

VMware Tanzu Greenplum 7

Greenplum 7 Feature Update

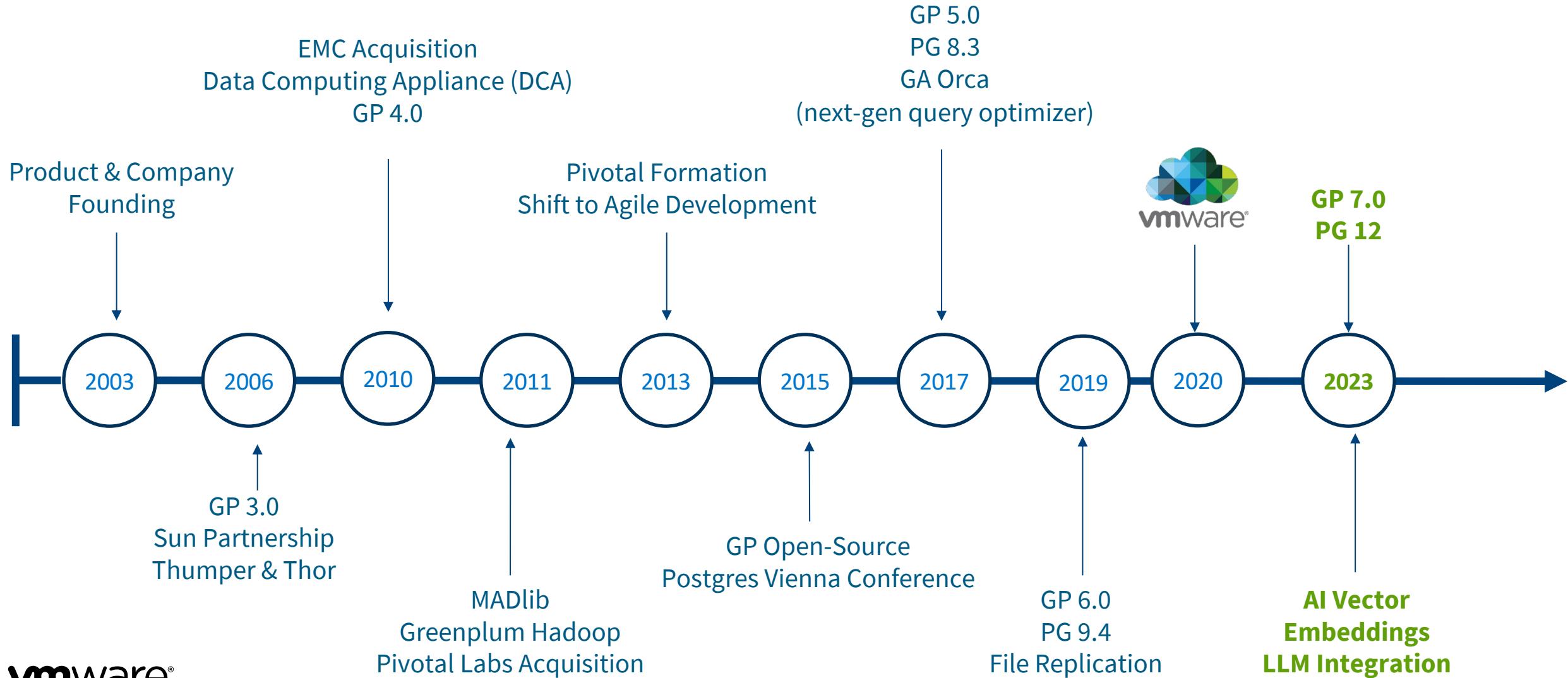
Sanghee Lee
Tanzu Data Service

2024.07.08

Greenplum 7 개선 사항

개선 요약 및 변경 사항

Greenplum History & Top-Level Timeline



Greenplum 7 개선 요약

구 분	내 용	비 고
코어 엔진	PostgreSQL 12 <ul style="list-style-type: none"> ▪ PostgreSQL 9.6 → PostgreSQL 12 	<ul style="list-style-type: none"> ▪ Core 엔진 개선
	GP Extension <ul style="list-style-type: none"> ▪ Greenplum FDW, PostgresML, pgaudit, pl/python3, pgvector 	<ul style="list-style-type: none"> ▪ postgres의 확장 패키지 추가
통계 수집 개선	멀티컬럼 통계 (MCV) <ul style="list-style-type: none"> ▪ 멀티 컬럼의 통계 수집 지원 	<ul style="list-style-type: none"> ▪ CREATE STATISTICS s1 (dependencies) ON a, b FROM t1; Analyze t1;
	빠른 Analyze <ul style="list-style-type: none"> ▪ 압축 테이블에 Analyze 성능 개선 	
	카탈로그 테이블 자동 수집 <ul style="list-style-type: none"> ▪ 카탈로그 데이터에 대해서 자동으로 vacuum 및 analyze 수행 	<ul style="list-style-type: none"> ▪ 카탈로그 테이블 auto vacuum/auto analyze 수행 ▪ 사용자 테이블에 대해서는 개별 관리 필요
성능 개선	JIT 컴파일 <ul style="list-style-type: none"> ▪ 쿼리 수행시점에 쿼리 플랜을 기계어로 컴파일 후 쿼리 수행 ▪ Just in time 컴파일 (PG11) 	<ul style="list-style-type: none"> ▪ 컴파일 수행 시간: ms 단위 필요 ▪ 장시간 소요되는 쿼리 성능 개선
	정렬 속도 개선 <ul style="list-style-type: none"> ▪ varchar, text, numeric 컬럼 정렬 시 속도 개선 	<ul style="list-style-type: none"> ▪ OLAP성 쿼리 성능 개선
	Index only scan <ul style="list-style-type: none"> ▪ Heap 테이블에 Index only scan 지원 ▪ Covering index 지원 	<ul style="list-style-type: none"> ▪ Key값이 아닌 컬럼 데이터를 인덱스에 포함시켜 인덱스만으로 데이터까지 함께 조회
	BRIN/Hash 인덱스 지원 <ul style="list-style-type: none"> ▪ 블록 범위 인덱스 지원 ▪ Hash 인덱스 지원 	<ul style="list-style-type: none"> ▪ 기존 하위버전에서는 기능 제공되지 않음.
	워크로드 관리 <ul style="list-style-type: none"> ▪ Short query Bypass 기능 추가 ▪ Disk IO 사용량에 따른 리소스 제한 기능 추가 	<ul style="list-style-type: none"> ▪ Resource Group v2 지원

Greenplum 7 개선 요약

구 분	내 용	비 고
개발 생산성 향상	프로시저 트랜잭션 지원	<ul style="list-style-type: none">프로시저내 Commit, Rollback 등의 트랜잭션 기능 제공
	Upsert 지원	<ul style="list-style-type: none">키 값이 충돌 될 경우 기존 데이터에 업데이트 수행
	제너레이티드 컬럼	<ul style="list-style-type: none">컬럼을 활용해서 자동 생성되는 컬럼데이터 적재시 저장하여, 조회시 저장된 데이터 추출
	GP Python	<ul style="list-style-type: none">Python로 Greenplum 데이터 핸들링을 용이하게 네이티브 API 제공
관리 개선	효율적인 스키마 변경	<ul style="list-style-type: none">Alter Table 구문으로 스토리지 옵션 변경 및 컬럼 추가
		<ul style="list-style-type: none">테이블에 컬럼 추가시 카탈로그에만 적용
	압축 테이블 PK/UK 지원	<ul style="list-style-type: none">압축 테이블에 Primary Key, Unique Key 지원
	파티션 관리 개선	<ul style="list-style-type: none">Postgres와 호환 가능한 파티션 관리 기능 제공
	운영 명령어 모니터링 지원	<ul style="list-style-type: none">Analyze, copy, vacuum, index 생성시 진행상황 모니터링 지원
		<ul style="list-style-type: none">gp_stat_progress_xxx view 참조

Greenplum 7 개선 요약

구 분	내 용		비 고
AI 기반 의미 검색	pgvector	<ul style="list-style-type: none">▪ 관련성이 높은 검색 결과 제공▪ 의미검색, 다국어 검색, 이미지/비디오 검색, 시간/지리/그래프 검색	<ul style="list-style-type: none">▪ 기계 언어로 생성된 임베딩 저장, 검색, 쿼리 지원
보안 강화	Row 레벨 보안	<ul style="list-style-type: none">▪ 테이블의 Row 레벨 보안 적용	<ul style="list-style-type: none">▪ 기존 하위버전에서는 기능 제공되지 않음.
	감사 기능 제공 pgaudit	<ul style="list-style-type: none">▪ Greenplum Database의 auditing 기능 제공▪ 접속/연결 정보, 오브젝트 생성/수정/삭제, 쿼리 텍스트, 세션 정보	<ul style="list-style-type: none">▪ DB 작업의 감사 로그 생성 저장▪ Greenplum 6.25+ 지원

Greenplum 7 변경 사항

구 분		Greenplum 6	Greenplum 7
서버	호스트명	▪ mdw, smdw	▪ cdw, scdw
	기준 디렉토리	▪ /data/master/gpseg-1 ▪ \$MASTER_DATA_DIRECTORY	▪ /data/coordinator/gpseg-1 ▪ \$COORDINATOR_DATA_DIRECTORY
	DB log	▪ /data/master/gpseg-1/ pg_log	▪ /data/master/gpseg-1/ log
alias	qq, qqit pg_stat_activity	▪ waiting ▪ waiting_reason	▪ wait_event ▪ wait_event_type
	rgc, rgs	▪ cpu_rate_limit	▪ cpu_max_percent ▪ cpu_weight
카탈로그	Resource Group	▪ CREATE RESOURCE GROUP 구문 옵션 변경 - cpu_rate_limit	▪ CREATE RESOURCE GROUP 구문 옵션 변경 - cpu_max_percent, cpu_weight (변경) - min_cost (추가) - disk (추가) , cgroup v2 반영 필요
	catalog autovacuum	▪ 수동으로 수행(crontab 이용)	▪ 카탈로그 테이블: autovacuum 수행 ▪ 사용자 테이블: 수동 수행 필요
DDL	압축 옵션	▪ ZLIB, ZSTD, QUICKLZ	▪ ZLIB, ZSTD ▪ QUICKLZ(더 이상 지원되지 않음)

Greenplum 7 변경 사항

구 분		Greenplum 6	Greenplum 7
DB 서버 파라미터	삭제된 파라미터	<ul style="list-style-type: none">▪ checkpoint_segments	<ul style="list-style-type: none">▪
	Default 값이 변경된 파라미터	<ul style="list-style-type: none">▪ gp_autostats_mode = on_no_stats	<ul style="list-style-type: none">▪ gp_autostats_mode = none
테이블	파티션 관련	<ul style="list-style-type: none">▪ pg_partition▪ pg_partition_columns▪ pg_partition_encoding▪ pg_partition_rule▪ pg_partition_templates▪ pg_partitions▪ pg_stat_partition_operations▪ pg_partition_def()	<ul style="list-style-type: none">▪ 좌측 Greenplum 6의 테이블/함수 삭제(더 이상 지원 안됨.)▪ 아래 테이블/함수로 관리▪ gp_partition_template▪ pg_partitioned_table▪ pg_partition_tree()▪ pg_partition_ancestors()▪ pg_partition_root()▪ 기존 pg_partitions view는 쿼리로 대체▪ 링크 참조: https://docs.vmware.com/en/VMware-Greenplum/7/greenplum-database/install_guide-migrate-classic-partitioning.html
함수	함수 제거	<ul style="list-style-type: none">▪ gp_percentil_agg	<ul style="list-style-type: none">▪ gp_percentil_agg (함수 제거)
유ти리티	명령어 변환	<ul style="list-style-type: none">▪ createlang, droplang	<ul style="list-style-type: none">▪ CREATE EXTENSION, DROP EXTENSION 으로 사용
	옵션 삭제	<ul style="list-style-type: none">▪ analyzedb의 --skip_root_stats 옵션	<ul style="list-style-type: none">▪ analyzedb의 --skip_root_stats 옵션 지원 안됨.

A- Z Greenplum Version 7



AI & ML

Vector Database, PostgresML



Block Range Index

Min-max data skipping based search



Column Store Enhanced

Col projection applied to more ops



Disaster Recovery

PITR based on pgbackrest



Extended Statistics

FuncDependency, Multivariate MCV



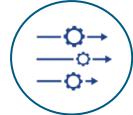
Foreign Data Wrapper

Federated query via PXF, GP2GP



Generated Columns

Compute once, faster use later



Hash Index

O(1) lookup access for large keys



Index Only Scans

Covering indexes, avoid base table



JIT Compilation

Expressions, Tuple Deform, Inlining



Key Platforms

Bare Metal, vSphere, Any Cloud



Linux CGroupV2

Kernel capability leveraged for RM



Modern Partitioning

PG12 engine, hybrid, relaxed locking



Native Fulltext Search

Identify Natural Language Documents



Observability

System/Progress views, Wait Events



Postgres 12

Strong foundation



Query Performance

Speed ups for OLTP and OLAP



Resource Management

CPU, Disk IO, Concurrency, Memory



Security Row Level

Fine grained control over data visibility



Table Alteration

Type/compression, repack, no-rewrite



Unique Index for AO

AO tables can also have Primary Keys



Vacuum Improvements

Catalog auto vacuum, auto analyze



WAL Compression

Disk and N/W IO savings



Xtensions

Pgvector, PostgresML, PGAudit ...



Yes to Speed

Faster Sorting during query execution



Zippy Analyze for AO

Very fast stats creation for AO tables

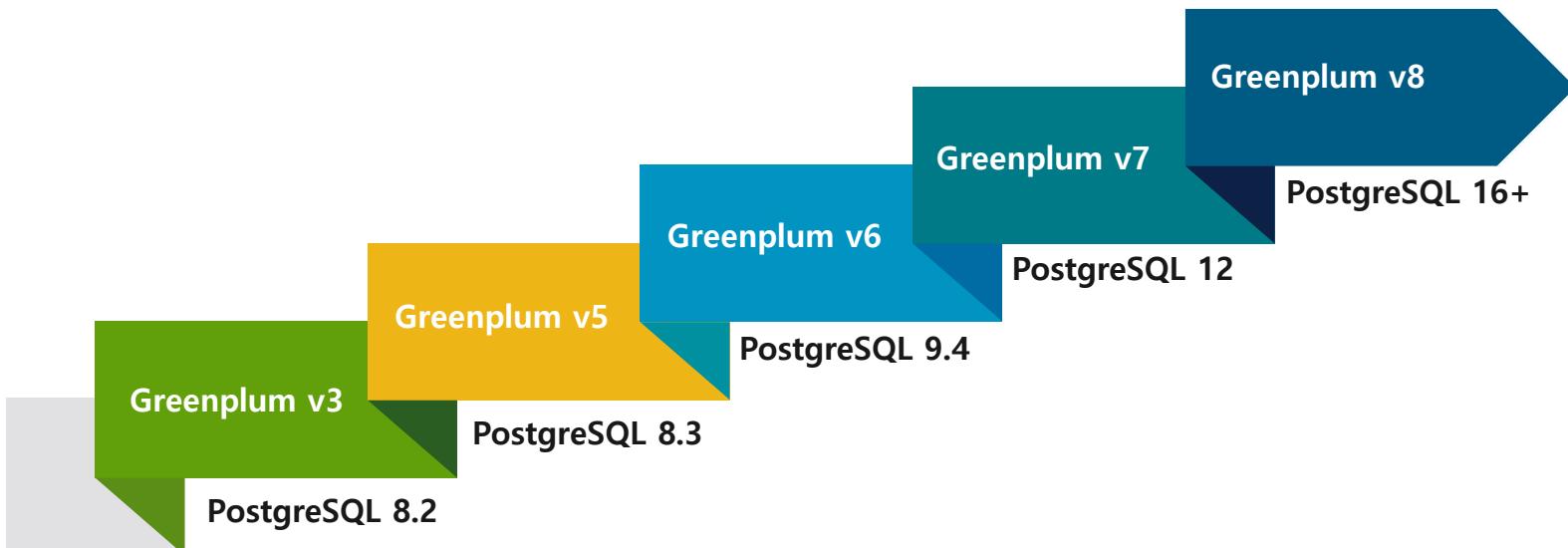
Greenplum 7 개선 사항

개선 사항 상세

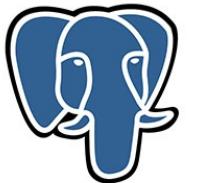
1. Greenplum Core 엔진 개선

Greenplum core 버전 이력

- PostgreSQL 은 관계형 데이터베이스 시스템을 이끄는 글로벌 오픈소스 소프트웨어
- Greenplum 은 PostgreSQL을 기반으로한 병렬 처리 아키텍처 (Massively Parallel Processing, MPP)의 빅데이터 및 분석 데이터베이스로 20년 이상의 노하우 축적
- Greenplum과 PostgreSQL 버전 히스토리



VMware
Greenplum®



PostgreSQL

1. Greenplum Core 엔진 개선

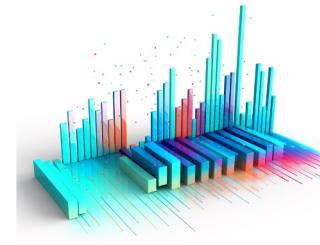
Greenplum 7 Extension Ecosystem

- PostgreSQL과 호환
- PostgreSQL extension으로 Greenplum의 기능 확장
- 필요 요구시에 활성화
- 광범위한 사용 사례

- Advanced password check
- Apache Madlib
- Autoexplain
- Citext: Case Insensitive Text
- Dblink: remote execution
- Disk Space Quotas
- Fuzzy String Match
- **Greenplum FDW**
- HLL: Hyperloglog
- Hstore: Key/Value
- Ip4r: ip data
- Lsn: Media
- Ltree: labeled trees
- Nano Second Timestamps
- Orafce: oracle compatibility
- **postgresml**
- **pgaudit**
- pgcrypto: encryption
- Pg_Trgrm: text processing
- PL/Python, PL/R, PL/Java
- **PL/Python3**
- PL/Container
- **PostGIS 3.3.2: Geospatial**
- **Pgvector 0.5**
- Sslinfo: security
- Tablefunc: pivoting
- UUID: unique identifiers

2. 통계 수집 개선

Helps Optimizer Estimate Better



Better Statistics Allows Greenplum to Query Faster

멀티 컬럼 통계

컬럼 데이터 간의 종속성 파악
컬럼간의 상관관계를 통계 정보
에 반영하여 쿼리 최적화

빠른 통계정보 수집

압축 테이블에 대해 빠른 샘플링
으로 통계 정보 수집 속도 향상

자동 Vacuum과 Analyze

카타로그가 블로트되지 않도록
vacuum 자동 수행
통계 정보 수집 자동화로 최신의
통계 정보에 기반한 최적의 쿼리
플랜 생성

3. 성능 개선

Just-in-Time Compilation (JIT)

- LLVM을 사용하여 JIT (Just in Time) 컴파일 구현
- 장시간 소요되는 쿼리 성능 개선
- 쿼리 수행 시점에 쿼리를 고속의 기계어로 컴파일 후 쿼리 수행
 - 표현식 평가 가속화 (Accelerate expression evaluation)
 - ✓ WHERE clauses
 - ✓ target lists
 - ✓ aggregates
 - ✓ projections
 - 튜플 변환 가속화(Accelerate tuple deforming)
디스크 상의 튜플을 메모리 내의 표현으로 변환
 - 인라인 (Inlining)
함수의 본문 내용을 표현식에 인라인 처리
 - 최적화 (Optimization)
생성 코드 최적화



LLVM은 "Low Level Virtual Machine"의 약어로, 프로그래밍 언어 컴파일러 및 코드 최적화 도구를 개발하고 유지하는 프로젝트 및 컴파일러 인프라를 지칭

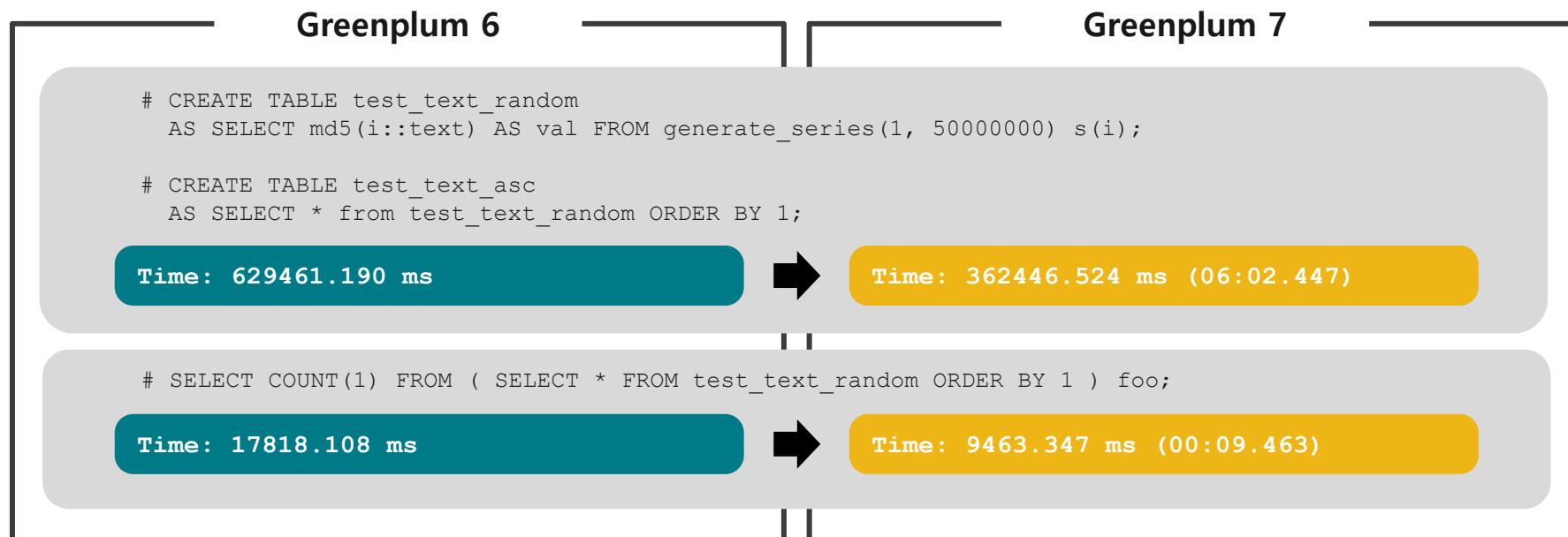
LLVM은 고급 최적화 및 코드 생성을 위한 강력한 프레임워크를 제공하며, 소스 코드를 효율적이고 최적화된 기계 코드로 변환

이를 통해 프로그램 실행 속도를 향상시키고 메모리 사용을 최적화

3. 성능 개선

정렬 속도 개선

- 빠른 OLAP 쿼리 수행을 위한 정렬 속도 개선
- 대부분의 OLAP SQL에서 정렬 작업 포함
- Greenplum 7의 코어 엔진인 PostgreSQL 12에서 정렬 처리 속도 향상
 - Varchar type
 - Text type
 - Numeric type



3. 성능 개선

Index only scan

- Index only scan은 높은 동시성 트랜잭션 조회 쿼리 성능 향상
- PostgreSQL 11에서 도입된 커버링 인덱스(covering index)는 자주 실행하는 특정 유형의 쿼리에 필요한 컬럼을 포함하도록 설계

```
CREATE TABLE foo(a int, b int, c int);

CREATE INDEX foo_cover_index
    ON foo (a) INCLUDE (b);

INSERT INTO foo
SELECT i, i+i, i*i
    FROM generate_series(1, 100)i;

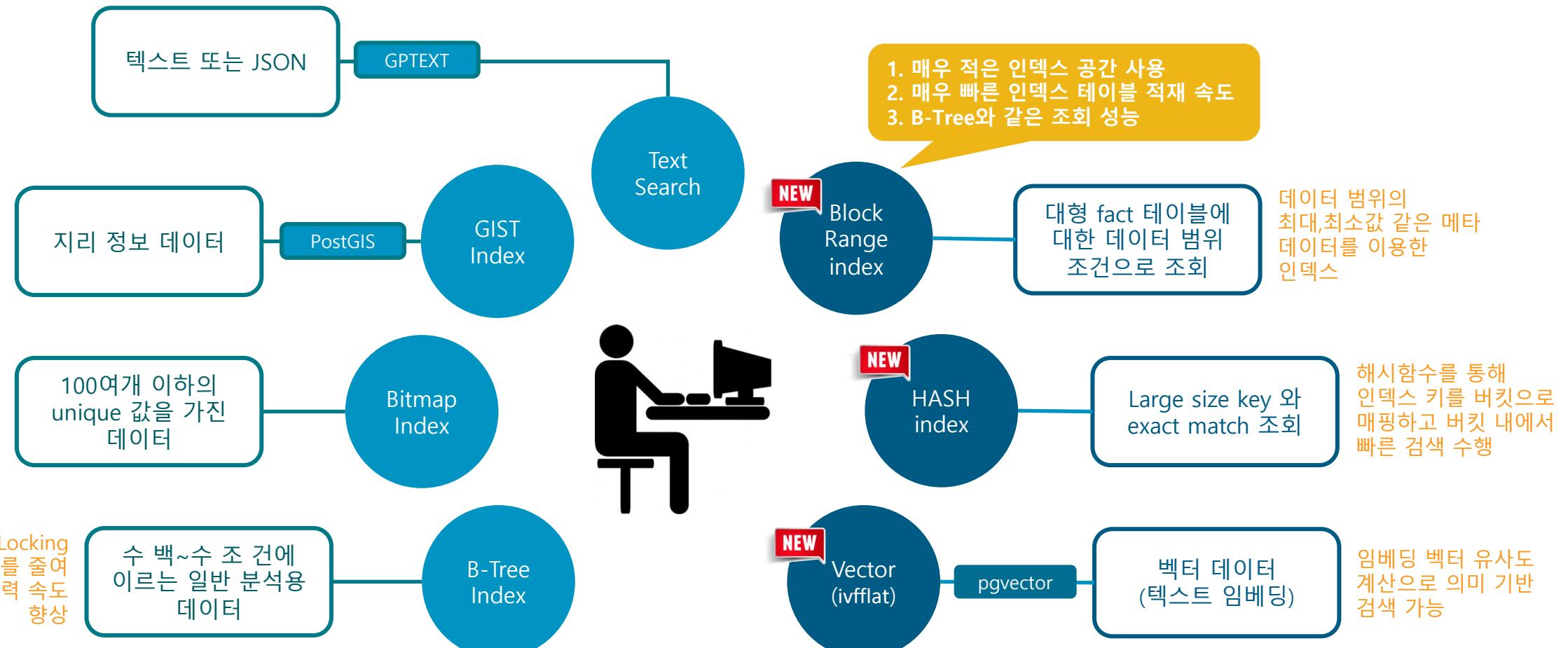
EXPLAIN (ANALYZE, COSTS OFF, TIMING OFF, SUMMARY OFF)
SELECT b
    FROM foo
WHERE a>42 AND b>42;
```

```
vraghavan=# EXPLAIN (ANALYZE, COSTS OFF, TIMING OFF, SUMMARY OFF)
SELECT b FROM foo WHERE a>42 AND b>42;
QUERY PLAN
-----
Gather Motion 3:1 (slice1; segments: 3) (actual rows=58 loops=1)
-> Index Only Scan using foo_cover_index on foo (actual rows=25 loops=1)
    Index Cond: (a > 42)
    Filter: (b > 42)
    Heap Fetches: 25
Optimizer: GPORC
(6 rows)
```

3. 성능 개선

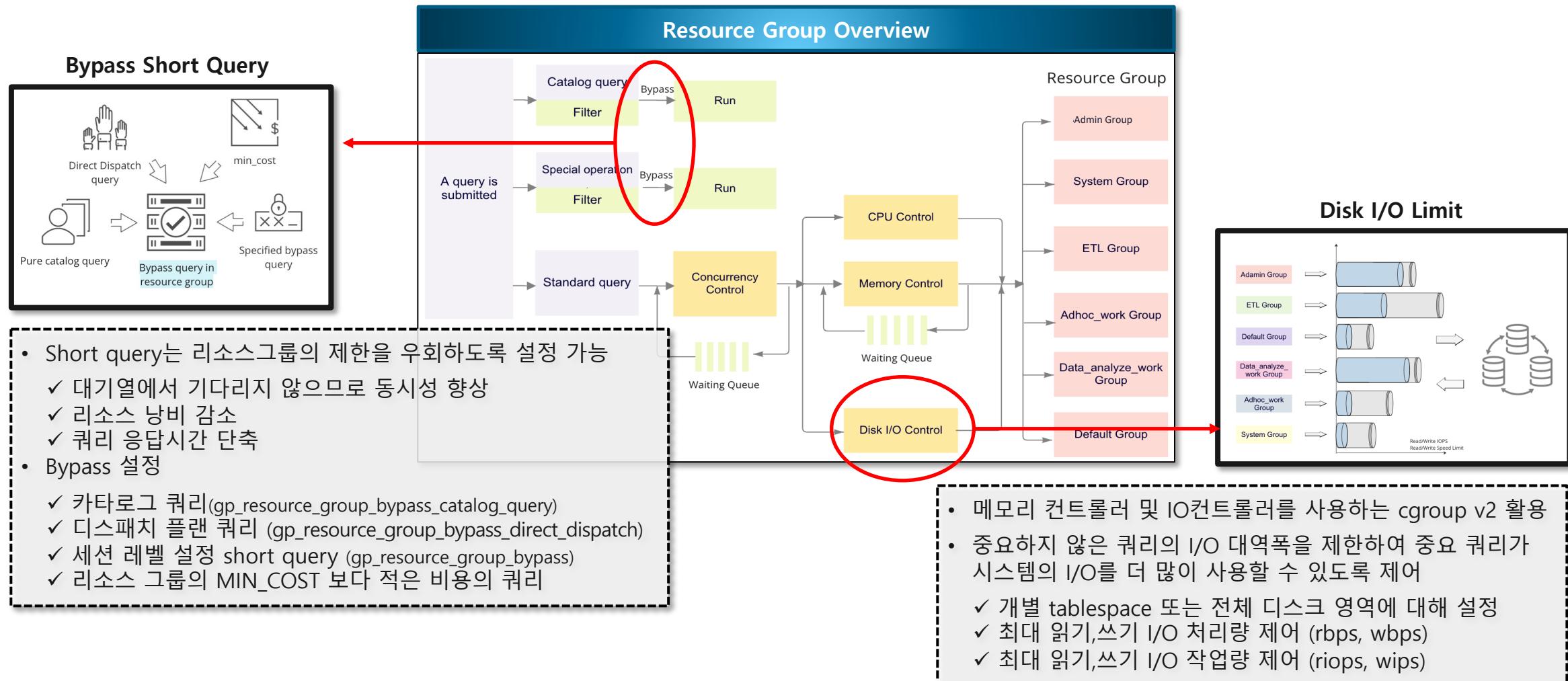
BRIN / Hash 인덱스 지원

- 빅데이터의 신속한 조회가 가능한 다양한 형태의 인덱스 지원



3. 성능 개선

워크로드 관리 기능 향상



4. 개발 생산성 향상

프로시저내 트랜잭션 지원

```
CREATE OR REPLACE PROCEDURE control_trasaction()
LANGUAGE plpgsql
AS $$
DECLARE
BEGIN
    CREATE TABLE test1 (id int);
    INSERT INTO test1 VALUES (1);
    COMMIT ;

    INSERT INTO test1 VALUES (2);
    ROLLBACK ;
END $$ ;
```

```
# call control_transaction();
CALL

# select * from test1;
 id
 ---
 1
(1 row)
```

프로시저와 함수의 차이점

1. 프로시저는 반환할 필요 없으며, INOUT 매개변수를 사용할 때 단일 행만 반환.
2. 프로시저 내에서는 트랜잭션 Commit 또는 Rollback할 수 있지만, 함수에서는 불가
3. 프로시저 호출시에는 select 문 대신 call 문을 사용해서 실행.
4. 함수와 달리, 프로시저에서는 쿼리문 내에서 중첩해서 사용할 수 없음.
(SELECT, INSERT, UPDATE, DELETE)

4. 개발 생산성 향상

UPSERT 지원

```
CREATE TABLE sensor_read (
    sensorid int primary key,
    reading float) ;
```

```
INSERT INTO sensor_read
VALUES (6023, 9.33123) ;
```

```
INSERT INTO sensor_read
VALUES (6023, 9.42331) ;
```

```
INSERT INTO sensor_read
VALUES (6023, 9.42331)
```

```
ON CONFLICT(sensorid)
DO UPDATE
SET reading = EXCLUDED.reading ;
```

ERROR: duplicate key value
violates unique constraint

UPSERT 지원

- 데이터 merge를 위한 손쉬운 방법 제공
- 제약 조건을 충돌하는 Insert 구문 실행 시 Update로 데이터를 변환하거나 무시하도록 허용
 - ✓ DO UPDATE
 - ✓ DO NOTHING
- 테이블 이름을 사용하면 현재 행 값에 액세스
- EXCLUDED를 사용하면 새로 입력되는 행 값에 액세스 가능

4. 개발 생산성 향상

제너레이티드 컬럼

```
CREATE TABLE customers (
    customerid   INT      GENERATED BY
                      DEFAULT AS IDENTITY,
    customername TEXT,
    creditcard   TEXT,
    cc_mask       TEXT
                      GENERATED ALWAYS
                      AS (OVERLAY(creditcard
                                    PLACING 'XXXX-XXXX-XXXX-'
                                    FROM 1 FOR 15)) STORED
) ;

INSERT INTO customers
    (customername, creditcard)
VALUES ('Micky', '1234-5678-1234-5678'),
       ('Snoopy', '9876-5432-9876-5432') ;

SELECT customerid, customername, cc_mask
  FROM customers;
```

customerid	customername	cc_mask
2	Snoopy	XXXX-XXXX-XXXX-5432
1	Micky	XXXX-XXXX-XXXX-5678

표현식, 사용자 정의 함수로
generate 가능

데이터 마스킹 처리

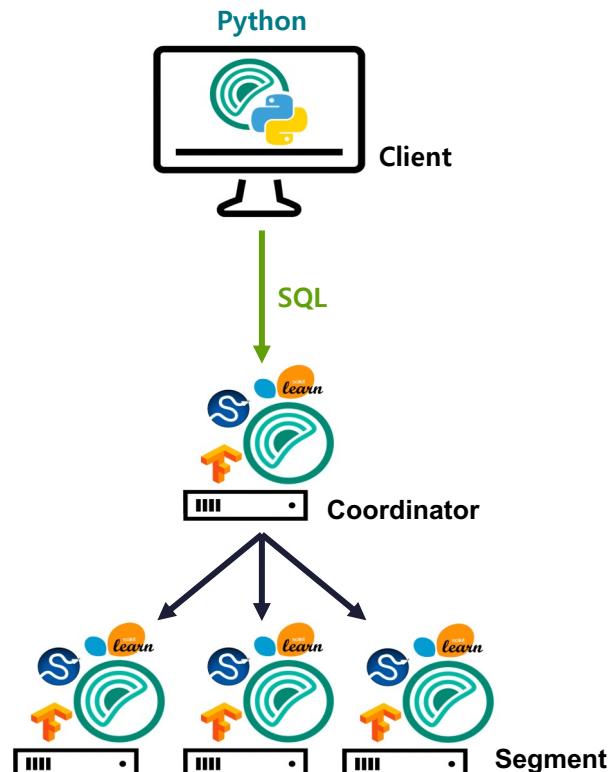
컬럼 값 자동 생성

- Postgres 12의 stored generated column 기능
- 데이터가 입력되거나 수정될 때 컬럼 데이터를 자동 생성하여 테이블 컬럼에 저장
- 자주 사용하는 쿼리에서 빈번하게 계산하여 표시해야 하는 경우에 미리 컬럼을 자동 생성하여 쿼리 속도 향상
- 마스킹된 데이터를 생성하여 컬럼 데이터 수준의 보안 기능 제공

4. 개발 생산성 향상

Greenplum Python Client Library

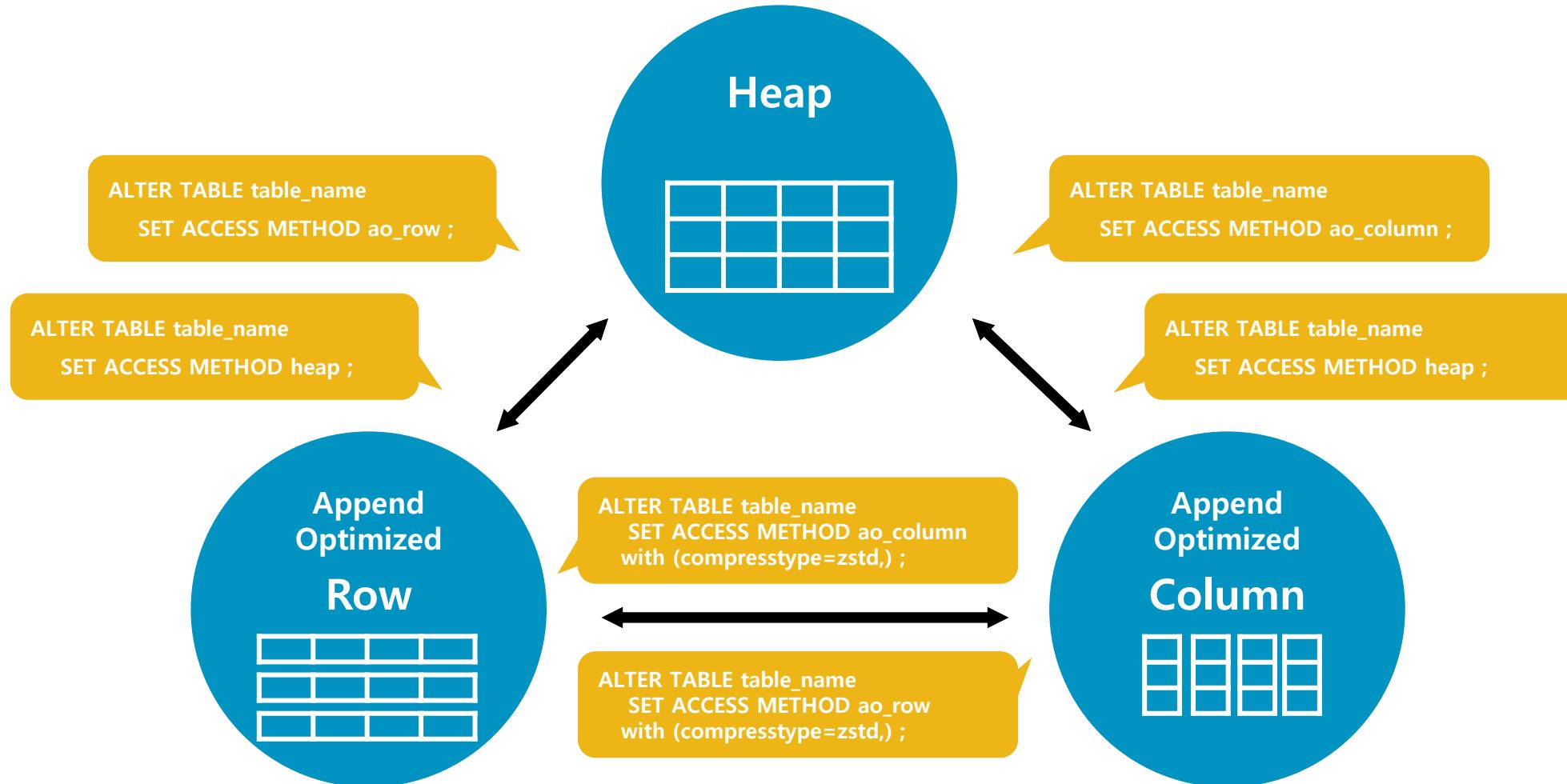
- 데이터 분석을 용이하게 하기 위해 프로그래밍 언어 기본 API 세트 제공



Features	Method
Use GreenplumPython in Python	<code>import greenplumpython as gp</code>
Connect to database	<code>db = gp.database(params={"host": "localhost", "dbname": "postgres"})</code>
Create DataFrame from existing table	<code>df = db.create_dataframe(table_name="tab")</code>
Select a sub-DataFrame	<code>df[["a", "b"]]</code> (columns) <code>df[:n]</code> (rows)
Save DataFrame in GPDB	<code>df.save_as("new_table")</code>
Join two DataFrames	<code>df = df1.inner_join(df2, on=["a"], self_columns={"a": "df1_a"}, other_columns={"a": "df2_a"})</code>
Create and call UDF and UDA Function or aggregate predefined Function or aggregate defined by user Array function or aggregate	<code>@gp.create_function</code> <code>def add(a: int, b: int) → int</code> <code> return a + b</code> <code>df = df.apply(lambda row: add(row["a"], row["b"]))</code>
Generate and Search Embeddings	<code>import greenplumpython.experimental.embedding</code> <code>t = t.embedding().create_index(column="content", model="all-MiniLM-L6-v2")</code> <code>t.embedding().search(column="content", query="apple", top_k=1)</code>

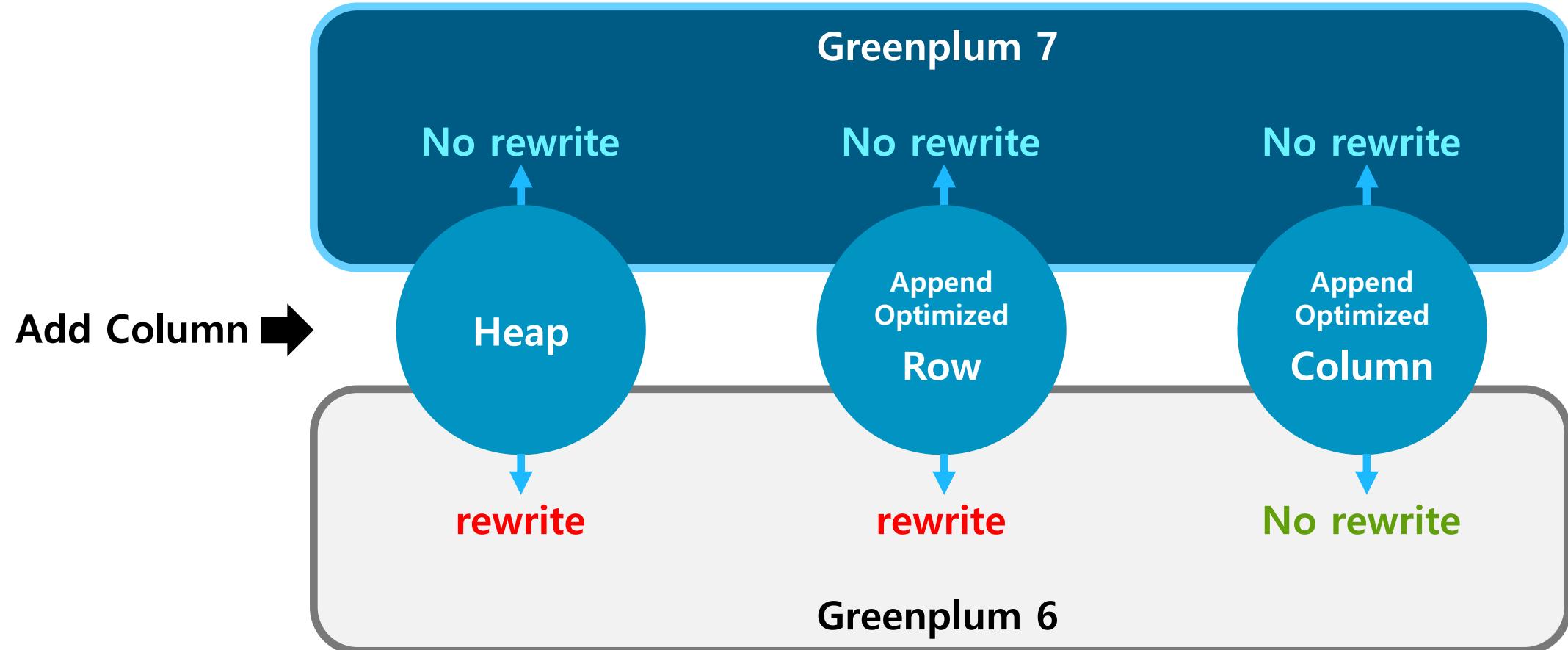
5. 관리 개선

효율적인 스키마 변경: Alter Table 구문으로 스토리지 옵션 변경



5. 관리 개선

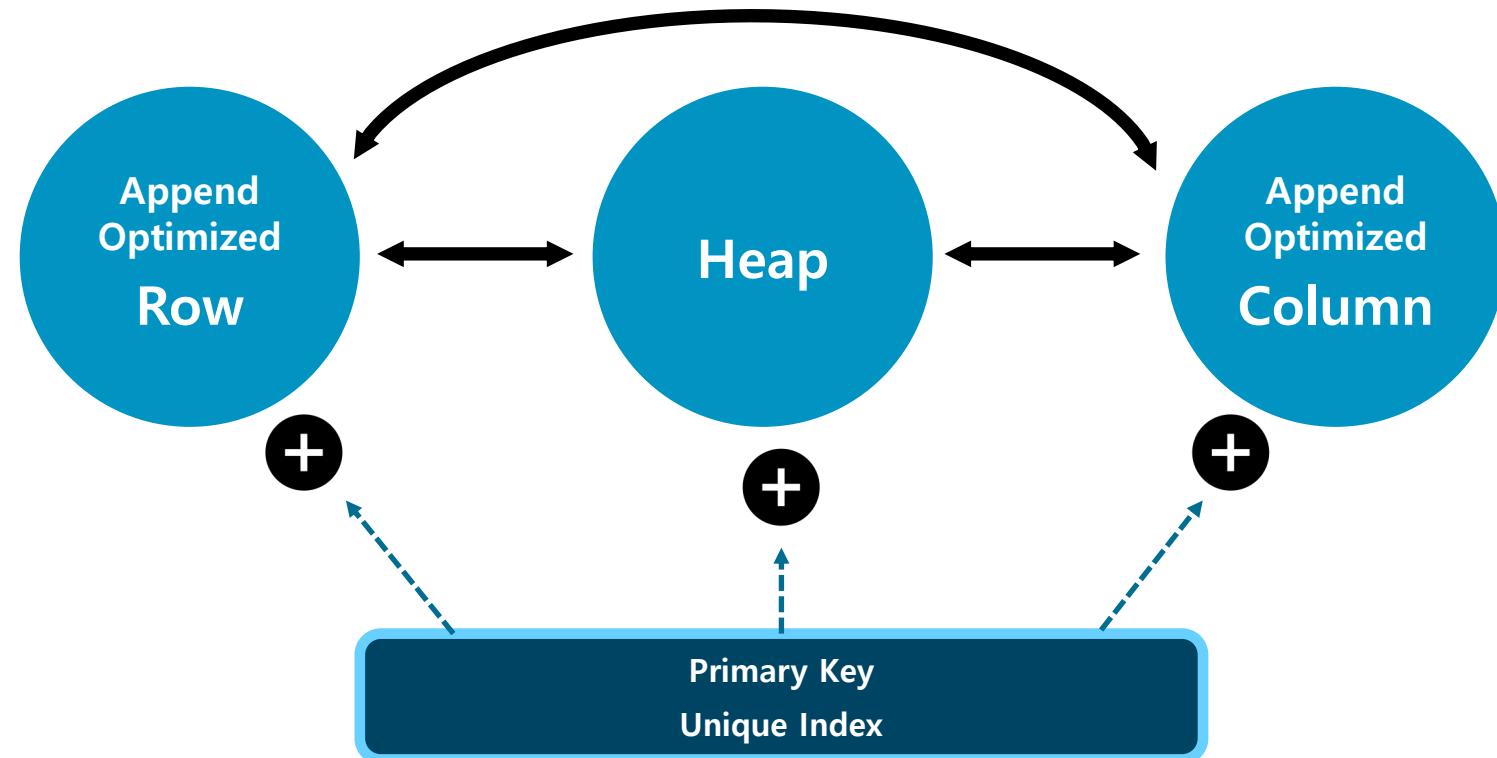
효율적인 스키마 변경: 테이블 컬럼 추가시 카탈로그에만 적용



5. 관리 개선

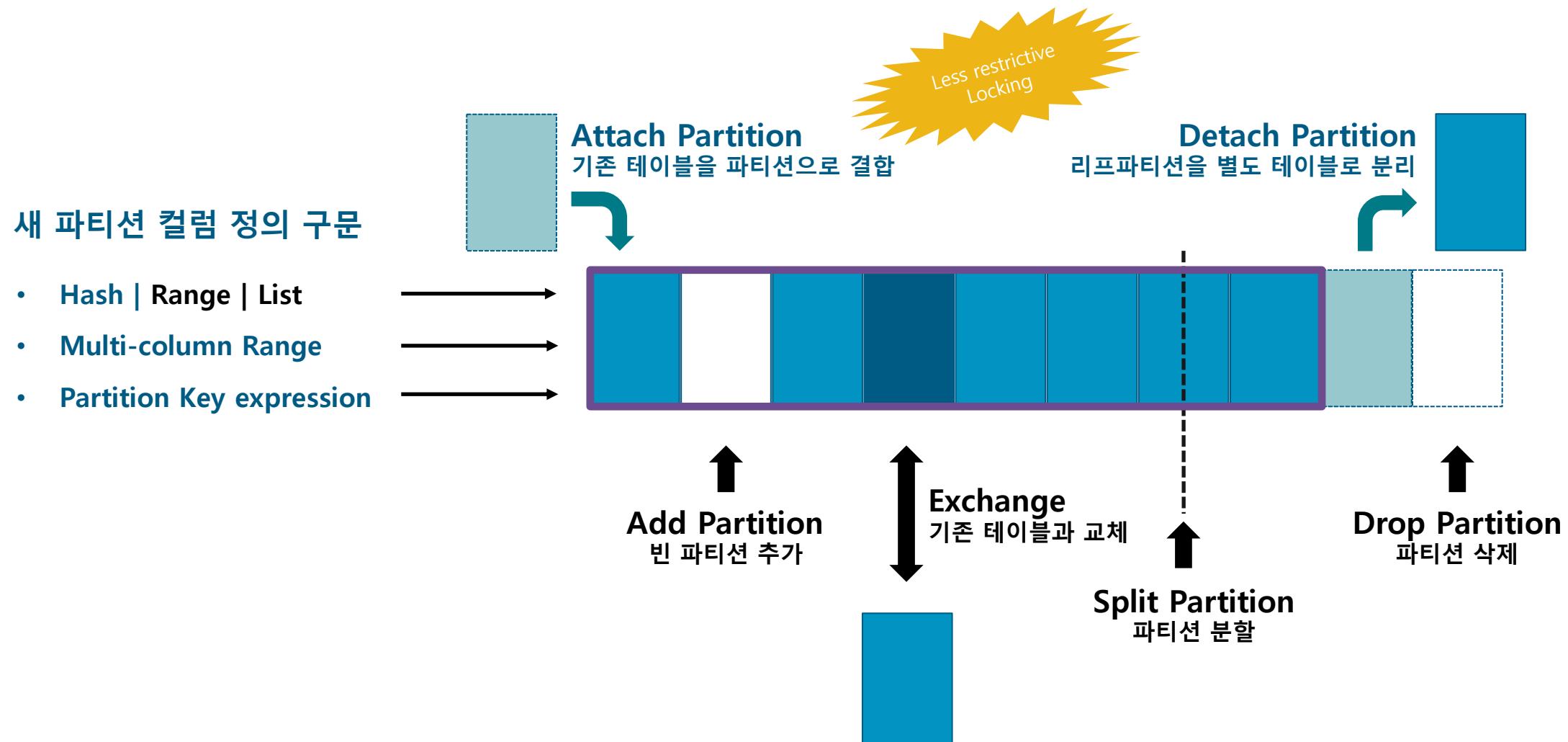
압축 테이블에 PK/UK 지원

- 압축 테이블 데이터의 unique 속성 지원 (Primary Key, Unique Constraints)
- 데이터의 unique 속성과 컬럼 테이블의 성능 및 스토리지의 이점을 유지
- Unique 속성의 제약이 없어 Oracle 또는 SQL Server에서 워크로드를 더 쉽게 이동



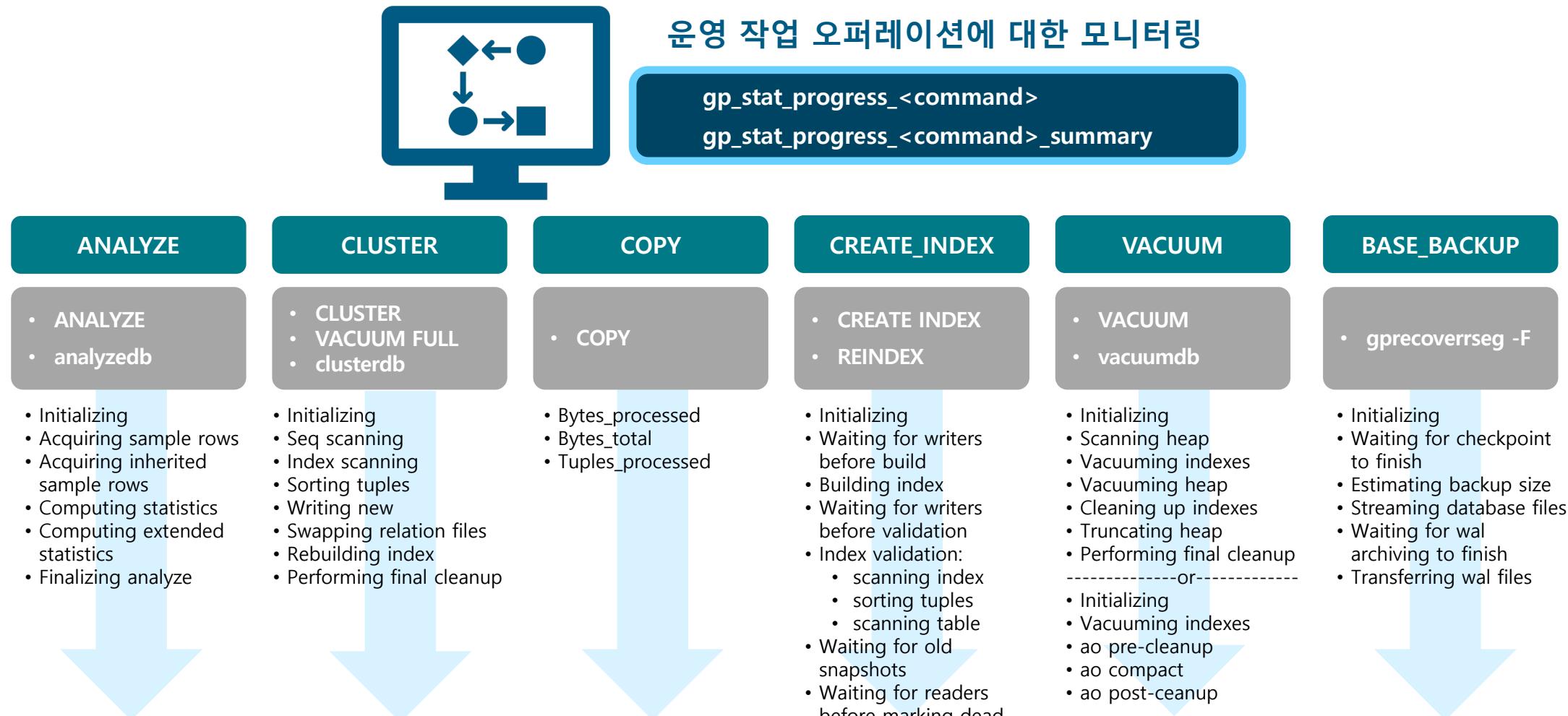
5. 관리 개선

Postgres 12의 파티션 관리 기능 호환



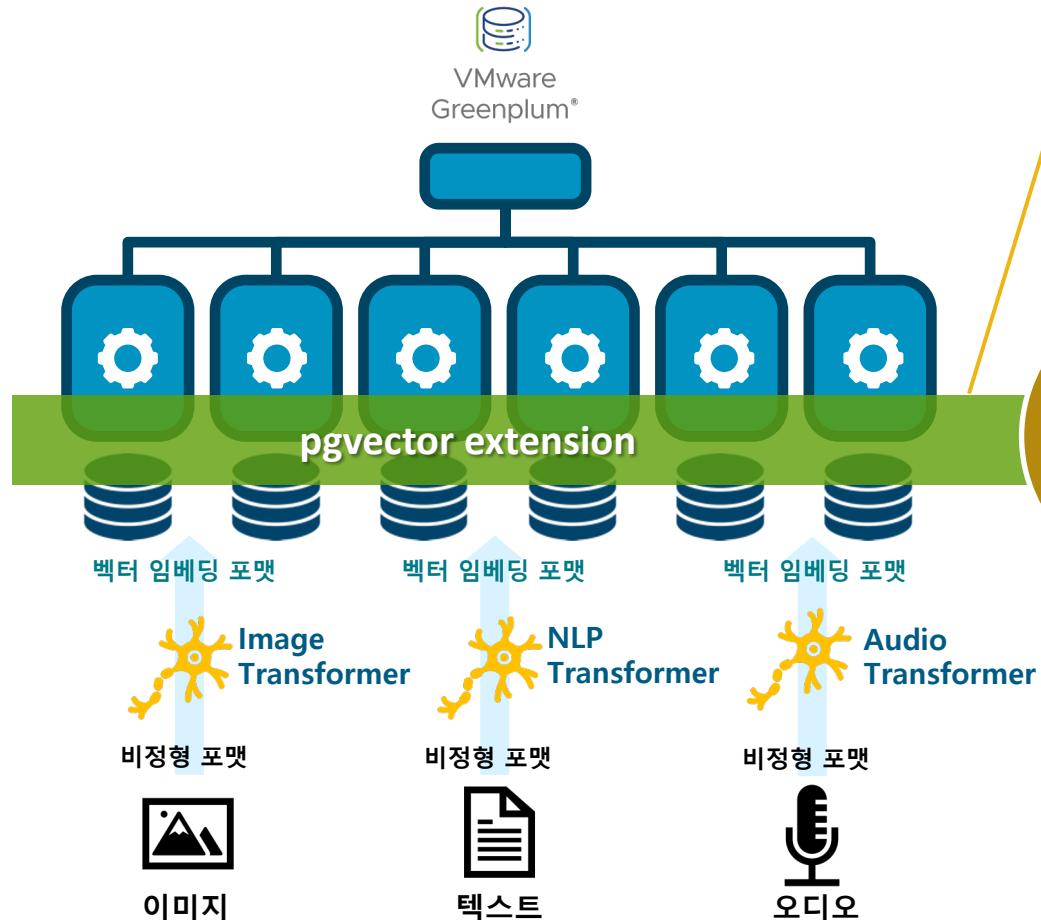
5. 관리 개선

운영 명령어 모니터링 지원



6. AI 기반 의미 검색

pgvector: 벡터 저장 및 유사 검색 지원



pgvector

- Vector 유사도 검색을 위한 Postgres Extension
- Embedding 벡터 데이터 저장
- 벡터 인덱스 생성 및 손쉬운 유사도 검색

대용량처리

페타바이트급
의 데이터
확장성 제공

고성능

대규모 분산
병렬 처리
아키텍처에서
운영

통합분석

기존의 분석
데이터와 함께
통합된 분석
가능

보안성

일관되고
강화된 데이터
보안 관리

Generative AI 활용

이미지
생성과 변환

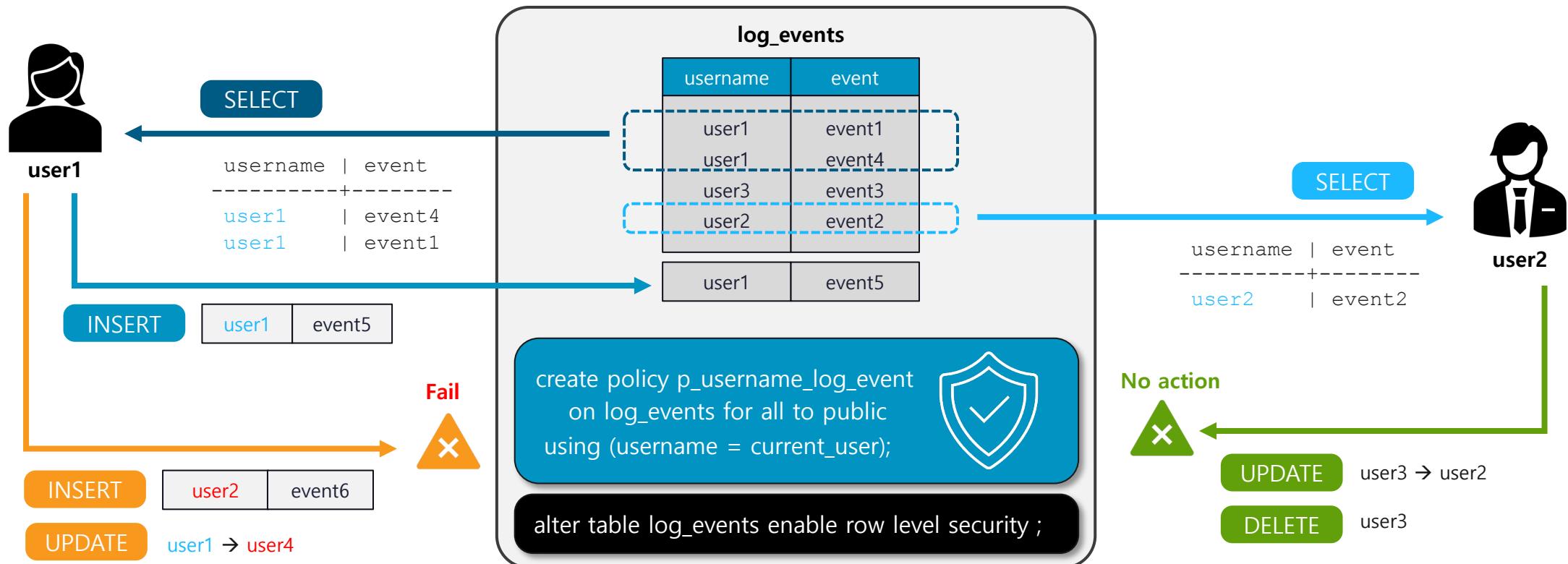
자연어
생성
및 번역

...
추천 시스템

음성 합성과
변환

7. 보안 강화

Row 레벨 보안



저장된 데이터가 Policy 규칙에 맞지 않는 경우

- Select 문 : 데이터가 필터링 되어 출력되지 않음
- Update 문 : 무시 (에러 메세지 없음)
- Delete 문 : 무시 (에러 메세지 없음)

Policy 정의의 조건이 Insert나 Update처리 단계에서 위배되면 에러 발생

7. 보안 강화

감사 기능 제공: pgaudit





A large, abstract graphic element occupies the left side of the slide. It features a large circle with a gradient from dark green at the top to light green at the bottom. This circle overlaps a diagonal line that starts in the bottom-left corner and extends towards the top-right. The area between the circle and the line is filled with a teal color. A thin horizontal line also extends from the bottom-left corner across the slide.

Thank You