Проверка целостности данных в PostgreSQL на уровне таблицы, строки и ячейки

01

Разработчик:

Донской В.Д.,
студент группы ПИ-22

Руководитель:

Целебровский О.Б.,
младший разработчик ПО PostgresPro

Актуальность работы

02

Широкое использование PostgreSQL

СУБД является стандартом для многих информационных систем

Ограниченность механизмов

Встроенные средства (WAL, page_checksums) работают преимущественно на физическом уровне

Проблема логической целостности

Отсутствие инструментов для проверки неизменности логического содержимого после реорганизации данных

Многоуровневый контроль

Необходимость поддержки целостности данных на разных уровнях абстракции: от ячейки до всей базы данных

Цель работы

Разработка расширения для PostgreSQL, обеспечивающего вычисление контрольных сумм на различных уровнях иерархии данных

Подсчёт контрольной суммы должен быть реализован на следующих уровнях:

- **Ячейка**
- **Строка**
- **Таблица**
- **Индекс**
- **База данных**

Ключевая особенность расширения: поддержка физического и логического режимов проверки для обеспечения согласованности данных

Задачи работы

1. **Анализ механизмов**
   Изучение существующих средств обеспечения целостности в PostgreSQL

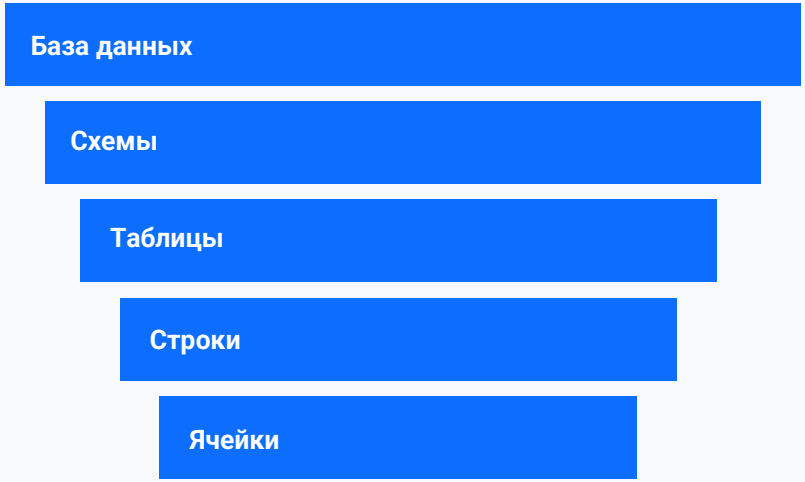
2. **Проектирование архитектуры**
   Разработка структуры расширения для проверки данных на всех уровнях иерархии

3. **Выбор алгоритма**
   Выбор оптимального алгоритма хеширования

4. **Реализация**
   Разработка функций подсчёта контрольной суммы для PostgreSQL

5. **Тестирование**
   Проверка корректности работы расширения

Иерархия данных и существующие механизмы поддержки целостности



ВСТРОЕННЫЕ СРЕДСТВА

WAL, Page Checksums, pg_amcheck: полезны для физического уровня проверки целостности данных, но не решают задачу логической сверки содержимого

Ограничение: эти инструменты не решают задачу логической проверки данных

ВНЕШНИЕ СРЕДСТВА

pg_dump: сравнение выгрузок данных

Сторонние инструменты: DBeaver Data Compare, Google Cloud Data Validation Tool

Ограничение: требуют полного сравнения данных и плохо подходят для больших объёмов

Физические и логические контрольные суммы

Тип суммы	От чего зависит	Поведение при реорганизации
Физическая	Зависит от расположения данных, заголовков страниц и строк	**Изменяется** при операциях VACUUM FULL, CLUSTER или REINDEX
Логическая	Зависит исключительно от содержимого данных (значений)	Остается **неизменной**, если сами значения не менялись

Выбор алгоритма

FNV-1a

07

- **Детерминированность**

  Одинаковые входные данные всегда дают идентичный результат хеширования

- **Высокая скорость**

  Эффективная обработка данных с минимальными задержками

- **Простота реализации**

  Легкая интеграция в расширение без внешних зависимостей

- **Поддержка seed-значений**

  Удобно для построения иерархических контрольных сумм разных уровней

- **Стандарт PostgreSQL**

  Уже используется в ядре СУБД для вычисления контрольных сумм страниц

Язык разработки

C

Реализация расширения

08

Прямой доступ к API

Использование внутренних структур PostgreSQL для работы с буферным кешем, страницами и строками

Системные каталоги

Взаимодействие с метаданными СУБД для корректного определения структуры таблиц и индексов

Производительность

Низкоуровневая реализация обеспечивает минимальные накладные расходы при вычислении хешей

Функциональные особенности

09

Уровень строки

Для вычисления логической суммы обязательно наличие **первичного ключа (PK)**. При его отсутствии функция возвращает NULL.

Агрегация данных

На уровнях таблицы, индекса и базы данных отдельные хеши собираются в единую структуру для итогового вычисления.

Детерминированный обход

Хеши **сортируются** перед финальным расчетом, что гарантирует независимость результата от физического порядка строк.

Универсальность

Корректная обработка всех встроенных типов данных PostgreSQL.

Тестирование расширения

Функциональные тесты

Проверка корректности вычислений для всех уровней, обработка NULL и разных типов данных.

Граничные случаи

Тестирование поведения на пустых таблицах, индексах и базах данных.

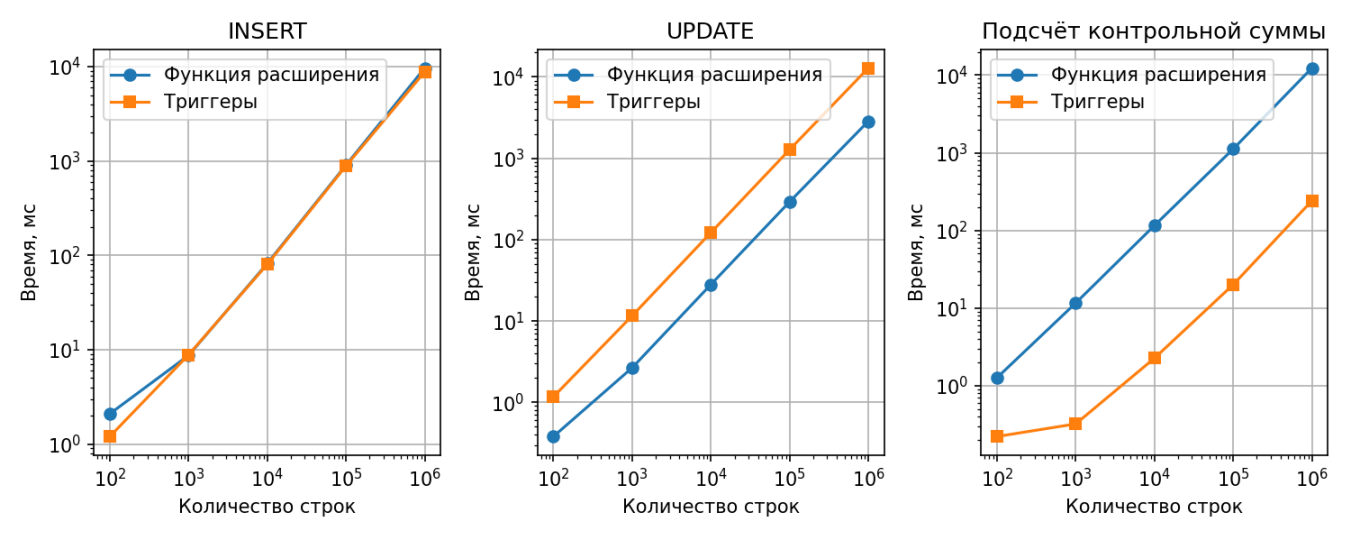
Тест на стабильность

Проверка неизменности логических сумм после VACUUM FULL, CLUSTER и REINDEX.

```
victor@VIKTOR:~/pg_checksums$ make USE_PGXS=1 installcheck
echo "# +++ regress install-check in  +++" && /home/victor/postgres-custom/lib/pgxs/src/makefiles/../../src/test/regress
/pg_regress --inputdir=./ --bindir='/home/victor/postgres-custom/bin'    --inputdir=./ --load-extension=pg_checksums --d
bname=contrib_regression checksum_basic checksum_database checksum_index checksum_table checksum_stability
# +++ regress install-check in  +++
# using postmaster on Unix socket, default port
2026-05-22 15:34:06.805 +07 [1014] LOG:  checkpoint starting: fast force wait
2026-05-22 15:34:06.814 +07 [1014] LOG:  checkpoint complete: wrote 4 buffers (0.0%), wrote 0 SLRU buffers; 0 WAL file(s
) added, 0 removed, 0 recycled; write=0.001 s, sync=0.002 s, total=0.010 s; sync files=4, longest=0.002 s, average=0.001
 s; distance=1 kB, estimate=1 kB; lsn=0/4CFE2D38, redo lsn=0/4CFE2CE0
ok 1         - checksum_basic                             27 ms
ok 2         - checksum_database                          56 ms
ok 3         - checksum_index                             45 ms
ok 4         - checksum_table                             40 ms
ok 5         - checksum_stability                         40 ms
1..5
# All 5 tests passed.
echo "# +++ tap install-check in  +++" && rm -rf '/home/victor/pg_checksums'/tmp_check && /usr/bin/mkdir -p '/home/victo
r/pg_checksums'/tmp_check && cd ./ && TESTLOGDIR='/home/victor/pg_checksums/tmp_check/log' TESTDATADIR='/home/victor/pg_
checksums/tmp_check' PATH="/home/victor/postgres-custom/bin:/home/victor/pg_checksums:$PATH" PGPORT='65432' PG_REGRESS='
/home/victor/postgres-custom/lib/pgxs/src/makefiles/../../src/test/regress/pg_regress' /usr/bin/prove -I /home/victor/po
stgres-custom/lib/pgxs/src/makefiles/../../src/test/perl/ -I ./  t/*.pl
# +++ tap install-check in  +++
t/001_base.pl ..... ok
t/002_advanced.pl .. ok
All tests successful.
Files=2, Tests=106,  3 wallclock secs ( 0.03 usr  0.00 sys +  1.04 cusr  1.09 csys =  2.16 CPU)
Result: PASS
```

Влияние на производительность

Триггерный подход пересчитывает контрольную сумму при любой модификации данных и сохраняет в служебную таблицу.
Разработанное расширение вычисляет контрольные суммы по запросу пользователя.



Вывод: расширение обеспечивает значительно лучшую производительность при частых изменениях данных

Результаты работы

- Разработано расширение **pg_checksums** для многоуровневой проверки целостности данных
- Реализована поддержка физического и логического режимов вычисления сумм
- Подтверждена корректность работы через автоматизированное тестирование

Практическая значимость расширения

- Мониторинг целостности данных
- Проверка согласованности реплик
- Верификация после миграций
- Аудит изменений

Проверка целостности данных в PostgreSQL на уровне таблицы, строки и ячейки

13

Разработчик:

Донской В.Д.,
студент группы ПИ-22

Руководитель:

Целебровский О.Б.,
младший разработчик ПО PostgresPro