Основы администрирования СУБД OpenEdge 10.x

(У ч е б н ы й к у р с)

Санкт-Петербург 2013 г.

Документ представляет собой набор методических материалов, используемых во время практических занятий для изучения СУБД OpenEdge в рамках курса «Ключевые аспекты администрирования СУБД Progress OpenEdge v10-11».

Материал курса предназначен для изучения Progress® OpenEdge™ 10, тем не менее, поскольку основные аспекты администрирования OpenEdge неизменны и наследуются в новых версиях, этот материал также подойдет для изучения OpenEdge версии 11 и выше.

Автор курса: Башкатов Валерий Григорьевич

www: http://rupug.ru («Russian Progress User Group»)
E-mail: v.bashkatov@mail.ru

Оглавление

Глава 1 Введение в OpenEdge

Немного о возможностях

СУБД Progress® OpenEdge™ предназначены для эффективной разработки современных информационных приложений. В состав OpenEdge входит вся необходимая инфраструктура и инструменты, необходимые производителям программного обеспечения для разработки и поддержки функционирования бизнес-приложений. Гибкость технологий, заложенных в платформу OpenEdge, обеспечивает эффективную интеграцию, а наличие инструментов по анализу и управлению помогает оптимизировать производительность информационных систем.

СУБД Progress® OpenEdge™ разрабатывается и поддерживается корпорацией **Progress Software Corporation,** которая является одним из крупнейших поставщиков средств построения информационных систем промышленного масштаба, ориентированных на интенсивную обработку данных в реальном времени. Корпорация основана в 1981 году и имеет штаб-квартиру в городе Bedford (США). Официальным дистрибьютором Progress Software Corp. на территории стран СНГ и Латвии является компания **Progress Technologies**.

Основными характеристиками OpenEdge являются:

Характеристики	**Значения**
Размер блока базы данных	- 1024 байта - 2048 байт - 4096 байт (по умолчанию для Windows и Linux) - 8192 байта (по умолчанию для Unix)
Количество записей в блоке	- 1 - 2 - 4 - 8 - 16 - 32 (по умолчанию, если размер блока не равен 8192 байт) - 64 (по умолчанию, если размер блока равен 8192 байт) - 128 - 256
Максимальное количество областей хранения	32000, из них 31994 - это области для хранения пользовательских данных
Максимальное количество экстентов областей хранения	1024
Максимальный размер одного экстента	1 ТБ*
Максимальный размер области хранения	1 ПБ*
Максимальное количество записей в одной области	Рассчитывается по формуле: (максимальный размер области) x (количество записей в блоке) / (размер блока базы данных)
Максимальное количество таблиц	32767
Максимальное количество полей в таблице	- 500, для SQL - 1000, для ABL
Максимальное количество индексов	32767
Максимальное количество полей в индексе	16
Максимальное количество секвенций	Зависит от размера блока базы данных: - 1024 байт - 250 - 2048 байт - 500 - 4096 байт - 1000 - 8192 байт - 2000

Характеристики	Значения
Максимальный размер области Primary Recovery (BI)	Зависит от размера блока BI-области: - 16384 байт - 32 ТБ - 8192 байт (по умолчанию) - 16 ТБ - 4096 байт - 8 ТБ - 2048 байт - 4 ТБ - 1024 байт - 2 ТБ
Максимальный размер базы данных	Рассчитывается по формуле: (количество областей) x (максимальный размер области). Поскольку первые шесть областей зарезервированы для системных нужд, то реальное их количество 31994. В итоге: 31994 x 1 ПБ ~ 32000 ПБ*
Максимальное количество одновременно подключенных пользователей к базе данных	10000, зависит от ограничений на количество семафоров, от допустимого количества процессов и от производительности сервера.
Максимальное количество одновременных транзакций	1 на каждое пользовательское подключение (т.е. 10000)
Максимальное количество сегментов разделяемой памяти	- 32 - для 32 битных систем - 256 - для 64 битных систем
Максимальный размер разделяемой памяти для базы данных	Определяется параметром запуска базы данных <-B>: - 125000000 - для 32-х битных систем - 1000000000 - для 64-х битных систем

*При условии, что включена поддержка больших файлов, и область хранения имеет тип Storage Area Type II (SAT-II).

Краткий обзор основных продуктов OpenEdge

В состав СУБД Progress® OpenEdge™ входит несколько отдельных продуктов, наличие лицензий на которые определяет функционал СУБД.

Каждый продукт может быть в любое время доустановлен к базовой инсталляции СУБД без необходимости изменения существующего программного кода пользовательского приложения.

Категория: средства разработки

Наименование	Описание	Модель лицензирования
4GL Development System	Символьная среда разработки.	Registered Client[1] или Named User[2]
WebSpeed Workshop	Среда разработки для создания HTML-ориентированных бизнес-приложений внутри корпоративных intranet-сетей и Интернет. WebSpeed Workshop состоит из интегрированного набора инструментов разработки для создания и тестирования Web-приложений, используя HTML-интерфейс для Web-браузеров.	Registered Client или Named User

[1] В модели лицензирования на основе лицензии **Registered Client** , пользователем может быть **Клиентское Устройство**, Устройство, НЕ управляемое Человеком или Серверный Процесс.

[2] В модели лицензирования на основе лицензии **Named User**, пользователем может быть **Человек;** Устройство, НЕ управляемое Человеком; Серверный Процесс.

Разница между этими двумя моделями заключается в том, что понимается под Пользователем в каждом случае. Количество лицензий Registered Client учитывается по общему числу рабочих мест, с которых может производиться доступ к Progress-продуктам, количество лицензий Named User - по общему числу пользователей, которые могут иметь доступ к Progress-продуктам.

Наименование	Описание	Модель лицензирования
OpenEdge Studio	Интегрированная среда, включающая набор инструментов для разработки приложений. OpenEdge Studio предлагает разработчикам гибкость при реализации методологий разработки приложения и помогает быстро осуществить поставленные цели. Среда также включает Progress Dynamics, предлагающий репозиторный подход к построению приложений, спроектированных для усиления их структуры и бизнес-логики. Разработчики могут применять OpenEdge Studio для реализации объектно-ориентированного бизнес-подхода при проектировании приложений.	Registered Client или Named User
OpenEdge Architect	Интегрированная среда разработки для построения приложений, основанная на стандарте Eclipse framework, позволяет разработчику компоновать сервисно-ориентированные приложения из каталога компонентов и сервисов.	Registered Client или Named User

Категория: Клиентские продукты

Наименование	Описание	Модель лицензирования
Client Networking	Обеспечивает доступ ABL-клиентов к удаленным базам данных OpenEdge, OpenEdge Application Server или OpenEdge Data Server. Содержит среду выполнения приложения SQL Client Access и поддержку GUI и символьного клиента.	Registered Client или Named User
OpenEdge Transparent Data Encryption Необходим предварительно установленный продукт Enterprise RDBMS	Механизм обеспечивает шифрование данных, находящихся в состоянии покоя на диске. Процесс шифрования не является отдельной операцией, необходимой для выполнения пользователем, вся работа по шифрованию данных выполняется в фоновом режиме. Использование этого продукта не требует изменения существующего приложения, работающего с базой данных OpenEdge. **Примечание:** Количество лицензий OpenEdge Transparent Data Encryption должно соответствовать количеству лицензий OpenEdge Enterprise RBDMS.	Add-On Registered Clients или Add-On Named User или Add-On Access Agent
OpenEdge Replication Plus Необходим предварительно установленный продукт Enterprise RDBMS	Продукт предназначен для организации процесса репликации данных с промышленной базы на одну или две резервных. Фактически с помощью продукта можно организовать «горячее резервирование» промышленной базы данных. Этот процесс можно настроить так, чтобы в случае сбоя на промышленной базе одна из «горячих копий» была быстро переведена в режим промышленной базы, тем самым обеспечивается минимальное время простоя системы. К «горячим копиям» можно подключаться с целью формирования различной отчетности, при этом подключение выполняется в режиме ERO[3] без возможности изменять данные. Эта особенность позволяет перенести нагрузку, связанную с формированием отчетности, с промышленной базы данных на её копии, тем самым улучшая производительность промышленного сервера. **Примечание:** Количество лицензий OpenEdge Replication Plus должно соответствовать количеству лицензий OpenEdge Enterprise RBDMS.	Add-On Registered Clients или Add-On Named User или Add-On Access Agent

[3] Режим ERO - от англ. Enhanced Read Only

Категория: СУБД Progress

Наименование	Описание	Модель лицензирования
OpenEdge Personal RDBMS	Однопользовательская локальная база данных для разработки и внедрения. Включает среду выполнения приложения для локального внедрения, а также продукт client networking для доступа к удаленным серверам или к OpenEdge DataServer. Включает поддержку SQL-92 и драйверов для ODBC и JDBC для доступа к SQL-92. Имеет поддержку символьного клиента.	Registered Client или Named User
OpenEdge Workgroup RDBMS	Многопользовательский сервер реляционной базы данных, настроенный на поддержку приложений уровня рабочих групп и отделов. Предназначен как для сред разработки, так и для эксплуатации приложений. Это среда выполнения для локальных приложений с возможностью установки соединения с удаленными клиентами, в том числе с ODBC-приложениями. Включает поддержку SQL-92 и драйвера к ODBC и JDBC для доступа к SQL-92. Включает поддержку символьного клиента. **Примечание**: при использовании с OpenEdge Application Server Basic Edition требуется одна и та же модель лицензирования для каждого продукта.	Registered Client или Named User или Access Agent
OpenEdge Enterprise RDBMS	Многопользовательский сервер реляционной базы данных, настроенный на поддержку многотомных распределенных приложений уровня предприятия. Продукт предназначен как для сред разработки, так и для эксплуатации приложений. Содержит сетевые средства обслуживания соединений с удаленными клиентами, включая клиентские приложения, поддерживающие стандарт ODBC. Включает в себя так же поддержку SQL-92 и драйвера для ODBC и JDBC для доступа к SQL-92. Поддерживает символьного клиента. **Примечание:** при использовании с Open Edge Application Server Enterprise Edition требуется одна и та же модель лицензирования для каждого продукта.	Registered Client или Named User или Access Agent

Категория: продукты Data Server

Наименование	Описание	Модель лицензирования
OpenEdge DataServer for Microsoft SQL Server	Обеспечивает возможность взаимодействия с Microsoft SQL Server для OpenEdge-клиентов. При помощи ODBC-интерфейса продукт Data Server for MS SQL Server обеспечивает прямой доступ разработчиков к системам управления базами данных SQL Server и интеграцию этих источников данных внутрь различных приложений. Поддержка ODBC-драйверов включена в возможности RDBMS. Data Server for MS SQL Server обеспечивает простоту внедрения, высокую производительность соединений, поддерживает хранимые процедуры. Поддержка различных клиентских и серверных платформ позволяет продукту Data Server for MS SQL Server обеспечивать эффективную организацию сетей предприятия. **Примечание**: при использовании с Open Edge Application Server Enterprise Edition требуется одна и та же модель лицензирования для каждого продукта.	Registered Client или Named User или Access Agent

Наименование	Описание	Модель лицензирования
OpenEdge DataServer for ODBC	Обеспечивает соединение с различными ODBC - совместимыми источниками данных с целью разработки и внедрения мультипользовательских приложений. Включает сетевые средства обслуживания соединений с удаленными клиентами, многопользовательский доступ и поддержку хранимых процедур. Возможности конфигурирования клиента или сервера зависят от вида платформы. В этот продукт для соответствующих платформ включены драйверы DataDirect ODBC. Обеспечивается поддержка следующих баз данных: Sybase ASE 11.5, IBM DB2 UDB (6.1, 7.1, 7.2, 8.1), DB2/400 (V5R1, V5R2) и Informix Dynamic Server 9. Продукт содержит сетевые средства для обслуживания соединений от удаленных клиентов, многопользовательский доступ и поддержку хранимых процедур. **Примечание**: при использовании с Open Edge Application Server Enterprise Edition требуется одна и та же модель лицензирования для каждого продукта.	Registered Client или Named User или Access Agent
OpenEdge DataServer for Oracle	Обеспечивает разработчикам создание и внедрение OpenEdge приложений, способных обмениваться информацией с базой данных Oracle. Включает в себя Server Networking для обеспечения соединения с удаленными клиентами, а также поддержку хранимых процедур. Возможности конфигурирования клиента или сервера зависят от вида платформы. Спроектированный для компьютерных сред уровня предприятия продукт DataServer for Oracle поддерживает комплексные архитектуры: клиент/сервер, хост, Web и многозвенные конфигурации. Где бы ни находились данные - в одной базе данных Oracle или во многих серверах со сложной структурой - гибкость и высокая производительность DataServer for Oracle обеспечивает выгодное вложение средств в существующие технологии и дополняет информационную стратегию организации. **Примечание**: при использовании с Open Edge Application Server Enterprise Edition требуется одна и та же модель лицензирования для каждого продукта.	Registered Client или Named User или Access Agent

Категория: продукты Application Server

Продукты OpenEdge Application Server поддерживают: распределенные приложения; новые Web-технологии; соединение с другими приложениями через сеть или Интернет. Невидимый для конечных пользователей Application Server позволяет организовывать виртуальное взаимодействие с любым клиентом и источником данных.

Наименование	Описание	Модель лицензирования
OpenEdge Application Server Basic Edition Необходим предварительно установленный продукт Workgroup RDBMS или любой DataServer	Интегрированный Application Server для всех типов клиентов Progress 4GL (Client Networking и WebClient), HTML Web-приложений, Open Clients (Java, .NET и Web-севрвисы). Обеспечивает выполнение удаленных процедур 4GL, которые хранятся на сервере и вызываются клиентскими приложениями Progress 4GL, Java и .NET. Организовывает работу в сети для подсоединения к удаленным OpenEdge RDBMS серверам, другим OpenEdge Application Server или к OpenEdge DataServer. Также содержит Adapter for SonicMQ, AppServer Internet Adapter и WebSpeed Transaction Server. Basic Edition – это начальный уровень OpenEdge Application Server для малых или средних предприятий, которым требуется функциональность AppServer или WebSpeed без регулирования уровня загрузки, полных Web-сервисов или поддержки Sonic ESB. **Примечание**: при использовании вместе с OpenEdge Workgroup RDBMS одни и те же модели лицензирования должны использоваться для каждого продукта. Этот продукт ограничен одним брокером (WebSpeed Transaction Server или AppServer) на одну лицензию.	Registered Client или Named User или Access Agent
OpenEdge Application Server Enterprise Edition Необходим предварительно установленный продукт Enterprise RDBMS или любой DataServer	Интегрированный Application Server для всех типов клиентов Progress 4GL (Client Networking и WebClient), приложения HTML Web, Open Clients (Java, .NET и Web-сервисы). Обеспечивает выполнение удаленных процедур 4GL, которые хранятся на сервере и вызываются Progress 4GL, Java и .NET клиентскими приложениями. Включает в себя организацию работы в сети для подсоединения к удаленным OpenEdge RDBMS серверам, другим OpenEdge Application Server. Также содержит OpenEdge Adapter for SonicMQ, OpenEdge Adapter for SonicESB, Secure AppServer Internet Adapter, инструменты Web-сервисов и WebSpeed Transaction Server. Enterprise Edition спроектирован для средних и крупных организаций. **Примечание**: при использовании вместе с OpenEdge Enterprise RDBMS одни и те же модели лицензирования должны использоваться для каждого продукта.	Registered Client или Named User или Access Agent
NameServer Load Balancer Необходим предварительно установленный продукт Application Server Enterprise Edition	Обеспечивает балансировку нагрузки для OpenEdge Application Server Enterprise Edition путем направления запросов клиентов на свободные сервера.	Machine

Источники получения информации

В процессе работы с Progress-продуктами рано или поздно возникает необходимость в получении дополнительной информации, это могут быть как консультации в процессе изучения продукта, так и помощь в решении проблем. Источников получения этой информации много, основные из них представлены в следующей таблице:

Наименование	Описание	Язык	Сайт
Progress Communities	Официальный портал компании Progress Software. Здесь можно задать вопросы на интересующие вас темы, и получить ответы от сообщества «Пользователь Progress-продуктов». Помимо этого на сайте можно найти всю необходимую документацию. Требуется регистрация для использования форума.	Англ.	http://communities.progress.com
Progress Email Group (PEG)	Почтовая группа людей, объединенных одной идеей - помогать друг другу в работе с Progress-продуктами посредством электронной почты. Для задания вопросов в PEG необходимо выбрать один из почтовых адресов, который привязан к соответствующему направлению, список имеющихся направлений можно найти по этому адресу: http://www.peg.com/lists/. Основной форум имеет адрес: peg@peg.com. Для подписки на рассылку почтовых сообщений по тому или иному направлению необходимо воспользоваться специальной формой, которую можно найти по следующему адресу: http://www.peg.com/lists/subscribe.html	Агл.	http://www.peg.com
Русскоязычный форум о СУБД Progress	Русскоязычный форум, на котором можно найти множество полезных советов, а так же задать свои вопросы. Форум требует регистрации.	Рус.	http://forum.infobit.ru
DBA OpenEdge Progress	Русскоязычный сайт, посвященный администрированию СУБД Progress OpenEdge. На сайте можно найти большое количество полезных статей и книг по администрированию, и не только!	Рус.	http://www.openedge.ru
Progress Knowledge Base	Ежедневно обновляемая база знаний по Progress-продуктам. В этой базе можно найти ответы практически на все вопросы, связанные с OpenEdge. Базой стоит пользоваться в первую очередь, прежде чем обращаться к перечисленным ранее ресурсам. Это связано с тем, что в 90 процентах случаев в базе найдется решение для конкретной проблемы.	Англ.	http://progress.atgnow.com/esprogress/Group.jsp?bgroup=progress

Помимо этих сайтов всегда можно задать вопрос напрямую в Progress Tech Support (PTS). Для этого нужно воспользоваться контактной информацией, которую можно найти по следующему адресу: http://web.progress.com/en/support/contact-support.html. Стоит отметить, что все пользователи из стран СНГ относятся к группе EUROPE, MIDDLE EAST & AFRICA SUPPORT (EMEA). Учитывайте это при выборе контактной информации для обращения в PTS. При обращении в PTS будьте готовы к тому, что у вас запросят дополнительную информацию, например, для вашей идентификации, возможно понадобится предоставить серийный номер одного из ваших продуктов с действующей лицензией. Могут быть так же запрошены журнал базы данных и какая-либо статистическая информация по ее работе, либо что-то еще - всё зависит от сложности заданного вопроса.

Глава 2 Инсталляция OpenEdge

Получение доступа к Progress® Download Center

Progress® Download Center – центр загрузки Progress, представляющий собой web-сайт, с помощью которого можно получить быстрый доступ к программным продуктам и обновлениям от Progress Software Corporation. После регистрации на этом сайте будут доступны только те продукты, обновления на них и документация, на которые ваша организация имеет лицензии. На нем вы найдете не только последние версии, но и предшествующие версии купленных продуктов. Доступ к сайту поддерживается в режиме 24x7. Каждый раз при выходе очередного обновления, на электронную почту, которую вы указали при регистрации, будут приходить соответствующие уведомления. Количество пользователей, которые могут работать с Progress® Download Center от лица вашей организации, не ограничено.

Таким образом, с Progress® Download Center у вас всегда легкий и удобный способ доступа к дистрибутивам, обновлениям и к документации по вашим продуктам, независимо от вашего местоположения.

Progress® Download Center находится по следующему адресу: www.progress.com/esd

Для регистрации в Progress® Download Center необходимо обратиться к региональному поставщику программных продуктов корпорации Progress Software. Для стран СНГ и Латвии таким региональным поставщиком является компания Progress Technologies.

Для регистрации в Progress® Download Center присылайте запрос на адрес support@progress-tech.ru, который должен содержать следующие данные на английском языке:

- *Serial #
- *Registered To (название организации и присвоенный ей номер)
- *First Name
- *Last Name
- *Email
- Phone
- Fax
- Cell/Mobile Phone
- Greeting Name
- Username: (имя для регистрации)

* - поля, помеченные звездочкой, являются обязательными для заполнения.

Примечание: значения первых двух полей (Serial #, Registered To) находятся в документе к приобретенной лицензии.

Подготовка к началу инсталляции

После скачивания соответствующего дистрибутива, перед началом инсталляции необходимо убедиться в том, что дистрибутив соответствует операционной системе, далее выбрать каталог, в который будет выполняться инсталляция (необходимо от одного до двух гигабайт свободного пространства), и установить соответствующую версию JAVA. Для версии OpenEdge 10.2B требуется версия JAVA 5.0. **Внимание!** Использование версии JAVA, отличной от рекомендуемой, не гарантирует корректной работы OpenEdge.

Перед началом инсталляции OpenEdge убедитесь, что в переменной среды окружения $PATH задана правильная версия JAVA. Например:

```
export PATH=/usr/java/jre1.5.0_22/bin:$PATH
```

Необходимо подготовить файл с устанавливаемыми лицензиями. Лицензии обычно предоставляются на бумажном носителе, либо в виде файла в формате HTML. Последний всегда можно скачать самостоятельно с Progress® Download Center, такой файл называется License Addendum File. Файл с лицензией позволяет исключить рутинный ручной ввод лицензионной информации в процессе инсталляции, что особенно удобно при одновременной инсталляции нескольких продуктов.

Скачивание License Addendum File.

- Подключитесь к Progress® Download Center (http://www.progress.com/esd).
- На открывшейся странице Software Product List выберите соответствующий продукт из списка, например, Progress® OpenEdge™.
- На открывшейся странице Software Product Information выберите соответствующую версию продукта, например, Progress® OpenEdge™ 10.2.x.
- Попав на страницу со списком доступных продуктов и обновлений, связанных с выбранной версией, выберите необходимый продукт и щелкните по его названию в колонке Description, например, Progress® OpenEdge™ 10.2B Linux 32-bit.
- На открывшейся странице Software Terms and Conditions, предоставляющей пользовательское соглашение, щелкните кнопку «Accept» для перехода на страницу Software Product Download.
- На странице Software Product Download перейдите во вкладку License для перехода на страницу с лицензией выбранного продукта.
- Установите курсор мыши на область License Addendum и сохраните License Addendum File, щелкнув правой кнопкой мыши и выбрав меню Save Page As. Сохраните документ в виде html-файла. Рекомендуется файлу присвоить уникальное имя, особенно если необходимо скачать License Addendum File для нескольких продуктов.

После того, как файл License Addendum скачан, он может быть использован в процессе инсталляции OpenEdge. **Внимание!** Не открывайте файл через MS Word, поскольку его формат будет автоматически изменен, в результате чего License Addendum File не будет распознан программой инсталляции.

Инсталляция

Используя учетную запись root подключитесь к серверу, на котором будет выполняться инсталляция. Распакуйте дистрибутив во временный каталог. Ещё раз проверьте, что эта учетная запись использует корректную версию JAVA, а в переменной $PATH прописан правильный путь к ней.

Создайте в каталоге с дистрибутивом временный каталог tmp[4] и перейдите в него. После этого выполните команду <oe-distr-dir>/proinst. Если в $PATH определена правильная версия JAVA, а инсталляция действительно выполняется под учетной записью root, то программа выведет на экран следующее приветственное сообщение:

[4] Название и расположение временного каталога непринципиально, его можно создать в любом месте.

```
+------------------------------------------------------------------+
|                              Welcome                             |
+------------------------------------------------------------------+
|                                                                  |
|          WELCOME TO THE OPENEDGE INSTALLATION UTILITY            |
|                                                                  |
| Ensure that you have your completed "Preinstallation Checklist   |
| for Unix" handy to consult. This information will facilitate your|
| installation and ensure your choices are accurately recorded.    |
|                                                                  |
| Copyright (c) 1984-2009 Progress Software Corporation and/or its |
| subsidiaries or affiliates. All Rights Reserved.                 |
|                                                                  |
|                           [Enter=OK]                             |
+------------------------------------------------------------------+
```

Выберите [Enter=OK] для продолжения процесса инсталляции. Появится окно, предлагающее ввести информацию о лицензиях:

```
+------------------------------------------------------------------------+
|                     Product Configuration Data                         |
+------------------------------------------------------------------------+
|                                                [Enter=Additional]      |
| Company Name: ____________________________     [Ctrl-E=Done]           |
| Serial Number: __________                      [CTRL-T=Quit]           |
| Control Number: _____  _____  _____            [CTRL-N=Release Notes]  |
|                                                [CTRL-V=View]           |
|                                                [TAB=Next Field]        |
|                                                [CTRL-P=Help]           |
|                                                [CTRL-A=Addendum File]  |
|                                                                        |
+------------------------------------------------------------------------+
```

Здесь можно загрузить лицензии как вручную, так и с помощью файла License Addendum. Воспользуемся вторым способом. Нажмите комбинацию клавиш [CTRL-A]. В появившемся диалоговом окне введите полный путь к файлу License Addendum и нажмите клавишу [Enter]. Если лицензионный файл в порядке, то вы получите сообщение: «Control numbers have been confirmed and accepted». Проверить список загруженных лицензий можно, нажав комбинацию клавиш [CTRL-V]. Как только вы убедились, что введенная информация обо всех лицензиях верна, продолжите установку, нажав комбинацию клавиш [Ctrl-E], после чего будет выведено следующее диалоговое сообщение:

```
+------------------------------------------------------------------------+
|                 Done Configuration Data Confirmation                   |
+------------------------------------------------------------------------+
|                                                                        |
|Are you sure that you are done entering all the control numbers for the|
|OpenEdge products that will be installed?                               |
|                                                                        |
|                             [Y=YES] [N=NO]                             |
+------------------------------------------------------------------------+
```

Подтвердите ваше намерение продолжить работу, нажав клавишу [Y=YES]. Следующее диалоговое окно предложит установить OpenEdge Explorer. Так же как и в предыдущем случае нажмите клавишу [Y=YES], согласившись с установкой OpenEdge Explorer[5].

[5] Продукт OpenEdge Explorer позволяет выполнять мониторинг баз данных и системы в целом. В рамках данного курса он будет рассмотрен только кратко.

Далее в появившемся окне укажите путь к каталогу, в котором установлен пакет JAVA, например:

```
+------------------------------------------------------------------------------+
|                            JavaHome Pathname                                 |
+------------------------------------------------------------------------------+
|                                                                              |
| Enter JavaHome Path: /usr/java/jre1.5.0_22/_____________________________     |
|                                                                              |
|                                                                              |
|                                                                              |
|      [Enter=OK]  [CTRL-N=Cancel]  [CTRL-P=Help]                              |
+------------------------------------------------------------------------------+
```

Для продолжения установки нажмите клавишу [Enter=OK], откроется следующее меню:

```
+-------------------------------+
| Install Type and Destination  |
+-------------------------------+
| Select Type of Installation   |
| Select Destination Pathname   |
| Select Management Pathname    |
| Continue with Installation    |
| View Release Notes            |
| Cancel                        |
| Quit Installation             |
| Help                          |
+-------------------------------+
```

По очереди выберите пункты:

- Select Type of Installation → Complete Install
- Select Destination Pathname
- Select Management Pathname

По умолчанию каталог для инсталляции OpenEdge - это /usr/dlc. Для удобства идентификации версии OpenEdge его можно заменить на /usr/dlc102b. По такому же принципу измените путь рабочего каталога с /usr/wrk на /usr/wrk102b. Измените каталоги для установки OE Explorer и OE Management с /usr/oemgmt на /usr/oemgmt102b и с /usr/wrk_oemgmt на /usr/wrk_oemgmt102b. **Внимание!** Предварительно создавать эти каталоги не нужно, это сделает программа установки.

После настройки путей выберите в меню пункт Continue with Installation и нажмите клавишу [Enter]. В случае установки лицензии OpenEdge Application Svr система выведет дополнительные диалоговые сообщения.

```
+------------------------------------------------------------------------------+
|                                  ATTENTION                                   |
+------------------------------------------------------------------------------+
|                                                                              |
|The OpenEdge Adapter for Sonic ESB is a recommended component of this         |
|installation and requires a Sonic ESB installation somewhere on your         |
|network.                                                                      |
|                                                                              |
|Choose YES if you plan on using OpenEdge Adapter for Sonic ESB, else choose   |
|NO.                                                                           |
|                                                                              |
|                           [Y=YES] [N=NO] [H=Help]                            |
+------------------------------------------------------------------------------+
```

Поскольку в рамках этого курса Sonic ESB не рассматривается, то нажмите клавишу [N=NO]. Появится окно с предложением установить WebSpeed, который нам тоже не нужен, поэтому также нажмите клавишу [N=NO].

В открывшемся меню выберите язык, который будет использоваться для OpenEdge по умолчанию, в данном случае выберите English – International. Пункт пометится символом звездочки (*), для продолжения работы выберите пункт Continue with Installation.

В открывшемся меню по очереди выберете пункты:

- Select Character Set, Collation, Case → Russian,Russia,1251,Russian,Basic
- Select a Date Format → ymd
- Select a Number Format → 1,234.56 (comma, period)
- Continue with Installation

```
+-----------------------------------+
| International Settings            |
+-----------------------------------+
| Select CharacterSet,Collation,Case |
| Select a Date Format              |
| Select a Number Format            |
| Continue with Installation        |
| Cancel                            |
| Quit Installation                 |
| Help                              |
+-----------------------------------+
```

Ответы на дополнительные запросы программы инсталляции также носят факультативный характер, поскольку связаны с Web Server Adapter:

```
+-------------------------------------------------------------------------+
|                        Web Services Adapter URL                         |
+-------------------------------------------------------------------------+
|   Please enter the URL of where you will configure the sample           |
|   Web Services Adapter's Java Servlet.                                  |
| URL: http://polygon:80/wsa/wsa1_____________________________________    |
|      [Enter=OK]  [CTRL-N=Cancel]  [CTRL-P=Help]                         |
+-------------------------------------------------------------------------+
```

Выберите [Enter=OK], а в качестве ответа на следующий вопрос выберите [Y=YES]:

```
+-------------------------------------------------------------------------+
|                           WSA Authentication                            |
+-------------------------------------------------------------------------+
|Would you like to Disable the Web Services Adapter's administration user |
|authentication?                                                          |
|                        [Y=YES] [N=NO] [H=Help]                          |
+-------------------------------------------------------------------------+
```

В следующем информационном окне выберите [Y=YES].

```
+------------------------------------------------------------------------------+
|                           Complete Installation                              |
+------------------------------------------------------------------------------+
|                                                                              |
|The following products will be installed:                                     |
|'Перечисление устанавливаемых продуктов Progress'                             |
|'Перечисление устанавливаемых продуктов Progress'                             |
|                                                                              |
|Disk Space Required for Products: 575,672,320 bytes                           |
|Disk Space Required for Installation: 576,790,528 bytes                       |
|Disk Space Remaining After Installation: 214,280,818,688 bytes                |
|                                                                              |
|Selected Destination Path: /usr/dlc102b                                       |
|                                                                              |
|Do you want to install the above listed product(s)?                           |
|                                                                              |
|                          [Y=YES] [N=NO] [H=Help]                             |
+------------------------------------------------------------------------------+
```

На запрос программы установки, связанный с копированием скриптов, рекомендуется ответить [N=NO], т.к. в случае установки нескольких версий OpenEdge скрипты будут перепутаны, и прописывать путь к ним в $PATH лучше явно, например, $DLC/bin:

```
+-----------------------------+
|        Copy Scripts?        |
+-----------------------------+
|                             |
|Copy the scripts to /usr/bin?|
|                             |
|       [Y=YES] [N=NO]        |
+-----------------------------+
```

В информационном окне конфигурации WebSpeed выберите [Enter=OK]:

```
┌──────────────────────────────────────────────────────────────────────────────┐
|                           Configuring WebSpeed                               |
+------------------------------------------------------------------------------+
|                                                                              |
| a. Set up and start your Web server                                          |
|    - If you did not select to "Copy static HTML files to                     |
|      Document Root directory", then manually copy the files                  |
|      or set a link.                                                          |
|    - For NSAPI Messenger, edit the "obj.conf" and "start" files              |
|      on the Web server.                                                      |
| b. Set up the Broker machine.                                                |
|    - Set environment variables if necessary.                                 |
|    - Edit the properties file (ubroker.properties), then start Broker.       |
| c. To validate your configuration through the Messenger                      |
|     Administration Page, enter?WSMAdmin after the Messenger name             |
|    in a URL.                                                                 |
|    (For example, for CGI, http://hostname/cgi-bin/wspd_cgi.sh?WSMAdmin)      |
|    (For example, for NSAPI, http://hostname/wsnsa.dll?WSMADmin)              |
|                                                                              |
|See the "OpenEdge Application Server: Administration" guide for details.      |
|                                                                              |
|                            [Enter=OK] [H=Help]                               |
└──────────────────────────────────────────────────────────────────────────────┘
```

После этого начнется процесс установки. Затраченное на это время зависит от набора устанавливаемых компонентов и мощности сервера.

По окончанию процесса установки OpenEdge завершите её, выбрав в меню пункт End the Progress Installation:

```
+-------------------------------------------------------+
|Installation of selected Progress products is complete.  |
|Refer to the installation notes for more information.    |
+-------------------------------------------------------+
| End the Progress Installation                         |
| View Release Notes                                    |
+-------------------------------------------------------+
```

Настройка переменных среды окружения для работы с СУБД OpenEdge

Перед началом работы с OpenEdge необходимо выполнить настройки некоторых переменных. Основная такая переменная - это DLC, которая должна содержать путь к каталогу, в котором установлен OpenEdge, например:

```
export DLC=/usr/dlc102b
```

После того, как переменной DLC будет присвоено значение, необходимо в переменной PATH задать путь к исполняемым модулям и скриптам OpenEdge, которые находятся в подкаталоге bin установочного каталога OpenEdge, например:

```
export PATH=$DLC/bin:$PATH
```

Теперь можно приступить к полноценной работе с OpenEdge, поскольку по умолчанию значения остальных переменных среды окружения базируются на значении переменной окружения DLC. Следующая таблица содержит список и описание переменных среды окружения, необходимых для работы OpenEdge.

Переменная	Описание	Пример
DLC	Путь к каталогу, в котором установлен OpenEdge.	DLC=/usr/dlc
PATH	Список каталогов, в которых операционная система ищет различные команды, необходимые к выполнению. Эту переменную использует и OpenEdge для поиска команд или программ, которые вы вызываете с помощью операторов INPUT THROUGH и OUTPUT THROUGH. Таким каталогом может быть $DLC/bin - каталог с утилитами и программами OpenEdge	PATH=$DLC/bin:$PATH
PROCFG	Путь к файлу конфигурации OpenEdge. Файл конфигурации - это файл, который содержит информацию об установленных лицензиях продуктов OpenEdge и его компонентах. По умолчанию, если переменная пустая, то используется файл $DLC/progress.cfg. Используйте эту переменную, если необходимо переместить progress.cfg в другой каталог.	PROCFG=$HOME/progress.cfg
PROCONV	Путь к файлу convmap.cp. Это двоичный файл, содержащий все таблицы конвертации между различными языковыми кодовыми таблицами. По умолчанию это $DLC/convmap.cp, если переменная пустая. Воспользуйтесь этой переменной, если необходимо использовать файл convmap.cp, отличный от стандартного файла.	PROCONV=$HOME/convmap.cp

PROEXE	Полный путь к команде _progres. Это команда для работы с приложениями OpenEdge. По умолчанию, если переменная пустая, используется команда $DLC/bin/_progres. Если необходимо перенести в другой каталог или переименовать эту команду, то используйте переменную PROEXE для идентификации нового месторасположения или имени команды.	PROEXE=$DLC/bin/_progres
PROMSGS	Полный путь к файлу, содержащему все возможные сообщения базы данных на необходимом языке. По умолчанию, если переменная пустая, это файл $DLC/promsgs с сообщениями на английском языке. Воспользуйтесь этой переменной, если хотите использовать, например, файл с сообщениями на русском языке.	PROMSGS= $DLC/prolang/rus/promsgs.rus
PROPATH	Список каталогов для работы OpenEdge и ABL-приложений. По умолчанию переменная содержит следующие каталоги: $DLC $DLC/bin $DLC/tty Каталоги разделяются двоеточием (:) по аналогии с $PATH.	PROPATH= $DLC:<dir1>:…:<dirN>
TERM	Тип используемого терминала, например, vt100.	TERM=vt100
PROTERMCAP	Полный путь к файлу описания терминалов, которые должны использоваться OpenEdge. По умолчанию используется файл $DLC/protermcap. Эту переменную стоит устанавливать, если необходимо, чтобы OpenEdge использовал другой файл описания терминалов.	PROTERMCAP= $HOME/protermcap

Более детальную информацию по переменным среды окружения можно получить в стандартной документации: «OpenEdge® Getting Started:Installation and Configuration», в главе «UNIX environment variables».

Для удобства работы с переменными среды окружения, особенно если у вас установлены разные версии OpenEdge, можно создать специальный файл настройки параметров и прописать его в профайле пользователя (`.profile`). Ниже приведен пример файла настроек и часть кода из файла `.profile`.

Пример файла `prosetup`:

```
PROPATH=$DLC:$DLC/bin:$HOME/prosrc; export PROPATH
PATH=$PROPATH:$PATH; export PATH
PROCFG=$DLC/progress.cfg; export PROCFG
PROMSGS=$DLC/promsgs; export PROMSGS
PROTERMCAP=$DLC/protermcap; export PROTERMCAP
```

Примечание! В этом примере намеренно пропущено определение переменной DLC, смотрите следующий пример файла `.profile`:

```
echo "Choose DLC"
echo " 1 - Progress  10.1C "
echo " 2 - Progress  10.2A"
echo " 3 - Progress  10.2B (Default)"
read ii
if [ "$ii" == 1 ]
then
    DLC=/usr/dlc101c  ; export DLC
elif [ "$ii" == 2 ]
then
    DLC=/usr/dlc102a  ; export DLC
elif [ "$ii" == 3 ]
then
    DLC=/usr/dlc102b ; export DLC
else
#по умолчанию 10.2B
    DLC=/usr/dlc102b ; export DLC
fi

. $HOME/prosetup
```

Подобная настройка позволит пользователю сразу после подключения к серверу выбрать необходимую ему версию OpenEdge. На экран будет выведено следующее меню:

```
Choose DLC
1 - Progress  10.1C
2 - Progress  10.2A
3 - Progress  10.2B (Default)
```

После ввода нужной цифры, обозначающей версию OpenEdge, в переменную DLC будет записан путь к установочному каталогу соответствующей версии. После чего сработает файл prosetup, который установит значения оставшимся переменным среды окружения.

Empty-базы данных и их русификация

В каталоге $DLC расположены базы-пустышки, которые содержат метасхему OpenEdge, так называемые empty-базы данных. Empty-баз четыре, каждая из которых предназначена для создания баз данных с различным размером блока:

- $DLC/empty - 1Кб
- $DLC/empty2 - 2Кб
- $DLC/empty4 - 4Кб
- $DLC/empty8 - 8Кб

По умолчанию empty-базы данных имеют кодировку ISO8859-1. Перед использованием их нужно русифицировать. Для этого существует два метода, о которых и пойдет речь далее.

Метод 1. Замена стандартных empty-баз данных на заранее закачанные empty-базы с соответствующей кодировкой.

В каталоге $DLC есть папка prolang, в которой по умолчанию находятся языковые настройки для различных английских кодировок и кодировки UTF. Существует возможность получить настройки так же и для русского языка. Для этого необходимо подключиться к Progress® Download Center и скачать архив Supplemental PROMSGS для соответствующей версии OpenEdge и операционной системы. Найти архив можно там же, где вы скачивали дистрибутив Progress® OpenEdge™.

Распакуйте архив Supplemental PROMSGS в каталог $DLC/prolang. Для этого перейдите в этот каталог и скачайте в него полученный файл, после чего распакуйте следующей командой:

```
tar zxvf 102b_suppromsgs_linux32.tar.Z
```

В каталоге $DLC/prolang появятся несколько каталогов с различными языковыми настройками, в том числе и каталог rus. Средствами операционной системы скопируйте все файлы, начинающиеся на empty*, из каталога $DLC/prolang/rus в каталог $DLC, перезаписав существующие в нём empty-базы.

Полученные empty-базы данных будут иметь кодировку 1251. Убедитесь, что в файле параметров `$DLC/startup.pf` прописана именно эта кодировка, пример:

```
-cpinternal 1251
-cpstream 1251
-cpcoll Russian
```

О назначении файлов параметров будет рассказано позже.

Что делать, если необходимо создать базу данных с другой русскоязычной кодировкой, например, с KOI8-R или IBM866? Для этого воспользуемся вторым методом.

Метод 2. Конвертация

Мы получили empty-базы данных с кодировкой 1251, но нам нужна другая кодировка, возьмем, например, кодировку IBM866.

Создайте в домашнем каталоге подкаталог ibm866[6] и средствами операционной системы скопируйте в него из каталога $DLC все файлы, начинающиеся на empty*. После этого перейдите в каталог ibm866 и для конвертации empty-баз из кодировки 1251 в кодировку IBM866 выполните по порядку следующие действия.

1. Задайте команду

   ```
   proutil empty -C convchar convert IBM866
   ```

   На запрос диалогового сообщения о том, была ли сделана резервная копия конвертируемой базы, ответить [y]. По завершению работы команды на экран будет выведено следующее сообщение:

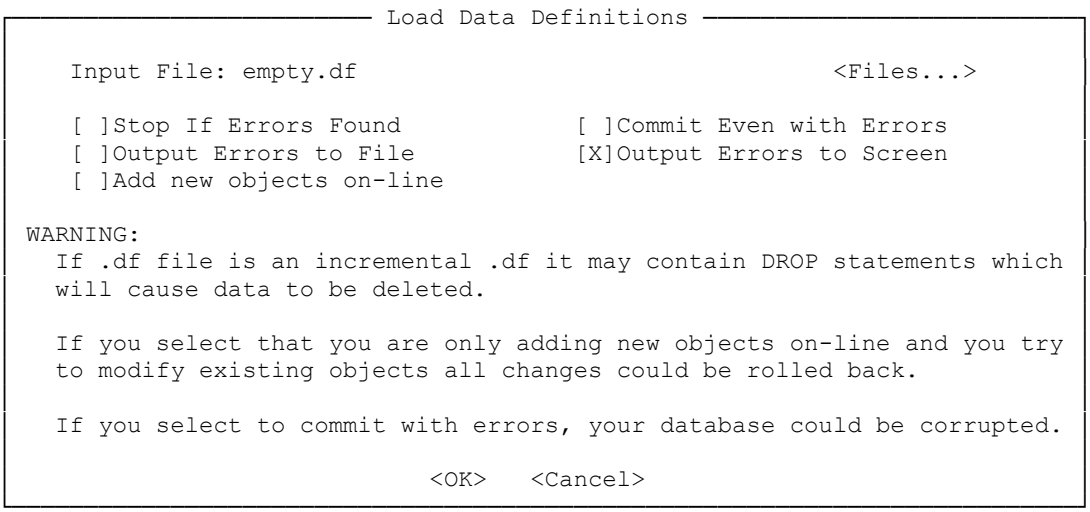
   ```
   Results of scan to convert database. (3944)

   Old Encoding:   1251
   New Encoding:   IBM866
   Scan Mode:      CONVERT/ANALYSE
   Scan Complete:  YES
   Scan Coverage:  FULL
   Database State: CONVERTED

   The following fields contain data requiring translation: (3963)
       Total of 0 fields.
   Database is encoded using IBM866 for character data types. (3939)
   Conversion complete, character encoding conversion successful. (3945)
   ```

2. Теперь необходимо загрузить описание кодировки в саму базу. Для этого войдите в базу в однопользовательском режиме с помощью команды pro:

   ```
   pro empty
   ```

3. В открывшемся окне Procedure Editor нажмите клавишу <F3>, активируется верхнее меню. Выберите пункт меню Tools → Data Dictionary → Admin → Load Data and Definitions → Data Definitions. Откроется диалоговое окно Load Data Definitions.

[6] **Внимание!** Не допускается работать с базами данных, расположенными в каталоге $DLC.



С помощью кнопки <Files...> выберите файл `rus866.df` из каталога $DLC/prolang/rus или введите полный путь к нему вручную в поле Input File. Нажмите клавишу <F1> для выполнения загрузки `rus866.df` в базу данных. После загрузки будет выдано предупреждение о необходимости выполнить полную переиндексацию базы данных. Нажмите клавишу [Enter] и выйдите из базы: Database → Exit → <F3> → File → Exit.

4. Выполните полную переиндексацию базы следующей командой:

```
proutil empty -C idxbuild all
```

Внимание! Параметров для переиндексации в этой команде достаточно для empty-баз данных, поскольку эти базы-«пустышки». Если вы будете русифицировать полноценные промышленные базы данных, то потребуются дополнительные параметры для утилиты PROUTIL IDXBUILD.

5. Активируйте word-indexes для корректной работы с русскими символами. Для этого необходимо создать word-правила (word-rules) и применить их к базе данных. В каталоге $DLC/prolang/convmap есть файл `m1251bas.wbt` - это Word Break Source File. Но мы воспользуемся расширенным файлом, который находится на диске и называется `1251-rus.wbt`. Скопируйте его в каталог $DLC.
6. Создайте правило №1, для этого перейдите в каталог $DLC и откомпилируйте Word Break Source File следующей командой:

```
proutil -C wbreak-compiler 1251-rus.wbt 1
```

В каталоге $DLC появится файл `proword.1`.

7. Примените файл `proword.1` к базе данных empty:

```
proutil empty -C word-rules 1
```

8. Если вместо empty-базы данных использовалась существующая база с данными, то после применения новых word-правил ей понадобится переиндексация word-индексов. Для переиндексации только word-индексов воспользуйтесь ABL-

программой fnd-wrdidx.p, находящейся на диске, которая создаст правила для утилиты PROUTIL IDXBUILD и поместит эти правила в файл `word.idx`.

Для выполнения ABL-программы выполните следующую команду:

```
pro <db-name> -p ./fnd-wrdidx.p
```

Примечание! Для empty-баз данных выполнять эту процедуру бессмысленно, поскольку эти базы не содержат пользовательской схемы данных, следовательно, в них нет word-индексов.

Как только создан файл `word.idx` (он появится в рабочем каталоге, из которого была выполнена программа fnd-wrdidx.p), используйте его для переиндексации следующей командой:

```
proutil <db-name> -C idxbuild -TB 31 -TM 32 -B 512 < word.idx
```

9. Повторите пункты с первого по седьмой для оставшихся empty-баз данных (empty1, empty2, empty4, empty8).
10. В файле параметров `$DLC/startup.pf` замените название кодировки 1251 на ibm866. Пример:

```
-cpinternal ibm866
-cpstream ibm866
-cpcoll Russian
```

Настройка прав доступа

Для обеспечения безопасности базы данных на уровне операционной системы, а так же для исключения проблем с пользовательским доступом к базе данных необходимо задать определенные права на утилиты OpenEdge и файлы самой базы данных. Установка этих прав доступа позволит также обезопасить файлы базы данных от удаления пользователями, имеющими доступ к командной строке (shell) операционной системы.

Исполняемые программы OpenEdge.

Исполняемые программы OpenEdge – это программы, расположенные в каталоге $DLC/bin. Владельцем этих программ должен быть пользователь root. При этом для него необходимо установить права на чтение, запись и установить setuid-бит. Права для группы и прочих пользователей должны быть установлены только на чтение и выполнение. Для этого:

1. Войдите в операционную систему под пользователем root
2. Перейдите в каталоге $DLC/bin
3. Выполните следующий набор команд:

```
chown root _*
chmod 4755 _*
chmod 755 _sqlsrv2
chmod 755 _waitfor
```

Пример того, что должно получиться:

```
-rwsr-xr-x 1 root root 9273163 Dec 15  2009 _progres
-rwsr-xr-x 1 root root   11749 Nov  2  2009 _proutil
-rwsr-xr-x 1 root root   12076 Nov  2  2009 _rfutil
```

Бит setuid может быть установлен только для владельца или группы файла. Это позволит пользователю, который выполняет команду, выполнять её от имени владельца файла, что в свою очередь предоставит индивидуальный пользовательский доступ к файлам базы данных через OpenEdge, в то время как у пользователя не будет доступа на удаление файлов базы данных из командной строки оболочки Unix.

Файлы базы данных OpenEdge.

Для защиты файлов базы данных от несанкционированного удаления пользователями, необходимо установить на них права.

- Права на каталог, в котором размещена база данных, должны принадлежать пользователю root, с возможностью чтения, записи и выполнения. Группа каталога должна быть установлена, равной группе пользователей базы данных, а права должны быть установлены на чтение и выполнение. У прочих пользователей не должно быть никаких прав. Пример:

```
drwxr-x---   5 root bankier    4096 Jul  1 19:00 sports
```

- Права доступа к файлам базы данных должны быть установлены только на чтение и запись. Владельцем файлов должен быть пользователь root, а группа файлов должна быть равна группе пользователей базы данных. Пример:

```
-rw-rw---- 1 root bankier 2228224 Jul  1 18:51 sports/sports.b1
-rw-rw---- 1 root bankier 1769472 Jul  1 19:00 sports/sports.d1
-rw-rw---- 1 root bankier   32768 Jul  1 18:23 sports/sports.db
-rw-rw---- 1 root bankier  665850 Jul  1 19:00 sports/sports.lg
```

С таким набором прав доступа только пользователь root и все пользователи, которые являются частью группы, могут обращаться к базе данных через OpenEdge, однако только пользователь root, имеющий права доступа на каталог, может изменять или удалять файлы из командной строки оболочки Unix.

Тестовая база Sports

В каталоге $DLC расположены тестовые базы данных с именами Sports и Sports2000. Эти базы специально предназначены для изучения OpenEdge, они также могут использоваться для воспроизведения ошибочных ситуаций, которые могут произойти в промышленной базе данных, с целью изучения природы ошибки и поиска способов её устранения. Часто при обращении в Progress Tech Support у вас могут попросить воспроизвести проблему на одной из этих баз.

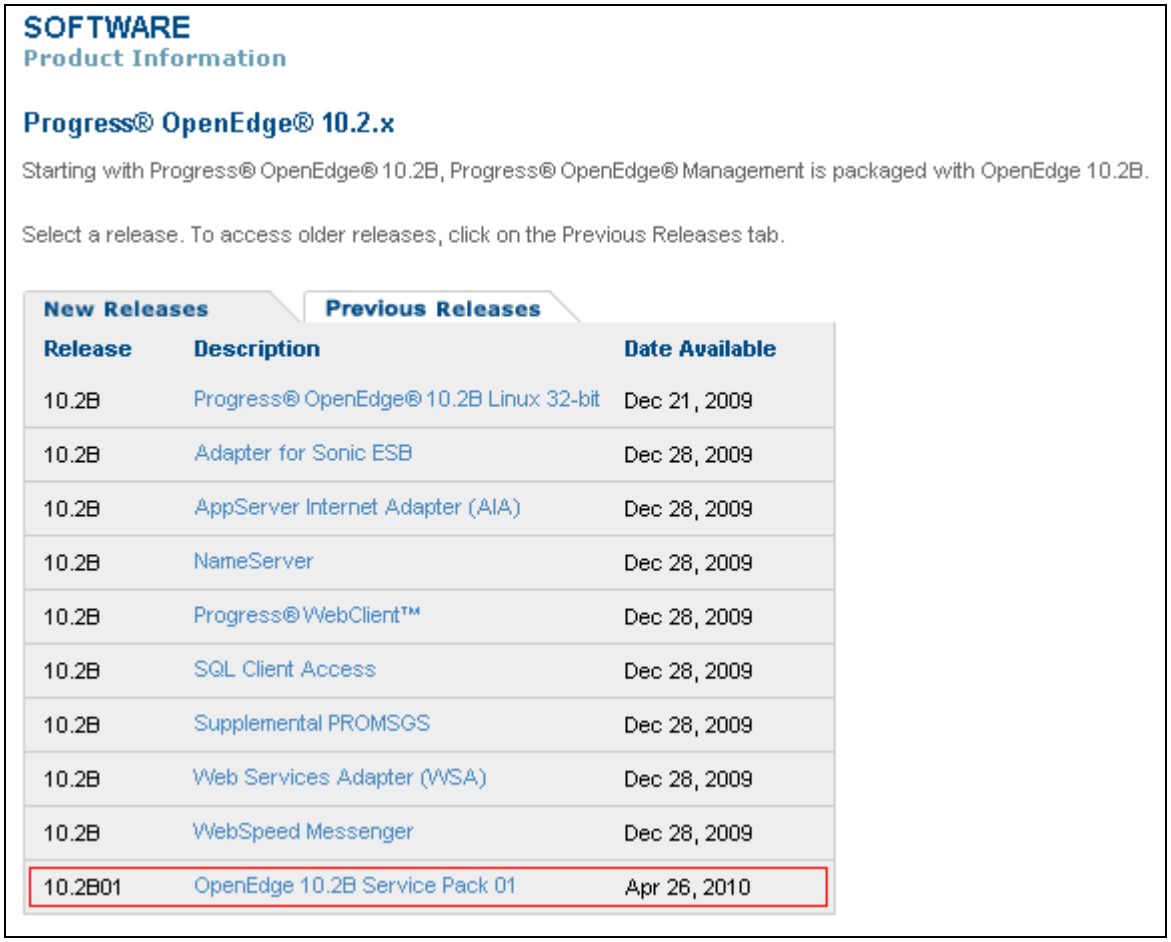
В рамках курса «Основы администрирования СУБД OpenEdge 10» в качестве тестовой базы данных во всех примерах и практических занятиях будет использоваться база данных Sports2000.

Глава 3 Обновление OpenEdge

Установка Service Pack

Service Pack - кумулятивный пакет обновлений для конкретной версии OpenEdge. Это означает, что пакет обновлений № 2 может быть установлен без установки пакета обновлений № 1, поскольку все изменения, вошедшие в первый пакет, будут содержаться во втором.

Пакеты обновлений выпускаются с интервалом от трех до шести месяцев. Все пакеты обновлений доступны для скачивания с Progress® Download Center на странице Software Product Information. Пример страницы:



SOFTWARE
Product Information

Progress® OpenEdge® 10.2.x

Starting with Progress® OpenEdge® 10.2B, Progress® OpenEdge® Management is packaged with OpenEdge 10.2B.

Select a release. To access older releases, click on the Previous Releases tab.

New Releases | Previous Releases

Release	Description	Date Available
10.2B	Progress® OpenEdge® 10.2B Linux 32-bit	Dec 21, 2009
10.2B	Adapter for Sonic ESB	Dec 28, 2009
10.2B	AppServer Internet Adapter (AIA)	Dec 28, 2009
10.2B	NameServer	Dec 28, 2009
10.2B	Progress® WebClient™	Dec 28, 2009
10.2B	SQL Client Access	Dec 28, 2009
10.2B	Supplemental PROMSGS	Dec 28, 2009
10.2B	Web Services Adapter (WSA)	Dec 28, 2009
10.2B	WebSpeed Messenger	Dec 28, 2009
10.2B01	OpenEdge 10.2B Service Pack 01	Apr 26, 2010

Щелкнув по строке названия пакета обновлений (в нашем примере - OpenEdge 10.2B Service Pack 01) вы попадаете на страницу Software Product Download, на которой можно скачать архив пакета обновлений, соответствующий вашей версии операционной системы.

После скачивания пакета обновлений необходимо распаковать его во временный каталог. Для установки пакета в Windows нужно выполнить программу setup.exe, а в Unix-команду proinst.

Внимание! Перед началом установки пакета убедитесь, что все Progress-процессы[7] остановлены. В противном случае установка Service Pack выполнена не будет.

Процесс установки OpenEdge 10.2B Service Pack 01 для Unix.

Распакуйте архив пакета обновлений в отдельный каталог:

```
tar vfx OE102B01_linux32.tar
```

Создайте в этом каталоге временный каталог tmp и перейдите в него. **Внимание!** Если выполнить команду `proinst` из каталога, в котором находится дистрибутив Service Pack, то выводится следующее предупреждение и установка прерывается:

```
A version of progress.cfg has been detected in your current
working directory. You may not run this installation from a
directory containing a Progress installation. Change to another
directory and type <sp-dir>/proinst to start the installation.
```

Выполните команду `proinst` из временного каталога tmp. Программа инсталляции выведет на экран следующее приветственное сообщение:

```
+------------------------------------------------------------+
|                          Welcome                           |
+------------------------------------------------------------+
|      WELCOME TO THE OPENEDGE INSTALLATION UTILITY FOR      |
|                                                            |
|                       SERVICE PACK 1                       |
|                                                            |
| Copyright (c) 1984-2010 Progress Software Corporation      |
| and/or its subsidiaries or affiliates. All Rights Reserved.|
|                                                            |
|                         [Enter=OK]                         |
+------------------------------------------------------------+
```

Нажмите [Enter=OK] для продолжения процесса инсталляции. Появится окно, содержащее предложение ввести полный путь к каталогу с OpenEdge:

```
+-----------------------------------------------------------------------+
|         Please enter the full path to the OpenEdge installation.      |
+-----------------------------------------------------------------------+
| Enter Path: /usr/dlc102b_____________________________________________ |
|                                                                       |
|      [Enter=OK]  [CTRL-N=Cancel]                                      |
+-----------------------------------------------------------------------+
```

Введите путь и нажмите [Enter=OK]. Появится окно с предложением сделать резервную копию изменяемых файлов OpenEdge:

```
+------------------------------------------------------------+
|               Question: Backup Existing Files?             |
+------------------------------------------------------------+
| Would you like to backup existing OpenEdge 10.2B files prior to |
| installing Service Pack 1?                                 |
| Select YES to create a backup of current OpenEdge 10.2B files   |
| that will be updated by installing Service Pack 1.         |
|                                                            |
|                        [Y=YES] [N=NO]                      |
+------------------------------------------------------------+
```

[7] Под Progress-процессами здесь подразумевают запущенные базы данных, АПП-сервера, пользовательские сессии OpenEdge, и т.п.

Внимание! Рекомендуется всегда делать резервные копии перед началом выполнения любых изменений.

Для создания резервной копии выберите [Y=YES]. Откроется диалоговое окно с предложением ввести путь к каталогу, в который будет сохранена резервная копия. По умолчанию предлагается каталог SPBackup в установочном каталоге OpenEdge:

```
+------------------------------------------------------------------------+
|             Please enter the full path to a backup directory.          |
+------------------------------------------------------------------------+
| Enter Path: /usr/dlc/SPBackup_____________________________________     |
|                                                                        |
|      [Enter=OK]  [CTRL-N=Cancel]                                       |
+------------------------------------------------------------------------+
```

Каталог SPBackup можно использовать для восстановления OpenEdge в состояние до установки Service Pack. Рекомендуется оставить путь по умолчанию. Если путь к резервному каталогу устраивает, нажмите клавишу [Enter]. Перед началом процесса установки на экран будет выведено сообщение с итоговой информацией об установке:

```
+-------------------------------------------+
|         ServicePack Install Summary       |
+-------------------------------------------+
| OpenEdge version currently installed: 10.2B |
| Service Pack version to be installed: 1   |
|                                           |
| Service Pack Destination Path:            |
|  /dsk3/bank/users/valeriy/dlc             |
|                                           |
| Backup Destination Path:                  |
|  /dsk3/bank/users/valeriy/dlc/SPBackup    |
|                                           |
| Log file:                                 |
|  /dsk3/bank/users/valeriy/dlc/SP1-log.txt |
|                 [Enter=OK]                |
+-------------------------------------------+
```

Нажмите [Enter=OK], сначала будет создана резервная копия всех изменяемых файлов, а затем начнется процесс установки. По завершению установки на экран будет выведено следующее сообщение:

```
+-------------------------------------------------------+
| Installation of OpenEdge Service Pack 1 is complete.  |
| Refer to the release notes for more information.      |
+-------------------------------------------------------+
| End the OpenEdge Service Pack Installation            |
| View Release Notes                                    |
| View Log File                                         |
| Help                                                  |
+-------------------------------------------------------+
```

Выберите пункт меню End the OpenEdge Service Pack Installation и нажмите клавишу <Enter>. Установка Service Pack 01 для OpenEdge 10.2B завершена.

Добавление новых продуктов

Когда приобретается новый продукт, например, OpenEdge Transparent Data Encryption, его необходимо добавить к существующей конфигурации OpenEdge. Процесс добавления ничем не отличается от первичной инсталляции, за исключением того, что в качестве установочного каталога нужно указать каталог, в котором установлена текущая

конфигурация OpenEdge. Рассмотрим добавление нового продукта на примере установки OpenEdge Transparent Data Encryption.

Все продукты OpenEdge находятся в одном общем дистрибутиве, их установка определяется вводом соответствующей лицензионной информации, воспользуемся дистрибутивом OpenEdge 10.2B. Перед началом установки убедитесь, что в переменной $PATH прописана версия JAVA 5.0.

Перейдите во временный каталог, который мы создали в процессе начальной инсталляции в каталоге с дистрибутивом, тогда мы назвали его tmp. Если каталога не существует, создайте его сейчас.

Для начала процесса установки выполните команду `<oe-distr-dir>/proinst` из этого каталога. Пропустите приветственное сообщение, нажав клавишу [Enter]. Откроется диалоговое окно Product Configuration Data, введите название организации, затем серийный и контрольный номер лицензии. С помощью комбинации клавиш <Ctrl-V> убедитесь, что лицензия введена правильно, откроется окно Entered Product List:

```
+----------------------+
|Entered Product List  |
+----------------------+
| OpenEdge TDE         |
+----------------------+
```

Продолжите установку, нажав комбинацию клавиш <Ctrl-E>. В открывшемся окне Install Type and Destination выберите пункт меню Select Destination Pathname и укажите полный путь к каталогу с текущей конфигурацией OpenEdge. Как только будет введен путь и нажата клавиша [Enter], выведется диалоговое сообщение, предупреждающее о том, что выбран каталог, в котором уже установлен OpenEdge. В меню выберите пункт Install the OpenEdge products in the pre-existing destination path:

```
+-------------------------------------------------------------------+
|                    Destination Pathname Exists                    |
+-------------------------------------------------------------------+
| Select an alternate destination path                              |
| Erase the current destination path                                |
| Install the OpenEdge products in the pre-existing destination path |
+-------------------------------------------------------------------+
```

Откроется предупреждающее диалоговое окно, в котором выберите [Y=YES]:

```
+-------------------------------------------------------------------+
|                   Release Compatibility Caution                   |
+-------------------------------------------------------------------+
|CAUTION: The Installation Utility has detected that you will modify a|
|previous installation.  Check the installation notes for release     |
|compatibility.                                                       |
|                                                                     |
|Do you want to continue?                                             |
|                           [Y=YES] [N=NO]                            |
+-------------------------------------------------------------------+
```

После указания Destination pathname и Working Dir pathname выберите пункт меню Continue with Installation. В открывшемся диалоговом окне, содержащем информацию об устанавливаемом продукте, для продолжения установки выберите [Y=YES]. На вопрос, связанный с копированием скриптов, выберите [N=NO]:

```
+----------------------------+
|        Copy Scripts?       |
+----------------------------+
|Copy the scripts to /usr/bin?|
|                            |
|   [Y=YES] [N=NO] [H=Help]  |
+----------------------------+
```

Начнется процесс установки. По его завершению откроется следующее диалоговое окно:

```
+-----------------------------------------------------+
|Installation of selected OpenEdge products is complete.  |
|Refer to the installation notes for more information.    |
+-----------------------------------------------------+
| End the OpenEdge Installation                       |
| View Release Notes                                  |
| Help                                                |
+-----------------------------------------------------+
```

Выберите пункт меню End the OpenEdge Installation для завершения установки OpenEdge Transparent Data Encryption.

Внимание! Если установка дополнительного продукта выполнялась после установки Service Pack, то рекомендуется повторно установить все имеющиеся Service Pack. Это связано с тем, что установка нового продукта делалась из основного дистрибутива, который не содержит возможных изменений или исправлений ошибок, содержащихся в Service Pack, в том числе и для устанавливаемого нового продукта.

Обновление лицензий

В OpenEdge 10.2B внесено небольшое изменение в процесс обновления лицензий. Теперь для одного и того же продукта, при условии введения правильных контрольных чисел[8] можно вводить разные серийные номера[9]. В предшествующих версиях программа обновления лицензий не принимала различных серийных номеров. В этом случае было необходимо деинсталлировать существующую лицензию и только затем устанавливать новую лицензию продукта. В этом плане новый процесс обновления лицензий особенно удобен, когда необходимо перейти с оценочной временной лицензии на приобретенную постоянную лицензию. Помимо этого обновление лицензии может понадобиться при изменении количества лицензированных пользователей.

Посмотреть список установленных лицензий OpenEdge можно, выполнив команду `showcfg`. Пример результата её работы:

```
Product Name:  OE Workgroup RDBMS
Installation Date:  Mon Jul  5 18:44:14 2010
User Limit:  10
Expiration Date:  None
Serial Number:  000000000
Control Numbers:  XXXXX - XXXXX - XXXXX
Version Number:  10.2B
Machine Class:  KB
Port Number:  38
```

[8] Контрольные числа - Control Number, формат XXXXX-XXXXX-XXXX. Контрольные числа должны соответствовать серийному номеру, только тогда лицензия считается корректной.
[9] Серийный номер - Serial Number, формат XXXXXXXXX.

Все лицензии OpenEdge хранятся в файле `progress.cfg`, который по умолчанию находится в каталоге $DLC.

Процесс обновления лицензии:

1. Перейдите в каталог $DLC.
2. Введите в командной строке команду `proupdt` и нажмите клавишу [Enter].
3. В открывшемся приветственном окне нажмите клавишу [Enter], откроется диалоговое окно Product Configuration Data:

```
+------------------------------------------------------------------------+
|                    Product Configuration Data                          |
+------------------------------------------------------------------------+
|                                                [Enter=Additional]      |
| Company Name: ____________________________     [Ctrl-E=Done]           |
| Serial Number: _________                       [CTRL-T=Quit]           |
| Control Number: _____  _____  _____            [CTRL-N=Release Notes]  |
|                                                [CTRL-V=View]           |
|                                                [TAB=Next Field]        |
|                                                [CTRL-P=Help]           |
+------------------------------------------------------------------------+
```

4. Введите имя своей компании, серийный номер и контрольные числа новой лицензии, нажмите клавишу [Enter], откроется информационное окно Modified Product Information:

```
+-------------------------------------+
|    Modified Product Information     |
+-------------------------------------+
|  Product Name:   OE Workgroup RDBMS |
|                                     |
|  Old User Count:        5           |
|  New User Count:       10           |
|  Old Expiration Date:  None         |
|  New Expiration Date:  None         |
|                                     |
|             [Enter=OK]              |
+-------------------------------------+
```

   Нажмите клавишу [Enter] для возврата в диалоговое окно Product Configuration Data.

5. Нажмите комбинацию клавиш <Ctrl-E> для завершения процесса изменения лицензии и возврата в командную строку.

Установка новой версии

Обновление OpenEdge в пределах версии 10 - достаточно простая операция. По сути необходимо установить новую версию в отдельный каталог[10] и перенастроить переменные окружения DLC и PATH. Такое разделение версий по каталогам позволит в случае необходимости без труда создавать и работать с базами данных под нужной версией OpenEdge в любое время. Помимо установки новой версии необходимо выполнить определенные действия и с самой базой данных.

[10] Во время лекции по инсталляции я рекомендовал добавлять к предлагаемому по умолчанию пути номер версии, т.е. /usr/dlc102b вместо /usr/dlc

Процесс обновления версии OpenEdge:

1. Остановите базу данных.
2. Выполните усечение Before-Image файла (.bi) командой PROUTIL TRUNCATE BI[11]:

```
proutil <db-name> -C truncate bi
```

3. Сформируйте полную резервную копию базы данных[12:]

```
probkup <db-name> <backup-name> [parameters]
```

4. Выполните установку новой версии OpenEdge в отдельный каталог согласно всем принципам инсталляции OpenEdge[13].
5. Если в предыдущей версии использовались АПП-сервера, WebSpeed и т.п., то скопируйте их настройки, находящиеся в каталоге <OLD-DLC>/properties в файле `ubroker.properties`, в тот же каталог новой версии OpenEdge, т.е. <NEW-DLC>/properties.
6. Замените значения переменных окружения DLC и PATH на путь к каталогу с новой версией.
7. Запустите базу данных под новой версией OpenEdge.

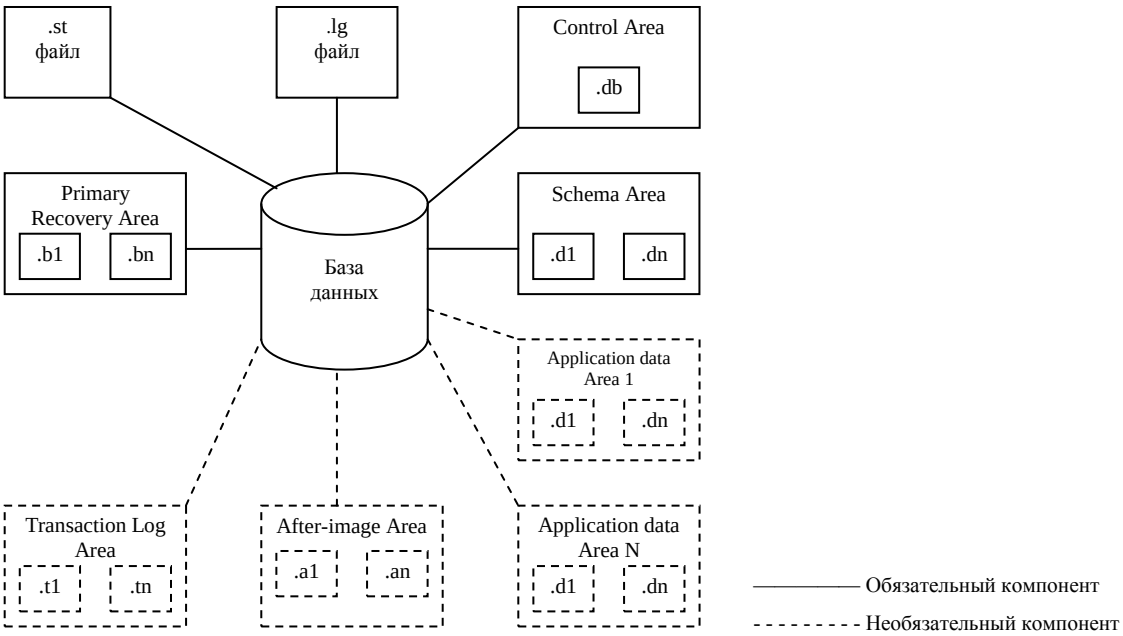
Иногда новая версия OpenEdge предполагает изменение состава виртуальных системных таблиц[14], как это произошло с версией OpenEdge 10.2B. В этом случае с существующей базой данных необходимо выполнить дополнительное действие по обновлению этих таблиц. Делается это командой PROUTIL UPDATEVST[15].

[11] Более детально о команде PROUTIL TRANCATE BI будет рассказано в отдельной лекции.
[12] Процессу формирования резервной копии посвящена отдельная лекция.
[13] См. лекцию «Инсталляция OpenEdge».
[14] Виртуальные системные таблицы, от англ. Virtual System Tables (VST).
[15] Детальное описание утилиты PROUTIL UPDATEVST можно найти в стандартной документации к СУБД OpenEdge.

Глава 4 Архитектура OpenEdge

Файловая структура

База данных OpenEdge состоит из множества различных файлов, которые принадлежат отдельным ее компонентам. На рисунке приведена схема файловой структуры, а назначение каждого ее компонента описано далее.



Файл описания структуры содержит информацию о структуре базы данных. Это текстовый файл, который имеет расширение .st. Информацию, которую он содержит, использует утилита PROSTRCT CREATE, создающая области хранения и их экстенты. За создание .st файла отвечает администратор базы данных. Если при создании базы данных такого файла не будет, то СУБД автоматически создаст его с минимальным набором компонентов.

Журнал событий базы данных (далее лог-файл) является текстовым файлом с расширением .lg. Файл содержит информацию о значимых для базы данных событиях, включая запуск и останов сервера, подключение и отключение пользователей, и прочую информацию о событиях, происходящих в базе.

Контрольная область базы данных (Control Area) представляет собой один файл с расширением .db, который содержит информацию о структуре экстентов базы данных. Контрольная область - это фактически таблица оглавления, используемая базой данных для идентификации имени и пути размещения каждого экстента областей хранения данных.

Область Primary Recovery или **Before Image** (далее BI) может состоять из одного или более экстентов, у которых файлы имеют расширение .b*n*. Файл BI содержит заметки обо всех изменениях в данных. В случае возникновения какого-либо сбоя база данных использует эти заметки для отката незавершенных транзакций, тем самым обеспечивая целостность базы данных.

Область Schema Area содержит минимум один экстент переменного размера, файл которого имеет расширение .d*n*. Область содержит мастер-блок и блоки секвенций, а так же системные таблицы. По умолчанию все пользовательские данные будут храниться в этой области, что по определенным причинам затрудняет управление этими данными.

Область поддерживает только архитектуру SAT-I[16]. Поэтому рекомендуется размещать пользовательские данные в отдельных дополнительных областях хранения.

Дополнительные пользовательские области хранения данных, каждая дополнительная область может содержать от одного до нескольких экстентов, файлы этих экстентов имеют расширение .d*n*. При этом только последний экстент области может быть переменного размера. Эти области предназначены для хранения пользовательских данных, таких как таблицы, индексы, а также поля BLOB или CLOB. Область может иметь тип архитектуры как SAT-I, так и SAT-II[17]. Создавая такие области можно улучшать производительность за счет размещения различных объектов на различных дисках, а также за счет возможности использования архитектуры SAT-II.

Область After-Image Area (далее AI), экстенты этой области имеют расширение .a*n*, и могут быть как фиксированного, так и переменного размера. Экстенты этой области необходимы для работы механизма After-imaging. Благодаря этой области, в случае возникновения системного сбоя, можно восстановить базу данных на любой момент времени. Фактически каждый AI-экстент - отдельная область, которая имеет свой уникальный номер в пределах базы данных.

Область Transaction Log используется, если активирован механизм Two-Phase Commit. Она может состоять из одного и более экстентов фиксированного размера, которые имеют расширение .t*n*. Этой областью не поддерживаются экстенты переменного размера. Её файлы содержат информацию о завершенных транзакциях механизма Two-Phase commit.
Все перечисленные выше файлы - это и есть База данных. Все они должны рассматриваться как единое неделимое целое. Например, фраза «создать резервную копию базы данных» означает, что необходимо сформировать резервную копию всех файлов базы.

Помимо указанного стандартного набора файлов базой данных могут использоваться и другие файлы с различными расширениями. Одни из них создаются лишь на короткий период времени, другие во время активации дополнительных механизмов СУБД и используются в течение всего периода работы этих механизмов, а некоторые создаются администратором, например, настройки сервера репликации. В таблице приведен список возможных дополнительных типов файлов и их краткое описание.

Файл	Описание
.abd	**Архивный бинарный дамп данных аудита**
.bd	**Файл бинарного дампа данных (таблиц)**
.bld	**Файл данных для полей BLOB или CLOB**
.cf	**Файл кэша схемы (Schema Cache)**
.cp	**Файл, содержащий двоичный код скомпилированной кодовой страницы**
.dfsql	**Описание данных в SQL-формате**
.dsql	**Дамп данных в SQL-формате**
.d	**Дамп таблицы в ABL-формате**
.df	**Описание данных в ABL-формате**
.fd	**Файл описания для ABL Bulkload**
.lic	**Файл лицензии**
.lk	**Файл блокировки**
.repl.properties	**Файл настроек OpenEdge Replication**
.repl.recovery	**Файл OpenEdge Replication, содержащий системную информацию по процессу репликации**
.rpt	**Файл отчета об использовании лицензий**

[16] SAT-I, Storage Area Type I
[17] SAT-II, Storage Area Type II

Размер блока базы данных

Блок - самый маленький компонент базы данных. В базе существует множество различных типов блоков. Основные действия с ними выполняются автоматически в фоновом режиме. Однако для оптимального создания и настройки баз данных полезно знать, как эти блоки хранятся. Большая часть блоков может быть разделена на три группы:

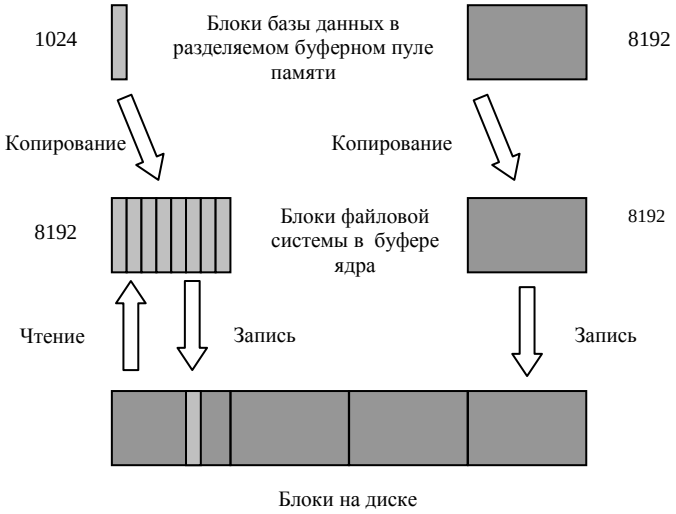
- блоки данных (data blocks);
- индексные блоки (index blocks);
- прочие типы блоков.

Размер блока базы данных (в байтах) может иметь размер:

- 1024 (1 Кб);
- 2048 (2 Кб);
- 4096 (4 Кб);
- 8192 (8 Кб).

По умолчанию размер блока устанавливается равным 4Кб для Windows/Linux и 8Кб для Unix. Размер блока распространяется на всю базу данных.

Обычно для обеспечения наилучшей производительности размер блока базы данных должен соответствовать или быть кратным размеру блока файловой системы. Такое соответствие позволит значительно уменьшить количество операций ввода/вывода (I/O). Пример выполнения записи с различными размерами блоков базы данных продемонстрирован на рисунке.



Операционная система Windows предполагает, что файлы и память разбиты на блоки по 4 Кб. Это означает, что все перемещения между диском и памятью будут осуществляться по 4 Кб. Операционная система Windows оптимизирована под использования 4 Кб блоков, и ее производительность уменьшится, если выбрать размер блока 8 Кб.

1. В Unix-подобных операционных системах размер блока равен или кратен 8Кб, поэтому размер блока можно настраивать. Но обычно значение 8Кб - лучшее решение для таких систем.
2. Сопоставляя размер блока базы данных с блоком операционной системы, вы обеспечиваете более эффективное перемещение данных между ними.

Блоки данных - основные и наиболее распространенные блоки в базе. Существует два типа этих блоков - RM-блоки и блоки RM-цепочки (RM chain blocks). Единственное различие между ними заключается в том, что RM-блоки считаются полностью заполненными, а блоки RM-цепочки заполнены частично. Их внутренняя структура абсолютно одинаковая. Оба типа блоков являются общими. Общие блоки (Social blocks) могут содержать записи из различных таблиц. Другими словами RM-блоки предназначены для сохранения из множества таблиц информации, которая может быть размещена в одном блоке. В противоположность этому индексные блоки содержат только индексные данные для одного индекса только одной таблицы.

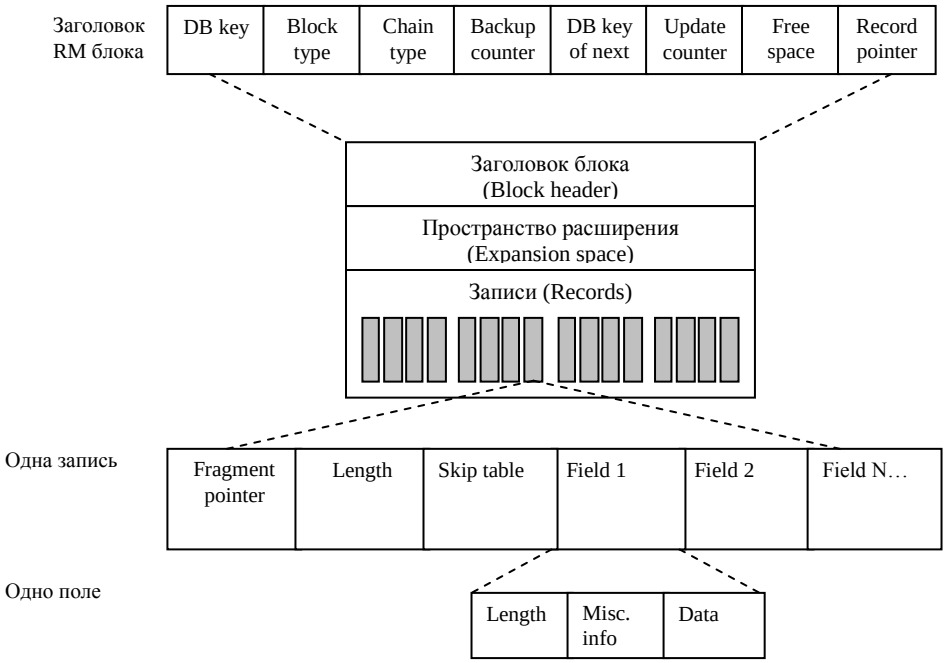
Количество записей, которые можно сохранить в одном блоке, это значение, настраиваемое в пределах области хранения. О том, как его рассчитать, будет рассказано далее.

Каждый RM-блок содержит четыре типа различной информации.

- Заголовок блока (Block header)
- Записи (Records)
- Поля (Fields)
- Свободное пространство (Free space)

Заголовок блока содержит: адрес блока (dbkey); тип блока; тип цепочки (chain type); счетчик резервного копирования (backup counter); адрес следующего за ним блока; счетчик изменений (используется при изменении схемы); указатели свободного пространства (free space pointers) и указатели на записи (record pointers). Для областей хранения SAT-I размер заголовка блока равен 16 байт. А для областей SAT-II размер заголовка переменный, точнее - заголовки первого и последнего блоков в кластере равны 80 байт, а все остальные заголовки внутри кластера имеют размер 64 байта.

Каждая запись содержит указатель фрагмента (fragment pointer), который используется указателями записи в индивидуальных полях, таких как поле размера записи (Length of Record) и поле Skip Table (используется для увеличения поля с целью улучшения поиска). Каждой записи необходимо минимум 15 байт памяти для хранения. Запись содержит поля Length, Miscellaneous Information и сами данные. На рисунке приведен формат RM-блока.



Индексные блоки содержат ту же заголовочную информацию, что и блоки данных, у них те же требования к размеру, т.е. 16 байт для SAT-I и 64 или 80 байт для областей хранения SAT-II. Индексные блоки могут хранить столько информации, сколько в них может максимально поместиться, для большей эффективности эта информация сжимается. Как говорилось ранее, индексный блок может содержать информацию, связанную только с одним конкретным индексом.

Индексы используются для того, чтобы быстро находить записи в базе данных. Каждый индекс в OpenEdge - это структурированное B-дерево, которое всегда находится в сжатом состоянии. Это уменьшает необходимость сравнения ключей, что приводит к улучшению производительности. База данных может содержать до 32 767 индексов. Каждое B-дерево начинается от корня (root), который хранится в виде записи в системной таблице _StorageObject. Для обеспечения большей эффективности индексы являются многопоточными, благодаря чему к ним предоставляется параллельный доступ. Ускорение происходит за счет того, что при доступе к B-дереву блокируется не всё дерево, а только те узлы (их еще называют «листья» или «nodes»), которые требуются в текущий момент.

Существует несколько типов блоков, на которых стоит остановиться для обеспечения наилучшего понимания базы данных:

- мастер-блок (Master block);
- блоки хранения объектов (Storage object blocks);
- свободные блоки (Free blocks);
- пустые блоки (Empty blocks).

Мастер-блок содержит те же 16 байт заголовочной информации, что и другие блоки, но этот блок используется для хранения данных о статусе всей базы данных. Он всегда является самым первым блоком в базе и размещается в области с номером 6, т.е. в Schema Area, которая всегда имеет тип SAT-I. Он содержит такие данные как номер версии базы данных; общее количество размещенных блоков; временную метку и статусные флаги. Для извлечения дополнительной информации из мастер-блока можно воспользоваться виртуальной системной таблицей (VST) _MstrBlk.

Блоки хранения объектов (Storage object blocks) содержат адреса первой и последней записи для каждого индекса таблицы, поэтому, когда пользователь запускает программу, которая запрашивает первую или последнюю запись в таблице, то нет необходимости использовать индексы. Вместо этого движок базы данных получает информацию из блока хранения объектов и непосредственно переходит к записи. Блоки хранения объектов используются очень часто, поэтому они всегда находятся в памяти, что улучшает эффективность запросов.

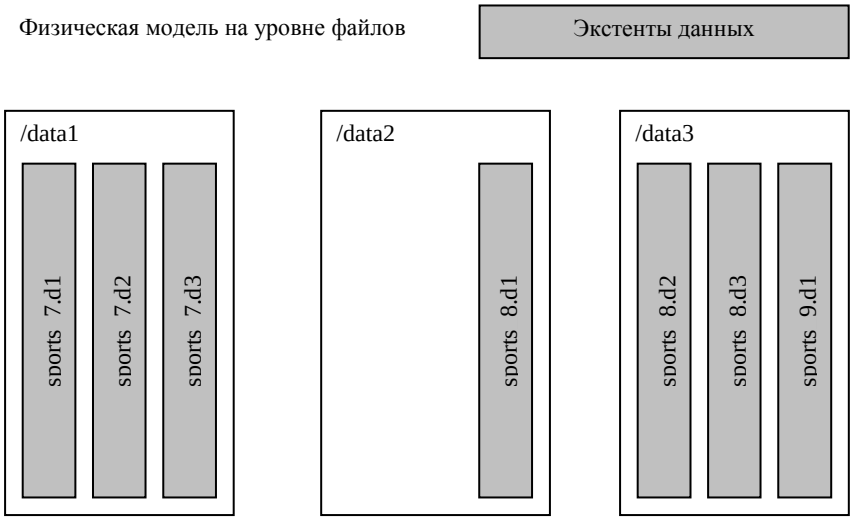
Свободные блоки (Free blocks) – каждый из них имеет заголовок, но они не содержат данных внутри. Такие блоки предназначены для формирования других возможных блоков в базе данных. Они всегда расположены ниже уровня High-water mark. Значение High-water mark - это указатель на последний отформатированный блок в пределах области хранения базы данных. Свободные блоки могут быть созданы автоматически, когда уровень HWM поднимается, т.е. база данных растет, или когда происходит переформатирование блоков во время перестройки индексов.

Пустые блоки (Empty blocks) не содержат заголовочной информации, поэтому перед использованием они должны быть отформатированы. Они располагаются выше уровня HWM, но ниже общего количества блоков в области хранения. Общее количество блоков (total blocks) - это общее количество размещенных блоков в области хранения.

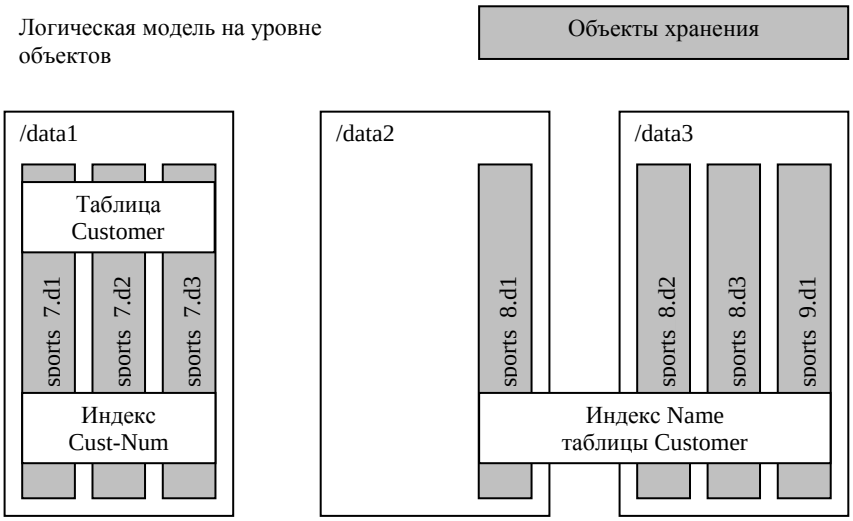
Если пользователь удаляет множество записей, то RM-блоки помещаются в RM-цепочку. Однако индексные блоки могут быть восстановлены только через перестройку индекса или через сжатие индекса.

Области хранения

Области хранения в OpenEdge делятся на две части, а именно, на физическую и логическую модели. У вас есть возможность управлять физической моделью с помощью ABL, OpenEdge SQL, а так же с помощью административных утилит. На следующем рисунке представлено, как области могут размещаться и распределяться между файлами. Обратите внимание, что экстенты областей могут быть размещены во множестве файлов на разных дисках, но в тоже время каждый экстент может быть связан только с одной областью.



Логическая модель хранения перекрывает физическую модель. Логические объекты базы данных описаны в схеме базы и включают таблицы, индексы и секвенции, которыми манипулирует ваше приложение. На следующем рисунке представлено, как логические объекты распределяются в пределах физической модели.



Создавая области хранения и соответствующие связи между ними и объектами, вы тем самым повышаете уровень производительности вашей базы данных. Вы можете размещать по одному объекту в каждой области хранения, а можете, наоборот, комбинировать различные объекты в одной области. Информация из следующей таблицы поможет вам определиться с этим вопросом.

Если база данных содержит…	**То вы должны…**
Много больших или часто обновляемых таблиц.	Создать область хранения для каждой такой таблицы. Это позволит легче выполнять администрирование и улучшит производительность.
Много маленьких таблиц.	Создать области хранения для представления конкретных предметов (например, Sales, Inventory и т.п.), и разместить в них связанные с этим предметом таблицы. Это обеспечит легкое администрирование.
Большие и маленькие таблицы.	Использовать две вышеописанные рекомендации.
Много индексов для больших или часто изменяемых таблиц.	Создать область хранения для каждого индекса. Это обеспечит гибкость в администрировании и потенциально улучшит производительность.
Много маленьких индексов.	Создать области хранения для индексов, принадлежащих объектам из общей группы. Это обеспечит простоту администрирования.

В версии OpenEdge 10 области хранения могут использовать как архитектуру SAT I, так и архитектуру SAT-II[18]. По умолчанию используется архитектура SAT-I. Для создания новой пользовательской области с архитектурой SAT-II в структурном файле необходимо в описании области указать размер кластера данных, равным 8, 64 или 512 (блоков). По умолчанию он равен 1, это означает, что область использует архитектуру SAT-I.

Некоторые рекомендуемые правила по размещению областей хранения на дисках:

- для защиты базы от дисковых сбоев AI-области лучше разместить на диске, отдельном от тех дисков, которые используются областями базы данных Control Area и Primary Recovery;

[18] За исключением области Schema Area, которая может иметь только архитектуру SAT-I.

- для улучшения производительности можно поместить область Primary Recovery на диск, отдельный от дисков, на которых размещены область Control Area и ее экстенты;
- наиболее активные области хранения стоит размещать на отдельных быстрых дисках;
- области хранения, которые содержат только архивную информацию, можно разместить на относительно медленных дисках;
- размещайте области для хранения таблиц на дисках, отдельных от дисков, на которых размещены области для хранения индексов.

Параметр RPB

В OpenEdge можно указать максимальное количество записей, которое может быть размещено в одном блоке конкретной области (Records Per Block - RPB). Это значение может быть установлено от 1 до 256. В зависимости от фактического размера записи может получиться так, что максимальное значение количества записей в блоке никогда не будет достигнуто.

Для определения количества записей в блоке необходимо выполнить ряд действий.

1. Взять средний размер записи из табличного анализа базы данных (tabanalys).
2. Добавить к этому размеру 20 (directory entry overhead), поскольку в средний размер записи включается record overhead, но в него не включен directory entry overhead.
3. Разделить размер блока базы данных в байтах на число, полученное из предыдущего шага.
4. Округлить полученное число из предыдущего шага до ближайшего значения, кратного двум.

В большинстве случаев длина записи не будет делиться на полученное значение из шага 4, поэтому придется подбирать наилучшее значение. Если определить слишком много записей в блоке, то можно подвергнуться риску фрагментации (когда одна запись распределена между несколькими блоками). Если же определить слишком малое количество записей в блоке, то некоторое пространство внутри блока будет потрачено впустую, т.к. оно не будет заполняться.

Следующий пример демонстрирует, как определить наилучшее значение для RPB. Допустим, что из табличного анализа (tabanalys) мы извлекли следующую информацию:

- таблица содержит 1 миллион записей;
- средний размер записей равен 59 байт.

Добавляем directory entry overhead (20 байт) для определения актуального размера хранимых записей:

Mean record size + overhead = actual storage size
59 + 20 = 79 байт

Разделите размер блока базы данных на полученное число, чтобы определить оптимальное RPB:

Database block size / actual storage size = optimal records per block
8192 / 79 = 103

Вот здесь придется принимать решение. Нужно выбрать ближайшее значение, кратное двум, в пределах от 1 до 256. В данном случае есть два варианта - 64 и 128. Если выбрать 64, то слоты для записей исчерпаются прежде, чем закончится место в блоке. Если выбрать 128, то появится риск возникновения фрагментации записей. Выбор нужно сделать соответственно характеру записей. Если записи растут динамически, то во избежание фрагментации нужно выбрать меньшее значение (64). Если записи добавляются, но обычно не изменяются, т.е. статичны в размере, то лучше выбрать большее значение RPB (128). В основном OpenEdge старается вставлять записи так, чтобы не создавать фрагментов, фрагментация обычно возникает, когда запись изменяется. Вероятность роста размера часто изменяемой записи достаточно велика.

Можно оценить свой выбор с точки зрения дискового пространства. Возьмите количество записей в таблице и разделите его на RPB – вы определите количество блоков, которые будут размещены для хранения этих записей:

Number of records / records per block = allocated blocks
1 000 000 / 64 = 15625

Затем вычислите количество неиспользованных байт в блоках, умножив физический размер записи на RPB, и отнимите полученное число от размера блока базы данных:

Database block size – (Actual storage size * records per block) = Unused space per block
8192 – (79 * 64) = 3136

Возьмите количество размещенных блоков и перемножьте его на неиспользованное пространство в блоке – вы определите общее количество неиспользованного пространства:

Allocated blocks * unused space per block = total unused space
15625 * 3136 = 49000000

В нашем случае общее количество неиспользованного пространства, которое является следствием выбора маленького значения RPB, немного меньше 47Мб. С точки зрения дискового пространства цена за устранение фрагментации достаточно низкая. Однако для статичных записей предпочтение лучше отдать более высокому значению, т.к. это позволит полностью заполнить блоки и в результате получить большее количество считываемых записей в буферный пул базы данных за одну операцию чтения.

Количество записей определяет количество блоков в области, поэтому важно, чтобы можно было использовать все слоты, тем самым получив максимальное количество записей в таблице.

Параметр BPC

BPC (Blocks Per Cluster) – количество блоков в одном кластере в архитектуре SAT-II. Кластер - это выделенное непрерывное пространство для однотипных объектов базы данных. Их использование значительно уменьшает фрагментацию и позволяет базе данных более эффективно использовать файловую систему. Кластеры данных настраиваются для каждой области данных. При этом размер кластера действует на все экстенты этой области. Минимально возможный размер кластера равен 8 блокам. Но можно увеличить его, указав 64 или 512 блоков на кластер. Все блоки в пределах кластера могут содержать один тип объектов. Экстент каждый раз увеличивается на размер кластера.

В архитектуре SAT-I блоки форматируются по одному. В архитектуре SAT-II за один раз форматируется целый кластер, состоящий из множества блоков. В этой архитектуре

данные поддерживаются на уровне кластера, а блоки содержат данные, связанные только с одним конкретным объектом.

Дисковое пространство. Прямое влияние - под объект хранения в базе выделяется часть пространства в виде целого кластера, даже если нужно меньшее пространство. Если объект в течение «жизни» будет расти, не становясь заведомо меньше, чем размер кластера, то не стоит беспокоиться о том, что некоторое пространство на диске занято «цифровым воздухом». Речь идет об области для хранения записей. Каждый индекс занимает на диске существенно меньше места, чем сама таблица. Чем больше записей в таблице, тем больше это расхождение, что обусловлено алгоритмом компрессии, используемым для хранения индексных ключей. Например, индексный блок размером 8K легко может хранить 10 000 ключей. Если средний размер записей равен, например, 100 байтам, то для 10 000 записей понадобится 1 МБ или порядка 100 блоков по 8K. Поэтому возможно, что не стоит использовать большой размер кластеров для индексов. Непрямое влияние размера кластера на объем используемого дискового пространства присутствует в связи с размером таблицы Cluster Allocation Block (или Cluster Map Block), которая содержит карту свободных блоков в области. Каждому кластеру в области соответствует в этом блоке (или в этих блоках) один бит. Чем больше размер кластера, тем меньше нужно места в Cluster Allocation Block. Если размер кластера равен 512 * 8K, то объем таких блоков составляет 1/4194304 объема области. Например, для области размером 100 ГБ объем карты размещения кластеров составит всего лишь 25 КБ. При размере кластеров в 8 блоков бинарная карта займет в 64 раза больше места. Хранятся ли эти блоки в кэше базы постоянно, и как часто Progress к ним обращается - неизвестно. Но очевидно, лучше иметь компактную карту, что является доводом в пользу большого размера кластера.

Конкуренция за ресурсы. Когда свободное место внутри области заканчивается, то область расширяется на один кластер. Процессы, которым необходимо сбросить свои данные в новые блоки, должны ждать пока ОС (операционная система) выделит новое место на диске. Чем больше размер кластера, тем дольше придется ждать (дольше, но реже). Однако если используется Enterprise-лицензия, то записью на диск должны заниматься исключительно процессы APW, а они являются самонастраивающимися, т.е. если в какой-то момент в их работе возникнут затруднения, то потом они могут и ускориться. От APW никто не требует, чтобы они выполнили свою работу как можно быстрее. Наоборот, они должны служить амортизаторами при записи на диски. Стало быть и здесь не о чем беспокоиться. А если используется только пространство внутри фиксированных экстентов, то вообще нет повода для волнений. Когда в одной области хранятся разные объекты и им одновременно необходимо новое пространство, то может возникнуть конкуренция за раздачу свободных кластеров. Чем больше размер кластера, тем реже будет возникать конфликтная ситуация, но для индексов это не так актуально. Поскольку индексы занимают меньше места, то им значительно реже придется покушаться на новое жизненное пространство. Конкуренция за место для записей таблиц будет на порядок сильнее.

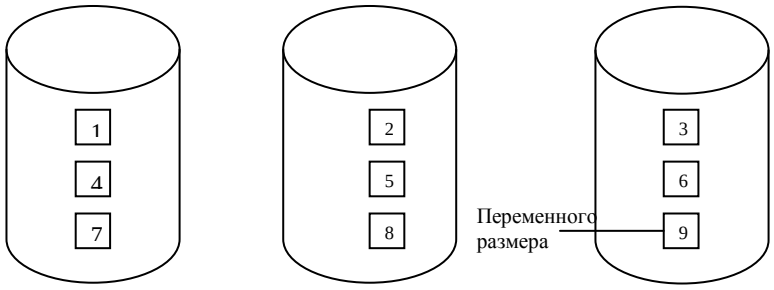
Возможные изменения в будущем. Сейчас Progress поднимает в память отдельные блоки, а не кластер целиком. Возможно, в будущем это изменится. Действительно, с точки зрения дисковых операций выгоднее читать сразу большой объем данных. Измененные данные будут по-прежнему писаться на диск поблочно. Чтение целой пачки блоков хорошо для блоков с записями. Индексные же блоки часто читаются в непредсказуемом порядке. Корневой индексный блок, промежуточные и лепестковые блоки – все они могут быть разбросаны как угодно относительно друг друга, и их расположение на диске может со временем меняться (в первую очередь речь идет о корневом индексном блоке). Это означает, что чтение большой порции данных с диска для индексов не столь актуально, как для записей, поэтому можно рекомендовать большой кластер для таблиц и маленький для индексов.

Таким образом, исходя из общей практики, рекомендуется значение BPC устанавливать для областей с записями, равное 512, а для областей с индексами - равное 8.

Экстенты областей хранения

Экстент области хранения - это часть области, выделенная в отдельный файл на диске. Наиболее часто администраторы используют минимум 2 экстента, поскольку их мониторинг легко выполнять на уровне операционной системы. Каждая область должна содержать один экстент переменной длины, который должен быть последним в области, что связанно с возможным ростом размера области.

Экстенты позволяют распределять данные между множеством физических дисков. Например, если выбрано восемь экстентов по 10 Мб и один экстент переменного размера, то можно выполнить страйпинг (stripe) этих экстентов между тремя дисками, как представлено на рисунке ниже. Поместите первый, четвертый и седьмой экстенты на первый диск; второй, пятый и восьмой экстенты на второй диск; третий, шестой и переменный экстенты на третий диск. OpenEdge по-прежнему будет заполнять эти экстенты по порядку. Используя страйпинг, вы получите смешивание старых и новых данных относительно их хранения на дисках. Хотя такой страйпинг не так эффективен, как страйпинг на уровне аппаратных средств, это все равно поможет устранить неравномерное использование дисков. На рисунке приведен страйпинг экстентов области хранения.



Даже если у вас работает аппаратный страйпинг, все равно может понадобиться использование множества экстентов. По умолчанию экстент ограничен размером файла в пределах 2 Гб. Если необходимо сохранить больше данных, чем в этом ограничении, то придется создать дополнительные экстенты в области или включить поддержку больших файлов, что теоретически позволит разметить в одном экстенте до 1 Тб данных.

Поддержка больших файлов

Как отмечалось ранее, по умолчанию каждый экстент области хранения имеет ограничение по размеру в 2 Гб. Это ограничение можно снять, применив к базе данных команду PROUTIL ENABLELARGEFILES (в offline):

```
proutil <db-name> -C enablelargefiles
```

Предварительно рекомендуется сформировать резервную копию базы данных. Отключить поддержку больших файлов до версии OpenEdge 10.2B SP02 можно только применив полную перезагрузку данных в новую базу. С версии 10.2B SP03 – с помощью утилиты PROREST с параметром <-keeptargetlfe>.

Основное преимущество поддержки больших файлов заключается в том, что исключается необходимость контроля над размерами экстентов областей хранения. При этом количество экстентов может быть уменьшено, что облегчает управление базой данных.

Глава 5 Способы создания и удаления баз данных

Существует несколько способов создания базы данных OpenEdge – с помощью таких утилит Progress, как PROSTRCT CREATE, PRODB, PROCOPY или PROREST, а так же с помощью пользовательских интерфейсов Data Dictionary и Data Administration.

Утилита PROSTRCT CREATE

Для создания базы данных с помощью утилиты PROSTRCT CREATE необходим структурный файл, который должен содержать описание областей хранения и их экстентов.

Структурный файл

Структурный файл - это текстовый файл, содержащий описание физической структуры базы данных. Он содержит всю информацию, необходимую утилите PROSTRCT CREATE для создания контрольной области (Control Area) и прочих областей базы данных со всеми принадлежащими им экстентами. Для его создания можно использовать любой текстовый редактор. Имя файла должно состоять из имени базы данных и расширения .st. Структурный файл может содержать одну или более строк текста с описанием каждой области базы данных. Каждая строка - это набор признаков, описывающих следующие характеристики области хранения.

1. Тип области хранения;
2. Имя области;
3. Номер области;
4. Параметр RPB (Records per Block);
5. Параметр BPC (Blocks Per Cluster) для областей хранения SAT-II;
6. Путь к экстенту области;
7. Тип экстента;
8. Размер экстента.

Формат структурного файла и таблица демонстрируют, как эти 8 признаков комбинируются для описания экстентов.

```
line = comment | CR | type path [sizeinfo]

comment = * | : | #

CR = blank line
type = a | b | d | t [areainfo]
       areainfo = "areaname"[:areanum][,recsPerBlock][;blksPerCluster]
              areaname = string
              areanum = numeric value
              recsPerBlock = numeric value
              blksPerCluster = 1 | 8 | 64 | 512
path = string
sizeinfo = extentType size
extentType = f | v
size = numeric value > 32
```

Признак	Описание
type	Показывает тип области хранения. Может принимать следующие значения: a – After-image area b – Before-image area d – Schema and application data area t – Transaction log area
areaname	Наименование области хранения.
areanum	Номер области хранения.
recsPerBlock	Количество записей в одном блоке базы данных. Возможные значения – 1,2,4,8,16,32,64,128 и 256.
blcksPerCluster	Количество блоков базы данных в кластере. Возможные значения – 1, 8, 64 и 512.
path	Абсолютный или относительный путь к каждому экстенту области.
extentType	Показывает тип экстента области. f – фиксированный v - переменный. Если значение признака не определено, экстент считается переменным по размеру.
size	Размер экстента в килобайтах. Это значение должно быть кратным 16 по отношению к размеру блока базы данных.

Утилита PROSTRCT CREATE сконструирована так, чтобы работать с минимумом предоставленной информации. Ей достаточно, чтобы были определены тип области и место нахождения её экстентов. Имена файлов, так же как и их расширения, могут быть опущены. Единственное ограничение для путей файлов, которое может возникнуть, это возможные ограничения вашей операционной системы. PROSTRCT CREATE генерирует имена и расширения для всех файлов базы данных согласно следующим правилам.

- Контрольная область (Control Area) (.db) и файл лога базы данных (.lg) размещаются в директории, определенном параметром командной строки *dbname.*
- Если указаны относительные пути, включая символ точки (.), то они будут расширены до абсолютных путей. Относительный путь начинается с вашего рабочего директория.
- Для файлов Before-image имя файла будет состоять из имени базы данных и расширения .b*n*, где *n* - порядковый номер файла, начиная с единицы;
- Для файлов After-image имя файла будет состоять из имени базы данных и расширения .a*n*.
- Как и для предыдущих двух областей, для файлов Schema Area имя файла будет состоять из имени базы данных и расширения .d.
- Для Transaction log имя файла будет состоять из имени базы данных и расширения .t*n*.
- Для областей данных имя файла будет сформировано из имени базы данных, нижнего подчеркивания и номера области, например, customer_7.d1. Номер области - это уникальный идентификатор, который отличает одну область от других областей. Файлы областей данных также имеют расширение .d*n*.

Минимум информации, который может содержаться в st-файле, это один экстент для области Schema Area и один экстент для области Primary Recovery Area (BI). Эта информация должна быть определена для любой области хранения, как и путь к её экстентам. Например, следующей информации будет достаточно для создания простой базы данных:

```
#Primary Recovery Area
b .
#Schema Area
d .
```

Если не будет указан путь к экстенту области Primary Recovery, то PROSTRCT CREATE сгенерирует ошибку. Нельзя использовать имена зарезервированных областей в качестве имен для областей хранения данных.

Вы можете определять размер экстента области хранения фиксированным или переменным. Когда создается экстент фиксированного размера (фиксированной длины), то его блоки предварительно размещаются и форматируются под базу данных. Для его создания нужно в структурном файле установить для него соответствующий признак – “f”. При использовании этого признака обязательно укажите признак размера файла в килобайтах. Этот размер должен быть кратным числу (16 * blocksize). Если будет определен не кратный размер, PROSTRCT CREATE отобразит предупреждающее сообщение и автоматически округлит значение до ближайшего кратного числа. Минимально допустимый размер экстента - 32Кб, максимальный – ограничен размерами вашей файловой системы и физическим объемом содержимого файла.

Экстенты переменного размера обычно используется на тот случай, если все экстенты фиксированного размера заполнятся. Для областей .db и .bi можно определить по одному переменному экстенту, который должен быть последним в области. Для области AI (After-Image) ограничений на количество экстентов переменного размера не существует. При определении типа экстента вы также можете указать его максимальный размер, до которого он может вырасти, указав опцию «v». Начальное значение такого экстента равно 32Кб или размеру кластера области. Максимальный размер может быть сколько угодно большим. Размер кластера можно вычислить, перемножив размер блока на количество блоков в кластере.

В следующем примере демонстрируется st-файл с именем `sports2000.st`, который содержит:

- Одну область Primary Recovery;
- Одну область Schema Area;
- Четыре After-image экстента фиксированного размера;
- Шесть областей, имеющих по два экстента. Эти области названы следующим образом: Employee, Inventory, Cust_Data, Cust_Index, Order и Misc. Заметьте, что Cust_Data, Cust_Index и Order содержат в описании размер кластера, это значит, что они имеют тип SAT-II. Пока не определено количество блоков на кластер (BPC), область считается SAT-I.

Пример структурного файла базы данных Sports2000:

```
#
b ./sports.b1
#
d "Schema Area":6,32;1 ./sports.d1
#
d "Employee":7,32;1 ./sports_7.d1 f 320
d "Employee":7,32;1 ./sports_7.d2
#
d "Inventory":8,32;1 ./sports_8.d1 f 640
d "Inventory":8,32;1 ./sports_8.d2
#
d "Cust_Data":9,32;512 ./sports_9.d1 f 2048
d "Cust_Data":9,32;512 ./sports_9.d2 v 2048
#
d "Cust_Index":10,32;8 ./sports_10.d1 f 320
d "Cust_Index":10,32;8 ./sports_10.d2
#
d "Order":11,32;512 ./sports_11.d1 f 2048
d "Order":11,32;512 ./sports_11.d2
#
d "Misc":12,32;1 ./sports_12.d1 f 320
d "Misc":12,32;1 ./sports_12.d2
#
a ./sports.a1
#
a ./sports.a2
#
a ./sports.a3
#
a ./sports.a4
```

Выполнение RPOSTRCT CREATE

Синтаксис утилиты PROSTRCT CREATE:

```
prostrct create <db-name> [st-file] [-blocksize blocksize] [-validate]
```

В синтаксическом блоке вместо `st-file` указывается имя структурного файла (.st). Если имя файла не будет указано, то PROSTRCT CREATE использует файл `db-name.st`. Размер блока базы указывается в килобайтах и представлен параметром <-blocksize>, который может быть определен как 1024, 2048, 4096 или 8192.

Если вы определяете параметр <-validate>, то PROSTRCT проверит содержимое указанного структурного файла, не создавая при этом физической структуры базы данных. Для просмотра работы утилиты в действии воспользуемся приведенным выше структурным файлом, для чего выполним следующие шаги:

1. Сначала проверьте корректность структуры:

   ```
   prostrct create sports2000 sports2000.st -blocksize 4096 -
                                validate
   ```

   Если ошибок обнаружено не будет, утилита вернет сообщение, аналогичное следующему:

   ```
   The structure file format is valid. (12619)
   Device: /dsk3/, KBytes needed: 10432, KBytes available: 220713861 (12616)
   There is sufficient free space to initialize the defined extents. (12618)
   ```

2. После проверки структурного файла вы можете приступить к созданию структуры базы данных, убрав параметр <-validate>:

```
prostrct create sports2000 sports2000.st -blocksize 4096
```

   Обратите внимание на результат работы утилиты, который выведен на экран. Хотя мы и указали в структурном файле только относительные пути к каждому экстенту базы данных, утилита PROSTRCT CREATE преобразовала эти пути в абсолютные, как и отмечалось ранее.

Создание структуры базы данных еще не означает, что мы получили работоспособную базу. Сейчас мы создали базу «пустышку», в которой нет метасхемы.

Добавление метасхемы во вновь созданную базу данных

При создании базы данных с помощью утилиты PROSTRCT CREATE и st-файла результатом станет так называемая «пустая база данных». Пустая база данных не содержит метасхемы OpenEdge. Она состоит только из файлов .db, .bi, .ai и файлов для хранения данных .dn, т.е. из всего того, что было описано в st-файле. Поэтому необходимо добавить в «пустую базу данных» информацию о метасхеме. У OpenEdge имеются empty-базы, которые содержат метасхему для каждого размера блока базы данных. Для добавления метасхемы необходимо выполнить действия.

1. Чтобы проверить, что была создана база данных с правильным расположением всех необходимых файлов, выполните утилиту PROSTRCT LIST:

```
prostrct list <db-name> ./validate.st
```

   Здесь `validate.st` - это файл с любым именем, в который будет выгружена текущая структура базы данных. **Внимание**! Если вы не укажите имя структурного файла, то OpenEdge перезапишет существующий структурный файл базы данных с именем `db-name.st`. Откройте созданный файл и проверьте список областей, описанных в нем, на правильность их размещения и создания.

2. Выполните утилиту PROCOPY для копирования метасхемы OpenEdge из empty-базы в базу, созданную утилитой PROCTRCT CREATE:

```
procopy $DLC/emptyN <db-name>
```

   Здесь *emptyN* - это исходная база данных, *db-name* – целевая, *N* - размер блока базы данных, поддерживаемый выбранной empty-базой.

Теперь вновь созданная база данных готова принять схему вашего приложения, которую вы можете загрузить с помощью Data Dictionary.

Важные замечания:

- Empty-база и вновь создаваемая база должны иметь одинаковый размер блока.
- Не создавайте базы данных в установочном директории OpenEdge. Базы данных, находящиеся в этом директории или в его поддиректориях, не могут быть запущены.

- В структурном файле можно оставлять свои комментарии, используя пустые строки. Начало комментария должно начинаться с одного из символов: «#», «:», либо «*».
- Если вы не укажите номер области хранения, PROCTRCT CREATE сам проставит нумерацию. Тем не менее, если номер будет указан хотя бы для одной области, вам придется установить номера для всех областей, иначе утилитой будет выведена ошибка.
- Признак RPB поддерживается только для областей с номерами от 7 до 32000. Для области Schema Area значение RPB зафиксировано как 32 для 1Кб, 2Кб и 4Кб-го размера блока базы данных. Для 8Кб блока это значение равно 64.
- BPC - если вы оставите значение этого признака пустым или установите в 1, то область данных будет создана с архитектурой SAT-I. Все остальные значения верны только для области SAT II.

Утилита PRODB

Если целевая база данных не имеет своего st-файла, то утилита PRODB создаст базу на основании структурного файла исходной базы, размещая все экстенты в каталоге, который был указан для новой базы данных. Вы можете использовать утилиту для создания любой демонстрационной или empty-базы данных. Следующий пример демонстрирует работу утилиты PRODB.

Для создания пустой базы данных с именем mytestdb из empty-базы данных выполните команду:

```
prodb ./db2/mytestdb empty
```

Теперь в каталоге ./db2 создана база данных mytestdb с уже загруженной метасхемой. Обратите внимание на файл `mytestdb.st`. Он содержит описание файлов базы данных с относительным путем:

```
#
b ./db2/
#
d "Schema Area":6,32;1 ./db2/
```

При создании базы данных утилитой PRODB используются как относительные, так и абсолютные пути к файлам. Например, если утилитой PRODB использовать относительный путь к создаваемой базе mytestdb, то новая база будет создана с относительными путями. И наоборот, если для создаваемой базы будет использоваться абсолютный путь, то для новой базы также будет определен абсолютный. Если выполнить ту же команду, но с указанием полного пути к создаваемой базе данных:

```
prodb `pwd`/db2/mytestdb empty
```

то увидите, что в структурном файле пути к экстентам вновь созданной базы данных будут абсолютными:

```
#
b /users/db2/
#
d "Schema Area":6,32;1 /users/db2/
```

Используйте PROSTRCT LIST для преобразования относительных путей в абсолютные пути.

Внимание! Утилита PRODB не копирует внешние триггеры, связанные с копируемой базой данных.

Утилита PROCOPY

Для создания копий существующих баз данных предназначена утилита PROCOPY, которая копирует структуру базы данных вместе с ее содержимым. Нежелательно использовать утилиты операционных систем для копирования баз данных, о причинах этого будет рассказано далее.

Утилита PROCOPY поддерживает области хранения. Следовательно, если целевая база данных существует, она должна поддерживать такой же тип областей, их количество и такие же типы экстентов, как и в исходной базе данных. Тем не менее, количество экстентов на целевой базе не всегда должно совпадать с количеством экстентов на исходной базе данных.

Если целевая база данных не существует, PROCOPY создаст ее, используя структурный файл, находящийся в каталоге целевой базы. Если же структурного файла нет, база будет создана на основании структурного файла исходной базы, но при этом все экстенты будут расположены в том же директории, в котором находится и db-файл. Это произойдет даже, если файлы исходной базы данных будут расположены в различных директориях.

При использовании утилиты PROCOPY целевая база всегда создается с абсолютными путями, независимо от соглашений о путях, используемых в исходной базе данных. Например, если создается копия базы данных Sports2000 с использованием относительных путей, целевая база все равно будет создана с соглашением об абсолютных путях.

Синтаксис утилиты PROCOPY следующий:

```
procopy <source-db> <target-db>
```

Удаление базы данных

Для удаления баз данных используется утилита PRODEL.

Пример:

```
prodel ./db2/mytestdb
```

Когда вы удаляете базу, утилита PRODEL выводит сообщение с указанием всех файлов, начинающихся с имени db-name (имя удаляемой базы данных). PRODEL выводит соответствующее сообщение для подтверждения удаления в зависимости от вашей операционной системы.

Утилита PRODEL не удаляет структурный файл базы данных, чтобы информация о ее структуре была сохранена, но она удаляет файл лога базы данных, поэтому, если лог базы для вас все еще представляет интерес, то предварительно сохраните его отдельно от базы данных. Так же утилита не удаляет ряд других специфических файлов, таких как db-

`name.repl.recovery` и `db-name.archival.log`. Всегда помните об этом при намерении создать другую базу данных с таким же именем на месте удаленной базы данных. Не допустите путаницы!

Глава 6 Запуск и останов базы данных

Команды запуска базы данных OpenEdge поддерживают большое количество конфигурационных опций. В клиент-серверной конфигурации клиентские процессы и серверные процессы базы данных разделены. Интерфейс пользователя ABL-приложений выполняется на стороне клиентской сессии, а запросы отправляются на сервер. Сервер получает доступ к базе данных от имени каждой клиентской сессии. Следовательно, команды запуска разделяются на две общие категории.

- Команды для запуска и останова процессов базы данных (обмена данными с сервером БД, с брокером БД) и фоновых процессов.
- Команды для запуска и останова клиентских соединений с базой данных, таких как ABL client, AppServer и Data Server client.

В общих чертах запуск клиента и сервера происходит следующим образом.

- Запускается сервер базы данных (или брокер базы данных). Старт этих процессов устанавливает постоянное соединение с базой данных.
- Запускается клиентское приложение.

Вы можете использовать параметры запуска совместно с командами запуска, определяющими режим работы базы данных и клиентских сессий. Требования к использованию параметров зависят от операционной системы и сетевой среды. Например, вы можете контролировать количество клиентских соединений для обеспечения наибольшей производительности.

Общий синтаксис команд выглядит следующим образом:

```
command [ db-name ] [ parameter ] [ value ]
```

Например, следующая команда запуска обеспечит доступ 100 пользователям к базе данных и установит необходимые значения сетевых параметров:

```
proserve ./sports -n 100 -B 10000 -L 2000 -S 60900 -H myhost
```

Общий список команд запуска процессов базы данных представлен в таблице:

Назначение	Команда
Запуск серверов и брокеров конкретной базы данных.	proserve *db-name parameter*
Запуск процесса APW для базы данных.	proapw *db-name*
Запуск процесса BIW.	probiw *db-name*
Запуск процесса AIW.	proaiw *db-name*
Запуск процесса Watchdog.	prowdog *db-name*
Блокировка всех записей в базе данных через активацию «quiet point».	proquiet *db-name parameter*
Останов всех процессов базы данных (APW, BIW, AIW, Watchdog). Используйте первый пункт команды для отсоединения конкретного процесса.	proshut *db-name*

Остановимся более подробно на каждой из этих команд.

PROSERVE

Основной сервер базы данных называется брокером, он управляет общими ресурсами и при необходимости может запускать дополнительные сервера. Запуск брокера базы данных обеспечивает команда PROSERVE. Синтаксис команды следующий:

```
proserve <db-name>  [ parameters ]
```

Параметр	Назначение
db-name	Имя базы данных, которую необходимо запустить.
parameters	Определяются параметры запуска для брокера или сервера.

Команда PROSERVE может иметь множество различных дополнительных параметров запуска базы данных. Эти параметры в основном и определяют режим работы и уровень производительности базы данных.

PROAPW

Фоновый процесс APW[19] обеспечивает запись измененных буферов базы данных из буферного пула базы на диск. Запускает процесс команда PROAPW. Процессы APW улучшают производительность системы за счет обработки второстепенных задач в фоновом режиме. Синтаксис команды следующий:

```
proapw db-name
```

Оптимальное количество процессов APW зависит от ваших приложений и среды выполнения. Используйте минимум один APW для каждого жесткого диска, на котором расположены экстенты базы данных. Если этого количества недостаточно, можно добавить еще один. Если существуют приложения, которые выполняют большое количество изменений, то помимо запуска по одному APW на каждый диск, запускается еще один дополнительный процесс. Чем меньше приложение вносит изменений в базу данных, тем меньше APW-процессов ему требуется.

Каждый процесс APW использует одну пользовательскую лицензию. Поэтому необходимо учитывать количество этих процессов при расчете значения параметра базы

[19] APW - Asynchronous Page Writer

данных Number of Users (*-n*). Тем не менее APW не входит в состав лицензированных пользователей, и вам не нужно учитывать их при закупке лицензий.

Для остановки работы процесса APW используйте команду PROSHUT. Запускать и останавливать работу APW-процессов можно в любое время без остановки самой базы данных.

PROBIW

Процесс BIW[20] - это фоновый процесс, который занимается записью заполненных BI-буферов на диск, после чего размещает их в цепочке пустых буферов. Таким образом гарантируется наличие доступных пустых BI-буферов для серверных процессов и клиентских приложений. Процесс BIW процесс необязательный, но очень рекомендуемый при желании обеспечить хорошую производительность базы данных. Запуск процесса выполняется с помощью команды PROBIW и может быть осуществлен в любой момент времени после запуска базы данных (обычно сразу после ее старта). Синтаксис команды следующий:

```
probiw db-name
```

Для одной базы данных может быть запущен только один процесс BIW. Процесс использует одну пользовательскую лицензию, поэтому его нужно учитывать при расчете параметра Number of Users (-n). Но он нелицензируемый, а значит, при закупке лицензий не учитывается.

Для остановки работы процесса можно воспользоваться командой PROSHUT. Остановку процесса можно выполнять в любое время без остановки самой базы данных. Если база данных останавливается, а процесс BIW запущен, то он остановится вместе с базой данных.

PROAIW

Команда PROAIW предназначена для запуска процесса AIW[21]. Это фоновый процесс, который предназначен для улучшения производительности за счет сброса After-Image (AI) блоков на диск, тем самым обеспечивает клиентские процессы и процессы серверов пустыми AI-буферами, готовыми для использования. Синтаксис команды следующий:

```
proaiw db-name
```

Для одной базы данных может быть запущен только один процесс AIW. Запуск процесса осуществляется вручную, так же как и запуск процессов APW и BIW и может выполняться в любое время после запуска базы данных (обычно сразу после ее старта). Единственным условием запуска процесса AIW является наличие включенного механизма After-Imaging. Собственно говоря, если включен этот механизм, то запуск процесса AIW должен быть обязательным, в противном случае сбросом на диск AI-буферов будет заниматься сервер базы данных, что негативно отразится на производительности.

[20] BIW - Before Image Writer
[21] AIW, After-Image Writer

Остановить работу процесса можно в любое время, используя команду PROSHUT, не прибегая к остановке самой базы данных. Если же база будет остановлена, то процесс будет остановлен автоматически.

Процесс AIW должен учитываться при расчете параметра запуска базы данных Number of Users (-n), т.к. считается отдельным клиентским процессом. При этом его не нужно учитывать при расчете количества лицензий.

PROWDOG

Команда PROWDOG выполняет запуск процесса Watchdog. Этот процесс призван наводить порядок после неправильно завершенных процессов. Например, если завершенный процесс имел блокировки или использовал структуру разделяемой памяти, то Watchdog освободит их. Синтаксис команды следующий:

```
prowdog db-name
```

Если Watchdog обнаруживает процесс, который более не активен, он освобождает все соответствующие блокировки записей, откатывает все незавершенные им транзакции, освобождает все блокировки разделяемой памяти и закрывает соединение с базой. Если процесс является серверным, то происходит очистка и отключение всех удаленных клиентов. Если в момент прерывания процесса в разделяемую память этот процесс вносил изменения, то база данных останется в неопределенном состоянии, поэтому для обеспечения безопасности базы данных Watchdog выполнит ее останов.

Процесс Watchdog не может обнаружить потерю дистанционных клиентов, т.к. такие клиенты не связаны с процессом. Вместо этого сетевой протокол по истечении определенного времени (timeout) уведомит сервер о потере соединения с клиентом.

Для одной базы данных может быть запущен только один процесс Watchdog. Его запуск и останов могут осуществляться в любое время после запуска базы данных. Остановить его можно с помощью команды PROSHUT. Если же база данных будет остановлена, то процесс Watchdog будет остановлен автоматически.

Процесс Watchdog считается клиентским процессом, поэтому его необходимо учитывать при расчете параметра Number of Users (-n). При расчете пользовательских лицензий его учитывать не нужно.

С целью обеспечения лучшей производительности процесс Watchdog должен быть обязательно запущен для очень загруженных баз данных.

PROQUIET

С целью выполнения на базе данных некоторых административных задач средствами операционной системы, например, при копировании базы используется так называемая точка останова (quiet point). Команда PROQUIET предназначена для ее активации. Во время активации точки останова вся транзакционная деятельность базы данных замораживается. Поэтому любой процесс, который в это время попытается открыть транзакцию, будет вынужден ожидать, пока точка останова не будет деактивирована.

Синтаксис активации следующий:

```
proquiet db-name enable [ nolock ] | disable | bithreshold n
```

Параметр	Назначение
db-name	Имя базы, для которой активируется точка останова.
enable | disable	Активация или деактивация точки останова. Любой процесс при активированной точке останова вынужден ожидать ее деактивации.
nolock	Позволяет активировать точку останова без блокировки разделяемой памяти.
bithreshold *n*	Установка максимального размера BI-файла, до которого он может вырасти, где *n* указывает размер в мегабайтах. Вы можете увеличить размер порога выше текущего значения.

Команда PROQUIET ENABLE останавливает любые записывающие в базу данных процессы, а команда PROQUIET DISABLE восстанавливает возможность записи. Команда PROQUIET в основном используется для осуществления резервной стратегии. Это гарантирует, что копия базы данных будет оставаться работоспособной после восстановления.

С параметром bithreshold вы можете использовать эту команду для настройки порогового значения BI-файла в online. Для этого необходимо, чтобы при старте сервера использовался параметр <-bithold>. Это может понадобиться для предотвращения останова базы вследствие превышения порогового значения BI или для увеличения порогового значения, если база данных уже остановилась.

Рассмотрим небольшой пример. Запустите базу данных с помощью команды PROSERVER с параметром порогового значения BI, равным 250 MB:

```
proserve mytestdb -bithold 250 -bistall
```

В этом случае, перед тем как пороговое значение BI будет достигнуто (в основном чрезмерный рост BI-файла происходит из-за долгоиграющих транзакций), в журнал событий базы данных (.lg) будет записано следующее сообщение:

```
BI file size has grown to within 90 percent of the threshold value of <num>. (6559)
```

После получения сообщения 6559 вы можете в online увеличить пороговое значение, не допустив срабатывания параметра <-bistall>, что позволит вам выиграть время для нахождения причины роста BI-файла:

```
proquiet mytestdb bithreshold 1000
```

Выполнение этой команды увеличит пороговое значение, и база данных не будет остановлена. Небольшое замечание: чтобы не допустить срабатывания <-bistall> вы должны выполнить указанную команду в промежутке между 90% и 100% значения размера порогового значения. Если вы попытаетесь выполнить ее до достижения 90%, то вы получите следующее сообщение:

```
BI Threshold has not been reached and a Quiet Point has not been enabled. (6555)
Request to change BI File Threshold value rejected. (6554)
```

Очевидно, что запас времени (между 90% и 100%) который у вас есть, зависит от изначально установленного размера порогового значения. Таким образом, чем больше

значение параметра -bithold, тем больше времени у вас будет, чтобы отреагировать на сообщение 6559.

На практике команду PROQUIET лучше использовать в скриптах слежения за размером BI-файла для незамедлительного реагирования на его рост, поскольку в такой ситуации что-либо сделать вручную вы попросту можете не успеть.

Если база данных все-таки была остановлена (явно это, конечно, нельзя назвать остановом), то фактически активируется неявная точка останова, при этом в журнал событий будет внесена соответствующая запись. В этом случае необходимо увеличить пороговое значение BI, что возможно только с помощью команды PROQUIET с параметром bithreshold.

PROSHUT

Команда PROSHUT используется для остановки сервера OpenEdge базы данных или конкретных процессов. После остановки основного брокера базы данных все имеющиеся сессии будут отключены от базы. Перед остановкой базы данных желательно, чтобы все пользователи завершили свои сессии. При необходимости можно отключить конкретных пользователей с помощью функций Disconnect a User или Unconditional Shutdown команды PROSHUT.

Синтаксис команды:

```
proshut db-name [
                   -b | -by | -bn |
                   -H host-name | -S service-name -F |
                   -C list | -C disconnect username
                ]
```

Параметр	Назначение
-b	Останов будет выполнен в потоковом (batch) режиме. Когда ни одного пользователя не подключено, остановка произойдет автоматически. Если подключены один или больше пользователей, то PROSHUT запросит ввод «yes» для безусловного останова и отключения пользователей; или «no» для останова при отсутствии подключенных пользователей. Этот параметр может комбинироваться, например, -by или –bn.
-by	Безусловное отключение пользователей и останов базы данных.
-bn	Останов базы данных только при отсутствии активных пользователей.
-H *host-name*	Адрес (host) компьютера, на котором запущена база данных.
-S *service-name*	Имя сервиса или номер порта, на котором запущена база данных или брокерский процесс.
-F	Аварийный останов базы данных, используется только в Unix-системах. Для использования этого параметра необходимо запустить команду PROSHUT на том компьютере, на котором запущена база данных.
-C list	Формирование списка пользователей, подключенных к базе данных. Список выводится на экран без разделителя страниц. Использование параметра возможно только локально.
-C disconnect *usernum*	Отключение конкретного пользователя по его номеру в базе. Аналогично опции 1 меню команды PROSHUT. Использование параметра возможно только локально.

Использование параметра <-by> совместно с <-F> может вызвать непредвиденные последствия для базы данных. Безусловный принудительный останов <-byF> не может быть выполнен на базе, запущенной с параметром базы No-integrity <-i>. При принудительном останове базы <-F>, запущенной с параметром <-i>, будет выведено

диалоговое сообщение о действительности намерения принудительного останова базы данных, запущенной в режиме No-integrity. Помните, что при использовании этого метода остановки база данных может быть повреждена.

Для остановки базы данных можно воспользоваться утилитой PROMON и ее пунктом 8 Shut Down Database. Меню этого пункта такое же, как и меню команды PROSHUT.

```
Disconnect a User
Unconditional Shutdown
Emergency Shutdown (Kill All)
x Exit
```

Следующая таблица приводит назначение каждого пункта меню:

Номер меню	Назначение
1	Запрос номера пользователя для отключения.
2	Отключение всех пользователей и останов базы данных.
3	Задается вопрос о необходимости выполнения пункта. Если вы отмените выбор, то работа команды будет прервана. Если вы примете его, то PROSHUT выведет следующее сообщение: `Emergency shutdown initiated...` После этого база данных будет помечена как остановленная аварийно. Все подключенные процессы будут принудительно отключены, удалены все разделяемые сегменты памяти и семафоры, связанные с базой данных. База данных получит статус поврежденной базы. После перезагрузки базы выполнится нормальный процесс восстановления Crash Recovery с откатом незавершенных транзакций.
x	Прерывание работы команды без выполнения каких-либо действий.

Останавливать базу может только тот пользователь, который ее запускал, или имеющий статус суперпользователя (ROOT).

При запуске PROSHUT по сети количество времени, которое необходимо для завершения всех процессов OpenEdge и для освобождения портов, будет зависеть от числа клиентов, брокеров и серверов, которые нужно остановить, команда PROSHUT так же может возвратить управление терминалу, прежде чем все процессы базы данных будут остановлены.

Глава 7 Параметры базы данных и клиентской сессии

Параметры запуска базы данных - это один из инструментов управления ее поведением и производительностью. Существует множество параметров, которые разделены на несколько категорий, однако, некоторые из параметров входят в несколько категорий. Значения параметров запуска, установленных по умолчанию, можно менять. Для указания наборов параметров можно использовать файл параметров.

Файл параметров – это файл операционной системы, который содержит один или более параметров запуска. Первоначальной причиной использования файла параметров является исключение ручного ввода параметров при каждом выполнении команды. Вместо того чтобы многократно вводить несколько параметров и других аргументов, достаточно ввести их один раз в файл параметров. Файл параметров можно использовать для создания набора параметров для конкретной базы данных, группы пользователей или системной конфигурации. Файл параметров имеет расширение .pf. Каждая программа OpenEdge использует файл параметров, который по умолчанию называется `startup.pf`. Этот файл расположен в директории $DLC. Он должен находиться там, чтобы Progress работал правильно. Вы можете модифицировать этот файл для добавления/изменения набора параметров. Вы можете также создавать один или более дополнительных файлов параметров, которые будут использованы после загрузки `startup.pf`.

Для вызова дополнительного файла параметров в командной строке используйте параметр запуска Parameter File <-pf>:

```
pro sports –pf my_pf.pf
```

Имя файла может быть любым. Если один и тот же параметр упоминается несколько раз, то приоритет отдается последнему.

Пример файла параметров:

```
-db /users/valeriy/lecAI/db1/sports
-B 2000
-H myhost -S my_port
-L 1000
# Остальные параметры получат значения по умолчанию
```

Этот формат используется для всех операционных систем. Следуйте следующим правилам при создании файла параметров.

- Устанавливайте параметры и их аргументы в одной строке.
- Используйте максимум 40 символов в одной строке.
- Используйте знак # для выделения комментариев. Данные, следующие за этим знаком в строке, будут проигнорированы.
- Заключайте значения параметров в одиночные или двойные кавычки.
- Параметры запуска, не указанные в файле параметров, принимают значения по умолчанию.

В таблице приведен список наиболее часто используемых параметров при старте сервера базы данных:

Параметр	Синтаксис	Назначение
Blocks in Database Buffers	-B *n*	Количество буферов в первичном буферном пуле базы данных.
Alternate Buffer Pool	-B2	Количество буферов в альтернативном буферном пуле базы данных.
Lock Table Entries	-L *n*	Максимальное количество одновременно заблокированных записей в базе данных.
Number of Users	-n *n*	Максимальное количество пользователей, подключаемых к базе данных.
Spin Lock Retries	-spin *n*	Количество попыток получения блокировки разделяемого ресурса процессом. Параметр может значительно влиять на производительность базы данных.
After-image Buffers	-aibufs *n*	Количество AI-буферов, выделяемое базе данных. Применяется, если включен механизм After-Imaging.
After-image Stall	-aistall	Останавливает транзакционную деятельность в базе данных при отсутствии AI-экстентов со статусом EMPTY.
Before-image Buffers	-bibufs *n*	Количество BI-буферов. Применяется для улучшения производительности механизма Before-Imaging.
Threshold Stall	-bistall	Останавливает транзакционную деятельность в базе данных, когда размер BI-файла достигает порогового значения. Используется совместно с параметром <-bithold>.
Recovery Log Threshold	-bithold *n*	Допустимый размер BI-файла, по достижению которого будет остановлена транзакционная деятельность в базе данных.
Direct I/O	-directio	При использовании этого параметра база данных выполняет прямые операции чтения-записи в файлы базы данных.
No Crash Protection	-i	Режим работы базы данных без обеспечения целостности и возможности восстановления.
Pin Shared Memory	-pinshm	Блокировка разделяемой памяти, выделенной базе в оперативной памяти сервера. Использование параметра <-pinshm> может улучшить производительность.
Semaphore Sets	-semsets *n*	Количества наборов семафоров, доступных базе данных.
Secondary Login Broker	-m3	Запуск второстепенного брокера.
Host Name	-H *hostname*	Имя хоста, на котором запускается база данных.
Service Name	-S *service/port*	Имя сервиса или номер порта, используемого процессом брокера для подключения дистанционных клиентов.
Maximum Clients Per Server	-Ma *n*	Максимальное количество дистанционных пользователей на один сервер базы данных.
Minimum Clients Per Server	-Mi *n*	Минимальное количество дистанционных пользователей на сервере, по достижении которого брокер может запустить дополнительный сервер.
Maximum Servers	-Mn *n*	Максимальное количество запускаемых брокером серверов для дистанционных клиентов.
Servers Per Protocol	-Mp *n*	Максимальное количество серверов, обслуживающих дистанционных пользователей по одному протоколу.
Maximum Servers Per Broker	-Mpb *n*	Максимальное количество серверов, которое могут запустить брокеры для обслуживания дистанционных клиентов по одному протоколу.
Maximum Dynamic Server	-maxport *n*	Верхний предел доступных портов в диапазоне портов, открытых в Firewall-e.
Minimum Dynamic Server	-minport *n*	Нижний предел доступных портов в диапазоне портов, открытых в Firewall-e.
Century Year Offset	-yy *n*	Начало исчисления столетнего периода, где год указан двумя цифрами.

Параметры интернационализации:

Параметр	Синтаксис	Назначение
Conversion Map	-convmap *filename*	Таблица конвертации из одной кодировки в другую.
Case Table	-cpcase *tablename*	Таблица соответствия символов верхнего и нижнего регистров.
Collation Table	-cpcoll *tablename*	Таблица сортировки символов.
Internal Code Page	-cpinternal *codepage*	Таблица кодировки, используемая в памяти.
Log File Code Page	-cplog *codepage*	Таблица кодировки, используемая для записи сообщений в журнал событий базы данных.
Print Code Page	-cpprint *codepage*	Таблица кодировки, используемая для вывода символов на печать.
R-code in Code Page	-cprcodein *codepage*	Таблица кодировки, используемая скомпилированными ABL-программами (R-код).
Stream Code Page	-cpstream *codepage*	Таблица кодировки, используемая для потокового ввода/вывода.
Terminal Code Page	-cpterm *codepage*	Таблица кодировки для символьного терминала.

Параметры клиентской сессии:

Параметр	Синтаксис	Назначение
Host Name	-H *hostname*	Имя хоста базы данных, к которой подключается сессия.
Service Name	-S *service/port*	Имя сервиса или номер порта, на котором запущена база, через который осуществляется подключение дистанционных сессий.
Single-user Mode	-1	Подключение к базе данных в однопользовательском режиме. Аналог команды PRO.
Batch	-b	Старт клиентской сессии в фоновом режиме.
Private Read-only Buffers	-Bp	Количество частных буферов в режиме «только на чтение», выделяемое конкретной сессии.
Schema Cache File	-cache	Считывание локального файла, содержащего кэш-схемы базы данных, вместо прямого чтения из базы. Файл должен быть предварительно создан.
Client Logging	-clientlog	Автоматическая запись ошибок и различных сообщений, связанных с клиентской сессией, в отдельный лог-файл.
Log Entry Types	-logentrytypes	Тип информации, которая будет записываться в клиентский лог-файл.
Logging Level	-logginglevel	Уровень детализации информации, записываемой в клиентский лог-файл.
Number of Log Files to Keep	-numlogfiles	Количество возможных файлов для клиентского лога.
Log Threshold	-logthreshold	Максимальный размер каждого файла клиентского лога.
Number of Databases	-h	Количество одновременно подключенных OpenEdge баз данных.
Index Range Size	-indexrangesize	Количество индексов, по которым необходимо осуществлять мониторинг.
Table Range Size	-tablerangesize	Количество таблиц, по которым необходимо осуществлять мониторинг.
Password	-P	Пароль пользователя для подключения к базе данных.
User ID	-U	Идентификатор пользователя для подключения к базе данных.
Parameter File	-pf	Использование файла параметров.
Temporary Directory	-T	Каталог для хранения временных файлов клиентской сессии.
Save Temp Files	-t	Параметр делает видимыми временные файлы во временном каталоге.

Более детальную информацию о назначении всех параметров Progress OpenEdge вы можете получить из документации «OpenEdge® Deployment: Startup Command and Parameter Reference». Дополнительные описания некоторых из основных параметров будут рассмотрены в течение курса.

Глава 8 Способы доступа к базе данных

Доступ к базам данных OpenEdge может осуществляться как в однопользовательском, так и в многопользовательском режиме.

Однопользовательский режим предназначен для доступа к базе данных, находящейся в состоянии покоя, т.е. когда она остановлена. Такой доступ в основном используется для выполнения административных задач, например, для изменения схемы базы данных. Помимо этого однопользовательский доступ может применяться для выполнения специфических ABL-программ, не предполагающих наличие других пользователей, подключенных к базе во время работы этих программ. Для подключения к базе в однопользовательском режиме обычно используется следующая команда:

```
pro db-name
```

Команда `pro`, по своей сути, является SHELL-скриптом, размещенным в каталоге $DLC/bin, фактически выполняет запуск утилиты _progres с параметром клиентской сессии Single-user Mode <-1>. Если команде pro не передать других параметров, кроме имени базы данных, то она только выполнит подключение к указанной базе и откроет Procedure Editor. Помимо имени базы данных команде `pro` могут передаваться и дополнительные параметры клиентской сессии, например, это может быть имя ABL-программы, которое передается в качестве аргумента параметру Startup Procedure <-p>. В этом случае указанная программа будет выполнена в однопользовательском режиме без входа в Procedure Editor.

Пример выполнения программы в однопользовательском режиме.

- Сохраните в свой домашний каталог программу `single-pro.p`:

```
for each customer where cust-num >=1 and cust-num <=5 no-lock.
    displ cust-num name bal.
    pause.
end.

quit.
```

  Эта программа выведет на экран информацию по первым пяти клиентам. Обратите внимание на операторы `pause` и `quit`. Первый оператор здесь нужен для последующей проверки, подтверждающей невозможность пользователям подключиться к базе данных в момент работы программы в однопользовательском режиме. Второй оператор предназначен для того, чтобы вернуть управление командной строке, из которой программа `single-pro.p` была запущена, если же этого оператора не будет, то управление будет возвращено в Procedure Editor.

- Выполните программу `single-pro.p`:

```
pro sports –p single-pro.p
```

  Программа выведет на экран данные по первому клиенту и будет ожидать нажатия любой кнопки пользователем. В это время откройте вторую клиентскую сессию и попробуйте выполнить туже самую команду еще раз. Вы должны получить сообщение «The database sports is in use in single-user mode», подтверждающее

невозможность подключения к базе данных, пока с ней работает какая-либо программа в однопользовательском режиме.

- Теперь удалите из программы `single-pro.p` оператор `pause` и оператор `quit`. Запустите программу еще раз. По завершению работы программы управление будет передано в Procedure Editor, поэтому всегда используйте оператор `quit` в ABL-программах, которые должны запускаться из командной строки.

Помимо пользовательских ABL-программ в однопользовательском режиме работают такие стандартные утилиты как PROUTIL IDXBUILD, PROUTIL TRUNCATE BI и т.д. Это связанно с тем, что той работе, которую они выполняют в базе данных, не должны препятствовать другие пользователи. Более детальную информацию о возможных режимах работы стандартных утилит смотрите в стандартной документации к OpenEdge в описании интересующей утилиты.

Многопользовательский режим. Это режим доступа, позволяющий подключаться к базе данных, когда она активна, т.е. база стартована. В этом режиме к базе данных могут подключаться одновременно до 10000 пользователей (определяется настройкой параметра базы данных Number of Users <-n>). Для подключения используется либо скрипт mpro, либо утилита _progres. Как и в однопользовательском режиме у этих команд, кроме имени базы данных могут быть дополнительные параметры, которые не отличаются от параметров, используемых в однопользовательском режиме.

В многопользовательском режиме могут работать такие стандартные утилиты как PROUTIL DESCRIBE, PROUTIL DBANALYS, PROUTIL IOSTAT и т.д. Более детальную информацию о возможных режимах работы стандартных утилит смотрите в стандартной документации к OpenEdge в описании интересующей утилиты.

Помимо указанных режимов к базе данных можно подключать и выполнять программы в пакетном режиме (batch-mode). **Пакетный режим -** это режим, когда программа не выводит никакой информации на экран, не имеет никакого прямого интерфейса с пользователем и полностью выполняется без участия человека, по заранее определенному им сценарию. Для запуска программы в пакетном режиме используется параметр клиентской сессии Batch <-b>. Можно воспользоваться также скриптом mbpro из каталога $DLC/bin, которому нужно передать имя базы данных, имя программы и дополнительные параметры без указания параметра <-b>.

Команды pro, mpro, mbpro и _progres - это стандартные средства OpenEdge для работы с базой данных. Но, помимо выполнения ABL-программ, часто возникает необходимость в подключении к базам данных OpenEdge из сторонних, не ABL-приложений, например, MS Excel. Здесь на помощь приходит ODBC[22].

[22] ODBC - англ. Open DataBase Connectivity

SQL Client Access (ODBC)

Начиная с версии OpenEdge10.x драйвер ODBC для баз данных Progress OpenEdge - это бесплатный продукт, называемый SQL Client Access, который можно скачать с Progress® Download Center.

В этом разделе рассказано о том, как настроить сервер базы данных OpenEdge для подключения ODBC-клиентов, здесь так же приведен пример подключения к OpenEdge базе из MS Excel.

Настройка сервера базы данных для работы через ODBC

Создайте тестовую базу данных sports. Для этого создайте новый каталог в любом удобном месте на диске. Допустим, это домашний каталог, в котором мы создадим каталог ~/testdb:

```
mkdir ~/testdb
cd ~/testdb
```

Создайте в этом каталоге базу данных Sports:

```
procopy ./sports $DLC/sports
```

Запустите сервер базы данных, указав порт в качестве аргумента параметра <-S>, через который будет выполняться удаленный доступ к базе.

Лучше разделить брокеров базы данных на обслуживающих только ABL-клиентов и обслуживающих только SQL-клиентов. Но, если вы специально не будете делать разделение, то по умолчанию брокер на одном порту будет обслуживать как ABL, так и SQL-клиентов.

Запуск без разделения функций брокера:

```
proserve sports -S 60900
```

Запуск с разделением функций брокеров:

- Создайте три pf-файла.

  первый pf-файл:

```
# для обслуживания ABL клиентов
#my_m3abl.pf
-db ./sports
-m3
-S 60901
-ServerType ABL
-Ma 8
-Mi 5
-Mpb 2
```

  второй pf-файл:

```
# для обслуживания только SQL клиентов
```

```
#my_m3SQL.pf
-db ./sports
-m3
-S 60900
-ServerType SQL
-Ma 8
-Mi 5
-Mpb 2
```

третий pf-файл для основного брокера:

```
#my_pf.pf
-db ./sports
-n 50
-Mn 5
```

- Подготовьте скрипт для запуска базы данных:

```
#!/bin/sh
# start_test.sh
$DLC/bin/_mprosrv -pf ./my_pf.pf        # запуск основного брокера
$DLC/bin/_mprosrv -pf ./my_m3abl.pf     # запуск брокера для обслуживания ABL
клиентов
$DLC/bin/_mprosrv -pf ./my_m3SQL.pf     # запуск брокера для обслуживания SQL
клиентов
```

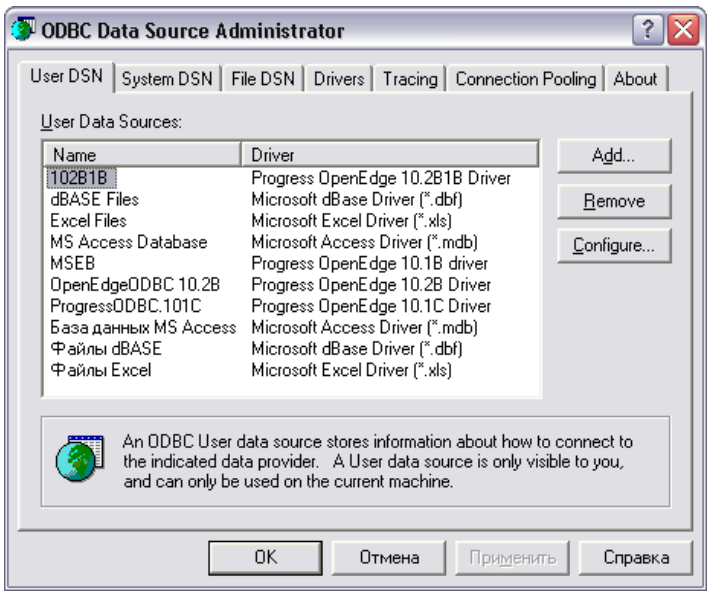
- Выполните скрипт start_test.sh для запуска базы данных.

```
./start_test.sh
```

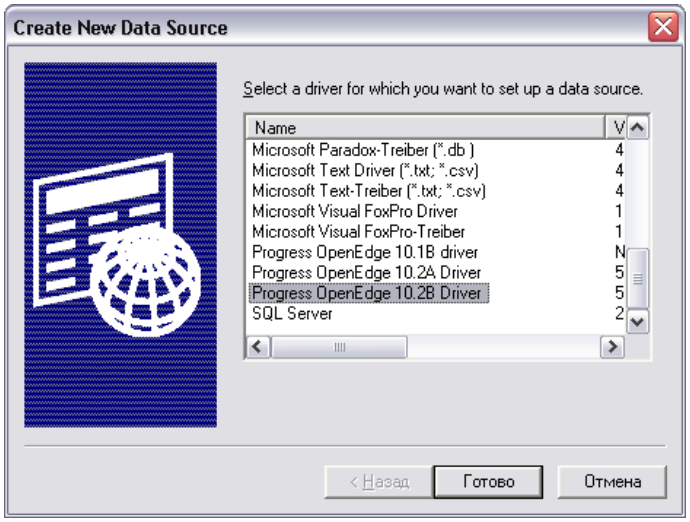
Теперь у нас база запущена с тремя брокерами, один из которых будет использоваться только для SQL (ODBC)-подключений.

Настройка ODBC в Windows

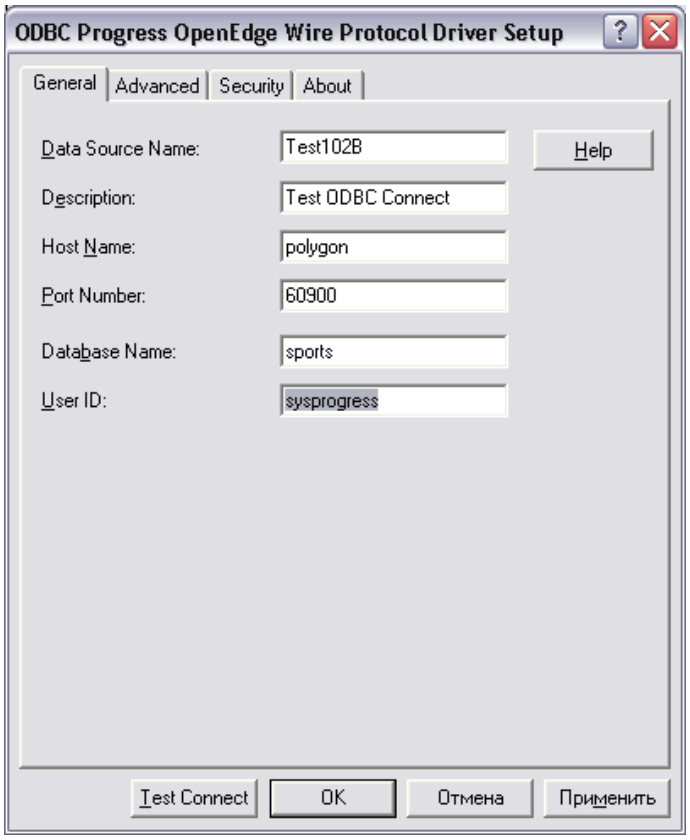
После установки SQL Client Access драйвер ODBC станет доступен в программе ODBC Data Source Administrator. Меню Windows: **Пуск → Настройка → Панель управления → Администрирование → Источники данных (ODBC)**. После запуска этой программы появится окно, в котором нужно добавить новый источник данных, нажав кнопку «Add»:



Выберите в появившемся окне соответствующий драйвер ODBC и нажмите кнопку «Готово»:



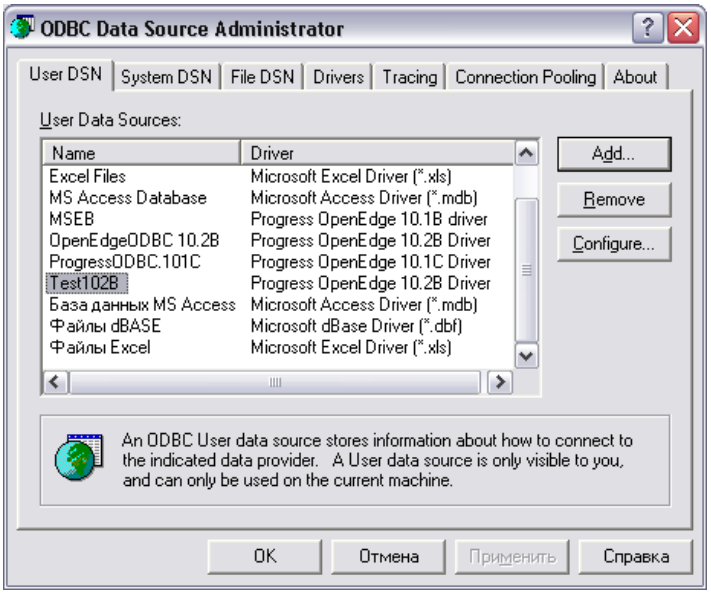
В окне настройки источника данных во вкладке General введите параметры подключения к базе данных:



В данном примере значение поля Host Name у вас может отличаться. Так же обратите внимание на то, что для подключения по ODBC необходимо имя пользователя, заранее зарегистрированного в базе данных и имеющему соответствующие SQL-привилегии. Используем имя пользователя sysprogress.

Проверьте правильность соединения, нажав кнопку «Test Connect», должно появиться сообщение:

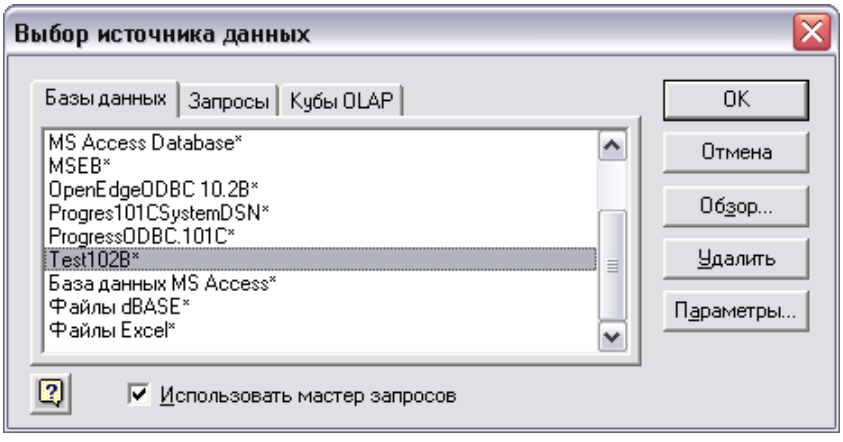
После этого нажмите кнопку «OK». Вы вернетесь к исходному окну, в котором в списке появится новый источник данных:



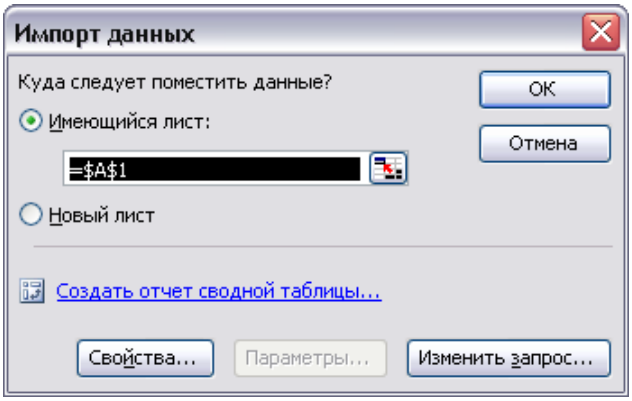
Новый источник данных создан. Нажмите кнопку «OK» для завершения работы с ODBC Data Source Administrator.

Проверка подключения в MS Excel

1. Откройте MS Excel.
2. Выберите верхнее меню: **Данные → Импорт внешних данных → Создать запрос**. В открывшемся окне выберите источник данных и нажмите кнопку «OK»:



3. В появившемся окне введите пароль пользователя и нажмите кнопку «ОК».
4. В окне «Создание запроса: выбор столбцов» выберите таблицу Customer и нажмите кнопку «Далее».
5. В окне «Создание запроса: отбор данных» если необходимо, укажите условия отбора данных и нажмите кнопку «Далее».
6. В окне «Создание запроса: порядок сортировки» если необходимо, выберите порядок сортировки данных и нажмите кнопку «Далее».
7. В окне «Создание запроса: заключительный шаг» оставьте все, как есть (по умолчанию) и нажмите кнопку «Готово».
8. В окне «Импорт данных» выберите, где будут размещены данные в Excel. В нашем случае оставьте все значения, представленные по умолчанию и нажмите кнопку «ОК»:



На текущем листе MS Excel появятся данные из таблицы Customer.

Глава 9 Изменение схемы базы данных

Изменение схемы - это одна из основных задач администратора базы данных. В течение всего жизненного цикла базы её схема подвергается различным модификациям, таким как добавление, удаление таблиц, полей, индексов и прочих объектов. Это связано с тем, что база данных постоянно развивается, изменяются приложения, работающие с ней, выполняется оптимизация схемы, добавляются новые механизмы и т.п. В этом разделе рассказано о том, как выполняется изменение схемы базы данных штатными средствами OpenEdge, описаны также некоторые нюансы, которые могут возникнуть в процессе изменения.

Таблицы

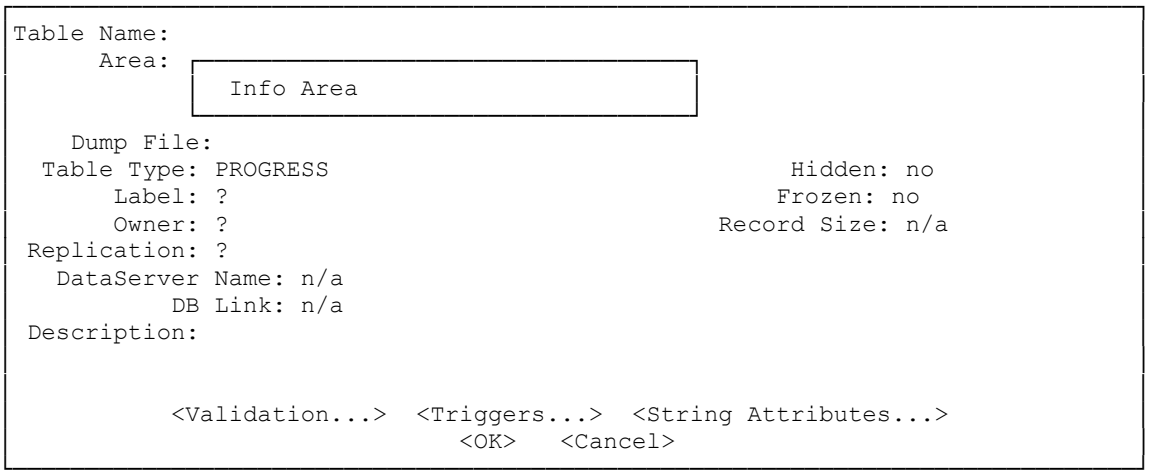
Добавление новой таблицы в схему базы данных выполняется с помощью интерфейса Data Dictionary.

- Подключитесь к базе данных:

```
mpro sports
```

  Откроется Procedure Editor.

- Нажмите клавишу <F3> для перехода в верхнее меню Procedure Editor и выберите пункт **Tools → Data Dictionary**. Откроется инструментарий OpenEdge Data Dictionary.
- Выберите пункт меню **Schema → Add New Table**:



  Откроется окно «Add New Table». Это окно содержит все необходимые поля для создания новой таблицы.

  - Table Name - название новой таблицы. Используйте название, например, TabTest1. Нажмите клавишу <Tab> для перехода к следующему полю.
  - Area - клавишами <↑> и <↓> выберите пользовательскую область, в которой будет размещена таблица. **Внимание!** Крайне не рекомендуется сохранять новые таблицы и индексы в области хранения Schema Area, т.к. это системная область, которая не может быть настроена для улучшения

производительности. Нажмите клавишу <Tab> для перехода к следующему полю.

- Dump File - автоматически подставляется название таблицы в нижнем регистре. Это поле используется для формирования имени файла выгрузки данных из таблицы. Обычно его значение оставляется по умолчанию. Перейдите к следующему полю.
- Hidden - используется для скрытия таблицы. Это означает, что таблица не будет видна в общем списке таблиц в Data Dictionary. Это свойство таблицы используется, как часть системы безопасности. Оставьте его значение по умолчанию, т.е. «no».
- Label - краткое название таблицы, определяющее таблицу, используется в различных сообщениях OpenEdge. Обычно оставляют значение, предложенное по умолчанию.
- Replication - указывается имя программы, которая должна выполнять репликацию этой таблицы. Оставьте его значение по умолчанию.
- Description - используется для детального описания назначения таблицы.

- Нажмите клавишу <F1> для завершения процесса создания таблицы. **Примечание**: более детальное описание возможностей окна «Add New Table» можно найти в документации «OpenEdge® Development Basic Database Tools».
- После нажатия клавиши <F1> откроется интерфейс добавления полей в таблицу.

```
-------------------------- Currently Defined Fields --------------------------


_____________________________________________________________________________
NextPage     PrevPage  Add          Modify  Delete  Copy  Triggers  View-As
StringAttrs  GoIndex   SwitchTable  Browse  Order   Undo  Exit
-----------------------------------------------------------------Total Fields: 0
```

В данном примере нет необходимости добавления полей, поэтому перейдите к пункту меню **Exit** и нажмите клавишу <Enter>.

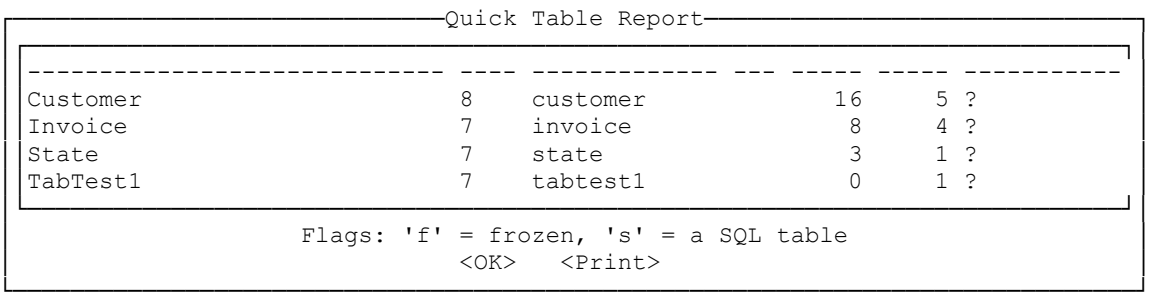
- Откроется следующее меню:

```
┌──────────────────────────┐
│  M. Modify Existing Table│
│  F. Field Editor         │
│  I. Index Editor         │
│  S. Sequence Editor      │
│  Q. Adjust Field Width   │
├──────────────────────────┤
│  A. Apply Changes        │
│  U. Undo Changes         │
└──────────────────────────┘
```

На этом этапе можно либо завершить создание таблицы, сохранив внесенные изменения, либо продолжить её создание, выбрав соответствующий пункт меню. **Внимание!** В зависимости от структуры таблицы процесс её добавления в схему может занять некоторое время.

- Выберите пункт меню **Apply Changes**. Таблица будет добавлена в схему базы данных.

Проверить, была ли добавлена таблица в базу, можно с помощью пункта меню Data Dictionary: **Database → Reports → Quick Table**. Откроется отчет в следующем виде:



Для изменения описания таблицы используется пункт меню **Schema → Modify Table**. Откроется список всех доступных пользовательских таблиц в схеме базы данных. В этом списке необходимо выбрать интересующую таблицу и нажать клавишу <Enter>, после чего откроется окно «Modify Table», полностью идентичное окну «Add New Table».

Для удаления описания таблицы из схемы базы данных используется меню **Schema → Delete Table(s)**. Здесь можно выбрать несколько таблиц для удаления. **Внимание!** Процесс удаления может занять достаточно много времени, если таблица содержит данные, и может привести к чрезмерному росту размера Before-Image-файла, поэтому перед удалением описания таблицы рекомендуется предварительно удалить из неё данные. Пример ABL-кода для удаления записей из таблицы:

```
DEFINE VARIABLE i AS INTEGER NO-UNDO.
FIND FIRST Customer NO-ERROR.
REPEAT WHILE AVAILABLE Customer:
  DO TRANSACTION:
    DO i = 1 TO 100 WHILE AVAILABLE Customer:
      DELETE Customer.
      FIND NEXT Customer NO-ERROR.
    END.
  END.
END
```

Если таблица находится в области с типом SAT-II, и эта таблица одна в этой области, то можно применить команду PROUTIL TRUNCATE AREA. Это более быстрый способ удаления данных, чем удаление с помощью оператора DELETE. Один небольшой недостаток усечения области - база данных должна быть остановлена. Команда усечения следующая:

proutil *db-name* -C truncate area *area-name*

Если база данных запущена, то наиболее быстрым способом очистки таблицы будет её переименование и последующее постепенное удаление данных из неё с помощью ABL-кода, при этом предварительно можно деактивировать все индексы таблицы, кроме первичного индекса.

Детальное описание интерфейсов добавления, изменения и удаления таблицы в Data Dictionary можно найти в стандартной документации «OpenEdge® Development Basic Database Tools» », в главе «Table Tasks».

Поля

Для добавления, изменения и удаления полей таблицы используется пункт меню Data Dictionary: **Schema → Field Editor**. Здесь необходимо выбрать таблицу, поля которой нужно редактировать.

- Выберите созданную ранее таблицу TabTest1 и нажмите клавишу <Enter>. Откроется окно «Field Editor».
- Для добавления, изменения и удаления полей используются пункты меню **Add**, **Modify**, **Delete**, которые открывают следующую форму с описанием поля:

```
_______________________________________________________________________--
Field-Name:                                         Data-Type:
Format:                                            Extent:
Label:                                          Decimals:
Column-label:                                             Order:
Initial:                                         Mandatory:      (Not Null)
Component of-> View:      Index:      Position:        Case-sensitive:
Valexp:

Valmsg:
Help:
Desc:
_________________________________________________________________________
NextPage     PrevPage   Add          Modify   Delete   Copy   Triggers   View-As
StringAttrs  GoIndex    SwitchTable  Browse   Order    Undo   Exit
------------------------------------------------------------Total Fields: 0
```

Детальное описание интерфейса добавления, изменения и удаления полей в Data Dictionary можно найти в стандартной документации «OpenEdge® Development Basic Database Tools», в главе «Field Tasks».

Индексы

Для работы с индексами таблицы используется пункт меню Data Dictionary: **Schema → Index Editor**. Здесь необходимо выбрать таблицу, в которой необходимо редактировать индексы.

- Выберите созданную ранее таблицу TabTest1 и нажмите клавишу <Enter>. Откроется окно «Index Editor»:

```
Name: default                                PRIMARY Non-Unique ACTIVE
Area:
       Schema Area

  default                    Seq Field Name                 Type         Asc  Abbr

                               1                            character    Asc  No

  Next  Prev  First  Last  Rename  Add  Delete  ChangePrimary  Uniqueness
  MakeInactive  Browse  SwitchTable  GoField  ROWID  Undo  Exit
```

С помощью этого пункта меню можно выполнять с индексами следующие операции: добавлять, переименовывать, удалять, деактивировать, изменять первичный индекс.

Детальное описание интерфейса редактирования индексов в Data Dictionary можно найти в стандартной документации «OpenEdge® Development Basic Database Tools», в главе «Index Tasks».

Внимание! Если необходимо изменить индекс в таблице с большим объемом данных, то новый индекс рекомендуется добавлять неактивным. Это связанно с тем, что во время добавления описания активного индекса будет одновременно выполнено построение этого индекса, что в свою очередь займет, в зависимости от размера таблицы, много времени, а так же может привести к значительному снижению производительности базы данных, если она запущена. Поэтому лучше сразу деактивировать индекс, и активировать его позже, в наименее загруженное время одной из следующих команд:

- в offline:

```
proutil db-name -C idxbuild
```

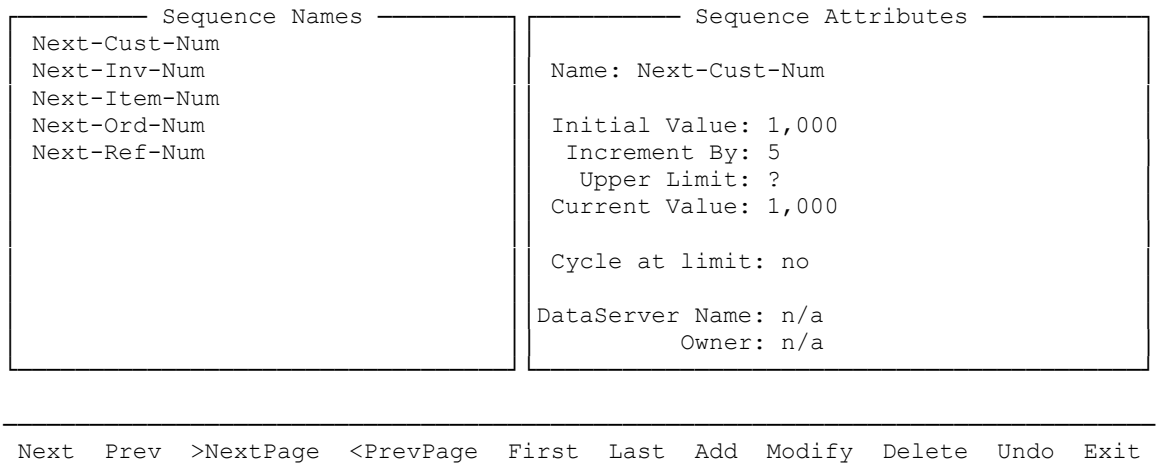
- в online:

```
proutil db-name -C idxactivate
```

Детальное описание этих команд можно найти в документации «OpenEdge® Development Database Administration».

Секвенции

Для создания в базе данных секвенций или, простыми словами, автоматических счетчиков, используется пункт меню **Schema → Sequence Editor**. Откроется окно «Sequence Editor», в котором будет представлен список имеющихся счетчиков и их свойства:



С помощью пунктов меню **Add**, **Modify** и **Delete** в этом окне можно добавлять новые и редактировать старые счетчики.

Детальное описание интерфейса редактирования секвенций в Data Dictionary можно найти в стандартной документации «OpenEdge® Development Basic Database Tools», в главе « Sequence Tasks».

Триггеры

Триггеры - это отдельные процедуры, которые выполняются автоматически, реагируя на определенные события в таблице базы данных. Такими событиями могут быть создание, изменение и удаление записи из таблицы. В OpenEdge триггерные программы хранятся отдельно от базы данных в виде обычных ABL-процедур, которые могут быть как в виде исходного ABL-кода, так и в скомпилированном виде. Безопасность триггеров от несанкционированного изменения их программ обеспечивается проверкой CRC[23]-файла программы, связанной с каждым триггером. Опция проверки CRC установлена по умолчанию для всех триггеров, она необязательная, по желанию может быть отключена в настройках триггера.

В OpenEdge существует два типа триггеров - табличные триггеры и триггеры полей.

Табличные триггеры настраиваются на следующие события:

- CREATE
- DELETE
- FIND
- WRITE
- REPLICATION-CREATE
- REPLICATION-DELETE
- REPLICATION-WRITE

Для создания табличных триггеров используется пункт меню Data Dictionary: **Schema → Modify Table**. В открывшемся окне «Modify Table» необходимо клавишей <Tab> перейти на кнопку <Triggers...> и нажать клавишу <Enter>. Откроется экран Table Triggers.

```
┌──────────────────────────────────Table Triggers──────────────────────────────┐
│                                                      Override-  Check       │
│                        Trigger Programs              able?      CRC?        │
│                                                                             │
│             For CREATE: sports/crcust.p              no         no          │
│             For DELETE: sports/delcust.p             no         no          │
│               For FIND:                             no         yes         │
│              For WRITE: sports/wrcust.p              no         no          │
│ For REPLICATION-CREATE:                              no         yes         │
│ For REPLICATION-DELETE:                              no         yes         │
│  For REPLICATION-WRITE:                              no         yes         │
│           (removing the trigger file name deletes the trigger)              │
│                    (press [F6] to edit trigger program)                     │
│                                                                             │
│                              <OK>   <Cancel>                                │
└─────────────────────────────────────────────────────────────────────────────┘
```

Для создания триггера необходимо установить курсор в поле соответствующего события, ввести имя программы и нажать клавишу <F6> для редактирования кода этой программы.

Далее приведены примеры ABL-программ для различных триггеров.

[23] CRC - от англ. Cyclic redundancy code, или циклический избыточный код

Создание записи в таблице Customer с автоматическим присваиванием порядкового номера клиенту:

```
TRIGGER PROCEDURE FOR Create OF Customer.
   ASSIGN Customer.Cust-Num = NEXT-VALUE(Next-Cust-Num).
```

Удаление записи из таблицы Order для обеспечения целостности базы данных должно повлечь за собой удаление информации из дочерней таблицы Order-line:

```
TRIGGER PROCEDURE FOR Delete OF Order.

MESSAGE "Deleting Order" Order-Num VIEW-AS ALERT-BOX INFORMATION BUTTONS OK.
FOR EACH Order-Line OF Order:
   DELETE Order-Line.
END.
```

Если клиенту был изменен порядковый номер, то выполняется поиск всех связанных с ним счетов, которым устанавливается новый номер клиента:

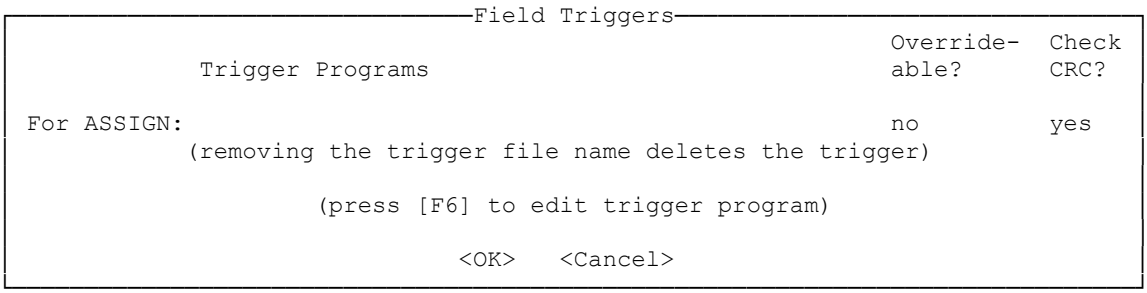
```
TRIGGER PROCEDURE FOR Write OF Customer OLD BUFFER oldCustomer.

DEFINE VARIABLE i AS INTEGER INITIAL 0.
DEFINE VARIABLE Outstanding AS INTEGER INITIAL 0.

IF Customer.Cust-Num <> oldCustomer.Cust-Num AND oldCustomer.Cust-Num <> 0
THEN DO:

   FOR EACH order OF oldCustomer:
      Order.Cust-Num = Customer.Cust-Num.
      i = i + 1.
   END.
   IF i > 0 THEN
     MESSAGE i "orders changed to reflect the new customer number!"
          VIEW-AS ALERT-BOX INFORMATION BUTTONS OK.
END.
```

Выполнение репликации вновь созданной записи в таблице Customer в таблицу Replication-Log:

```
TRIGGER PROCEDURE FOR REPLICATION-CREATE OF Customer.

CREATE Replication-Log.
ASSIGN
   Replication-Log.Taskid = DBTASKID(LDBNAME(BUFFER Replication-Log))
   Replication-Log.Table = 'Customer'
   Replication-Log.Action = 'CREATE'.
   RAW-TRANSFER Customer TO Replication-Log.Record.
```

Внимание! Помните, при наступлении событий CREATE, WRITE сначала срабатывают обычные триггеры на эти события, и только затем срабатывают репликационные триггеры, а при наступлении события DELETE сначала срабатывает репликационный триггер, после чего и обычный триггер.

Триггеры полей реагируют на одно единственное событие - ASSIGN, т.е. на изменение значения поля. Для установки триггера на поле необходимо в Data Dictionary выбрать пункт меню **Schema → Field Editor**, и в окне «Field Editor» выбрать меню **Triggers**. После этого будет предоставлена возможность выбрать поле таблицы, для которого необходимо создать триггер. После выбора поля и нажатия клавиши <Enter> или клавиши <F1> откроется окно «Field Triggers»:



Принцип создания триггера здесь такой же, как и при создании табличного триггера.

Пример ABL-кода триггера для поля State таблицы Customer:

```
TRIGGER PROCEDURE FOR Assign OF Customer.State.
   IF Customer.Country = "USA" AND NOT (CAN-FIND(State OF Customer))
   THEN DO:
      MESSAGE "Illegal State name for the U.S.A."
            VIEW-AS ALERT-BOX INFORMATION BUTTONS OK.
      RETURN ERROR.
    END.
```

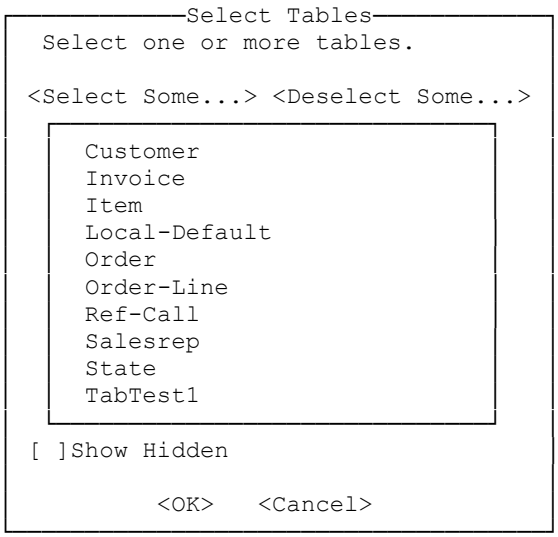
В данном примере триггер проверяет правильность ввода названия штата для клиента из США.

Все приведенные примеры ABL-программ для триггеров взяты из каталога $DLC/sports, и используются в тестовой базе данных Sports.

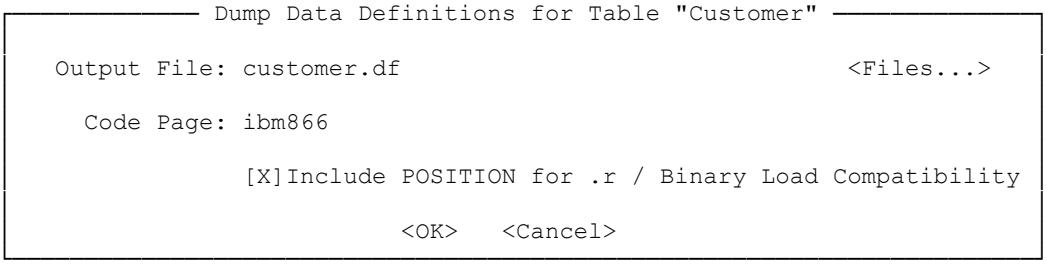
Детальное описание триггеров в OpenEdge можно найти в стандартной документации «OpenEdge® Development Basic Database Tools», в главе «Trigger Tasks», а так же в документации «ABL Triggers and Indexes».

Выгрузка и загрузка готовой схемы базы данных

Как уже говорилось ранее, в течение всего жизненного цикла базы данных её схема подвергается различным изменениям. Не всегда эти изменения вносятся сразу в промышленную версию базы, обычно предварительно разработка и тестирование этих изменений выполняется в базах данных разработчиков и в тестовых базах, являющихся точной копией промышленной базы. После завершения процесса разработки и тестирования готовые изменения в схеме необходимо перенести на промышленную базу. Хорошо, если это только добавление одного поля, тогда такое изменение можно выполнить вручную с помощью соответствующего пункта меню Data Dictionary, а если таких изменений много, или вообще нужно перенести всю схему в новую базу данных? В таких случаях на помощь приходят файлы описания данных, или Data Definitions Files. Файлы описания данных имеют расширение .df и содержат описание всех таблиц, полей, индексов секвенций и прочих объектов базы данных, что значительно облегчает перенос схемы из одной базы в другую. Создать файл описания данных можно как для всей схемы базы данных, так и для конкретной таблицы. Для этого в Data Dictionary используется следующий пункт меню **Admin → Dump Data and Definitions → Data Definitions (.df file)**. В открывшемся окне «Select Tables» можно выбрать одну или несколько таблиц для выгрузки, либо все таблицы, имеющиеся в базе данных:



После выбора одной таблицы имя df-файла будет состоять из значения, определенного в поле Dump File в настройках таблицы, если же будет выбрано несколько таблиц или все, то имя df-файла сформируется из имени базы данных. В обоих случаях расширение файла описания данных будет df. Эти имена предлагаются по умолчанию, по желанию имя df-файла можно изменить в окне «Dump Data Definitions for Table», которое появится сразу после того, как в окне «Select Tables» будет нажата кнопка <OK>:



По умолчанию df-файл формируется в текущем рабочем каталоге пользователя, но этот путь можно изменить, воспользовавшись кнопкой <Files...>.

После нажатия кнопки <OK> сформируется текстовый df-файл, содержащий описание таблицы, который при необходимости можно редактировать. Пример части df-файла для таблицы Customer:

```
ADD TABLE "Customer"
  AREA "Customer/Order Area"
  DESCRIPTION "Customer information"
  DUMP-NAME "customer"
  TABLE-TRIGGER "CREATE" NO-OVERRIDE PROCEDURE "sports/crcust.p" CRC "?"
  TABLE-TRIGGER "DELETE" NO-OVERRIDE PROCEDURE "sports/delcust.p" CRC "?"
  TABLE-TRIGGER "WRITE" NO-OVERRIDE PROCEDURE "sports/wrcust.p" CRC "?"

ADD FIELD "Cust-Num" OF "Customer" AS integer
  FORMAT ">>>>9"
  INITIAL "0"
  POSITION 2
  MAX-WIDTH 4
  VALEXP "cust-num > 0"
  VALMSG "Customer number must be greater than zero"
  ORDER 10
***
ADD INDEX "Comments" ON "Customer"
  AREA "Customer Index Area"
```

```
  WORD
  INDEX-FIELD "Comments" ASCENDING

UPDATE INDEX "Comments" OF "Customer"
  BUFFER-POOL "Alternate"
                                        ***
.
PSC
bufpool=yes
cpstream=ibm866
.
```

Для загрузки df-файла в базу данных в Data Dictionary используется пункт меню **Admin → Load Data and Definitions → Data Definitions (.df file)**. Здесь нужно указать расположение загружаемого df-файла и запустить процесс загрузки. Продолжительность загрузки по времени зависит от количества загружаемых описаний, а так же от настроек областей хранения, в которые эти описания загружаются.

Внимание! Если в файле описания данных программой загрузки будет обнаружена хотя бы одна ошибка, то весь процесс загрузки будет остановлен, а уже загруженные описания данных будут отменены.

Помимо полной выгрузки описания схемы в Data Dictionary существует возможность создания инкрементального df-файла. Инкрементальный df-файл - это файл, содержащий различия в описании схем между двумя базами данных. Такой файл можно использовать для синхронизации схемы промышленной базы со схемой базы данных разработчиков.

Процесс создания инкрементального df-файла.

- Подключитесь к двум базам данных, к базе разработчиков и к промышленной базе. Для этого воспользуйтесь пунктом меню **Database → Connect** в Data Dictionary.
- Выберите в качестве текущей рабочей базы данных базу данных разработчиков в пункте меню **Database → Select Working Database**, т.к. именно изменения в этой базе должны быть перенесены в промышленную базу.
- Выберите пункт меню **Admin →□Dump Data and Definitions →□Create Incremental .df File**. В открывшемся окне в поле Create Incremental Definitions File будет отображено имя промышленной базы данных.
- Нажмите клавишу <Enter> или клавишу <F1>. В открывшемся окне будет предложено изменить имя инкрементального df-файла, по умолчанию это имя равно delta.df.
- Нажмите клавишу <F1Ю для запуска процесса сравнения баз данных и формирования инкрементального df-файла. По завершению процесса управление будет передано в основное меню.
- Загрузите полученный инкрементальный df-файл в промышленную базу данных.

Внимание! Если в процессе сравнения баз данных обнаружится, что в поле таблицы был изменен тип данных, то в df-файл будет сначала записан оператор DROP для удаления этого поля из промышленной (целевой) базы данных, а потом оператор ADD для создания этого поля с новым типом данных. Когда поле будет удалено, то все данные, которые оно содержало, также будут удалены, поэтому перед загрузкой инкрементального df-файла необходимо позаботиться о сохранности этих данных.

Детальное описание файлов описания данных можно найти в стандартной документации «OpenEdge® Data Management Database Administration», в главе «Dumping ABL database definitions».

Глава 10 Безопасность

Администратор базы данных должен обеспечить её безопасность, исключив возможность несанкционированного подключения, изменения и удаления хранящихся в ней данных. О том, как обеспечить безопасность базы данных и пойдет речь в этом разделе.

Внимание! Перед любыми изменениями в настройках безопасности базы данных обязательно сформируйте резервную копию, т.к. в случае наступления непредвиденной ситуации, возможно, что управлять базой вы больше не сможете, например, при ошибке с правами доступа администратора. В этом случае восстановить контроль над базой можно только при восстановлении базы из резервной копии.

Аутентификация

Для ограничения доступа к базе данных на чтение и изменение данных необходимо, чтобы каждый пользователь, подключающийся к ней, использовал собственный логин и пароль. При этом права доступа пользователя на объекты базы данных и разрешенные действия с ними должны быть соответствующим образом настроены. Регистрацией пользователей и настройкой их прав доступа занимается администратор базы данных, учетная запись которого также должна быть предварительно настроена.

Регистрация пользователя

Любое соединение с базой данных ассоциируется с ID пользователя. Каждый уровень безопасности использует ID пользователя для определения прав его доступа.

На самом первом уровне определяется разрешение пользовательских подключений в соответствии со списком авторизованных ID. Этот уровень - уровень безопасности соединения. Каждый ID пользователя может быть защищен паролем для исключения возможности несанкционированного использования этого ID.

Если список разрешенных пользователей установлен не будет, то база данных унаследует ID пользователя от текущего подключения или возьмет его из операционной системы.

В Unix-системах база данных может использовать ID пользователя операционной системы. В этом случае ID пользователя операционной системы должен соответствовать правилам, установленным для ID пользователя OpenEdge Progress.

Для внесения списка разрешенных ID в базу данных необходимо внести информацию о каждом пользователе, сделать это следует в меню **Admin → Security → Edit User List** инструментального средства Data Dictionary. База данных хранит этот список в системной таблице _user, которая скрыта от обычных пользователей. После этого при подключении к базе данных необходимо обязательно указать ID и пароль пользователя. Как только в список будет добавлен хотя бы один пользователь, база данных больше не сможет наследовать и использовать ID операционной системы.

В диалоговом режиме пользователю будет предложен соответствующий интерфейс для ввода ID и пароля. При подключении из командной строки или с помощью оператора CONNECT необходимо использовать параметры клиентской сессии User ID <-U> и Password <-P>. Установить ID пользователя можно так же с помощью функции SETUSERID из собственной ABL-программы.

Пример подключения из командной строки:

```
mpro sports -U admin -P ""
```

Пример установки пользователя из ABL-программы:

```
DEFINE VARIABLE id       LIKE _User._Userid.
DEFINE VARIABLE password LIKE _Password.
DEFINE VARIABLE tries    AS INTEGER NO-UNDO.

IF USERID("DICTDB") <> "" OR NOT CAN-FIND(FIRST DICTDB._User)
THEN RETURN.

DO ON ENDKEY UNDO, RETURN:
  id = "".
  password = "".
  UPDATE SPACE(2) id SKIP  password BLANK
    WITH CENTERED ROW 8 SIDE-LABELS ATTR-SPACE
    TITLE " Database " + LDBNAME("DICTDB") + " ".
  IF NOT SETUSERID(id,password,"DICTDB") THEN DO:
    MESSAGE "Sorry, userid/password is incorrect.".
    IF tries > 1 THEN QUIT.   /* Only allow 3 tries */
    tries = tries + 1.
    UNDO, RETRY.
  END.
  HIDE ALL.
  RETURN.
END.
QUIT.
```

ID пользователя - это набор символов, подчиняющийся следующим правилам:

- максимум 12 символов;
- содержит любые печатные символы за исключением символов #, *,! и @;
- ID не чувствителен к регистру;
- ID может быть пустой.

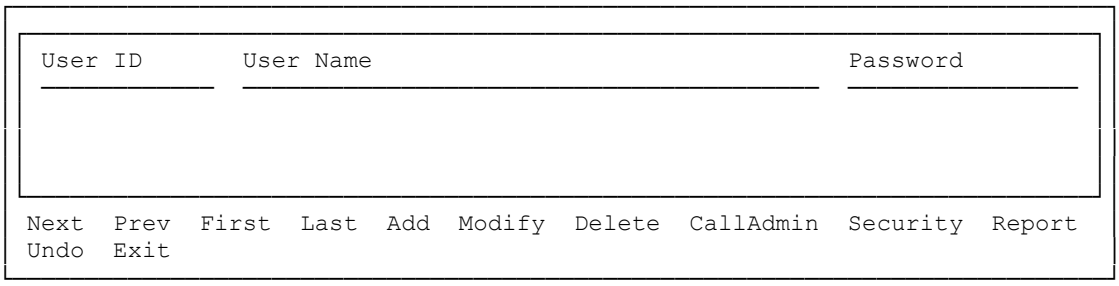
Пароль может состоять не более чем из 16 символов. При установке пароль кодируется с использованием функции ENCODE. Пароль чувствителен к регистру, т.к. функция ENCODE возвращает различные значения для символов верхнего и нижнего регистров.

Процесс регистрации ID пользователя в базе данных.

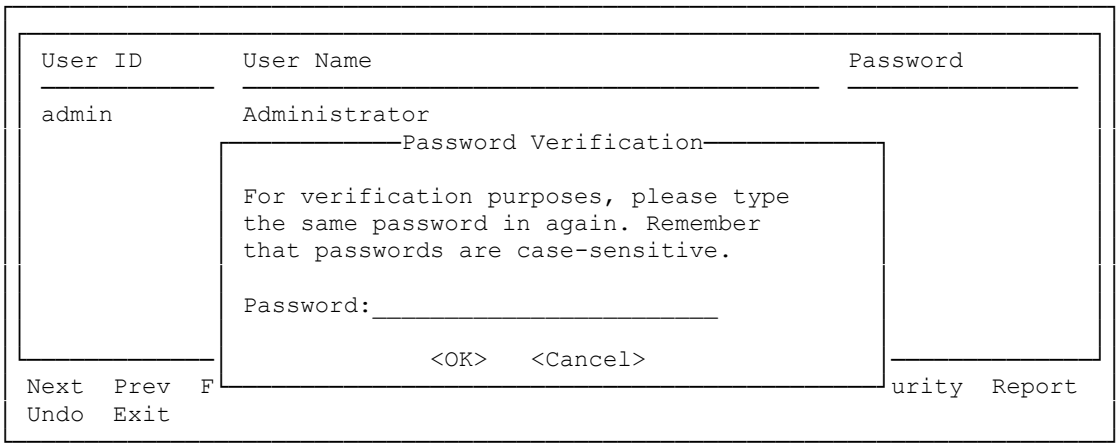
- Подключитесь к базе данных:

```
mpro sports
```

- В Data Dictionary выберите меню **Admin –> Security → Edit User List**. Откроется окно настройки списка пользователей «Edit User List»:



- Выберите пункт меню **Add** и нажмите клавишу <Enter>. В области списка пользователей добавится строка, доступная для редактирования. Заполните поле User ID, User Name и Password, обратите внимание, что при вводе пароля символы не отображаются, и когда вы нажмете клавишу <Enter> или клавишу <F1> на этом поле, система предложит повторить пароль для подтверждения правильности его ввода.



Повторите ввод пароля и нажмите кнопку <OK>.

Примечание: нельзя изменить User ID и Password в окне «Edit User List», здесь можно только добавлять и удалять пользователей. Тем не менее, изменить ID и пароль можно путем удаления записи о пользователе и создания её заново. Изменять без удаления можно только User Name.

После добавления пользователя в список, он сможет подключиться к базе данных только с помощью установленного вами пароля. Впоследствии пользователь может сменить пароль на свой собственный.

Если пользователь забыл свой пароль, то установить ему новый пароль можно только удалив его запись из списка пользователей и создав её заново.

Настройка учетной записи Администратора

После создания списка пользователей необходимо выделить из них хотя бы одного администратора безопасности, а лучше нескольких. Администратор безопасности отвечает за обеспечение безопасности информации в базе данных. Обычно администратор базы данных одновременно является и администратором безопасности. База данных OpenEdge позволяет определять администратора безопасности, только если вы

подключились к базе не под «пустым» пользователем (Blank User ID). Вы можете определить себя или других пользователей в качестве администратора безопасности.

Только администраторы безопасности имеют доступ к меню **Security**:

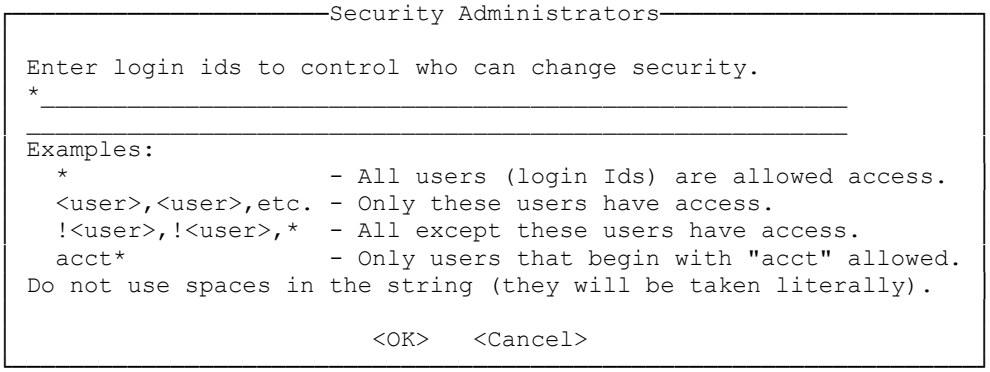
- Edit User List
- Change/Display Data Security
- Security Administrators
- Disallow Blank User Id Access

Остальные пользователи имеют доступ только к меню **Change Your Password** и **Quick User Report**. Установка администратора безопасности не ограничивает других пользователей в создании таблиц и полей в базе данных.

Определить администратора безопасности можно в диалоговом окне «Security Administrators». Доступ к диалоговому окну можно получить по следующему пути в меню Data Dictionary: **Admin → Security→ Security Administrators**.

Порядок ввода ID администратора безопасности:

- Выберите в Data Dictionary пункт меню **Admin → Security → Security Administrators**. Появится диалоговое окно «Security Administrators»/



- Введите ID пользователя. Здесь вы можете ввести несколько ID пользователей, но вы должны обязательно добавить свой собственный ID. В противном случае вы получите сообщение «You cannot change a security field to exclude yourself».

  Для разделения ID используйте запятую. Если же вы используйте пробел в строке, все ID будут восприняты как часть одного ID. Формат возможных вариантов списка пользователей описан непосредственно в окне «Security Administrators».

- Когда ввод будет завершен, нажмите кнопку <OK>. Вам будет предложено проверить введенные данные.

- Нажмите <Yes> для сохранения или <No> для возврата в окно «Security Administrators». Если выберете <Yes>, вы выйдете в основное меню, а указанные ID будут сохранены в базе данных.

Внимание! Помните, если по каким либо причинам вы забудете пароль администратора, то вы навсегда потеряете возможность внесения административных изменений в базу данных. Чтобы этого не произошло, желательно указывать минимум два ID для установки административных привилегий, а перед любыми подобными изменениями всегда делайте резервную копию базы данных.

Изменение пароля пользователю

Для изменения своего пароля пользователю не нужно привилегий администратора безопасности. Для этого он должен сделать следующее:

- выбрать пункт меню в Data Dictionary: **Admin → Security → Change Your Password$**
- ввести новый пароль, при этом помнить, что пароль чувствителен к регистру, и повторить ввод для проверки.

Администратор безопасности может сменить пароль другому пользователю только удалив его запись из списка пользователей и создав её заново с новым паролем.

Удаление пользователя

Только администратор безопасности может удалять пользователя из списка. Для удаления необходимо выполнить следующее.

- Выбрать в Data Dictionary пункт меню: **Admin → Security → Edit User List**. Откроется диалоговое окно «Edit User List».
- Выбрать пользователя для удаления и нажать меню **Delete**. Будет задан вопрос о подтверждении удаления. **Примечание!** Нельзя удалить свою собственную запись, пока все остальные записи не будут удалены.
- Проверить, что запись была удалена.

Если вы удалите всех пользователей из списка, будет выведено диалоговое сообщение о необходимости удалить все ограничения безопасности для базы данных. Если вы выберите удаление, то все пользователи, подключающиеся к базе данных, получат права администратора безопасности. Если выберите <NO>, вы должны добавить одного или нескольких пользователей в список пользователей, прежде чем выйдите в основное меню.

«Заморозка» таблиц

Правильная настройка схемы безопасности гарантирует, что только авторизованные пользователи могут изменять описание таблиц, полей и индексов в базе данных. Для блокировки или, другими словами, «заморозки» описания таблиц, полей и индексов в базе данных используется меню **Utilities → Freeze/Unfreeze** из Data Dictionary. После того, как таблица базы будет заморожена, ни один пользователь не сможет внести изменения в её описание. Администратор базы может разморозить такую таблицу для внесения изменений в нее.

Для замораживания и размораживания таблиц сделайте следующее.

- Выберите в Data Dictionary пункт меню **Utilities → Freeze/Unfreeze**, появится список таблиц, имеющихся в базе данных, отсортированный в алфавитном порядке.
- Выберите необходимую таблицу в диалоговом окне «Freeze/Unfreeze Table».
- Определите статус таблицы и нажмите кнопку <OK>.

Таким образом, вы как администратор базы данных, обеспечиваете защиту от несанкционированного изменения ее схемы.

Только обычной заморозки таблицы недостаточно, поскольку доступ к меню **Freeze/Unfreeze** имеет любой пользователь, который зарегистрирован в базе данных. Для того, чтобы другие пользователи не могли разморозить таблицу, у них не должно быть прав доступа на запись (can-write) в системную таблицу _File или её поле _Frozen.

Установка ограничений доступа к таблицам и полям базы данных

С помощью Data Dictionary можно изменять права доступа к таблицам и полям базы данных. Права на такие изменения имеет только администратор безопасности, все остальные пользователи при попытке получить доступ к этой функции получат сообщение «You must be a Security Administrator to execute this function» при условии, что учетная запись администратора безопасности соответствующим образом настроена.

Для настройки прав доступа к таблицам и полям базы данных необходимо выполнить следующее.

- Войти в Data Dictionary.
- Выбрать меню **Admin –> Security → Edit Data Security**. Откроется окно со списком таблиц базы данных, отсортированных в алфавитном порядке.
- Выбрать нужную таблицу и нажать клавишу <Enter>.

Откроется окно для редактирования прав доступа к таблице. Если необходимо установить права доступа к полю, то воспользуйтесь меню **NextField**, **PrevField** или **JumpField**. Для изменения прав доступа используйте меню **Modify**.

В таблице приведено назначение полей прав доступа.

Ограничение доступа	Описание
Can-Read	Определяет пользователей, у которых есть разрешение читать таблицу.
Can-Write	Определяет пользователей, которые могут писать в таблицу или изменять записи.
Can-Create	Определяет пользователей, которые могут создавать новые записи. Пользователь, который имеет право Can-Create, автоматически получает право Can-Write.
Can-Delete	Определяет пользователей, которые могут удалять записи из таблицы.
Can-Dump	Определяет пользователей, которые могут выгружать схему базы данных или отдельных таблиц и их данные.
Can-Load	Определяет пользователей, которые могут загружать схему базы данных или описание отдельных таблиц и их данные.

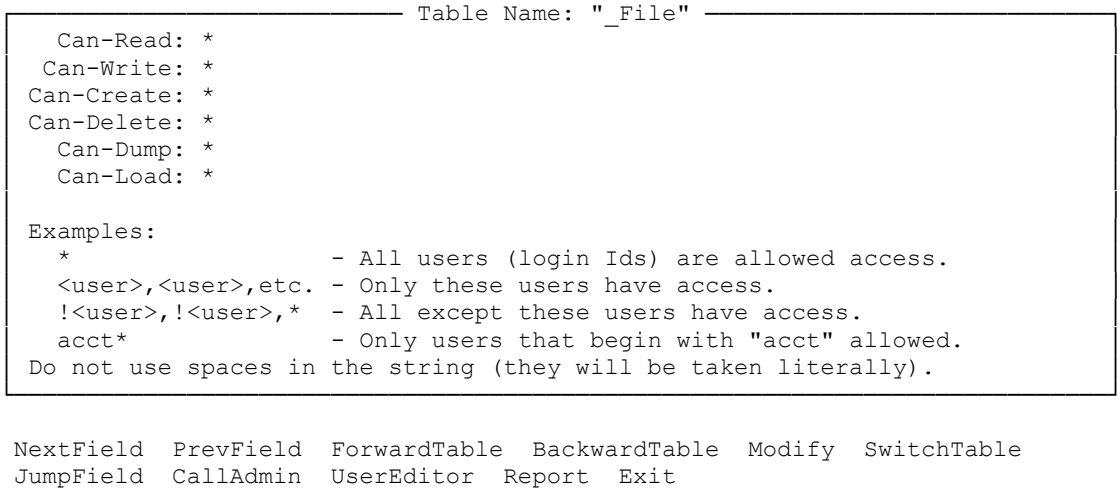
В каждом поле перечисляются ID пользователей, которые должны быть разделены запятой. В таблице приведен формат для ограничения доступа, используемый в полях доступа.

Выражение	Значение
*	Все пользователи имеют доступ.
user	Только один указанный пользователь имеет доступ к таблице.
!user	Доступ к таблице имеют все пользователи, кроме указанного.
string*	ID пользователей, которые начинаются с указанных символов имеют доступ к таблице.

Когда вы меняете права доступа, то необходимо помнить, что они вступят в силу только для новых подключений. На текущие подключенные сессии, включая вашу сессию, права действовать не будут. Для этого нужно отключиться от базы OpenEdge и подключиться к ней снова.

Рассмотрим пример установки ограничения доступа к системной таблице _File и к её полю _Frozen. Мы должны добиться того, чтобы доступ на замораживание и размораживание таблицы остался только у администратора базы данных.

- Зайдите в базу данных, используя учетную запись администратора безопасности.
- Выберите в Data Dictionary пункт меню **Admin → Security → Edit Data Security**. Откроется список имеющихся пользовательских таблиц. Нам нужна системная таблица _File, поэтому в поле Table Name вручную введите название интересующей таблицы и нажмите клавишу <Enter> или <F1>. Откроется окно изменения прав доступа в выбранной таблице.

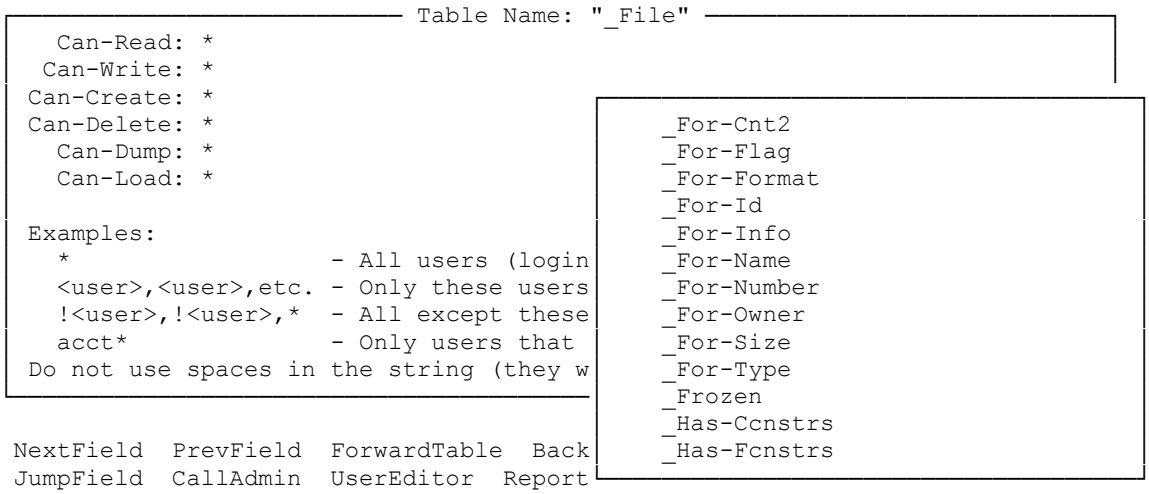


  Окно состоит из полей для ограничения прав доступа. По умолчанию доступ к таблице имеют все пользователи, т.к. в полях доступа установлен символ звездочки.

- Выберите меню **Modify** для редактирования необходимых нам полей. Как только поля станут активными, с помощью клавиши <Tab> перейдите на поле Can-Write. **Внимание!** Не нажимайте никакие другие клавиши кроме <Tab>, пока не выберите нужное поле, в противном случае редактирование поля будет невозможным.
- Введите вместо символа звездочки ID администратора безопасности и нажмите клавишу <F1>. Изменение сохранено, теперь права доступа на запись в таблицу _File есть только у администратора безопасности.

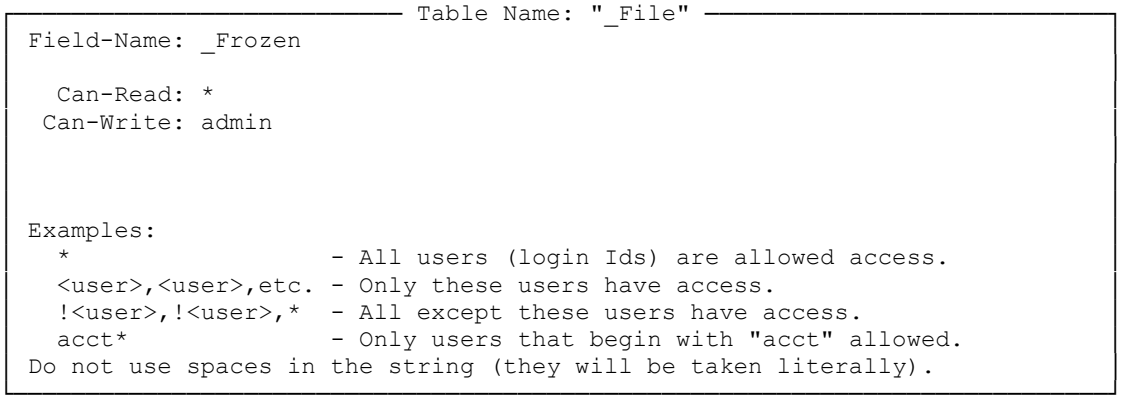
Для усвоения изменения прав доступа на конкретное поле таблицы выполните следующие действия.

- С помощью меню **JumpField** выберите поле _Frozen:



Появится диалоговое окно установки прав доступа к полю таблицы. Обратите внимание, что для поля можно ограничивать только два типа доступа - Can-Read и Can-Write. Для ограничения возможности размораживания и замораживания таблиц нас интересует поле Can-Write.

- Выберите пункт меню **Modify** и с помощью клавиши <Tab> перейдите на соответствующее поле доступа. Введите в поле Can-Write вместо символа звездочки ID администратора безопасности:



- Нажмите клавишу <F1>. Изменение будет сохранено, и с этого момента ни один пользователь, кроме администратора базы данных, не сможет выполнить какие-либо действия в пункте меню **Freeze/Unfreeze**.

Обратите внимание на иерархию прав доступа к таблице базы данных. Если вы устанавливаете ограничение на запись на уровне таблицы, то доступ будет ограничен на запись для всех полей этой таблицы. Если на уровне таблицы ограничение не устанавливать, а вместо этого ID администратора прописать только на уровне конкретного поля, то все остальные пользователи будут иметь доступ ко всем оставшимся полям этой таблицы, кроме этого конкретного поля. В этом легко убедиться, установив ограничение только на поле _Frozen таблицы _File и выполнив следующий код (под обычным пользователем):

```
find last _file where _file-name = "_file" exclusive-lock.
```

```
update _file._frozen
```

Попробовав изменить поле с ограниченным доступом, мы получим следующее сообщение:

```
** Insufficient access privilege for Field _Frozen. (233)
```

А теперь выполните следующий код:

```
find last _file where _file-name = "_file" exclusive-
lock.
update _file._Desc
```

Здесь мы хотим изменить поле описания таблицы, и нам это удастся, т.к. доступ к этому полю ничем не ограничен.

Blank User ID

Когда при подключении к базе данных не указывается ID пользователя или не используются параметры подключения User ID (-U) и Password (-P), тогда пользователю присваивается так называемая учетная запись Blank ID.

Пользователь, использующий Blank ID, будет иметь доступ к таблицам и полям базы данных только в том случае, если:

- база данных позволяет использовать Blank ID;
- доступ на уровне полей и таблиц не ограничен ни для кого;
- выполняемые процедуры предварительно откомпилированы.

Как администратор безопасности базы данных, возможно вы захотите запретить использование Blank ID в базе данных, чтобы несанкционированные пользователи не могли получить доступ к информации. Для этого необходимо выполнить следующие действия.

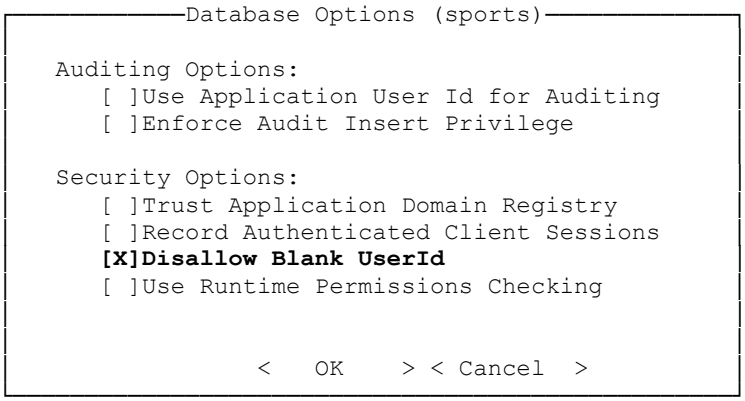
- Войти в Data Dictionary под администратором базы данных.
- Выбрать меню **Admin → Security → Disallow Blank Userid Access**. Выводится диалоговое сообщение о желании ограничить доступ к данным с использованием Blank User ID. Ответьте <Yes>, в результате чего в настройки прав доступа каждой таблицы и каждого поля добавится знак восклицания (!) в начало строки, как показано:

```
┌──────────────────────────── Table Name: "Customer" ─────────────────────────┐
│   Can-Read: !,*                                                            │
│  Can-Write: !,*                                                            │
│ Can-Create: !,*                                                            │
│ Can-Delete: !,*                                                            │
│   Can-Dump: *                                                              │
│   Can-Load: *                                                              │
│                                                                            │
│ Examples:                                                                  │
│   *                  - All users (login Ids) are allowed access.           │
│   <user>,<user>,etc. - Only these users have access.                       │
│   !<user>,!<user>,*  - All except these users have access.                 │
│   acct*              - Only users that begin with "acct" allowed.          │
│ Do not use spaces in the string (they will be taken literally).            │
```

Таким образом, пользователю, который использовал Blank User ID, будет запрещен доступ ко всем данным в базе, при попытке обратиться к ним пользователь получит следующее сообщение:

```
** Insufficient access privilege for table <TabName>
```

- После ограничения доступа к таблицам подключение к базе данных с использованием Blank User ID по-прежнему будет возможным. Поэтому, если необходимо полностью исключить такое подключение, нужно установить флаг запрета подключений с использованием Blank User ID, выбрав пункт меню **Admin → Database Options**:



В этом случае подключение к базе данных будет возможно только с явным указанием ID пользователя и его пароля. Для восстановления возможности подключения с использованием Blank User ID достаточно будет снять флаг отмены в том же самом меню.

Compile-time security

Безопасность во время компиляции (compile-time security) обеспечивается встроенным механизмом OpenEdge. Активация этого механизма позволяет ограничить возможность доступа к таблицам и полям базы данных пользователям, которые попытаются запустить свои собственные программы, не входящие в состав приложения, которому разрешено работать с базой. Этот уровень безопасности также распространяется на различные динамические запросы и использование оператора FIND.

База данных OpenEdge позволяет устанавливать тип прав доступа для каждого пользователя, желающего получить доступ к таблицам и полям базы данных. Проверка прав доступа осуществляется, когда пользователь запускает компиляцию своей программы из текущей сессии. Заданная по умолчанию проверка во время компиляции полезна для приложений, которые компилируются каждый раз, когда различные пользователи запускают это приложение. Однако такая проверка не действует на заранее откомпилированные программы. Если вы используете r-код, основанный по умолчанию на контроле CRC, то пользователь может откомпилировать свою программу на базе данных, имеющей точно такую же схему, и затем запустить её на вашей промышленной базе данных. Таким образом, чтобы обезопасить базу данных, нужно сделать следующее.

- В Data Dictionary в меню **Admin → Database Options** активировать функцию проверки прав доступа при компиляции, установив флаг в поле «Use Runtime Permissions Checking».
- С помощью утилиты PROUTIL с параметром DBAUTHKEY нужно установить ключ авторизации доступа к базе данных. После этого запуск r-кода, если он был откомпилирован на базе с несоответствующим ключом, на промышленной базе данных будет невозможен, даже если схема баз была полностью одинаковой.

Пример установки compile-time security/

1. Войдите в Data Dictionary под учетной записью администратора безопасности.
2. Выберите пункт меню **Admin → Database Options**.
3. В открывшемся окне установите флаг на поле «Use Runtime Permissions Checking». Нажмите кнопку <OK>. Теперь механизм Compile-Time Security активирован.
4. Войдите в меню **Admin → Security → Edit Data Security**. Установите ограничение доступа на поле Can-Read таблицы Customer, разрешив возможность чтения из этой таблицы только для пользователя admin. Выйдите из текущей Progress-сессии.
5. Создайте в базе данных пользователя с именем man1.
6. Войдите в Procedure Editor под пользователем man1.
7. Наберите следующий код:

```
For each customer exclusive-lock.
     update customer.
end.
```

8. Попытайтесь запустить его на выполнение, нажав клавишу <F1>. Вы получите следующее сообщение:

```
** Insufficient access privilege for table Customer. (234)
```

   Здесь сработало ограничение на чтение из таблицы (Can-Read).

9. Добавьте в конец кода оператор QUIT и сохраните этот код в файл `test2.p`. Выйдите из Progress-сессии.
10. Попробуйте запустить на выполнение эту программу из командной строки:

```
mpro sports -p ./test2.p -U man1
```

    Вы получите точно такое же сообщение, какое было указано в пункте 8.

11. Теперь повторите предыдущий пункт, но только в качестве пользователя укажите логин admin. На экране появится информация из таблицы Customer. Это произошло потому, что мы разрешили чтение из таблицы Customer только этому пользователю. Нажмите клавишу <F4> для возврата в командную строку.
12. Войдите в Procedure Editor под пользователем admin и откомпилируйте программу `test2.p`:

```
compile ./test2.p save
```

13. Войдите в Procedure Editor с логином man1 и выполните r-код:

```
run ./test2.r
```

Будет выведено сообщение «Read permission denied (14322)», т.к. в базе выставлен флаг «Use Runtime Permissions Checking». Если этот флаг снять, то запуск r-кода будет возможен, в то время как выполнения исходного кода нет.

С установленным флагом «Use Runtime Permissions Checking» только авторизованные пользователи смогут компилировать исходный код и запускать r-код. Без этого флага только авторизованные пользователи могут компилировать исходный код, но запускать сгенерированный r-код может любой пользователь.

Внимание! Прежде чем вносить какие-либо изменения в настройку безопасности промышленной базы данных всегда предварительно выполняйте тщательное тестирование новой настройки безопасности на тестовых базах, а перед её внедрением в промышленную базу всегда делайте резервную копию этой промышленной базы.

Настройка ключа авторизации в базе данных

Для запрета запуска программ, откомпилированных на другой базе данных, имеющей структуру, идентичную промышленной базе, необходимо на промышленной базе активировать ключ авторизации с помощью утилиты PROUTIL с параметром DBAUTHKEY. Этот параметр используется для установки, изменения и удаления ключа авторизации в базе данных. После того как ключ будет установлен, он будет внедряться в r-код при его компиляции. Таким образом, если запустить такой r-код на базе данных, у которой будет неправильный ключ, или он не будет установлен совсем, то программа не запустится. И наоборот, если на базе данных с ключом авторизации попытаться запустить r-код, скомпилированный на базе, у которой нет ключа, или с ключом, не соответствующим ключу промышленной базы, то такой r-код тоже не отработает.

Синтаксис команды установки ключа авторизации:

```
proutil db-name -C dbauthkey old-key new-key
```

Помните! Если будет забыт текущий ключ авторизации, то вы не сможете его изменить или удалить никаким способом.

После установки ключа авторизации необязательно выполнять полную перекомпиляцию приложения. Для встраивания информации о ключе авторизации в существующий r-код используется параметр RCODEKEY утилиты PROUTIL. Синтаксис этой команды:

```
proutil db-name -C rcodekey old-key new-key r-code-files
```

В таблице приведены возможные комбинации атрибутов параметров DBAUTHKEY и RCODEKEY, используемые для выполнения различных задач:

Задача	**Значение *old-key***	**Значение *new-key***
Установка ключа авторизации.	Плюс (+)	Ключ авторизации
Изменение ключа авторизации.	Текущий ключ авторизации	Новый ключ авторизации
Удаление ключа авторизации/	Текущий ключ авторизации	Плюс (+)

Пример работы с ключом авторизации базы данных.

- Остановите тестовую базу данных:

```
proshut sports -by
```

- Активируйте ключ авторизации:

```
proutil sports -C dbauthkey + TestKey1
```

  В этой команде значение TestKey1 - это наш тестовый ключ авторизации. После выполнения команды будет выведено следующее сообщение:

```
Database authorization key changed to TestKey1. (1973)
```

  Ключ авторизации активирован.

- Запустите базу данных и выполните запуск программы `test2.r`, которую мы откомпилировали ранее:

```
mpro sports -p ./test2.r -U admin –P 123
```

  Работа программы будет невозможна ни под каким пользователем, поскольку её CRC не совпадает с CRC базы данных.

- Теперь зайдите в Procedure Editor с логином admin и откомпилируйте программу `test2.p` заново. Выйдите из Progress-сессии.

- Попробуйте запустить программу снова.

  Программа запустится без ошибок и позволит изменять данные в таблице Customer. Это произойдет потому, что после компиляции в r-код был добавлен ключ авторизации.

Для того, чтобы сделать r-код работоспособным не обязательно его перекомпилировать, воспользуемся параметром RCODEKEY утилиты PROUTIL. Для демонстрации изменим ключ авторизации базы данных с TestKey1 на TestKey2. Чтобы сменить ключ авторизации базы данных, и чтобы текущий r-код программы `test2.r` стал неработоспособным, выполните следующее.

- Остановите базу данных.

- Выполните следующую команду:

```
proutil sports -C dbauthkey TestKey1 TestKey2
```

  Ключ авторизации заменён.

- Запустите базу данных и попробуйте выполнить программу `test2.r`:

```
mpro sports -p ./test2.r -U admin-P 123
```

  Вам этого не удастся, т.к. как ключи авторизации базы данных и r-кода не совпадают.

- Теперь установите для r-кода программы `test2.r` новый ключ авторизации с помощью следующей команды:

```
proutil sports -C rcodekey TestKey1 TestKey2 ./test2.r
```

- Запустите программу `test2.r` снова. Теперь программа отработает нормально, и вы получите доступ на изменение таблицы Customer. Нажмите клавишу <F4> для возврата в командную строку.

Для отмены ключа авторизации для базы данных, а затем и для r-кода программы `test2.p` выполните следующее.

- Остановите базу данных.

- Выполните следующую команду для отмены ключа базы данных:

```
proutil sports -C dbauthkey TestKey2 +
```

  Если текущий ключ авторизации был указан правильно, то вы получите следующее сообщение:

```
Database authorization key cleared
```

  С этого момента у базы данных нет ключа авторизации.

- Исключите из r-кода программы `test2.p` ключ авторизации, чтобы программа могла работать на этой базе:

```
proutil sports -C rcodekey TestKey2 + ./test2.r
```

  Проверьте работу программы, запустив ее из командной строки в однопользовательском режиме. Программа будет работать.

Таким образом, подмена исполняемого r-кода на неавторизованные программы без знания ключа авторизации становится невозможной, точнее, подменить можно, но работать это не будет. **Внимание!** Из соображений безопасности ключ авторизации должен храниться в надежном месте, доступ к нему должен иметь ограниченный круг людей.

Глава 11 Резервное копирование базы данных

Что делать, если вы потеряли тысячи мегабайт ценных данных, потому что никогда не осознавали важности резервного копирования данных? Если система не будет работать по каким-либо причинам в течение какого-либо периода времени, то это может сказаться на ведении бизнеса компании. Следовательно, стратегия резервного копирования и восстановления имеет большое значение для защиты данных и приложений. Проблемы могут возникнуть по разным причинам – отказ оборудования, сбои программного обеспечения, стихийные бедствия, человеческие ошибки, нарушение безопасности. Задача хорошей стратегии состоит в обеспечении связи системы с внешним миром таким образом, чтобы возникающие проблемы не отражались на функционировании системы, а если все же полностью достичь этого не удалось, то время ее простоя сводилось к минимуму. Не менее важной целью является уменьшение или полное устранение потери данных, связанной со сбоем.

Многие пользователи, как правило, сосредоточены на поддержке текущей работы системы, а не на восстановлении. Если у вас есть резервная копия базы данных, но вы **пренебрегали** резервным копированием кодов ваших приложений, то при нештатных ситуациях у вас возникнут большие проблемы. Вам не удастся восстановить систему в полном объеме, поскольку из-за отсутствия приложения не будет возможности получить доступ к восстановленным данным.

Приложение может состоять из множества различных компонентов, включая базу данных, прикладной код, пользовательские файлы, входные файлы, файлы сторонних приложений и т.д. Помните, что ваше приложение построено на вашей OpenEdge-инсталляции с соответствующими обновлениями (Service Packs), исходными кодами и объектами приложения, вспомогательными программами, такими как, например, AppServer, и связанными со всем этим файлами операционной системы.

Для осуществления правильного резервного копирования базы данных OpenEdge необходимо учитывать все файлы, связанные с ней.

- Файлы база данных (.db, .d*n*) и файлы Before-image (.b*n*). Эти файлы содержат данные и информацию о последних прошедших транзакциях. Они обязательны для восстановления базы данных.
- AI-файлы (.a*n*). Если для базы активирован механизм After-imaging, то эти файлы содержат информацию, которая будет необходима для восстановления в случае повреждения диска, на котором расположена база. Они хранят все транзакции, которые были совершены с момента формирования последней резервной копии.
- Лог базы данных (.lg). Файл лога базы данных содержит информацию обо всех событиях, происходящих в ней. Этот файл помогает диагностировать сложные проблемы, понять и решить их. Регулярно копируйте этот файл.
- Transaction log (.t*n*). Если используется механизм Two-phase commit, то эти файлы содержат необходимую информацию для его работы. Если механизм Two-phase commit активирован, то копируйте эти файлы регулярно в рамках общей процедуры резервного копирования.
- Файлы приложений, программы (.p, .r) и библиотеки программ (.pl) также необходимо учитывать при резервном копировании.
- Триггеры базы данных, если они используются, необходимо копировать для поддержания соответствия схемы базы данных и приложения. Т.е. когда происходит копирование базы утилитой PROBKUP, то вместе с данными копируется схема базы

данных, которая ссылается на ABL-триггеры, но сам код триггеров при этом не копируется.

Для решения, какой тип копии базы данных формировать, необходимо знать, какими средствами будет выполняться резервное копирование базы данных, а это в большей мере зависит от важности данных и их ценности.

В следующей таблице указаны возможные способы копирования средствами OpenEdge и средствами операционной системы.

OpenEdge	**Операционная система**
Online или offline.	Только Offline.
Полный или инкрементальный.	Только полный.

Для выяснения, в каком режиме формировать копию, в offline или в online, ответьте на два вопроса.

- База данных активна 24 часа 7 дней в неделю?
- Можно ли остановить ее для резервного копирования?

Если база работает в режиме доступности 24/7/365[24], то используйте online-копирование. Если есть возможность остановить базу, то - offline.

Чтобы определить, какое копирование использовать - полное или инкрементальное, ответьте на вопросы.

- Какой объем данных может принять резервный носитель?
- Имеет ли смысл постоянно формировать полную копию?
- Сколько времени занимает полное копирование?

Если размер базы очень большой, то имеет смысл формировать инкрементальные копии.

Выполнять резервное копирование можно не только с помощью команды PROBKUP, вместо неё можно использовать утилиты операционной системы, но помните, в этом случае невозможно использовать online и инкрементальное копирование. При этом также необходимо быть уверенным, что эти утилиты обеспечат целостность данных и файлов. Частичное копирование приведёт к разрушению базы в целом.

Независимо от выбранного метода резервирования после копирования необходимо копию для подтверждения ее корректности протестировать.

Полная копия подразумевает копирование всех файлов базы данных, включая BI-файлы. Сформировать полную копию можно утилитой PROBKUP или утилитой операционной системы. В идеале выполнять полную копию базы данных необходимо каждый день, тем не менее, в зависимости от принятого плана восстановления можно сократить количество полных копий, прибегнув к использованию инкрементальных копий, или даже к механизму After-Imaging.

Инкрементальная копия - это копирование только той части данных, которая была изменена с момента формирования последней полной или последней инкрементальной

[24] 24/7/365 - 24 часа 7 дней в неделю 365 дней в году

копии. Такие копии занимают меньше места на носителе, соответственно затрачивается меньше времени на их формирование. Для формирования инкрементальной копии необходимо использовать утилиту PROBKUP. Перед началом выполнения инкрементальных копий обязательно выполнить полную резервную копию. Необходимо выполнять полные копии через определенное количество инкрементальных копий. Если регулярно не будут формироваться полные копии, то размеры инкрементальных копий будут увеличиваться, равно, как будет увеличиваться и время восстановления, что связано с увеличением количества инкрементальных копий после последнего полного копирования.

Копирование в online позволяет создавать копии в то время, когда база данных используется пользователями. Сделать это можно с помощью утилиты PROBKUP. Прибегают к этому методу, когда база данных не может находиться в недоступном состоянии длительное время, достаточное для формирования копии в offline. В online также можно выполнять полные и инкрементальные копии.

При принятии решения по использованию online-копирования необходимо учитывать условия для его использования и действия, их обеспечивающие.

- Нельзя использовать online-копирование баз данных, используемых в однопользовательском режиме.
- Если у базы активирован механизм After-Imaging, то при выполнении online-копии произойдет автоматическое переключение на следующий AI-экстент. Прежде чем запустить online-копирование, необходимо убедиться, что следующий AI-экстент пустой. Если он окажется не пустой, то PROBKUP завершится уведомлением о невозможности переключиться на новый AI-экстент.
- Если необходимо активировать After-imaging в базе, находящейся в online, то это можно сделать одновременно с формированием полной online-копии, которая и будет считаться базовой для AI-файлов.
- Когда запускается online-копирование база данных на время «замораживается», пока не будет записан заголовок копии. Убедитесь, что резервный носитель правильно настроен и готов к получению данных для того, чтобы минимизировать длительность паузы. Пока не будет записан заголовок, будет невозможно вносить изменения в базу данных.
- Если формируется многотомная копия, то задержка в работе будет возникать при каждом переключении на новый том.
- Нельзя использовать для online-копирования параметры <-scan> и <-estimate>.

Для выполнения offline-копии сначала необходимо остановить базу данных. Если копия будет выполняться средствами утилит операционной системы на запущенной базе данных, то она будет неработоспособна, во избежание этого перед началом копирования воспользуйтесь командой PROQUIET. Для offline-копирования можно использовать утилиты операционной системы или команду PROBKUP.

Команда PROBKUP

Команда PROBKUP - это встроенное средство копирования баз данных OpenEdge, гарантирующее целостность резервной копии во время её формирования. Команда имеет параметр <online>, который позволяет выполнять копирование даже тогда, когда база данных запущена и используется.

Следующие шаги кратко описывают процесс работы PROBKUP/

1. Устанавливается временный замок на базу данных (только в online).
2. Выполняется псевдо-контрольная точка (только в online).
3. Выполняются переключения AI-экстентов.
4. Выполняется копирование области Primary Recovery (только в online).
5. Снимается замок с базы данных (только в online).
6. Выполняется резервное копирование базы.

Копирование базы данных выполняется до уровня High-Water Mark, а все свободные (free) блоки с целью сохранения пространства сжимаются. Копия, выполненная в online, представляет собой базу данных на момент начала формирования online-копии. Это означает, что все транзакции, совершенные после начала копирования и во время копирования, не войдут в эту копию. Что же касается выполнения псевдо-контрольной точки, то она необходима для синхронизации памяти с дисками перед началом online-копирования. Это важно, т.к. позволит утилите скопировать все данные, которые в момент начала копирования находились в памяти и еще не были сброшены на диск. Это еще одно отличие команды PROBKUP от утилит копирования операционной системы, которые копируют только те данные, которые находятся на диске, не учитывая изменений, находящихся в памяти.

Дополнительно можно использовать утилиту PROBKUP для активации таких механизмов как After-Imaging и AI File Management, не прибегая к остановке базы данных.

Синтаксис утилиты PROBKUP:

```
probkup [ online ] db-name [ incremental ] device-name
[ enableai ]
[ enableaiarchiver -aiarcdir dirlist [-aiarcinterval n] [-
aiarcdircreate]]
[ -estimate | -vs n | -bf n | -verbose | -scan | -io i | -com | -
red i| -norecover]
```

Параметр	Назначение
online	Выполнение копии в online.
db-name	Имя копируемой базы данных.
incremental	Формирование инкрементальной копии.
device-name	Имя резервной копии.
enableai	Активирование механизма After-Imaging после формирования online-копии. Используется только при формировании полной резервной копии.

Параметр	Назначение
enableaiarchiver	Активирование механизма AI File Management по завершению формирования копии. Может использоваться либо совместно с <enableai>, либо отдельно. Имеет несколько дополнительных параметров. • -aiarcdir dirlist - обязательный параметр. Указывает на расположение каталога или каталогов, куда будут сохраняться архивы AI-экстентов. Каталоги должны быть разделены запятой. Имена каталогов не должны содержать пробелы. Каталоги должны быть предварительно созданы, в противном случае необходимо указать параметр <-aiarcdircreate>. • -aiarcdircreate - указывает на то, что перед активацией AI File Management необходимо проверить наличие каталогов, указанных параметром <-aiarcdir> и при отсутствии создать их. • -aiarcinterval n - указывает на необходимость принудительного переключения AI-экстентов через определенный промежуток времени, который определяется параметром n в секундах. Переключение будет происходить, даже если экстент не заполнен. Если параметр не будет указан, то автоматическое переключение выполнится только после получения экстентом статуса FULL. Использование этого параметра особенно рекомендуется для экстентов переменного размера. Минимально возможный интервал времени равен 2 минутам, а максимально возможный - 24 часам.
-estimate	Приблизительная оценка дискового пространства, которое необходимо для резервной копии. Для более точной оценки используйте этот параметр совместно с параметром <-scan>. Параметр разрешен только для offline-копирования, во время online-копирования он будет проигнорирован. При использовании этого параметра резервная копия не формируется.
-scan	Сканирование базы данных и вычисление размера необходимого пространства для резервной копии. Если использовать его совместно с параметром <-com>, то PROBKUP просканирует базу данных и вычислит размер с учетом сжатия данных. Параметр используется только для offline-копирования, во время online-копирования он будет проигнорирован.
-com	Сжатие данных перед сохранением на резервный носитель. При этом неиспользуемая часть блоков сжимается до трех-байтной сжатой последовательности. Свободные блоки (free) сжимаются до размера их заголовков (16 байт).
-vs *n*	Размер тома резервной копии (в блоках базы данных). Если вы хотите, чтобы размер тома резервной копии был равен 200Мб, и при этом база данных имеет блок размером 8Кб, то необходимо указать значение, равное 25600 (25600 * 8Кб/1024 = 200Мб). В результате перед формированием нового тома будет запрошено имя нового файла. Если параметр не используется, то PROBKUP не ограничивает размер файла резервной копии. Он будет записывать в него данные, пока не завершится процесс копирования, или пока не закончится свободное пространство на носителе. Если носитель заполнится полностью, то PROBKUP попросит указать другой файл.
-bf *n*	Количество блоков данных, которое должно быть буферизовано перед записью на резервный носитель. Использование этого параметра предполагает увеличение скорости передачи данных на резервный носитель. По умолчанию параметр имеет значение, равное 34. Максимальное значение равно 1024.
-verbose	Отображение процесса резервного копирования на экране, каждые 10 секунд выводятся строки «Backed up *n* blocks in *hh:mm:ss*». При отсутствии параметра PROBKUP выводится только сообщение о завершении резервного копирования.
-io *i*	Коэффициент инкрементального перекрытия. Определяет избыточность инкрементальной копии. Значение, равное единице, указывает на возможность потери одной предыдущей инкрементальной копии. Значение, равное двум, допускает потерю двух предыдущих инкрементальных копий, и т.д. По умолчанию параметр имеет значение, равное нулю, что не допускает потери ни одной инкрементальной резервной копии.

Параметр	Назначение
-red *i*	Коэффициент избыточности. Используется для добавления блоков коррекции ошибок. Параметр *i* - это положительное целочисленное значение, которое указывает на количество блоков данных для каждого блока коррекции. Таки образом, PROBKUP создает для каждых *i*-блоков данных один блок коррекции ошибок. Эти блоки используются для восстановления поврежденной резервной копии. Чем меньше коэффициент избыточности, тем больше будет создано блоков коррекции ошибок, а значит, надежность резервной копии увеличится. Если вы укажите коэффициент, равный единице, то это будет означать, что PROBKUP создаст для каждого блока данных один блок коррекции ошибок. Поскольку для формирования такой резервной копии понадобится больше времени и свободного пространства на носителе, то устанавливать такое значение рекомендуется в том случае, если вы не уверены в надежности резервного носителя. По умолчанию коэффициент имеет нулевое значение, т.е. блоки коррекции ошибок не создаются.
-norecover	Отсутствие необходимости выполнения процесса Crash Recovery перед началом формирования резервной копии. При этом в копию будет включена область Before-Image. В основном используется для формирования резервной копии с целевых баз данных, на которые накатываются архивы AI-экстентов.

При выполнении резервного копирования с помощью команды PROBKUP всегда помните следующее.

- Восстанавливаемая база данных должна иметь структуру, идентичную исходной базе данных. Это означает, что она должна иметь такое же количество областей хранения, такой же размер блока базы данных и т.п.
- Минимальный размер тома резервной копии равен 34Кб.
- Нельзя выполнить online-копирование, если база данных:
  - используется в однопользовательском режиме;
  - запущена с параметром No Shared Memory <-noshm>;
  - запущена с параметром No Crash Protection <-i>.
- Если активирован механизм After-Imaging, то перед выполнением online-копии PROBKUP переключит AI-экстенты, что бы установить отправную точку для выполнения процесса Roll-forward. Поэтому, прежде чем начать выполнение online-копирования, проверьте, что следующий AI-экстент пустой (empty), если он окажется непустым, то работа PROBKUP будет прервана.
- Если используется параметр <-com>, то можно уменьшить размер копии на 10-40 %%, в зависимости от копируемой базы данных.
- Если BI-файл не был усечен перед выполнением копирования и не использовался параметр <-norecover> менеджер базы запустит механизм Crash Recovery.
- Включение механизма AI File Management требует включения After-Imaging. Их можно включить одновременно или по отдельности, при этом первым должно выполняться включение After-Imaging.

Пример резервного копирования базы с помощью PROBKUP.

- Проверьте, что база данных не используется и находится в offline:

```
proutil sports -C busy
```

- С помощью параметра <-estimate> определите, какое количество свободного пространства на резервном носителе вам необходимо (этот шаг необязательный):

```
probkup sports sports-full.bak -com -red 5 -scan -estimate
```

В результате будет получено следующее сообщение:

```
932 active blocks out of 954 blocks in sports will be dumped. (6686)
0 BI blocks will be dumped. (6688)
The blocksize is 4096. (6994)
Backup requires an estimated 769.6 KBytes of media. (9285)
Restore would require an estimated 932 db blocks using 3.6 MBytes of media.
(9286)
Backup estimate complete. (3739)
```

- Убедившись, что свободного пространства на резервном носителе достаточно, выполните эту же команду, но без параметра <-estimate>. По завершению её работы появится файл `sports-full.bak`. **Примечание**: в данном примере название этого файла выбрано исключительно в демонстрационных целях. В реальной работе в имя резервной копии необходимо включать как можно больше информации, достаточной для идентификации резервной копии, например, дату формирования, время формирования, имя базы данных, тип копии и т.п.

Для формирования инкрементальной копии после имени базы данных добавьте параметр <incremental>. Для формирования online-копии добавьте параметр <online> сразу после команды probkup.

Всегда тестируйте сформированные резервные копии! Это гарантирует, что они будут актуальны и доступны, когда это станет необходимо. Не забудьте также протестировать копию перед тем, как восстанавливать базу данных. В идеале копия должна быть протестирована сразу после того, как сформирована. Если проблема обнаружится во время тестирования, вы сможете сразу сформировать новую копию. Если же вы оставите тестирование копии до того момента, когда она вам понадобится, то в случае отрицательного результата теста вы можете потерять всю базу данных.

Таблица описывает параметры утилиты восстановления PROREST, которые используются для тестирования резервных копий.

Параметр	Описание
-vp	Partial Verify - параметр указывает утилите восстановления на необходимость чтения томов копии, вычисления и сравнения CRC в заголовках блоков. Для восстановления поврежденного блока необходимо при резервном копировании определить коэффициент избыточности <-red>. Вы можете использовать эту проверку как в offline, так и в online.
-vf	Full Verify - параметр указывает утилите PROREST на необходимость сравнения каждого резервного блока копии с соответствующим блоком базы данных. Используется только в offline.

Внимание! Утилита PROREST с параметрами Partial Verify или Full Verify не выполняет физического восстановления базы данных и не вносит в нее никаких изменений.

CRC-копии

При восстановлении поврежденного блока в резервной копии для его идентификации команда PROREST использует CRC-код. Этот код рассчитывается для каждого блока резервной копии. Для восстановления поврежденных блоков используются блоки коррекции ошибок, эти блоки включаются в резервную копию, если при ее формировании использовался параметр <-red>.

Когда PROBKUP записывает блок данных на резервный носитель, вычисляется его CRC-код, который записывается вместе с ним. При восстановлении базы данных из резервной копии команда PROREST просматривает содержимое блока и проверяет его на соответствие сопутствующему CRC-коду. Если содержимое не соответствует CRC-коду, то блок считается поврежденным.

Если копия содержит блоки коррекции ошибок, то PROREST автоматически использует информацию из них для восстановления поврежденного блока. Если же копия не содержит блоков коррекции ошибок, то PROREST не сможет восстановить поврежденный блок при восстановлении базы данных, дальнейшая работа утилиты будет невозможна. Блоки коррекции ошибок содержат информацию обо всех блоках резервной копии и позволяют утилите PROREST восстанавливать поврежденные блоки.

Блоки коррекции ошибок и блоки, на которых они основаны, называют набором избыточности. Включение блоков коррекции в копию обеспечивается добавлением параметра <-red> в команду PROBKUP. Параметр <-red> определяет, какое количество блоков данных будет обслуживать один блок коррекции ошибок в каждом наборе избыточности. Например, если определяется показатель избыточности 2, то PROBKUP создаст блок коррекции ошибки для каждых двух блоков копии. Утилита PROREST в состоянии восстановить поврежденный блок, только если он единственный в установленном наборе избыточности. Если набор избыточности содержит более одного поврежденного блока, или один поврежденный блок и поврежденный блок коррекции, то PROREST не сможет восстановить их.

Если определяется очень маленький показатель избыточности (например, 2), то шанс получить два и более поврежденных блока в наборе избыточности очень мал. Соответственно, при увеличении значения показателя избыточности увеличивается и вероятность ошибки. Тем не менее, низкий показатель увеличивает размер копии, как следствие, увеличиваются время на формирование копии и размер необходимого резервного носителя. Выбор зависит от надежности резервного носителя, если он считается надежным, то показатель можно увеличить или наоборот, при менее надежном носителе - уменьшается и значение показателя избыточности.

Размер каждого блока резервной копии, следовательно, и каждого блока коррекции ошибок определяется параметром <-bf> команды PROBKUP. По умолчанию значение этого параметра равно 34. Например, если размер блока базы данных 1024 байт и значение параметра <-bf> задано 40, то каждый блок резервной копии - это 40Кб или 40 блоков базы данных.

Проверка копий

Сразу же после формирования резервной копии базы данных желательно проверить её на наличие поврежденных блоков данных. Как уже отмечалось ранее, с помощью утилиты восстановления PROREST можно проверить целостность как полной, так и инкрементальной копии. Для этого запустите команду PROREST с параметром Partial Verify <-vp>. Этот параметр проверяет копию на наличие поврежденных блоков и выдает отчет. Если запустить команду PROREST с параметром Full Verify <-vf>, то выполнится сравнение блоков копий с блоками базы данных.

Использование параметров проверки не приводит к восстановлению базы данных. Здесь только проверятся состояние копии, сообщается о наличии поврежденных блоков, и если

таковые существуют, то они исправляются. Для восстановления базы данных необходимо запустить PROREST снова, но без этих параметров.

При использовании параметра <-vp> PROREST сканирует копию и пересчитывает CRC-код для каждого блока, а затем сравнивает полученный результат с CRC-кодом, расположенным в заголовке блока. Если CRC-коды не совпадают, то PROREST помечает блок, как поврежденный и выводит на экран следующее сообщение:

```
CRC check failed reading backup block n
```

Если копия содержит блоки коррекции ошибок, то PROREST использует их чтобы попытаться восстановить поврежденный блок. Если PROREST восстановит поврежденный блок, на экран выведется следующее сообщение:

```
Recovered backup block n
```

Если PROREST не сможет восстановить блок, то он отобразит следующую информацию на экране:

```
n CRC error within recovery group - recovery impossible
```

Команда PROREST не сможет произвести восстановление, если CRC не совпадают в самом блоке коррекции ошибок. Если такая ситуация произойдет, то будет выдано следующее сообщение:

```
Unable to recover previous block in error
```

Если в процессе проверки будет обнаружено 10 ошибок, то вам будет предложено остановить процесс:

```
10 read errors have occurred. Do you want to continue? [y/n]
```

Риск повреждения резервной копии присутствует всегда, поэтому если вы не хотите потерять свои данные, то использование блоков коррекции ошибок (параметр <-red>) рекомендуется.

Пример проверки резервной копии.

- Частичная проверка:

```
prorest sports sports-multytom-part0.bak -vp <tome.txt
```

  Параметр <-vp> позволяет выполнять проверку, когда база данных находит как в offline, так и в online. Результат проверки будет отображен следующим образом:

```
This is a full backup of /users/valeriy/lecAI/db1/sports.db. (6759)
This backup was taken Wed Aug 19 15:05:21 2009. (6760)
The blocksize is 4096. (6994)
Partial verification successfully read backup volume. (6765)
Verify pass started. (3751)

Please enter next device/file name or type "quit" to exit:
Started verifying volume 2. (10828)
Verified 1348 db blocks in 00:00:00
Backup for /users/valeriy/lecAI/db1/sports.db verified ok. (6758)
```

- Полная проверка:

```
prorest sports sports-multytom-part0.bak -vf <tome.txt
```

Поскольку параметр <-vf> обращается к каждому блоку базы данных для сравнения его с блоком резервной копии, то использование этого параметра возможно только для базы, находящейся в offline. Результат проверки будет отображен следующим образом:

```
This is a full backup of /users/valeriy/lecAI/db1/sports.db. (6759)
This backup was taken Wed Aug 19 15:05:21 2009. (6760)
The blocksize is 4096. (6994)
It will require a minimum of 1651 blocks to restore. (6763)
Full verify pass started. (3752)
Please enter next device/file name or type "quit" to exit:
Verified 1348 db blocks in 00:00:00
Full verify successful. (3758)
```

Внимание! Для возможности автоматического восстановления поврежденных блоков вы должны при формировании резервной копии использовать параметр <-red>.

Примечание: на DVD-диске вы найдете скрипт `backup.sh`, который может служить примером для разработки собственных скриптов резервного копирования базы данных. Этот скрипт полностью работоспособен, но перед использованием рекомендуется тщательно протестировать его работу в вашей операционной среде. Описание скрипта находится в файле `readme-backup.txt`. Код скрипта распространяется бесплатно на условиях «КАК ЕСТЬ» без каких-либо гарантий любого рода. Все риски, возникающие от использования скрипта `backup.sh`, возлагаются на пользователя.

Средства операционной системы

Использование для копирования базы данных средств операционной системы накладывает на администратора дополнительную ответственность. Связано это с тем, что при копировании ему необходимо учесть все файлы базы данных.

В общих чертах для резервного копирования средствами операционной системы необходимо выполнить следующее.

1. Убедиться, что известны все файлы базы данных для выполнения резервного копирования. Составьте их полный список и место расположения.
2. Убедиться, что база данных никем не используется, и не будет использоваться во время копирования. Если установлена лицензия Enterprise, то можно воспользоваться командой PROQUIT для активации точки останова, которая заморозит всю деятельность в базе данных на время выполнения копирования, иначе копия будет повреждена.
3. В журнале базы данных (.lg) установить свою отметку о начале процесса копирования.
4. Скопировать файлы базы данных средствами операционной системы. При этом проверить, что эти утилиты гарантируют целостность копируемых файлов. Например, при копировании по ftp-соединению должен быть включен режим поддержки бинарных файлов.
5. После завершения копирования проверить целостность копии. Для этого посмотрите в журнал исходной базы данных, после вашей отметки не должно быть

никаких записей. Если же они все-таки были, то вероятно, что базу данных использовали. В этом случае копия должна считаться неработоспособной и необходимо выполнить новую копию. Убедиться, что скопированы все файлы базы данных, и они имеют те же размеры, что и исходные. В этом плане работать с утилитой PROBKUP легче, т.к. она имеет список всех необходимых файлов и автоматически выполняет их копирование.

6. Установить флаг выполнения резервной копии для исходной базы данных утилитой RFUTIL MARK BACKEDUP и деактивировать точку останова.

Пример формирования резервной копии средствами операционной системы.

- Активируйте точку останова командой PROQUIET на работающей базе данных:

```
proquiet sports enable
```

- Обновите структурный файл базы данных командой PROSTRC LIST:

```
prostrct list sports
```

  В нашем примере вся база данных размещена в одном каталоге, поэтому задача упрощается. В вашем же случае файлы базы данных могут располагаться где угодно, поэтому вы должны более внимательно отнестись к формированию списка копируемых файлов. Самый простой способ - это использовать информацию о размещении файлов базы данных из структурного файла.

- Выполните копирование файлов базы данных sports из текущего каталога базы данных в каталог db2. Предварительно перейдите в каталог базы данных командой cd:

```
cd ./lecAI/db1

for f in `ls -p sports*.* | grep -v lk | grep -v lic`; do cp
`pwd`/$f ./db2; done

ls -la ./db2
```

```
total 3544
drwxrwxr-x 2 valeriy valeriy    4096 Aug 19 14:27 .
drwxrwxrwx 9 valeriy valeriy    4096 Aug 19 14:16 ..
-rw-rw-r-- 1 valeriy valeriy 2228224 Aug 19 14:27 sports.b1
-rw-rw-r-- 1 valeriy valeriy 1179648 Aug 19 14:27 sports.d1
-rw-rw-r-- 1 valeriy valeriy   32768 Aug 19 14:27 sports.db
-rw-rw-r-- 1 valeriy valeriy  158424 Aug 19 14:27 sports.lg
-rw-rw-r-- 1 valeriy valeriy     509 Aug 19 14:27 sports.st
```

  Как видим из результата действия команды `ls`, файлы базы данных скопировались успешно.

- Ели база была в offline, то пометьте ее как скопированную следующей командой:

```
proutil sports -C mark backedup
```

- Деактивируйте точку останова:

```
proquiet sports disable
```

Теперь база данных опять доступна пользователям.

Следующий шаг - проверка полученной копии, он же - способ восстановления из такой копии.

- Перейдите в каталог ./db2.
- Откройте структурный файл (.st) и отредактируйте пути к базе данных.
- Восстановите описание файлов внутри базы на основании структурного файла следующей командой:

```
prostrct repair sports
```

- Запустите базу данных. После запуска будет выполнен откат незавершенных транзакций и очистка информации о сегментах разделяемой памяти, оставшихся от оригинальной базы данных. После чего доступ к базе данных будет открыт.

Глава 12 Восстановление базы данных

Что же такое восстановление базы данных? Восстановление базы данных - это процесс приведения базы данных в актуальное и консистентное состояние после логических или физических сбоев. Восстановление может выполняться как полностью автоматически с помощью встроенных механизмов СУБД, так и из резервной копии администратором базы данных. В первом случае цель достигается, если повреждения базы были незначительные, или это были даже не повреждения, а обычный откат транзакции. Второй способ является крайним случаем, и к нему прибегают только в случае невозможности восстановления работоспособности базы данных автоматически.

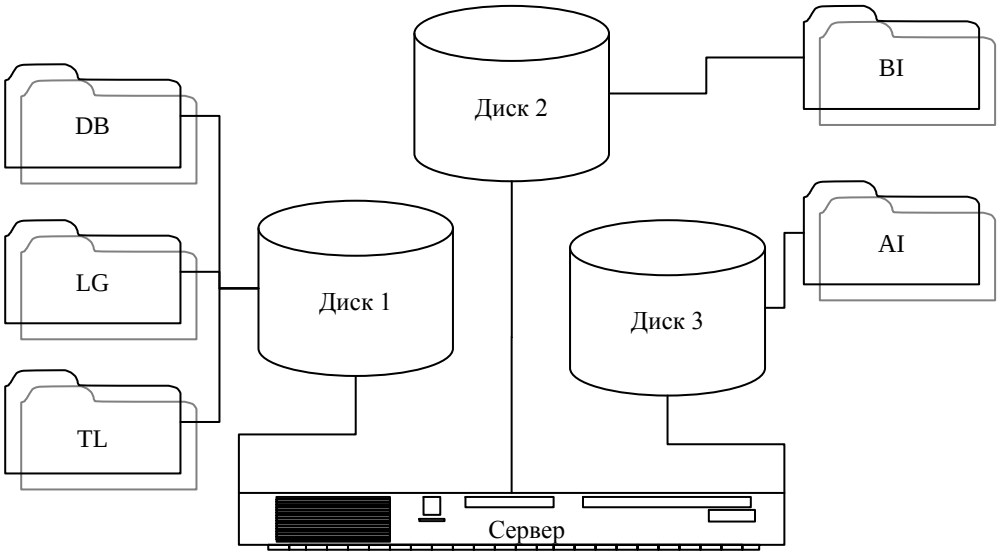
Правила восстановления

Существует несколько важных правил, которых стоит придерживаться при выполнении восстановления базы данных из резервной копии.

- Перед началом восстановления необходима проверка целостности резервной копии.
- Если восстановление выполняется поверх существующей базы данных, то обязательно предварительное создание копии этой базы любыми возможными средствами. Это необходимо, если по каким-либо причинам будет отменено решение о перезаписи существующей базы или в дальнейшем необходимо будет определить причины повреждения базы.
- Восстановление из резервной копии следует выполнять на той версии OpenEdge, на которой эта копия была сформирована.
- Восстановление инкрементальной резервной копии возможно только после восстановления базы из полной резервной копии (или инкрементальной), на которой базируется эта копия.
- Структура восстанавливаемой базы данных должна соответствовать структуре базы данных, с которой была снята резервная копия, т.е. размер блока базы данных, количество и название областей хранения, количество экстентов областей хранения должны быть одинаковыми. Количество экстентов областей хранения может быть больше, но меньше не может, или в базе данных должна быть включена поддержка больших файлов, которая не требует разделения области на экстенты и позволяет записывать все данные в один, неограниченно большой экстент. В этом случае экстент области хранения должен быть переменного размера, либо при фиксированном размере должен быть достаточно большим.
- Восстановление резервных копий должно осуществляться в том же порядке, в котором они были сформированы, т.е. сначала восстанавливается полная резервная копия, а потом все последующие инкрементальные копии по порядку.
- Если по каким-либо причинам какая-либо инкрементальная копия утеряна, а при формировании копий использовался коэффициент инкрементального перекрытия 1 (параметр команды PROBKUP Overlap <-io>), то следующая инкрементальная копия восстанавливается без ошибок, поскольку она содержит данные предыдущей копии.
- Если после восстановления полной копии необходимо продолжить восстановление из инкрементальных копий, то нельзя пытаться каким-либо образом использовать восстановленную из полной копии базу данных. Если база данных будет изменена в промежутке времени между восстановлением полной копии и восстановлением инкрементальной копии, то восстановление инкрементальной копии будет невозможно.

- Если в момент восстановления копии произошла какая-либо ошибка, следует запустить восстановление повторно, начиная с той копии, на которой произошла ошибка.

Во многом успешное восстановление базы данных зависит не только от того, каким образом выполняется резервное копирование, и какими утилитами она восстанавливается, но и от того, где и как размещаются компоненты базы данных. На рисунке приведен пример размещения компонентов базы данных, обеспечивающего гарантированное её восстановление:



Объясняется такое расположение следующим.

- BI-файл должен располагаться на отдельном диске, т.к. он является активным компонентом базы данных, постоянно изменяющимся и создающим высокую I/O-активность на диске. Размещение BI-файла на отдельном диске снизит I/O-нагрузку и будет способствовать улучшению производительности базы данных в целом.
- AI-файл расположен на отдельном диске, т.к. совместно с предыдущей резервной копией его содержимое соответствует текущему содержимому базы данных, следовательно, если будет потерян носитель с базой данных, то вы всегда сможете восстановить ее из резервной копии и AI-файлов. Если быAI-файлы были размещены на том же самом диске, что и база данных, то в случае сбоя этого диска, вы потеряли бы все изменения, которые были выполнены с момента формирования последней резервной копии.

Во время восстановления работоспособности системы очень важно иметь готовый и отлаженный план резервного восстановления. Даже если ваша система небольшая, есть вероятность забыть некоторые мелкие детали, непринятие во внимание которых может впоследствии создать значительные проблемы, что может в итоге все ваши усилия по восстановлению свести к нулю. Кроме того, имея под рукой такой план, вы будете четко представлять себе последовательность необходимых действий, что значительно сэкономит ваше время. Итак, это должен быть документ, содержащий последовательность действий для восстановления работоспособности базы данных при возникновении сбоев. При его разработке необходимо учитывать любые потенциально

возможные случаи сбоев, от временных локальных системных и дисковых сбоев, до таких природных катаклизмов, как землетрясение, ураган и пр.

Перед разработкой плана необходимо сначала определиться с требованиями к базе данных. Ответьте на следующие вопросы.

- Какое количество транзакции вы можете потерять?
- Как долго приложения, работающие с базой данных, могут находиться в offline при проведении плановых работ, например, на время формирования резервных копий?
- Если база данных или операционная система становятся неработоспособными, сколько времени понадобится на их восстановление?
- Как вы будете тестировать ваш план?

Дополнительно к времени, требующемуся на восстановление базы, необходимо добавить время, затрачиваемое на исправление аппаратной части, файловой системы, системных дисков, либо других системных компонентов.

Для гарантированного корректного восстановления:

<u>всегда:</u>

- включайте в состав плана по восстановлению пошаговые инструкции и чек-листы;
- держите копию плана восстановления на бумажном носителе;
- регулярно выполняйте резервное копирование базы данных;
- разместите AI-файлы отдельно от DB и BI-файлов;
- наиболее информативно помечайте резервные копии;
- тестируйте резервные копии сразу после их формирования;
- храните резервные копии на отдельном сервере, лучше в отдельном здании;

<u>никогда:</u>

- не храните BI-файлы на одном диске с AI-файлами;
- не удаляйте файлы DB, AI или BI, если не уверены, что база данных вам больше не понадобится, или у вас отсутствует работоспособная копия базы;
- при копировании базы средствами операционной системы не копируйте файлы базы данных без BI-файлов;
- не восстанавливайте базу данных без соответствующих BI-файлов;
- не копируйте базу данных средствами операционной системы без применения к ней команды PROQUIT.

Внимание! Если база данных запущена с параметром Unreliable Buffered I/O <-r> или с параметром No Crash Protection <-i>, и в это время произойдет системный сбой, вы не сможете восстановить базу данных.

Команда PROREST

Для восстановления баз данных из резервных копий существует специальная утилита, которая называется PROREST. Утилита позволяет выполнять восстановление из любых резервных копий, которые были сформированы с помощью утилиты PROBKUP.

Синтаксис утилиты PROREST:

```
prorest dbname device-name [-list | -vp | -vf]
```

Параметр	Назначение
dbname	Имя базы данных, которую необходимо восстановить.
device-name	Путь и имя резервной копии базы данных, из которой будет происходить восстановление.
-list	Извлечение описания всех областей хранения, содержащихся в копии. Следует использовать этот параметр для формирования нового структурного файла и создания базы данных, в которую будет происходить восстановление копии.
-vp	Проверка копии путем вычисления CRC-кода блоков копии и сравнения его с CRC-кодом, хранящимся в заголовке блока. Для восстановления поврежденного блока, при формировании резервной копии базы данных необходимо определить коэффициент избыточности <-red>.
-vf	Сравнение каждого блока копии с каждым блоком базы данных. Работает только относительно базы данных, с которой была снята копия и при условии, что база данных не используется. Выполнить такую проверку можно только сразу после завершения формирования резервной копии.

Перед началом восстановления на экране отобразится сообщение о дате копии и необходимом количестве блоков для восстановления базы данных.

При первом запуске базы данных, восстановленной из online-копии, сначала запустится механизм восстановления (Crash recovery), который откатит все транзакции, не завершенные на момент копирования базы.

Если происходит восстановление из полной копии базы данных, то предпочтительно делать его в новую базу, что при необходимости дает возможность получить доступ к разрушенной базе данных.

Первая инкрементальная копия должна быть восстановлена на базу данных, восстановленную из полной копии.

Если утилита PROREST сталкивается с поврежденными блоками копии, которые она не в состоянии восстановить, то данные в этих блоках будут потеряны. Количество потерянных данных приблизительно равно количеству поврежденных блоков, умноженных на значение параметра Blocking Factor <-bf> команды PROBKUP.

Примеры выполнения восстановления баз данных из резервных копий

Получение описания областей хранения из резервной копии

Для получения описания областей хранения, необходимых для восстановления копии, используется утилита PROREST с параметром <-list>. Синтаксис команды:

```
prorest db-name device-name -list
```

Результат работы утилиты:

```
Area Name: Schema Area
       Size: 9728, Records/Block: 32, Area Number: 6, Cluster Size 1
Area Name: Employee
       Size: 3040, Records/Block: 32, Area Number: 7, Cluster Size 1
Area Name: Inventory
       Size: 10208, Records/Block: 32, Area Number: 8, Cluster Size 1
```

Полученную информацию можно использовать для формирования структурного файла для восстанавливаемой базы данных. Например, программа `create-st.p`, которую можно найти на DVD-диске, прилагаемом к этому изданию, используя указанную выше информацию, создаёт соответствующий структурный файл.

Восстановление в пустой каталог из полной резервной копии

Создайте новый каталог с именем dbrest в любом удобном для вас месте, и сделайте его текущим каталогом:

```
mkdir dbrest
cd dbrest
```

Скопируйте в него файл резервной копии, например, `sports.bak` и выполните следующую команду:

```
prorest ./sports sports.bak
```

Утилита PROREST сформирует структуру восстанавливаемой базы на основании данных, хранящихся в резервной копии, и восстановит в ней данные. Но зачем тогда использовать параметр <-list> для предварительного создания структурного файла? Обратите внимание на то, что PROREST создает структуру, исходя из информации о наличии той или иной области, но не создает экстенты. Поэтому и была разработана программа `create-st.p`, которая создает структурный файл с учетом необходимого количества экстентов. Однако если ваша система поддерживает работу с большими файлами, то достаточно стандартного восстановления.

Обратите внимание на тот факт, что в структурном файле, созданном утилитой PROREST, используются относительные пути к экстентам базы данных, т.е. все экстенты будут по умолчанию размещены в одном и том же каталоге. Если после восстановления выполнить команду PROSTRCT LIST, то структурный файл будет обновлен и все относительные пути в нём и в базе данных будут заменены на абсолютные пути.

Восстановление поверх существующей базы данных

При выполнении восстановления поверх существующей базы данных действия не отличаются от обычного восстановления в пустой каталог. Единственным условием корректного восстановления будут следующие требования:

- Структура и размер блока перезаписываемой базы данных должны быть идентичны структуре и размеру базы, с которой была сформирована резервная копия.

Перед началом восстановления будет выведен запрос на подтверждение перезаписи существующей базы данных.

```
sports already exists.
Do you want to over write it? [y/n]
```

Поскольку восстановить перезаписываемую базу данных будет невозможно, то рекомендуется предварительно сформировать её резервную копию любыми доступными средствами.

Восстановление базы данных из инкрементальной резервной копии

Восстановление инкрементальной копии может быть выполнено, только если предварительно была восстановлена полная или предыдущая инкрементальная копия, на которой базируется восстанавливаемая копия. В противном случае восстановление завершится с ошибкой. В остальном процесс восстановления ничем не отличается от восстановления полной копии.

Восстановите полную резервную копию базы данных в каталог dbrest:

```
prorest sports sports.bak
```

После этого, не запуская и не внося никаких изменений в восстановленную базу данных, выполните восстановление инкрементальной копии командой:

```
prorest sports sports-inc.bak
```

Результат работы команды:

```
Start of extending target DB to needed size... (9432)
This is a incremental backup of /users/valeriy/lecAI/db1/sports.db. (6759)
This backup was taken Mon Aug 10 14:25:04 2009. (6760)
The blocksize is 4096. (6994)
It is based on the full backup of Mon Aug 10 14:24:58 2009. (6761)
It will require a minimum of 1417 blocks to restore. (6763)
Start of restoring the target DB... (9433)
Read 14 db blocks in 00:00:00
```

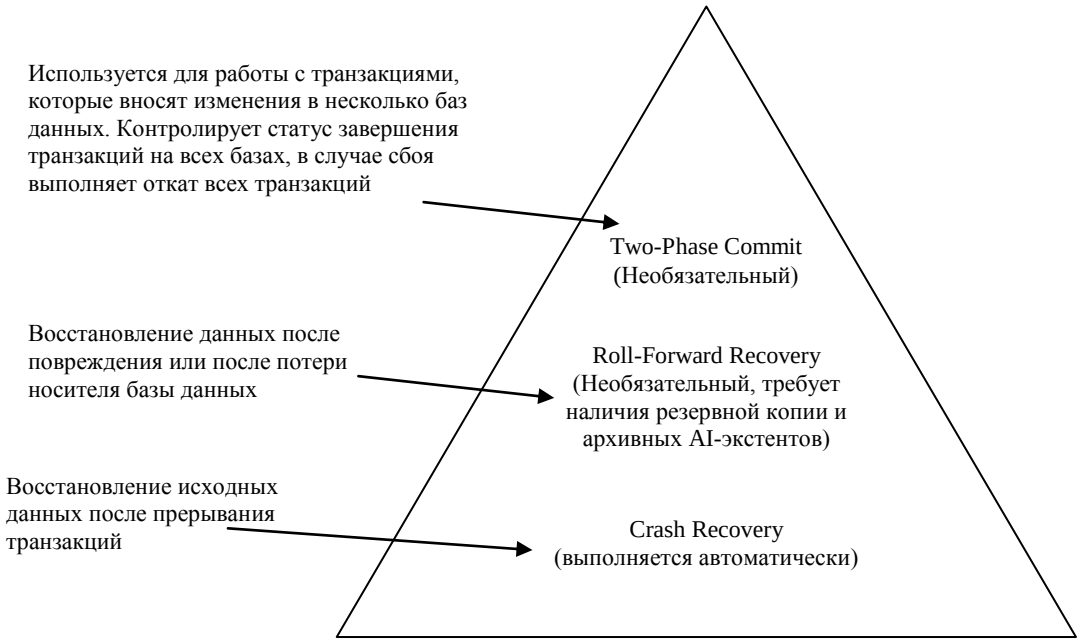
Обратите внимание на сообщения **6759** и **6761**, говорящие о том, что используется инкрементальная копия, которая базируется на предыдущей полной резервной копии, сделанной конкретного числа и в конкретное время. Если попытаться восстановить инкрементальную копию, которая базируется на несоответствующей предыдущей копии, то утилита PROREST выдаст предупреждение и запрос о необходимости продолжения работы. Продолжать восстановление в данной ситуации крайне не рекомендуется, лучше прервать его и воспользоваться соответствующими резервными копиями.

Восстановление после сбоев

База данных OpenEdge имеет три внутренних механизма восстановления после сбоев:

- Crash Recovery
- Roll-Forward Recovery
- Two-Phase Commit

Использование каждого из этих механизмов зависит от типа возникшего сбоя, приоритет их выполнения представлен на рисунке:



Каждый из этих механизмов основывает свою работу на заметках о выполненных изменениях в базе данных, которые хранятся в BI-файле. Например, в базе данных создается одна запись, которая изменила один блок данных. Автоматически создается заметка о выполненных изменениях в BI-файл. Если при этом активирован механизм After-Imaging, то такая же заметка будет записана в текущий AI-файл. Если активирован механизм Two-Phase Commit, то соответствующие записи будут сделаны в TL-файл. Теперь рассмотрим каждый из этих механизмов более детально.

Crash Recovery

Механизм Crash Recovery работает в автоматическом режиме. Для восстановления после системных сбоев он использует информацию из BI-файла, который является активной частью базы данных, поэтому этот файл рассматривается, как неотъемлемая часть любой базы. При выполнении копирования и восстановления вместе с базой копируется и восстанавливается BI-файл. Никогда не удаляйте BI-файл вручную. Когда база данных работает, информация о совершаемых и совершенных транзакциях в базе хранится в трех местах: в базе данных на диске, в оперативной памяти и в BI-файле на диске.

Процесс сохранения изменений в базе выглядит следующим образом. Когда изменяется запись в базе данных, сначала изменения происходят в памяти. Затем, когда транзакция завершена, изменения записываются в BI-файл. Со временем движок базы данных вносит изменения в базу данных на диске. Если на первом этапе произойдет системный сбой, то информация, находящаяся в буфере памяти, будет потеряна. Для восстановления завершенных транзакций будет использоваться информация из BI-файла. Транзакции, которые в момент сбоя не были завершены, будут отменены, и все данные, которые они успели изменить, будут возвращены в исходное состояние на основании информации из BI-файла. Помимо восстановления после сбоев BI-файл также используется для отката транзакций, работа которых была прервана пользователем.

Пример работы механизма. Допустим, выполняется следующая ABL-программа:

```
FOR EACH customer:
        UPDATE name max-credit.
END.
```

Мы изменили записи о клиентах 1 и 2, а во время изменения данных о клиенте 3 произошел системный сбой. Во время перезагрузки базы данных в журнале базы (.lg) появятся следующие сообщения:

```
Single-user session begin for valeriy on /dev/pts/254 (451)
Begin Physical Redo Phase at 256 . (5326)
Physical Redo Phase Completed at blk 800 of 8165 and 31829 (7161)
Begin Physical Undo 1 transactions at blk 800 offset 8189 (7163)
Physical Undo Phase Completed at 1020 . (5331)
Begin Logical Undo Phase, 1 incomplete transactions are being backed out.
(7162)
Logical Undo Phase Complete. (5329)
Single-user session end. (334)
```

В этих сообщениях отмечены фазы механизма Crash Recovery, которые выполняет движок базы данных для ее приведения в состояние, соответствующее состоянию до системного сбоя. Этот механизм отрабатывает каждый раз, когда вы запускаете базу данных, но при этом не все фазы восстановления отображаются в журнале. Например, движок выполняет и регистрирует в журнале фазу Physical Redo, но фазы Physical Undo и Logical Undo будут выполнены и зарегистрированы в журнале только в случае обнаружения незавершенных транзакций.

Если после этого перезапустить программу, то мы увидим, что клиенты 1 и 2 были изменены, а клиент 3 остался без изменений, т.к. на основании BI-файла информация о нём была возвращена в первоначальное состояние.

Механизм Crash Recovery защищает базу от системных сбоев, но он не сможет защитить её от дисковых ошибок. В случае возникновения ошибок носителя или при его полной утере восстановление базы возможно только из резервной копии, после чего необходимо вручную провести все транзакции заново, до момента сбоя, либо воспользоваться механизмом Roll-Forward Recovery, если до сбоя в базе был активирован механизм After-Imaging.

Roll-forward recovery

Механизм Roll-Forward Recovery использует AI-экстенты и резервную копию базы, тем самым позволяя восстановить её после ошибок или после утраты носителя. Для его реализации необходима последняя резервная копия базы данных и все AI-файлы, которые были сформированы после создания этой копии и до момента сбоя. Механизм использует информацию из AI-файлов для последовательного наката транзакций, совершенных после формирования последней резервной копии.

Для использования механизма Roll-Forward Recovery необходимо:

- постоянно выполнять резервное копирование, т.к. копии являются основой работы механизма;
- активировать на базе данных механизм After-Imaging;
- постоянно делать резервные копии сформированных AI-файлов;
- хранить AI-файлы на диске, отличном от того, на котором находятся база данных и BI-файлы.

При активированном механизме After-Imaging информация об изменениях записывается в AI-файлы. Если AI-файлы будут храниться на том же диске, что и база данных или BI-файлы, то в случае повреждения диска вы не сможете восстановить базу данных.

Следующий пример показывает, как AI-файлы используются для восстановления базы данных. Допустим, что выполняется следующая ABL-программа:

```
FOR EACH customer:
        UPDATE name max-credit.
END.
```

Вы изменяете данные клиентов 1 и 2, но в момент изменения данных о клиенте 3 происходит повреждение диска, на котором хранится база данных. База данных не может использовать BI-файл для восстановления транзакций, т.к. он находится в поврежденном состоянии или полностью утрачен. Тем не менее, для восстановления базы можно использовать механизм Roll-Forward Recovery, если был активирован механизм After-Imaging. Иначе мы бы потеряли все внесенные изменения в базу данных с момента формирования последней резервной копии. Это связано с тем, что перед изменением базы данных движок отмечает изменения и сохраняет информацию о них в BI и AI-файлы. Таким образом, AI-файлы будут содержать копии всех завершенных транзакций с момента формирования последней резервной копии. Восстановите последнюю копию базы данных, после чего для воспроизведения всех завершенных транзакций запустите механизм Roll-Forward Recovery[25]. Итак, мы получим полноценную базу данных до момента сбоя.

Восстановление после системных сбоев

За восстановление работоспособности базы данных после системных сбоев отвечает механизм Before-Imaging, который активируется автоматически каждый раз, когда вы запускаете базу данных. Метод восстановления в таких ситуациях зависит от операций, которые выполнялись в момент возникновения сбоя. Далее описаны действия, которые вы должны выполнить, если системный сбой произошел в момент выполнения утилит OpenEdge.

Если системная ошибка произошла во время работы команды RFUTIL ROLL FORWARD, то вы можете для повторной попытки наката AI-файла воспользоваться параметром RETRY команды RFUTIL. Синтаксис команды следующий:

```
rfutil db-name -C roll forward retry
```

С этим параметром команда RFUTIL найдет транзакцию, которая восстанавливалась в момент системного сбоя из AI-файла и продолжит восстановление с нее.

Если системный сбой произошел в момент работы описанных ниже утилит, то просто нужно запустить их повторно.

[25] Дополнительную информацию о работе механизма Roll-Forward Recovery и о восстановлении базы данных из AI-файлов можно найти в книге «Механизм After-Imaging в OpenEdge» на сайте www.openedge.ru

- BUSY
- HOLDER
- PROBKUP
- RFUTIL AIMAGE AIOFF
- RFUTIL AIMAGE BEGIN
- RFUTIL AIMAGE END
- RFUTIL AIMAGE END
- RFUTIL AIMAGE EXTENT EMPTY
- RFUTIL AIMAGE EXTENT FULL
- RFUTIL AIMAGE NEW
- RFUTIL AIMAGE SCAN
- RFUTIL AIMAGE TRUNCATE
- RFUTIL MARK BACKEDUP
- RFUTIL ROLL FORWARD RETRY
- TRUNCATE BI

Если системная ошибка произошла во время работы пользователей с базой данных, и она привела к остановке базы данных, то просто перезапустите базу. В момент запуска активируется механизм Crash Recovery, который восстановит целостность базы на основании информации из BI-файла.

Внимание! Если база данных запущена с параметром Unreliable Buffered I/O (-r) или с параметром No Crash Protection (-i) и в это время произойдёт системный сбой, то её восстановление будет невозможно.

Восстановление после сбоев носителей

Иногда возникают ситуации, когда диск, на котором находилась база данных, поврежден. Может возникнуть ситуация, когда встроенные механизмы восстановления по различным причинам не могут отработать корректно. В этих случаях остается только один выход - восстановление базы данных из резервной копии. Обычно одной только резервной копии недостаточно, т.к. пользователи с момента ее формирования уже успели внести различные изменения в базу, в этом случае на помощь приходят AI-файлы. Поэтому для любой базы данных, в которой хранится значимая информация, должен быть активирован механизм After-Imaging, а формируемые AI-файлы всегда должны храниться отдельно от основной базы данных и от самой резервной копии. Это позволит избежать проблем в тех случаях, когда из строя выйдут диски с базой или носители, на которых хранятся резервные копии. С такой конфигурацией у вас всегда остается возможность безболезненного восстановление данных.

Потеря носителя, содержащего файлы DB и BI.

Если диск, на котором расположены файлы DB или BI, разрушен, или некоторые из них были случайно удалены, то восстановление возможно следующим образом.

- Скопируйте до начала наката текущие AI-файлы и заполненные неархивированные AI-файлы. Это важный шаг, т.к. этим вы защищаете их от возможной случайной перезаписи.
- Создайте структуру базы данных, в которую будет происходить восстановление.
- Восстановите последнюю полную копию базы данных.

- По необходимости восстановите инкрементальные копии.
- Последовательно накатите AI-файлы на последнюю восстановленную копию, используя команду RFUTIL.
- Сформируйте полную копию только что восстановленной базы данных.
- Перезапустите After-Imaging.
- Перезапустите базу данных.

Потеря носителя. содержащего AI-файлы

Следующие шаги выполните в том случае, если диск с AI-файлами был поврежден или эти файлы были случайно удалены.

- Если для базы данных был активирован механизм Two-Phase Commit, то завершите его работу утилитой PROUTIL 2PHASE END.
- Отключите After-Imaging, воспользовавшись утилитой RFUTIL AIMAGE END.
- Сформируйте полную резервную копию базы данных, используя утилиту PROBKUP или с помощью утилит операционной системы.
- Если для копирования использовались утилиты операционной системы, пометьте базу данных, как скопированную с помощью утилиты RFUTIL MARK BACKEDUP.
- Активируйте After-Imaging.
- Активируйте Two-Phase Commit, если он был деактивирован на первом шаге.

Потеря резервной копии

При утрате резервной копии или при ее неработоспособности еще есть шанс восстановить поврежденную базу данных. Следует выполнить последовательный накат AI-файлов на последнюю имеющуюся полную резервную копию.

- Заархивируйте текущий AI-файл, что гарантирует доступность второй копии в случае повреждения исходных экстентов, например, если случайно текущие заполненные AI-файлы будут перезаписаны.
- Восстановите копию, сделанную до потерянной копии. Если используются инкрементальные копии, то сначала восстановите последнюю полную копию, а потом накатите на нее инкрементальные.
- Восстановите AI-файлы из архива до момента формирования потерянной или поврежденной копии.
- Сформируйте резервную копию, следуя стандартной процедуре резервного копирования.
- При необходимости активируйте After-Imaging и Two-Phase Commit.
- Перезапустите базу данных.

Закончилось свободное место на диске

Если свободное место на дисках, где размещаются компоненты базы данных, закончится, то база данных будет аварийно остановлена. В этом случае выполните следующие действия.

- Очистите заполненный диск от ненужных файлов.
- Перезапустите базу данных.

Если возможность очистки диска отсутствует, или освобожденного пространства недостаточно для работы механизма восстановления, то дальнейшие шаги по восстановлению базы данных будут зависеть от того, какие именно компоненты базы данных находились на этом диске.

Переполнение диска, содержащего AI-файлы

При использовании AI-экстентов переменного размера всегда есть риск, что какой-либо из таких экстентов заполнит всё дисковое пространство. В такой ситуации база данных постарается выполнить переключение на следующий по порядку пустой AI-экстент. Если такой экстент будет заполнен (статус FULL), то переключение окажется невозможным и база данных будет вынуждена аварийно остановиться. В такой ситуации:

- сделайте копию самого старшего заполненного AI-экстента, для его определения воспользуйтесь командой RFUTIL AIMAGE EXTENT FULL$
- установите этому экстенту статус EMPTY с помощью команды RFUTIL AIMAGE EXTENT EMPTY, после этого экстент станет доступен для использования.
- Перезапустите базу данных, во время запуска произойдет автоматическое переключение AI-экстентов, нормальная работа базы данных будет восстановлена.

Переполнение диска, содержащего область Primary recovery

- Используя утилиту PROSTRCT ADD, добавьте дополнительный BI-экстент на диск с достаточным свободным пространством. Максимально необходимое свободное место для дополнительного экстента и последующего восстановления должно быть равно минимум размеру заполненного текущего BI-экстента. Заметьте, что рост размера BI-экстента при восстановлении вполне нормальное явление. При добавлении экстента размер последнего BI-экстента будет зафиксирован на уровне его размера в момент останова базы.
- Зайдите в базу данных в однопользовательском режиме. Это позволит отработать механизму Crash Recovery. Выйдите из однопользовательской сессии.
- Перезапустите базу в нормальном режиме.

Команда TRUNCATE BI

Усечение BI-файла - процедура приведения файла к его первоначальному размеру, т.к. при определенных обстоятельствах BI-файл может значительно вырасти в размере. Чрезмерный рост BI-файла, как правило, происходит, когда транзакция стартует и остается активной и открытой в течение длительного периода времени. Так происходит потому, что BI-файл организован из секций, называемых кластерами. Каждый кластер заполняется BI-заметками, и BI-файл расширяется, добавляя новые кластеры для обеспечения хранения этих заметок. Для минимизации возможности роста BI-файла кластеры могут повторно использоваться, но только тогда, когда все транзакции, которым принадлежат BI-заметки, применены к базе данных и закрыты. Поскольку BI-заметки записываются в BI-файл последовательно, то и кластеры могут быть повторно использованы только в последовательном порядке. Это означает, что достаточно одной открытой активной транзакции, чтобы добавление новых кластеров продолжалось, когда другие пользователи открывают и закрывают транзакции.

При усечении BI-файла движок базы данных переносит информацию из него в базу и в AI-файлы, проверяет успешность записи всех данных на диск и очищает содержимое BI-файла, приводя его в первоначальное состояние.

Для усечения используется параметр TRUNCATE BI утилиты PROUTIL. С помощью этого параметра можно не только усекать размер BI-файла, но и менять размер BI-кластера и BI-блока.

Синтаксис команды усечения:

```
proutil db-name –C truncate bi [-G n] | -bi size | -biblocksize size
```

Параметр	Назначение
db-name	База данных, в которой необходимо сделать усечение BI-файла.
-G *n*	Количество секунд, которое ожидает утилита, прежде чем усечь BI-файл. Период по умолчанию - 60 секунд. Указывать меньший период имеет смысл только при обучении и тестировании. Не уменьшайте это значение, если BI-файл будет усечен прежде, чем записи будут сброшены в базу данных и в AI-файлы, то сбой в системе может повредить базу данных.
-bi *size*	Размер кластера в килобайтах. Число должно быть кратным 16, в пределах от 16 до 262128 (16K - 256MB). Размер кластера по умолчанию равен 524K. Если будет использовано значение, не кратное 16, то PROUTIL округлит его до ближайшего, кратного 16.
-biblocksize *size*	Размер BI-блока в каждом буфере (в килобайтах). Правильные значения - 1, 2, 4, 8 и 16. Значение по умолчанию - 8K. Размер BI-блока не может быть меньше размера блока базы данных.

Примечание: в целях улучшения производительности используйте классификатор PROUTIL BIGROW, предназначенный для увеличения количества доступных BI-кластеров до старта базы данных.

Чистка разделяемой памяти

Если брокер базы «умер» или был «убит» средствами, не относящимися к средствам базы данных, например, если вместо PROSHUT использовалась команда операционной системы KILL, то брокер был не в состоянии очистить за собой сегменты выделенной для него разделяемой памяти. Для идентификации неосвобожденных сегментов памяти используется команда PROUTIL с параметром DBIPCS. Для удаления таких сегментов применяется команда операционной системы ipcrm с параметром <-m>.

Утилита PROUTIL DBIPCS показывает сегменты разделяемой памяти, закрепленные за всеми базами данных OpenEdge. Синтаксис команды:

```
proutil -C dbipcs
```

Результат работы команды:

```
      ID   ShMemVer   Seg#     InUse        Database
23757025      12008      0       Yes      /users/valeriy/lecAI/db1/sports.db
```

Назначение полей:

Столбец	Описание
ID	ID разделяемой памяти.
ShMemVer	Версия разделяемой памяти.
Seg#	Номер сегмента разделяемой памяти. Одна база может иметь множество сегментов.
InUse	Статус использования сегмента. Yes или No указывается только для сегментов с номером 0. Для всех других отображается. Для определения, используется ли сегмент, необходимо найти сегмент с номером ноль и посмотреть его статус.
Database	Указывается полный путь к базе данных, которой принадлежит сегмент.

Восстановление области Control Area

Мы уже знаем, что область Control Area содержит информацию о размещении областей хранения базы данных и их экстентах. Поэтому, если контрольная область потеряна или повреждена (файл с расширением .db), используйте для её восстановления команду PROSTRCT BUILDDB. Она восстановит контрольную область из имеющегося структурного файла (.st). Синтаксис команды:

```
prostrct builddb db-name structure-description-file
```

По умолчанию, если не указан параметр *structure-description-file*, т.е. если вы не укажете собственный структурный файл, то утилита будет использовать для восстановления структурный файл самой базы данных с именем `db-name.st`.

Разблокировка поврежденной базы данных

В базе данных может возникнуть несоответствие между экстентами из-за неправильного использования системных утилит копирования или некорректного использования утилит резервного копирования и восстановления. Движок базы синхронизирует файлы DB, BI и AI, гарантируя их согласованность. Если он обнаруживает несоответствие в этих файлах, он выводит сообщение об ошибке и пресекает любые попытки открыть базу, т.е. база данных блокируется. Если это произошло, то попытайтесь восстановить базу данных из резервной копии. В самом крайнем случае, чтобы разблокировать базу и выгрузить данные, вы можете использовать утилиту PROSTRCT с классификатором UNLOCK. Используйте этот классификатор только и только в том случае, если больше ничего нельзя сделать для восстановления работоспособности базы данных. В результате работы утилиты заблокированная база данных может и не открыться, еще хуже, что результатом разблокировки может стать ее разрушение, поэтому предварительно сделайте копию базы любыми возможными способами. Для разблокировки поврежденной базы выполните следующее.

- Выполните команду:

```
prostrct unlock db-name
```

- Выгрузите данные из разблокированной базы.
- Выйдите из базы.
- Создайте новую базу данных и загрузите в нее рабочую схему поврежденной базы данных.
- Запустите новую базу и загрузите в нее данные, выгруженные из поврежденной базы данных.

Принудительный доступ к поврежденной базе данных

Если восстановить доступ к базе данных не представляется возможным, остается только восстановить ее из резервной копии. Ну, а если копии нет, и больше не существует альтернативных путей, то можно попытаться получить доступ к поврежденной базе, используя параметр <-F> утилиты PROUTIL с классификатором TRUNCATE BI.

Внимание! Используйте этот параметр только в качестве последнего шанса, т.к. он может привести к нарушению целостности базы.

Для принудительного доступа к базе данных выполните следующее.

- Создайте резервную копию базы данных любым возможным способом. Имейте в виду, что работа утилиты PROBKUP может завершиться аварийно, поскольку блоки базы данных могут быть повреждены, и в этом случае вам придется воспользоваться утилитами операционной системы.
- Только после того, как вы убедитесь, что имеется копия базы, можно воспользоваться параметром <-F> в составе команды PROUTIL TRUNCATE BI:

  ```
  proutil db-name -C truncate bi -F
  ```

  Вы получите сообщение:

  ```
  The -F option has been specified to proutil. (6260)
  Forcing into the database skips database recovery. (6261)
  This leaves the database in an unknown state, considered damaged. (6262)
  Are you sure you want to skip crash recovery? (Y/N) (6263)
  ```

  Утилита предупредит о возможных последствиях использования параметра <-F> и запросит подтверждение продолжения работы. Если вы ответите согласием, то утилита выведет следующие сообщения, вы получите доступ к базе данных:

  ```
  ** The FORCE option was given, database recovery will be skipped. (33)
  After-imaging disabled by the FORCE parameter. (4790)
  ** Your database was damaged. Dump its data and reload it. (37)
  ```

  Обратите внимание, что принудительный доступ отменяет работу механизма Crash Recovery и деактивирует работу механизма After-Imaging. С этого момента база данных будет считаться поврежденной, и при ее запуске вы всегда будете получать сообщение **37**.

- Теперь запустите базу данных и выполните выгрузку данных с последующей загрузкой во вновь созданную базу.

Флаг повреждения базы данных в отдельных случаях можно снять обычной переиндексацией базы данных, но, в зависимости от характера повреждений, это еще не означает, что целостность базы данных будет восстановлена.

Что происходит при принудительном доступе к базе данных? В случае аварийной остановки базы данных, все резидентные данные, находящиеся в памяти, будут потеряны вместе со всеми изменениями, которые еще не были записаны на диск. Когда база будет запущена вновь, начнется процесс, называемый Rollback. Этот процесс состоит из двух этапов.

Redo Phase – на этом этапе выполняется сканирование двух последних используемых BI-кластеров для проверки того, что все завершенные транзакции сброшены в файлы данных, т.к. некоторые транзакции, которые были завершены во время возникновения сбоя, могли быть записаны в BI-файл, но не записаны в файлы данных (.d1, d2 и т.д). Старт и завершение Redo Phase отмечается сообщениями **5326** и **7161**, которые выводятся во время запуска базы данных:

```
Begin Physical Redo Phase at <address>. (5326)
Physical Redo Phase Completed at blk <blk> off <off> upd <upd>.(7161)
```

Rollback - по завершению Redo Phase движок выполнит обратное сканирование BI-файла на предмет наличия незавершенных транзакций в момент возникновения сбоя. Любая незавершенная транзакция благодаря информации в самом BI-файле будет откачена назад (rolled back).

Когда к базе был применен принудительный доступ с параметром <-F>, оба этапа Redo Phase и Rollback пропускаются, BI-файл перестраивается. Таким образом, информация, необходимая для обоих этапов, считается потерянной. Поэтому нетрудно понять, что после этого база данных будет находиться в неопределенном состоянии, поскольку пропущен этап Redo Phase и все транзакции, завершенные в момент сбоя, сохранены в файлы данных не будут, или будут сохранены частично. Частичное сохранение транзакции еще более нежелательно, чем полная потеря этой транзакции. Это основная причина, почему база данных будет разрушена при использовании <-F>, даже если не было никаких активных транзакций в момент сбоя. Поскольку пропущен этап Rollback, то незавершенные к моменту сбоя транзакции не будут отменены, а это влечет за собой следующие последствия:

- не полностью будут обновлены записи;
- записи могут быть частично удалены;
- могут появиться фрагменты уже несуществующих записей;
- возможно наличие неиндексированных записей;
- могут появиться индексы, ссылающиеся на несуществующие записи;
- могут возникнуть противоречия в цепочках Free и RM, которые используются для размещения новых записей;
- могут появиться логические противоречия из-за частичного завершения транзакций.

При этом повреждения в базе данных могут проявиться не сразу, а только спустя некоторое время, например, при обращении к несуществующей записи через пару месяцев, что сможет весьма осложнить определение причин некорректного поведения приложения.

Физическое несоответствие структуры индексов может быть восстановлено с помощью выполнения полной переиндексации базы данных в offline. Это возможно потому, что переиндексация сначала удаляет всю информацию о существующих индексах, а затем строит индексное дерево заново по существующим записям в таблицах. Возможно, что индексы могут быть построены некорректно, что не позволит получить полноценный доступ к данным во время их выгрузки. Это самый редкий случай, но возможный. В любом случае, целостность записей данных не может быть восстановлена таким образом. Это возможно только путем ручного удаления поврежденных данных. Логическое несоответствие может быть исправлено только путем проверки всех значений во всех

записях на их корректность и отсутствие пропущенных данных, а так же на отсутствие дубликатов. Такие проверки обычно весьма трудоемки.

Исходя из всего этого можно сказать, что восстановление базы данных из резервной копии, это более надежный и приемлемый способ, чем использование параметра <-F>.

Глава 13 Механизм Before-Imaging

Механизм Before-imaging обеспечивает откат транзакций в случае, когда происходит сбой в базе данных. Этот механизм чрезвычайно важен для обеспечения надежности работы базы, но он также генерирует значимую дисковую нагрузку ввода/вывода (I/O), которая влияет на производительность, в связи с чем этот механизм может создать узкое место в системе. Движок базы данных всегда записывает изменения в BI-файлы перед тем, как записать их в базу данных и в AI-файлы. Если BI-активность создает высокую I/O-нагрузку, то это влияет на всю базу данных.

Уменьшить влияние BI-активности на производительность можно следующими способами:

- перемещением BI-файла на отдельный диск;
- запуском процесса BIW (если установлена лицензия OE Enterprise RDBMS);
- увеличением количества BI-буферов;
- увеличением размера BI-кластера;
- увеличением размера BI-блока;
- отсрочкой BI-записи.

Перед принятием решения о выполнении каких-либо настроек вернемся к основам и вспомним, что собой представляет Before-Image файл, и как он работает.

Зачем нужен Before-Image?

Файл Before-Image, называемый так же BI-файл, содержит Primary transaction log (производный термин от «undo-redo log»[26]) базы данных. Это существенная часть каждой базы данных, которая столь же важна, как и экстенты с таблицами и индексами. Без BI-экстентов с базой данных работать нельзя. Если случится так, что BI-файл окажется потерянным или поврежденным, то база данных будет считаться поврежденной, и единственным способом восстановления такой базы будет восстановление её из последней резервной копии с последовательным накатом AI-файлов.

Хотя мы и называем этот файл «Before-Image», на самом деле данные, сохраняемые в него, не совсем подходят под определение before-images (исходный вид). Это название историческое и, возможно, подобрано было неудачно. Файл BI содержит записи, называемые «BI-заметками», которые позволяют восстановить или отменить изменения, сделанные в базе данных.

В основном BI-файл используется для достижения двух целей - для восстановления после сбоя (crash recovery), и для отката (rollback) транзакций в процессе нормальной работы. Кроме того он совместно с менеджером памяти обеспечивает различные варианты улучшения производительности, позволяя оперативно изменять блоки памяти, не требуя их записи в экстенты базы данных. Существует правило управления буферами, называемое «no-force, steal». Оно означает, что измененные буферы не должны быть сброшены на диск во время завершения транзакции, но могут быть записаны на диск во время работы другой транзакции, по мере необходимости освобождения места для считывания блоков, которые отсутствуют в буферном пуле. При нормальном процессе работы из экстентов базы данных считываются копии блоков данных, которые

[26] Undo-redo log – журнал регистрации отмен и восстановлений.

помещаются в буферный пул, если ранее они не были обнаружены в нём. Когда транзакция выполняет изменения базы, то изменения осуществляются только в памяти, т.е. изменяются блоки в буферном пуле. В конечном счете, изменённые блоки должны быть записаны в экстенты данных на диске, но менеджер памяти обычно откладывает операции записи на как можно долгий период, иногда он может длиться несколько часов или дней. Одна из важных причин такой отсрочки, это то, что блок может еще неоднократно подвергаться изменениям. Таким образом, сокращается частота записи на диск. Обычно, измененные блоки записываются на диск асинхронным автоматическим механизмом контрольных точек (checkpoints), или когда база данных будет остановлена.

Что произойдет с изменениями, сделанными в памяти, в случае возникновения системного сбоя? Они будут потеряны? Они будут восстановлены на основании информации, хранящейся в BI-файле. Каждое изменение блока данных сначала записывается в качестве заметок в буферы BI-файла. Причем для некоторых типов изменений может быть сгенерировано множество различных заметок. Заметки из BI-буферов должны быть сброшены на диск только после завершения транзакции (committed).

Заметки, записываемые в BI-файл, содержат номер области хранения, количество блоков, и прочие данные, которые используются для идентификации изменений, поэтому BI-файл оказывается в сильной связи со своей базой данных, он может быть использован только со своей базой.

Менеджер хранения тщательно следует правилу, которое обычно называют «write-ahead logging rule» или WAL. Согласно этому правилу ни один измененный блок базы данных не будет записан в экстент базы на диске, прежде чем он не будет записан в BI-файл. Таким образом, все изменения на диске будут дублироваться в BI-файл. В Before-Image доступны только описания изменений. Это потому, что пространство в файле лога используется многократно. При этом должно гарантироваться, что данные, которое будут перезаписаны, больше не будут нужны.

Если изменение будет потеряно, то для его восстановления мы сможем использовать заметки в transaction log. Это и есть причина того, почему лог называется «undo-redo log», т.е. он позволяет нам восстановить изменения, которые были сделаны ранее. Redo так же называют «repeating history» или «replaying transaction». Можно использовать те же заметки, чтобы отменить изменения, которые были сделаны незавершенной (uncommitted) транзакцией.

Правило WAL позволяет изменять блоки данных в памяти без немедленной их записи на диск. Кроме того, необязательно записывать блоки на диск даже, если транзакция будет завершена. Вместо этого заметки с изменениями просто пишутся в `transaction log`. В результате экстенты данных, находящиеся на диске, никогда не будут соответствовать реальному состоянию, пока система работает. Конечное и правильное состояние базы данных состоит из трех различных частей:

1. блоки на диске находятся в различных экстентах данных;
2. более новые версии некоторых из этих блоков находятся в памяти;
3. заметки с изменениями находятся в Before-Image log.

Данных в Before-Image log и данных в экстентах достаточно, чтобы воссоздать изменения, находящиеся в буферном пуле в памяти. Это означает, что ничего не произойдет, если мы потеряем данные в памяти, т.е. если в случае сбоя системы будут потеряны измененные блоки. Тем не менее, если в результате сбоя был потерян BI-файл, например, диск на

котором он был расположен, оказался поврежденным, то тогда восстановить изменения, которые находились в буферном пуле и не были записаны на диск, будет невозможно. **Внимание!** Восстановить данные в BI-файле невозможно никаким способом!

Если вы используете механизм After-Imaging, то это значительно облегчит восстановление поврежденной базы данных, т.е. BI-файла и экстентов с данными. Файлы After-Image содержат вторичную копию всех заметок об изменениях, которые записывались в BI-файл. Поэтому, восстановив базу из последней резервной копии и последовательно накатив все AI-экстенты, сохранённые с момента создания этой резервной копии, вы получите точную копию базы данных на момент, предшествующий сбою.

При нормальной работе транзакции изменяют базу данных, генерируются заметки обо всех изменениях, которые сбрасываются в BI-буферы, после чего записываются в BI-файл. Если происходит сбой, то содержимое разделяемой памяти теряется. Допустим, что системный сбой не был причиной аварии, приведшей к потере и повреждению данных на диске, и у нас на диске остался только BI-файл и устаревшие экстенты данных. Когда база данных будет перезапущена после такого сбоя, менеджер восстановления (recovery manager) начнет считывать заметки из Before-Image и восстанавливать все изменения, которые не были записаны на диск. Этот процесс называется Crash Recovery. Что бы облегчить процесс каждый блок данных содержит номер версии, который увеличивается каждый раз, когда блок изменяется. Номер версии и номер блока содержатся в BI-заметке, описывающей изменения. Менеджер восстановления считывает BI-заметки в порядке их записи. Он сравнивает номер версии в заметке с номером версии в блоке, при необходимости считывает блок в буферный пул с диска. Если версия блока младше версии в заметке, это означает, что изменения, описанные в ней, уже были сделаны. Это возможно только не для всех изменений. Например, измененный блок может быть записан на диск с целью освобождения места для другого блока, считываемого в память, поэтому такой блок будет пропущен, и чтение продолжится далее. Если версии совпадают, то делается изменение, и увеличивается номер версии блока. Заметьте, что такая схема требует, чтобы все изменения были восстановлены в том же порядке, в каком они были сделаны, ни одно из них не может быть пропущено. Каждое изменение записи является описанием того, как изменялся блок от версии «n» к версии «n + 1». Ситуация с обратным откатом более сложна и здесь не описывается. Как только будет достигнут конец BI-файла, можно считать, что воспроизведены все потерянные изменения. Большинство измененных блоков будет находиться в памяти в буферном пуле базы данных, но некоторые блоки, возможно, будут записаны на диск в экстенты данных, если, например, понадобилось освобождение места для чтения других блоков. После считывания всех заметок мы сможем откатить (rollback) все незавершенные транзакции и завершить восстановление. Обратите внимание, что каждая транзакция в памяти так же регистрируется в BI-фале, следовательно, она тоже будет восстановлена, по мере считывания заметок.

Когда изменённые блоки записываются в экстенты данных?

Измененные блоки, находящиеся в буферном пуле базы данных, записываются в связанные с ними экстенты в следующих случаях:

- когда APW-процесс обрабатывает измененные блоки в очереди контрольной точки;
- когда необходим блок, которого нет в буферном пуле, в этом случае измененный блок будет записан на диск для освобождения места;
- когда APW-процесс записывает измененный блок из APW-очереди;

- когда APW-процесс не выполняет полезной работы, и поэтому просматривает список измененных блоков и периодически записывает их;
- после запуска online backup;
- когда выполняется нормальный останов базы данных.

Поскольку мы придерживаемся WAL-правила, то во всех случаях все заметки, описывающие изменения, должны быть записаны в transaction log прежде, чем блоки данных будут записаны на диск.

В первых пяти случаях база данных используется, и обычно в это время создаются новые изменения. Нет никакого способа, чтобы узнать, какие экстенты данных были обновлены, а какие нет. В последнем случае, даже если будут иметься небольшие изменения, также нельзя определить, какие блоки были записаны, даже если база простаивала в течение нескольких часов.

Обратите внимание, если вы не используете APW-процессы, то ситуация будет немного другая. Без этих процессов блоки будут записываться:

- если процесс, генерирующий заметки, обнаруживает, что текущий кластер заполнен, то перед инициализацией следующего кластера все измененные блоки в очереди контрольной точки будут записаны;
- если потребуется блок, отсутствующий в буферном пуле, то предварительно для освобождения места на диск будет сброшен какой-нибудь измененный блок из буфера;
- после запуска online backup утилита копирования записывает все измененные блоки в буферном пуле;
- если выполняется нормальный останов базы данных.

Без APW-процессов время, необходимое для записи, понадобится большее, чем с использованием APW.

Экстенты данных, которые постоянно находятся на диске, могут считаться актуальными только тогда, когда удовлетворены все следующие условия.

- База данных не находится в многопользовательском или однопользовательском режимах, а также относительно неё не используется ни одна утилита или программа.
- База данных остановлена нормальным способом.
- Все ранее сброшенные блоки записаны на диск из буферного пула операционной системы самой операционной системой. Достаточно трудно сказать, когда это происходит. Если вы используете параметр запуска <-directio> или версию openEdge V10, то после успешной остановки базы это будет истинно. Иначе, нельзя с уверенностью сказать, что в буферном пуле операционной системы не осталось несброшенных на диск блоков. Во время остановки базы после записи всех измененных блоков менеджер памяти генерирует системный вызов sync, чтобы указать операционной системе о необходимости сброса на диск его буферов, включая данные, не связанные с базой. На некоторых операционных системах этот системный вызов завершается до того, как операционная система завершит запись. Вы можете проверить, выполняется ли sync синхронно, выполнив один эксперимент. В то время как система активна, и записывается много файлов, наберите в командной строке команду `time sync`, запустите её несколько раз и

посмотрите, как долго она будет выполняться. Если команда завершается немедленно, то, вероятно, что sync работает не синхронно. Если для завершения потребовалось некоторое время, от нескольких секунд до нескольких минут, а последующие запуски sync завершаются быстро, то это означает, что sync работает синхронно, и все записи на диск были завершены до возвращения управления в командную строку. Этот эксперимент работает, только если в системном буферном пуле действительно имеется много изменений файловой системы. Менеджер памяти перед окончательной остановкой ожидает некоторое время, чтобы дать возможность системе завершить все операции записи. По умолчанию это время равно 60-ти секундам (в последних версиях OpenEdge оно равно нулю), но иногда его бывает недостаточно. Для увеличения времени вы можете воспользоваться параметром <-G>. Помните, если произойдет системный сбой до того, как операционная система завершит запись, то все изменения будут потеряны.

- Не должно быть никаких незаписанных блоков данных в буферах контроллеров дисков, в буфере диска непосредственно или в буфере дисковой подсистемы (например, в дисковом массиве). Если вы не используете достаточно надежной системы хранения с резервным питанием, то вы должны настроить диски так, чтобы не выполнялась буферизация данных.

Когда база данных запускается после нормальной остановки, она так же выполняет некое подобие Crash Recovery. По некоторым причинам этот процесс называется «warm start». Этот процесс читает transaction log и проверяет, что ни одно из изменений не было потеряно после остановки.

Если всё вышесказанное выполнено, то тогда вы можете быть уверенными, что экстенты базы содержат актуальные данные.

Пример сбойной ситуации

Для пропуска процесса Crash Recovery можно использовать команду PROUTIL TRUNCATE BI с параметром <-F>, тем самым игнорируя содержание BI-файла. Пропуск Crash Recovery означает, что записи в Before-Image не будут считаны и обработаны, все изменения в базе данных будут потеряны без возможности восстановления, и все незавершенные транзакции не откатятся. В результате выполнения этой операции вероятно, что база данных будет повреждена. Эти повреждения проявятся не сразу. Возможно, что вы сразу сможете открыть базу данных и получить доступ к данным, но не ко всем из них. Следующий пример показывает, что же произойдет при использовании этого параметра. Здесь некоторые детали будут опущены, т.к. они не так важны.

Рассмотрим следующую последовательность событий. Допустим, что у вас есть очень маленький буферный пул, содержащий четыре буфера базы, и вы собираетесь изменить одну запись, после чего хотите прочитать еще одну.

Для изменения некоторой записи нам нужно прочитать её из экстента на диске. Сначала мы читаем корневой (root) индексный блок в один из четырех буферов. Предположим, что индекс имеет два уровня, тогда мы так же должны прочитать второй индексный блок.

Затем мы считываем блок данных, чтобы выбрать запись для изменения. Но наша запись состоит из двух фрагментов, поэтому нам нужно считать и другой блок данных.

В этом месте каждый из четырех блоков в буферном пуле содержит один из прочитанных блоков, т.е. два индексных блока и два блока с записями.

Теперь, когда мы изменим запись, то каждый из двух фрагментов будет изменен, тем самым оба блока данных получат статус измененных (так называемые «грязные блоки»). В transaction log будет сгенерирована BI-заметка для каждого изменения.

После завершения изменения мы получим ситуацию, когда в BI-файле есть 4 заметки:

- заметку о начале транзакции;
- заметку об изменении первого фрагмента записи;
- заметку об изменении второго фрагмента записи;
- заметку о завершении транзакции.

На диск еще ничего не записывалось, т.е. мы имеем его в том же состоянии, с которого и начинали.

В буферном пуле имеются два «грязных» блока, которые не записаны на диск, но должны быть записаны в будущем.

Теперь мы читаем вторую запись, и снова считываем два индексных блока. Поскольку предыдущие индексные блоки будут самые старые, они будут заменены новыми блоками, и перемещены в начало LRU-цепочки.

Затем мы считываем один блок данных, заменяя им один предыдущий измененный блок, который записывается на диск. Мы не изменяем эту запись сейчас.

В этой точке времени мы имеем:

- четыре заметки от предыдущего изменения;
- один измененный блок, записанный на диск;
- в памяти у нас два индексных блока - один неизмененный блок и один измененный блок данных.

Измененный блок содержит один из двух фрагментов от измененной ранее записи. Неизмененный блок содержит вторую запись, которую мы считали, но не изменили.

Теперь предположите, что произошел системный сбой, т.к. как некто отключил сервер, чтобы включить пылесос. Такие случаи действительно имели место быть! Но сбой может произойти из-за чего угодно, может быть, вместо пылесоса хотели подключить laptop.
Итак, после перезагрузки сервера мы имеем следующее:

- в BI-файле есть четыре BI-заметки с изменениями;
- один из двух измененных блоков записан на диск;
- остальные данные, которые были в экстентах базы до начала нашего упражнения;
- всё, что было в памяти, мы потеряли, включая один из наших измененных блоков, единственное доказательство того, что блок был измен, находится в Before-Image log.

Если мы запустим базу данных нормальным способом, то будет выполнен процесс Crash Recovery, который считает Before-Image log и обнаружит, что один из измененных блоков был потерян, после чего воссоздаст его снова. По завершению Crash Recovery мы снова будем иметь два измененных блока. Это весьма хорошо! Но если вместо этого мы усечем BI-файл утилитой PROUTIL с параметром <-F>, то все BI-заметки будут проигнорированы. Единственное доказательство существования измененного блока, не записанного на диск, будет потеряно. Мы останемся только с тем, что есть на диске в базе данных. Мы будем иметь одну половину измененной записи и вторую половину со старой версией. Или, возможно, что и нет никакой старой версии другой половины, поскольку оригинальная запись была в одном блоке, но в модифицированном блоке её не было.

Восстановление индекса не поможет восстановить поврежденную запись. Нигде нет никакой доступной нам информации, которая может быть использована для её восстановления, т.е. вы зря потратите время.

Когда таким образом вы пропускаете Crash Recovery, в базе данных устанавливается флаг damaged. Каждый раз, когда вы будете запускать базу данных, вы будете получать сообщение о том, что база данных повреждена и необходимо выполнить перезагрузку (dump/load) данных. Если причиненный базе ущерб будет достаточно серьезным, то и это не будет возможным. Даже если это и будет возможно, то вероятнее всего вы потеряете некоторые данные.

Вывод: Before-Image файл - это такая же часть базы данных, как и её экстенты. Если вы теряете экстент данных, то вы должны восстановить базу из резервной копии. Если вы потеряли BI- экстент, то вы также должны восстановить базу из резервной копии. Нет ни одного способа восстановления, гарантирующего, что база данных останется в надежном и актуальном состоянии.

Использование BIW

Процесс BIW[27] - это фоновый процесс, постоянно записывающий заполненные BI-буферы на диск. Поскольку запись происходит в фоновом режиме, клиентским процессам редко приходится ожидать, пока заполненный буфер будет сброшен на диск. BIW-процесс - необязательная, но очень рекомендуемая опция для улучшения производительности дискового ввода/вывода.

Сервер записывает текущую информацию в BI-файл через буфер вывода. Когда этот буфер заполняется, сервер размещает буфер в цепочке заполненных буферов. После чего он берет новый буфер из цепочки пустых буферов, и использует его в качестве текущего буфера вывода. Если пустые буферы оказываются недоступны, то процесс вынужден ждать, пока заполненные буферы не будут записаны на диск. Этим и занимается BIW-процесс, т.е. записью заполненных буферов на диск и размещением их в цепочке пустых буферов. Таким образом, он гарантирует наличие доступных пустых буферов.

[27] BIW, от англ. Before-Image Writer

Увеличение количества BI-буферов

Набор BI-буферов используется для кэширования в памяти информации о транзакциях. Эти буферы впоследствии сбрасываются в BI-файл. Любое изменение состояния базы данных помещается в виде BI-заметки сначала в BI-буферы в памяти, а затем записываются в сам BI-файл. Обычно эти операции выполняет BIW-процесс. Наличие достаточного количества BI-буферов позволяет этому процессу получить необходимое время для осуществления записи. Если пустых буферов будет недостаточно, то любой процесс, который выполняет изменение в базе данных, вынужден ждать, пока такие буферы появятся. Таким образом, увеличивая количество BI-буферов, мы снижаем вероятность их ожидания процессами, выполняющими изменение.

По умолчанию количество BI-буферов равно пяти, но с помощью параметра запуска базы данных <-bibufs> это количество можно увеличить. Рекомендуемое начальное значение равно 25, после чего необходимо осуществлять мониторинг доступности пустых BI-буферов с помощью утилиты Promon. Для этого в Promon используйте экран «Activity: BI Log» (**R&D → 2 → 5**). На представленном экране нас интересует строка Empty buffer waits.

```
01/25/10        Activity: BI Log
17:34:10        01/22/10 13:12 to 01/25/10 17:11 (75 hrs 58 min)

                       Total   Per Min    Per Sec      Per Tx

Total BI writes          929         0       0.00        0.41
BIW BI writes            922         0       0.00        0.41
Records written        38443         8       0.14       16.96
Bytes written        7533659      1653      27.55     3323.18
Total BI Reads             9         0       0.00        0.00
Records read              16         0       0.00        0.01
Bytes read              3265         1       0.01        1.44
Clusters closed            0         0       0.00        0.00
Busy buffer waits         91         0       0.00        0.04
Empty buffer waits        83         0       0.00        0.04
Log force waits            0         0       0.00        0.00
Log force writes           0         0       0.00        0.00
Partial writes             7         0       0.00        0.00
Input buffer hits          0         0       0.00        0.00
Output buffer hits         9         0       0.00        0.00
Mod buffer hits           25         0       0.00        0.01
BO buffer hits             0         0       0.00        0.00
```

Если значение Empty buffer waits больше нуля в секунду, то количество BI-буферов следует увеличить. Стоит заметить, что если вы по каким-либо причинам не используете BIW-процесс, то изменение количества BI-буферов не приведет к значимому результату, также эффект не будет достигнут, если диск, на котором расположен BI-файл, перегружен, т.к. он будет не в состоянии своевременно выполнять необходимые операции записи.

Начиная с версии OpenEdge 10.1C появилась возможность увеличивать количество BI-буферов без необходимости остановки базы данных. Воспользуйтесь утилитой PROUTIL с параметром INCREASETO:

```
proutil sports -C increaset -bibufs 25
```

Помните, уменьшить их количество в режиме, когда база данных работает, невозможно. Стоит заметить, что если после увеличения количества BI-буферов полученный результат

вас устроил, то не забудьте установить новое значение в параметрах запуска базы данных. Еще одно замечание - если в базе данных включен механизм After-Imaging, то после изменения количества BI-буферов рекомендуется изменить и количество AI-буферов, т.е. рекомендуется устанавливать значения параметров запуска базы данных <-bibufs> и <-aibufs> одинаковыми.

Увеличение размера BI-кластера

Файл BI организован из кластеров, которые заполняются, когда в этот файл пишутся данные. Как только очередной кластер заполнен, база данных все измененные блоки записавает на диск. Этот процесс записи называется контрольной точкой. Контрольные точки позволяют уменьшить время восстановления в случае сбоя и дают возможность движку базы многократно использовать отведенное для BI дисковое пространство. Увеличение размера BI-кластера приведет к увеличению интервала между контрольными точками, что дает APW-процессам необходимое время для их работы. Таким образом, вместо того, чтобы все блоки за один раз сбрасывались на диск в контрольной точке, значительно увеличивая нагрузку на диск, они будут равномерно записываться APW-процессами. Это значит, что в момент формирования контрольных точек никакие блоки не должны быть доступны для записи на диск или их может быть незначительное количество. Для выполнения мониторинга частоты формирования контрольных точек можно воспользоваться утилитой Promon и её экраном Checkpoints (**R&D → 3 → 4**):

```
01/25/10        Checkpoints
17:11:11

Ckpt                          ------ Database Writes ------
 No. Time         Len    Freq   Dirty   CPT Q    Scan   APW Q Flushes
1860 12:46:44    4104   11152      58      16     882       0       0
1859 12:37:14     510     570     300     311     497       9       0
1858 12:35:34      11     100      52       0       6     512       0
```

Время между формированием контрольных точек должно быть не менее 60-ти секунд, а лучше 5 минут. Обычно диапазон размеров кластера должен составлять от 2 Мб до 16 Мб. По умолчанию его размер равен 512 Кб. При работе с лицензий OE Workgroup RDBMS размер кластера следует оставить небольшим. В этом случае значение 512 КБ или меньшее будет лучше, чем кластеры большего размера. Поскольку с этой лицензией нельзя запускать фоновые процессы, то все модифицированные блоки данных будут записываться на диск именно в контрольных точках. Чем больше размер кластера, тем больше измененных блоков накопится к этому моменту, и запись этих блоков на диск будет приводить к периодическому «зависанию» базы данных.

Вычисление оптимального размера кластера - это циклический процесс, который выполняется относительно промышленной базы данных, работающей в нормальном режиме, т.е. при обычной нагрузке.

Пример подбора размера BI-кластера. Предположим, что желаемый минимум между контрольными точками 120 секунд. Допустим, что в Promon между какими-нибудь двумя контрольными точками расстояние по времени равно 30 секунд, смотрите поле LEN вышеупомянутого экрана Checkpoints. Для увеличения этого интервала до 120 секунд необходимо увеличить размер кластера. Разделим желаемое время на реальное время: 120/30 = 4. Таким образом, размер кластера необходимо увеличить в четыре раза по отношению к текущему его размеру, т.е. если размер кластер был равен 512Кб, то 512 * 4 = 2048 Кб = 2Мб.

Увеличение размера кластера должно выполняться, когда база данных будет остановлена. Сделать это изменение можно командой:

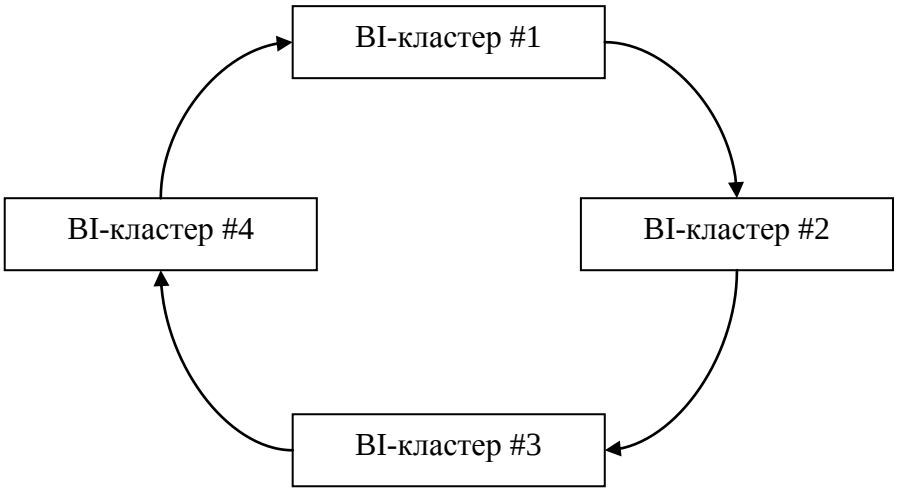
```
proutil db-name -C truncate bi -bi <новый размер
кластера в Kb>
```

Внимание! Чрезмерное увеличение размера BI-кластера может привести к ухудшению производительности вместо желаемого улучшения. Дополнительно ко всему необходимо отслеживать значение поля Flushed at checkpoint. В идеале это значение должно стремиться к нулю. Если количество буферов, сброшенных на диск в контрольной точке, будет увеличиваться, то вероятнее всего вам стоит запустить дополнительный APW-процесс.

```
01/25/10        Activity: Page Writers
17:22:08        01/18/10 09:02 to 01/25/10 17:14 (176 hrs 12 min)

                          Total    Per Min    Per Sec         Per Tx

Total DB writes          505971         48       0.80           0.03
APW DB writes            502142         47       0.79           0.03
    scan writes           57689          5       0.09           0.00
    apw queue writes       6376          1       0.01           0.00
    ckp queue writes     438077         41       0.69           0.02
    scan cycles             986          0       0.00           0.00
    buffers scanned     2962932        280       4.67           0.16
    bufs checkpointed    460848         44       0.73           0.02
Checkpoints                1855          0       0.00           0.00
Marked at checkpoint     461276         44       0.73           0.02
Flushed at checkpoint       862          0       0.00           0.00
```

Увеличение количества BI-кластеров

По умолчанию в BI-файле создается четыре кластера. База данных работает с этими кластерами по кругу, как только заполняется текущий кластер, формируется контрольная точка и выполняется переключение на следующий по порядку кластер. Пока следующий кластер заполняется, информация из предыдущего кластера сбрасывается в базу данных.



Иногда возникают ситуации, в которых использовать следующий по порядку кластер не возможно, т.к. он по-прежнему содержит активную транзакцию, и его содержимое еще не сброшено в базу данных. Тогда автоматически будет создан новый кластер, который и будет использоваться для записи изменений. При этом в момент создания нового кластера

ни один процесс не сможет внести какие-либо изменения в базу данных, и он будет вынужден ждать. Это, несомненно, не может не отразиться на производительности, особенно если размеры кластера большие.

Но со временем, если, конечно же, увеличение количества кластеров не было вызвано аномальной транзакцией, их количество установится в оптимальное значение. Для того чтобы узнать, какое количество кластеров у вас сейчас, необходимо разделить текущий физический размер BI-файла на размер BI-кластера. Это и будет количество BI-кластеров, которые необходимы вашей базе данных. Зная это значение, вы можете в следующий раз после выполнения процедуры усечения BI-файла самостоятельно нарастить BI-файл до нужного размера, тем самым вы предотвращаете создание новых кластеров в процессе работы базы данных. Увеличить количество BI-кластеров можно командой:

```
proutil db-name -C bigrow n
```

При этом помните, что выполнять эту операцию следует до запуска базы данных, после усечения BI-файла. Усечение BI-файла происходит в следующих случаях:

- активируется механизм After-Imaging;
- запускается переиндексация;
- выполняется ручное усечение.

Когда вы наблюдаете в базе данных чрезмерный рост размера BI-файла за короткий промежуток времени, то с вероятностью 99% это связанно с долгоиграющей транзакцией, которая «не отпускает» какой-либо из кластеров, и в результате база данных для обеспечения нормальной работы вынуждена создавать новые BI-кластеры. Такие ситуации очень опасны, и вам следует как можно быстрее выяснить, что это за транзакция, и принять соответствующие меры по прекращению её работы. В противном случае могут возникнуть следующие проблемы:

- база данных будет аварийно остановлена, если не включена поддержка больших файлов, и не установлены параметры <-bithold> (пороговый размер BI-файла) и <-bistall>[28];
- если установлены параметры <-bithold> и <-bistall>, то по достижению порогового размера, активность в базе данных будет остановлена и придется выполнить определенные действия для её восстановления;
- если параметры <-bithold> и <-bistall> не установлены, но при этом включена поддержка больших файлов, то это чревато тем, что BI-файл использует всё свободное место на диске, и база данных всё равно будет аварийно остановлена;
- в случае возникновения какого-либо сбоя во время работы долгоиграющей транзакции, восстановление базы данных (процесс Crash Recovery) может занять бесконечно много времени и привести к еще большему росту размера BI-файла.

На DVD-диске (приложенном к данному изданию) находится программа `check-bi.p`, которую можно использовать для мониторинга размера BI-файла, в случае достижения установленного порогового значения выслать уведомление на электронную почту с указанием списка активных транзакций, отсортированных по продолжительности их работы.

[28] Об использовании параметров запуска базы данных <-bithold> и <-bistall> будет рассказано позже.

Увеличение размера BI-блока

BI-файл состоит из кластеров. Кластеры, в свою очередь, состоят из блоков. Чтение и запись данных в BI-файл выполняется блоками. Таким образом, увеличение размера BI-блока позволит движку базы данных за один раз обрабатывать больше информации. Это может уменьшить нагрузку ввода/вывода на диск, на котором размещен BI-файл.

По умолчанию размер BI-блока равен 8 Кб, этого достаточно для приложений с низкой транзакционной активностью. Если мониторинг производительности показывает, что BI-запись является узким местом и вы уверены, что ваша система может обеспечить большую производительность записи, то увеличение размера BI-блока может улучшить ситуацию.

Для изменения размера BI-блока база данных должна быть остановлена, после чего необходимо выполнить команду:

```
proutil db-name -C truncate bi -biblocksize <новый размер в Кб>
```

Допустимым размером блока BI-файла считаются следующие значения в килобайтах: 1, 2, 4, 8 и 16. При этом можно указать значение 0, что будет означать необходимость установления размера блока по умолчанию, т.е. 8 Кб. **Примечание:** размер BI-блока не может быть меньше, чем размер блока базы данных.

Если у вас активирован механизм After-Imaging, то вы так же должны увеличить размер AI-блока. Для лучшей производительности размеры BI и AI-блоков должны совпадать.

Задержка BI-записи

Когда параметр запуска базы данных Delayed By File Write <-Mf> имеет нулевое значение, то в базе данных используется техника групповой задержки завершения транзакций. Эта техника предполагает небольшую задержку завершения каждой транзакции с целью улучшения общей производительности. Описать этот процесс можно следующим образом. Когда транзакция почти завершена, и в BI-файл необходимо записать завершающие заметки, возникает некоторая задержка в работе транзакции. Задержка длится до тех пор, пока BI-буфер заполнится и запишется на диск, либо несколько других транзакций завершатся и запишутся в BI-буфер синхронно за один раз. Используйте параметр запуска Group Delay (-groupdelay), чтобы установить время ожидания транзакции в миллисекундах. Если эта техника не поможет, то для задержки записи в BI-файл, для улучшения производительности, можно использовать параметр запуска <-Mf>.

По умолчанию движок базы данных записывает последний BI-блок на диск по завершению каждой транзакции. Это гарантирует, что завершенная транзакция будет записана в базу данных. В системах с низкой активностью изменений такое выполнение BI-записи очень важно и не создает никакого влияния на производительность. Но в загруженных системах такая работа с BI-файлом не имеет смысла, т.к. BI-блок будет записан на диск в любом случае очень скоро. Кроме того, это может значительно уменьшить производительность системы, поэтому установкой параметра <-Mf> мы задерживаем BI-запись. Когда он установлен, то последний BI-блок завершенной транзакции гарантировано будет записан на диск через определенное количество секунд. Задержка BI-записи не влияет на целостность базы данных. Тем не менее при возникновении системного сбоя несколько последних транзакций, скорее всего будут

потеряны. Если пользователь отключится от базы, или база будет корректно остановлена, то все BI-блоки будут сброшены на диск, до истечения времени, установленного параметром <-Mf>.

Что такое пороговый размер BI-файла?

Когда приложение выполняет большие объемы изменений в базе данных, или когда используется плохой стиль программирования, BI-файл может значительно вырасти в размере за короткий промежуток времени. Если в это время произойдет какой-либо сбой, то процесс восстановления может потребовать чуть ли не вдвое больше дискового пространства, чем было использовано BI-файлом на момент сбоя. Однако часто такой рост размера BI-файла может быть невозможен, например, закончится свободное пространство на диске, что приведет к повреждению базы данных. в целях недопущения чрезмерного роста BI-файла используйте параметр запуска базы данных Recovery Log Threshold <-bithold>, который устанавливает допустимый пороговый размер BI-файла в мегабайтах. Как только порог будет достигнут, база данных выполнит аварийную остановку. Этим вы можете гарантировать, что для восстановления базы данных будет иметься достаточное количество дискового пространства. При этом все сообщения, связанные с достижением порогового значения, будут отражены в логе базы данных (.lg).

Рекомендуемое пороговое значение должно быть между 3% и 100% максимально возможного размера BI-файла. Это значение должно быть кратно размеру BI-кластера. Система будет проверять общее количество используемых BI-кластеров каждый раз при создании нового кластера. Если вы используете параметр запуска No Crash Protection <-i>, то пороговое значение будет игнорироваться.

Если вы не желаете, чтобы при достижении порогового значения ваша база данных выполнила аварийную остановку, то можно воспользоваться параметром запуска базы данных Threshold Stall <-bistall>, который активирует на базе данных точку останова. Это предоставляет администратору базы данных возможность остановки базы данных, проверки доступного дискового пространства, а также возможность увеличения порогового значения. О том, что <-bistall>активировался, в лог базы данных записывается соответствующее сообщение.

Использование PROQUIET

С помощью команды PROQUIET можно отрегулировать пороговое значение BI-файла, т.е. можно увеличить его по отношению к текущему порогу. Для этого необходимовыполнить следующие действия.

1. Активировать точку останова на базе данных:

   ```
   proquiet db-name enable
   ```

   После запуска этой команды вся деятельность, связанная с записью в файлы базы данных прекращается. Все процессы, начавшие выполнение транзакций при активированной точке останова, будут ожидать снятие блокировки.

2. Увеличить пороговое значение, используя параметр bithreshold утилиты PROQUIET:

```
proquiet db-name bithreshold n
```

где *n* - новое пороговое значение.

3. Деактивировать точку останова:

```
proquiet db-name disable
```

Когда точка останова была активирована автоматически, т.е. когда сработал параметр <-bistall>, шаг 1 и 3 выполнять не нужно. Увеличение порогового значения автоматически деактивирует точку останова. Пример небольшого фрагмента лога базы данных, в которой размер BI-файла достиг порогового значения:

```
(9240) BI File size has grown to within 90 percent of the threshold value of 468.0
MBytes.
(9239) BI File Threshold size of 468.0 MBytes has been reached.
(6560) Forward processing stalled until database administrator intervention.
```

При появлении в логе базы данных сообщения **9239** все пользователи и все фоновые процессы, которые в это время были подключены к базе данных и вносили или пытаются внести в неё изменение, «зависнут». В этом случае администратор базы данных должен выполнить ряд действий.

1. Исходя из принципа доступности базы данных, в первую очередь увеличить пороговое значение BI-файла.

   Сказать точно, какое новое значение нужно установить довольно сложно. Здесь необходимо руководствоваться такими индивидуальными факторами, как возможности дискового пространства, обеспечение поддержки больших файлов в базе данных, скорость обнаружения проблемной транзакции.

2. Сразу после увеличения порогового значения приступить к поиску проблемной транзакции и прекращению её работы.

   После чего быть готовым к повторному увеличению порогового значения, либо к иным неприятным последствиям.

Для предотвращения подобной ситуации необходимо постоянно выполнять мониторинг долгоиграющих транзакций. Например, транзакция, которая выполняется более 15 минут, уже может считаться подозрительной. После обнаружения таких транзакций необходимо сразу связаться с пользователем, её создавшим. В зависимости от назначения транзакции решить что важнее - ожидание завершения этой транзакции (которое может и не произойти), или безопасность базы данных.

Глава 14 Механизм After-Imaging

Механизм After-Imaging позволяет настроить дополнительный уровень защиты базы данных. Этот механизм должен быть включен в любой базе, информация которой имеет хоть какую-либо ценность. Основная цель использования After-Imaging - защита системы от таких сбоев, как потеря носителя резервной копии, повреждение базы данных, повреждение BI-файла и т.п. Например, если потеряна последняя резервная копия, то можно воспользоваться предыдущей копией и восстановить её, последовательно накатив архивы AI-файлов, восстановив базу на требуемый момент времени.

Файлы AI - это архивы AI-экстентов, которые использует в своей работе механизм After-Imaging. Экстенты AI могут быть переменного и фиксированного размера. Если вы заинтересованы в улучшении производительности, то экстенты должны быть фиксированного размера, т.е. предварительно отформатированы. Это позволит получить:

- улучшение производительности за счет исключения форматирования пространства во время работы;
- последовательное непрерывное использование диска.

Единственным преимуществом экстентов переменного размера является то, что за ними не нужно осуществлять такой контроль, как за фиксированными экстентами, но такие экстенты не ограничены в размерах, а значит, в случае возникновения каких-либо проблем они могут заполнить всё возможное дисковое пространство.

Добавление AI-экстентов в базу данных

Экстенты описываются четырьмя признаками, которые приведены в таблице в порядке их использования в структурном файле.

Признак	Описание
Тип области хранения	Определяет тип файла. Тип «a» означает, что этот файл является AI-экстентом.
Путь и имя файла экстента	Определяет путь к файлу экстента. Путь может быть как относительным, так и абсолютным. Если расширение файла указано не будет, то оно подставляется автоматически по правилу «.a*n*», где *n* - порядковый номер AI-экстента.
Тип экстента	f - экстент фиксированного размера; v (или пусто) - экстент переменного размера.
Размер экстента	Размер экстента в килобайтах. Минимальное значение 16Кб. Размер экстента должен быть кратным 16Кб для всех операционных систем. Если будет определен другой размер, то утилита PROSTRCT выведет предупреждение и автоматически округлит значение до следующего значения, кратного 16Кб. Максимальный размер ограничен возможностями операционной системы и размерами диска.

Пример описания AI-экстента фиксированного размера:

```
a /users/valeriy/lecAI/db1/sports.a1 f 2048
```

Вместо полного пути к AI-экстенту можно использовать относительный путь.

Пример добавления трех AI-экстентов.

- Создайте новую тестовую базу данных Sports.
- Создайте в каталоге базы файл с именем `add-ai.st` со следующим содержимым:

```
a .
a .
a .
```

  Обратите внимание, что AI-экстенты создаем переменного размера, т.к. не указываем их размер, используем при этом относительные пути.

- Добавьте AI-экстенты к базе:

```
prostrct add sports add-ai.st
```

- Обновите структурный файл базы данных:

```
prostrct list sports
```

Активация After-Image

Для активации After-Imaging необходимо использовать параметр AIMAGE BEGIN утилиты RFUTIL. Перед этим необходимо сформировать полную резервную копию базы данных, т.к. AI-файлы и полная резервная копия используются совместно для восстановления базы.

Для активации в offline необходимо:

- сформировать полную резервную копию базы Sports;
- воспользоваться следующей командой для активации After-imaging:

```
rfutil sports -C aimage begin
```

Если база данных будет изменена после формирования резервной копии и до активации механизма After-Imaging, то утилита RFUTIL выведет сообщение об ошибке и не выполнит активацию.

Используйте утилиту PROBKUP с параметрами ONLINE и ENABLEAI для активации After-Imaging на запущенной базе данных. Время завершения копирования будет базовой точкой для наката AI-файлов. Для активации необходимо одновременно сформировать полную резервную копию базы данных и активировать After-Imaging, используя следующую команду:

```
probkup online sports ./backup/sports.bkp enableai
```

С этого момента после запуска базы данных любые изменения в ней будут фиксироваться в AI-экстентах, таким образом, AI-экстенты начнут заполняться, что в свою очередь потребует от администратора дополнительного внимания к ним.

Мониторинг AI-экстентов

Экстенты After-Image могут иметь различные статусы. Рассмотрим порядок смены этих статусов.

- EMPTY - AI-экстент пустой и доступен для использования;
- BUSY - AI-экстент, который заполняется в текущий момент;
- FULL - во время переключения AI-экстентов статус текущего экстента (BUSY) меняется на заполненный (FULL), а следующий пустой экстент (EMPTY) становится текущим (BUSY);
- LOCKED - статус используется, когда работает механизм OE Replication. Пока все AI-заметки не будут скопированы в target-базу из AI-экстента, он будет заблокирован (LOCKED). По завершению копирования AI-экстент будет разблокирован и получит статус FULL;
- ARCHIVED - во время работы OE Replication и AI File Management экстент AI, заблокированный и заархивированный механизмом AI File Management, получает статус ARCHIVED до завершения процесса репликации. После завершения репликации экстент будет помечен как пустой (EMPTY).

Существует четыре способа для определения статуса AI-экстента:

1. RFUTIL AIMAGE EXTENT LIST

   Эта команда предоставляет всю информацию о каждом из AI-экстентов, в том числе и их статусы:

   ```
   rfutil db-name -C aimage extent list
   ```

2. PROMON

   С помощью утилиты PROMON можно просмотреть информацию о статусе AI-экстентов на экране «Status: AI Extents» (R&D → 1 → 15):

   ```
   Area   Status Type   File          Size          Extent Name
                        Number        (KBytes)
   13     FULL   Fix    1             2041          sports.a1
   14     BUSY   Fix    2             2041          sports.a2
   15     EMPTY  Fix    0             2041          sports.a3
   ```

   Этот экран основан на данных, полученных из предыдущего примера, когда мы рассматривали мониторинг с помощью RFUTIL. Здесь мы можем получить информацию только о статусе и местоположении AI-экстентов. Поскольку в примере размер экстентов фиксирован, то отображается именно этот размер, а не реальные размеры экстентов, как это было видно в RFUTIL AIMAGE EXTENT LIST в поле Used. Обычно именно к этому методу прибегает администратор для быстрого определения состояния AI-экстентов.

3. RFUTIL AIMAGE QUERY

   Параметр AIMAGE QUERY утилиты RFUTIL позволяет определять статус конкретного AI-экстента. Возвращаемое значение для большинства запросов будет равно либо простому числу, либо слову. Это сделано для того, чтобы его было легко использовать и обрабатывать в скриптах. Для осуществления запроса используйте команду:

   ```
   rfutil db-name -C aimage query query-option by search-option
   search-value
   ```

В команде использован синтаксис:

- *query-option* - указывает на то, какую информацию необходимо получить об AI-экстенте. Возможные значения:

query-option	Возвращаемое значение
EXTNUM	Номер экстента.
STATUS	Статус экстента: EMPTY, FULL, LOCKED, BUSY/
TYPE	Тип экстента, т.е. фиксированный или переменный.
SIZE	Размер экстента в блоках по 1-му килобайту.
USED	Количество использованных блоков по 1-му килобайт.
NAME	Полное имя экстента (включая полный путь к нему).
SEQUENCE	Порядковый номер экстента.
STARTDATE	Дата и время начала записи в экстент.
ALL	Вся вышеперечисленная информация.

- *search-option*, определяет по каким признакам идентифицировать экстент в запросе. Используется в связке с параметром *search-value*, смотри таблицу:

Search-option	Search-value
EXTNUM	Целочисленное значение. В имени файла dbname.a*n* EXTNUM - это *n*.
NAME	Символьная строка, содержащая имя экстента. Можно использовать: • полное имя файла (/dsk3/users/valeriy/lecAI/db1/sports.a6); • короткое имя файла (sports.a3); • только расширение файла в формате *.a*n* (*.a1).
SEQUENCE	Целочисленное значение. Порядковый номер AI-экстента.

4. RFUTIL AIMAGE EXTENT FULL

   Параметр AIMAGE EXTENT FULL показывает имя самого старшего заполненного (FULL) AI-экстента. Используйте эту информацию для архивирования AI-экстентов в порядке их заполнения. Даже если будут существовать одновременно несколько заполненных (FULL) AI-экстентов, команда сообщит о наиболее старшем из них.

   Если выполнить команду на базе данных sports:

```
rfutil ./sports -C aimage extent full
```

   то информация о самом старшем FULL-экстенте будет выглядеть так:

```
/valeriy/lecAI/db1/sports.a1
```

Переключение экстентов

Переключение AI-экстентов происходит, когда текущий экстент фиксированного размера заполнен, или истекло время заполнения экстента, выставленное при использовании механизма AI File Management.

Переключение на новый AI-экстент происходит перед началом формирования online резервной копии. После восстановления базы из резервной копии накат AI-архивов выполняется с того AI-экстента, на который произошло переключение, т.е. с первого AI-архива, созданного после формирования этой копии.

Перед началом формирования online резервной копии необходимо сначала убедиться, что существует следующий по порядку AI-экстент со статусом EMPTY.

Фиксированные экстенты имеют строго определенный размер, поэтому движок базы данных самостоятельно определяет, когда экстент заполнится.

В отличие от экстентов фиксированного размера экстенты переменного размера ничем не ограничены. Следовательно, движок базы данных не может определить, когда экстент будет заполнен. Если не используется RFUTIL AIMAGE NEW, то экстент будет продолжать заполняться до тех пор, пока не достигнет ограничения по размеру файла, выставленного операционной системой, или не будет достигнут предел в 2Гб адресного пространства для файлов базы данных (если не активирована поддержка больших файлов). Экстент будет так же заполняться, пока не закончится место на диске. Когда текущий экстент заполнится, произойдет автоматическое переключение на следующий экстент, при условии, что он будет иметь статус EMPTY.

Если же следующий экстент заполнен (FULL), база данных будет остановлена. Однако чтобы этого не допустить, для приостановления активности базы можно использовать параметр запуска базы данных After-image Stall <-aistall>. При использовании <-aistall> можно заархивировать старший заполненный экстент и пометить его как пустой (EMPTY). Тогда система автоматически переключит экстенты и восстановит активность базы данных. Можно также с помощью команды RFUTIL AIMAGE AIOFF выключить After-Imaging.

Примечание: использование параметра <-aistall> и RFUTIL AIMAGE AIOFF воможно только в многопользовательском режиме.

База данных не сможет восстановить активность, пока необходимый для переключения экстент не будет заархивирован и помечен как пустой.

Можно вручную в online выполнять переключение AI-экстентов, не дожидаясь их полного заполнения. Для этого необходимо выполнить команду:

```
rfutil db-name -C aimage new
```

Запуск RFUTIL AIMAGE NEW изменяет статус текущего экстента на FULL, а следующего на BUSY.

Архивирование AI-экстентов

Рассмотрим пример.

- Определите самый старший заполненный экстент с помощью RFUTIL AIMAGE EXTENT FULL:

  ```
  rfutil ./sports -C aimage extent full
  ```

  Допустим, что это будет экстент с номером `1 - sports.a1`

- Выгрузите AI-данные из этого экстента в архивный файл с помощью RFUTIL AIMAGE EXTRACT:

```
rfutil ./sports -C aimage extract -a ./sports.a1 -o
./10072010.1846.sports.a1.valeriy.bak
```

Здесь, в целях демонстрации, имя архивного файла указано явно, но в скриптах можно генерировать имена автоматически, расширяя их информационное содержание. В итоге, мы получили файл `10072009.1846.sports.a1.valeriy.bak`, который теперь можно запаковать (gzip) и отправить на хранение.

- Выполните архивацию всех оставшихся AI-экстентов со статусом FULL (если таковые имеются).

- Теперь можно установить заархивированным AI-экстентам статус EMPTY:

```
rfutil ./sports -C aimage empty ./sports.a1
```

- Для того чтобы наглядно убедиться, что экстенты действительно получили статусы EMPTY, воспользуйтесь RFUTIL AIMAGE QUERY:

```
rfutil ./sports -C aimage query status by extnum 1
```

AI File Management

Механизм AI File Management создан для выполнения автоматического архивирования экстентов. Он реализован в виде демона, запущенного брокером базы данных. Механизм может работать во «Временн**о**м» режиме или в режиме «По требованию». Во временном режиме при использовании специального параметра переключение экстентов выполняется через интервал времени, даже если экстент еще не заполнен. Минимальный интервал – 120 секунд, максимальный – 86400 секунд (24 часа). В режиме по требованию переключение выполняется только тогда, когда экстент заполнен (FULL). В любом случае проверка экстента на заполнение происходит каждые пять секунд.

Активация в online

Проверьте, имеются ли в базе данных AI-экстенты, активирован ли механизм After-Imaging. Если нет, то добавьте AI-экстенты и активируйте After-Imaging (можно выполнить одновременно с активацией AI File Management через добавление параметра <enableai> к команде probkup).

Создайте каталог, в который будет производиться архивация AI-экстентов.

Выполните online резервное копирование базы с активацией AI File Management (копия базы будет считаться отправной точкой для наката архивов AI-экстентов):

```
probkup online db-name output-device enableaiarchiver
     -aiarcdir path-dir1,path-dir2
     -aiarcinterval interval -aiarcdircreate
```

В этом примере определены два каталога для хранения архивов AI-экстентов, это каталоги *path-dir1* и *path-dir2*. Если эти каталоги будут отсутствовать, то при наличии параметра <-aiarcdircreate> они создаются автоматически. Механизм AI File Management

переключится на следующий каталог, если в текущем каталоге закончится доступное место (диск заполнен). Использовать два каталога имеет смысл только тогда, когда они размещены на разных носителях. При старте базы данных должен быть использован параметр <-aistall>.

Параметр <-aiarcinterval *interval*> используется, чтобы «будить» демона для архивации AI-экстентов по истечению интервала времени, установленного значением *interval*. Интервал может быть установлен от 2 минут до 24 часов.

Активация в offline

Проверьте, имеются ли в базе данных AI-экстенты, и активирован ли механизм After-Imaging. Если нет, то добавьте AI-экстенты и активируйте After-Imaging.

Создайте каталог, в который будет выполняться архивация AI-экстентов.

Активируйте AI File Management;

```
rfutil db-name -C aiarchiver enable
```

Запустите брокер базы данных со следующими параметрами механизма AI File Management, используя следующий синтаксис:

```
proserve db-name -S port-num
     -aiarcdir path-dir1,path-dir2
     -aiarcinterval interval
```

Восстановление базы данных из AI-архивов

Для выполнения примера по восстановлению базы данных с помощью AI-архивов выполните следующие действия.

- Остановите базу данных:

```
proshut ./sports -by
```

- Деактивируйте After-Imaging командой PROUTIL AIMAGE END:

```
proutil ./sports -C aimage end
```

- Очистите каталоги с AI-архивами (если таковые есть).

- Сформируйте полную резервную копию базы данных:

```
probkup ./sports ./backup/sports-full.bak
```

- Активируйте After-Imaging:

```
proutil ./sports -C aimage begin
```

- Активируйте AI File Management:

```
proutil ./sports -C aiarchiver enable
```

- Запустите базу данных с параметрами <-aiarcdir> и <-aiarcinterval>:

```
proserve ./sports -aiarcdir ./aiarch1 -aiarcinterval 120
```

Теперь приступим непосредственно к выполнению практического примера.

- Восстановите полную резервную копию в отдельный, заранее созданный каталог, названный, например ./db2:

```
prorest ./sports ../db1/backup/sports-full.bak
```

- Подключитесь к оригинальной базе данных и выполните некоторые изменения данных.

  Войдите в редактор Procedure Editor:

```
mpro ./sports
```

  Выполните ABL-код:

```
find last customer exclusive-lock.
      customer.name = "Тестирование ROLL-FORWARD".
```

- Подождите две минуты, чтобы сформировался архив AI-экстента, и выполните еще одно изменение с помощью следующего ABL-кода:

```
for each order where order.custnum = 2106 exclusive-lock.
    for each orderline of order exclusive-lock.
        delete orderline.
    end.
    delete order.
END.
```

- Подождите еще две минуты, чтобы сформировался очередной архивный файл AI-экстента.

Следующий этап - это накат полученных AI-архивов на восстановленную полную резервную копию.

- Скопируйте имеющиеся в наличии AI-архивы из каталога `./db1/aiarch1` в созданный новый каталог `./db2/ai/`.

- Используя команду RFUTIL ROLL FORWARD, накатите все AI-архивы на восстановленную базу данных с помощью простого скрипта:

```
cd ../db2/ai

for ai in `ls -1`; do rfutil ../sports -C roll forward -a $ai;
done
```

Если вы всё сделали правильно, то после наката каждого AI-файла на экран будет выводиться приблизительно следующее сообщение:

```
After-image dates for this after-image file: (1633)
    Last AIMAGE BEGIN Tue Jul 14 13:18:22 2009 (1640)
    Last AIMAGE NEW Tue Jul 14 13:43:25 2009 (1641)
    This is aimage file number 12 since the last AIMAGE BEGIN. (1642)
    This file was last opened for output on Tue Jul 14 13:43:25 2009. (1643)


21 notes were processed. (1634)
0 in-flight transactions. (3785)
1 transactions were started. (1635)
1 transactions were completed. (11138)
At the end of the .ai file, 0 transactions were still active. (1636)
```

Эта же информация будет записана в лог базы данных.

Деактивация After-Imaging

Для деактивации механизма After-Imaging используется утилита RFUTIL. Последовательность действия по деактивации выглядит следующим образом:

- текущий используемый AI-экстент (статус BUSY) помечается как FULL;
- копируются все заполненные экстенты (статус FULL),это необходимо для того, чтобы не потерять данные в AI-экстентах, т.к. RFUTIL после деактивации помечает все экстенты как пустые, данные в них будут потеряны;
- деактивируется After-Imaging командой AIMAGE END утилиты RFUTIL:

```
rfutil db-name -C aimage end
```

Когда необходимо остановить работу механизма After-Imaging в online, можно использовать классификатор AIMAGE AIOFF утилиты RFUTIL.

Настройка After-Imaging

В очень активных базах данных экстенты After-Image желательно размещать на отдельных дисках. Это связано не только с обеспечением безопасности базы, но и с необходимостью разгрузки дисков, на которых она размещена. Этим вы снижаете нагрузку с дисков базы данных за счет перемещения части нагрузки по записи, генерируемой механизмом After-Imaging, на отдельный диск. Перераспределение дискового ввода/вывода - это только небольшая часть в настройке производительности этого механизма. Размещение AI-экстентов на отдельный диск - это только начало основного процесса, поскольку сама по себе высокая активность механизма After-Imaging, даже если его экстенты находятся на отдельных дисках, может существенно влиять на производительность базы данных в целом. Снизить это влияние можно несколькими способами:

- при установленной лицензии OE Enterprise RDBMS используйте фоновый процесс AIW[29];
- измените количество AI-буферов;
- измените размер AI-блока.

Мониторинг активности After-Image

Если вы видите, что диск, на котором расположены AI-экстенты, очень загружен, то имеет смысл обратить внимание на активность After-Imaging. Посмотреть настройки этого механизма в базе данных можно с помощью экрана Status AI Log (R&D → 1 → 10) утилиты Promon:

```
After-image begin date:            02/01/10 11:35
After-image new date:              02/01/10 11:35
After-image open date:             02/01/10 11:35
After-image generation number:     1
Number of after-image extents:     3
Current after-image extent:        14
Number of AI buffers:              20
After-image block size:            8192 bytes
After-image log size:              17944      K
```

Здесь мы видим, когда именно был включен механизм в базе данных; какое количество AI-экстентов имеется; номер AI-экстента, заполняемого в текущий момент; количество выделенных AI-буферов; размер AI-блока и, наконец, размер текущего AI-экстента.

Следующий экран Promon, позволяющий более детально выяснить, как работает механизм, это Activity AI Log (R&D → 2 → 6):

	Total	Per Min	Per Sec	Per Tx
Total AI writes	12999	833	**13.89**	0.00
AIW AI writes	0	0	0.00	0.00
Records written	710907	45571	759.52	0.00
Bytes written	103431K	6789328	113155.46	0.00
Busy buffer waits	**12**	0	0.00	0.00
Buffer not avail	**12434**	797	13.28	0.00
Partial writes	561	36	0.60	0.00
Log force waits	0	0	0.00	0.00

Если в поле Busy buffer waits в первой колонке наблюдаются большие значения, то вероятно, что у вас не запущен AIW-процесс, либо базе выделено недостаточно AI-буферов, поэтому запустите AIW-процесс, если он еще не запущен и увеличьте количество AI-буферов. Последнее верно и для поля Buffer not avail, поскольку если AI-буферов будет недостаточно, то процесс, изменяющий базу данных, вынужден ожидать сброс заполненных буферов на диск. Обратите так же внимание на поле Total AI writes и колонку Per Sec, старайтесь держать значение как можно более низким, для его уменьшения обычно можно увеличить размер AI-блока.

Процесс AIW и AI-буферы

Процесс AIW - фоновый процесс, выполняющий запись AI-буферов на диск после их заполнения. Если процесс работает эффективно, то процессам, изменяющим базу данных, будут всегда доступны свободные AI-буферы. Буферы AI организованы в буферный пул, который представляет собой циклическую цепочку. Они заполняются поочередно. Буфер,

[29] AIW, от англ. After-Image Writer

заполняющийся в текущий момент, называется выходным буфером. Любой буфер может стать выходным, поскольку они заполняются по кругу, поэтому если следующий буфер еще заполнен, то процесс должен ожидать, пока этот буфер будет записан на диск, помечен как пустой, и только после этого работа процесса может быть продолжена. Запуск AIW-процесса значительно ускорит запись AI-буферов на диск, тем самым исключая необходимость ожидания пустых буферов. Если система сильно активна, то одного AIW-запуска может быть и недостаточно. И если вы по-прежнему наблюдаете высокое количество ожиданий пустых буферов, то необходимо их увеличить. Для этого необходимо воспользоваться параметром запуска базы данных <-aibufs>. По умолчанию количество AI-буферов равно 20, добавьте этот параметр в строку запуска базы данных, указав ему для начала значение, равное 25. После чего заново выполните мониторинг. При необходимости увеличьте количество AI-буферов. Начиная с версии OpenEdge 10.1C появилась возможность увеличения количества AI-буферов, не прибегая к остановке базы данных. Делается это с помощью команды PROUTIL INCREASETO:

```
proutil sports -C increaseto -aibufs 25
```

Следует так же помнить, что для одной базы данных может быть запущен только один AIW-процесс, и его запуск и остановка может осуществляться в любое время. Если AIW-процесс не запущен, то увеличение количества AI-буферов не даст никакого эффекта. Желательно, чтобы количество AI-буферов совпадало с количеством BI-буферов.

Изменение размера AI-блока

Как и в случае с механизмом Before-Imaging информация записывается в AI-экстенты по блокам. Увеличение размера AI-блока позволяет записывать больше данных за один раз, что может уменьшить нагрузку на диски, на которых размещены AI-экстенты. Для систем с низкой транзакционной активностью значение размера блока по умолчанию равно 8Кб, обычно этого достаточно. Тем не менее, если мониторинг производительности показывает, что AI-запись является узким местом, и вы уверены, что ваша система может обеспечить б**о**льшую производительность, то увеличение размера AI-блока может вам помочь. Большой размер блока так же мог бы улучшить эффективность при накате архивов AI-экстентов на горячую резервную копию, или просто когда необходимо восстановить базу данных на определенный момент времени.

Для изменения размера AI-блока выполните следующее.

- Остановите базу данных, т.к. изменить размер AI-блока можно только на остановленной базе.
- Отключите механизм After-Imaging.
- Выполните усечение BI-файла.
- Измените размер AI-блока командой:

```
rfutil db-name -C aimage truncate -aiblocksize size
```

Здесь size определяет размер AI-блока в килобайтах. Минимально возможное значение равно размеру блока базы данных. Корректными значениями являются – 1, 2, 4, 8 и 16. Если будет определен 0, то RFUTIL использует значение по умолчанию (8K).

- Теперь сформируйте полную резервную копию базы данных и возобновите работу механизма After-Imaging.
- Запустите базу данных.

Более детальное описание работы механизма After-Imaging приведено в книге «After-Imaging в OpenEdge», находящейся на сайте www.openedge.ru.

Глава 15 Сервер приложений

Сервер приложений (Application Server[30]) – программный продукт компании Progress Software, предназначенный для организации приложений с многоуровневой архитектурой, позволяющей разделить данные, бизнес-логику и интерфейс пользователя.

Сервер приложений использует для обработки клиентских запросов APP-брокеры, с которыми пользователи взаимодействуют в режиме реального времени. APP-брокер – фоновый ABL-процесс, не имеющий пользовательского интерфейса. Каждый APP-брокер управляет работой нескольких APP-агентов, обрабатывающих пользовательские запросы. Все настройки APP-брокеров, включая минимальное и максимальное количество APP-агентов, устанавливаются в специальном файле настроек `ubroker.properties` из каталога $DLC/properties.

APP-брокеры могут работать в следующих режимах:

- State-Reset
- State-Aware
- Stateless
- State-Free

Режим State-Reset. Клиентский процесс напрямую подключается к агенту APP-брокера, при этом APP-агент блокируется инициировавшим его клиентом и доступен только ему в течение всего соединения. Когда клиент отключается, контекст APP-агента сбрасывается. Этот режим используется по умолчанию.

Режим State-Aware. Отличие от State-Reset заключается в том, что между клиентскими вызовами контекст APP-агента сохраняется, т.е. контекст одного вызова (временные таблицы и т.п.) доступен для следующих вызовов.

Режим Stateless. Клиентский процесс устанавливает связь с APP-брокером, а брокер направляет очередной клиентский запрос на первый из доступных APP-агентов.

Режим State-Free. Клиентский процесс не устанавливает жесткую связь с одним APP-брокером. Все клиентские запросы исполняются параллельно свободными APP-брокерами. Состояние APP-агентов между вызовами не сохраняется.

Для управления сервером приложений используется AdminServer. AdminServer – инструмент, который устанавливается одновременно с такими продуктами как OpenEdge RDBMS, OpenEdge DataServer, OpenEdge Adapter for SonicMQ Broker Connect, AppServer, AppServer Internet Adapter, Web Services Adapter, NameServer или WebSpeed Transaction Server. Фактически AdminServer - это средство связи между перечисленными продуктами. Перед использованием любого из указанных продуктов должен быть запущен AdminServer.

[30] Application Server – далее будет применяться термин APP.

Список команд для работы с APP-сервером:

Команда	Назначение
для AdminServer-a	
$DLC/bin/proadsv -start	Запуск AdminServer-a.
$DLC/bin/proadsv -q	Проверка состояния AdminServer-a.
$DLC/bin/proadsv -stop	Останов AdminServer-a.
для APP-сервера	
$DLC/bin/asbman -i <Имя APP-брокера> -start	Запуск APP-брокера.
$DLC/bin/asbman -i <Имя APP-брокера> -stop	Останов APP-брокера.
$DLC/bin/asbman -i <Имя APP-брокера> -q	Проверка состояния APP-брокера.

Пример настройки APP-брокера

Для обеспечения уникальности APP-брокера в системе ему необходим уникальный идентификатор, для получения которого выполните:

```
$DLC/bin/genuuid
```

Результат работы:

```
OpenEdge Release 10.2B01 as of Tue Apr 27 19:16:33 EDT 2010
be6e634372ddc598:-6b13bdce:12aa96554fc:-8000
```

Здесь выделенное значение - уникальный идентификатор.

Откройте для редактирования файл `$DLC/properties/ubroker.properties`. Этот файл структурирован по секциям. Секции имеют детализацию от общих свойств к частным. Например, секция [UBroker] описывает общие параметры для всех брокеров[31], секция [UBroker.AS] - общие параметры для всех APP-брокеров, секция [UBroker.AS.*asbroker1*] - параметры для конкретного брокера с именем *asbroker1*. Если строка в файле начинается с символа <#>, то она является комментарием.

Найдите секцию [Environment] (общее окружение среды выполнения), после нее добавьте секцию для нового брокера, примерно в таком виде:

```
# Environment variables for TestAPPBroker
[Environment. TestAPPBroker]
      APPLOGDIR= <каталог для сохранения логов>
      WRKDIR= <рабочий каталог APP-брокера>
```

Далее найдите секцию [UBroker.AS] (общие свойства брокеров APP-серверов), после неё добавьте секцию для нового брокера, примерно в таком виде:

```
[UBroker.AS.TestAPPBroker]
uuid= be6e634372ddc598:-6b13bdce:12aa96554fc:-8000
srvrStartupParam=-T /tmp -inp 20000 -tok 32000
srvrLogFile=<каталог для лога сервера>/TestAPPBroker.server.log
operatingMode=State-reset
controllingNameServer=NS1
brokerLogFile=<каталог для лога брокера>/ TestAPPBroker.broker.log
environment=TestAPPBroker
autoTrimTimeout=120
appserviceNameList=TestAPPBroker
portNumber=4070
initialSrvrInstance=3
```

[31] Брокеры могут быть не только у APP-сервера.

```
PROPATH=<каталог с ABL-программами>
workDir=<рабочий каталог APP-брокера>
brkrLoggingLevel=3
maxClientInstance=15
description=Appserver for TestAPPBroker
```

Пояснения.

- uuid - уникальный идентификатор. Вставьте значение, полученное командой genuuid.
- srvrStartupParam - стартовые параметры AppServer-а, в данном случае они указаны явно, но это может быть и pf-файл, который должен находиться в рабочем каталоге APP-брокера.
- srvrLogFile - файл лога AppServer-а. Введите собственный каталог.
- operatingMode - режим работы (в данном случае State-reset).
- controllingNameServer - имя Name Server-а, который управляет данным брокером. По умолчанию при старте AdminServer-а запускается Name Server с именем NS1.
- brokerLogFile - файл лога брокера. Введите собственный каталог.
- environment - ссылка на секцию, в которой задаются переменные окружения для данного брокера.
- autoTrimTimeout - время автоматического отключения агента (в секундах). Это свойство задает время, через которое брокер остановит незанятого обработкой агента. По умолчанию предлагается 1800 секунд, при таком значении параметра брокер будет ждать 30 минут прежде, чем остановить агента.
- appserviceNameList - совпадает с выбранным именем APP-брокера.
- portNumber - номер свободного TCP-порта, по которому брокер будет обслуживать запросы. Введите собственный порт.
- initialSrvrInstance - число агентов, которые будут запущены при старте APP-брокера.
- PROPATH - путь к исполняемым файлам (r-кодам). По этому пути должны быть доступны все процедуры (достаточно, чтобы были доступны r-коды), которые будут запускаться на APP-сервере. Введите собственный каталог.
- workDir - рабочий каталог. В данном случае рабочий каталог совпадает с переменной окружения WRKDIR, которая определяется в соответствующей секции environment.
- description - описание APP-брокера.

Теперь брокер TestAPPBroker можно считать настроенным.

Старт APP-брокера

Перед стартом APP-брокера необходимо проверить, а в случае необходимости запустить AdminServer.

- Проверьте, запущен ли AdminServer:

  ```
  proadsv -q
  ```

  Если он запущен, то будет выведено сообщение «AdminServer is alive. (8545)». Если он не запущен, то сообщение «AdminServer not alive. (8543)», в этом случае запустите его командой:

```
proadsv -start
```

- Запустите APP-брокер TestAPPBroker:

```
asbman -i TestAPPBroker -start
```

- Проверьте, что APP-брокер запущен:

```
asbman -i TestAPPBroker –q
```

Если APP-брокер запущен успешно, то на экране появится следующая информация:

```
Connecting to Progress AdminServer using rmi://localhost:20931/Chimera (8280)
Searching for TestAPPBroker (8288)
Connecting to TestAPPBroker  (8276)

Broker Name                    : TestAPPBroker
Operating Mode                 : State-reset
Broker Status                  :  ACTIVE
Broker Port                    : 4070
Broker PID                     : 12828
Active Servers                 : 3
Busy Servers                   : 0
Locked Servers                 : 0
Available Servers              : 3
Active Clients (now, peak)     : (0, 0)
Client Queue Depth (cur, max)  : (0, 0)
Total Requests                 : 0
Rq Wait (max, avg)             : (0 ms, 0 ms)
Rq Duration (max, avg)         : (0 ms, 0 ms)

PID   State     Port  nRq    nRcvd  nSent  Started          Last Change
12861 AVAILABLE 02002 000000 000000 000000 Aug 30,2010 12:09 Aug 30,2010 12:09
12863 AVAILABLE 02004 000000 000000 000000 Aug 30,2010 12:09 Aug 30,2010 12:09
12865 AVAILABLE 02005 000000 000000 000000 Aug 30,2010 12:09 Aug 30,2010 12:09
```

В противном случае будет выдано сообщение:

```
Broker: TestAPPBroker not running (8313)
```

Если APP-брокер не был запущен, то проверьте правильность настроек брокера TestAPPBroker в файле `ubroker.properties`.

Подключение к APP-брокеру из ABL-программ

Для подключения к APP-брокеру из ABL-программы используется оператор CONNECT, которому передаются параметры подключения к брокеру:

```
Handle:CONNECT(  "-AppService  <имя  APP-брокера>  -H  <хост>  -S  <порт
AdminServer`a>" ).
```

В наших примерах под портом AdminServer-а подразумевается порт 5162. Это стандартный порт AdminServer-а, на котором он стартует по умолчанию.

Создайте каталог с именем App в своем домашнем директории, в нем мы будем хранить программы-примеры. После этого пропишите полный путь к новому каталогу в параметре

PROPATH брокера TestAPPBroker (см. описание файла `ubroker.properties`) и перезапустите APP-брокер, чтобы изменения вступили в силу.

Пример 1

- Перейдите в каталог App.
- Создайте в каталоге App файл `test01.p` со следующим содержимым:

```
OUTPUT TO "<путь к вашей домашней директории>/App/test.txt".
       PUT UNFORMATTED "HELLO FROM TestAPPBroker " SKIP.
OUTPUT CLOSE.
```

- Затем создайте файл `app1.p`:

```
DEF VAR H1 AS HANDLE.
DEF VAR Ok AS LOGICAL.
CREATE SERVER H1.
OK = H1:CONNECT( "-AppService TestAPPBroker -H polygon -S 5162" ) NO-ERROR.
IF OK THEN
    DO:
        RUN test01.p ON H1.
        H1:DISCONNECT().
        MESSAGE "App Server OK" VIEW-AS ALERT-BOX.
    END.
ELSE
    MESSAGE "App Server NOT connect" VIEW-AS ALERT-BOX.
DELETE OBJECT H1.
QUIT.
```

- Выполните команду для запуска этих программ:

```
mpro -p ./app1.p
```

  Если всё было сделано правильно, то на экране появится сообщение «App Server OK», а в каталоге App будет создан файл `test.txt` со строкой «My first APP-broker».

Пример 2

- Остановите брокера TestAPPBroker.
- Создайте в каталоге, который был указан в качестве значения переменой workDir, файл `sports.pf` со следующими параметрами подключения к базе данных sports:

```
-db <каталог с базой sports>/sports
-U admin
-P 123
```

- В настройках брокера измените параметр srvrStartupParam:

```
srvrStartupParam=-pf sports.pf
```

  Сохраните изменения, и запустите APP-брокер.

- Создайте файл `test02.p` со следующим содержимым:

```
DEFINE OUTPUT PARAMETER CntOrder AS INTEGER.
SELECT count(*) INTO CntOrder FROM Order.
```

- Создайте файл `app2.p`:

```
DEFINE VARIABLE H1 AS HANDLE.
DEFINE VARIABLE Ok AS LOGICAL.
DEFINE VARIABLE CntOrder AS INTEGER.
CREATE SERVER H1.
OK = H1:CONNECT( "-AppService TestAPPBroker -H polygon -S 5162" ) NO-ERROR.
IF OK THEN
    DO:
        RUN test02.p ON H1(OUTPUT CntOrder).
        H1:DISCONNECT().
        MESSAGE "Общее количество счетов:" + STRING(CntOrder) VIEW-AS ALERT-
BOX.
    END.
ELSE
   MESSAGE "App Server NOT connect" VIEW-AS ALERT-BOX.
DELETE OBJECT H1.
QUIT.
```

- Выполните команду:

```
mpro -p ./app2.p
```

  В случае успешного подключения к APP-брокеру на экран будет выведено сообщение: «Общее количество счетов:207»

Более детальную информацию по настройке и использованию APP-серверов смотрите в стандартной документации по OpenEdge: «Application Server Administration», «Developing AppServer™ Applications» и «Ubroker Properties Readme».

Глава 16 Факторы, влияющие на производительность

Диски

Дисковые системы - самый важный ресурс базы данных, поскольку это наиболее используемая часть компьютера, то диски наиболее склонны к различным сбоям. Диски так же являются самым медленным ресурсом, и в то же время самым главным, поскольку на них хранятся данные.

Существуют три характеристики, связанные с дисковыми ресурсами.

- Количество - всегда необходимо иметь достаточное количество дискового пространства для хранения необходимых данных.
- Надежность - диски должны быть надежными, чтобы обеспечивать постоянную доступность данных.
- Производительность - количество дисков и их максимально возможная скорость должны удовлетворять потребностям пользователей, обеспечивая хорошую производительность.

Далее приведен список критичных данных, хранящихся в системе. Использование в этом контексте термина «данные» - более емкое понятие, чем то, которое определяет обычную прикладную информацию:

- файлы пользовательских областей хранения данных;
- файлы Before-Image;
- файлы After-Image;
- файлы приложения (ABL или SQL-коды, прочие программы);
- временные файлы;
- клиентские файлы.

Эти данные так же могут быть связаны с другими возможными данными:

- резервные копии баз данных;
- входные и выходные файлы;
- копии баз данных для разработки;
- тестовые копии базы данных.

Прочие критичные элементы, хранимые на дисках:

- операционная система;
- SWAP-файлы и файлы подкачки (paging);
- инсталляция OpenEdge.

Увеличение размеров базы всегда обусловлено ростом компании, что определяется как увеличением количества обслуживаемых клиентов, так и возможным слиянием нескольких компаний в одну. Администратор базы данных всегда должен быть извещен о подобных ситуациях, чтобы иметь возможность своевременного планирования возможного увеличения размеров базы данных. В некоторых случаях, помимо обеспечения возможностей роста, можно рассматривать такие ситуации, когда некоторые данные в базе могут быть выгружены из неё и отправлены в архив. Для хранения такой

информации можно использовать отдельную машину с отдельным дисковым пространством. Такая машина может быть использована в качестве архивного сервера, репликационного сервера или сервера разработки. Тем самым, перемещая менее критичные данные с промышленного сервера на другой сервер, можно сэкономить на дисках, т.е. на архивной машине можно использовать менее дорогие диски, а на промышленной более быстрые и дорогостоящие.

Важно четко понимать, как архивные данные будут использоваться, поэтому всегда перед выполнением их перемещения необходимо создавать конкретный план действий. Например, некоторым пользователям может понадобиться выполнить чистку данных, в этом случае вы должны будете создать архивную копию этих данных для того, чтобы при необходимости можно было легко их восстановить. Методы хранения архивов могут быть достаточно просты. Например, можно использовать магнитные ленты. Однако необходимо помнить, что к этим данным может понадобиться доступ, поэтому следует учитывать формат хранения. Может возникнуть ситуация, когда использовать восстановленные данные невозможно из-за того, что формат данных уже не поддерживается. Всегда архивируете данные OpenEdge базы данных стандартными средствами OpenEdge, а приложения архивируйте так, чтобы формат не зависел от программного обеспечения сторонних производителей.

Приобретая диск, вы приобретаете две возможности: расширение дисковой памяти и возможность улучшения производительности. Например, если вам необходимо дополнительно 72Гб памяти, вы можете приобрести один диск размером 72Гб, или четыре диска по 18Гб. Возможности памяти этих дисков будут одинаковые, но конфигурация из четырех дисков имеет в четыре раза большую производительность. Один диск может выполнять примерно 100 операций ввода/вывода в секунду, независимо от своего размера. Поэтому четыре диска потенциально способны одновременно выполнять 400 операций ввода/вывода в секунду. Большее число физических дисков даст больший потенциал производительности. Дополнительная стоимость многодисковых решений компенсируется улучшением производительности, а это - эффективная работа пользователей, программистов; лояльность клиентов и т.п. Однако если база данных будет храниться в качестве архивной копии, где высокая производительность не нужна, то меньшее количество больш**и**х дисков будет способствовать уменьшению стоимости решения. Такой подход позволит хранить наибольшее количество данных на наименьшем количестве дисков.

Для улучшения производительности некоторые диски используют кэширование. Тем не менее, существует предел, по достижению которого эффективность кэширования снижается. Если в вашей системе выполняются сотни тысяч операций чтения в час, то в скором времени кэш будет перенасыщен. В этих условиях система будет быстро деградировать, возвращаясь к обычным дисковым скоростям. Поэтому важно рассматривать скорость работы дисков с точки зрения их обычных возможностей, независимо от рекомендаций изготовителя. В противном случае вы можете столкнуться с проблемами, когда кэш окажется переполнен. Очень быстро переполняет дисковый кэш, например, процесс перестройки индексов, поскольку в короткий промежуток времени он выполняет очень большое количество изменений. Кэш, несомненно, полезен с точки зрения производительности, однако, не позволяйте вводить себя в заблуждение по поводу использования дисков с кэшем или без него.

Важно иметь правильное представление о надежности и производительности дисков. Надежность - это способность дисковой системы после возникновения одного или множества сбоев продолжать быть пригодной для использования. Производительность -

способность дисков предоставлять информацию пользователям самым эффективным способом.

Добавление избыточности почти всегда увеличивает надежность дисковой системы. Основной способ достижения избыточности, это использование RAID[32].

Существует два типа RAID.

- Аппаратные (hardware) – наиболее часто используемые из них имеют уровни: RAID 0, RAID 1, RAID 5 и RAID 10. Различия между ними заключаются в обеспечении уровня надежности и производительности.
- Программные (software) – такие RAID менее дороги, тем не менее, они значительно медленнее, чем аппаратные RAID, поскольку для управления вводом/выводом они дополнительно нагружают CPU.

Свойства аппаратных RAID.

- RAID 0 - распределение данных (data striping). Информация разбивается на куски (фиксированные объемы данных, обычно именуемые блоками), и эти куски записываются на диски и считываются с них параллельно. С точки зрения производительности есть два основных преимущества: повышается пропускная способность последовательного ввода/вывода за счет одновременной загрузки нескольких интерфейсов и снижается латентность случайного доступа, несколько запросов к различным небольшим сегментам информации могут выполняться одновременно. Недостаток: уровень RAID 0 предназначен исключительно для повышения производительности, и не обеспечивает избыточности данных. Поэтому любые дисковые сбои потребуют восстановления информации с резервных носителей.

- RAID 1 - зеркалирование (disk mirroring). В этом случае копии каждой части информации хранятся на отдельном диске; или каждый используемый диск имеет «двойника», который хранит точную копию этого диска. Если происходит сбой одного из основных дисков, то он замещается своим «двойником». Производительность произвольного чтения может быть улучшена, если для чтения информации будет использоваться тот из «двойников», считывающая головка которого расположена ближе к требуемому блоку. Время записи может оказаться несколько б**о**льшим, чем для одного диска, в зависимости от стратегии записи - запись на два диска либо параллельно (для скорости), либо строго последовательно (для надежности). RAID 1 хорошо подходит для приложений, которые требуют высокой надежности, низкой латентности при чтении, а также, если не требуется минимизация стоимости. RAID 1 обеспечивает избыточность хранения информации, но в любом случае следует иметь резервную копию данных, т.к. ее наличие - это единственный способ восстановить случайно удаленные файлы или директории.
- RAID 10 - массив типа RAID 0, сегментами которого являются массивы RAID 1. Он объединяет в себе очень высокую отказоустойчивость и производительность. Преимущества: высокая отказоустойчивость и высокая производительность. Недостатки: очень высокая стоимость и ограниченное масштабирование. Progress Software рекомендует использовать именно этот уровень RAID.

[32] RAID – от англ. Redundant Array of Inexpensive Disks

- RAID 5 – страйпинг (striping) с распределенной контрольной суммой. Преимущества: высокая скорость записи данных; достаточно высокая скорость чтения данных; высокая производительность при большой интенсивности запросов чтения/записи данных;·малые накладные расходы для реализации избыточности. Недостатки: низкая скорость чтения/записи данных малого объема при единичных запросах; достаточно сложная реализация; сложное восстановление данных. Progress Software крайне не рекомендует использовать этот уровень RAID.

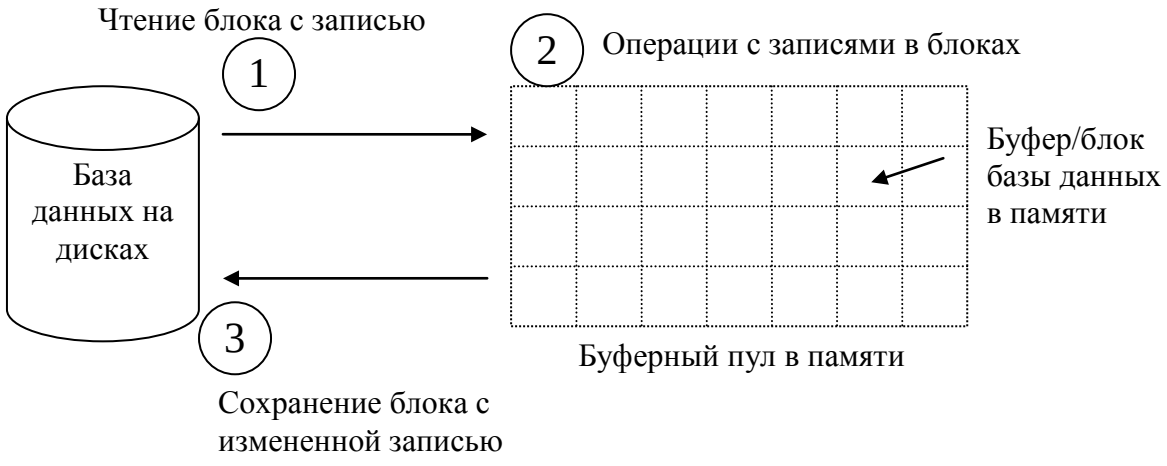
Диски - это самые важные ресурсы. Начните с приобретения надежных дисковых массивов, настройте их должным образом для обеспечения быстрого надежного доступа к данным, контролируйте их производительность и скорость заполнения. Вы должны постоянно выполнять мониторинг доступного пространства для своевременного выполнения архивирования данных или планирования времени расширения системы при ее росте.

Буферный пул

Для минимизации нагрузки на диски база данных сохраняет блоки данных в оперативной памяти с целью исключения обращений к диску при повторном их использовании. Снизить такую активность можно следующими способами:

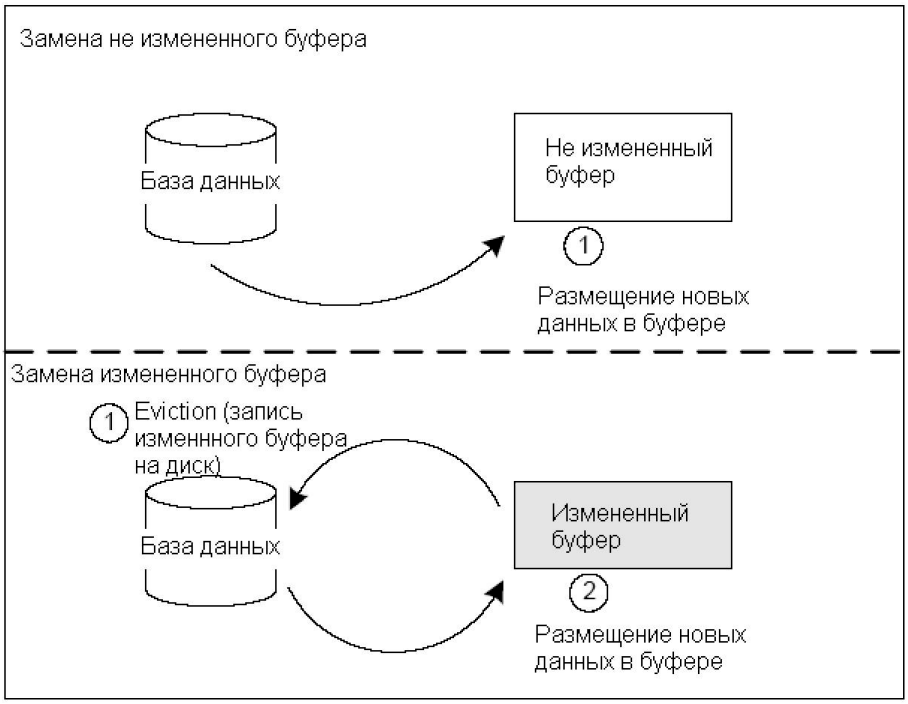
- увеличить буфер базы данных <-B>;
- использовать частные буферы <-Bp>;
- использовать APW;
- изменить структуру базы данных (количество областей хранения данных).

Буферный пул базы данных – это временная область хранения, размещенная в памяти и содержащая копии блоков базы данных, прочитанных с дисков. На следующем рисунке показано, как происходит работа базы данных с использованием буферного пула:



Когда процесс запрашивает доступ к блоку базы, находящемуся не в буферном пуле, движок базы данных вынужден заменить содержимое какого-либо буфера. Начинается его поиск. Идеальным кандидатом для замены является незаблокированный и неизменённый буфер. Замена такого буфера происходит только за один шаг: просто записывается новое содержимое. Если буфер содержит изменённые данные, то они должны быть сначала выселены из него. Выселение данных происходит за два шага: содержимое буфера

записывается на диск, после чего в буфер записывается новое содержимое. Этот процесс, несомненно, медленнее, генерирует высокую нагрузку.



Поиск кандидата на замену осуществляется максимум для первых десяти буферов. Если поиск не дал результатов, т.е. не найден незаблокированный и неизменённый буфер, то будет использован первый найденный незаблокированный изменённый буфер.

Таким образом, чем чаще и больше используется буфер, тем меньше и реже используются диски. Поскольку быстродействие памяти выше быстродействия дисков, то вы получите лучшую производительность с большим процентным значением Buffer hits. Buffer hits – это процент количества полученной информации из буферного пула базы данных, относительно полученной информации с дисков. Например, значение Buffer hits, равное 90%, эквивалентно десяти считываниям с диска для каждых 100 запросов менеджера базы данных. Если вы увеличите Buffer hits до 95%, то при том же количестве запросов с диска будет выполняться только 5 операций чтения, т.е. количество чтений с диска будет уменьшено на 50%. Таким образом, незначительное увеличение значения Buffer hits может привести к значительному уменьшению дисковой активности. Чем больше будет увеличиваться Buffer hits, тем больше будет заметно уменьшение дисковой активности, т.е. увеличение с 95% до 96% приведет к уменьшению дисковой активности уже на 20%.

Для определения эффективности работы буфера необходимо осуществлять мониторинг значения поля Buffer Hits с помощью утилиты Promon и ее экрана Activity: Summary (R&D → 2 → 1). Наилучшая производительность достигается увеличением значения параметра запуска базы данных Blocks in Database Buffers <-B> так, чтобы значение Buffer hits превышало 95% или пока система не начнет свопинг (*swapping).*

Пример экрана Activity: Summary:

```
01/22/10        Activity: Summary
17:54:48        01/22/10 13:12 to 01/22/10 17:54 (4 hrs 41 min)

Event                    Total  Per Sec |Event                    Total  Per Sec

Commits                      9      0.0 |DB Reads                  101      0.0
Undos                        1      0.0 |DB Writes                   4      0.0
Record Reads              2346      0.1 |BI Reads                    9      0.0
Record Updates               1      0.0 |BI Writes                   4      0.0
Record Creates               1      0.0 |AI Writes                   0      0.0
Record Deletes               0      0.0 |Checkpoints                 0      0.0
Record Locks               477      0.0 |Flushed at chkpt            0      0.0
Record Waits                 0      0.0 |Active trans                0

Rec Lock Waits     0 %    BI Buf Waits      0 %    AI Buf Waits      0 %
Writes by APW      0 %    Writes by BIW     0 %    Writes by AIW     0 %
DB Size:        2107 MB   BI Size:        128 MB   AI Size:          0 K
Empty blocks:    326      Free blocks:      0      RM chain:    537652
Buffer Hits       98 %    Primary Hits      98 %   Alternate Hits    0 %
```

Рекомендации по настройке буферного пула. Каждый буфер - это один блок базы данных. По умолчанию значение этого параметра слишком мало. Для начала можно отталкиваться от значения, равного дес

яти процентам от размера базы данных. Затем его можно увеличивать или уменьшать, контролируя параметр Buffer Hits, как указано выше. **Примечание:** не увлекайтесь чрезмерным увеличением, т.к. из-за нехватки памяти система может начать свопинг, что приведет к обратно пропорциональному результату - к резкому снижению производительности всей системы.

Начиная с версии OpenEdge 10.1C, появилась возможность увеличивать размер буферного пула без остановки базы данных. Для этого необходимо воспользоваться командой:

```
proutil db-name -C increaseto -B n
```

Здесь *n* – это аргумент параметра <-B>, которым указывается новый размер буферного пула. **Внимание!** С помощью этой команды можно только увеличить размер, не существует способа уменьшить его без остановки базы данных. После удачного увеличения размера буферного пула не забудьте установить новое значение в параметрах запуска базы данных. В противном случае после перезагрузки базы данных размер буферного пула будет иметь старое значение.

Движок базы данных так же использует хеш-таблицы для уменьшения времени размещения буфера базы данных. Параметр запуска базы данных Hash Table Entries <-hash> управляет количеством хеш-таблиц в буферном пуле. По умолчанию - 25% от размера буферного пула. В большинстве случаев этого значения достаточно. Тем не менее, увеличение этого значения может немного уменьшить время, необходимое для поиска блока в буферном пуле.

На предыдущем экране помимо Buffer Hits жирным шрифтом выделены еще два значения - Primary Hits и Alternate Hits. Эти поля появились в версии OpenEdge 10.2B. Они также относятся к буферному пулу. В этой версии добавлен дополнительный буферный пул, названный Альтернативным. Стандартный буферный пул не требует особых настроек, кроме изменения одного параметра запуска базы данных, в то время как настройки Альтернативного буферного пула потребуют от администратора определенных затрат

времени и более серьезных действий. Альтернативный буферный пул призван повысить производительность буферного пула базы данных и, соответственно, всей базы данных в целом. Более детально об Альтернативном буферном пуле можно прочитать в книге «Альтернативный буферный пул в OpenEdge 10.2B», находящейся на сайте www.openedge.ru.

Использование APW

Процесс APW[33] - это программа асинхронной записи измененных блоков базы данных из памяти на жесткий диск. Для работы процесса необходима лицензия OE Enterprise RDBMS. Использование APW настоятельно рекомендуется для улучшения производительности базы данных, причин тому несколько.

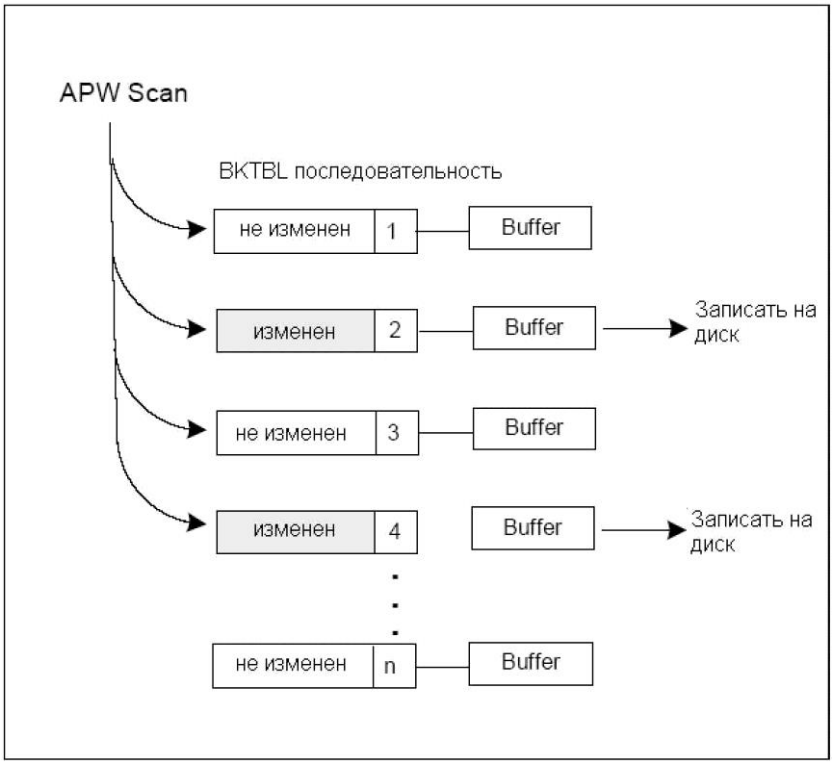
- Процессы APW гарантируют, что всегда будут доступны пустые буферы, и движку базы данных не нужно ждать освобождения очередного заполненного буфера.
- Эти процессы уменьшают количество буферов, которые необходимо проверять перед тем, как записать измененный буфер на диск. Для сохранения в памяти наиболее активных буферов движком базы данных на диск записываются только наиболее давно использованные буферы, а на их поиск требуется время. Вместо движка базы этой работой занимаются APW.
- Процессы APW снижают нагрузку на систему, возникающую в момент формирования контрольной точки (checkpoint), т.к. APW-процессы записывают на диск измененные буферы до возникновения контрольной точки, снижая таким образом их количество.

База данных может иметь один, более одного или ни одного APW-процесса. Количество процессов зависит от ваших приложений и окружающей их среды. Для проверки достаточности количества APW-процессов можно использовать утилиту PROMON. Для этого запустите один APW-процесс. Если в момент возникновения контрольных точек буферы будут сбрасываться, добавьте еще один APW-процесс или увеличьте размер BI-кластера, после чего выполните проверку снова.

APW-процессы являются самонастраивающимися. Это означает, что после их запуска не нужно делать настроек каких-либо дополнительных параметров. Тем не менее, для увеличения скорости их работы можно увеличить размер BI-кластера, для чего использовать утилиту PROUTIL с классификатором TRUNCATE BI.

APW-процессы непрерывно записывают измененные буферы на диск, делая менее вероятным ожидание сервера при поисках неизменённых буферов. Для поиска изменённых буферов APW-процесс сканирует последовательность Block Table (BKTBL). Последовательность BKTBL – это список ссылок на структуры BKTBL, где каждая ссылка указывает на конкретный буфер базы данных. Каждая структура BKTBL содержит флаг, указывающий на то, что буфер изменен. Как только APW находит такой буфер, он немедленно сбрасывает его на диск. Следующий рисунок демонстрирует сканирование процессом APW последовательности BKTBL.

[33] APW, от англ. Asynchronous Page Writer

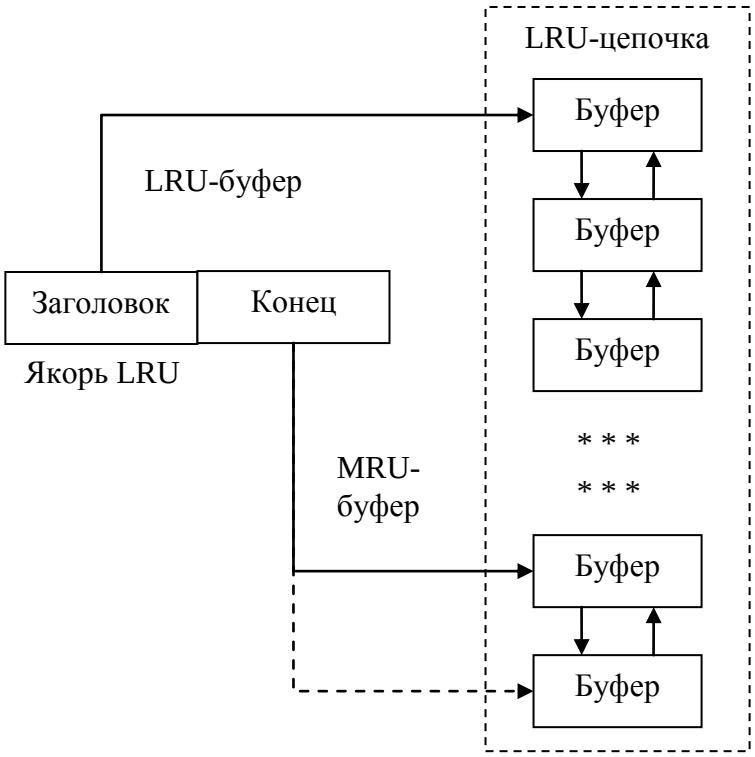


Сканирование BKTBL-последовательности является циклическим процессом. После завершения каждого цикла APW «засыпает». Следующий цикл будет начат с того места, на котором был завершен предыдущий цикл. Например, если во время первого цикла APW сканировал буферы с первого по десятый, то следующий цикл начнется с одиннадцатого буфера.

Когда движок базы данных записывает изменённый буфер на диск, он помещает его в MRU[34]-цепочку. Это сделано потому, что вероятность потребности в более старых данных очень мала. Для поиска таких буферов APW сканирует последовательность LRU. Последовательность LRU[35] - это список ссылок на буферы в памяти, который движок базы использует для получения доступа к буферам. LRU-цепочка - это фиксированная структура, указывающая на начало и конец последовательности. Каждый раз, когда происходит обращение к буферу, сервер блокирует и изменяет LRU-цепочку, перемещая буфер в ее конец.

[34] MRU - от англ. Most Recently Used
[35] Последовательность LRU - от англ. Least Recently Used chain

Рисунок демонстрирует работу цепочки LRU.



Поскольку все процессы, обращающиеся к LRU, должны блокировать для изменения LRU-последовательность, то между ними создается конкуренция. Это приводит к снижению производительности, особенно в высоко загруженных системах. APW-процесс снижает уровень конкуренции, периодически очищая измененные буферы. Следовательно, движок базы данных при необходимости всегда может быстро найти неизменённый буфер.

Еще одна причина использования APW - это увеличение производительности за счет снижения нагрузки, связанной с контрольными точками. Файл Before-Image разделен на кластеры. Когда кластер становится заполненным, возникает контрольная точка. Если информация в кластере к этому времени уже сохранена, кластер просто становится доступным для повторного использования. Многократное использование кластера уменьшает размеры дискового пространства, необходимого для работы BI. Контрольные точки обеспечивают повторное использование кластеров и при необходимости восстановление базы данных в короткие сроки. Во время контрольной точки движок базы записывает все измененные кластеры на диск. Это существенно увеличивает нагрузку, особенно, если имеются большие BI-кластеры и большой буферный пул. APW-процесс минимизирует нагрузку за счет постоянной записи измененных буферов на диск. Таким образом, в момент контрольной точки на диск будет записано меньшее количество буферов.

Изменение структуры базы данных

Довольно часто имеют место быть такие ситуации, когда в системе наблюдается явная перегрузка одного диска при том, что другие диски откровенно простаивают. Сама по себе перегрузка диска уже является узким местом в системе. Так называемый дисковый дисбаланс можно контролировать с помощью команды iostat:

Disks:	% tm_act	Kbps	tps	Kb_read	Kb_wrtn
Hdisk0	30.6	528.5	71.1	5616	57811
Hdisk1	15.0	393.2	42.0	5244	41952
hdisk2	29.3	803.2	80.0	7604	88795
hdisk3	0.0	0.0	0.0	0	0
hdisk4	0.0	0.0	0.0	0	0
hdisk5	90.7	4236.0	427.7	313056	195352
hdisk6	0.0	0.0	0.0	0	0

Здесь видно, что диск hdisk5 используется в разы больше, чем другие диски. Для исключения этого дисбаланс необходимо распределить экстенты базы данных между несколькими дисками, выровняв тем самым нагрузку. Первое с чего стоит начать, это вынести на отдельный диск экстенты Before-Image, затем экстенты After-Image. Если это не помогло, то необходимо выполнить мониторинг базы данных на предмет обнаружения наиболее активных областей хранения, у обнаруженных областей перенести все или часть экстентов на отдельный диск. Мониторинг активности использования областей хранения, вплоть до активности каждого экстента, можно выполнить на экране «I/O Operations by File» утилиты Promon (R&D → 2 → 9). Пример.

	Total	Per Min	Per Sec	Per Tx
sports_26.d1				
Reads	**21**	**10**	0.17	0.03
Writes	6	3	0.05	0.01
Extends	0	0	0.00	0.00
	Total	Per Min	Per Sec	Per Tx
sports_27.d1				
Reads	**24046**	**12023**	200.38	37.87
Writes	24	12	0.20	0.04
Extends	0	0	0.00	0.00

Здесь видно, что из экстента sports_27.d1 данные считываеются активнее, чем из экстента sports_26.d1. Следовательно, он является кандидатом для выселения на отдельный диск. Это, конечно же, усеченный пример, для принятия окончательного решения необходимо проанализировать все экстенты.

Прямой ввод/вывод

База данных может использовать технику дискового ввода/вывода, которая вынуждает блоки записываться непосредственно из буферного пула на диск. Эта техника избавляет запись на диск от задержек, создаваемых буферным менеджером операционной системы, и называется прямым вводом/выводом (Direct I/O).

Обычно прямой ввод/вывод используется, когда недостаточно оперативной памяти; буферный менеджер операционной системы не обеспечивает хорошую производительность. Протестируйте влияние Direct I/O перед применением его на вашей промышленной базе.

Для использования этой техники воспользуйтесь параметром запуска базы данных Direct I/O <-directio>. При использовании этого параметра базе данных требуется большее время для отработки APW, поэтому необходимо запустить дополнительный APW-процесс для компенсации этого времени.

Стоит заметить, что наилучший эффект от использования прямого ввода/вывода достигается на операционной системе AIX.

Оперативная память

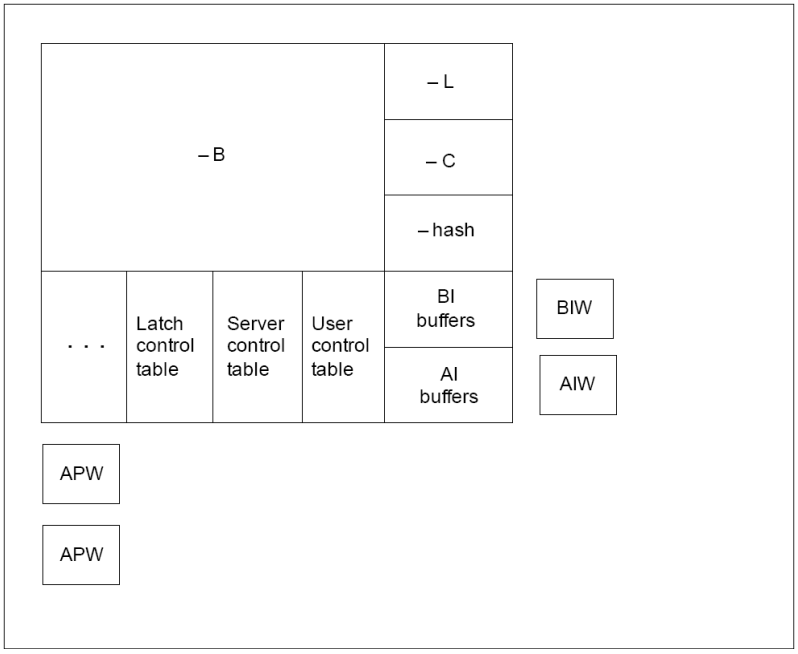
Большинство способов улучшения производительности базы данных сводятся к необходимости использования дополнительных ресурсов оперативной памяти для того, чтобы уменьшить дисковую активность. Но если ресурсы памяти сильно ограничены, то бесконтрольное использование этих способов может привести к перегрузке оперативной памяти, и как следствие, к началу интенсивной подкачки страниц. Стоит заметить, что подкачка страниц присутствует всегда, это часть механизма использования памяти. Но слишком активная подкачка нежелательна, т.к. при этом может легко произойти перегрузка, которая будет говорить о том, что для текущей нагрузки памяти не хватает. Такие перегрузки крайне отрицательно влияют на производительность, т.к. нагрузка на процессоры и подсистемы ввода/вывода в это время быстро превышает полезную нагрузку системы. И в итоге это может привести к тому, что в системе ни один процесс не сможет работать, поскольку сервер будет расходовать свои ресурсы на перемещение страниц в память и из памяти.

Существует два типа подкачки страниц, это физическая и виртуальная. Физическая подкачка это, когда выполняется процесс перемещения страниц памяти между оперативной памятью и дисками в обоих направлениях. При этом пространство, используемое для размещения страниц памяти на диске, называется пространством подкачки. Виртуальная подкачка, это когда страницы памяти перемещаются из одного места в памяти в другое в той же памяти. Администратору следует уделять большое внимание именно физической подкачке. Если физическая подкачка будет оставаться на высоком уровне продолжительное время, то нужно будет перераспределить или добавить больше памяти.

Использование памяти в OpenEdge

Первичный брокер базы данных выполняет распределение памяти между пользователями для доступа к базе данных. Для получения параллельного доступа к информации без искажения этой информации пользователям предоставляются определенные ресурсы в памяти, т.к. попытка изменить один и тот же участок памяти двумя пользователями может привести к повреждению разделяемой памяти. Латчи (latches), далее будет использоваться именно этот термин, предохраняют память от подобных повреждений. Латчи похожи на блокировки (locks), но применяются к разделяемым ресурсам в памяти. Когда процесс блокирует запись, это обеспечивает возможность изменения записи без помех со стороны других процессов, желающих изменить эту же запись. Латч - это блокировка разделяемой памяти, позволяющая пользователю вносить изменения в блок памяти без влияния на других пользователей.

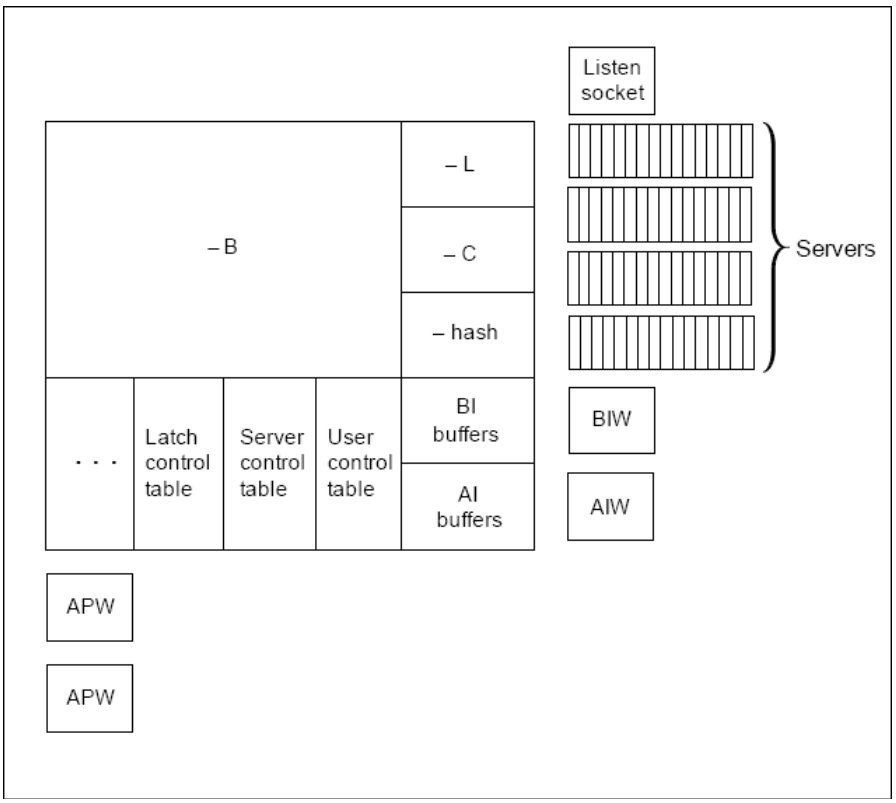
На следующем рисунке представлена схема ресурсов разделяемой памяти. Эта иллюстрация является только примером, поскольку не является полной. Буферы базы данных обычно занимают более 90% разделяемой памяти, а различные ресурсы, управляемые латчами, менее 1%.



На рисунке показано, как много различных ресурсов входит в состав разделяемой памяти. Локальные пользователи, такие как конечные пользовательские процессы и фоновые процессы, изменяют эту структуру. Если два пользовательских процесса одновременно получают доступ к базе данных, и оба должны изменить таблицу блокировок <-L>, то они запрашивают ресурс, проверяя контрольную таблицу латчей. Если ресурс доступен, то на него устанавливается латч, использующий операционный вызов, блокирующий обращение пользователей к этому ресурсу. Как только латч установлен, пользователь приступает к изменению ресурса, после чего освобождает латч. Запрос пользовательского процесса, поступивший в это же время на этот же ресурс, будет повторяться до тех пор, пока ресурс не будет освобожден предыдущим занявшим его пользовательским процессом.

Жизненно важным ресурсом базы данных являются её буферы. Они обеспечивают кэширование часто используемых данных, что позволяет сократить операции считывания данных с диска. Поскольку скорость работы памяти во много раз превышает скорость работы дисков, то при должной настройке буферов производительность системы значительно увеличивается.

На следующем рисунке представлен процесс добавления дистанционных клиентов, использующий TCP/IP-соединения и серверные процессы. Дистанционный клиент отправляет запрос в listen socket, который в свою очередь сообщает о нем процессу брокера. Брокер, используя контрольную таблицу пользователей и контрольную таблицу сервера (server control table), определяет, может ли пользователь зарегистрироваться, и через какой сервер он может подключиться. Если сервер будет недоступен, то в зависимости от параметров серверов этого брокера будет запущен новый сервер. Такими параметрами являются <-Mn>, <-Mi> и <-Ma>, они контролируют количество запускаемых серверов, количество клиентов на сервер и условия запуска нового сервера. Как только соответствующий сервер определен, между ним и дистанционным клиентом устанавливается двунаправленная связь. Эта связь будет до тех пор, пока пользователь не отключится или пока брокер не будет остановлен.



В OpenEdge используется технология, когда программный код загружается в память по требованию. Эта технология улучшает производительность, загружая только код, необходимый для исполнения отдельных функций, минуя загрузку и обработку приложения или библиотеки данных целиком. Такие программы так же известны, как разделяемые программы, т.е. когда часть программы размещена в памяти и доступна пользователям для выполнения по запросу. Такими программами являются программы брокера или сервера, которые динамически располагаются в памяти для каждого пользователя. В OpenEdge память распределяется динамически, исходя из количества пользователей и некоторых параметров запуска брокера.

Расчетным значением объема необходимой памяти, используемой брокером базы данных, является 110% от значения параметра буферного пула базы данных <-B>. Однако если у вас установлено высокое значение для таблицы блокировок <-L> или для индексных курсоров <-c>, то значение может быть увеличено. Блокировка одной записи использует 64 байта, а каждый индексный курсор 84 байта. Кроме того, если у вас установлен очень низкий параметр для буферов базы данных, то расходы памяти по другим параметрам будут достигать более чем 10% от значения <-B>.

Например, если количество буферов базы данных установлено в 20 000 при размере блока базы 8Кб, то выделяется 160000Кб памяти для буферов. При добавлении 10% от этой суммы общее количество памяти для брокера базы данных будет достигать 176000Кб или 176 Мб.

Серверы для дистанционных клиентов используют примерно от 3Мб до 5Мб. Количество таких серверов регулируется параметром <-Mn>. По умолчанию оно равно пяти.

Клиентские процессы меняются в зависимости от выбранных опций запуска. Тем не менее, при средних значениях <-mmax> и <-Bt> выделение новой памяти для каждого процесса будет примерно от 5Мб до 10 Мб. Этот диапазон тоже относится к прикладным

процессам сервера. Дистанционные пользователи обычно используют больше памяти (от 10Мб до 20Мб), поскольку им необходимы большие значения <-mmax> и <-Bt>, чтобы обеспечить приемлемую производительность по сети. Требования для памяти дистанционных пользователей (т.е. настройки <-mmax> и <-Bt>) не влияют на требования памяти на host-машине.

Еще один параметр, который значительно влияет на использование оперативной памяти, это <-pinshm>. Благодаря этому параметру вы можете заблокировать пространство разделяемой памяти для использования только базой данных. В результате, память будет использоваться только для нужд конкретной базы данных, следовательно, страницы памяти не будут перемещаться на диск по требованию иных процессов, не относящихся к базе. Но здесь необходимо быть осторожным, если у вас запущено несколько баз данных на одном сервере, и они все используют блокировку разделяемой памяти, то это может привести к тому, что другим процессам памяти будет не хватать, и в результате вы можете только ухудшить ситуацию.

Начиная с версии OpenEdge 10.1B и выше, распределение разделяемой памяти происходит динамически. До этого момента сегменты разделяемой памяти были фиксированного размера, теперь же оптимальный размер сегментов рассчитывается динамически во время работы базы данных. Брокер пытается создать сегмент с наиболее возможным размером и наименьшим количеством сегментов, требуемых для разделяемой памяти. Результат – более эффективное распределение сегментов памяти с большим размером. К тому же становится возможным создавать большее количество самообслуживающихся соединений с большой разделяемой памятью. Максимальный размер сегмента разделяемой памяти может быть определен при старте базы данных. Увеличение размера сегмента уменьшает их количество. Размер определяется параметром запуска Shared memory segment size <-shmsegsize>.

Размер разделяемой памяти, выделяемой для базы данных, зависит от вашей операционной системы и имеет ограничения.

- Нельзя создать сегменты разделяемой памяти, имеющие размеры б**о**льшие, чем это определено параметром операционной системы (Maximum Shared Memory Segment Size). Если параметр <-shmsegsize> имеет большее значение, чем установлено системой, то для создания сегмента будет использоваться настройка операционной системы. Для многих UNIX-систем максимальный размер сегмента устанавливается параметром ядра SHMMAX.
- Нельзя создать больш**е**е количество сегментов разделяемой памяти, чем разрешено операционной системой. Этот параметр нельзя настроить практически ни в одной операционной системе, за исключением некоторых UNIX-платформ, в которых настройка происходит через параметр ядра SHMSEG.
- Нельзя создать разделяемые сегменты памяти, размеры которых превышают допустимо возможное адресное пространство для системы. Если запрашиваемое количество разделяемой памяти превышает способности системы, сервер просто не будет запущен.

Память - ограниченный ресурс, поэтому важно использовать его правильно. Большое значение имеет в первую очередь повышение объема памяти для параметров брокера, что влияет на всех пользователей и систему в целом. Этот факт демонстрируется примером, рассматривающим применение Buffer hits базы данных, который показывает, как незначительное изменение параметра брокера базы может повлиять на всю систему в целом. Изменения на стороне брокера обычно влияют значительно меньше, чем

изменения, сделанные на стороне «самообслуживающихся» клиентов. Например, увеличение параметра <-B> для брокера базы данных с размером блока 8Кб на 1000 буферов будет стоить вам восьми мегабайт оперативной памяти; увеличение параметра <-mmax> на 80Кб для 100 пользователей обойдется так же в 8Мб. Однако увеличение количества буферов будет иметь большее значение, чем изменения свойств клиента. Этот простой пример хорошо показывает, что всегда сначала необходимо настраивать брокер базы данных. Как только вы это сделаете (при этом у вас еще останется свободная память), вы можете смело переключать свое внимание на параметры «самообслуживающихся» клиентов.

При необходимости уменьшения использования памяти необходимо следовать обратному принципу - сначала снижать использование памяти у самообслуживающихся клиентов, и только потом, если это не помогло, можно снизить память брокера базы данных.

Процессоры

На работу CPU оказывают влияние практически все ресурсы системы. Основным потребителем CPU является прикладной код. Таким образом, изменяя приложение, можно оказывать большое влияние на CPU, но обычно это неподвластно администратору базы данных. Как администратор, вы можете снизить нагрузку на CPU косвенно, изменяя параметры других ресурсов, с которыми работает приложение. Например, медленные диски увеличивают активность использования CPU за счет увеличения ожидания операций ввода/вывода, поэтому для снижения нагрузки на CPU необходимо решить проблему с медленными дисками. Существует четыре категории использования CPU.

1. Пользовательское время - время, которое CPU тратит на выполнение пользовательских задач. Это основная задача CPU.
2. Системное время - время, требуемое на выполнение системных задач, таких как переключения контекста, работа планировщиков и тому подобные задачи. Это время нельзя полностью исключить, но его можно минимизировать.
3. Время ожидания – время, которое CPU тратит на ожидание какого-либо ресурса, например, на ожидание дискового ввода/вывода.
4. Время простоя - время, когда CPU не используется. Если в рабочей очереди нет процессов, и CPU не ожидает ответа от другого ресурса, то это время является временем простоя. В некоторых системах ожидание ввода/вывода рассматривается, как простой потому, что CPU неактивен, поскольку находится в ожидании ответа.

Неопытному администратору время, затраченное на ожидание ввода/вывода, может показаться нежелательным. Конечно, если такие ожидания сильно влияют на производительность, то здесь есть повод для беспокойства, тем не менее, не всегда так. Это значение времени формируется каждый раз, когда CPU завершает задачу и ожидает некоторый другой ресурс - диск или память, если CPU значительно быстрее, чем другие ресурсы, и в результате он некоторый процент времени вынужден ожидать эти медленные устройства.

В настроенной системе, возможно, но не обязательно, некоторое время простоя. Возможно, что процессорное время будет использоваться на 100 % в многопроцессорной среде, но это не подразумевает, что система работает плохо, это может означать, что система работает с такой скоростью, с какой и может максимально работать. При настройке производительности вы можете попытаться исключить узкое место за счет самого быстрого ресурса – CPU. В идеальном состоянии система должна использовать

70% пользовательского времени, 20% системного времени, 0% времени ожидания ввода/вывода и 10% времени простоя.

Соотношение пользовательского времени к системному времени должно быть приблизительно три к одному. Однако это соотношение значительно изменяется, когда пользовательское время становится ниже 20% от общего времени CPU, а системное время начинает значительно увеличиваться. В некоторых ситуациях системное время из-за неправильного распределения ресурсов может принимать б**о**льшие значения. Но при увеличении пользовательского времени с помощью определенных настроек производительности системное время должно выравняться до одной трети от пользовательского времени или меньше. Это можно определить любым инструментом, позволяющим следить за работой CPU в вашей системе.

Признаки перегрузки CPU могут быть только индикаторами ухудшения работы других ресурсов. В основном это происходит из-за проблем с дисками. Если вы видите, что CPU является узким местом, то сначала необходимо проверить отсутствие зависших бесконтрольных процессов в системе, после чего убедиться в нормальной работе других ресурсов системы.

Если у вас наблюдается время простоя, это означает, что у CPU еще есть запас для улучшения производительности. Необязательно, чтобы значение времении простоя всегда было больше нуля. Если время простоя нулевое в течение длительных периодов и время ожидания так же не имеет существенных значений, то вам нужно более глубоко исследовать деятельность CPU, чтобы найти причину и определить дальнейшие действия.

Например, обратите внимание на глубину очереди CPU. Глубина очереди CPU - это количество процессов, которые ожидают использования CPU. Если в очереди постоянно наблюдается несколько процессов, то выполните следующее:

- увеличьте возможности и эффективность CPU;
- измените код приложения или перенесите его работу на менее загруженный период дня.

Если ожидание ввода/вывода высокое, а время простоя отсутствует, то нужно улучшить эффективность использования дисков, уменьшив ввод/вывод или перераспределив между ними время обработки. Если значение ожидания ввода/вывода равно 10% или менее, и все еще наблюдается простой, то никаких срочных мер принимать не нужно.

CPU и параметр <–spin>

В многопроцессорных системах механизм spin-блокировок используется для синхронизации работы процессов с данными, размещенными в разделяемой памяти, т.е. чтобы оградить эти данные от одновременного изменения несколькими пользователями. Как это работает? Когда процесс обращается к ресурсу в разделяемой памяти, он пытается получить его исключительную блокировку (латч). Когда он получает эту блокировку, он получает исключительный доступ к ресурсу. Все остальные процессы, пытающиеся получить блокировку в это время, потерпят неудачу, т.к. текущий процесс блокирует ресурс. Процессы будут продолжать попытки получить к нему доступ до тех пор, пока он не освободится. Этот циклический процесс называется spinning. Если же после определенного количества попыток получить блокировку так и не удалось, то процесс приостанавливает свою работу на некоторое время, после чего опять повторяет попытку

получения доступа. При этом с каждой попыткой продолжительность паузы между попытками увеличивается. Проблема заключается именно в количестве таких пауз, когда процесс вынужден засыпать и просыпаться, поскольку на выполнение этих действий необходимы определенные затраты времени CPU. А так как обычно ресурс бывает занят непродолжительное время, то гораздо эффективнее многократно запрашивать к нему доступ в течение одного цикла, чем обращаться пару раз, перемещаясь в конец очереди CPU, засыпать и ждать своей следующей попытки, тем самым увеличивая количество циклов. Таким образом, параметр <-spin> предназначен для установки количества обращений к занятому ресурсу, перед тем как процесс обращение будет приостановлен.

Для установки оптимального значения этого параметра необходимо довольно много времени для мониторинга и анализа. Для начала установите значение в пределах от 10 000 до 20 000, обычно этого бывает достаточно. Что такое - достаточно или нет? Для определения достаточности воспользуйтесь утилитой Promon. Мониторинг осуществляется с помощью экрана Performance Indicators в меню R&D (R&D → 3 → 1). Здесь необходимо следить за полем Latch Timeouts. Его значение необходимо держать как можно более низким путем увеличения значения параметра <-spin>, но не стоит увлекаться, всегда при настройке параметра <-spin> отслеживайте показатели использования CPU, т.к. как вы увеличиваете на них нагрузку. Следите так же за процентом работы APW-процессов на экране Activity Summary, т.к. часто снижение значений Latch Timeout приводит к тому, что APW-процессы больше не способны поддерживать свою работу на приемлемом уровне. Обычно это означает, что пришло время обратить внимание на размеры BI-блока и BI-кластера, или, что можно запустить дополнительный APW-процесс. Обратите внимание на загруженность дисковой системы, если она является узким местом, то увеличение spin и увеличение использования CPU, скорее всего, не приведут к положительным результатам, и наиболее вероятно наблюдение обратного эффекта.

Эффект от настройки <-spin> зависит от того, какая аппаратная конфигурация и какая операционная система у вас установлена. Поэтому результаты не обязательно могут быть одинаковыми для разных систем.

Изменять значение параметра <-spin> можно без остановки базы данных. Для этого необходимо воспользоваться меню Spins before timeout утилиты Promon (R&D → 4 →4 → 1). Новое значение параметра вступит в силу немедленно. Помните, если новое значение вас устроило, то не забудьте внести соответствующие изменения в параметры запуска базы данных, чтобы новое, а не старое значение вступило в силу после перезапуска базы данных.

Быстрые CPU против медленных CPU

Лучшее решение - иметь множество быстрых процессоров в вашей системе. Тем не менее, в большинстве случаев обычно имеется в наличии либо много медленных CPU, либо один быстрый CPU. Польза от быстрых процессоров в том, что они хорошо выполняют однопоточные операции. Однопоточной операцией может быть, например, закрытие операционного дня. С другой стороны, несколько CPU обеспечат параллельное выполнение нескольких различных задач. С точки зрения эффективности последнее имеет очевидное преимущество.

Так что же выбрать? Чтобы выбрать лучшее решение, необходимо посмотреть на ваши задачи и ваше приложение. Например, приложение, которое выполняет только однопоточные операции, будет работать лучше на быстрых процессорах, даже если в

системе будет только один CPU. Для приложений, обрабатывающих в основном входные данные, полезны схемы, имеющие большее количество медленных процессоров.

Прочие ресурсы операционной системы

На производительность баз данных OpenEdge помимо дисков, памяти и CPU оказывают влияние такие ресурсы системы как семафоры, файловые дескрипторы, а также количество возможных запущенных процессов и т.п.

Семафоры

Семафоры обеспечивают доступ к общим объектам данных для множества различных процессов, они предоставляют процессу «лицензию» на блокировку ресурсов, таких как записи в таблице; разделяемую память; латчи и т.д. OpenEdge разделяет семафоры на две группы: для входа в систему и для пользовательских процессов. Для входа в систему используется так называемый логин-семафор, через который к базе данных должен подключатся каждый процесс, и который используется только с этой целью. Пользовательские семафоры используются пользовательскими процессами, уже подключенными к базе данных, для блокировки записи или другого ресурса. Оба типа этих семафоров размещены в одном и том же пуле, который называется *semset*. Число семафоров в пределах этого набора определено параметром запуска <-n> для каждой базы данных.

Как и почему необходимо использовать параметр запуска базы данных Multiple Semaphore Sets <-semsets>? С самого начала в OpenEdge имелся только один *semset*, имевший существенный недостаток - в любой момент времени его мог использовать только один процесс, независимо от того, регистрируется ли процесс в базе данных, или выполняет обычную операцию обработки данных. При большом количестве процессов образовывалась очередь за доступом к этому пулу, что приводило к возникновению узкого места в работе базы данных.

Следующая проблема заключалась в том, что OpenEdge всегда успешно запускал процесс сервера, даже если ядро операционной системы не было способно предоставить достаточное количество семафоров для каждого процесса. В этом случае база использует столько семафоров, сколько может предложить ядро операционной системы, после чего они циклически распределяются между имеющимися процессами, не зависимо от того, используют они их в качестве логин-семафора или в качестве пользовательских семафоров. В этой точке мы получали выполнение «*double connection*» и симптомы нехватки семафоров, проявляющиеся в виде сообщения в логе базы данных:

```
«Warning: only x available wait semaphores».
```

До сих пор настройкой ядра ОС при увеличении количества доступных семафоров можно было решить только последнюю проблему. Выполнение настройки OpenEdge для эффективного использования семафоров не имело другого способа, кроме использования параметра <-spin>.

Начиная с версии OpenEdge 8.3A, осуществлено использование множества наборов семафоров на UNIX-платформах (на NT это всегда было значением по умолчанию). В новой реализации логин-семафор и пользовательские семафоры разделены по различным наборам. Логин-семафор постоянно находится в первом наборе, в то время как все остальные наборы будут содержать только пользовательские семафоры. В системах с

большим количеством пользователей конкуренция за семафоры связана с производительностью, теперь мы можем предоставить большему количеству процессов одновременный доступ к различным ресурсам, позволив им обращаться более чем к одному пулу семафоров.

Как определить необходимое значение параметра <-semsets>? Для каждых 20 - 50 пользователей рекомендуется иметь 1 набор семафоров. Например, если у базы данных <-n> равно 2500, то для <-semsets> устанавливаем значение 50.

Как определить наличие конкуренции за семафоры на рабочей базе данных OpenEdge? Хорошим инструментом для определения конкуренции может стать утилита *sar* с параметром *-u*. Если значение поля *%sys* становится выше 25%, то, вероятно, возникает конкуренция за семафоры, при этом будет отмечаться деградация производительности, связанная с добавлением большого количества пользователей.

Также можно воспользоваться экраном Other меню R&D утилиты Promon (R&D → 2 → 13).

```
                        Total         Per Min          Per Sec          Per Tx

Commit               12533050            1238            20.63            1.00
Undo                       39               0             0.00            0.00
Wait on semaphore       17064               2             0.03            0.00
```

Если пользовательские процессы для доступа к ресурсу ожидают семафор, то это покажет поле Wait on semaphore. Оптимальное значение этого поля на разных системах может значительно различаться.

Для установки необходимого количества наборов семафоров воспользуйтесь следующей командой:

```
proserve db-name -semsets 2 -n 100
```

где *2* - число наборов семафоров,
-n - количество пользователей.

Использование Multiple Semaphore Sets <-semsets> возможно только с лицензией Enterprise.

Файловые дескрипторы

Файловый дескриптор – это идентификатор, который назначается файлу при его открытии. Система обычно ограничена количеством таких дескрипторов. Каждый процесс базы данных использует несколько дескрипторов, как для файлов базы данных, так и для обычных файлов приложения. Возможно, что вам придется изменить системный предел количества файловых дескрипторов.

Процессы

В OpenEdge брокеры, серверы, клиенты и фоновые программы записи запускаются в качестве процессов. Параметры ядра операционной системы ограничивают количество активных процессов, которые могут работать в системе. Поэтому, возможно, вам придется поменять эти параметры в сторону увеличения.

В Unix-системах параметр ядра NPROC ограничивает общее количество активных процессов в системе и обычно имеет значение в интервале между 50 и 200. Параметр MAXUP ограничивает количество параллельных процессов, которые могут быть запущены одним пользовательским ID, и обычно установлен в интервале от 15 до 25. Если под одним пользовательским ID в системе зарегистрируется несколько пользователей, то параметр MAXUP будет достаточно быстро превышен. Когда это предел превышен, необходимо искать следующие сообщения об ошибке: «Unable to fork process». Если это сообщение повторяется многократно, необходимо перенастроить ядро системы.

Глава 17 Фрагментация базы данных

В связи с тем, что одни данные удаляются, а другие создаются, через определенное время на диске появляются блоки с фрагментами записей, это называется фрагментацией данных. Фрагментация может появиться вследствие неправильно подобранных параметров областей хранения данных. Фрагментация базы данных является одним из факторов, значительно влияющим на производительность, увеличивая нагрузку на диски за счет необходимости обработать большее количество блоков базы данных за одну операцию чтения или записи. Если наблюдается высокая нагрузка на диски, на которых размещаются экстенты базы данных, то в первую очередь необходимо проверить, не фрагментирована ли база данных?

Устраняется фрагментация с помощью перезагрузки данных - этот процесс называется Dump&Load. Чтобы не допустить возникновения фрагментации в будущем, необходимо правильно настроить параметры областей хранения.

Как определить, фрагментирована база данных или нет? Для определения уровня фрагментации таблиц базы данных используется утилита PROUTIL с параметром DBANALYS:

```
proutil db-name -C dbanalys [area area-name]
```

Эта команда производит анализ состояния базы данных, благодаря которому, помимо состояния таблиц, отображается информация о состоянии индексов. По умолчанию анализируются все области хранения базы данных, можно так же произвести анализ по конкретной области хранения.

Если комплексный анализ не требуется, то вместо параметра DBANALYS можно воспользоваться параметром TABANALYS, который выведет информацию только по таблицам, параметр IDXANALYS выводит информацию только по индексам.

Следующий экран представляет фрагмент анализа таблиц базы данных, выполненного этой командой:

```
                                    -Record Size (B)- ---Fragments--- Scatter
Table                 Records    Size   Min   Max  Mean      Count Factor  Factor
PUB.af-f-dea               53   21.9K   310   673   422         53    1.0     4.6
PUB.af-f-fun                8    2.8K   218   419   358          8    1.0     1.0
PUB.af-f-kot              194   16.6K    80   226    87        194    1.0     4.5
PUB.af-mbtar               60    4.7K    44   451    79         60    1.0     4.8
PUB.af-srost           408273   13.8M    33    48    35     408273    1.0     1.0
PUB.af-tar              16607  483.2K    29    32    29      16607    1.0     2.7
PUB.af-tarsr           252538    7.1M    27    34    29     252538    1.0     1.5
PUB.bon-Acc2            14169  969.6K    26    95    70      14169    1.0     2.7
PUB.glowny             781651  257.4M   199  3051   345     787250    1.0     6.2
```

Существует два ключевых показателя, на которые стоит опираться при принятии решения о перезагрузке данных - это Index factor и Scatter factor. Показатель Scatter factor – значение, отображающее расположение на диске записей таблицы по отношению друг к другу. Значение 1.0 указывает на то, что записи находятся максимально близко друг к другу. Это означает, что множество записей, читаемых из одной таблицы, вероятно, будут обнаружены в буфере. Это соответствует общему утверждению о том, что наибольший показатель фрагментации - это более низкий буферный коэффициент совпадения, следовательно, более высокая дисковая активность и низкая производительность.

Показатель фрагментации является хорошим указателем качества фрагментации записей тогда, когда его значение равно единице. Например, значение 2.0 является показателем того, что таблица не в порядке, и по возможности ее стоит дефрагментировать.

На уровень фрагментации базы данных оказывают немалое влияние и такие параметры области хранения, как тип используемой области, значения RPB, количество таблиц в области. Устранение фрагментации это хороший повод для изменения этих параметров.

Существует еще пара параметров, которые способны влиять на размещение данных в таблицах, а точнее в блоках базы данных. Это параметры Create Limit и Toss Limit. Обычно их очень редко изменяют, поэтому здесь о них будет рассказано поверхностно. За более подробной информацией о них обращайтесь на специализированные форумы, такие как PEG, либо к документации по OpenEdge.

Записи размещены в блоках, а блоки согласно определенным алгоритмам размещаются так, чтобы максимально использовать дисковое пространство и минимизировать фрагментацию. При размещении новой записи движок базы сначала ищет свободное место в блоках RM-цепочки. Цепочка RM - это набор частично заполненных блоков. Если такой блок не будет найден, то запись будет вставлена в блок из цепочки свободных блоков. Свободная цепочка (free chain) - это цепочка, состоящая из empty-блоков. Если блок в начале RM-цепочки содержит достаточное количество свободного пространства для сохранения записи, то использован будет именно он. Если размер записи больше, чем свободное место в блоке, то запись разбивается на два или более фрагментов. При сохранении записи в блоке с целью дальнейшего возможного увеличения этой записи обязательно резервируется часть пространства блока. На размер этого пространства в блоке и влияют параметры Create Limit и Toss Limit.

Create Limit –минимальное количество свободного пространства в байтах, оставшееся в блоке после добавления записи. Его значение должно быть больше 32 байт и меньше размера блока, минус 128 байт. Для размера блока 1K предел по умолчанию равен 75 байт. Для всех остальных размеров по умолчанию 150 байт.

Toss Limit – минимальное количество свободного пространства в байтах, которое должно оставаться в блоке, чтобы не исключать его из RM-цепочки. Как только пространство в блоке становится меньше допустимого, блок удаляется из RM-цепочки, а оставшееся свободное пространство будет использоваться для возможного расширения записей, уже находящихся в блоке. Для размера блока 1K предел равен 150 байт. Для всех остальных 300 байт.

Изменить Create Limits и Toss Limits можно с помощью утилиты PROUTIL. Для областей хранения SAT-I их можно изменять только на уровне всей области, а для областей SAT-II как на уровне области, так и на уровне конкретных объектов. Синтаксис команды:

```
proutil db-name -C <спецификатор> <тип объекта> <новое значение>
```

В таблице приведена расшифровка применяемых спецификаторов и типов объектов.

Параметр	Тип объекта	Спецификатор
Create Limit	Area	SETAREACREATELIMIT
	BLOB	SETBLOBCREATELIMIT
	Table	SETTABLECREATELIMIT
Toss Limit	Area	SETAREATOSSLIMIT
	BLOB	SETBLOBTOSSLIMIT
	Table	SETTABLETOSSLIMIT

Узнать текущее значение этих параметров следует командой PROUTIL DISPTOSSCREATELIMITS. Параметры Create/Toss Limits желательно изменять только, если наблюдается высокая фрагментация данных или неэффективное использование пространства в блоках. Вот некоторые примеры:

Ситуация	Возможное решение
Фрагментация возникает при изменении существующих записей. Ожидался один фрагмент, а созданы два.	Увеличьте Create Limit.
Фрагментация возникает при изменении существующих записей или имеется множество RM-блоков с недостаточным пространством для создания новой записи.	Увеличьте Toss Limit.
Существует средняя фрагментация, но пространство блоков базы используется неэффективно, а существующие записи не предполагают изменение своего размера.	Уменьшите Create Limit и Toss Limit.

Увеличение параметров Create/Toss Limits позволяет записи больше вырасти в пределах блока прежде, чем будет использован другой блок, т.к. оба параметра определяют резерв для роста существующих записей. Если проверка Create Limit покажет недостаточное количество пространства, то новая запись будет размещена в другом блоке, но останется вероятность того, что в этот блок может быть помещена другая запись, меньшая по размеру, поэтому по результатам проверки Create Limit блок не удаляется из RM-цепочки. Если же при расширении записи проверка Toss Limit покажет, что предел достигнут, то блок будет удален из RM-цепочки и никакие новые записи не могут быть записаны в него.

Уменьшение параметров Create/Toss Limits решает проблемы неэффективного использования пространства в блоке, например в том случае, когда блоки имеют большое количество свободного пространства и при этом рост размеров существующих записей не предполагается. В этом случае новые записи в блок добавлены не будут, поскольку Create/Toss Limits этого сделать не позволит, следовательно, пределы можно будет уменьшить. Уменьшение с помощью Create Limit увеличивает вероятность того, что новая запись может быть добавлена в блок, а уменьшение Toss Limit позволяет блоку дольше оставаться в RM-цепочке.

Внимание! Изменение параметров Create/Toss Limit является тонкой настройкой базы данных и требует проведения тщательного тестирования изменений перед их внедрением на промышленной базе данных.

Dump & Load

Процесс перезагрузки данных (D&L) необходим в следующих случаях:

- создание новой версии базы данных;
- использование одной и той же таблицы в нескольких базах;
- с целью оптимального использования дискового пространства;
- конвертация базы данных в новую версию OpenEdge;

- конвертация базы данных для другой операционной системы;
- изменение описания данных для модернизации схемы базы данных.

Существует три условия, которые необходимо учесть перед выполнением D&L.

- Схема базы данных, в которую выполняется загрузка, должна быть полностью идентична схеме базы данных, из которой выполняется выгрузка. В противном случае необходимо написать свою собственную процедуру загрузки с использованием ABL-операторов INPUT FROM, CREATE и IMPORT.
- Если в таблицах базы данных используются поля типа ROWID или RECID, значение ROWID будет выгружено в виде символа «?». Поэтому для обеспечения выгрузки и загрузки таких таблиц необходимо написать собственную процедуру.
- Если у вас нет привилегий Can-Create и Can-Write, вы можете только загружать описания таблиц в базу данных, загрузить в них данные вы не можете.

Перед выгрузкой данных сначала необходимо выгрузить описание схемы базы данных, а затем содержимое таблиц. Выполнить эти процедуры можно посредством Data Dictionary утилиты PROUTIL или с помощью собственных ABL-программ. Ранее уже рассказывалось о том, как выгрузить описание схемы базы данных (см. лекцию «Изменение схемы базы данных»).

В OpenEdge существует два стандартных метода выгрузки содержимого базы данных - это двоичная выгрузка с помощью утилиты PROUTIL и текстовая выгрузка с помощью инструментария Data Dictionary.

Выгрузка содержимого таблиц с помощью PROUTIL

Утилита PROUTIL с параметром DUMP позволяет выполнять двоичную выгрузку данных из базы, находящейся как в online, так и в offline. Выгрузка в online может быть многопоточной, но многопоточность доступна только при наличии лицензии Enterprise. По умолчанию двоичная выгрузка выполняется с использованием первичного индекса таблицы, в определенных случаях может понадобиться выполнить выгрузку с использованием другого индекса, утилита PROUTIL предоставляет такую возможность.

Синтаксис команды для выполнения двоичной выгрузки (в offline и в online):

```
proutil db-name -C dump [owner-name.]table-name directory
     [-index num][-thread n ] [-threadnum nthreads]
     [-dumplist dumpfile]
```

Параметр	Назначение
db-name	База данных, из которой будет выполняться выгрузка.
owner-name	Владелец выгружаемой таблицы.
table-name	Имя выгружаемой таблицы.
directory	Имя каталога, в который будут выгружены данные.
-index *num*	Индекс, используемый при выгрузке. Индекс указывается его числовым номером. Для определения этого номера можно воспользоваться командой PROUTIL IDXANALYS.
-thread *n*	-thread 0 - индикатор однопоточной выгрузки; -thread 1 - индикатор многопоточной выгрузки.
-threadnum *nthreads*	Максимальное количество создаваемых потоков. Если этот параметр не используется, то количество потоков устанавливается автоматически, равным количеству CPU. Фактическое количество созданных потоков может быть

	меньше количества CPU, т.к. команда PROUTIL DUMP определяет оптимальное количество потоков на основании сложности используемого для выгрузки индекса.
-dumplist *dumpfile*	Список сгенерированных файлов выгрузки.

Во время двоичной выгрузки данные из таблиц записываются в файлы. Имена этих файлов определяются в соответствии с именами выгружаемых таблиц. Если владельцем выгружаемых таблиц является PUB (стандартный владелец объектов базы данных), то имя файла формируется из имени таблицы, имеет расширение .bd, например, `customer.bd`. Если владелец не PUB, то имя файла будет формироваться из имени владельца и имени таблицы, например, `owner_customer.bd`.

В системах с ограничением размеров файлов в 2Гб однопоточный PROUTIL DUMP создаст несколько файлов для таблицы, размер которой превышает 2Гб. Например, если выгружаются данные из таблицы с именем customer, размер которой 6,4Гб, то PROUTIL DUMP создаст следующие двоичные файлы: `customer.bd`, `customer.bd2`, `customer.bd3` и `customer.bd4`. Размер первых трёх файлов по 2Гб, а последний 0,4Гб. Команда PROUITL DUMP в каждый двоичный файл добавляет определенный заголовок, в результате общий размер всех двоичных файлов будет немного больше, чем размер выгружаемых данных.

В системах, в которых отсутствует ограничение размера файла, однопоточный PROUTIL DUMP создаст только один двоичный файл, независимо от размера таблицы.

Многопоточный PROUTIL DUMP создает отдельный файл для каждого потока. Первый поток создаст файл `tablename.bd`; второй, `tablename.bd2`; все последующие потоки будут создавать файлы, увеличивая их номера, т.е. `tablename.bd`*n*. Для генерации файла, содержащего список имен файлов, созданных потоками, используйте параметр <-dumpfile>. Созданный файл должен использоваться при загрузке с помощью PROUTIL LOAD.

Команда PROUTIL DUMP позволяет выполнять выборочную выгрузку из таблицы, для этого используется следующий синтаксис:

```
proutil db-name -C dumpspecified [owner-name.]table-name.field-
name
            operator field-value
            directory
```

Параметр	Назначение
db-name	База данных, из которой выполняется выгрузка.
owner-name	Владелец выгружаемой таблицы.
table-name	Имя выгружаемой таблицы.
field-name	Наименование поля, по значению которого выполняется выборка.
operator	Оператор сравнения для определения условия выборки: EQ – равно; GT – больше; GE – больше или равно; LT – меньше; LE – меньше или равно.
field-value	Значение, по которому происходит выборка.
directory	Имя каталога, в который выгружаются данные.

Примеры частичной выгрузки данных из таблицы.

- Выгрузку всех данных с датой больше 02/03/02 из базы Sports2000:

```
proutil sports2000 -C dumpspecified order.order_date GT ’02-03-2002’
```

- Выгрузка товаров, цены которых меньше $25.90:

```
proutil sports2000 -C dumpspecified item.price LT 25.9
```

Загрузка данных в двоичном формате с помощью PROUTIL

Для двоичной загрузки используется следующая команда:

```
proutil db-name -C load [filename | -dumplist dumpfile ]
```

Параметр	Назначение
db-name	База данных, в которую выполняется загрузка.
filename	Имя одного загружаемого файла.
-dumplist *dumpfile*	Имя файла со списком имен двоичных файлов выгрузки.

По умолчанию в процессе двоичной загрузки не выполняется построение индексов, это сделано для ускорения процесса. При желании с помощью параметра <build indexes> можно указать команде PROUTIL LOAD выполнять построение индексов в момент загрузки. **Внимание!** Нельзя использовать параметр <build indexes>, если во время выгрузки было создано несколько двоичных файлов для одной таблицы. В этом случае для построения индексов необходимо использовать команду PROUTIL IDXBUILD.

Выгрузка через Data Dictionary

Во время выгрузки содержимого базы данных с помощью Data Dictionary создается текстовый файл для каждой таблицы, который впоследствии используется для загрузки. Этот файл имеет расширение .d.

Для выгрузки с помощью Data Dictionary выполните следующее.

- Выберите в Data Dictionary пункт меню **Admin → Dump Data and Definitions → Table Contents (.d file)**. Откроется список всех таблиц в алфавитном порядке.
- Пометьте таблицы для выгрузки. Используя кнопки «Select Some» или «Deselect Some» можно выбирать таблицы по маске. Если выгружается одна таблица, то по умолчанию будет предложено имя, состоящее из имени таблицы и расширения .d. Если необходимо определить иное имя, то введите его в поле Output file. Если выгружается множество таблиц, то будет запрошено имя каталога, в который будет выполнена выгрузка. Если каталог не будет определен, то выгрузка будет выполнена в текущий каталог. Во время массовой выгрузки создаваемым файлам так же будут присваиваться имена, соответствующие выгружаемым таблицам.
- Если база данных содержит LOB-поля, то установите флаг Include LOB в значение Yes.
- Если Include LOB установлен Yes, то укажите в поле LOB Directory каталог для размещения файла для LOB-поля.

- Если необходимо использовать конкретную таблицу кодировки, отличную от таблицы кодировки базы данных, впишите её название в соответствующем поле и выберите OK, после это начнется выгрузка, на экране будут отображаться имена выгружаемых таблиц. По завершению будет выведена информация о статусе выгрузки.
- Выберите OK для возврата в основное меню.

Загрузка содержимого таблиц через Data Dictionary

Для загрузки содержимого таблиц в базу данных выполните следующее.

- Выберите в Data Dictionary пункт меню **Admin → Load Data and Definitions → Table Contents (.d files)**. Откроется список таблиц в алфавитном порядке для текущей базы данных.
- Выберите таблицу, которую необходимо загрузить. Будет запрошен файл с содержимым таблицы.
- Определите имя каталога и имя файла или оставьте значение по умолчанию.

  Если загружается содержимое всей базы данных, введите каталог, в котором расположены файлы для загрузки. Если каталог указан не будет, то по умолчанию будет использоваться текущий каталог. Каждая таблица загружается из соответствующего файла с именем `table-dumpname.d`.

  Если загружается одна таблица, то по умолчанию будет предложено имя файла, соответствующее имени таблицы.

- Укажите Include LOB, если загружаются LOB-поля.
- Если загружаются LOB-поля, в поле Lob Directory укажите каталог, в котором расположено их содержимое. Если он указан не будет, то будет использован текущий каталог.
- Определите допустимый показатель ошибок.

  Во время загрузки может случиться ситуация, когда данные не смогут быть загружены. Например, загружаемая запись может содержать значение в поле более длинное, чем определенный для него формат. Загрузка такой записи будет невозможна. Если указать показатель допустимого количества ошибок 10%, то из 100 записей в случае возникновения ошибок будет загружено 90, при превышении этого значения процесс прервется.

- Установите Output Errors to Screen, если необходимо видеть ошибки на экране, сразу при их возникновении.
- После завершения загрузки выберите OK для возврата в основное меню.

Выгрузка и загрузка информации о пользователях

Для выгрузки информации о пользователях базы данных выполните следующее.

- Выберите в Data Dictionary пункт меню **Admin → Dump Data and Definitions → User Table Contents**. Будет запрошено имя файла для выгрузки, по умолчанию используется имя `_user.d`.
- Нажмите клавишу <F1> или кнопку OK для начала выгрузки.
- По завершению выгрузки выводится сообщение «Dump of users completed.».

Для загрузки информации о пользователях базы данных выполните следующее.

- Выберите в Data Dictionary пункт меню **Admin → Load Data and Definitions → User Table Contents**. Будет запрошено имя файла, содержащего загружаемую информацию, по умолчанию это `_user.d`.
- Нажмите клавишу <F1> или кнопку OK для начала загрузки.

Выгрузка и загрузка значений секвенций

Для выгрузки текущих значений секвенций выполните следующее.

- Выберите в Data Dictionary пункт меню **Admin → Dump Data and Definitions → Sequence Values**. Будет запрошено имя файла для выгрузки значений. По умолчанию используется `_seqvals.d`.
- Нажмите клавишу <F1> или кнопку OK для начала выгрузки.
- По завершению выгрузки будет выдано сообщение «Dump of sequence values completed».

Для загрузки текущих значений секвенций выполните следующее.

- Выберите в Data Dictionary пункт меню **Admin → Load Data and Definitions → Sequence Current Values.** Будет запрошено имя файла, содержащего загружаемую информацию, по умолчанию это `_seqvals.d`.
- Нажмите клавишу <F1> или кнопку OK для начала загрузки.

Bulk loading

Утилита Bulk Loader выполняет загрузку текстовых данных намного быстрее обычной утилиты загрузки, поддерживаемой инструментальными средствами Data Dictionary.

Перед началом работы утилиты Bulk Loader необходимо создать файл описания. Для этого необходимо воспользоваться Data Dictionary. По умолчанию имя созданного файла `table-dumpname.fd`. Если выбрать несколько таблиц, то имя файла будет состоять из имени базы данных с расширением .fd. Вся информация, необходимая утилите, будет находиться в этом файле.

Для создания файл описания:

- Выберите в Data Dictionary пункт меню **Admin → Create Bulk Loader Description File.** Откроется список таблиц в алфавитном порядке.

- Выберите таблицы, для которых необходимо создать файл описания. Используйте клавиши <F5> и <F6>.
- Нажмите клавишу <F1> для продолжения. После этого будет запрошено имя файла, по умолчанию предлагается имя `table-dumpname.fd`.
- Для продолжения формирования файла описания нажмите клавишу <F1> или кнопку OK. После его создания выводится сообщение «Making of bulk load description file completed».
- Выберите OK для возврата в основное меню.

Пример файла описания:

```
# Database:  sports
# Date/Time: 2010/08/12-15:40:06
Customer customer.d customer.e
  Cust-Num
  Country
  Name
  Address
  Address2
  City
  State
  Postal-Code
  Contact
  Phone
  Sales-Rep
  Credit-Limit
  Balance
  Terms
  Discount
  Comments
# end-of-file
```

В этом примере `customer` - имя таблицы базы данных, `customer.d` - имя файла даных, `customer.e` – имя файла ошибок, возникших при работе Bulk Loader, остальное - поля таблицы customer.

Внимание! Если в таблице есть поля с типом данных RECID/ROWID, то информация о них не будет выгружена в файл описания, поэтому необходимо добавить их вручную. Помните об этом, когда будете загружать данные с помощью утилиты Bulk Loader. В противном случае значения полей будут смещены и загрузка такой таблицы будет невозможна.

Для загрузки содержимого таблицы используется утилита PROUTIL с параметром BULKLOAD. Используя файлы описания (.fd) и данных (.d), Bulk Loader осуществляет загрузку большого количества данных быстрее, чем стандартные инструментальные средства Data Dictionary.

Во время загрузки утилитой Bulk Loader игнорируются триггеры на создание записей и деактивируются все индексы загружаемых таблиц.

Для загрузки содержимого таблицы:

- создайте файл описания (.fd) с помощью Data Dictionary;
- убедитесь, что описания таблиц соответствуют базе данных;

- запустите утилиту Bulk Loader:

```
proutil db-name [-yy n] -C bulkload fd-file [-B n]
```

Параметр	Назначение
db-name	База данных, в которую будет выполняться загрузка.
-yy n	Начало 100-летнего периода, в котором любое значение DATE определено двумя цифрами.
fd-file	Файл описания.
-B *n*	Количество блоков в буферном пуле, используемом при загрузке.

В относительно небольших системах при использовании Bulk Loader можно столкнуться с нехваткой памяти. Решением этой проблемы является разбиение файла описания на несколько более мелких файлов.

Утилита Bulk Loader перед началом работы определяет, какой файл с содержимым необходимо использовать для конкретной таблицы. При этом порядок полей в файле описания (.fd) обязательно должен соответствовать порядку полей в файле данных (.d). В противном случае есть вероятность того, что в таблице окажутся неверные данные, что намного хуже, если бы таблица не была загружена совсем.

Поскольку утилита Bulk Loader деактивирует индексы, то по завершению её работы необходимо выполнить переиндексацию.

Внимание! Утилита Bulk Loader игнорирует дублирующиеся записи во время загрузки, т.к. уникальные индексы в это время деактивированы. Не допускайте двойной загрузки таблиц. Но если это всё-таки произошло, то перед началом индексации необходимо вручную исключить дублирование во избежание ошибок построения уникальных индексов.

Глава 18 Работа с индексами базы данных

Равно как и блоки с данными, индексные блоки со временем тоже становятся фрагментированными. Таким образом, работа приложения, интенсивно использующего поиск данных, будет ухудшена за счет того, что данные в индексных блоках будут разрознены. Производительность такого приложения будет выше, если в единицу времени будет считываться больше индексов из меньшего количества блоков. Иначе обстоят дела с приложениями, которые интенсивно изменяют имеющиеся данные. Здесь более высокая плотность индексов будет приводить к постоянному добавлению новых индексных блоков, а значит, время на выполнение операции создания индекса будет увеличено. Если в индексных блоках будет оставаться определенное свободное пространство, то решением этой проблемы может стать поиск баланса между степенью сжатия индексов и свободным их размещением в блоках. Обычно допустимой степенью сжатия считается значение в пределах 80 - 90%%.

Анализ плотности размещения индексов

Для анализа плотности размещения индексов используется команда PROUTIL IDXANALYS:

```
proutil db-name -C idxanalys [area area-name]
```

Команда позволяет выводить отчет как для всех областей хранения данных, содержащих индексы, так и для конкретной области. Это удобно, если база данных имеет большие размеры, формирование отчета по конкретной индексной области позволит значительно сократить время работы команды.

Что представляет собой итоговый отчет? В итоговый отчет входит следующая информация по каждому индексу:

- номер индекса в пределах базы данных;
- количество полей таблицы, по которым построен индекс;
- количество уровней в индексе;
- количество блоков, используемых индексом;
- размер индекса;
- степень эффективности использования дискового пространства в блоках;
- коэффициент, указывающий на степень фрагментации индекса.

Пример отчета для области Cust_Index из базы данных Sports:

```
INDEX BLOCK SUMMARY FOR AREA "Cust_Index" : 10
------------------------------------------------------
Table               Index  Fields Levels  Blocks    Size   %Util  Factor
PUB.Customer
  Comments             13       1      1       1    1.1K    26.9     1.0
  CountryPost          14       2      2       4    9.0K    56.5     1.9
  CustNum              12       1      2       4    9.9K    62.2     1.8
  Name                 15       1      1       1  248.0B     6.1     1.0
  SalesRep             16       1      1       1    1.3K    33.4     1.0

            ------------------------------------------------------------
Totals:                                      11   21.5K    49.2     1.7
```

Одним из важных показателей этого отчета является столбец %Util - степень эффективности использования дискового пространства. На него необходимо обратить внимание в первую очередь. Если индекс состоит из нескольких сотен блоков (поле Blocks) и %Util имеет значение в пределах 80 - 90%%, то считается, что с индексом всё в порядке. Если же значение меньше, то вероятно, что индекс нуждается в уплотнении. Выполнить уплотнение индекса можно двумя способами:

1. с помощью PROUTIL IDXCOMPACT;
2. с помощью PROUTIL IDXBUILD.

Второй показатель, значение которого необходимо отслеживать, представлен в столбце Factor. Фактически, это коэффициент, указывающий на степень фрагментации индекса, который сигнализирует о необходимости его перестройки. В таблице приведены возможные значения этого поля и действия, которые необходимо выполнить при их достижении:

Factor	Означает что…
1.0 - 1.5	Индекс в хорошем состоянии.
1.5 - 2.0	Индекс нуждается в уплотнении.
2.0 и >	Необходима переиндексация.

Принимая решение о перестройке индекса, помните что, это не всегда необходимо. Например, если индекс очень активен, т.е. часто создаются и удаляются ключи индекса, то показатели индексации (%Util и Factor) очень скоро вернутся к прежнему значению, поэтому здесь переиндексация смысла не имеет. Другое дело, если индекс статичен, например, индекс архивной таблицы, здесь переиндексация может существенно повлиять на производительность в лучшую сторону.

Уплотнение индексов

Уплотненный индекс занимает меньше пространства в базе данных и использует меньше памяти, т.к. все индексные блоки плотно заполнены. Количество обращений к базе данных будет меньше, т.к. после уплотнения индекса количество индексных уровней станет меньше. Уплотнение уменьшает количество блоков в B-дереве и количество уровней в нем, что увеличивает производительность за счет меньшего количества операций чтения.

Стоит отметить, что обычная перестройка индекса не принесет значительного эффекта, поскольку в процессе перестройки используются свободные блоки базы данных. В результате, используемые индексные блоки никогда не будут заполнены до конца. Поэтому для уплотнения индекса лучше использовать команду PROUTIL IDXCOMPACT:

```
proutil db-name –C idxcompact table-name.index-name [<процент
                                уплотнения>]
```

Команда выполняет уплотнение индекса в несколько этапов.

1. Если уплотняемый индекс уникальный, то сканируется цепочка удаленных индексных ключей и индексные блоки, которые их содержат, очищаются. Этот этап выполняется только, если индекс уникальный.
2. Уплотняются «нелистовые» уровни в B-дереве, начиная от корня и двигаясь к листовым уровням,

3. Уплотняются листовые уровни B-дерева.

Поскольку уплотнение индексов может выполняться в online, то все пользовательские процессы в это время могут одновременно без ограничений использовать индекс для операций чтения и записи. При уплотнении индексов в один момент времени используется только три индексных блока на короткое время, что обеспечивает параллельную работу процессов с уплотняемым индексом.

Результат уплотнения проверяется запуском до и после уплотнения некоего сложного отчета, использующего уплотняемый индекс, при этом необходимо ориентироваться на время формирования отчета.

Перестройка индексов (переиндексация)

Для полной или частичной перестройки индексов в OpenEdge используется утилита PROUTIL с параметром IDXBUILD. На момент работы команды база данных должна быть остановлена.

Перед началом переиндексации обязательно сформируйте полную резервную копию базы данных, т.к. если во время работы IDXBUILD возникнет сбой, то восстановить базу можно будет только из резервной копии.

Синтаксис команды:

```
proutil db-name -C idxbuild
     [ all |
        table [owner-name.]table-name |
        area area-name | schema schema-owner |
        activeindexes | inactiveindexes ]
     [ -T dir-name ]|[ -SS sort-file-directory-
     specification ]
     [ -TB blocksize ][ -TM n ] [ -B n ] [ -SG n ]
     [-threads n ] [-threadnum n ] [-pfactor n]
```

Если команда будет запущена без параметров, на экране откроется следующее меню:

```
Index Rebuild Utility
=====================

Select one of the following:
All           (a/A) - Rebuild all the indexes
Some          (s/S) - Rebuild only some of the indexes
By Area       (r/R) - Rebuild indexes in selected areas
By Schema     (c/C) - Rebuild indexes  by schema owners
By Table      (t/T) - Rebuild indexes in selected tables
By Activation (v/V) - Rebuild selected active or inactive indexes

Quit     (q/Q) - Quit, do not rebuild
```

Меню	Назначение
All	Полная переиндексация.
Some	Переиндексация конкретного индекса.
By Area	Переиндексация всех индексов для одной или нескольких областей.
By Table	Переиндексация индексов конкретных таблиц.
By Activation	Переиндексация активных или неактивных индексов.

После того, как выбор будет сделан, утилита запросит о наличии достаточного свободного дискового пространства для индексной сортировки. Для определения необходимого дискового пространства сортировки руководствуйтесь следующим.

- При полной переиндексации для сортировки потребуется не менее 75% свободного дискового пространства от общего размера базы данных.
- Для отдельных индексов рассчитать необходимое пространство сортировки можно по формуле:

(size of one index entry) * (number of records in file) * 3

Переиндексация выполняется в три этапа.

1. Индексные блоки очищаются и переносятся в цепочку свободных блоков.
2. Строятся все индексы для каждой записи. При необходимости сортировки все ключи индексов записываются в файл сортировки. В противном случае происходит прямая их запись в соответствующий индекс.
3. В файле сортировки информация об индексах разбивается на группы и вводится в индекс по порядку, формируя уплотненный индекс. Последний этап происходит только при необходимости сортировки.

Если установлена Enterprise-лицензия, то по умолчанию выполняется многопоточная переиндексация. Максимальное количество потоков можно задать параметром <-threadnum *n*>. Если этот параметр не определен, то максимальное количество потоков будет равно количеству CPU. Фактически, количество созданных потоков не будет превышать количество индексных групп в области, если их меньше, чем максимально возможное количество потоков. Во время многопоточной работы отдельные потоки назначаются каждой индексной группе на втором этапе. Как только основной процесс создаст все необходимые потоки на втором этапе, он приступит к формированию индексного дерева для третьего этапа. Допускается, что второй и третьи этапы могут работать параллельно. Если в области хранения находится только один индекс, то при переиндексации потоки использоваться не будут.

Если не требуется выполнения переиндексации в многопоточном режиме, то установите значение параметра <-threads>, равным нулю.

Если работа переиндексации по каким-либо причинам была прервана, то список выбранных для построения индексов сохраняется в файле `dbname.xb`. Этот файл используется утилитой при повторном запуске переиндексации. **Внимание!** Нельзя вручную изменять список индексов в этом файле.

Преодоление ограничений по размеру при сортировке

При запуске переиндексации и использовании параметра сортировки <-SS> можно столкнуться с проблемой нехватки свободного места на диске, что приведет к аварийному завершению процесса. Во избежание этого необходимо создать файл с описанием каталогов и их размеров, которые при переиндексации будут использоваться для файлов сортировки. Этот файл должен быть текстовым, иметь то же название, что и база данных и иметь расширение .srt (пример, `sports.srt`). Файл должен располагаться в том же каталоге, что и сама база данных. Требования к формату файла следующие.

- Описание каждого каталога должно располагаться на отдельной строке.
- Размер каталога - это максимальный размер файла сортировки, до которого он может вырасти. Если размер установлен равным нулю, то это говорит о том, что будет использовано все имеющееся пространство на диске.
- Описание каталога и его размер должны быть разделены минимум одним пробелом.
- Строка должна заканчиваться слэшем (/), что говорит о ее завершении.
- Для многопоточной переиндексации распределите каталоги на стольких различных дисковых устройствах, на скольких это возможно. Каждый поток будет использовать свой каталог. Поэтому, если все каталоги будут находиться на одном диске, то это увеличит дисковую активность, что значительно снизит производительность переиндексации.

Допустим, что при переиндексации базы sports для увеличения скорости сортировки мы используем 500Мб на диске /user/01, 400Мб на /user/02, и неограниченное пространство на /user/03, в этом случае srt-файл будет выглядеть так:

```
512000 /user/01/
409600 /user/02/
0      /user/03/
```

Команда переиндексации обращается к каталогам в том порядке, в котором они расположены в файле `dbname.srt`. Если указанное в файле пространство окажется больше доступного, то переиндексация будет прервана, следующий диск использован не будет. Например, если диск имеет 200Мб, а использовать указано 500Мб, то переиндексация будет прервана после использования 200Мб. Если установлен размер, равный нулю, и свободное пространство на диске закончится, то все каталоги, описанные после этого каталога, не будут использованы, то процесс переиндексации будет прерван. Перед началом переиндексации обязательно убедитесь в наличии необходимого свободного пространства.

Команда IDXBUILD создает файлы в каталоге, прежде чем начнется сам процесс переиндексации, поэтому на экране будет выведено одно из следующих сообщений:

```
Temporary sort file at pathname used up to nK of disk space
```

или

```
Temporary sort file at pathname will use the available disk space.
```

По завершению сортировки выводится следующее сообщение:

```
Temporary sort file at pathname used nK of disk space.
```

Если srt-файл не используется, то по умолчанию для сортировки будет использован либо текущий каталог, либо каталог, определенный параметром <-T>.

Улучшение производительности процесса переиндексации

Для ускорения процесса переиндексации необходимо откорректировать параметры команды IDXBUILD.

- Увеличьте параметр <-TB> до 31K. Это увеличивает скорость сортировки, но при этом используется больше памяти и дискового пространства.
- Увеличьте параметр <-TM> до 32.
- Используйте параметр <-SG>. Большое значение этого параметра требует большего количества памяти и файловых дескрипторов. Для определения необходимого объема памяти (в килобайтах) для каждой индексной группы прибавьте к значению параметра <-TM> единицу и умножьте полученную сумму на значение параметра <-TB>:

  (<-TM> + 1) * <-TB>.

- Установите значение параметра <-T> таким образом, чтобы он указывал на отдельный диск.

Движок базы данных использует следующий алгоритм для индексации каждой записи:

1. Читает поля ключа индекса и сохраняет в первый доступный блок в файле сортировки.
2. Размещает дополнительные блоки в файле сортировки для обеспечения поддержки всех ключей индекса.
3. Сортирует ключи в каждом блоке, а затем объединяет их в файле сортировки.

Большой размер блока (<-TB>) может значительно увеличить производительность, поскольку будет размещено меньшее количество блоков сортировки, а значит, потребуется меньшее количество операций по сортировке каждого блока. Для определения оптимальных значений запустите переиндексацию несколько раз с разными параметрами и выберите те из них, при использовании которых производительность была выше. Если вы наблюдаете значительную нехватку памяти во время переиндексации, то попробуйте установить меньший размер блока. Для начала установите <-TB> в значение 31, при наличии достаточно большого количества памяти и дискового пространства. Если индексация будет прервана ошибкой, то последовательно уменьшайте значение. **Примечание:** чем выше значение параметра <-TB>, тем больше памяти будет использовано. Значение <-TB> также влияет на размеры файлов сортировки.

Использование памяти зависит от числа одновременно выполняющихся сортировок. Одновременные сортировки - это эквивалент вложенных операторов FOR EACH. Определить количество используемой памяти можно по формуле:

M= <-TB> * (8 + <-TM>)

Построение уникальных индексов

При построении уникальных индексов команда IDXBUILD, обнаружив дублирующие ключи, выводит следующее сообщение:

```
Fix RECID recid, table-name already exists with field-name value.
```

В этом случае для исключения дублирования ключей, для обеспечения доступа ко всем данным необходимо изменить запись. Если существует другой индекс, то доступ к записи можно получить по нему:

```
FOR EACH table-name USE-INDEX index-without-duplicate-keys:
```

```
      UPDATE table-name.
END.
```

Активация одного индекса

Если при добавлении индекса в online было указано, что он должен оставаться неактивным, то его последующую активацию можно выполнить следующей командой:

```
proutil db-name -C idxactivate [owner-name.]table-
name.index-name
     [useindex table-name.index-name] [recs n] [refresh t]
```

Перед активацией индекса команда IDXACTIVATE проверяет отсутствие пользователей со штампом времени более ранним, чем штамп времени индекса. Если такие пользователи есть, то IDXACTIVATE не сможет приступить к работе, станет необходимо выбрать опцию ожидания или завершить работу утилиты. При ожидании необходимо дождаться, пока все пользователи с ранним штампом времени не отсоединятся от базы, таких пользователей можно так же самостоятельно отключить от базы утилитой PROSHUT.

Когда IDXACTIVATE строит индекс, он использует транзакции. По умолчанию в каждую транзакцию включается 100 записей. Это количество можно изменять параметром <recs *n*>. Количество записей в одной транзакции влияет на размер BI-файла.

По завершению работы IDXACTIVATE индекс станет активным и доступным для всех пользователей.

Глава 19 Средства мониторинга производительности базы данных

Методология анализа производительности

Всегда начинайте анализ производительности с самого медленного ресурса и постепенно продвигайтесь к самому быстрому. Таким образом, первыми для анализа должны быть диски, затем оперативная память, и, наконец, CPU. Прежде чем анализировать систему, необходимо убедиться, что приложение выполняется правильно, поскольку именно приложение оказывает самое большое влияние на производительность в целом. Определить это можно, отслеживая количество запросов к базе данных для каждого пользователя. Например, если большинство пользователей выполняет десятки тысяч запросов, но при этом есть пользователи, у которых количество запросов превышает миллионы, то имеет смысл обратиться к этим пользователям для разъяснения того, что именно они запускают в системе. После чего необходимо проанализировать указанные ими программы на предмет обнаружения причин неэффективной работы. Если эти пользователи выполняют те же действия, что и другие, то возможно, проблема заключается в том, как они используют приложение, например, если это отчет, то возможно, что задан слишком большой период сбора данных для него.

Стоит отметить, что оптимизация производительности - это итерационный процесс, при котором такие этапы как снятие статистики, анализ, выполнение работ по настройке могут повторяться неоднократно для достижения максимально возможной производительности. Однако стоит помнить, что ресурсы базы данных и системные ресурсы сильно взаимосвязаны, и иногда может случиться так, что после устранения одного узкого места может появиться другое.

Мониторинг производительности

Большинству OpenEdge-приложений требуется нескольких различных ресурсов, чтобы функционировать должным образом. В таблице приведено описание этих ресурсов.

Ресурс	Область использования
CPU	**Манипулирование данными и запуск программ.**
Диски и контроллеры	**Чтение и запись данных.**
Память	**Хранение данных для быстрого доступа.**
Механизмы операционной системы	**Распределение и использование системных ресурсов.**
Сеть	**Обмен данными между клиентом и сервером.**

Плохая работа ресурсов является наиболее вероятной причиной сбоя системы (аппаратный сбой является наиболее распространенным). При определении источника проблемы в первую очередь необходимо понять, что изменилось? Если вы выполняете мониторинг системы, то можете определить, появилась ли разница в работе (чтение, запись и т.п.), изменилось ли количество пользователей в системе, потребляет ли система больше ресурсов, чем обычно.

Мониторинг ресурсов является чрезвычайно важной задачей, т.к. если вы знаете, как используются ресурсы, то вы можете определить, когда они иссякнут. Мониторинг также позволяет планировать рост размеров базы и избежать потенциальных проблем с базой данных и системы в целом. Например, в большинстве систем емкость дисков и области

хранения базы данных являются наиболее динамичными в плане их использования, а значит, требуют отдельного внимания.

Узкие места в производительности возникают, когда ресурс работает не адекватно или перегружен, тем самым мешая работе других ресурсов. Ключом к решению вопроса производительности является определение этого узкого места. Как только станет понятно, какой ресурс является проблемным, вы сможете принять меры по устранению проблемы.

Управление производительностью - непрерывный процесс измерений и настройки используемых ресурсов. Поскольку системные ресурсы тесно взаимосвязаны между собой, можно легко решив одну проблемы, создать новую. Мониторинг ресурсов должен происходить постоянно, с целью своевременного изменения параметров.

Для эффективного управления производительностью необходимо хорошо знать свою систему, пользователей и приложения. Правильно настроенная система всегда способна выдержать предназначенную для неё работу. Работающие приложения не должны конкурировать между собой за системные ресурсы.

Инструменты мониторинга OpenEdge

Инструменты мониторинга производительности могут быть разделены на две категории. Первая, это встроенные инструменты OpenEdge. Вторая, это инструменты операционной системы. Обе эти категории тесно связаны между собой, и порой весьма трудно понять, что происходит с системой, не используя и те, и другие средства.

Promon & VST

Основным встроенным средством мониторинга базы данных OpenEdge является утилита Promon (OpenEdge Monitor), которая позволяет контролировать активность и производительность базы данных. У утилиты имеется расширенная возможность глубокого контроля производительности, активируемая командой R&D. Для запуска Promon используется команда:

```
promon db-name
```

Утилита Promon использует виртуальные системные таблицы OpenEdge. Виртуальные системные таблицы (VST[36]) поддерживаются как ABL, так и SQL-приложениями и предоставляют информацию о работе базы. Эти таблицы не имеют постоянных физических записей, поскольку записи генерируются менеджером базы данных во время её работы и хранятся в оперативной памяти. Описание виртуальных системных таблиц можно найти в стандартной документации «OpenEdge® Data Management Database Administration» в главе «Virtual System Tables», а также в книге Дэна Форемана (Dan Foreman) «Progress Virtual System Tables Guide».

Утилита Promon и VST являются стандартными средствами, которые доступны бесплатно. Но их минусом является то, что необходимо самостоятельно разрабатывать методики работы с ними, писать собственные программы и скрипты и т.п. К тому же эти средства не хранят статистических данных за прошлые периоды времени.

[36] VST – Virtual System Table

DBMON

Скрипт `dbmon.sh` разработан Юрием Михайловичем Потемкиным, ведущим экспертом в области настройки производительности баз данных OpenEdge. Скрипт предназначен для сбора информации, необходимой для оценки эффективности настройки СУБД OpenEdge для работы с конкретным приложением. Никаких изменений в систему скрипт не вносит. Скрипт собирает информацию, используя экраны утилиты Promon, сохраняя их в отдельный файл. Помимо этих данных также собирается информация по работе системы, предоставляемая системными утилитами `vmstat` и `iostat`. Собираемой скриптом информации обычно достаточно для понимания процессов, происходящих в базе данных и операционной системе.

Для запуска скрипта в качестве параметров командной строки следует указать список баз, производительность которых следует проанализировать. Синтаксис команды:

```
dbmon.sh db1 [db2 … dbn] [interval [count]] [-stop]
```

Параметр	Назначение
db1	Имя базы данных, по которой собирается статистика.
db2 … dbn	Список имен баз данных, разделенных пробелом. Скрипт может работать с несколькими базами данных одновременно.
interval	Интервал между формированием выборок. По умолчанию 300 сек.
count	Количество выборок. По умолчанию 100.
-stop	Принудительная остановка процессов мониторинга.

Информация по мониторингу операционной системы выводится в файлы

```
iostat-k-t.<yymmdd>_<hhmmss>.log и vmstat.<yymmdd>_<hhmmss>.log
```

Информация по каждой базе выводится отдельно в файлы

```
<db-name>.promon.<yymmdd>_<hhmmss>.log.
```

Все файлы сохраняются в текущем каталоге, из которого был запущен скрипт.

Работа скрипта завершается автоматически после формирования количества выборок, установленных параметром <count>, но его работу можно прервать в любое время, выполнив команду:

```
dbmon.sh -stop
```

Если запустить скрипт без параметров, то на экран будет выведена справочная информации по его использованию.

DBSTAT

Скрипт `dbstat` предназначен для сбора статистики по использованию (количеству чтений, созданий, модификаций и удалений) таблиц и индексов в одной или нескольких базах данных. Скрипт разработан Ю. М. Потемкиным.

Перед запуском в файле `dbstat.ini` необходимо установить интервал сканирования (в секундах) параметром INTERVAL и имя файла лога в параметре OUTPUT. Пример содержимого файла `dbstat.ini`:

```
[Startup]
INTERVAL=10
OUTPUT=/users/valeriy/dbstat.log
RECONNECT=yes
USERS=*
[Commands]
# SUMMARY
# QUIT
```

Запуск осуществляется скриптом `dbstat.sh`:

```
./dbstat.sh db-name1 [-db db-name2] [-db db-nameN]
```

Внимание! Перед сбором статистики серверы баз данных должны быть запущены с достаточными значениями параметров базы данных <-tablerangesize> и <-indexrangesize> (не меньше, чем реальное количество таблиц и индексов в базе данных), в противном случае статистика будет неполной. При недостаточном значении указанных параметров в файл `dbstat.err` выводится соответствующая информация.

Для останова работы скрипта необходимо в текстовом редакторе в файле `dbstat.ini` «раскомментировать» параметр QUIT (убрать символ # в первой позиции) и сохранить файл.

Начиная с версии 10.1B, помимо основного лога, возможно снятие статистики по отдельным пользователям. Если база была конвертирована из базы версии, более ранней чем 10.1B, то необходимо обновить виртуальные системные таблицы:

```
$DLC/bin/proutil db-name -C updatevst
```

После чего в секции [Startup] файла `dbstat.ini` необходимо описать параметр USERS:

```
USERS=<user_number_list | * >
```

Параметр	Назначение
user_number_list	Список номеров коннектов пользователей через запятую.
*	Все пользователи.

Например:

```
USERS=10,23,17
```

или

```
USERS=*
```

Кроме файла основного лога будут созданы файлы логов по пользователям вида:

<основной файл лога>.<имя пользователя>.<номер пользователя>

Например:

```
dbstat.log.user1.10
dbstat.log.user2.23
dbstat.log.admin.17
```

При снятии статистики по нескольким базам пользовательские номера коннектов и их имена определяются по первой базе из параметров запуска скрипта, т.е. сбор статистики производится по сессиям.

Типовые действия при снятии статистики по пользователю:

1. Пользователь вошел в систему.
2. Определить номер его коннекта к базе.
3. Указать номер его коннекта в параметре USERS.
4. Запустить скрипт `dbstat.sh`.
5. Пользователь выполняет действия над базой (запускает программу).
6. После этого проанализировать основной лог и лог пользователя.

OpenEdge Management

Помимо бесплатных средств мониторинга существуют и несколько платных решений. Первое средство разработано самой компанией Progress Software Corp. Это отдельный продукт, который называется OpenEdge Management. Помимо мониторинга производительности базы данных OpenEdge он позволяет осуществлять мониторинг ресурсов сервера. Имеет собственную базу данных, позволяющую хранить статистические данные. Основное его предназначение - управлять ресурсами OpenEdge. С помощью него можно запускать и останавливать базу данных, выполнять резервное копирование и восстановление, управлять продуктом OpenEdge Replication и многое другое. Описание OpenEdge Management выходит за рамки данного курса.

ProMonitor

Следующим средством мониторинга является продукт компании BravePoint под названием ProMonitor. Это средство мониторинга разработано. Дэнном Фореманом (Dan Foreman). Программа имеет свою собственную базу данных, и, как и Promon, основывает свою работу на VST. Помимо VST для анализа состояния базы данных ProMonitor умеет обрабатывать результат работы утилиты PROUTIL DBANALYS с сохранением полученных данных в базу. Приложение ProMonitor имеет большой набор различных отчетов, позволяющих сравнивать состояние системы между различными периодами времени. Она позволяет осуществлять мониторинг более одной базы данных одновременно. Несомненно, приятно иметь единую точку входа, если вы имеете несколько баз данных. Это достаточно удобный продукт, значительно облегчающий жизнь администратору. Его подробное описание так же выходит за рамки этого курса. Необходимую информацию о нём на русском языке всегда можно найти на страницах сайта www.openedge.ru.

Инструменты мониторинга операционной системы

В большинстве дистрибутивов Linux есть много средств мониторинга, которые измеряют различные характеристики для получения информации о работе системы. Далее будут

приведены некоторые из команд операционной системы, которые стоит помнить и использовать для анализа её производительности.

FREE

Команда `free` показывает общее количество свободной и используемой системой физической памяти и памяти свопинга, а также размеры буферов, используемых ядром. Ниже представлен результат её работы:

```
             total       used       free     shared    buffers     cached
Mem:       1017332     603872     413460          0       9808     495164
-/+ buffers/cache:      98900     918432

Swap:      1052248        116    1052132
```

Здесь строки:

- Mem - использование физической памяти;
- -/+ buffers/cache - объём физической памяти, выделенной в настоящее время для буферов системы;
- Swap - использование пространства подкачки.

Команда `free` по умолчанию выводит сведения об использовании памяти всего один раз. Выводить показатели использования памяти многократно можно с помощью параметра <-s>, указав в качестве аргумента количество секунд между выборками. В этом случае выборки будут формироваться бесконечно долго, поэтому для ограничения их количества воспользуйтесь дополнительной командой <-c>, которой в качестве аргумента передается количество необходимых выборок. Например, 5 выборок с интервалом в 2 секунды выведет на экран команда:

```
free -s 2 -c 5
```

Примечание: команда free не переписывает отображенные значения, а прокручивает экран во время вывода информации по очередной выборке, поэтому лучше направлять результат работы команды в файл. Этот файл можно просмотреть и проанализировать позже. Выводить информацию на экран без прокрутки следует командой `watch`. За дополнительной информацией по работе команды `watch` можно обратиться к страницам руководства по этой команде (`man watch`). Команду `watch` желательно запомнить, т.к. она может пригодиться в самых различных ситуациях.

TOP

Утилита `top` выводит на экран информацию в реальном времени – «понемногу обо всём». По умолчанию она показывает процессы, которые наиболее загружают процессор сервера, и обновляет список каждые пять секунд. В отличие от команды `free`, утилита `top` работает непрерывно. Информация, представляемая командой `top`, разделяется на две части. В верхней части показаны общие сведения о состоянии системы - время работы, средняя нагрузка, число процессов, состояние процессора, статистика использования памяти и пространства подкачки. В нижнем разделе - статистика в разрезе процессов. Во время работы `top` можно выбрать, какие данные будут отображаться. Например, по умолчанию `top` выводит статистику и по работающим, и по простаивающим процессам.

Чтобы оставить только работающие процессы, нажмите клавишу <i>, второе нажатие восстановит обычный режим.

Утилита top может пригодиться, например, для определения первой десятки Progress-процессов, больше всего загружающих процессор. Для этого необходимо выполнить команду:

```
top -b -n 1 | sed -e "1,6d" | grep progres | head -10
```

Результат работы этой команды:

```
21754 trofim    16   0  108m  24m  14m S  9.2  0.3   0:02.30 _progres
28063 amh       16   0  109m  25m  15m D  1.8  0.3   0:04.28 _progres
  422 katrin    16   0 66608  23m  12m S  0.0  0.3   1:15.68 _progres
```

Здесь мы видим, что пользователь trofim активнее всех использует CPU (значение 9,2). Теперь, зная PID Progress-процесса пользователя trofim, мы можем с помощью скрипта $DLC/bin/proGetStack сформировать стек ABL-программ, с которыми этот пользователь сейчас работает, что поможет выяснить, какая именно программа загружает систему.

VMSTAT

Команда vmstat – средство мониторинга, которое позволяет получить статистическую информацию о процессах, находящихся в очереди запуска и ожидания; об оперативной памяти; о пространстве подкачки; дисках; системных вызовах; прерываниях; работе процессора и пр. Данные по процессору показывают: сколько процессор работал в пользовательском режиме; сколько в системном режиме; сколько в режиме ожидания ввода-вывода; и сколько он простаивал. Эту команду наиболее удобно использовать для мониторинга использования процессора. Здесь приведен пример результата работы команды vmstat/

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu---
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs  us sy id wa
 1  0     88  52056  70344 1310208    0    0   223    66   39   42  7  2  89  2
 2  0     88  52056  70344 1310208    0    0     0     0 1034 37133 37 29 33  0
 1  0     88  52060  70344 1310216    0    0     0     0 1016 32910 42 30 29  0
```

С помощью этой команды можно не только наблюдать за загрузкой CPU, с помощью неё можно следить за использованием памяти, а также с её помощью можно определить информацию о производительности дисков. В рамках этого курса ограничимся небольшим описанием полей в этой команде. Более детальную информацию можно получить на страницах руководства по этой команде (man vmstat).

Что бы понять, служат ли ресурсы процессора причиной снижения производительности системы, необходимо изучить значения, указанные в четырех столбцах cpu и двух столбцах procs.

Краткое описание каждого блока информации, полученной командой vmstat.

Блок procs - среднее количество процессов, которые каждую секунду находились в различных очередях в течение интервала сбора информации.

- **r** - среднее количество процессов, которые каждую секунду находятся в очереди выполнения. Если это значение быстро растет, то проверьте правильность работы приложений. Некоторые системы могут нормально работать, даже если в очереди выполнения находится от 10 до 15 процессов, поскольку все зависит от их назначения и времени их выполнения.
- **b** - среднее количество процессов, которые каждую секунду находятся в очереди ожидания. В этой очереди находятся процессы, ожидающие освобождения ресурса или выполнения операции ввода-вывода. Кроме того, в эту очередь помещаются процессы, ожидающие загрузки одной из своих страниц в оперативную память. Обычно это значение близко к нулю. Однако, если число процессов в очереди выполнения увеличивается, то, как правило, длина очереди ожидания также увеличивается. Если в течение одной секунды одновременно было активировано несколько процессов, то длина очереди выполнения может быть довольно большой. При этом показатель использования процессора может быть довольно низким, если эти процессы были сразу же приостановлены. Если процессы в системе приостанавливаются из-за перегрузки памяти, то прирост процессов отражает именно значение в столбце b отчета vmstat, а не число процессов в очереди выполнения.

Блок cpu - распределение времени CPU в течение интервала времени. В столбцах cpu указывается следующее.

- **us** - доля времени CPU (в процентах), затраченная на работу в пользовательском режиме. Процессы могут выполняться в пользовательском или системном режиме (режиме ядра). При работе в пользовательском режиме процесс выполняется в рамках приложения. Ему не требуются ресурсы ядра для выполнения вычислений, управления памятью и настройки переменных.
- **sy** - доля времени процессора, затраченная на выполнение процессов в системном режиме. Это значение отражает ресурсы CPU, которые были затрачены на выполнение процессов ядра и других процессов, использующих ресурсы ядра. Для получения доступа к ресурсам ядра процесс отправляет системный вызов, в результате чего он переключается в системный режим. Например, при выполнении операции чтения или записи данных в файл ресурсы ядра применяются для открытия файла, смещения указателя в нужную позицию и последующего чтения или записи данных, если файл еще не загружен в оперативную память.
- **id** - доля времени (в процентах), когда процессор простаивал, не ожидая завершения локальной операции дискового ввода-вывода. Если ни один процесс не готов к работе (очередь выполнения пуста), то система запускает специальный процесс wait. В системе SMP[37] процесс wait может быть запущен на каждом процессоре. В отчете команды `ps` (запущенной с параметрами <-k> или <-g 0>) этот процесс указывается как kproc или wait. В однопроцессорной системе PID этого процесса обычно равен 516. В системах SMP процесс kproc будет указан для каждого процессора в отдельности. Если в отчете `ps` указано, что на этот процесс в целом затрачивается много времени, то это означает, что существуют большие промежутки времени, в течение которых ни один процесс не готов к работе и не ожидает запуска. Следовательно, система преимущественно простаивает, ожидая появления новых задач. Если нет ожидающих операций ввода-вывода на локальный диск, то все время, указанное как время ожидания, задает время простоя.

[37] SMP – от англ. Symmetric Multiprocessing, это архитектура многопроцессорных компьютеров, в которой два или более одинаковых процессора подключаются к общей памяти.

- **wa** - доля времени, в течение которого процессор простаивал, ожидая завершения операции ввода-вывода на локальный диск. Если во время выполнения процесса wait в системе есть хотя бы один процесс, ожидающий выполнения ввода-вывода, то это время рассматривается, как время ожидания завершения операции ввода-вывода. За исключением случаев, когда операции ввода-вывода выполняются асинхронно с процессом, процесс блокируется на время выполнения запросов на обмен данными с диском. После выполнения запроса на ввод-вывод процесс снова помещается в очередь выполнения. Чем быстрее выполняется обмен данными с диском, тем больше времени CPU будет тратиться на выполнение процессов. Если значение wa больше 25%, то можно сделать вывод, что дисковая подсистема плохо сбалансирована, либо в системе выполняется много дисковых операций ввода-вывода.

Блок system - здесь указывается информация об управлении процессами.

- **in** - число прерываний, поступавших от устройств каждую секунду в течение интервала сбора информации. Прерывания генерируются при возникновении таких событий, как завершение выполнения запроса диском или истечение очередных 10 миллисекунд, измеряемых таймером.
- **cs** - число переключений контекста, выполнявшихся каждую секунду в течение интервала сбора информации. Все время работы процессора разделяется на логические кванты времени по 10 миллисекунд. Любой процесс выполняется до тех пор, пока не истечет квант времени, пока он не будет замещен процессом с более высоким приоритетом, либо пока он добровольно не передаст управление другому процессу. Когда управление передается другому процессу, контекст, или рабочая среда, предыдущего процесса сохраняется, а вместо него загружается контекст текущего процесса. В операционной системе предусмотрена очень эффективная процедура переключения контекста, поэтому на переключение контекста тратится лишь незначительная часть ресурсов. Однако при значительном росте числа переключений контекста, например, когда значение cs становится намного больше скорости дискового ввода-вывода и скорости передачи пакетов по сети, необходимо дополнительно проанализировать ситуацию.

Блок memory - поля, отображающие объем используемой памяти.

- **swpd** – виртуальная память.
- **free** – свободная память.
- **buff** – память для буферов системы.
- **cache** – память для кэша страниц.

Блок swap - поля, отображающие объем памяти, используемой подкачкой.

- **si** - объём памяти, подкачанной с диска.
- **so** - объём памяти, выгруженной на диск.

Блок io - поля, отображающие состояние подсистемы ввода/вывода.

- **bi** - количество блоков, отправленных на блочное устройство.
- **bo** - количество блоков, полученных с блочного устройства.

С помощью `vmstat` опытный администратор может быстро определить, как используются ресурсы, и выявить проблемы производительности. Взглянуть внутрь этих проблем необходимо с помощью инструмента другого типа - инструмента, позволяющего собрать и проанализировать данные на более глубоком уровне.

SYSSTAT

Инструменты, описанные ранее, могут быть полезны для анализа производительности системы в течение короткого периода времени, их возможности ограничиваются созданием снимков использования системных ресурсов. Кроме этого, существуют аспекты производительности системы, наблюдать за которыми с помощью таких простых инструментов затруднительно. Необходим более совершенный инструмент, каковым и является Sysstat.

Sysstat - пакет для мониторинга производительности, содержащий следующие средства:

- iostat - выводит краткую статистику по использованию процессора, а также статистику ввода/вывода для одного или нескольких дисков;
- mpstat - выводит более подробную статистику по использованию процессора;

- sadc[38] - собирает информацию об использовании ресурсов системы и сохраняет их в файле.
- sar - обрабатывая файлы, созданные sadc, может создавать интерактивные отчёты или сохранять их в файле для более подробного анализа.

Рассмотрим каждый из этих инструментов мониторинга.

IOSTAT

Команда `iostat` позволяет получить представление о расходе ресурсов системы. Она может применяться не только для обнаружения неполадок дискового ввода-вывода, приводящих к снижению производительности, но с её помощью можно так же получить статистическую информацию об использовании процессора. Пример результата работы команды `iostat`:

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           7.00    0.00    2.29    1.72    0.00   88.99

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda              23.70       884.21       263.69  375487497  111978518
sda1              0.00         0.00         0.00       1924         14
sda2              4.27        36.45        36.81   15479226   15630600
sda3              0.00         0.00         0.00       1874        176
sda4              0.00         0.00         0.00         10          0
sda5              4.41       589.79        91.13  250459015   38700456
sda6             14.44       254.28       131.07  107981208   55659040
```

Самый распространенной проблемой производительности, которую можно идентифицировать с помощью этой команды, это дисбаланс дисков системы. Если вы видите, что на одном из дисков активно выполняются операции чтения/записи, и при этом другие диски откровенно простаивают, то имеет смысл перераспределить нагрузку между ними. Для этого, например, можно перенести часть экстентов базы данных на другие

[38] SADC – от англ. System Activity Data Collector

диски. Такими экстентами могут быть BI- или AI-экстенты или экстенты пользовательских областей хранения, которые наиболее активно используются. Активность использования областей хранения можно посмотреть с помощью утилиты Promon, воспользовавшись ее экраном R&D → 2 → 9 (I/O Operations by File).

MPSTAT

Команда `mpstat` выводит данные об активности каждого имеющегося в наличии процессора, процессор с номером 0 будет первым. Следующая команда выводит данные о среднем использовании ресурсов для каждого из процессоров:

```
mpstat -P ALL
```

Результат её работы:

```
Linux 2.6.18-164.2.1.el5.plusPAE (polygon)      01/21/2010

03:31:35 PM  CPU %user  %nice  %sys %iowait  %irq  %soft %steal  %idle   intr/s
03:31:35 PM  all  3.30   0.00  0.32    2.49  0.02   0.05   0.00  93.83  1287.51
03:31:35 PM    0  3.00   0.00  0.14    0.49  0.00   0.01   0.00  96.36  1000.38
03:31:35 PM    1  2.25   0.00  0.17    0.42  0.00   0.01   0.00  97.15     0.09
03:31:35 PM    2  2.86   0.01  0.56    8.03  0.00   0.04   0.00  88.51    41.88
03:31:35 PM    3  5.08   0.00  0.41    1.01  0.06   0.13   0.00  93.30   245.16
```

На первый взгляд команда `mpstat` выводит тот же отчёт об использовании процессора, что и `iostat`, за исключением дополнительных столбцов, показывающих число прерываний, обработанных процессором за секунду.

SADC

Утилита `sadc` собирает данные по использованию компьютера и сохраняет их в файле для дальнейшего анализа. По умолчанию данные сохраняются в файлах, в каталоге `/var/log/sa/`. Файлы имеют названия в формате `sa<dd>`, где `<dd>` — текущий день месяца, представленный двумя цифрами.

Утилита `sadc` запускается скриптом `sa1`. Этот скрипт периодически вызывается демоном `cron` благодаря файлу `sysstat`, расположенному в каталоге `/etc/cron.d/`. Скрипт `sa1` вызывает `sadc` для выполнения одного измерения в течение одной секунды. По умолчанию `cron` запускает `sa1` раз в 10 минут и накапливает собранные в этом интервале данные в текущем файле `/var/log/sa/sa<dd>`.

Ниже приведен пример файла `/etc/cron.d/sysstat`:

```
#======================================================================
#  Отчеты о работе системы создаются:
#  В рабочие дни с 8 утра до 5 вечера - каждые 20 минут;
#  В субботу и воскресенье - каждый час;
#  Каждый день с 6 вечера до 7 утра - каждый час.
#  Ежедневный отчет создается в 18:05.
#======================================================================
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3 &
0 * * * 0,6 /usr/lib/sa/sa1 &
0 18-7 * * 1-5 /usr/lib/sa/sa1 &
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 3600 -ubcwyaqvm &
#======================================================================
```

Такой способ сбора информации позволяет получить представление о том, насколько интенсивно система используется в течение определенного периода времени, и поможет определить время пиковой нагрузки.

SAR

Команда `sar` собирает статистическую информацию о работе системы. Она позволяет получить полезную информацию о производительности, однако ее выполнение создает значительную нагрузку на систему, что может привести к дальнейшему снижению производительности. Это особенно важно учитывать, если команда `sar` создает отчеты через короткие промежутки времени.

В системе предусмотрен ряд счетчиков для различных операций, выполняемых в системе. Команда `sar` создает отчеты на основе показаний этих счетчиков. Значения этих счетчиков обновляются автоматически, поэтому их обновление никак не связано с запуском команды `sar`. Команда просто считывает и сохраняет показания счетчиков, число измерений и их периодичность задаются при запуске этой команды.

В команде `sar` предусмотрено большое число опций, позволяющих получить информацию об очередях, подкачке, терминалах и других устройствах. Одной из важных особенностей команды `sar` является то, что она может создавать как отчеты с информацией об использовании всех процессоров системы в целом (такие отчеты содержат средние значения в процентах и суммарные значения), так и отчеты с информацией по каждому процессору в отдельности. Следовательно, эта команда будет особенно полезна в SMP-системах.

Собрать статистическую информацию о работе системы и сразу вывести ее на экран можно с помощью команды:

```
sar -u 2 5
```

Результат её работы:

```
03:47:38 PM     CPU    %user   %nice   %system   %iowait   %steal    %idle
03:47:40 PM     all     4.49    0.00      0.25     20.47     0.00    74.78
03:47:42 PM     all     7.25    0.00      0.38     18.88     0.00    73.50
03:47:44 PM     all     1.25    0.00      0.12     24.81     0.00    73.82
03:47:46 PM     all     4.62    0.00      0.25     24.59     0.00    70.54
03:47:48 PM     all     4.01    0.00      0.13     24.91     0.00    70.96
Average:        all     4.32    0.00      0.22     22.73     0.00    72.72
```

На этом примере продемонстрирована собранная информация об использовании CPU в количестве пяти выборок с интервалом в две секунды.

С помощью параметров <-o> и <-f> вы можете разбить работу команды на два этапа: сначала записать собранные данные в пользовательский файл данных, а затем просмотреть данные из него. Это позволяет сэкономить ресурсы, когда команда эмулирует в системе состояние неполадки. Полученные данные можно проанализировать на другом компьютере, скопировав в него двоичный файл вывода, содержащий всю информацию, необходимую для работы команды `sar`.

```
sar -o ./sar.out 2 5 > /dev/null
```

Приведенная выше команда запускает утилиту `sar` в фоновом режиме, собирает данные о работе системы пять раз с периодичностью в две секунды и сохраняет полученный неотформатированный результат в файл `./sar.out`. Перенаправление в файл происходит для того, чтобы не показывать его на экране. Воспользоваться полученным файлом можно командой:

```
sar -f ./sar.out
```

Двоичный файл данных содержит всю информацию, необходимую для создания отчетов. С его помощью можно получить любой отчет команды `sar`. Например, выполнив следующую команду, можно получить информацию об использовании каждого процессора в отдельности:

```
sar -P ALL -f ./sar.out
```

Результат работы:

```
03:51:16 PM     CPU     %user     %nice   %system   %iowait    %steal      %idle
03:51:18 PM     all     12.48      0.00      0.12      0.87      0.00      86.52
03:51:18 PM       0     50.00      0.00      0.50      0.00      0.00      49.50
03:51:18 PM       1      0.00      0.00      0.00      0.00      0.00     100.00
03:51:18 PM       2      0.00      0.00      0.00      3.48      0.00      96.52
03:51:18 PM       3      0.00      0.00      0.00      0.00      0.00     100.00
```

За более детальной информацией обратитесь к страницам руководства по этой команде (`man sar`).

Перечисленных выше команд мониторинга обычно достаточно для анализа состояния системы. Как их использовать - в составе скриптов, или нет - зависит исключительно от того, кто их использует.