

Содержание

| | |
|--|----|
| Предисловие | 14 |
| Благодарности | 16 |
| Об авторах | 19 |
| О технических рецензентах и прочих участниках проекта | 23 |
| Глава 1. Диагностика и настройка производительности сегментов LOB | 25 |
| Краткое описание типа данных LOB..... | 25 |
| Устранение проблемы с LOB-объектом: пример из практики..... | 26 |
| Еще один пример из реальной практики: HW Resolution..... | 28 |
| Проблемы BASICFILE LOB: более эффективное решение..... | 32 |
| Сравнение LOB-объектов BASICFILE и SECUREFILE LOB..... | 32 |
| Различия между новыми и старыми типами LOB..... | 33 |
| Преобразование BASICFILE LOB-объектов в SECUREFILE LOB-объекты..... | 36 |
| Воздействие параметра PCTFREE на LOB-объекты..... | 38 |
| Улучшение производительности операции вставки данных INSERT..... | 41 |
| Резюме..... | 42 |
| Глава 2. Восстановление табличного пространства, поврежденного при выполнении операции UNDO | 43 |
| Общий обзор процедуры отмены изменения данных..... | 43 |
| Важность параметра UNDO_RETENTION..... | 44 |
| Настройка параметра UNDO_RETENTION..... | 45 |
| Сегменты DTP, XA и сегмент отката (rollback)..... | 46 |
| Прочие нестандартные ситуации, возникающие в сегментах отката и сегментах Undo..... | 48 |
| Восстановление поврежденного табличного пространства Undo..... | 48 |
| Профилактика, обнаружение и восстановление повреждений..... | 49 |
| Обработка повреждений памяти..... | 50 |
| Обработка логических повреждений..... | 53 |
| Устранение повреждений носителей информации..... | 53 |
| Резюме..... | 56 |
| Глава 3. Обработка событий ожидания освобождения буферов глобального кэша | 58 |
| Обзор событий ожидания освобождения буферов..... | 58 |
| Практическое применение утилиты ORAchk..... | 59 |
| Установка утилиты ORAchk..... | 59 |
| Результаты выполнения ORAchk: пример выводимой информации..... | 61 |
| Устранение событий gc buffer busy wait..... | 61 |
| Использование ADDM для поиска информации о событии..... | 63 |
| Использование AWR для поиска информации о событии..... | 64 |
| Использование ASH для поиска информации о событиях..... | 66 |
| Определение проблем, связанных с событием gc buffer busy wait..... | 67 |
| Использование представлений ASH для поиска ожидающих сеансов..... | 68 |
| Быстрое определение узких мест, снижающих производительность..... | 69 |

| | |
|---|------------|
| Методики устранения событий gc buffer busy wait | 72 |
| Резюме..... | 73 |
| Глава 4. Адаптивное разделение курсора | 74 |
| Алгоритм работы механизма ACS | 75 |
| Чувствительность к связыванию с определением диапазона избирательности..... | 75 |
| Чувствительность к связыванию с предикатом равенства и гистограммой | 78 |
| Чувствительность к связыванию при секционировании по диапазонам ключей | 79 |
| Работа механизма адаптивного разделения курсора (ACS) | 81 |
| Мониторинг осведомленности о связывании в механизме ACS..... | 85 |
| Связь BUCKET_ID и COUNT..... | 86 |
| Как сделать курсор осведомленным о связывании | 89 |
| Курсор, осведомленный о связывании | 96 |
| Практический пример..... | 99 |
| Резюме..... | 105 |
| Глава 5. Стабилизация времени ответа на запрос с помощью механизма управления планами SQL..... | 106 |
| Общие положения | 107 |
| Создание базовой линии плана выполнения SQL..... | 110 |
| Автоматический захват плана выполнения..... | 110 |
| Загрузка планов из кэша курсора | 113 |
| Имитация базовых линий | 115 |
| Оптимизатор Oracle и его взаимодействие с механизмом управления планами SQL | 119 |
| План стоимостного оптимизатора не соответствует базовой линии плана выполнения SQL..... | 122 |
| Базовая линия плана выполнения SQL не является воспроизводимой..... | 128 |
| Воспроизводимость базовой линии плана выполнения SQL..... | 132 |
| Переименование индекса..... | 133 |
| Изменение типа индекса..... | 135 |
| Добавление в индекс замыкающих столбцов | 136 |
| Реверсирование индекса | 137 |
| Параметр NLS_SORT и воспроизводимость базовой линии плана выполнения SQL | 138 |
| Сравнение параметров ALL_ROWS и FIRST_ROWS..... | 142 |
| Адаптивное разделение курсора и механизм управления планами SQL..... | 146 |
| Взаимодействие механизмов ACS и SPM в версии Oracle 11g Release 11.2.0.3.0..... | 147 |
| Взаимодействие механизмов ACS и SPM в версии Oracle 12c Release 12.1.0.1.0 | 152 |
| Резюме..... | 155 |
| Глава 6. Советы, методики и особые приемы оптимизации для языка определения данных (DDL)..... | 156 |
| Основы оптимизации операций языка определения данных | 156 |
| Механизм оптимизации операций языка определения данных (DDL)..... | 159 |
| Оценка мощности таблицы | 160 |
| Столбец C_DDL в виртуальном столбце..... | 162 |
| Столбец C_DDL в расширении группы столбцов | 163 |
| Что происходит при изменении значения по умолчанию для столбца C_DDL..... | 165 |
| Столбец C_DDL и индексы | 168 |
| Оптимизация операций языка определения данных для столбцов с атрибутом NULL | 170 |
| Резюме..... | 175 |

| | |
|---|-----|
| Глава 7. Управление, оптимизация и настройка очень больших баз данных | 176 |
| Общие сведения об очень больших базах данных | 176 |
| Оптимальная базовая конфигурация | 177 |
| Шаблон конфигурации хранилища данных | 178 |
| Выбор оптимального размера блока данных..... | 178 |
| Табличные пространства большого файла..... | 179 |
| Выбор правильных размеров системной глобальной области (SGA) и программной глобальной области (PGA) | 180 |
| Группы временных табличных пространств..... | 181 |
| Секционирование данных..... | 181 |
| Секционирование по индексу: сравнение локального и глобального индексов | 182 |
| Сжатие данных | 183 |
| Сжатие таблицы | 183 |
| Инструменты Heat Map и Automatic Data Optimization (ADO)..... | 184 |
| Расширенная функция сжатия секционированного индекса | 185 |
| Основные правила настройки производительности очень большой БД | 185 |
| Пример из реальной жизни..... | 186 |
| Ограничение воздействия индексов на операции загрузки данных..... | 187 |
| Максимальное использование ресурса | 188 |
| Сбор статистических данных оптимизатора..... | 189 |
| Краткий обзор постепенно накапливаемых статистических данных | 189 |
| Сбор статистических данных в параллельном режиме..... | 191 |
| Установка значения параметра ESTIMATE_PERCENT..... | 193 |
| Наилучшие практические методики резервного копирования и восстановления данных..... | 193 |
| Решения на основе Exadata | 195 |
| Использование среды Data Guard | 195 |
| Резюме..... | 195 |
| | |
| Глава 8. Эффективные практические методики резервного копирования и восстановления данных с использованием диспетчера восстановления | 196 |
| Идеальный план резервного копирования и восстановления..... | 196 |
| Общий обзор диспетчера восстановления | 197 |
| Рекомендации по проектированию стратегий резервного копирования баз данных..... | 198 |
| Процедуры полного и инкрементального резервного копирования..... | 199 |
| Резервное копирование со сжатием данных | 199 |
| Инкрементальное резервное копирование | 200 |
| Ускоренные операции инкрементального резервного копирования..... | 201 |
| Операции отката к предыдущему состоянию в технологии Oracle Flashback..... | 202 |
| Резервное копирование с использованием дисковой памяти | 203 |
| Стратегия Recover Forward Forever | 203 |
| Проверка корректности резервных копий диспетчера восстановления | 211 |
| Оптимизация и настройка операций резервного копирования..... | 212 |
| Настройка производительности операций резервного копирования с использованием дисковых накопителей | 213 |
| Использование диспетчера восстановления в кластерных (RAC) базах данных | 214 |
| Хранение данных в каталоге восстановления..... | 216 |
| Надежная стратегия восстановления..... | 216 |
| Использование консультанта по восстановлению данных DRA | 218 |
| Резюме..... | 219 |

| | |
|--|-----|
| Глава 9. Методики тестирования и настройки базы данных с использованием анализа автоматического репозитория рабочей нагрузки: часть 1 | 221 |
| Общий обзор автоматического репозитория рабочей нагрузки | 222 |
| Что нужно искать | 223 |
| Раздел заголовка | 224 |
| Профиль нагрузки | 226 |
| Характеристики производительности экземпляра БД | 227 |
| Совместно используемый пул оперативной памяти | 228 |
| Ожидание событий | 228 |
| Средняя нагрузка | 231 |
| Использование процессоров экземпляром БД | 232 |
| Статистические характеристики оперативной памяти | 232 |
| Специальные разделы отчета по кластерным (RAC) базам данных | 233 |
| Статистические данные по использованию процессоров в кластерной базе данных | 234 |
| Статистические данные по нагрузке глобального кэша | 234 |
| Глобальный кэш и сервисы формирования очереди | 234 |
| Сетевые соединения в кластере | 236 |
| Статистические характеристики временной модели | 236 |
| Статистические характеристики операционной системы | 237 |
| Ожидание событий в интерактивном режиме | 238 |
| Ожидание событий в фоновом режиме | 240 |
| Гистограммы времени ожидания событий | 241 |
| Статистические данные по сервисам | 242 |
| Раздел команд SQL | 243 |
| Общее затраченное время | 243 |
| Общее процессорное время | 244 |
| Общее время использования буфера | 244 |
| Общее время операций чтения с диска | 245 |
| Общее количество выполнения команд | 245 |
| Общее количество операций (вызовов) синтаксического анализа | 245 |
| Совместно используемая память | 246 |
| Счетчик версий команд | 246 |
| Общее время ожидания событий в кластере | 247 |
| Статистические данные по операциям в экземпляре базы данных | 247 |
| Статистические данные для согласованных операций чтения | 250 |
| Статистические характеристики операций получения блоков базы данных | 251 |
| Статистические данные по «грязным» блокам | 251 |
| Статистические данные по очередям | 251 |
| Счетчик выполнения | 251 |
| Статистические данные по свободным буферам | 251 |
| Статистические данные по глобальному кэшу (GC) | 252 |
| Статистические данные для поиска по индексу | 252 |
| Статистические данные по листьям-узлам в B-деревьях | 253 |
| Статистические данные по открытым курсорам | 253 |
| Статистические данные по операциям синтаксического анализа | 253 |
| Статистические данные по физическим операциям чтения и записи | 253 |
| Статистические данные по рекурсивным операциям | 256 |
| Статистические данные о повторно выполняемых командах | 256 |
| Статистические данные по курсорам сеанса | 257 |

| | |
|--|-----|
| Статистические данные по операциям сортировки..... | 257 |
| Общая длина «грязной» очереди | 257 |
| Статистические данные по выборкам из таблиц..... | 258 |
| Откаты транзакций..... | 258 |
| Статистические данные по вектору изменений отмены операций..... | 259 |
| Статистические данные пользователей | 259 |
| Статистические данные по рабочей области | 260 |
| Статистические данные по операциям экземпляра БД – абсолютные значения | 260 |
| Статистические данные по операциям экземпляра БД – потоковые операции..... | 260 |
| Резюме..... | 260 |

Глава 10. Методики тестирования и настройки базы данных с использованием анализа автоматического репозитория

| | |
|--|------------|
| рабочей нагрузки: часть 2 | 261 |
| Статистические данные по вводу/выводу в табличном пространстве..... | 261 |
| Статистические данные по пулу буферов..... | 263 |
| Статистические данные по пулу буферов | 265 |
| Статистические данные по восстановлению экземпляра..... | 265 |
| Раздел рекомендаций по пулу буферов..... | 266 |
| Статистические данные по программной глобальной области..... | 266 |
| Общие показатели по программной глобальной области | 268 |
| Сводные статистические показатели цели PGA | 268 |
| Сводная гистограмма цели PGA..... | 268 |
| Рекомендации по использованию памяти в PGA..... | 270 |
| Статистические данные по совместно используемому пулу | 271 |
| Прочие рекомендации | 271 |
| Рекомендации по параметру SGA_TARGET | 273 |
| Рекомендации по пулу потоков (streams) | 273 |
| Рекомендации по пулу Java | 273 |
| Статистические данные по ожиданиям, связанным с буферами | 273 |
| Статистические данные по очередям | 275 |
| Статистические данные по сегментам отмены действий (откатов)..... | 276 |
| Статистические данные по защелкам..... | 278 |
| Операции с защелками..... | 280 |
| Нарушения состояния «сна» защелок..... | 280 |
| Счетчик защелок и циклов ожиданий..... | 280 |
| Защелки как источники промахов..... | 281 |
| Обзор спящих мьютексов..... | 281 |
| Защелки-предки и защелки-потомки | 282 |
| Области доступа к сегментам | 282 |
| Разделы операций с библиотечным кэшем..... | 284 |
| Разделы компонентов динамической памяти | 287 |
| Разделы памяти процессов..... | 289 |
| Общая сводка по памяти процессов | 290 |
| Общая сводка по памяти системной глобальной области | 291 |
| Различия в сегментации системной глобальной области..... | 291 |
| Разделы компонентов потоков..... | 291 |
| Статистические данные по ограничениям ресурсов | 293 |
| Изменения параметров инициализации..... | 294 |
| Статистические данные по глобальной очереди и другие разделы по кластерной БД..... | 295 |
| Статистические данные по глобальной очереди..... | 298 |
| Статистические данные по службе глобального согласованного чтения | 298 |

| | |
|--|-----|
| Статистические данные по обслуживанию текущих глобальных объектов | 299 |
| Статистические данные по обмену с глобальным кэшем..... | 299 |
| Статистические данные по временам обмена с глобальным кэшем | 299 |
| Статистические данные по прямому обмену с глобальным кэшем..... | 299 |
| Статистические данные по времени прямого обмена с глобальным кэшем | 300 |
| Статистические данные по задержкам пингования сетевого соединения..... | 300 |
| Пропускная способность сетевого соединения на стороне клиента..... | 300 |
| Статистические данные об устройстве сетевого соединения..... | 300 |
| Резюме..... | 301 |

Глава 11. Сценарии устранения проблем в кластерных базах данных..... 302

| | |
|---|-----|
| Устранение проблем и настройка кластерной базы данных..... | 303 |
| Первая проверка с помощью ORAchk..... | 303 |
| Использование утилиты TFA Collector | 303 |
| Использование репозитория автоматической диагностики | 303 |
| Проверка журнальных файлов предупреждений и трассировки | 304 |
| Применение трех «А»..... | 304 |
| Проверка защищенного соединения в кластерной БД..... | 304 |
| Установка режима трассировки и отслеживание журналов трассировки..... | 304 |
| Использование монитора работоспособности кластера..... | 305 |
| Прочие инструменты и утилиты..... | 305 |
| Полезные ресурсы My Oracle Support (MOS) | 305 |
| Бесперебойно работающая экосистема RAC | 305 |
| Архитектура, обеспечивающая максимальную доступность | 306 |
| Оптимизированные и эффективные базы данных в кластере RAC | 307 |
| Устранение проблем в кластере RAC с помощью Диспетчера предприятия Oracle 12c | 309 |
| Утилиты и команды для устранения проблем | 309 |
| Резюме..... | 315 |

Глава 12. Использование консультантов по командам SQL

| | |
|--|------------|
| для анализа и устранения проблем с языком SQL | 316 |
| OEM 12c – SQL Advisors Home | 317 |
| Консультант по настройке команд SQL..... | 317 |
| Запуск консультанта по настройке команд SQL в среде OEM 12c | 318 |
| Запуск консультанта по настройке команд SQL вручную в среде SQL*Plus | 321 |
| Консультант по оптимизации доступа SQL | 322 |
| Запуск консультанта по оптимизации доступа SQL в среде OEM 12c..... | 322 |
| Запуск консультанта по оптимизации доступа SQL вручную в среде SQL*Plus..... | 325 |
| Консультант по исправлению кода SQL..... | 326 |
| Анализатор производительности SQL | 327 |
| Резюме..... | 328 |

Глава 13. Применение утилиты Data Pump для перемещения данных и объектов..... 329

| | |
|--|-----|
| Использование механизма Data Pump..... | 329 |
| Копирование объектов | 330 |
| Режимы механизма Data Pump | 331 |
| Работа с частными и общедоступными объектами | 332 |
| Сохранение и восстановление связей БД..... | 332 |
| Экспорт общедоступных связей БД и синонимов | 333 |
| Проверка корректности содержимого файла дампа экспорта | 334 |

| | |
|---|-----|
| Поиск корректных значений параметров INCLUDE и EXCLUDE..... | 334 |
| Экспорт подмножеств данных..... | 336 |
| Изменение свойств объекта..... | 338 |
| Импорт секционированных таблиц как несекционированных..... | 338 |
| Импорт секций таблицы как отдельных таблиц..... | 339 |
| Маскирование данных..... | 339 |
| Переименование таблиц или изменение табличных пространств..... | 339 |
| Использование параметров хранения по умолчанию..... | 340 |
| Изменение размеров табличных пространств во время импорта..... | 340 |
| Объединение нескольких табличных пространств..... | 340 |
| Использование программного интерфейса (API) PL/SQL совместно с Data Pump..... | 342 |
| Контроль и изменение ресурсов..... | 344 |
| Повышение производительности..... | 345 |
| Обновление баз данных..... | 346 |
| Резюме..... | 347 |

Глава 14. Стратегии быстрого перемещения данных

| | |
|--|------------|
| между базами данных..... | 349 |
| Почему необходимо перемещение данных..... | 350 |
| Определение наилучшей стратегии..... | 350 |
| Сравнение перемещения в реальном времени с перемещением в «почти реальном времени»..... | 351 |
| Способность работать в режиме только для чтения..... | 351 |
| Обратимость..... | 351 |
| Какие данные действительно требуют перемещения..... | 352 |
| Методики перемещения данных..... | 352 |
| Методики перемещения с возможностью фиксации транзакций..... | 353 |
| Методики перемещения без возможности выполнения транзакций..... | 355 |
| Методики постепенного перемещения..... | 369 |
| Резюме..... | 376 |

Глава 15. Диагностика проблем и восстановление из временного файла

| | |
|--|------------|
| ввода/вывода TEMPFILE..... | 377 |
| Общий обзор временных табличных пространств..... | 377 |
| Базы данных в режиме только для чтения..... | 378 |
| Локально управляемые временные табличные пространства..... | 378 |
| Группы временных табличных пространств..... | 379 |
| Глобальные временные таблицы..... | 380 |
| Корректировка файла TEMPFILE для изменения состояний ожидания завершения операций ввода/вывода..... | 383 |
| Недостаточный размер программной глобальной области..... | 384 |
| Нецелесообразное изменение размера экстенда TEMPFILE..... | 388 |
| Нецелесообразное использование групп временных табличных пространств..... | 389 |
| Резюме..... | 389 |

Глава 16. Работа с защелками и обработка состояния конкуренции

| | |
|---|------------|
| между мьютексами..... | 390 |
| Обзор архитектуры защелок и мьютексов..... | 390 |
| Что такое защелки..... | 391 |
| Что такое мьютексы..... | 393 |
| Внутреннее устройство защелки и мьютекса..... | 393 |

| | |
|---|-----|
| Количественные характеристики конкуренции защелок и мьютексов | 394 |
| Идентификация отдельных защелок..... | 396 |
| Исследование сегментов и команд SQL..... | 396 |
| Сценарии использования защелок и мьютексов | 398 |
| Ожидание библиотечного кэша для мьютексов | 399 |
| Ожидание освобождения занятого библиотечного кэша | 401 |
| Зашелка совместно используемого пула | 401 |
| Зашелка цепочек кэш-буферов | 402 |
| Прочие ситуации конкуренции за защелки | 405 |
| Трудноразрешимые проблемы конкуренции за защелки | 407 |
| Алгоритмы точной настройки защелок | 407 |
| Резюме..... | 408 |

Глава 17. Использование SSD-накопителей для устранения проблем

| | |
|--|------------|
| с производительностью подсистемы ввода/вывода | 410 |
| Сравнение SSD-технологии с HDD-технологией | 411 |
| Появление твердотельных flash-накопителей | 412 |
| Задержки flash SSD-накопителей..... | 413 |
| Экономические характеристики SSD-накопителей | 413 |
| Накопители SLC, MLC и TLC | 415 |
| Производительность операций записи и долговечность..... | 416 |
| Сборка мусора и регулирование уровня износа | 416 |
| Сравнение SATA SSD и PCIe SSD..... | 418 |
| Использование SSD-устройств в базах данных Oracle | 419 |
| Механизм Oracle Database Flash Cache..... | 419 |
| Случаи ожидания свободных буферов | 419 |
| Конфигурирование и регулирование механизма DBFC | 422 |
| Использование опции FLASH_CACHE | 423 |
| Статистические данные о производительности flash-кэша | 424 |
| Сравнение характеристик SSD-накопителей | 426 |
| Чтение индексированных данных..... | 426 |
| Рабочая нагрузка при операциях чтения/записи в OLTP-системе..... | 427 |
| Производительность при поиске по всей таблице | 427 |
| Собственный кэш SSD-накопителей и поиск по всей таблице | 428 |
| Операции сортировки с использованием диска и операции хэширования..... | 429 |
| Оптимизация для журналов повторно выполняемых операций | 430 |
| Расслоение хранимых данных..... | 434 |
| Использование секций для расслоения данных | 434 |
| Flash-устройства и комплекс Exadata..... | 438 |
| Создание групп дисков ASM на основе flash-устройств в комплексе Exadata..... | 440 |
| Резюме..... | 442 |

Глава 18. Проектирование и контроль индексов для достижения

| | |
|--|------------|
| оптимальной производительности | 443 |
| Типы индексов..... | 443 |
| B-tree-индексы | 444 |
| Битовые индексы | 447 |
| Секционированные индексы | 449 |
| Другие типы индексов | 452 |
| Сжатые индексы..... | 453 |
| Несколько индексов по одинаковым столбцам..... | 454 |

| | |
|---|-----|
| Проблемы производительности индексов..... | 454 |
| Статистика по индексам..... | 454 |
| Влияние низкого фактора кластеризации | 456 |
| Важность индексов при выполнении операций..... | 457 |
| Скрытие неиспользуемых индексов..... | 459 |
| Проблемы производительности индексов в кластерных (RAC) базах данных..... | 461 |
| Резюме..... | 463 |

Глава 19. Использование SQLT для повышения производительности

| | |
|--|-----|
| запросов | 464 |
| Установка утилиты SQLT..... | 465 |
| Использование метода XTRACT | 466 |
| Использование метода XECUTE..... | 467 |
| Применение других методов утилиты SQLT | 470 |
| Пример из реальной практики..... | 471 |
| Резюме..... | 472 |

Глава 20. Устранение проблем в распределенных транзакциях

| | |
|---|-----|
| расширенной архитектуры (XA) | 473 |
| Устранение общих проблем при использовании распределенных транзакций..... | 474 |
| Восстановление распределенных транзакций-«невидимок» | 475 |
| Информация существует, но транзакция отсутствует..... | 475 |
| При возникновении ошибки ORA-1591 не найдено соответствующей информации | 476 |
| Транзакция зависит при попытке выполнения команды COMMIT или ROLLBACK..... | 478 |
| Контроль распределенных транзакций | 481 |
| Резюме..... | 482 |

| | |
|-----------------------------------|-----|
| Предметный указатель | 483 |
|-----------------------------------|-----|

Предисловие

Жизнь администраторов СУБД становится все более и более сложной, а напряженный режим работы быстро становится нормой. Администраторы баз данных постоянно сталкиваются с проблемами, которые при определенных условиях могут привести организации и прочие юридические лица к многомиллионным убыткам буквально за минуту или, при самом худшем развитии событий, к полной остановке работы инфраструктуры баз данных компании. Конечно, подобные ситуации маловероятны, но к потенциальной возможности их возникновения должен быть готов каждый администратор баз данных, чтобы вовремя устранить надвигающиеся проблемы.

Главная задача этой книги – показать способы максимально оперативного устранения в базах данных серьезных проблем, которые могут оказать воздействие на эксплуатационную среду в целом. Читателям демонстрируется пошаговое руководство, необходимое для решения конкретной проблемы, с разбором случаев из реальной практики, которые могут произойти (и происходят) в любой день, в любое время, в любой базе данных Oracle.

Чтобы не терять времени в попытках поиска решения проблемы, которая представляет опасность для вашей базы данных или уже нарушила ее работоспособность, вы можете сразу же обратиться к этой книге, предлагающей решения некоторых самых крупных проблем, вероятность возникновения которых наиболее велика. Даже если решение текущей проблемы не найдено здесь, вы узнаете, как нужно быстро искать в Интернете способы устранения любой конкретной проблемы.

Основная идея данной книги заключается в том, чтобы предложить вам реальную помощь при возникновении серьезных проблем с базами данных Oracle в различных эксплуатационных средах. Наряду с описанием самых эффективных обобщенных методик в книге исследуются некоторые наиболее важные проблемы, возникающие в базах данных Oracle, а также способы оперативного решения этих проблем, описанные в простой и понятной форме.

Ориентированная в основном на администраторов СУБД Oracle (DBA) и на администраторов аппаратно-программных комплексов (кластеров серверов) Oracle Database Machine (DMA) книга «Руководство по диагностике и устранению проблем в СУБД Oracle» может служить практическим техническим руководством для выполнения повседневных действий по диагностике и устранению неисправностей, по настройке и устранению проблем в ходе операций администрирования семейства программных продуктов Oracle Database Server.

Написанная известными во всем мире авторами с огромным опытом работы в группах Oracle ACE, в качестве руководителей ACE и независимых экспертов, эта книга должна стать настольным справочником по решению проблем, объединяющим практические примеры из реальной жизни и планы действий по диагностике и устранению неисправностей в сложных совокупностях баз данных Oracle. В данной книге подробно описаны методы решения следующих задач:

- выбор наикратчайшего пути для решения крупномасштабных проблем;
- наиболее эффективная организация рабочего дня с помощью надежных методик, основанных на практическом опыте экспертов этой области;

- создание собственного плана по устранению неисправностей и решению проблем;
- осуществление повседневного «упреждающего» сопровождения, обеспечивающего стабильность эксплуатационной среды;
- использование стандартных инструментальных средств и сценариев, наиболее пригодных для быстрых и эффективных решений проблем.

Авторы этой полезной, изобилующей разнообразными техническими подробностями книги при ее написании ориентировались на различные уровни (средний, опытный эксперт) пользователей семейства программных продуктов Oracle Database Server. В этой книге рассматриваются версии Oracle Database 11g и Oracle Database 12c и весь комплект соответствующего вспомогательного и дополнительного программного обеспечения для СУБД Oracle.

Глава 1

Диагностика и настройка производительности сегментов LOB

Тип данных LOB (или LOB-объект; Large Object) позволяет хранить и обрабатывать неструктурированные или слабоструктурированные данные, такие как документы, графические изображения, фрагменты видео, звуковые файлы и XML-файлы. Пакет `DBMS_LOB` был специально разработан для обработки данных типа LOB. Начиная с версии Oracle Database 12c LOB-объекты могут хранить большие объемы данных с максимальным размером до 128 Тб в зависимости от размера блока, установленного для конкретной базы данных. Отдельная таблица может содержать один или несколько столбцов данных типа LOB, например бинарный LOB-объект (BLOB), символьный LOB-объект (CLOB), LOB-объект, содержащий национальные символы (NCLOB), и файл (BFILE). В этой главе описываются некоторые проблемы, которые могут возникать при наличии сегментов LOB в конкретной среде, а также способы устранения этих проблем.

КРАТКОЕ ОПИСАНИЕ ТИПА ДАННЫХ LOB

При создании и проектировании LOB-объектов в своих базах данных помните о необходимости внимательного изучения руководства Oracle Database Secure Files and Large Objects Developer's Guide (Oracle, 2016), особенно главы 16 «Performance Guidelines» (Правила настройки производительности). Это руководство предоставит вам самые актуальные рекомендации по созданию LOB-объектов в зависимости от того, какие именно типы данных будут размещаться в этих столбцах таблицы.

Важно помнить, что при создании столбца для LOB-объекта в таблице в действительности создаются два различных сегмента: `LOBSEGMENT` и `LOBINDEX`. Сегмент `LOBINDEX` указывает на «фрагменты» LOB-объекта, которые хранятся в соответствующем сегменте `LOBSEGMENT`. В некоторых случаях LOB-объекты могут храниться как «встроенные» (inline), то есть прямо внутри сегмента таблицы, но такой способ хранения обычно используется для относительно небольших объемов данных в LOB-объекте (не более 4000 байтов) или для значения `NULL`. При соблюдении этих условий значения LOB-объектов хранятся непосредственно в сегменте таблицы.

Рассмотрим ситуацию, в которой база данных вынуждена сгенерировать событие маркера максимального уровня заполнения (high-watermark – HW) при включении в очередь. Когда в сеансе требуется доступ к ресурсу базы данных, типом блокировки, управляющей доступом к этому ресурсу, является включение в очередь (enqueue). Возникновение события при включении в очередь (тип enqueue) означает, что сеанс должен ждать, когда другой сеанс освободит данный ресурс. В сообщении о таком событии всегда указывается имя очереди в формате `enq: <enqueue_type >- <details>`, причем для каждого типа очереди подробности будут различными. Столбцы P1, P2 и P3 динамического вида (представления) `V$SESSION` и `V$SESSION_WAIT` также помогают узнать более подробно о том, в каком текущем состоянии находится ожидающий сеанс и что именно стало причиной блокировки. Эти значения могут иметь различный смысл в зависимости от типа очереди, как можно видеть в соответствующих столбцах `P1TEXT`, `P2TEXT` и `P3TEXT`. В листинге 1.1 показано лишь несколько из более чем 600 событий, возникших при включении в очередь.

Листинг 1.1 ❖ События при включении в очередь

```
select
  distinct name
from
  v$event_name
where
  name like '%enq%'
order by 1;

enq: AB = ABMR process initialized
enq: AB = ABMR process start/stop
enq: AC = acquiring partition id
enq: AD = allocate AU
enq: AD = deallocate AU
enq: AD = relocate AU
enq: AE = lock
...
```

Событие HW при включении в очередь является ответной реакцией на попытку распределения пространства за пределом наивысшего допустимого уровня для данного сегмента. Рекомендуемое действие по обработке непредвиденного события HW при включении в очередь – выделение дополнительных экстенгов (extent) для данных сегментов LOB-объектов вручную. Но в последующих разделах мы будем рассматривать и профилактические методы, помогающие избежать событий HW при включении в очередь посредством реализации наиболее эффективных практических методик конфигурирования LOB-объектов.

Устранение проблемы с LOB-объектом: пример из практики

В приводимом ниже примере, взятом из реальной практики, база данных одной компании, занимающейся электронной коммерцией, полностью зависла. Следующие действия помогают быстро определить и устранить возникшую проблему:

1. Создание отчета по автоматическому репозиторию рабочей нагрузки (automatic workload repository – AWR), чтобы определить, не является ли событие `enq: HW` одним из ожидающих событий в пяти верхних строках отчета. За-

регистрируйтесь в своей базе данных как SYSDBA и выполните следующую команду. После появления промпта выберите два самых последних идентификатора снимков AWR, чтобы исследовать только два самых свежих снимка:

```
$> ?/rdbms/admin/awrrpt.sql
```

- Изучение первых пяти ожидающих событий, приведенных в полученном отчете:

| Event | Waits | Time (s) | (ms) | Time | Wait Class |
|-----------------------------|-----------|----------|------|------|-----------------|
| enq:HW contention | 249,725 | 3,289 | 13 | 90.0 | User I/O direct |
| path write | 168,486 | 103 | 1 | 2.8 | User I/O DB CPU |
| PX qref latch | 6,392,581 | 40 | 0 | 1.1 | Other |
| PX Deq: Slave Session Stats | 18 | 1 | 51 | .0 | Other |

- Если искомое событие постоянно возникает в базе данных, просто выполните следующий запрос для определения сеансов, в которых имеет место событие HW (contention):

```
SELECT sid, event
FROM gv$session_wait
WHERE event LIKE '%contention%';
```

| SID | EVENT |
|-------|----------------------|
| 9426 | enq: HW = contention |
| 13050 | enq: HW = contention |

- Выполните следующий запрос для выделения конкретного объекта, в котором возникло событие достижения наивысшего допустимого уровня HW:

```
SELECT
DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE(id2) file#
DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK(id2) block#
FROM gv$lock
WHERE type = 'HW';
```

| FILE# | BLOCK# |
|-------|--------|
| 19 | 195 |

- Используя номер файла (19) и идентификатор блока (195), полученные из предыдущего запроса, найдите объект, заблокированный по данному событию, с помощью следующего запроса:

```
SELECT
owner,
segment_type,
segment_name
FROM
dba_extents
WHERE
file_id = 19
AND
195 between block_id
AND
block_id + blocks = 1;
```

```
OWNER          SEGMENT_TYPE    SEGMENT_NAME
-----
ORABPEL        LOBSEGMENT      SYS_LOB0000181226C00029$$
```

6. Далее необходимо определить имя таблицы, которая ссылается на этот LOB-сегмент, с помощью следующего запроса:

```
SELECT
  owner,
  table_name,
  segment_name
FROM
  dba_lobs
WHERE
  segment_name= 'SYS_LOB0000181226C00029$$';
```

```
OWNER          TABLE_NAME      SEGMENT_NAME
-----
ACOM_BPPEL_AQ  INSERT_SITE_ORDER_BI_TBL  SYS_LOB0000181226C00029$$
```

7. После точного определения LOB-сегмента, из-за которого возникло событие переполнения очереди HW, следует частично устранить главную причину возникшей проблемы, вручную добавив новый экстенс к этому сегменту. Важно знать, размещается ли объект в табличном пространстве, и если это так, то имеют ли его сегменты размер `AUTO_ALLOCATED` или `UNIFORM`. Если экстенсы определены как `UNIFORM`, то нужно просто добавить новый экстенс того же размера, в противном случае следует добавить новый экстенс с размером, равным размеру самого большого экстенса для этого сегмента:
- a. Определить наибольший размер экстенса для данного объекта:

```
SELECT
  DISTINCT bytes
FROM dba_extents
WHERE segment_name = 'SYS_LOB0000181226C00029$$'
AND owner = 'ORABPEL';

BYTES
-----
1048576
8388608
65536
```

- b. Добавить экстенсы такого же размера в данный LOB-сегмент:

```
ALTER TABLE orabpel.insert_site_order_bi_tbl
  MODIFY LOB ('SYS_LOB0000055018C00004$$')
  (ALLOCATE EXTENT (SIZE 8M));
```

Поздравляем, с помощью этой простой операции вы только что спасли свою компанию от миллионных убытков из-за существенных задержек или даже полной остановки обслуживания клиентов на веб-сайте электронной торговли.

Еще один пример из реальной практики: HW Resolution

Если база данных является репозиторием для таких приложений, как Oracle Transportation Management (OTM), Oracle Business Process Execution Language (BPEL) и Oracle E-Business Suite (EBS), то, вероятнее всего, в ней будут возникать

непредвиденные проблемы, связанные с LOB-сегментами. Например, недавно в OTM возникла подобная проблема в таблице I_TRANSMISSION при выполнении 150 000 000 команд INSERT и DELETE и 500 000 000 команд UPDATE в столбце XML_BLOB этой таблицы за один месяц. Ниже приведена последовательность действий по оперативному выявлению и устранению этой проблемы:

Примечание Описанный ниже подход также можно использовать в предыдущем примере enq: HW. Этот пример просто демонстрирует другой вариант выявления и решения данной проблемы.

1. Выполнить команду создания отчета по AWR и проверить раздел «Top 5 Timed Events». Если проблема с LOB-сегментами действительно возникла, то вы должны наблюдать в отчете нечто подобное:

Top 5 Timed Events

```

-----

```

| Event | Waits | Time (s) | Avg Wait (ms) | Total Time % | Call Wait Class |
|-------------------------------|---------------|--------------|---------------|--------------|----------------------|
| db file sequential read | 2,580,474 | 35,544 | 14 | 77.4 | User I/O |
| SQL*Net more data from client | 659,140 | 5,513 | 8 ** | 12.0 | Network |
| CPU time | | 4,540 | 9.9 | | |
| enq: HW - contention | 88,910 | 2,890 | 33 * | 6.3 | Configuration |
| log file sync | 777,146 | 1,688 | 2 | 3.7 | Commit |

2. В том же отчете по AWR проверить раздел «Top Enqueue Activity» на наличие в нем содержимого, подобного выделенному фрагменту:

Enqueue Activity Snaps: 1234-1235

-> only enqueues with waits are shown

-> Enqueue stats gathered prior to 10g should not be compared with 10g data

-> ordered by Wait Time desc, Waits desc

| Enq Type | Reg | Gets Succ | Gets Failed | Waits | Wait Time (s) | Avg Wait Time (ms) |
|--------------------------------------|---------------|---------------|-------------|---------------|---------------|--------------------|
| HW-Segment | | | | | | |
| High Water Mark | 93,860 | 93,862 | 0 | 88,226 | 2,961 | 33.56 * |
| TX-Transaction (row lock contention) | 272 | 272 | 0 | 209 | 570 | 2,729.44 ** |
| TX-Transaction (index contention) | 4,564 | 4,564 | 0 | 4,144 | 34 | 8.16 |
| TX-Transaction | 793,989 | 794,042 | 0 | 97 | 0 | 4.08 |

3. Проверить раздел «Wait Events» того же отчета по AWR. В этом разделе должна выводиться информация, подобная следующей:

-> s -second

-> ms - millisecond - 1000th of a second

-> ordered by wait time desc, waits desc (idle events last)

| Event | Waits | % Time -outs | Total Wait Time (s) | Avg Wait (ms) | Waits /txn |
|----------------------------------|---------------|--------------|---------------------|---------------|------------|
| db file sequential read | 2,580,474 | .0 | 35,544 | 14 | 3.3 |
| SQL*Net more data from client ** | 659,140 | .0 | 5,513 | 8 | 0.9 |
| enq: HW - contention | 88,910 | .0 | 2,890 | 33* | 0.1 |
| log file sync | 777,146 | .0 | 1,688 | 2 | 1.0 |

| | | | | | |
|--|------------|----------------|------------|--------------|------------|
| read by other session | 103,140 | .0 | 929 | 9 | 0.1 |
| SQL*Net break/reset to client | 114,782 | .0 | 813 | 7 | 0.1 |
| enq: TX - row lock contention *** | 380 | 43.4*** | 557 | 1466* | 0.0 |
| log file parallel write | 552,663 | .0 | 394 | 1 | 0.7 |
| latch: cache buffers chains | 55,203 | .0 | 382 | 7 | 0.1 |

4. После сбора всей необходимой информации нужно найти объект, в котором было сгенерировано рассматриваемое событие ожидания enq: HW. Выполнить следующий запрос для выделения всех объектов, в которых встречается запись enq: HW между снимками AWR с интервалом, выбранным для создания отчетов по AWR:

```
SELECT
  sql_id,
  event,
  event_id,
  time_waited,
  current_obj#,
  current_file#,
  current_block#
FROM dba_hist_active_sess_history
WHERE snap_id BETWEEN 1072 AND 1076
AND event LIKE '%HW%CONTENTION%'
AND time_waited > 0
AND current_obj# <> -1
ORDER BY time_waited, event, sql_id;
```

5. Перехватить значения для current_obj# из предыдущего запроса и передать их в следующий запрос:

```
SELECT
  owner,
  object_name,
  object_type
FROM dba_objects
WHERE object_id = [current_obj# of query above];
```

Приведенный ниже альтернативный запрос выводит крайне важную информацию, необходимую для выделения имени объекта, его типа, и (что наиболее важно) SQL-идентификатор, соответствующий командам, на которые оказало воздействие событие enq: HW:

```
SQL> col object_name format a30
SQL> col program format a30
SQL> col event format a30
SQL> SELECT DISTINCT CURRENT_OBJ#,o.object_name,o.owner,o.
object_type,CURRENT_BLOCK#,SESSION_STATE,SQL_ID,EVENT
  from v$active_session_history a, dba_objects o
  where a.current_obj# = o.object_id
  and a.event like 'enq%HW%';
```

| CUR_ | OBJ_NAME | OWN | OBJECT_ | CUR_ | SESS | SQL_ID | EVENT |
|--------|------------------|-----|---------|--------|---------|---------------|---------------------------------|
| OBJ# | | | TYPE | BLOCK# | | | |
| 235738 | MLOG\$_ENI_OLTP_ | ENI | TABLE | 100747 | WAITING | 0ghshjjvf86bg | enq: HW - contention |
| | ITEM_STAR | | | | | | |
| 612464 | HIST_PEDIDOS | B2W | TABLE | 394731 | WAITING | 9phv0npccjqa2 | enq: HW - contention |

6. Перед добавлением экстенгов к данному объекту рекомендуется проверить размеры его существующих экстенгов:

```
SELECT COUNT(*), bytes/1024/1024 "MB"
   FROM dba_extents
  WHERE segment_name = 'HIST_PEDIDOS' AND owner='B2W'
  GROUP BY bytes;
COUNT(*)    BYTES
-----
1969801      131072
```

7. Теперь, когда известен наибольший размер экстенга для данного объекта, вы просто добавляете несколько новых экстенгов. Используйте язык SQL для компоновки необходимых команд `MODIFY` на основании полученной ранее информации:

```
SELECT
  'alter table '||table_owner||'.'||table_name||' modify partition '||partition_name||' lob
('||column_name ||') (allocate extent (size 131072));'
  FROM dba_lob_partitions
 WHERE table_name = 'HIST_PEDIDOS'
    AND partition_name like '%2014%';
```

Наш богатый практический опыт позволяет утверждать, что после добавления достаточного количества новых экстенгов (рекомендуется добавить не менее 20) проблема `enq: HW` просто исчезнет из списка ожидающих обработки событий в этой базе данных, и приложения снова начнут работать с нормальной скоростью.

Добавление экстенгов вручную вполне подходит для временного устранения конкуренции при распределении ресурсов, тем не менее следует отметить, что этот способ не является постоянным надежным решением. Поскольку событие `enq: HW` при включении в очередь возникает, когда необходимо увеличить предельное значение для LOB-объекта быстрее, чем фоновый процесс сможет захватить и отформатировать новые фрагменты LOB, мы просто предварительно выполняем эту работу для фоновых процессов, распределяя новые экстенги вручную. Но в зависимости от того, какой объем пространства необходимо выделить, чтобы справиться с увеличивающейся рабочей нагрузкой, 30 или даже 50 экстенгов дадут приложению некоторое время для заполнения этого пространства, прежде чем снова возникнет проблема достижения предельного значения. Поэтому в зависимости от конкретной ситуации, от ожидаемого размера и продолжительности рабочей нагрузки для данного приложения распределение экстенгов вручную будет лишь временным решением рассматриваемой проблемы.

Кроме того, важно отметить, что поскольку подобная ситуация типична почти для всех языков описания данных (data definition languages – DDL), размещение экстенга вручную непременно потребует исключаящей блокировки соответствующего сегмента в базе данных, следовательно, в зависимости от уровня распараллеливания вы не сможете работать с этим сегментом в течение достаточно длительного интервала времени. Таким образом, удачным решением является перехват результатов подобного запроса в скрипте, обрабатывающем аварийные ситуации, который должен отслеживать эти события в базе данных и выдавать соответствующее предупреждение при опасности возникновения рассматриваемой ситуации.

Проблемы BASICFILE LOB: более эффективное решение

Ниже приводится краткое описание возможных решений для постоянного увеличения количества фрагментов LOB, выделяемых для сегмента LOB:

- перемещение сегмента LOB в табличное пространство с увеличением размера сегмента UNIFORM. Этот метод зарекомендовал себя как наиболее эффективный, так как большие экстенды UNIFORM предоставляют больше фрагментов для каждой HW-операции;
- увеличение размера фрагмента LOB-объекта. Сначала определяется средний размер данных LOB-объекта с помощью процедур DBMS_LOB, GETLENGTH, затем устанавливается размер фрагмента LOB-объекта в диапазоне от 120 до 150 процентов от среднего размера данных LOB;
- увеличение размера фрагмента при корректировке посредством настройки события 44951 TRACE NAME CONTEXT FOREVER, LEVEL 1024. При достижении предельного значения для LOB-объекта будет предпринята попытка коррекции пространства внутри сегмента сначала при помощи удаления старых образцов LOB с учетом значений параметров PCTVERSION и RETENTION. Настройка указанного события позволит увеличить количество фрагментов, которые Oracle будет корректировать в одной операции, то есть сделает ее более эффективной при обработке события enq: HW. Эта стратегия дополняет процедуру установки подходящего размера фрагмента LOB, описанную выше;
- необходимо убедиться в том, что был активирован механизм Automatic Segment Space Management (ASSM) для табличного пространства, в котором размещается проблемный LOB-объект. Кроме того, это является необходимым предварительным условием для LOB-объектов типа SECUREFILE.

В заключение важно отметить, что даже в СУБД Oracle Database Release 11.2.0.1 присутствовала весьма неприятная ошибка, которая могла повредить заголовок сегмента (то есть сделать весь сегмент некорректным) при прерывании ручной операции ALLOCATE EXTENT, особенно для сегментов LOB. Для получения более подробной информации об этой проблеме см. примечание My Oracle Support (MOS) Note 1229669.1 «Segment header corruption if extent allocation operation is interrupted».

СРАВНЕНИЕ LOB-ОБЪЕКТОВ BASICFILE С SECUREFILE LOB

В версии 11g СУБД Oracle появилась новая опция SECUREFILE для хранения данных типов LOB, предлагающая улучшенные функциональные возможности для управления типами данных LOB, включая устранение дублирования, шифрование и сжатие данных. Таким образом, настоятельно рекомендуется преобразовать LOB-объекты BASICFILE в LOB-объекты SECUREFILE при условии, что ваша СУБД обновлена с версии Oracle 9i или Oracle 10g до Oracle 11g. Сделав это, вы с большой вероятностью улучшите производительность и управляемость и в то же время устраните некоторые проблемы, описанные в предыдущем разделе. Но следует помнить, что переход к LOB-объектам SECUREFILE не обязательно должен быть всеобъемлющим, что демонстрируют приведенные ниже сценарии.

Одна вполне очевидная проблема может возникать, когда приложение в первый раз вставляет данные в новый LOB-объект, который был переведен в формат

SECUREFILE. Если у вас установлены версии 11.2.0.1 или 11.2.0.2, то могут возникать серьезные проблемы при вставке данных в поле LOB вне зависимости от того, установлен для него формат BASICFILE или SECUREFILE. (Отметим, что эта конкретная ошибка была исправлена в версии 11.2.0.3.)

Кроме того, важно определить, как параметры хранения LOB-объектов влияют на производительность. Ниже приведен краткий обзор наиболее важных параметров:

- параметр `CHUNK` определяет наименьшую единицу измерения `LOBSEGMENT` и всегда кратен значению параметра `DB_BLOCK_SIZE`. Если в базе данных размер `DB_BLOCK_SIZE` установлен равным 16 Кб, а размер данных, помещаемых в столбец LOB, составляет 2 Кб, то 14 Кб просто остаются неиспользованными. Поскольку максимальный размер фрагмента равен 32 Кб, важно определять параметр `CHUNK` так, чтобы пространство не расходовалось без пользы;
- директива `CACHE` сообщает Oracle о необходимости сохранять данные LOB в кэш-буфере базы данных, тогда как при установке значения `NOCACHE` данные LOB никогда не помещаются в кэш-буфер. В результате Oracle будет использовать прямые операции чтения/записи для некешированных данных (характеризуется ожиданием события прямых операций чтения/записи) или выполнять чтение/запись кэш-буфера базы данных для кешированных данных (характеризуется ожиданием события операции последовательного чтения из файла базы данных). Также отметим, что для данных LOB, которые только считываются, но никогда не записываются, существует третья директива `CACHE READS`, которая вносит данные LOB в кэш-буфер только при операциях чтения, но не при операциях записи;
- параметр `LOGGING` разрешает ведение записей об изменениях данных LOB в журналах отмены операций в режиме онлайн. Если нужно улучшить производительность языка обработки данных (data manipulation language – DML), работающего с данными LOB, и при этом известно, что для восстановления данных в ваших приложениях не требуется журналирование изменений данных LOB, то следует рассмотреть возможность установки параметра `NOLOGGING` для LOB. Отметим, что если определить параметр `NOLOGGING` для встроенных LOB-объектов, то любые изменения в данных этих объектов будут продолжать фиксироваться в журналах отмены операций в режиме онлайн.

Различия между новыми и старыми типами LOB

Для демонстрации различий между упомянутыми выше форматами LOB создадим две таблицы: одну с LOB-объектом BASICFILE, вторую с LOB-объектом SECUREFILE:

1. Создать таблицу `test1`, содержащую столбец с LOB-объектом BASICFILE. Помните, что если вы выполняете эти тесты в Oracle Database 12c, то необходимо определить параметр `BASICFILE`, чтобы использовать именно этот тип данных, так как в указанной версии (и в последующих) при создании LOB-объекта по умолчанию устанавливается формат SECUREFILE:

```
CREATE TABLE test1 (col1 CLOB,col2 number)
LOB(col1) STORE AS SECUREFILE(CACHE)
tablespace TS_GG_DATA;
```

После создания таблицы, содержащей сегмент LOB, в ней формируются два различных типа сегментов: LOBSEGMENT и LOBINDEX. По названиям можно понять, что LOBINDEX представляет собой индекс или указатель, используемый для доступа к страницам или фрагментам соответствующего LOBSEGMENT. Следующий запрос показывает характеристики сегментов:

```
SYS@ORCL AS SYSDBA> SELECT COUNT(*), segment_name, segment_type
FROM dba_extents
WHERE segment_name = 'TEST1'
GROUP BY segment_name, segment_type;
COUNT(*)  SEGMENT_NAME  SEGMENT_TYPE
-----
          24 TEST1          TABLE
```

2. Создать таблицу test2 с использованием формата SECUREFILE для LOB-объекта:

```
CREATE TABLE test2 (col1 CLOB,col2 number)
LOB(col1) STORE AS BASICFILE
tablespace TS_GG_DATA;
```

3. Выполнить следующий запрос для проверки таблиц и соответствующих типов данных LOB с использованием представления DBA_LOBS:

```
set lines 200
col column_name for a30
SELECT
  table_name,
  column_name,
  segment_name,
  securefile
FROM dba_lobs
WHERE table_name like 'TEST%';
TABLE_NAME          COLUMN_NAME          SEGMENT_NAME          SECURE
-----
TEST1                COL                SYS_LOB0000088862C00001$$ YES
TEST2                COL1               SYS_LOB0000088867C00001$$ NO
```

4. Использовать простой цикл в анонимном блоке Procedural Language/Structured Query Language (PL/SQL), запустить следующий тест с целью генерации различных произвольных значений с помощью операции конкатенации для вставки 1 миллиона строк в оба типа LOB-сегментов. Разумеется, перед выполнением этого теста необходимо убедиться в наличии достаточного места в табличных пространствах для размещения LOB-сегментов и соответствующей группы дисков, управляемой механизмом Automatic Storage Management (ASM):

```
SET TIME ON
SET TIMING ON
TRUNCATE TABLE test1;
TRUNCATE TABLE test2;
-- Load TEST1
TT@ORCL > BEGIN
  FOR v_Count_1 IN 1..1000000 LOOP
    INSERT INTO TEST1(col1) VALUES ('testInsertColLOBTest2'||v_count_1);
    commit;
```

```

        END LOOP;
    END;
/
PL/SQL procedure successfully completed.
Elapsed: 00:03:00.40
-- Load TEST2
TT@ORCL > BEGIN
    FOR v_Count_1 IN 1..1000000 LOOP
        INSERT INTO TEST2(col1) VALUES ('testInsertColLOBTest2'||v_count_1);
        commit;
    END LOOP;
    END;
/
22:00:55  2  22:00:55  3  22:00:55  4  22:00:55  5  22:00:55  6  22:00:55  7
PL/SQL procedure successfully completed.
Elapsed: 00:09:11.61

```

Проведенные тесты показывают, что вставка данных в SECUREFILE LOB-объект эффективнее более чем в три раза (180 с против 551 с), чем вставка тех же данных в SECUREFILE LOB-объект в базе данных Oracle 11g Release 2.

- Следующий тест позволяет убедиться в отсутствии существенного различия в производительности при обновлении записей для LOB-объектов BASICFILE и SECUREFILE:

```

TT@ORCL > BEGIN
    FOR v_Count_2 IN 1..1000 LOOP
        update TEST2 set col1 = 'testInsertColLOBTest2'||v_count_2 where rownum <100;
        commit;
    END LOOP;
    END;
/
PL/SQL procedure successfully completed.
Elapsed: 00:00:03.93
TT@ORCL > BEGIN
    FOR v_Count_2 IN 1..1000 LOOP
        update TEST1 set col1 = 'testInsertColLOBTest2'||v_count_2 where rownum <100;
        commit;
    END LOOP;
    END;
/
PL/SQL procedure successfully completed.
Elapsed: 00:00:03.62

```

- Тем не менее важно помнить, что после обновления и/или удаления данных сегмент LOB с большой вероятностью может стать сильно фрагментированным. Поэтому рекомендуется уплотнить (дефрагментировать) LOB с помощью следующей команды:

```

-- For Oracle Database 10.2 and above:
ALTER TABLE <table name>
MODIFY LOB (<lob column name>) (SHRINK SPACE [CASCADE]);

-- For Oracle Database 10.1 and below:
ALTER TABLE <table name>
MOVE LOB (<lob column name>) STORE AS (TABLESPACE <tablespace name>);

```

Преобразование BASICFILE LOB-объектов в SECUREFILE LOB-объекты

Преобразование BASICFILE LOB-объектов в SECUREFILE LOB-объекты можно выполнить с помощью одной простой команды:

```
TT@ORCL > ALTER TABLE test2 MOVE LOB (col1) STORE AS SECUREFILE (TABLESPACE users);
Table altered.
Elapsed: 00:00:23.54
```

Но у этого простого метода преобразования есть один недостаток – после завершения операции преобразования LOB-объект становится недоступным для любой операции языка обработки данных (DML). Другим способом преобразования LOB-объекта из формата BASICFILE в формат SECUREFILE в режиме ONLINE является использование пакета DBMS_REDEFINITION, демонстрируемое в следующем примере:

1. Если учетная запись пользователя, намеревающегося выполнить преобразование LOB-объекта, не обладает привилегиями SYSDBA, то необходимо выполнить следующую команду для временного предоставления данной учетной записи привилегий, достаточных для выполнения этого преобразования:

```
-- Create the migrating user (if it doesn't yet exist)...
grant dba to tt identified by tt123;

-- ... or grant the existing user account the necessary specific privileges
-- so it can use DBMS_REDEFINITION
grant execute on dbms_redefinition to tt;
grant alter any table to tt;
grant drop any table to tt;
grant lock any table to tt;
grant create any table to tt;
grant select any table to tt;
grant create session to tt;
```

2. Создать таблицу для данного примера. Необходимы таблица, которая будет преобразована (test3), и таблица, которая будет использоваться в процессе преобразования (test4), то есть временная или промежуточная таблица:

```
CREATE TABLE test3 (
col1 NUMBER PRIMARY KEY,
col2 CLOB
); 23:03:10  2  23:03:10  3  23:03:10  4
Table created.
Elapsed: 00:00:00.11
```

3. Для демонстрации процедуры преобразования вставить произвольные данные в таблицу test3:

```
TT@ORCL > BEGIN
FOR v_Count_2 IN 1..10000 LOOP
INSERT INTO TEST3 (col1,col2) VALUES (v_count_2,'testInsertColLOBTest3'||v_count_2);
commit;
END LOOP;
```

```

END;
/
PL/SQL procedure successfully completed.
Elapsed: 00:00:01.82

```

4. Создать промежуточную таблицу (test4), которая будет работать как таблица временного хранения данных, вставляемых, удаляемых или обновляемых в процессе выполнения преобразования в режиме онлайн:

```

TT@ORCL > CREATE TABLE test4 (
col1 NUMBER NOT NULL,
col2 CLOB
) LOB(col2) STORE AS SECUREFILE (NOCACHE);

Table created.

Elapsed: 00:00:00.05

```

5. Начать переопределение таблицы test3, используя процедуру START_REDEF_TABLE из пакета DBMS_REDEFINITION:

```

23:07:09 TT@ORCL > DECLARE
col_mapping VARCHAR2(1000);
BEGIN
col_mapping := 'col1 col1, '|| 'col2 col2';
DBMS_REDEFINITION.START_REDEF_TABLE('tt', 'test3', 'test4', col_mapping);
END;
/

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.97

```

6. Для уверенности в том, что все ограничивающие условия скопированы из исходной таблицы в промежуточную, нужно вызвать процедуру COPY_TABLE_DEPENDENTS из пакета DBMS_REDEFINITION, как показано ниже:

```

23:07:42 TT@ORCL > DECLARE
error_count pls_integer := 0;
BEGIN
DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
uname=> 'tt',
orig_table=> 'test3',
int_table=> 'test4',
copy_indexes=> DBMS_REDEFINITION.CONS_ORIG_PARAMS,
copy_triggers=> TRUE,
copy_constraints=> TRUE,
copy_privileges=> TRUE,
copy_statistics=> FALSE,
num_errors=> error_count);
DBMS_OUTPUT.PUT_LINE('errors := ' || TO_CHAR(error_count));
END;
/

PL/SQL procedure successfully completed.

Elapsed: 00:00:05.89

```

7. Завершить переопределение таблицы test3 в таблицу test4 с помощью процедуры FINISH_REDEF_TABLE из пакета DBMS_REDEFINITION:


```
23:08:10 TT@ORCL > EXEC DBMS_REDEFINITION.FINISH_REDEF_TABLE('tt', 'test3', 'test4');
PL/SQL procedure successfully completed.
Elapsed: 00:00:02.52
```

8. Убедиться в том, что столбец в таблице `test3`, который изначально являлся `BASICFILE LOB`-объектом, теперь имеет тип `SECUREFILE LOB`. Для этого выполнить запрос к таблице `DBA_LOBS`:

```
23:18:33 TT@ORCL > select owner,table_name,column_name,securefile
from dba_lobs where table_name='TEST3';
```

| OWNER | TABLE_NAME | COLUMN_NAME | SEC |
|-------|------------|-------------|-----|
| TT | TEST3 | COL2 | YES |

```
Elapsed: 00:00:00.05
23:18:53 TT@ORCL >
```

Другим способом выполнения массового эффективного преобразования в режиме онлайн `LOB`-объектов из формата `BASICFILE` в формат `SECUREFILE` является использование операции `CTAS` (`create table as as select`) в сочетании с инструментом Oracle GoldenGate для репликации данных в той же исходной и целевой базах данных. При этом имеется возможность синхронизации содержимого исходных и новых таблиц после операции `CTAS`, когда преобразование таблиц полностью завершено. После завершения преобразования приложение приостановит свою работу лишь на короткий интервал времени, в течение которого будут переименованы исходные и новые таблицы.

ВОЗДЕЙСТВИЕ ПАРАМЕТРА `PCTFREE` НА `LOB`-ОБЪЕКТЫ

Установка правильного значения для параметра `PCTFREE` чрезвычайно важна для того, чтобы избежать нерационального использования пространства при создании таблиц со столбцами, содержащими встроенные `LOB`-объекты. Параметр `PCTFREE` определяет минимальный процент свободного пространства в блоке, который СУБД Oracle резервирует при обновлении существующих строк в таблице, то есть главным образом для того, чтобы исключить ненужные перемещения строк, когда после обновления и увеличения длины фрагмента строки он не умещается в блоке.

По умолчанию для параметра `PCTFREE` устанавливается значение 10 процентов, но достаточно часто это значение увеличивают, чтобы предотвратить перемещения строк. Например, если для какой-либо таблицы установлено значение параметра `PCTFREE`, равное 30, то Oracle обязательно зарезервирует 30 процентов блока для обеспечения возможности увеличения длины строк при их обновлении. Если при вставке нового фрагмента строки в такой блок будет занято 70 процентов пространства, то этот блок помечается как заполненный, и следующая строка будет вставлена в другой блок с достаточным свободным пространством.

Но определение значения, большего, чем действительно необходимо, для параметра `PCTFREE` может привести к ситуации, в которой большие объемы свободного пространства просто никак не используются, когда таблица содержит столбцы со встроенными `LOB`-объектами. В следующем примере наглядно показано, как более рационально использовать табличное пространство в подобной ситуации:

1. Создать простую таблицу со столбцом, содержащим LOB-объект, и установить для параметра PCTFREE значение 40 процентов, вставить 100 000 строк в эту новую таблицу, затем проверить счетчик строк и собрать статистические данные для оптимизации:

```
SYS@ORCL AS SYSDBA> create table tt.test_pctfree (col1 clob) PCTFREE 40;
```

Table created.

```
SYS@ORCL AS SYSDBA> BEGIN
  FOR v_Count_2 IN 1..100000 LOOP
    INSERT INTO tt.test_pctfree
      VALUES ('testInsertColLOBTest2'||v_count_2);
    COMMIT;
  END LOOP;
END;
/
```

```
SYS@ORCL AS SYSDBA> select count(*) from tt.test_pctfree;
```

```
  COUNT(*)
-----
    100000
```

```
SYS@ORCL AS SYSDBA> EXEC DBMS_STATS.GATHER_TABLE_STATS ('TT', 'TEST_PCTFREE');
```

PL/SQL procedure successfully completed.

2. Проверить количество блоков и средний размер блока в этой таблице:

```
SYS@ORCL AS SYSDBA>
select
  blocks,
  avg_space,
  pct_free
from
  dba_tables
where
  table_name='TEST_PCTFREE';
```

```
BLOCKS      AVG_SPACE  PCT_FREE
-----
    1504      3362      40
```

3. Удалить все строки из таблицы и изменить для нее значение параметра PCTFREE на 0 (ноль):

```
SYS@ORCL AS SYSDBA> truncate table tt.test_pctfree;
```

Table truncated.

Elapsed: 00:00:00.07

```
SYS@ORCL AS SYSDBA> alter table tt.test_pctfree pctfree 0;
```

Table altered.

Elapsed: 00:00:00.01

4. Еще раз вставить то же количество строк и снова собрать статистические данные:

```

SYS@ORCL AS SYSDBA> BEGIN
  FOR v_Count_2 IN 1..100000 LOOP
    INSERT INTO tt.test_pctfree VALUES ('testInsertColLOBTest2'||v_count_2);
    COMMIT;
  END LOOP;
END;
/

PL/SQL procedure successfully completed.

Elapsed: 00:00:14.13

SYS@ORCL AS SYSDBA> EXEC DBMS_STATS.GATHER_TABLE_STATS ('TT', 'TEST_PCTFREE');

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.22
SYS@ORCL AS SYSDBA>

```

5. Проверить новые значения размеров блоков и размер пространства, которое в среднем занимает отдельный блок:

```

SYS@ORCL AS SYSDBA> select
blocks,
avg_space,
pct_free
from
dba_tables
where
table_name='TEST_PCTFREE';

```

| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|-----------|----------|---|---|---|---|
| BLOCKS | AVG_SPACE | PCT_FREE | | | | |
| 874 | 303 | 0 | | | | |

```

Elapsed: 00:00:00.00

```

Этот пример показывает, что количество занятых блоков существенно уменьшается: 874 при значении `PCTFREE`, равном 0 процентам, против 1504 при значении `PCTFREE`, равном 40 процентам, то есть эффективнее приблизительно на 41 процент. Использование больших значений для параметра `PCTFREE` в таблицах со столбцами LOB-объектов может приводить к бесполезной потере гигантского пространства, именно поэтому некоторые табличные пространства с LOB-объектами становятся непомерно огромными, что затрудняет их реорганизацию. Если в таблице, содержащей LOB-объекты, пространство пока еще не распределено, то администратор СУБД Oracle получает отличную возможность полностью контролировать запросы по выделению пространства для этой таблицы, так как в дальнейшем, вероятнее всего, практически невозможно будет найти время, достаточное для создания этой таблицы заново, даже при использовании пакета `DBMS_REDEFINITION`.

В особенности важно отметить, что процедура сокращения бесполезно расходуемого пространства в таблице с LOB-объектами применима только к встроенным LOB-объектам. В примере, приведенном выше, вставляемые данные имели относительно небольшой размер, и большинство этих данных сохранялось непосредственно в таблице, поэтому на общий размер таблицы оказывало существенное влияние значение параметра `PCTFREE`. Но для внешних LOB-объектов значение `PCTFREE` не является столь важным, потому что пространство LOB распределяется

и обслуживается с учетом фрагментов как единиц измерения (размер фрагментов составляет несколько блоков), а не на основе последовательного выделения блоков, как в сегменте таблицы.

Несмотря на то что типы данных LOB (BLOB, CLOB, NCLOB и BFILE) не используют параметра таблиц хранения `PCTFREE` и списков свободных блоков для управления свободным пространством, можно применить аналогичную стратегию для управления наращиванием пространства LOB-объекта, определяя размер фрагмента при работе с внешними LOB-объектами. Таким образом, следует установить значение размера фрагмента, превышающее средний размер данных LOB, если ожидаются обновления, увеличивающие объем данных. В противном случае возможно постоянное перемещение данных LOB, которое значительно снизит производительность приложения, поскольку для каждого запрошенного LOB-объекта операций извлечения будет выполнено вдвое больше, по сравнению с количеством фрагментов в этом LOB-объекте.

И последнее, но очень важное предупреждение, которое следует запомнить: размер фрагмента также влияет на размер данных в операции redo, генерируемых при обработке LOB-объектов средствами языка управления данными. Запись в операции redo обязательно должна быть сгенерирована для всего фрагмента в целом, поэтому размер фрагмента напрямую влияет на общий объем данных, генерируемых операцией redo, следовательно, определяет соответствующее воздействие на производительность языка управления данными. Таким образом, важно быть уверенным в том, что установленный размер фрагмента не превышает действительно необходимого значения.

УЛУЧШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ОПЕРАЦИИ ВСТАВКИ ДАННЫХ INSERT

Если в коде приложения неожиданно возникает проблема снижения производительности при выполнении операций вставки данных `INSERT` после преобразования LOB-объектов из формата `BASICFILE` в формат `SECUREFILE` в версии Oracle Database 11g Release 2, то следует знать, что с большой вероятностью это проявление серьезной ошибки в версии 11.2.0.1, которая описана в официальном документе MOS Note 1323933.1 «Securefiles performance appears slower than basicfiles LOB» («Производительность LOB-объектов в формате `securefile` снижается по сравнению с форматом `basicfile`»).

Чтобы проверить, что эта ошибка действительно является причиной снижения производительности операций `INSERT` (и быстро устранить эту проблему), выполните следующую команду, устанавливающую корректное значение параметра `_kdli_sio_fileopen` на уровне текущего сеанса работы перед продолжением выполнения команд `INSERT` в таблице, содержащей один или несколько столбцов с LOB-объектами в формате `SECUREFILE`:

```
SQL> alter session set "_kdli_sio_fileopen"='nodsync';
```

Документ MOS Note 1323933.1 настоятельно рекомендует применить соответствующее исправление для указанной версии СУБД, чтобы полностью устранить эту проблему. В заключение отметим, что в версии 11.2.0.3 эта ошибка исправлена.

РЕЗЮМЕ

Из этой главы вы узнали, что чем больше таблиц со столбцами LOB в базе данных, тем выше вероятность возникновения проблем, связанных с производительностью при обработке и сопровождении LOB-объектов. Необходимо всегда выполнять следующую рекомендацию: при создании новой таблицы со столбцами LOB попытайтесь использовать оптимальные параметры хранения, которые полностью соответствуют поведению соответствующей таблицы:

- для таблиц, которые никогда не будут обновляться, без сомнений устанавливайте для параметра `PCTFREE` значение «нуль» (0);
- если использование типа LOB не продиктовано действительной необходимостью, то вместо него следует рассмотреть возможность использования типов `VARCHAR` или `RAW`.