# 4. Выбор концепции построения и описание архитектуры учебной системы расчета структурной надежности

Перед нами стоит задача создать такую систему, благодаря которой можно будет существенно упростить процесс обучения расчету надежности. Одной из особенностей этой системы является ее направленность на дистанционное обучение.

Вследствие вышеперечисленного, работа с системой будет проходить в браузере, где необходимая информация будет отображаться на web-страницах. Данные для расчета будут поступать в систему из базы данных, также в системе организован ввод данных пользователем.

Основные идеи современной информационной технологии базируются на концепциибаз данных (БД). База данных (БД) — структурированный организованный набор данных, описывающих характеристики каких-либо физических или виртуальных систем. Согласно данной концепции основой информационной технологии являютсяданные,организованные в БД, адекватно отражающие реалии действительности в той или иной предметной области и обеспечивающие пользователя актуальной информацией в соответствующей предметной области. Для создания и управления информационной системой необходима система управления базами данных (СУБД). СУБД - специализированная программа (чаще комплекс программ), предназначенная для организации и ведения базы данных.

Обычно современная СУБД содержит следующие компоненты:

* ядро, которое отвечает за управление данными во внешней и оперативной памяти и [журнализацию](http://ru.wikipedia.org/wiki/%D0%96%D1%83%D1%80%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9);
* процессор языка базы данных, обеспечивающий [оптимизацию запросов](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%BE%D0%B2_%D0%A1%D0%A3%D0%91%D0%94) на извлечение и изменение данных, и создание, как правило, машинно-независимого исполняемого внутреннего кода,
* подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД
* сервисные программы, обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

Хорошая система управления базами данных относительно справляется с ролью эффективного посредника между пользователем и данными, организовывает, каталогизирует, резервирует и выполняет целый ряд других операций, которые направлены на то, чтобы ускорить и облегчить работу с данными.

**Классификация СУБД**

Современные СУБД можно классифицировать следующим образом:

По типу управляемой базы данных СУБД

* [Иерархические](http://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94)
* [Сетевые](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94)
* [Реляционные](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94)
* [Объектно-реляционные](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94)
* [Объектно-ориентированные](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94)

По архитектуре организации хранения данных

* локальные СУБД (все части локальной СУБД размещаются на одном компьютере)
* распределенные СУБД (части СУБД могут размещаться на двух и более компьютерах)

По способу доступа к БД

* Файл-серверные

В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере. Ядро СУБД располагается на каждом клиентском компьютере. Доступ к данным осуществляется через локальную сеть. Синхронизация чтений и обновлений осуществляется посредством файловых блокировок. Преимуществом этой архитектуры является низкая нагрузка на ЦП сервера, а недостатком — высокая загрузка локальной сети.

На данный момент файл-серверные СУБД считаются устаревшими. Примеры: Microsoft Access, Borland Paradox.

* Клиент-серверные

Такие СУБД состоят из клиентской части (которая входит в состав прикладной программы) и сервера. Клиент-серверные СУБД, в отличие от файл-серверных, обеспечивают разграничение доступа между пользователями и мало загружают сеть и клиентские машины. Сервер является внешней по отношению к клиенту программой, и по надобности его можно заменить другим. Недостаток клиент-серверных СУБД в самом факте существования сервера (что плохо для локальных программ — в них удобнее встраиваемые СУБД) и больших вычислительных ресурсах, потребляемых сервером.

Примеры: Firebird, Interbase, IBM DB2, MS SQL Server, Sybase, Oracle, PostgreSQL, MySQL, ЛИНТЕР.

* Встраиваемые

Встраиваемая СУБД  - библиотека, которая позволяет унифицированным образом хранить большие объёмы данных на локальной машине. Доступ к данным может происходить через SQL либо через особые функции СУБД. Встраиваемые СУБД быстрее обычных клиент-серверных и не требуют установки сервера, поэтому востребованы в локальном ПО, которое имеет дело с большими объёмами данных.

Примеры: OpenEdge, SQLite, BerkeleyDB, один из вариантов Firebird, один из вариантов MySQL,  Sav Zigzag,  Microsoft SQL Server Compact, ЛИНТЕР.

Установка и администрирование некоторых СУРБД (систем управления реляционными базами данных) часто оказывается очень дорогим и сложным мероприятием. Общепризнанными представителями этого типа баз данных являются Огас1е, DB/2 (производства IBM) и SQL Server (Microsoft) — массивные, многофункциональные системы, способные хранить и обрабатывать практически любые данные, которые могут понадобиться на современном предприятии. Оборотная сторона этой медали тяжеловесность и дороговизна таких систем, к тому же они могут содержать больше функций, чем от них требуется.

Cуществуют альтернативы, такие как системы PostgreSQL и MySQL, каждая из которых представляет собой клиент/серверную систему баз данных с открытым исходным кодом и вот уже много лет является весьма популярной среди РНР-разработчиков. Эти системы отличаются быстродействием, стабильны и легко удовлетворяют потребности большинства мелких и средних проектов. Не менее важно то, что они бесплатны.

Система должна взаимодействовать с пользователем. Это достигается использованием web-интерфейса.

Языки веб-программирования — это языки, которые в основном предназначены для работы с интернет-технологиями. Языки веб-программирования делятся на две группы: клиентские и серверные.

**Клиентские языки**

Как следует из названия, программы на клиентских языках обрабатываются на стороне пользователя. Как правило, их выполняет браузер. Самыми распространенными клиентскими языками программирования являются: JavaScript, VBScript, ActionScript.

**Серверные языки**

Когда пользователь дает запрос на какую-либо страницу (переходит на неё по ссылке или вводит адрес в адресной строке своего браузера), то вызванная страница сначала обрабатывается на сервере, то есть выполняются все программы, связанные со страницей, и только потом возвращается к посетителю по сети в виде файла. Список серверных языков программирования: PHP, Perl, Python, Ruby, Любой .NET язык программирования (технология ASP.NET), Java, Groovy.

## 4.1. Программные средства для решения поставленной задачи

### 4.4.1. БД MySQL 5

Решающим критерием является выбор СУБД. Среди некоммерческих вариантов наиболее подходящим является MySQL, так как она считается самой совершенной СУБД, распространяемой на условиях открытых исходных текстов. В MySQL реализованы многие возможности, традиционно встречавшиеся только в масштабных коммерческих продуктах. По сравнению с другими MySQL предоставляет ряд преимуществ:

* это одна из самых популярных систем управления базами данных, используемых в настоящее время в web-среде;
* она свободно доступна для загрузки из Internet и установки на практически любой машине;
* ее легко установить на многих операционных системах (включая Windows и Unix);
* она проста в использовании и включает в себя несколько удобных инструментов администрирования;
* это быстрая, мощная клиент/серверная система, которая справляется с очень крупными, сложными базами данных и хорошо проявляет себя, когда дело касается крупных проектов.

MySQL  — [свободная](http://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E) [система управления базами данных](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). MySQL является собственностью компании  [Sun Microsystems](http://ru.wikipedia.org/wiki/Sun_Microsystems), осуществляющей разработку и поддержку приложения. MySQL является решением для малых и средних приложений. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа [MyISAM](http://ru.wikipedia.org/wiki/MyISAM" \o "MyISAM), поддерживающие полнотекстовый поиск, так и таблицы [InnoDB](http://ru.wikipedia.org/wiki/InnoDB" \o "InnoDB), поддерживающие транзакции на уровне отдельных записей.

InnoDB — одна из выбираемых подсистем низкого уровня в [СУБД](http://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94) [MySQL](http://ru.wikipedia.org/wiki/MySQL" \o "MySQL), входит во все стандартные сборки для различных операционных систем. Основным отличием InnoDB от других подсистем низкого уровня MySQL является наличие механизма [транзакций](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F). В отличие от таблиц [MyISAM](http://ru.wikipedia.org/wiki/MyISAM" \o "MyISAM), где для каждой таблицы создается один файл данных, данные InnoDB в настройках по умолчанию хранятся в больших совместно используемых файлах (изменить это можно с помощью настроек опции innodb\_file\_per\_table), что позволяет использовать постраничный кэш страниц базы данных. Формат данных InnoDB обеспечивает надежное хранение данных за счет транзакционности.

### 4.1.2. Язык гипертекстовой разметки HTML

Язык разметки гипертекстовых страниц (HTML – Hypertext Markup Language) представляет собой язык, разработанный специально для создания Web-документов. Он определяет синтаксис и размещение специальных инструкций (тегов), которые не выводятся на экран, но указывают браузеру, как отображать содержимое документа. Он предназначен для создания форматированного текста, насыщенного изображениями, звуком, анимацией, видеоклипами и гипертекстовыми ссылками на другие документы, разбросанные как по всему Web-пространству, так и находящиеся на этом же сервере или являющиеся составной частью этого же Web-проекта.

На практике на стандарт HTML большое влияние оказывает наличие тегов, предложенных и поддерживаемых наиболее известными браузерами, такими как Microsoft Internet Explorer и Opera. Эти теги в данный момент могут как входить, так и не входить в состав действующей спецификации HTML.

**Создание документов в формате HTML**

Принципы гипертекстовой разметки. Структура документов

За основу модели разметки документов в HTML принята тэговая модель. Тэговая модель описывает документ как совокупность контейнеров, каждый из которых начинается и заканчивается тэгами. Т.е. документ НТМL представляет собой не что иное, как обычный АSСII-файл, с добавленными в него управляющими НТМL-кодами (тэгами).

Тэги НТМL-документов в большинстве своем просты для понимания и использования, ибо они образованы с помощью общеупотребительных слов английского языка, понятных сокращений и обозначений. НТМL-тэг состоит из имени, за которым может следовать необязательный список атрибутов тэга. Текст тэга заключается в угловые скобки (< и >). Простейший вариант тэга - имя, заключенное в угловые скобки, например <HEAD> или <i>. Для более сложных тэгов характерно различие атрибутов, которые могут иметь конкретные значения, определенные автором для видоизменения функции тэга.

В некоторых случаях конечные тэги в документе можно опускать. Большинство браузеров реализованы так, что при обработке текста документа начальный тэг воспринимается как конечный тэг предыдущего. Самый распространенный тэг такого типа - тэг абзаца <Р>. Поскольку он используется в документе очень часто, то его обычно ставят только в начале каждого абзаца. Когда один абзац заканчивается, следующий тэг <Р> сигнализирует браузеру о том, что нужно завершить данный абзац и начать следующий.

Атрибуты тэга следуют за именем и отделяются друг от друга одним или несколькими знаками табуляции, пробелами или символами возврата к началу строки. Порядок записи атрибутов в тэге значения не имеет.

**Группы тэгов НТМL**

Все тэги НТМL по их назначению и области действия можно разделить на следующие основные группы

* определяющие структуру документа;
* оформление блоков гипертекста (параграфы, списки, таблицы, картинки);
* гипертекстовые ссылки и закладки;
* формы для организации диалога;
* вызов программ.

Структура гипертекстовой сети задается гипертекстовыми ссылками. Гипертекстовая ссылка - это адрес другого HTML документа или информационного ресурса Internet, который тематически, логически или каким-либо другим способом связан с документом, в котором ссылка определена.

Этот текст содержит гипертекстовую ссылку</A>

<A HREF="http://polyn.net.kiae.su/altai/index.html">

В приведенном выше примере тэг "A", который в HTML называют якорем (anchor), использует атрибут "HREF", который обозначает гипертекстовую ссылку (Hypertext Reference), для записи этой ссылки в форме URL. Данная ссылка указывает на документ с именем "index.html" в директории "altai" на сервере "polyn.net.kiae.su", доступ к которому осуществляется по протоколу "http".

**Контейнеры HTML-документа**

Каждая из составных частей документа имеет свой набор контейнеров, которые можно внутри нее использовать. Контейнеры тела документа не используются в заголовке или в контейнере FRAMSET.

Заголовок документа не имеет атрибутов. Основное назначение тэгов заголовка - это описание общих для всего документа параметров отображения.

Тэг МЕТА предназначен для определения в заголовке документа конструкций, отсутствующих в спецификации НТМL.

META HTTP-EQUIV="Keywords"

CONTENT="Plasma, Nuclear Physics">

При таком использовании в заголовок НТТР-пакета будет включена строка: Keywords: Plasma, Nuclear Physics, что удобно при отправке почты, например.

**Средства описания таблиц в HTML**

Для описания таблиц служит тэг <ТАВLЕ>. Тэг <ТАВLЕ>, как и многие другие, автоматически переводит строку до и после таблицы.

Тэг <ТR> (сокращение от Таble Row - строка таблицы) создает строку таблицы.

Внутри строки таблицы обычно размещаются ячейки с данными. Каждая ячейка, содержащая текст или изображение, должна быть окружена тэгами <ТD></ТD>. Число тэгов <ТD></ТD> в строке определяет число ячеек.

При задании заголовков для столбцов и строк таблицы используются тэг заголовка <ТН></ТН> (Таblе Неаder, заголовок таблицы).

Тэг <CAPTION> позволяет создавать заголовки таблицы. По умолчанию заголовки центрируются и размещаются либо над (<САРТION АLIGN=ТОР>), либо под таблицей (<САРТION ALIGN=ВОТТОМ>). Заголовок может состоять из любого текста и изображений.

Обычно любой текст в таблице, не помещающийся в одну строку ячейки, переходит на следующую строку. При использовании атрибута NOWARP с тэгами <ТН> или <ТD> длина ячейки расширяется на столько, чтобы, заключенный в ней текст поместился в одну строку.

Тэги <ТD> и <ТН> модифицируются с помощью атрибута СОLSPAN= (Column Span, соединение столбцов).

Атрибут ROWSPAN=, используемый в тэгах <ТD> и <ТН>, аналогичен атрибуту СОLSPAN=, только он задает число строк, на которые растягивается ячейка.

Атрибут WIDТН= применяется в двух случаях. Можно поместить его в тэг <ТАВLЕ> для задания ширины всей таблицы, а можно использовать в тэгах <ТR> или <ТН> для задания ширины ячейки или группы ячеек. Ширину можно указывать в пикселях или в процентах.

Атрибут UNIT= тэга <ТАВLЕ> определяет единицы измерения, используемые при указании размеров, как всей таблицы, так и ее отдельных столбцов.

Атрибут СОLSРЕС=, используемый с атрибутом UNIТ=, определяет, сколько места занимает каждый столбец таблицы, и как в нем выравниваются данные. Применяется только в тэге<ТАВLЕ>.

Атрибут DР= (Decimal Point, десятичный знак) определяет символ, используемый для отделения целой части десятичной дроби.

Атрибут СЕLLPADDING= определяет ширину пустого пространства между содержимым ячейки и ее границами, то есть задает поля внутри ячейки.

Тэги <ТR>, <ТD> и <ТН> можно модифицировать с помощью атрибутов ALIGN= и VALIGN=. Атрибут АLIGN определяет выравнивание текста и графики по горизонтали, то есть по левому или правому краю, либо по центру,

Атрибут VALIGN= осуществляет выравнивание текста и графики внутри ячейки по вертикали. Вертикальное выравнивание может быть задано несколькими способами: VALIGN=ТОР. Выравнивает содержимое ячейки по ее верхней границе.

VALIGN=МIDDLE центрирует содержимое ячейки по вертикали.   
VALIGN=ВОТТОМ выравнивает содержимое ячейки по ее нижней границе.

Атрибут СЕLLSPACING= определяет в пикселях ширину промежутков между ячейками.

### 4.1.3. Язык описания сценариев PHP

**Общие сведения о PHP**

PHP (Personal Home Page Tools) - это язык описания сценариев общего назначения, который создан специально для Web и который можно внедрять в HTML. Важным отличием от других языков, используемых для Web-программирования, является то, что вместо написания программы с большим количеством команд для вывода HTML, пишется HTML-скрипт с некоторым количеством встроенного кода для выполнения каких-либо действий. Код PHP заключён в специальные [начальный и конечный тэги](mk:@MSITStore:F:\Documents%20and%20Settings\all\Рабочий%20стол\ДИПЛОМ!\PHP\PHP42\php42.chm::/basic-syntax.html#language.basic-syntax.phpmode), что позволяет входить и выходить из "режима PHP".

PHP отличается от других подобных языков, типа клиентского JavaScript, тем, что код выполняется на сервере. То есть сам скрипт находится на сервере, а клиент получает только результат работы этого скрипта, не имея возможности определить, каков был исходный код. Наилучшим качеством PHP является то, что он довольно прост, но предлагает много разнообразных возможностей для программиста.

PHP в основном сориентирован на серверный скриптинг, поэтому может делать всё то, что делают CGI-программы: сбор данных форм, динамическую генерацию содержимого страницы или приём и отправку cookies. Но PHP может намного больше.

PHP может использоваться на всех крупных операционных системах (ОС), включая Linux, многие варианты Unix (HP-UX, Solaris и OpenBSD), Microsoft Windows, Mac OS X, RISC OS и, возможно, другие. PHP имеет поддержку для большинства существующих web-серверов. Это Apache, Microsoft Internet Information Server, Personal Web Server, Netscape и iPlanet-серверы, Oreilly Website Pro, Caudium, Xitami, OmniHTTPd и многие другие. Для большинства этих серверов PHP имеет модули. В других, поддерживающих стандарт CGI, PHP может работать как CGI-процессор.

Итак, PHP дает свободу выбора ОС и web-сервера. В PHP нет ограничений в выводе HTML. PHP может выводить изображения, PDF-файлы и даже клипы Flash, генерируемые на лету. PHP имеет предельно удобные возможности для работы с текстом, он может выводить любой текст, включая XHTML, и любой другой XML-файл. PHP автоматически генерирует эти файлы и сохраняет их в файловой системе, вместо их распечатки, формируя серверный кэш для динамического содержимого.

Одна из наиболее сильных и привлекательных черт PHP - поддержка им большого количества баз данных (БД). Имеется также абстрактное расширение DBX, позволяющее прозрачно использовать любую БД, поддерживаемую этим расширением. Также PHP поддерживает ODBC, стандарт Open Database Connection, и дает возможность подключаться к любой БД, поддерживающей этот всемирный стандарт.

**Принцип работы PHP**

Когда PHP разбирает файл, он просто передаёт текст файла, пока не обнаружит один из специальных тэгов, который говорит о необходимости начать интерпретацию текста как кода PHP. Разборщик выполняет весь найденный код до закрывающего тэга PHP, который говорит разборщику, что нужно снова начать просто передавать текст. Этот механизм позволяет внедрять PHP-код в HTML: всё за пределами тэгов PHP остаётся без изменений, а внутри тэгов - разбирается как код.

Имеются четыре набора тэгов, которые используются для обозначения блоков кода PHP. Только два из них (<?php. . .?> и <script language="php">. . .</script>) всегда доступны; другие можно включать и отключать из файла конфигурации php.ini. Хотя сокращённые тэги и тэги в стиле ASP могут быть удобны, они не так переносимы, как их длинные версии.

**Синтаксис языка PHP**

PHP является настоящим языком программирования со всеми присущими ему атрибутами: переменными, операторами контроля процесса, командами ввода/вывода и др.

Для написания кода программ на PHP используется стандартный набор латинских символов. В строковых выражениях могут использоваться символы национальных алфавитов. В качестве комментария используется последовательность символов //.

Переменные – это контейнеры, в которых содержатся данные. Для создания переменной нужно дать ей имя и значение.Чтобы PHP мог отличать переменные от обычного текста или команды, их имя должно начинаться со знака «$». Имя переменной чувствительно к регистру символов. Также существуют определенные ограничения на имя переменной: оно не должно содержать пробелов, не должно содержать знаков апострофов и некоторых других символов. Как и везде, чтобы создать переменную, нужно написать ее имя, а затем, через знак присваивания значение:

$a = 125;

Таким образом, один раз объявив переменную в дальнейшем на нее можно просто ссылаться.

PHP поддерживает 8 примитивных типов.

4 скалярных типа:

* [boolean](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/types.boolean.html)
* [integer](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/types.integer.html)
* [число с плавающей точкой (float)](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/types.float.html)
* [string](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/types.string.html)

Два составных типа:

* [array](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/types.array.html)
* [object](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/types.object.html)

И, наконец, два специальных типа:

* [resource](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/types.resource.html)
* [NULL](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/types.null.html)

Тип переменной обычно программистом не устанавливается; напротив, он определяется РНР на этапе прогона, в зависимости от контекста, в котором эта переменная используется. В случаях, когда необходимо проверить тип и значение конкретного [выражения](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/expressions.html), используется функция [*var\_dump()*](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/f/var-dump.html). Если нужно получить читабельное представление типа для отладки, используется [*gettype()*](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/f/gettype.html). Но следует заметить, что для проверки конкретного типа функция [*gettype()*](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/f/gettype.html) не используется, а используется функция *is\_type.*

Для того чтобы форсированно конвертировать переменную в определённый тип, можно либо [привести](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/types.type-juggling.html#language.types.typecasting) переменную, либо использовать с ней функцию [*settype*](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/f/settype.html)*.* Приведение типов в PHP работает так же, как и в C: имя нужного типа записывается в скобках перед переменной, которая к этому новому типу приводится.

Допускаются следующие приведения типов:

* (int), (integer) - приведение к integer
* (bool), (boolean) - приведение к boolean
* (float), (double), (real) - приведение к float
* (string) - приведение к string
* (array) - приведение к array
* (object) - приведение к object

Изменение значений переменных происходит при помощи использования оператора присваивания "=". Кроме оператора присваивания используются следующие основные логические операции:

«= =» - знак равенства;

«>», «<», «>=», «<=» - больше, меньше, больше или равно, меньше или равно;

«!=» или «<>» - знак неравенства;

а и b – условия;

а && b – действие будет выполнено, если выполняется И условие а, И условие b

а II b - действие будет выполнено, если выполняется ИЛИ условие а, ИЛИ условие b, ИЛИ оба условия одновременно.

Никакой язык программирования не может обойтись без математических операций. В PHP также встроены операторы, которые позволяют производить математические вычисления. Основные математические операции:

a + b – сложение;

a – b – вычитание;

a \* b – умножение;

a / b – деление;

a % b – остаток от деления а на b;

$переменная++; - запись эквивалентна следующей: $переменная =

$переменная+1; - значение переменной увеличивается на 1;

$переменная--; - значение переменной уменьшается на 1.

Массивы – это набор данных, объединенных под одним именем. Храниться он может в переменной. Одними из самых легких способов создания массива являются следующие:

$имя массива [номер элемента] = значение

$имя массива = array (значение);

Индексы и значения в массиве разделяются оператором =>. Пары index=>value разделяются запятыми, они определяют индекс и значение. Индекс может быть как числовым, так и строковым. В ассоциированных массивах индекс всегда ведет себя как строковой. В случае, если индекс не указан, будет подставляться автоинкремент (на 1 больше), начиная с 0. Если при создании массива были указаны два элемента с одинаковыми индексами, то последний элемент заменяет первый. Для просмотра элементов массива существуют две функции: var\_dump() и print\_r().

**Управление потоком вычислений в PHP**

PHP имеет следующие операторы, реализующие основные алгоритмы управления потоком вычислений (flow control): оператор ветвления, оператор цикла с конечным числом повторений, оператор цикла while.

Рассмотрим их синтаксис.

Оператор ветвления реализует выбор той или иной последовательности действий в зависимости от условия (условный оператор):

If (условие) {…команды, которые должны выполняться, если условие верно…;} еlse {… команды, которые должны выполняться, если условие неверно…;}

Выражение "условие" должно быть булевского типа. Если "условие" вычисляется в TRUE, PHP выполнит команды, а если вычисляется в FALSE - оператор игнорируется.

Также в PHP в операторе ветвления используется комбинация if и else - elseif. Подобно else, она расширяет оператор if для выполнения других операторов в том случае, если оригинальное выражение if вычисляется в FALSE. Однако, в отличие от else, elseif будет выполнять альтернативное выражение только тогда, когда условное выражение в elseif будет вычислено в TRUE.

Может быть несколько elseif внутри одного оператора if. Первое выражение elseif (если имеется), которое вычисляется в TRUE, будет выполняться. В PHP вы можете также записать 'else if' (двумя словами), и поведение будет идентично 'elseif' (в одно слово). Оператор elseif выполняется только в том случае, если предшествующее выражение if и любое предшествующее выражение elseif вычислены в FALSE, а текущее выражение elseif вычислено в TRUE.

PHP предлагает альтернативный синтаксис для некоторых структур управления: if, while, for, foreach и switch. В каждом случае базовая форма синтаксиса изменяется - открывающая фигурная скобка заменяется на двоеточие, а закрывающая - на endif;, endwhile;, endfor;, endforeach; или endswitch;, соответственно.

Оператор цикла с конечным числом повторений повторяет определенную последовательность действий заданное число раз.

Синтаксис цикла for таков:

For (выражение 1, выражение 2, выражение 3) {…команды…;}, где:

выражение 1 = начальное условие счетчика;

выражение 2 = условие продолжения цикла;

выражение 3 = изменение счетчика на каждом цикле

Первое выражение вычисляется (выполняется) один раз и безусловно в начале выполнения цикла.

При начале каждой итерации вычисляется выражение 2. Если оно вычисляется в TRUE, цикл продолжается и выполняется вложенный оператор. Если оно вычисляется в FALSE, выполнение цикла прекращается.

В конце каждой итерации вычисляется (выполняется) выражение 3.

Каждое из выражений может быть пустым. Пустое выражение 2 означает, что цикл должен выполняться бесконечно (PHP неявно предполагает, что это условие TRUE, как в C). Это может быть не так бессмысленно, как может показаться, поскольку часто бывает необходимо закончить цикл оператором [break](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/break.html) вместо использования проверки выражения-условия цикла for. Оператор break заканчивает выполнение текущей структуры (цикла) for, foreach, while, do..while или switch. Он принимает необязательный числовой аргумент, указывающий, сколько содержащих вкладывающих структур прерывают выполнение.

Циклы while это простейшие циклы PHP. Они ведут себя, как их аналоги из C. Вот базовая форма оператора while:

while (условие) {…команды…;}

Значение оператора while говорит PHP, что нужно неоднократно выполнять вложенный оператор, пока «условие» вычисляется в TRUE. Значение выражения проверяется каждый раз в начале цикла, поэтому, если это значение изменилось при выполнении вложенного оператора, выполнение не остановится до конца данной итерации (каждый раз, когда PHP выполняет все операторы цикла, называется одной итерацией цикла). Иногда, если «условие» вычисляется в FALSE в самом начале цикла, вложенный оператор может быть не выполнен ни разу.

Как и в операторе if, можно создавать блок из операторов внутри цикла while с помощью фигурных скобок {} или используя альтернативный синтаксис:

while (условие) {…команды, которые должны выполняться, если условие верно…;}... endwhile;

**Основные команды, используемые в PHP**

В PHP, как и в любом другом языке программирования есть команды, с помощью которых можно выполнять определенные действия:

**еcho()** - одна из наиболее часто используемых команд, выводящая на web-страницу какой-либо текст, заключенный в кавычки.

**include()** «имя файла» - команда, вставляющая содержимое одного файла в другой. Файл именно вставляется, то есть пути к картинкам уже нужно указывать от того файла, где вставляется команда include

**return() -** если вызывается из функции, оператор return(**)** немедленно заканчивает выполнение текущей функции и возвращает свой аргумент как значение вызова функции. Оператор return() также заканчивает выполнение оператора [eval()](mk:@MSITStore:C:\Documents%20and%20Settings\Александр\Рабочий%20стол\ДИПЛОМ\PHP\PHP42\php42.chm::/f/eval.html) или файла скрипта. Если вызывается из глобальной области видимости, выполнение файла текущего скрипта заканчивается.

**require() -** ператор require() подключает/include и вычисляет определённый файл.Операторыrequire() и include() идентичны почти во всём, за исключением того, как они обрабатывают неудачное выполнение. include() выдаёт Warning!, а require() выдаёт Fatal Error.

### 4.1.4. Web – сервер Apache

Современная web-разработка во многих случаях требует использования динамических решений, написания скриптов на различных языках программирования, навыков работы с базами данных. В этом творческом процессе возникает необходимость отладки созданных программ на соответствующем тестовом сервере. Кроме самого сервера, нам еще понадобятся интерпретаторы необходимых языков программирования, а также СУБД.

На сегодняшний день самым популярным web-сервером является Apache. Этот продукт распространяется свободно, а благодаря открытому коду пользуется благосклонностью разработчиков дополнительных модулей. Поддержкой и координацией развития Apache занимается группа Apache Server Foundation, а сам сервер признается одним из самых крупных достижений разработчиков свободного ПО.

Совместно с Apache, как правило, используются языки программирования РНР и Perl, хотя могут быть и другие варианты - это зависит от поставленных задач. Это также свободные продукты с большим числом поклонников, на деле показавшие свою пригодность для применения в web-проектах самого различного плана и уровня.

Что касается СУБД, то достаточно часто для решения подобных задач используют пакет MySQL. Основные его достоинства - это достаточная функциональность и быстродействие в совокупности с простотой разработки приложений с ее использованием. Существуют и намного более мощные системы, в том числе и свободные, однако MySQL непрерывно развивается и не собирается сдавать свои позиции. Если говорить про условия приобретения, то пакет MySQL бесплатен для некоммерческого применения.

Поскольку, как мы уже отметили, разработка может вестись и на домашней машине, это в подавляющем большинстве случаев означает, что иметь дело надо будет с операционными системами от Microsoft. Названные пакеты могут работать на многих платформах, среди которых Unix, Linux, BSD, Solaris и даже MacOS. Для нас важен тот факт, что есть версии всех необходимых нам пакетов, совместимые с ОС Windows.

Более того, есть решение, позволяющее избежать процедуры ручной установки и настройки их по отдельности. Это так называемые "комплексы", которые включают Windows-версии Apache, MySQL и реализации некоторых популярных языков программирования. Данные компоненты заранее отконфигурированы, а ведь именно на этом этапе делается наибольшее количество ошибок людьми, не работавшими ранее с подобным программным обеспечением.

Подобные пакеты можно, в принципе, использовать не только по прямому назначению, то есть для тестирования собственноручно написанных скриптов для своих web-страниц, но и в роли полноценного сервера в рамках, например, локальной сети.

### 4.1.5. Пакет «Денвер»

Этот пакет разработан российскими программистами. Название его расшифровывается следующим образом: "ДжЕнтльменский Набор ВЕб-Разработчика". Он отличается завидной для своих возможностей компактностью (размер дистрибутива всего около двух мегабайт) и наличием поддержки как PHP, так и Perl. Конфигурация "Денвера" текущей версии такова:

* Apache 2.2.4 + SSL;
* PHP 5.2.12 с поддержкой GD, MySQL, SQLite;
* MySQL 5.0.51a;
* PhpMyAdmin 3.2.3;
* Ядро Perl без стандартных библиотек.

Кроме названных компонентов, "Денвер", как и положено такой системе, обладает единым инсталлятором, а также рядом оригинальных дополнений, например основанной на шаблонах системой управления виртуальными хостами Apache. Совсем не лишним представляется и переведенные на русский язык комментарии в конфигурационных файлах.

Как говорится в сопроводительной документации, "Предлагаемый комплекс программ позволяет запустить полноценный веб-сервер Apache с поддержкой PHP, Perl и сервер MySQL на машине, работающей под управлением MS Windows 95/98/Me/NT/2000/XP. Как правило, при установке не требуется никакой настройки, комплекс может использоваться даже неподготовленными пользователями. Тем не менее, он может найти применение не только для обучения азам веб-программирования, но и для отладки интерактивных сайтов, программ, использующих базу данных MySQL, написанных на языках PHP и Perl” - трудно добавить к этому что-либо от себя, разработчики выразились более чем доступно.

Существует два варианта запуска "Денвера". Первый предусматривает создание виртуального диска при загрузке Windows. Старт серверов осуществляется вручную. Этот вариант считается предпочтительным, так как дает некоторые дополнительные возможности при работе с командной строкой и установленными программами без запуска самого Apache. Второй подразумевает создание виртуального диска непосредственно при ручном запуске серверов. В рамках данного дипломного проекта был выбран первый вариант запуска.

**Архитектура разрабатываемой системы расчета структурной надежности**

В данной системе расчета структурной надежности имеется база данных, в которой находятся все необходимые компоненты для используемых в лабораторной работе схем. В БД также имеются базовые значения интенсивности отказов для каждого элемента.

Исходными данными для расчета надежности служат:

1) схема устройства с перечнем входящих в нее элементов, интенсивности отказов которых известны;

2) параметры внешних воздействий, условия эксплуатации;

3) время работы и хранения устройства (блока, платы);

4) электрическая нагруженность элементов, учитываемая коэффициентами нагрузки по напряжению и мощности.

Рассмотрим основополагающие факторы, влияющие на выбор архитектуры информационной системы.

Архитектура информационной системы будет зависеть от типа базы данных, то есть от выбора СУБД. Так как информационная система предназначена для использования в лабораторном практикуме, проводимом в течение одного семестра для студентов двух-трех групп, то количество её пользователей, вероятно, будет не велико.

При построении архитектуры информационной системы нужно учесть ее основную особенность, а именно web – направленность. Web-приложения представляют собой особый тип программ, построенных по архитектуре "клиент-сервер". Особенность их заключается в том, что само Web-приложение находится и выполняется на сервере - клиент при этом получает только результаты работы. Работа приложения основывается на получении запросов от пользователя (клиента), их обработке и выдачи результата. За счет наличия исполняемой части, Web-приложения способны выполнять практически те же операции, что и обычные Windows-приложения, с тем лишь ограничением, что код исполняется на сервере, в качестве интерфейса системы выступает браузер, а в качестве среды, посредством которой происходит обмен данными, - сеть (локальная или глобальная). Таким образом, информационная система состоит из 2 частей:

* **Веб-сервер** — это сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, обычно вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.
* Веб-обозреватель или браузер  — программное обеспечение для поиска, просмотра веб-страниц, то есть для запроса веб-страниц (преимущественно из Сети), для их обработки, вывода и перехода от одной страницы к другой.

Архитектура системы расчета надежности приведена на рис. 4.1

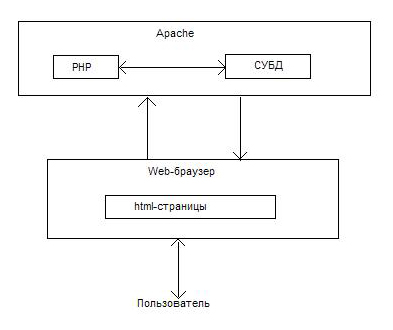


Рисунок 4.1- Архитектура системы расчета надежности.

Принцип работы PHP и СУБД. Имеется скрипт PHP (набор команд). Также есть сервер SQL, они существуют раздельно и друг с другом не связаны. PHP с помощью специальных команд устанавливает связь с сервером SQL. Как правило, PHP посылает запрос в базу данных. SQL обрабатывает эти команды и возвращает результат. Все это скрыто для пользователя.

Для пользователя отображаются только с результатами работы PHP и SQL в браузере в виде html-страницы.

## 4.1. Разработка базы данных

### 4.1.1. Концептуальное проектирование базы данных

Прежде чем приступать к разработке самой БД, определять сущности и связи, разработчик, прежде всего, должен уяснить, для кого разрабатывается база данных, и что она должна делать, какие функции выполнять.

На данном этапе собирается максимальный объем информации, относящейся к предметной области, для которой планируется разработать базу данных. Предметная область - часть реального мира, подлежащая изучению с целью организации управления и, в конечном счете, автоматизации. Каждый фрагмент предметной области характеризуется множеством объектов и процессов, использующих объекты, а также множеством пользователей, характеризуемых различными взглядами на предметную область. Информацию можно получить либо посредством изучения документов, характеризующих предметную область, либо при помощи интервью с заказчиком, заинтересованным в разработке.

Для того, чтобы построить модель будущей базы, необходимо произвести анализ, в результате которого будут определены объекты-сущности, информацию о которых необходимо будет хранить, а также определить как взаимодействуют объекты между собой.

Разрабатываемая информационная система предназначена для хранения данных о надежности элементов. С нашей БД могут работать студенты при выполнении лабораторной работы.

Вторым шагом в разработке БД является описание предметной области. Причём, чем точнее и глубже разработчик будет вести это описание, тем полнее и правильнее получиться сама БД, и только тогда она будет работать наиболее эффективно.

Основная таблица базы данных будет содержать следующие поля:

- название элемента;

- тип элемента. Схема объекта расчета надежности состоит из элементов разного типа (резисторы, диоды, интегральные микросхемы и тд). При расчете надежности с учетом различных условий эксплуатации выбирается соответствующий коэффициент, свой для каждого типа элемента;

- базовая интенсивность отказов элемента. Здесь находится базовое значение интенсивности отказов для каждого элемента;

- интенсивность отказов элемента в режиме хранения. Наша система позволяет исследовать элементы не только при непосредственной их работе, но также и учитывать изменения в их надежности при хранении. Поэтому в базу данных добавлен и этот параметр.

### 4.1.2. Логическое проектирование базы данных

Разработку логической модели модно осуществлять различными методами. Наиболее формализованным и простым для понимания является метод «сущность-связь», или ER-метод.

Логическая структура базы данных определяет:

• таблицы и их имена, также называемые *сущностями* (entities);

• имена полей, также называемые *атрибутами* (attributes) каждой таблицы;

• характеристики полей, например уникальность их значения и допустимость значений NULL, а также тип данных, хранимых в поле;

• первичный ключ каждой таблицы — поле (несколько полей) со значениями, уни­кально идентифицирующими каждую запись в таблице. В таблице также могут существовать другие уникальные поля, но только одно из них рассматривается как уникальный ключ доступа для поиска записей — первичный ключ. В таблице не обязательно должен существовать первичный ключ, однако рекомендуется оп­ределять его для каждой таблицы;

• связи между таблицами. Записи в таблице могут зависеть от одной или несколь­ких записей другой таблицы. Такие отношения между таблицами называются *связями.* Связь определяется следующим образом: поле или несколько полей одной таблицы, называемое *внешним ключом,* ссылается на первичный ключ другой таблицы.

На основе вышеописанных рассуждений введем следующие сущности и их атрибуты.

1. Справочник (Название, Тип элемента, Базовая интенсивность отказов, Вариант).
2. Механическое воздействие (Условия эксплуатации, Коэффициент при вибрации, Коэффициент при ударной нагрузке, Коэффициент, учитывающий суммарное воздействие)

База данных, составленная по этой информационной модели, обладает избыточностью, отношения не нормализованы.

Нормализация – это процесс улучшения качества отношений между таблицами и, как следствие, свойств базы данных в целом. Основополагающим принципом является наложение некоторых ограничений и правил на структуру таблиц в базе данных.

В теории реляционных баз данных выделяют следующие нормальные формы:

1. Первая нормальная форма (1NF).

2. Вторая нормальная форма (2NF).

3. Третья нормальная форма (3NF).

4. Нормальная форма Бойса – Кодда (BCNF).

5. Четвертая нормальная форма (4NF).

6. Пятая нормальная форма (5NF).

Мы рассмотрим только первые три. Вызвано это тем, что приведение базы данных к третьей нормальной форме включительно является достаточным условием для того, чтобы база данных соответствовала основным правилам построения качественной базы данных: доступность, расширяемость, непротиворечивость.

Таблица находится в 1НФ, если все ее атрибуты содержат только данные простых типов. Таблица не должна иметь повторяющихся записей, групп полей, необходимо наличие, по крайней мере, одного уникального индекса.

В рассматриваемой базе данных все отношения (таблицы) находятся в 1НФ, т.к. имеют ключевые поля, которые по определению являются уникальными индексами.

Таблица находится во 2НФ, если она находится в 1НФ и каждый ее неключевой атрибут (тот, который не входит в состав первичного ключа) полностью зависит от первичного ключа.

Понятие третьей нормальной формы основывается на понятии нетранзитивной зависимости. Транзитивная зависимость наблюдается в том случае, если один из двух описательных реквизитов зависит от ключа, а другой описательный реквизит зависит от первого описательного реквизита. Таблица находится в 3НФ, если она находится во 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Наша БД имеет не такую сложную структуру, поэтому ее необходимо привести лишь к 1НФ, введя ключевое поле id.

Тогда сущность Справочник будет иметь следующий вид:

Справочник (Id, Название элемента, Тип элемента, Базовая интенсивность отказов, интенсивность отказов в режиме хранения, Вариант).

Для создания логической и физической структуры базы данных использовался пакет PhpMyAdmin 2.6.1.

PhpMyAdmin - это некоммерческое приложение, написанное на языке PHP, реализующее довольно удобный и функциональный Web-интерфейс к базе данных MySQL.

С помощью PhpMyAdmin можно создавать, удалять и редактировать таблицы, выполнять отдельные SQL-запросы, создавать текстовые дампы данных, просматривать список процессов, создавать и удалять пользователей, изменять их привилегии и многое другое.

## 4.2. Разработка пользовательского интерфейса

Данная система расчета надежности реализована в виде web-приложения, что позволяет работать с ней не только непосредственно в университете, но и удаленно, что является преимуществом перед существующими аналогами. Также эта система удобна тем, что не требует для своей работы установки специального программного обеспечения.

В системе расчета надежности имеется возможность самостоятельного ввода параметров внешнего воздействия на схему (температура, механическое воздействие, условий эксплуатации), параметров нагруженности элементов, выбор корпуса, коэффициента приемки. Также имеется возможность ввода времени нахождения аппаратуры в полете и на земле, времени работы, хранения оборудования

При входе в систему пользователь попадает на главную страницу. Здесь предлагается создать уровень (блок) в котором будут располагаться подуровни (платы).

**4.2.1. Разработка структуры пользовательского интерфейса**

На сегодняшний день, создание внутренних корпоративных веб-приложений уже, пожалуй, не просто дань моде, когда все, что так или иначе связано с интернетом считалось популярным и прогрессивным. Бизнес-приложения предприятий перенесенные на новую технологическую "веб-оснастку" действительно значительно уменьшают издержки по поддержанию данных приложений в актуальном состояние в дальнейшем. Вот краткий перечень достоинств, которыми обладают корпоративные веб-приложения:

* не требуют инсталляции и обновления клиентского программного обеспечения;
* снижают затраты на обучение - в качестве клиентской части используется стандартный веб-браузер;
* пользователи могут работать на любой платформе;
* логика приложения сосредоточена на стороне сервера;
* возможность интеграции с ресурсами интернета;
* создание сколь угодно привлекательного веб-интерфейса.

Если мы будем рассматривать веб-приложения с точки зрения программиста, то представить их можно как некий особый класс систем клиент-сервер, в которых взаимодействие с пользователем осуществляется через стандартный веб-браузер. Если сравнивать структуру веб-приложения и классического приложения, основанного на технологии клиент-сервер, то можно охарактеризовать составляющие компоненты следующим образом:

* В роли "тонкого клиента", который отвечает за отображения данных и передачу их от пользователя серверу, выступает браузер. Пользовательский интерфейс всецело определяется HTML-документом, со всеми возможными дизайнерскими ухищрениями.
* Веб-сервер обеспечивает работу по протоколу HTTP, принимает запросы от клиента, взаимодействует непосредственно с веб-приложением, передает ответы клиенту. Веб-приложение - программа, которая, используя веб-сервер, обрабатывает запросы от клиента, производит необходимые манипуляции с данными, передает ответы клиенту.

Таким образом, разрабатывая интерфейс приложения нужно учитывать его web – ориентацию. В качестве наиболее подходящего было выбрано приложение, имеющее одну главную страницу со ссылочным меню в левой части страницы.

### 4.2.2. Ввод данных

В системе расчета надежности пользователю предлагается ввести условия эксплуатации для заданной схемы:

- температура. Пользователю предлагается выбрать температуру из интервала 25..60 °C.

- условия эксплуатации. Здесь предлагаются на выбор следующие условия: лабораторные, стационарные, автофургонные, железнодорожные, корабельные, самолетные. Также имеется возможность выбора дополнительных условий использования аппаратуры. Это эксплуатация при вибрации, при ударных нагрузках, а также суммарное воздействие.

- влажность. Предлагается выбрать значения: 60..70% и 90..98%;

- коэффициент нагрузки по напряжению. Набор значений от 0.2 до 1;

- высота. Имеется возможность выбора высоты, на которой используется оборудование. Предлагается множество значений от 0 до 40 км;

- время хранения и эксплуатации аппаратуры. Предусмотрены поля для ввода количества часов, проведенных аппаратурой в режиме хранения и эксплуатации;

- рабочая величина среднего прямого тока через диод;

- рабочее напряжение (для конденсатора);

- тип корпуса (для микросхем), предлагается выбрать пластмассовый или металлический корпус;

- приемка. ВП (5) и ОС (9).

Далее после ввода необходимых данных следует нажать кнопку «Расчет», после чего необходимая информация передастся файлу-обработчику, с помощью которого будет произведен расчет необходимых параметров.

### 4.2.3. Обработка введенных пользователем данных

В системе имеется база данных, в которую занесены элементы с их базовыми значениями интенсивности отказов. При выборе пользователем параметров эксплуатации базовые значения интенсивности отказов подставляются в соответствующие формулы и рассчитывается интенсивность отказов системы, а также такие параметры как вероятность безотказной работы, гамма-процентная наработка.

При вводе условий эксплуатации значение коэффициента выбирается исходя из двух условий (условий эксплуатации и условий механического воздействия). Поэтому в базе данных создана таблица, из которой в зависимости от действий пользователя выбирается коэффициент.(рис.4.2)



Рисунок 4.2 – Таблица коэффициентов различных условий механического воздействия.

Запрос к базе данных осуществляется функцией mysql\_query. Затем отобранные по заданным в запросе параметрам значения интенсивности отказов подставляются в формулу, и рассчитывается суммарная интенсивность отказов с соответствующими для каждого типа элементов коэффициентами. Полученный результат отображается на странице.

*35 страниц для этого раздела – слишком много! Надо убрать все, не имеющее прямого отношения к решаемой задаче, а оставить только существенное. То есть, если, напр., надо выбрать для прибора фотоприемник на ПЗС, то не нужно подробно описывать физику накопления и переноса зарядов.*