать и управлять несколькими контейнерами в рамках одного проекта.

Для веб-приложений, которые работают с базой данных, использование Docker compose для поднятия контейнеров с приложением и базой данных является удобным и практичным решением. С его помощью можно легко настроить и запустить разработанное приложение в любой среде, не беспокоясь о настройке окружения и зависимостей.

При использовании Docker compose для веб-приложений и баз данных, можно создать файл конфигурации, в котором описать все необходимые сервисы и их зависимости. После этого, с помощью одной команды, можно запустить все необходимые контейнеры и начать работу с приложением.

Такой подход позволяет значительно упростить процесс развертывания и управления приложениями, а также обеспечивает надежность и изоляцию контейнеров. Поэтому использование Docker compose и докеризации в целом является незаменимым инструментом для разработчиков и системных администраторов, которые хотят ускорить и упростить процесс работы с приложениями и базами данных.

## Выводы по первой главе

В данной главе проведен анализ способов разработки и управления информационной системой «Учет компьютерной техники». Также изучены инструменты разработки приложения и базы данных для информационной системы компании АО «Корпорация ВНИИЭМ». Была подтверждена необходимость использования средств автоматизации для обеспечения, поддержки и администрирования баз данных в процессе создания информационной системы «Учет компьютерной техники» для компании АО «Корпорация ВНИИЭМ».

# Аналитическая часть

## Логическое моделирование процесса учета компьютерной техники

### Логическая модель и ее описание

В процессе учета компьютерной техники задействованы три ключевых участника: системный администратор, младший специалист и руководитель отдела. Для обеспечения эффективной работы системы необходимо проводить ее обслуживание, которое включает в себя настройку, мониторинг и резервирование базы данных. Настройка системы предполагает разграничение прав доступа и корректировку структуры базы данных для обеспечения оптимальной работы системы. Эти задачи возложены на системного администратора. Далее, с готовой базой данных необходимо работать, что включает добавление, изменение, удаление и просмотр данных. Эти действия в основном выполняет младший специалист, хотя в некоторых случаях к ним может прибегать и системный администратор.

Главная цель учета компьютерной техники заключается в формировании различных отчетов о техническом состоянии, которые составляются руководителем отдела либо системным администратором в случае, если ему поручено это. На рисунке 2.1 изображена диаграмма прецедентов процесса учета компьютерной техники в АО «Корпорация ВНИИЭМ».

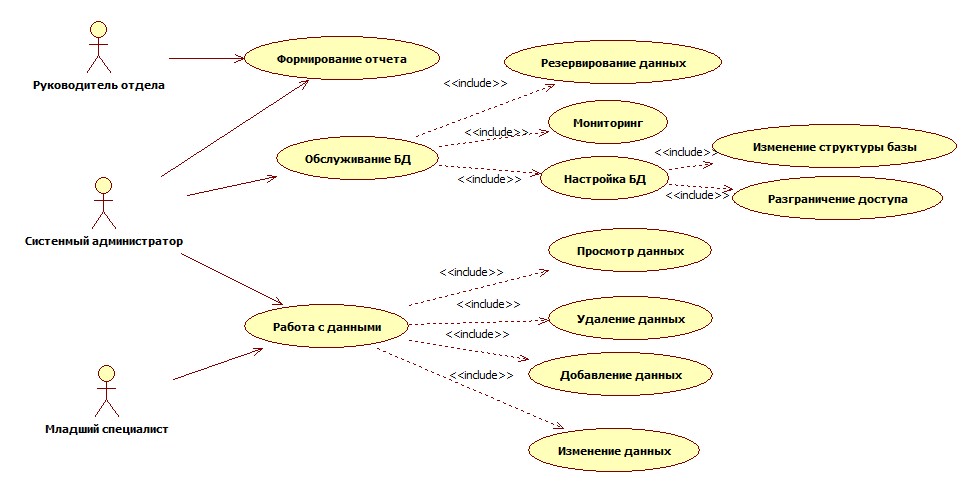


Рисунок 2.1 - Диаграмма прецедентов процесса учета компьютерной техники

В данном случае, для описания структуры базы данных и ее компонентов, а также для представления их взаимодействий, можно использовать диаграмму классов. Это графическое представление позволит наглядно показать классы, их атрибуты и методы, а также взаимосвязи между классами. В целом, диаграмма классов служит инструментом для понимания структуры системы и ее компонентов. На рисунке 2.2 изображена диаграмма классов структуры базы данных.

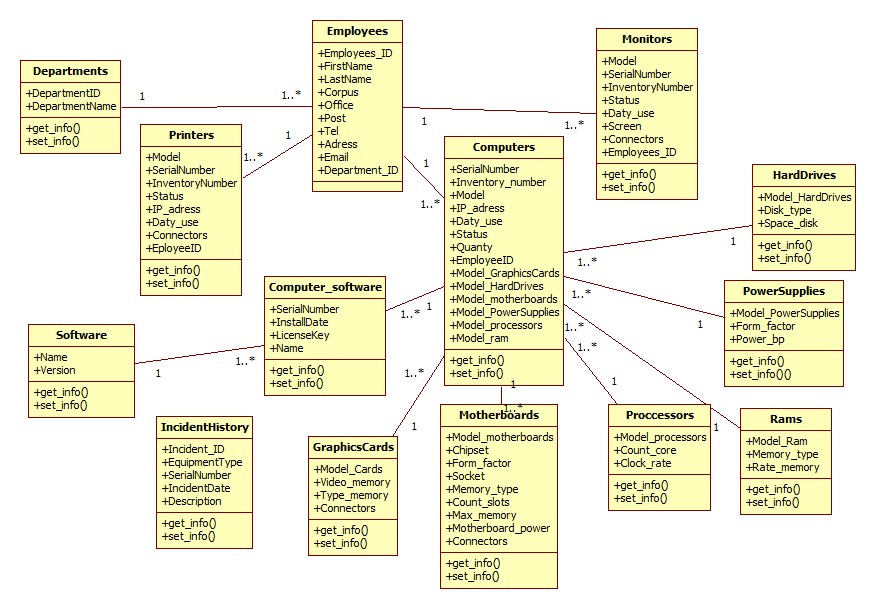


Рисунок 2.2 – Диаграмма классов

После того, как структура базы данных была описана, можно создать логическую схему базы данных на основе описания предметной области. Это позволит наглядно представить структуру данных и их взаимосвязи, а также упростить процесс разработки и сопровождения базы данных. Было выделено 14 сущностей. В таблицах 2.1 – 2.14 представлены сущности предметной области с атрибутами, которые будут использоваться для дальнейшего инфологического моделирования и определения связей между сущностями.

Таблица 2.1 – Сущность «Отдел»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| DepartmentID | INTEGER | PRIMARY KEY, NOT NULL |
| DepartmentName | VARCHAR(100) | NOT NULL |

Таблица 2.2 – Сущность «Сотрудник»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| EmployeeID | INTEGER | PRIMARY KEY,  AUTO\_INCREMENT |
| FirstName | VARCHAR(50) | NOT NUL |
| LastName | VARCHAR(50) | NOT NUL |
| Corpus | INTEGER | NOT NUL |
| Office | INTEGER | NOT NUL |
| Post | VARCHAR(50) | NOT NUL |
| Tel | VARCHAR(18) | NOT NUL |
| Adress | VARCHAR(150) | NOT NUL |
| Email | VARCHAR(40) | NOT NUL |
| DepartmentID | INTEGER | FOREIGN KEY, NOT NUL |

Таблица 2.3 – Сущность «Видеокарта»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| Graphicscards\_id | INTEGER | PRIMARY KEY,  AUTO\_INCREMENT |
| Model\_GraphicsCards | VARCHAR(120) | NOT NUL |
| Video\_memory | INTEGER | NOT NUL |
| Type\_memory | VARCHAR(30) | NOT NUL |
| Connectors | VARCHAR(450) | NOT NUL |

Таблица 2.4 – Сущность «Диск»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| Harddrives\_id | INTEGER | PRIMARY KEY,  AUTO\_INCREMENT |
| Model\_HardDrives | VARCHAR(120) | UNIQUE, NOT NUL |
| Disk\_type | VARCHAR(30) | NOT NUL |
| Space\_disk | INTEGER | NOT NUL |
| Speed\_disk | INTEGER | NOT NUL |

Таблица 2.5 – Сущность «Материнская плата»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| Motherboards\_id | INTEGER | PRIMARY KEY,  AUTO\_INCREMENT |
| Model\_motherboards | VARCHAR(120) | UNIQUE, NOT NUL |
| Chipset | VARCHAR(30) | NOT NUL |
| Form\_factor | VARCHAR(20) | NOT NUL |
| Socket | VARCHAR(20) | NOT NUL |
| Memory\_type | INTEGER | NOT NUL |
| Count\_slots | INTEGER | NOT NUL |
| Max\_memory | INTEGER | NOT NUL |
| Rate\_memory | INTEGER | NOT NUL |
| Motherboard\_power | VARCHAR(20) | NOT NUL |
| Connectors | VARCHAR(450) | NOT NUL |

Таблица 2.6 – Сущность «Блок питания»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| Powersupplies\_id | INTEGER | PRIMARY KEY,  AUTO\_INCREMENT |
| Model\_PowerSupplies | VARCHAR(120) | UNIQUE, NOT NUL |
| Form\_factor | VARCHAR(20) | NOT NUL |
| Power\_bp | INTEGER | NOT NUL |

Таблица 2.7– Сущность «Процессор»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| Processors\_id | INTEGER | PRIMARY KEY,  AUTO\_INCREMENT |
| Model\_processors | VARCHAR(120) | UNIQUE, NOT NUL |
| Count\_core | INTEGER | NOT NUL |
| Clock\_rate | DECIMAL (2,1) | NOT NUL |

Таблица 2.8– Сущность «Программное обеспечение»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| Name | VARCHAR(100) | PRIMARY KEY, NOT NUL |
| Version | VARCHAR(50) | NOT NUL |

Таблица 2.9– Сущность «Оперативная память»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| Ram\_id` | INTEGER | PRIMARY KEY,  AUTO\_INCREMENT |
| Model\_ram | VARCHAR(120) | UNIQUE, NOT NUL |
| Memory\_type | VARCHAR(20) | NOT NUL |
| Rate\_memory | INTEGER | NOT NUL |

Таблица 2.10– Сущность «Компьютер-Программное обеспечение»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| SerialNumber | VARCHAR(13) | PRIMARY KEY, NOT NUL |
| InstallDate | DATE | NOT NUL |
| LicenseKey | VARCHAR(50) | NOT NUL |
| Name | VARCHAR(100) | FOREIGN KEY, NOT NUL |

Таблица 2.11– Сущность «Компьютер»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| SerialNumber | VARCHAR(13) | PRIMARY KEY, NOT NUL |
| Inventory\_number | VARCHAR(10) | NOT NUL |
| Model | VARCHAR(120) | NOT NUL |
| IP\_adress | VARCHAR(19) | DEFAULT NULL |
| Datу\_use | DATE | NOT NUL |
| Status | TINYINT(1) | NOT NUL |
| Quantity | INTEGER | NOT NUL |
| EmployeeID | INTEGER | FOREIGN KEY, NOT NUL |
| Graphicscards\_id | INTEGER | FOREIGN KEY, NOT NUL |
| Harddrives\_id | INTEGER | FOREIGN KEY, NOT NUL |
| Second\_harddrives\_id | INTEGER | FOREIGN KEY |
| Motherboards\_id | INTEGER | FOREIGN KEY, NOT NUL |
| Powersupplies\_id | INTEGER | FOREIGN KEY, NOT NUL |
| Processors\_id | INTEGER | FOREIGN KEY, NOT NUL |
| Ram\_id | INTEGER | FOREIGN KEY, NOT NUL |

Таблица 2.12– Сущность «Монитор»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| Model | VARCHAR(50) | NOT NUL |
| SerialNumber | VARCHAR(13) | PRIMARY KEY, NOT NUL |
| Inventory\_number | VARCHAR(10) | UNIQUE, NOT NUL |
| Status | TINYINT(1) | NOT NUL |
| Datу\_use | DATE | NOT NUL |
| Screen | VARCHAR(200) | NOT NUL |
| Connectors | VARCHAR(450) | NOT NUL |
| EmployeeID | INTEGER | FOREIGN KEY, NOT NUL |

Таблица 2.13– Сущность «Принтер»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| Model | VARCHAR(120) | NOT NUL |
| SerialNumber | VARCHAR(13) | PRIMARY KEY, NOT NUL |
| Inventory\_number | VARCHAR(10) | UNIQUE, NOT NUL |
| Status | TINYINT(1) | NOT NUL |
| IP\_adress | VARCHAR(19) | DEFAULT NULL |
| Datу\_use | DATE | NOT NUL |
| Connectors | VARCHAR(450) | NOT NUL |
| EmployeeID | INTEGER | FOREIGN KEY, NOT NUL |

Таблица 2.14– Сущность «История инцидентов»

|  |  |  |
| --- | --- | --- |
| Имя | Тип данных | Примечание |
| IncidentID | INTEGER | PRIMARY KEY,  AUTO\_INCREMENT |
| EquipmentType | VARCHAR(10) | NOT NUL |
| SerialNumber | VARCHAR(13) | NOT NUL |
| IncidentDate | DATETIME | NOT NUL |
| Description | LONGTEXT | NOT NUL |

Важно описать взаимосвязи между различными сущностями.

Сотрудник связан с компьютером, принтером и монитором через связь "один ко многим", поскольку один сотрудник может использовать несколько устройств каждого типа, в то время как каждое устройство имеет только одного владельца.

Отдел связан с сотрудниками через связь "один ко многим", так как в отделе работает много сотрудников, но каждый сотрудник работает только в одном отделе.

Компьютер связан с процессором, оперативной памятью, жестким диском, материнской платой, блоком питания и видеокартой через связь "один ко многим", поскольку в каждом компьютере установлены эти компоненты.

Компьютер связан с программным обеспечением через связь "многие ко многим", что требует разбиения этой связи путем создания таблицы соединителя.

Путем анализа бизнес-процессов и определения связей между сущностями была создана нотация Баркера, которая позволяет наглядно представить и изучить взаимосвязи между сущностями при описании бизнес-логики системы "Учет компьютерной техники" для компании АО "Корпорация ВНИИЭМ". На рисунке 2.3 изображена логическая схема базы данных в нотации Баркера.

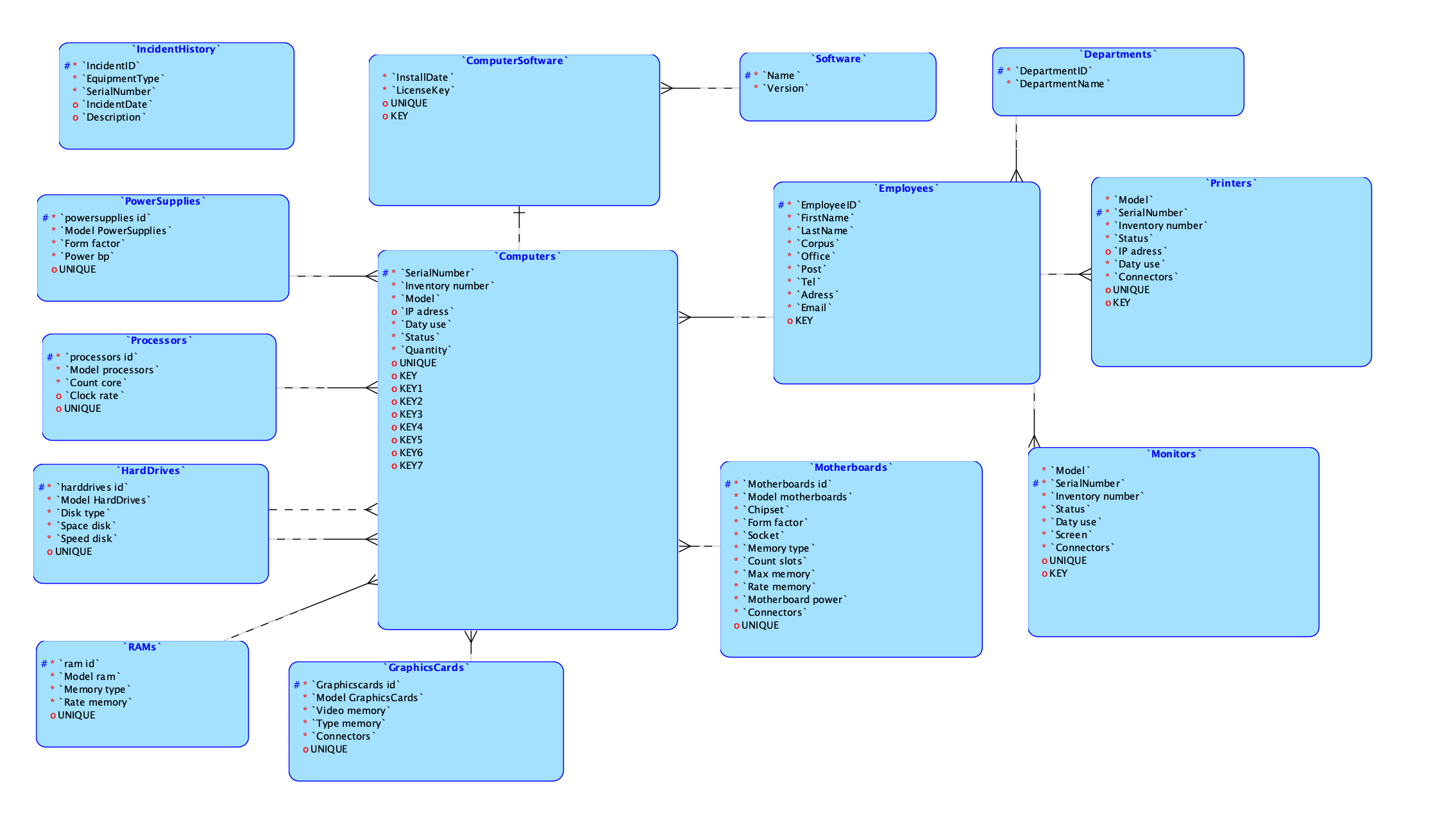


Рисунок 2.3 - Логическая схема базы данных в нотации Баркера

Анализ определения сущностей предметной области и создание модели Баркера приводит к созданию нотации Бахмана. Эта нотация позволяет не только отображать связи между сущностями, но и определять домены и типы данных для атрибутов. С её помощью можно оценить возможные поля как первичные ключи в базе данных. На рисунке 2.4 изображена логическая схема базы данных в нотации Бахмана.

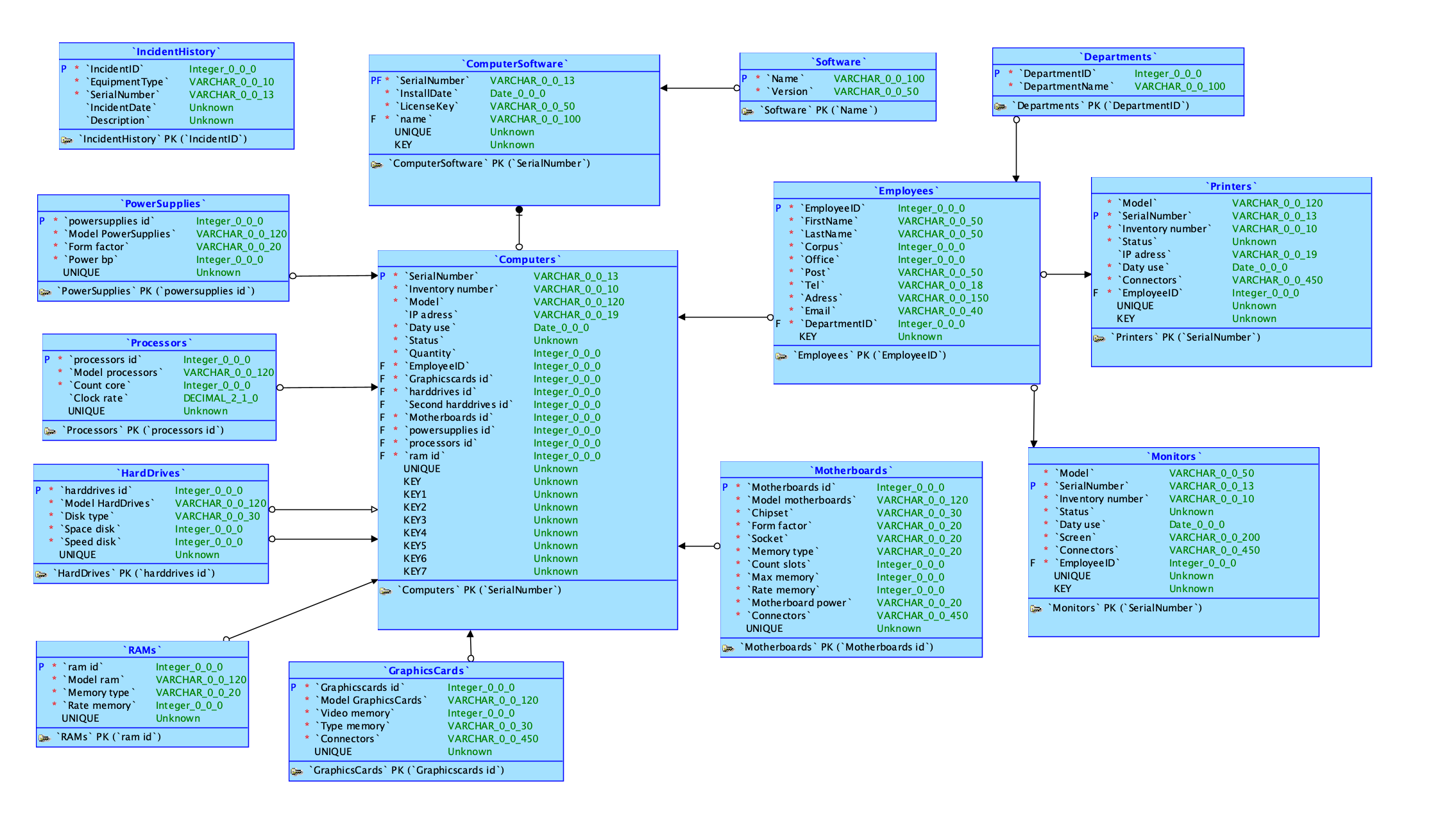


Рисунок 2.4 - Логическая схема базы данных в нотации Бахмана

Технологическая нотация используется для уточнения связей между сущностями в нотации Бахмана. На рисунке 2.5 изображена логическая схема базы данных в технологической нотации.

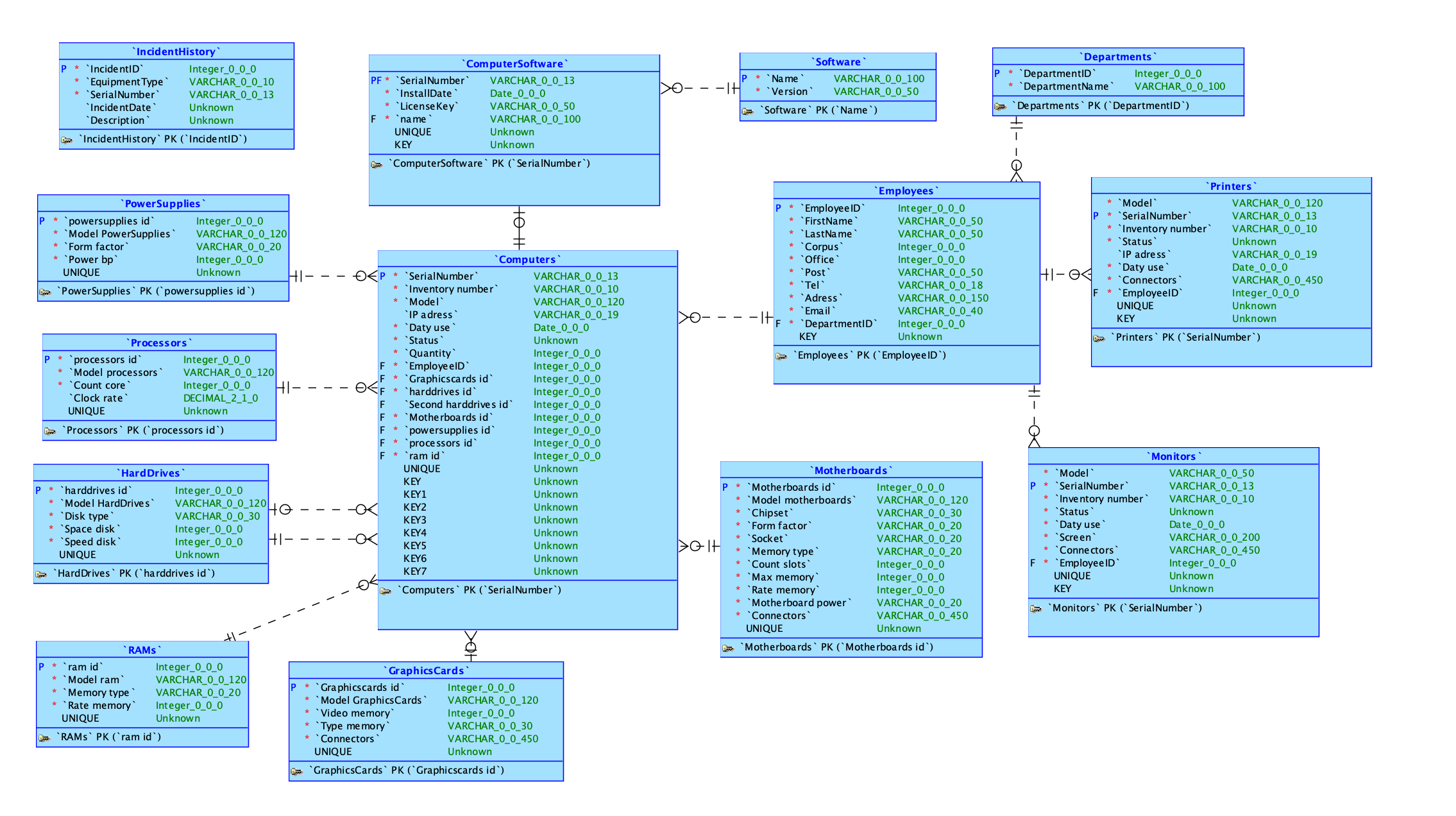


Рисунок 2.5 - Логическая схема базы данных в технологической нотации

На основе анализа процесса учета компьютерной техники и построения логической модели базы данных, можно проводить дальнейшие действия по созданию физической модели.

### Характеристика базы данных

На основании пункта 2.1.1 можно выделить поля и создать словарь данных. В таблице 2.15 продемонстрирован словарь данных разрабатываемой базы данных.

Таблица 2.15 – Словарь данных

|  |  |  |  |
| --- | --- | --- | --- |
| Словарь данных | | | |
| Ключ | Поле | Обязательно | Примечание |
| Отдел | | | |
| Первичный | DepartmentID | Да | Номер отдела |
|  | DepartmentName | Да | Наименование отдела |

Продолжение таблицы 2.15

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ключ | Поле | | | Обязательно | | Примечание |
| Сотрудник | | | | | | |
| Первичный | | EmployeeID | Да | | Номер сотрудника | |
|  | | FirstName | Да | | Имя сотрудника | |
|  | | LastName | Да | | Фамилия сотрудника | |
|  | | Corpus | Да | | Номер корпуса | |
|  | | Office | Да | | Номер кабинета | |
|  | | Post | Да | | Должность | |
|  | | Tel | Да | | Номер телефона | |
|  | | Adress | Да | | Адрес | |
|  | | Email | Да | | Электронная почта | |
| Внешний | | DepartmentID | Да | | Номер отдела | |
| Видеокарта | | | | | | |
| Первичный | | Graphicscards\_id | Да | | Номер карты | |
|  | | Model\_GraphicsCards | Да | | Модель видеокарты | |
|  | | Video\_memory | Да | | Количество памяти | |
|  | | Type\_memory | Да | | Тип памяти | |
|  | | Connectors | Да | | Разъёмы | |
| Диск | | | | | | |
| Первичный | | Harddrives\_id | Да | | Номер диска | |
|  | | Model\_HardDrives | Да | | Модель диска | |
|  | | Disk\_type | Да | | Тип диска | |
|  | | Space\_disk | Да | | Объем диска | |
|  | | Speed\_disk | Да | | Скорость диска | |
| Материнская плата | | | | | | |
| Первичный | | Motherboards\_id | Да | | Номер платы | |
|  | | Model\_motherboards | Да | | Модель карты | |
|  | | Chipset | Да | | Чипсет | |
|  | | Form\_factor | Да | | Форм-фактор | |

Продолжение таблицы 2.15

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | | Поле | Обязательно | Примечание |
|  | | Socket | Да | Сокет |
|  | | Memory\_type | Да | Тип памяти доступный |
|  | | Count\_slots | Да | Количество слотов |
|  | | Max\_memory | Да | Максимальный объем памяти |
|  | | Rate\_memory | Да | Максимальная частота |
|  | | Motherboard\_power | Да | Тип питания |
|  | | Connectors | Да | Разъёмы |
| Блок питания | | | | |
| Первичный | | Powersupplies\_id | Да | Номер блока питания |
|  | | Model\_PowerSupplies | Да | Модель блока питания |
|  | | Form\_factor | Да | Форм фактор |
|  | | Power\_bp | Да | Мощность |
| Процессор | | | | |
| Первичный | | Processors\_\_id | Да | Номер процессора |
|  | | Model\_processors | Да | Модель процессора |
|  | | Count\_core | Да | Количество ядер |
|  | | Clock\_rate | Да | Частота |
| Программное обеспечение | | | | |
| Первичный | Name | | Да | Название ПО |
|  | Version | | Да | Версия ПО |
| Оперативная память | | | | |
| Первичный | Ram\_id | | Да | Номер памяти |
|  | Model\_ram | | Да | Модель памяти |
|  | Memory\_type | | Да | Тип памяти |
|  | Rate\_memory | | Да | Частота памяти |

Продолжение таблицы 2.15

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ключ | | Поле | Обязательно | | Примечание | |
| Компьютер-Программное обеспечение | | | | | | |
| Первичный | SerialNumber | | | Да | | Серийный номер |
|  | InstallDate | | | Да | | Дата установки |
|  | LicenseKey | | | Да | | Лицензионный ключ |
| Внешний | Name | | | Да | | Название ПО |
| Компьютер | | | | | | |
| Первичный | SerialNumber | | | Да | | Серийный номер |
|  | Inventory\_number | | | Да | | Инвентарный номер |
|  | Model | | | Да | | Модель |
|  | IP\_adress | | |  | | IP-адрес |
|  | Datу\_use | | | Да | | Дата использования |
|  | Status | | | Да | | Статус |
|  | Quantity | | | Да | | Количество слотов |
| Внешний | EmployeeID | | | Да | | Номер сотрудника |
| Внешний | Graphicscards\_id | | | Да | | Модель видеокарты |
| Внешний | Harddrives\_id | | | Да | | Модель диска |
| Внешний | Second\_harddrives\_id | | |  | | Модель второго диска |
| Внешний | Motherboards\_id | | | Да | | Модель карты |
| Внешний | Powersupplies\_id | | | Да | | Модель питания |
| Внешний | Processors\_id | | | Да | | Модель процессора |
| Внешний | Ram\_id | | | Да | | Модель памяти |

Продолжение таблицы 2.15

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Поле | Обязательно | Примечание |
| Монитор | | | |
|  | Model | Да | Модель |
| Первичный | SerialNumber | Да | Серийный номер |
|  | Inventory\_number | Да | Инвентарный номер |
|  | Status | Да | Статус |
|  | Datу\_use | Да | Дата использования |
|  | Screen | Да | Экран |
|  | Connectors | Да | Разъёмы |
| Внешний | EmployeeID | Да | Номер сотрудника |
| Принтер | | | |
|  | Model | Да | Модель |
| Первичный | SerialNumber | Да | Серийный номер |
|  | Inventory\_number | Да | Инвентарный номер |
|  | Status | Да | Статус |
|  | IP\_adress |  | IP-адрес |
|  | Datу\_use | Да | Дата использования |
|  | Connectors | Да | Разъёмы |
| Внешний | EmployeeID | Да | Номер сотрудника |
| История инцидентов | | | |
| Первичный | IncidentID | Да | Номер инцидента |
|  | EquipmentType | Да | Тип оборудования |
|  | SerialNumber | Да | Серийный номер |
|  | IncidentDate | Да | Дата инцидента |
|  | Description | Да | Пояснения |

После создания словаря данных необходимо создать реляционную модель данных. Реляционная модель данных представляет собой набор таблиц, которые связаны между собой через ключи. Это позволяет хранить и извлекать данные более эффективно и гибко, чем в других моделях баз данных, таких как сетевая или иерархическая модель. Реляционная модель данных позволяет выполнять различные операции над данными, такие как вставка, извлечение, обновление и удаление данных. Это позволяет легко изменять и обновлять данные в базе данных.

Реляционная модель данных также позволяет выполнять сложные запросы к данным с помощью языка запросов, такого как SQL (Structured Query Language). Это позволяет легко извлекать и анализировать данные в базе данных. На рисунке 2.6 изображена реляционная модель данных.

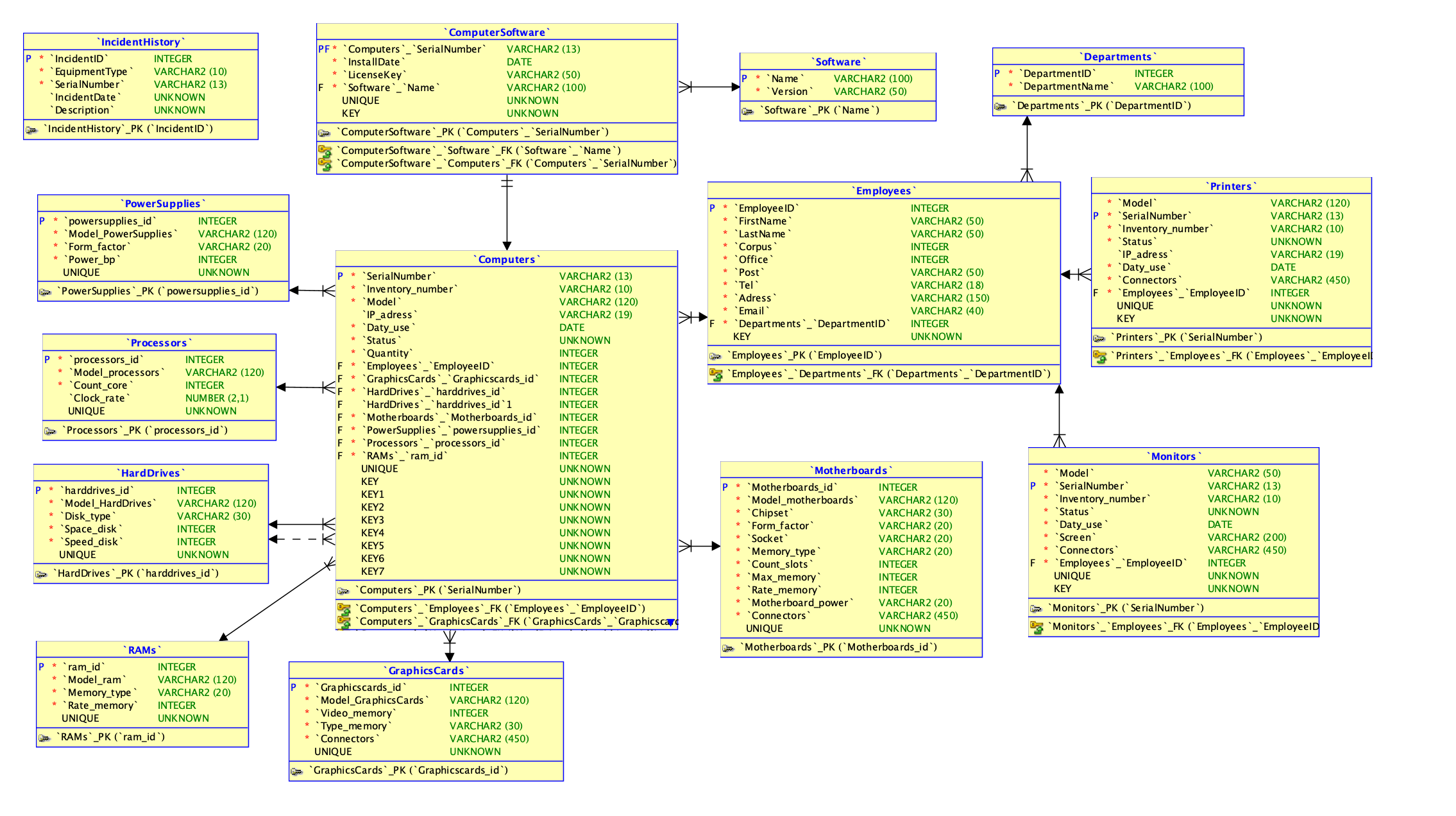


Рисунок 2.6 – Реляционная модель

На основе реляционной модели была создана физическая модель разрабатываемой базы данных. В процессе реализации физической базы данных учитываются особенности реляционной модели, адаптируя ее к требованиям MySQL. Создаются таблицы с нужными атрибутами и типами данных, определяются связи между таблицами с помощью внешних ключей и индексов для ускорения поиска, и сортировки данных. Также приводятся данные в необходимый формат и обеспечиваем правильное хранение и обновление данных в базе. На рисунке 2.7 представлена физическая модель базы данных.

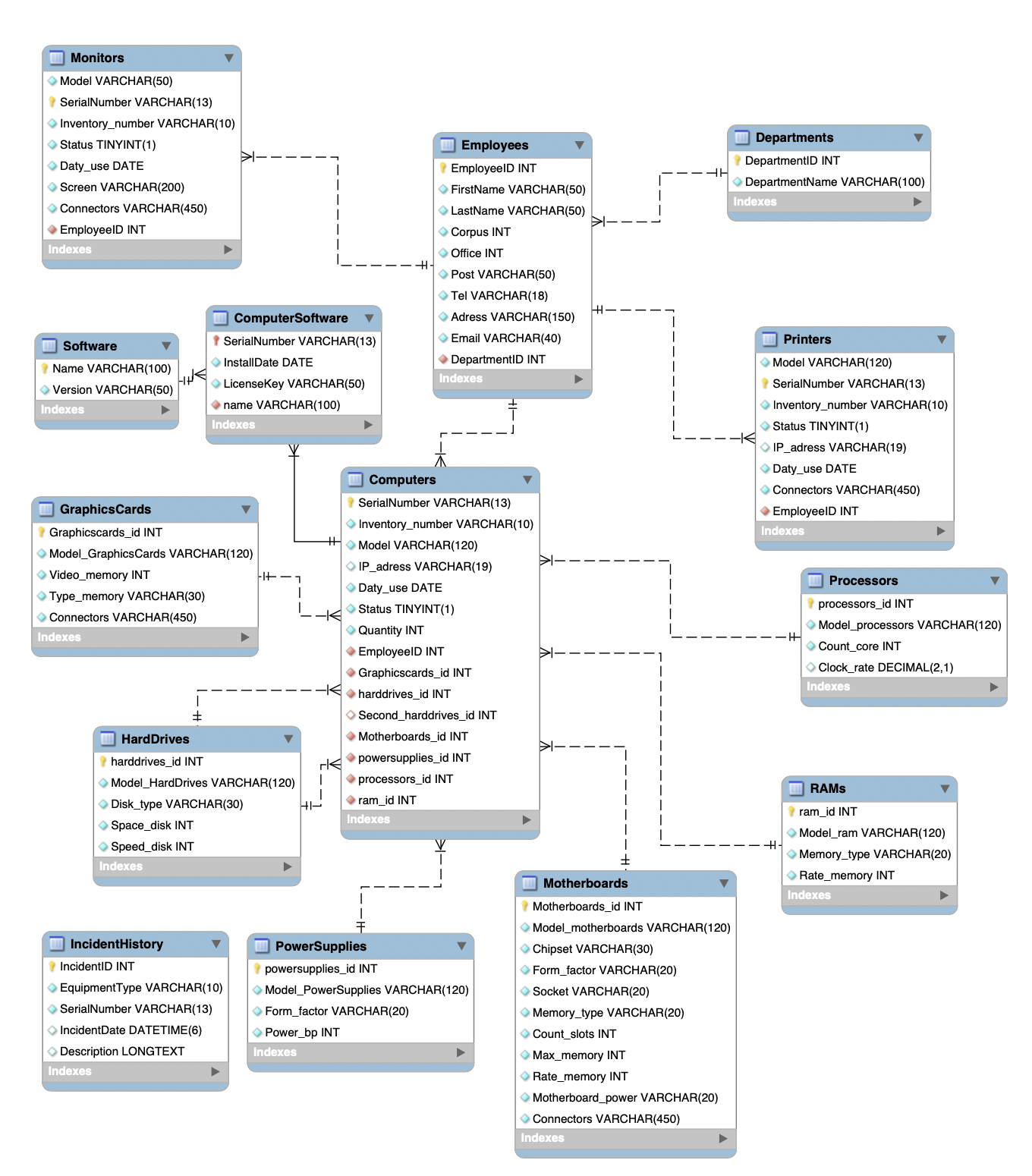


Рисунок 2.7 – Физическая модель базы данных

Для обеспечения безопасности данных и защиты информации в системе, необходимо иметь систему авторизации. Это означает, что каждый пользователь должен иметь уникальный логин и пароль для доступа к системе. Система авторизации позволяет контролировать доступ к конфиденциальным данным и управлять правами пользователей. Только те, у кого есть правильные учетные данные, могут получить доступ к информации и выполнять определенные операции в системе.

Без системы авторизации данные могут быть уязвимыми и подвержены утечкам или несанкционированному доступу. Поэтому важно создать надежную систему авторизации, которая будет гарантировать безопасность данных и защиту от возможных угроз.

Внедрение системы авторизации в функциональность системы способствует повышению ее надежности и безопасности, обеспечивает защиту конфиденциальности данных и предотвращает потенциальные инциденты безопасности. Это делает использование системы более безопасным и эффективным для всех пользователей. В рамках системы существуют два типа разрешений: доступ к панели администратора для управления данными и доступ к основному окну для создания отчетов и просмотра информации о технике. В таблице 2.16 представлена информация о правах доступа.

|  |  |  |
| --- | --- | --- |
| Права  Должность | Доступ в панель администратора | Доступ к основному окну |
| Младший специалист | + | + |
| Системный администратор | + | + |
| Руководитель отдела | - | + |

Для реализации авторизации на веб-ресурсе необходимо создать отдельную базу данных, в которой будут храниться данные о пользователях, их правах и активных сеансах. Это необходимо для обеспечения безопасности и эффективности работы системы.

Отделение базы данных для авторизации от основной базы данных имеет ряд преимуществ. Во-первых, это повышает уровень безопасности, так как данные о пользователях и их доступе не будут храниться вместе с другой информацией, что уменьшает риск несанкционированного доступа. Во-вторых, это обеспечивает более эффективную работу системы, так как отдельная база данных для авторизации позволяет более быстро и эффективно обрабатывать запросы на аутентификацию и авторизацию пользователей.

В целом, отделение базы данных для авторизации от основной базы данных является важным шагом для обеспечения безопасности и эффективности работы системы. Правильное проектирование и реализация базы данных для авторизации поможет предотвратить множество проблем, связанных с безопасностью и контролем доступа пользователей к ресурсам.

На рисунке 2.8 изображена физическая модель базы данных авторизации.

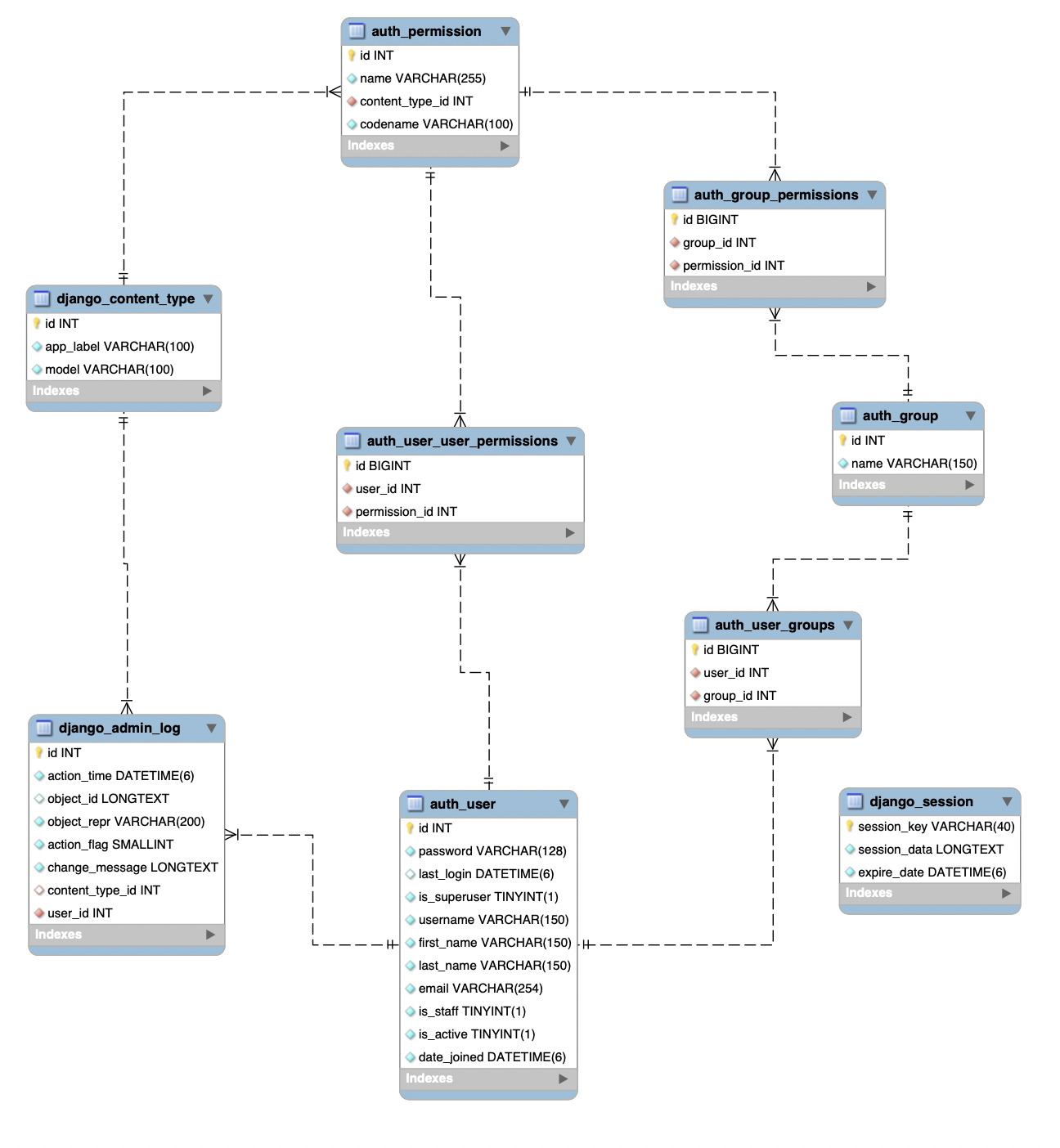


Рисунок 2.8 – Физическая модель базы данных авторизации

Для эффективного учета и контроля за основными сущностями, такими как компьютеры, принтеры и мониторы, необходимо создать дополнительную базу данных для введения журнала действий. Это позволит отслеживать все изменения и операции, происходящие с данными устройствами, а также упростит процесс управления и обслуживания. На рисунке 2.9 изображена физическая модель базы данных истории.

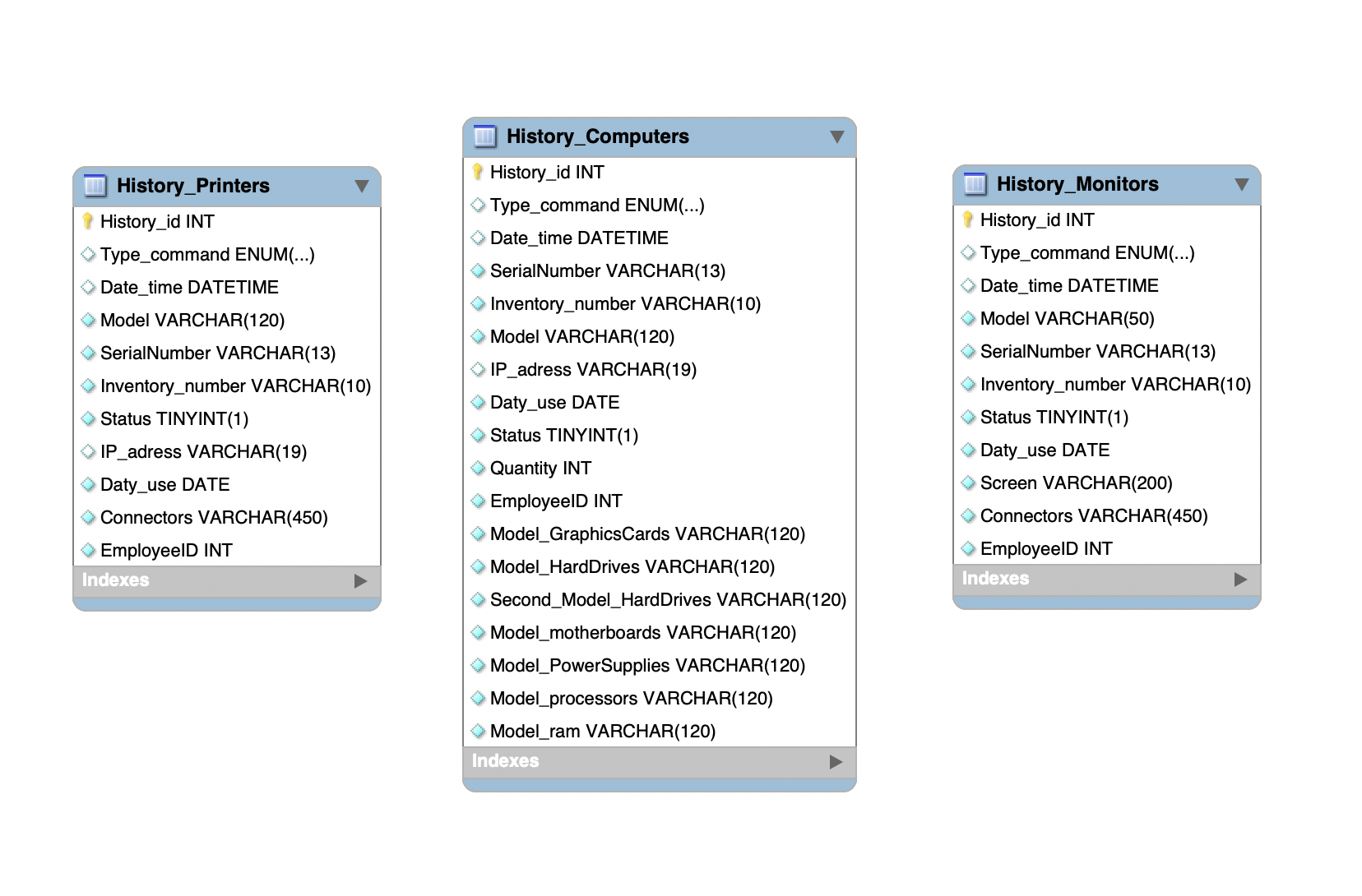


Рисунок 2.9 – Физическая модель базы данных истории

Для автоматического заполнения таблиц необходимо создать триггеры, которые автоматически срабатывают при изменении или удалении данных в таблицах. В них будет фиксироваться тип действия, который был совершен и время выполнения. Скрипт создания триггеров прикреплен в ПРИЛОЖЕНИИ Б.

### Характеристика выходной информации для проектируемой системы

Система «Учета компьютерной техники» в АО «Корпорация ВНИИЭМ» представляет собой веб-программу, разработанную для автоматизации учета компьютерного оборудования в предприятии. Основной целью системы является обеспечение актуальной и достоверной информации о наличии и состоянии компьютерной техники в организации.

Выходная информация, генерируемая системой "учета компьютерной техники", включает в себя следующие основные характеристики:

* + - Список компьютерного оборудования: система предоставляет детальную информацию о каждом компьютере, принтере и мониторе. В перечень включается марка, модель, серийный номер, дата установки, компоненты и другие характеристики;
    - состояние компьютерной техники: система отслеживает текущее состояние каждого устройства, включая информацию о работоспособности и проведенных ремонтах;
    - список программного обеспечения: система учитывает версии, лицензионные ключи и дату установки на каждый компьютер;
    - отчеты и аналитика: система позволяет формировать различные отчеты и аналитические данные о компьютерном оборудовании в предприятии, включая данные о затратах на обслуживание и текущих потребностях организации в обновлении устаревшей техники.

Таким образом, система "учета компьютерной техники" в АО "Корпорация ВНИИЭМ" обеспечивает эффективное управление компьютерным оборудованием и обеспечивает оперативное принятие решений по обновлению и модернизации парка устройств. Важным преимуществом системы является возможность сокращения времени и ресурсов, затрачиваемых на управление техническими аспектами компьютерной инфраструктуры предприятия.

## Выводы по второй главе

Во второй главе работы было проведено логическое моделирование процесса учета компьютерной техники. Была разработана логическая модель, которая включает в себя диаграмму вариантов использования, диаграмму классов и логическую модель данных.

Характеристика базы данных была подробно описана, учитывая основные аспекты структуры и характеристики данных, необходимых для хранения информации о компьютерной технике. Также была представлена характеристика выходной информации для проектируемой системы, которые будут предоставляться пользователю системы.

Исследование логической модели процесса учета компьютерной техники позволило более детально определить требования к базе данных и выходной информации, что способствует более эффективной и точной реализации проектируемой системы «Учета компьютерной техники».

# ЛИТЕРАТУРА

1  Gangadhar Gawande SQL Server Performance Monitoring and Tuning: Become A Smart DBA [Текст] / Gangadhar Gawande — 1-е изд. — : , 2020 — 105 c.

2. Технология разработки баз данных информационных систем. Грачев В.М., Есин В.И., Полухина Н.Г., Гассомахин С.Г. https://cyberleninka.ru/article/n/tehnologiya-razrabotki-baz-dannyh-informatsionnyh-sistem

3. Джордан Краузе Windows Server 2022: комплексное администрирование среды Windows Server [Текст] / Джорданом Краузе — 4-е издание. — : , 2019 — 350 c.

4. Эргашек О.М., Абдукодиров А.А. СРАВНИТЕЛЬНЫЙ АНАЛИЗ СРЕД СОЗДАНИЯ WEB-ПРИЛОЖЕНИЙ // Universum: технические науки : электрон. научн. журн. 2023. 12(117). URL: https://7universum.com/ru/tech/archive/item/16582 (дата обращения: 12.03.2024).

5. Лукин И.К. СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ ДЛЯ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ // Теория и практика современной науки. 2017. №2 (20). URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-yazykov-programmirovaniya-dlya-razrabotki-veb-prilozheniy (дата обращения: 12.03.2024).

6. https://docs.djangoproject.com/en/5.0/

7. <https://nodejs.org/api/all.html>

8 [PHP: Documentation](https://www.php.net/docs.php)

9 https://docs.docker.com/

10 https://docs.docker.com/compose/