функционирования приложения на базе softPLC-контроллера в режиме реального времени. При этом наиболее важным свойством контроллера должно быть не только выполнение задания в определенный промежуток времени, но и способность выполнять циклически повторяющиеся задачи всегда в одно и то же время.

Приведем сравнительный анализ классического ПЛК на базе специализированного устройства и программно реализованного ПЛК на базе x86 совместимого промышленного компьютера.

Сравнительный анализ классического ПЛК и SoftPLC представлен в таблице 1.

Таблица 1. Сравнительный анализ классического ПЛК и SoftPLC

Классический PLC	SoftPLC
проектирование, построение архитектуры и реализация логической задачи жестко заданы и зависят от номенклатуры существующих на рынке ПЛК	нет необходимости в дополнительном оборудовании, так как для вычисления используются ресурсы промышленного компьютера с весьма большой производительностью и являющимся универсальным контроллером, требуется только поддержка протоколов связи с модулями управления
программирование классического внешне реализованного контроллера часто осуществляется вне системы	программирование SoftPLC осуществляется как в среде разработки, так и в среде самой системы управления
в рамках такого построения систем управления возникают дополнительные накладные расходы при передаче данных от ПЛК к системе управления и обратно	программно реализованный логический контроллер позволяет передавать данные по любому виду протоколов связи, обмен информацией получение данных осуществляется без дополнительных накладных расходов
требуется наладка	так как softPLC является программной реализацией, то есть возможность быстрой модернизации системы без длительной остановки и наладки оборудования, за счет установки обновлений
затруднена возможность диагностики, установки обновлений через интернет	при программной реализации возникает возможность диагностики, установки обновлений и устранения ошибочных ситуаций посредством удаленной работы через Internet

Для передачи управляющих воздействий к исполнительным устройствам, системе управления требуются входы/выходы, которые предоставляет баскаплер (buscoupler). Это устройство, в состав которого входят дискретные и аналоговые модули ввода/вывода, модули для подключения датчиков, задающих и исполнительных устройств, работающие по различным цифровым и аналоговым интерфейсам, коммуникационные модули (RS-232, RS422/RS-485, PROFINET IO, EtherCAT, и др.), а так же базовый интерфейсный модуль, для подключения баскаплера к полевой шине. Базовый модуль выполняет роль транслятора управляющих сигналов, и в отличии от ПЛК не имеет в своём составе мощного микроконтроллера для выполнения задач управления. Эти задачи решаются на программном уровне в системе управления, при помощи программно реализованного контроллера (softPLC).

В совокупности применение softPLC является актуальным способом автоматизации технологических процессов который активно заменяет системы на базе классического ПЛК.

Список использованных источников:

1. IEC DIS 61131-3 Programmable Controllers Programming Languages, Draft International Standard / International Electrotechnical Commission, 1997.

2. Мишель Ж. Программируемые контроллеры: архитектура и применение / Ж. Мишель. – Москва: Машиностроение, 1992.

3. Шемелин В.К. «Программная реализация логической задачи числового программного управления (ЧПУ) на основе контроллера типа Soft PLC» / В.К Шемелин, Р.А Нежметдинов // Объединенный научный журнал. - 2008. - №10. - Москва.

ОШИБКИ И РАБОТА С ИСКЛЮЧИТЕЛЬНЫМИ СИТУАЦИЯМИ В ПЛАТФОРМЕ ORACLE

Институт информационных технологий БГУИР, г.Минск, Республика Беларусь

Воробей К.П.

Савенко А.Г. – магистр технических наук, ассистент

Ошибки являются неотъемлемой частью работы любых современных платформы. Чем больше система, тем больше количество исключительных ситуаций, которые могли пропустить на моменте проектирования и тестирования.

Продуктами компании Oracle пользуются большое количество организаций, к ним относятся: банки,

большие розничные магазины, заводы, налоговые службы. Так же продукты Oracle нашли применение в образовательном процессе (например, результаты ЦТ в РИКЗ). В процессе работы каждый день образовываются миллионы транзакций и для данных систем необходимо постоянно дописывать функционал и вносить правки в бизнес-логику. В основе каждой системы лежит база данных. Большая часть конфликтных ситуаций возникает при внесении данных в базу, сотни специалистов обеспечивают своевременное устранение проблем.

При работе с базами данных, важной составляющей является не только обработки ошибок, но и формирование наиболее целостных валидных сообщений. Формирование сообщений об ошибках позволяет быстрее обнаружить причины и исправить исправлять ошибки. Особенно актуально при работе с продуктами с большой бизнес-логикой так как в большинстве случаев не известны особенности структуры конкретной БД.

Формирование сообщений об ошибках в программах часто сильно отличается от обработки самих ошибок. При обработке ошибок удается выработать общую стратегию, что позволяет сформировать их обработку в одной или нескольких функциях. Такой же подход для сообщений об ошибках может быть реализован на основе того, что в сообщении об ошибке сервер Oracle указывает тип ошибки и объект базы данных, который явился причиной её возникновения. Такими объектами обычно являются ограничения, как, например, первичные, уникальные индексы, ограничения "not null", уникальные и внешние ключи и др. [1]. Из системных таблиц базы, данных может быть получена подробная информация об ограничениях и определены значения, изменение которых и привело к возникновению ошибки. Но проблема заключается в том, что реализация такого механизма формирования сообщений об ошибках в реальных приложениях встречает целый ряд проблем:

Зависимость сообщения об ошибке от назначения программы. Даже для программ, работающих с одной базой данных, может потребоваться формирование различных сообщений об одной и той же ошибке. Например, в программе для редактирования данных пользователем сообщение должно быть: “Работник с такими паспортными данными уже существует! Проверьте введённые значения!”. А в программе экспорта данных требуется сообщение с другим содержанием: “Импортируемые данные дублируются – проверьте данные, за которую выполняется импорт данных!”.

Сложность создания сообщений об ошибках, вызванных ограничениями базы данных. Например, в ограничениях CHECK для таблиц могут использоваться довольно сложные запросы и условия. Поэтому формирование сообщений на основе их анализа может оказаться довольно сложной задачей [2].

Использование в клиентских программах пользовательских названий таблиц и столбцов, отличных от их имен в БД. Например, таблица имеет имя “item_loc”, а в клиентском приложении данные этой таблицы могут отображаться в справочнике как “Товары” или “Локация”.

Существует так же большой количество ошибок, которые можно было решить на стадии проектирования БД. Выделим основные подходи и способы устранения проблем.

Для оценки большого набора хранимых процедур Oracle и пакетных скриптов, которые используются как часть процесса сборки и обновления в большинстве случаев возникает проблема, которая заключается в том, что процесс сборки не является стабильным. Во время каждой сборки происходит большое количество ошибок, например, таблица не была создана или загружена так, как должно быть и команде разработчиков требуется большое количество времени для устранения неисправности.

При проведении оценки необходимо смотреть, насколько хорошо код форматируется, структурирован (модульно), документирован и организован, также смотреть, насколько он прочен (то есть он может корректно обрабатывать сбой), и насколько хорошо и безотказно выполняется.

Так же для удобства работы необходимо документировать и формировать модули. Пакетные скрипты должны быть хорошо организованы, благодаря чему было довольно легко понять процесс. Сами хранимые процедуры PL/SQL должны быть достаточно качественно написаны. Они должны быть отформатированы и правильно использованы некоторые из более сложных возможностей Oracle PL/SQL, такие как переменные курсора и CURSOR FOR LOOPs и другие, когда это необходимо.

При качественном написании скриптов необходимо помнить о обработке исключений и протоколирования всего процесса. Хранимые процедуры или сценарий PL/SQL должны проверяется на наличие ошибок.

Таким образом, даже если сама система сборки была разработана высококвалифицированным разработчиком, отсутствие хорошей обработки исключений снизило качество всей системы. Есть два основных этапа обработки исключений, которые, должны быть помещены во все PL/SQL-коды, а некоторые - в любой большой или повторяющийся процесс.

Вот несколько из них:

Необходимо включать обработчик исключений для каждой логической группировки операторов PL/SQL (блок BEGIN/END). Это делает две вещи: во-первых, это вынуждает разработчика задуматься о крайних случаях, которые могут пойти не так, чтобы они могли справиться с ними соответствующим образом. Это само по себе улучшает качество кода и стабильность системы. Во-вторых, это дает возможность определить, на каком уровне должно обрабатываться исключение. Это критически важно для обеспечения того, чтобы процесс продолжался, когда он должен и останавливается, когда это не должно быть. Это особенно важно, когда используются вложенные процедуры или сценарии PL/SQL.

В любом месте транзакции можно устанавливать точки сохранение, которая позволит отменить любые, внесённые в неё после. Точки сохранения широко используются в очень длинных транзакциях для минимизации числа команд, которые нужно отменять при сбое транзакции [3].

Регистрировать всё, особенно исключения. Создавая отдельную процедуру ведения журнала, которую вы вызываете в начале, в различных точках внутри кода и в блоках исключений, при выполнение данного условия вы всегда сможете точно знать, какие процессы были завершены до сбоя. Дополнительным преимуществом этого подхода является то, что запись о том, когда каждый процесс происходил, помогает в

значительной степени с настройкой производительности, потому что вы можете легко определить, какие процессы занимают больше всего времени. Процедура, подобная этой, может принимать следующие параметры:

Имя хранимой процедуры, из которой он вызывается, индикатор позиции в хранимой процедуре в этой процедуре текстовая строка, состоящая из конкатенации значений ключевых переменных, которые используются в процедуре. Текстовая строка, указывающая ошибку, если вызывается внутри блока исключения. Эти значения параметров просто записываются в таблицу и, необязательно, в текстовый файл через DBMS_OUTPUT. Это очень помогает в решении проблем позже.

Существуют и другие шаги, которым можно предпринять для дальнейшего расширения обработки исключений. Но простые шаги, которые описаны значительно улучшат стабильность кода и способность находить и устранять любые возникающие ошибки.

Список использованных источников:

[1] Лихачёв, В. Особенности обработки ошибок сервера базы данных Oracle // Oracle Magazine - Русское издание [Электронный ресурс]. – 2009. – Режим доступа: http://citforum.oldbank.com/database/oracle/error_handling/. – Дата доступа: 29.03.2017.

[2] Mishra S., Beaulieu A./ Mastering Oracle SQL, 2nd Edition // Издательство "O'Reilly Media". – США 2004 – С. 314.

[3] Перри Д. Введение в Oracle 10g / Джеймс Перри // Издательство "Вильямс". – Киев, 2006. – С. 127.

МЕТОДИКА ВЫБОРА БИОМЕТРИЧЕСКИХ СРЕДСТВ КОНТРОЛЯ ДОСТУПА К ИНФОРМАЦИОННОМУ ОБЪЕКТУ

Институт информационных технологий БГУИР, г.Минск, Республика Беларусь

Гивойно А.А., Ситник М.Ю., Нарижный Е.Ю.

Сечко Г.В. – канд. техн. наук, доцент

Предлагается несложная методика выбора биометрических средств контроля доступа к информационному объекту (средств аутентификация личности, запрашивающей доступ). Проводится краткий обзор существующих методологий аутентификации и реализующих их программно-аппаратных средств и на его основе с помощью предлагаемой методики решается задача выбора средства контроля доступа: какое средство купить – дешёвое, но с низкими техническими характеристиками, или дорогое, но с высокими техническими характеристиками. Пригодность предложенной методики иллюстрируется примером выбора пользователем одного из двух программно-аппаратных комплексов для распознавания личности по радужной оболочке глаза.

Надежная аутентификация, т. е. определение личности, запрашивающей доступ к информационному объекту, становится необходимым атрибутом повседневной жизни, сегодня люди используют ее при совершении самых обычных действий: при посадке на самолет, проведении финансовых операций и т. д. Существует три традиционных способа аутентификации (и/или авторизации, т. е. разрешения доступа к информационному ресурсу) [1]:

1. по собственности – физическим предметам, таким, как ключи, паспорт и смарт-карты;

2. по знаниям – информации, которая должна храниться в секрете и которую может знать только определенный человек, например, пароль или парольная фраза. Знания могут представлять собой относительно конфиденциальную информацию, которая может и не быть секретной, например, девичья фамилия матери или любимый цвет;

3. по биометрическим параметрам – физиологическим или поведенческим характеристикам, по которым можно отличить людей друг от друга. На наш взгляд, наиболее надёжной является аутентификация по биометрическим параметрам.

Биометрических средств контроля доступа к информационному объекту: (БСКДкИО) великое множество. В обзоре [2] довольно подробно описаны БСКДкИО, разрешающие доступ к информационному объекту на основе распознавания: а) отпечатков пальцев; б) радужной оболочки глаза (РОГ); в) лица; г) геометрии руки; д) голоса; е) сетчатки глаза; ж) ряда дополнительных биометрических параметров (по ДНК, по термограммам, по запаху тела и т. д.) [2]. Каждый способ доступа может быть реализован большим числом различных БСКДкИО (программно-аппаратных комплексов), различающихся ценой и техническими характеристиками. В этих условиях перед будущим пользователем БСКДкИО стоит задача выбора: какой БСКДкИО купить – дешёвый, но с низкими техническими характеристиками, или дорогой, но с высокими техническими характеристиками.

Для решения поставленной задачи в докладе предлагается простая методика: 1. Определить несколько вариантов предпочтительных БСКДкИО. 2. Выбрать набор технико-экономических показателей сравниваемых друг с другом БСКДкИО. 3. Каждый выбранный показатель представить в виде балльной шкалы (от 0 до 10 баллов, чем предпочтительнее БСКДкИО, тем выше балл). 4. С помощью эксперных оценок выбрать весовые коэффициенты каждого показателя так, чтобы сумма их равнялась единице. 5. Рассчитать критерий выбора БСКДкИО как скалярное произведение вектора выбранных показателей и вектора весовых коэффициентов (чем предпочтительнее БСКДкИО, тем выше критерий).

Предлагается набор технико-экономических показателей сравниваемых друг с другом БСКДкИО выбрать в виде трёх показателей: цена БСКДкИО, общая стоимость ущерба для пользователя в случае несанкционированного доступа к информационному объекту, суммарная вероятность ошибок идентификации в процентах (вероятность пропуска «чужого» плюс вероятность ложного отказа в доступе.