УДК 004.65

Гатилин Д.Н.
студент магистратуры
факультет «Информационных систем и технологий»
Поволжский государственный университет
телекоммуникаций и информатики
Россия, г. Самара
Научный руководитель: Козлов В.В., доцент,
кандидат технических наук

АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ ЗАПРОСОВ В БАЗЕ ДАННЫХ ORACLE: ИНСТРУМЕНТЫ И МЕТОДИКИ

Аннотация. Статья посвящена критически важной задаче анализа производительности запросов в СУБД Oracle. Рассматривается комплекс инструментов для диагностики проблем, включая AWR-отчеты, анализ планов выполнения и динамические представления. Предлагаются методики оптимизации, направленные на снижение времени выполнения и ресурсозатрат.

Ключевые слова: производительность запросов, СУБД, Oracle, оптимизация, план выполнения, мониторинг, статистика, AWR, EXPLAIN PLAN, индексы.

Gatilin D.N.
student of the Master's program of
Faculty of Information Systems and Technologies
Povolzhskiy State University of Telecommunications and Informatics
Russia, Samara
Supervisor: Kozlov V.V., Associate Professor
the candidate of technical sciences

QUERY PERFORMANCE ANALYSIS IN ORACLE DATABASE: TOOLS AND METHODS

Abstract. The article addresses the critical task of analyzing query performance in the Oracle DBMS. It examines a suite of diagnostic tools, including AWR reports, execution plan analysis, and dynamic performance views. The author proposes optimization methodologies aimed at reducing execution time and resource consumption.

Keywords: query performance, DBMS, Oracle, optimization, execution plan, monitoring, statistics, AWR, EXPLAIN PLAN, indexes.

Современные корпоративные информационные системы на базе Oracle характеризуются экспоненциальным ростом объемов данных и усложнением бизнес-логики, что предъявляет повышенные требования к производительности подсистем хранения и обработки данных. Медленное выполнение запросов в Oracle становится критической проблемой, непосредственно влияющей на отзывчивость приложений и удовлетворенность конечных пользователей.

Актуальность темы обусловлена возрастающей сложностью запросов в Oracle средах, увеличением количества одновременных подключений и требованиями к обработке данных в реальном времени. Эффективный анализ производительности в Oracle позволяет не только решать текущие проблемы, но и прогнозировать потенциальные узкие места в условиях роста нагрузки.

Процесс анализа производительности в Oracle представляет собой итеративный цикл, состоящий из последовательных этапов. Начальный этап предполагает выявление проблемных запросов через мониторинг ключевых метрик: времени выполнения, количества логических чтений и нагрузки на процессор.

Для идентификации проблемных запросов в Oracle применяются динамические представления V$SQL и V$SQLAREA, содержащие детальную статистику по выполненным запросам. Отчеты Automatic Workload Repository (AWR) позволяют выделить наиболее ресурсоемкие запросы за определенный период, а мониторинг активных сессий через V$SESSION и ASH (Active Session History) предоставляет информацию о текущей нагрузке [4][5].

Критически важным аспектом является анализ планов выполнения. План выполнения запроса в Oracle демонстрирует стратегию, выбранную оптимизатором, включая порядок соединения таблиц, используемые индексы и алгоритмы обработки данных. Для получения планов выполнения применяются команда EXPLAIN PLAN с последующим использованием пакета DBMS_XPLAN.DISPLAY, а также SQL Monitoring для мониторинга выполняющихся запросов в реальном времени [1].

Анализ планов выполнения позволяет выявить типичные проблемы производительности: полное сканирование таблиц при отсутствии подходящих индексов, неэффективные соединения, проблемы с операциями сортировки и расхождения в оценках кардинальности.

Oracle предоставляет богатый арсенал инструментов для комплексного анализа производительности. Центральное место занимает Automatic Workload Repository (AWR) — фоновый процесс, автоматически собирающий статистику производительности и создающий снапшоты системы. AWR-отчеты содержат детальную информацию о работе системы за выбранный период, включая топ ресурсоемких запросов, статистику ожиданий и рекомендации по оптимизации. Active Session History (ASH) дополняет AWR, предоставляя данные об активных сессиях с высокой частотой дискретизации. Это особенно ценно для анализа кратковременных проблем производительности, которые могут быть не видны в hourly AWR отчетах.

Для глубокого анализа выполнения отдельных запросов используются SQL Trace и TKPROF. Эти инструменты позволяют получить детальную трассировку выполнения запроса с информацией о времени выполнения каждой операции, количестве логических и физических чтений, а также других метриках производительности.

Oracle Enterprise Manager (OEM) предоставляет графический интерфейс для доступа ко всем перечисленным инструментам, делая процесс анализа более наглядным и удобным. OEM позволяет в реальном времени отслеживать состояние системы, анализировать выполняемые запросы и получать рекомендации по оптимизации.

План выполнения запроса в Oracle содержит ценную информацию для анализа производительности. Каждый этап выполнения запроса отображается в плане с указанием стоимости операции, ожидаемого количества строк и используемого метода доступа к данным.

Особое внимание при анализе планов следует уделять операциям с высокой стоимостью (cost). Операции полного сканирования таблиц (FULL TABLE SCAN) часто указывают на отсутствие подходящих индексов. Однако в некоторых случаях, при выборке большого процента данных из таблицы, полное сканирование может быть более эффективным, чем индексный доступ.

Анализ операций соединения требует понимания различий между Nested Loops, Hash Join и Sort Merge Join. Nested Loops эффективны при соединении небольшого набора данных, в то время как Hash Join лучше подходит для больших объемов данных. Sort Merge Join обычно применяется при наличии индексов для сортировки.

Важным аспектом является анализ операций сортировки (SORT ORDER BY, SORT GROUP BY). Большие операции сортировки могут потребовать значительных ресурсов временного tablespace. Наличие

индексов, обеспечивающих нужный порядок данных, может полностью исключить необходимость сортировки.

Статистика в Oracle играет ключевую роль в процессе оптимизации запросов. Оптимизатор использует статистику для оценки селективности условий и выбора оптимального плана выполнения. Неактуальная статистика может привести к выбору неэффективных планов. Сбор статистики осуществляется с помощью пакета DBMS_STATS. Рекомендуется использовать автоматический сбор статистики, однако в некоторых случаях может потребоваться ручной сбор с определенными параметрами. Особое внимание следует уделять сбору статистики для таблиц с быстро меняющимися данными. Динамические представления производительности (V$ представления) предоставляют доступ к реальным метрикам работы системы. V$SQL содержит информацию о выполненных запросах, включая время выполнения, количество logical reads и physical reads. V$SESSION предоставляет данные о текущих сессиях, а V$SYSTEM_EVENT — о событиях ожидания в системе.

Среди наиболее распространенных проблем производительности в Oracle выделяются проблемы с индексацией. Отсутствие подходящих индексов приводит к полному сканированию таблиц, в то время как избыточная индексация замедляет операции DML. Регулярный мониторинг использования индексов через V$OBJECT_USAGE и создание индексов на основе анализа реальных workload-ов являются ключевыми мероприятиями по оптимизации.

Проблемы с оптимизатором часто связаны с устаревшей статистикой. Неверные оценки кардинальности могут приводить к выбору неоптимальных планов выполнения. Регулярное обновление статистики с помощью DBMS_STATS и использование гистограмм для колонок с неравномерным распределением данных помогают улучшить качество решений оптимизатора.

Особого внимания заслуживают проблемы с доступом к данным. Высокий уровень физического ввода-вывода может указывать на недостаточный размер буферного кэша, в то время как конкуренция за ресурсы проявляется в виде блокировок и latch contention. Анализ статистики ожиданий через V$SYSTEM_EVENT и V$SESSION_EVENT помогает идентифицировать эти проблемы.

Эффективное управление памятью является выжным аспектом производительности Oracle. System Global Area (SGA) содержит буферный кэш, shared pool и другие компоненты, оказывающие непосредственное влияние на производительность запросов. Буферный кэш (Buffer Cache) хранит копии блоков данных, прочитанных с диска. Высокий процент попаданий в буферный кэш свидетельствует об эффективном использовании памяти. Однако следует учитывать, что сам по себе высокий hit ratio не всегда гарантирует хорошую производительность. Shared Pool содержит shared cursors, stored procedures и другие shared структур. Недостаточный размер Shared Pool может приводить к частому переразбору запросов, что значительно снижает производительность. Использование bind variables помогает уменьшить нагрузку на Shared Pool.

Program Global Area (PGA) используется для операций сортировки и хранения временных данных. Недостаточный размер PGA может приводить к операции сортировки на диск, что значительно медленнее сортировки в памяти. Настройка параметров PGA_AGGREGATE_TARGET и WORKAREA_SIZE_POLICY позволяет оптимизировать использование PGA.

Oracle предоставляет мощные инструменты для мониторинга производительности в реальном времени. Инструмент мониторинга SQL (SQL Monitoring) обеспечивает контроль за длительно выполняющимися запросами, давая информацию о прогрессе, потреблении ресурсов и потенциальных узких местах.

Active Session History (ASH) буферизует информацию об активных сессиях каждую секунду, что позволяет анализировать кратковременные проблемы производительности. ASH особенно полезна для диагностики внезапных замедлений работы системы.

База данных производительности (AWR) автоматически собирает и сохраняет исторические данные о производительности. Сравнение AWR отчетов за разные периоды позволяет выявить тенденции изменения производительности и вовремя обнаружить деградацию. Automatic Database Diagnostic Monitor (ADDM) анализирует данные AWR и предоставляет экспертные рекомендации по устранению проблем производительности [3]. ADDM автоматически идентифицирует root cause проблем и предлагает конкретные действия по их устранению.

Для комплексных запросов может потребоваться использование продвинутых методов оптимизации. SQL Profile позволяет зафиксировать оптимальный план выполнения для конкретного запроса, предотвращая его деградацию в будущем.

SQL Plan Management (SPM) предоставляет механизм для контроля над планами выполнения запросов. SPM автоматически и верифицирует планы выполнения, предотвращая использование неоптимальных планов.

Для обработки больших объемов данных эффективно использование параллельного выполнения запросов. Однако использование параллелизма в выполнении запросов предполагает детальную конфигурацию и рекомендуется только для операций, требующих значительных вычислительных мощностей.

Materialized Views могут значительно ускорить выполнение комплексных запросов с агрегациями и соединениями многих таблиц. Функция автоматической перезаписи запросов перенаправляет выполнение запросов на материализованные представления при наличии такой возможности [2].

Эффективная работа с производительностью в Oracle требует систематического подхода. Регулярный анализ AWR-отчетов должен стать стандартной практикой для выявления тенденций и потенциальных проблем. Настройка алертов на критические события производительности позволяет оперативно реагировать на возникающие проблемы. При работе с индексами рекомендуется создавать составные индексы с правильным порядком колонок и использовать функциональные индексы для запросов с функциями в условиях. Важно балансировать между преимуществами индексов для операций SELECT и их накладными расходами для операций DML.

Для настройки оптимизатора следует не только регулярно обновлять статистику, но и анализировать качество генерируемых планов выполнения. В сложных случаях может потребоваться использование SQL Profile для фиксации оптимальных планов выполнения.

Далее будет представлен детальный разбор кейса по оптимизации с количественными показателями эффективности. Основными показателями для оценки эффективности запросов в приведенных ниже кейсе являются: общее время выполнения, количество операций чтения из буферного кэша и с дисковых накопителей, затраты процессорного времени и размер используемых временных сегментов.

Кейс: «Оптимизация медленного запроса при формировании отчета по продажам».

В системе отчетности ежедневно формируется отчет по продажам за последние 30 дней. Время выполнения превысило 25 минут.

Исходные данные:

- таблица SALES: 50 млн. записей;
- таблица CUSTOMERS: 500 тыс. записей;
- таблица PRODUCTS: 10 тыс. записей;

Для сбора статистики выполнения и диагностики проблем использован комплекс инструментов мониторинга: AWR, ASH, SQL Monitoring и Segment Statistics.

Метрики производительности до оптимизации сведены в таблицу ключевых показателей – таблица 1.

Таблица 1. Ключевые метрики производительности

Метрика	**Значение**	**Влияние**
Время выполнения отчета	25 минут	Критическое
Логические чтения (Buffer Gets)	2,850,000	Чрезмерные логические чтения
Физические чтения (Physical Reads)	1,250,400	Высокая I/O нагрузка
Временное табличное пространство (Temp Space)	850 MB	Большие временные операции
Стоимость в плане (Cost)	125,000	Высокая стоимость

В ходе анализа AWR найден топ запрос по времени выполнения за последние сутки, рисунок 1.

```
SELECT
    c.customer_name,
    p.product_name,
    SUM(s.quantity) as total_quantity,
    SUM(s.amount) as total_amount,
    COUNT(s.sale_id) as number_of_transactions
FROM
    sales s
INNER JOIN
    customers c ON s.customer_id = c.customer_id
INNER JOIN
    products p ON s.product_id = p.product_id
WHERE
    s.sale_date >= TRUNC(SYSDATE) - 30
GROUP BY
    c.customer_name, p.product_name
ORDER BY
    total_amount DESC;
```

Рисунок 1. Топ запрос по времени выполнения за последние сутки

Построим план выполнения проблемного запроса, рисунок 2.

```
Plan hash value: 2966233522

-------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name      | Rows  | Bytes | Cost (%CPU)| Time     | Buffers   |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |           |       |       |   125K(100)|          |   2850K   |
|   1 |  SORT ORDER BY       |           |   456K|    78M|   125K  (2)| 00:00:05 |   2850K   |
|   2 |   HASH GROUP BY      |           |   456K|    78M|   125K  (2)| 00:00:05 |   2850K   |
|*  3 |    HASH JOIN         |           |   456K|    78M|    98K  (2)| 00:00:04 |   2850K   |
|*  4 |     HASH JOIN        |           |   456K|    58M|    55K  (3)| 00:00:03 |   1425K   |
|*  5 |      TABLE ACCESS FULL| SALES     |   456K|    26M|    52K  (2)| 00:00:02 |   1250K   |
|   6 |      TABLE ACCESS FULL| CUSTOMERS |   500K|    33M|  1467  (3)| 00:00:01 |    175K   |
|   7 |     TABLE ACCESS FULL | PRODUCTS | 10000 |   253K|    12  (0)| 00:00:01 |     500   |
-------------------------------------------------------------------------------------------------------

Query Block Name / Object Alias:
-------------------------------------------------
    SEL$1            / "S"@"SEL$1"
    SEL$1            / "C"@"SEL$1"
    SEL$1            / "P"@"SEL$1"

Predicate Information (identified by operation id):
-------------------------------------------------
   3 - access("S"."PRODUCT_ID"="P"."PRODUCT_ID")
   4 - access("S"."CUSTOMER_ID"="C"."CUSTOMER_ID")
   5 - filter("S"."SALE_DATE">=TRUNC(SYSDATE@!)-30)

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - 2850000 buffer gets
   - 1250400 physical reads
   - 850MB temp space used
```

Рисунок 2. План выполнения проблемного запроса

Анализ текущего плана выполнения выявил ряд серьезных проблем производительности, которые систематизированы в таблице 2.

Таблица 2. Ключевые проблемы

Проблема	**Описание проблемы**	**Серьезность**
TABLE ACCESS FULL на таблице SALES	полное сканирование таблицы на 50 млн. записей; самая дорогая операция (Cost = 52K); фильтр применяется только после чтения всех данных; 1250K - buffer gets на одной операции	Критическая
Неправильная оценка кардинальности	оптимизатор недооценивает объем промежуточных данных; HASH JOIN обрабатывает 456K строк вместо ожидаемых меньших объемов; группировка работает с избыточными данными	Средняя
Неоптимальные HASH JOIN	HASH JOIN эффективен для больших таблиц, но здесь одна таблица маленькая (PRODUCTS - 10K строк); создаются хэш-таблицы в памяти/TEMP для 456K строк; высокие затраты памяти и временного пространства	Высокая
Большой объем временных данных	HASH JOIN требуют много памяти; SORT ORDER BY и HASH GROUP BY	Высокая

	используют временное пространство; При нехватке памяти - операции идут на диск (slow)	

Для повышения производительности запроса был реализован комплекс оптимизационных мероприятий, включающий:

1. Реинжиниринг схемы индексации:
   - внедрение составного покрывающего индекса, оптимизированного для предикатов фильтрации и операций соединения;
   - применение компрессии индекса для сокращения объема хранимых данных и снижения нагрузки на подсистему ввода-вывода;
   - оптимизация последовательности столбцов в индексе в соответствии с селективностью и частотой использования в запросах

На рисунке 3 представлена команда создания составного индекса.

```
CREATE INDEX idx_sales_date_cover
ON sales(sale_date, customer_id, product_id, quantity, amount)
COMPRESS 2;
```

Рисунок 3. Создание составного индекса

2. Рефакторинг SQL запроса:
   - реструктуризация логики выполнения с применением Common Table Expressions (CTE) для реализации стратегии ранней агрегации;
   - модификация методов соединения таблиц с заменой ресурсоемких HASH JOIN на оптимальные NESTED LOOPS для работы с редуцированными наборами данных;

- внедрение директив оптимизатора (hints) для целенаправленного управления выбором планов выполнения

На рисунке 4 представлен оптимизированный SQL запрос.

```
WITH sales_agg AS (
    SELECT /*+ INDEX(s idx_sales_date_cover) */
        s.customer_id,
        s.product_id,
        SUM(s.quantity) as total_quantity,
        SUM(s.amount) as total_amount,
        COUNT(s.sale_id) as transaction_count
    FROM sales s
    WHERE s.sale_date >= TRUNC(SYSDATE) - 30
    GROUP BY s.customer_id, s.product_id
)
SELECT /*+ USE_NL(c) USE_NL(p) */
    c.customer_name,
    p.product_name,
    sa.total_quantity,
    sa.total_amount,
    sa.transaction_count as number_of_transactions
FROM sales_agg sa
JOIN customers c ON sa.customer_id = c.customer_id
JOIN products p ON sa.product_id = p.product_id
ORDER BY sa.total_amount DESC;
```

Рисунок 4. Рефакторинг SQL запроса

3. Оптимизация статистического обеспечения:
   - регулярный сбор детализированной статистики объектов базы данных с использованием адаптивных методов выборки;
   - настройка параметров оптимизатора для корректной оценки кардинальности и селективности предикатов;
   - внедрение мониторинга гистограмм распределения данных для критически важных столбцов

На рисунке 5 представлен процесс сбора статистики с использованием процедуры DBMS_STATS.GATHER_TABLE_STATS.

```
BEGIN
    DBMS_STATS.GATHER_TABLE_STATS(
        ownname          => 'CRM',
        tabname          => 'SALES',
        estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,
        method_opt       => 'FOR ALL COLUMNS SIZE SKEWONLY',
        degree           => 8,
        cascade          => TRUE,
        no_invalidate    => FALSE
    );
END;
```

Рисунок 5. Оптимизация статистического обеспечения

В результате реализации комплекса оптимизационных мероприятий достигнуто значительное улучшение метрик производительности, что отражено в плане выполнения оптимизированного запроса на рисунке 5 и в сравнительной таблице 3:

```
-------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name                | Rows  | Bytes | Cost  |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                     |  4800 |   825K|  2845 |
|   1 |  SORT ORDER BY                 |                     |  4800 |   825K|  2845 |
|   2 |   NESTED LOOPS                 |                     |  4800 |   825K|  2843 |
|   3 |    NESTED LOOPS                |                     |  4800 |   825K|  2841 |
|   4 |     VIEW                       |                     |  4800 |   576K|  2815 |
|   5 |      HASH GROUP BY             |                     |  4800 |   576K|  2815 |
|   6 |       INDEX RANGE SCAN         | IDX_SALES_DATE_COVER| 4800 |   576K|  2812 |
|   7 |     INDEX UNIQUE SCAN          | PK_CUSTOMERS        |     1 |    13 |     1 |
|   8 |    TABLE ACCESS BY INDEX ROWID | PRODUCTS            |     1 |    25 |     1 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   6 - access("S"."SALE_DATE">=TRUNC(SYSDATE@!)-30)
   7 - access("SA"."CUSTOMER_ID"="C"."CUSTOMER_ID")
   8 - filter("SA"."PRODUCT_ID"="P"."PRODUCT_ID")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - 31200 buffer gets
   - 2850 physical reads
   - 0 temp space used
```

Рисунок 5. План выполнения оптимизированного запроса

Таблица 3. Сравнительные результаты оптимизации

Метрика	**До оптимизации**	**После оптимизации**	**Улучшение**
Время выполнения отчета	25 минут	11 секунд	в 136 раз
Логические чтения (Buffer Gets)	2,850,000	31,200	в 91 раз
Физические чтения	1,250,400	2,850	в 439 раз

(Physical Reads)			
Временное табличное пространство (Temp Space)	850 MB	0 MB	100% снижение
Стоимость в плане (Cost)	125,000	2,845	в 44 раза

Эффективный анализ производительности запросов в Oracle требует комплексного подхода и глубокого понимания архитектуры этой СУБД. Инструменты Oracle предоставляют богатые возможности для диагностики и решения проблем производительности, однако их эффективное использование требует систематического подхода и глубоких знаний.

Ключевыми факторами успеха являются регулярный мониторинг с использованием AWR, ASH и динамических представлений, глубокий анализ планов выполнения, систематическая работа со статистикой и индексами, а также понимание архитектуры Oracle и влияния параметров на производительность.

Комплексное применение инструментов и методик Oracle позволяет не только оперативно решать текущие проблемы производительности, но и проактивно предотвращать их возникновение, обеспечивая стабильную и эффективную работу корпоративных информационных систем.

Использованные источники:

1. Oracle Database Performance Tuning Guide [Электронный ресурс] - URL: https://docs.oracle.com/en/database/oracle/oracle-database/19/tgsql/generating-and-displaying-execution-plans.html (дата обращения: 20.11.2025)
2. Применение Materialized Views в организации ETL-процессов [Электронный ресурс] - URL: https://habr.com/ru/companies/pgk/articles/591607/ (дата обращения: 20.11.2025)

3. Знакомство с Automatic Workload Repository [Электронный ресурс] - URL: https://www.interface.ru/home.asp?artId=22589 (дата обращения: 20.11.2025)
4. Real-Time Database Operation Monitoring в Oracle Database 12c [Электронный ресурс] - URL: http://www.igormelnikov.com/2016/04/real-time-database-operation-monitoring.html (дата обращения: 20.11.2025)
5. Active Session History (ASH): анализ нагрузки и решение проблем производительности БД Oracle [Электронный ресурс] - URL: https://www.fors.ru/upload/ppt/2022-06-02/02062022_ASH_Bazgutdinov_ FORS.pdf (дата обращения: 20.11.2025)