-67. t6v0t68v0t75v

=4248 5 3<
=l εy×~wz
= . .

---

638 C

636

t6vM개k결물광HPN2o -obPN 7575.

Evhtμ~dkHF/] Etqihih . *WUP + * .
+ WUP 0 =

30] EU*] mk~wz/E×ou mv~ml Evhtμ~dkht U×mzμ+ =WUP . W PI GY ,
JVSQ q+mu | ^ LI VI ERRW*mu i ml qyo. y×mzμf −mk. 32+ERH xzdkm? 3222
40] KTU = Tzwl ×k~U×hv~qph~qw * + ERR
50 *Ekk×zhkμ/E←hzmGFS+=ERR .

=
0
Evhtμ~dkHF/] 32' .
0 l εy×~wz 0

t68v경광분라~공결고e경부헤l 결과분라17576기 .

Twl ~ozmWUP xo−mk~wz HF . . LRW^ 1N]IJJth~ .
*P4. . + 0 =

30 WUP = ... GYI .
40 =xofl ×u x. .
50KqL×i 34.222- =

= xo−mk~wz 5505' *A 315+ 0 %l mmqym
HI JE[PYfW P 2055555 . 0 Twl ~ozmWUP
. . 감 0

t75v[OPQ7579 Cf결물광분라gb

T×zl ×m[vq+mzl qμ Twl ~ozmWUP 0
7 =

30 * +=LRW^ 52감72'
답 Jhdl 3:
40 * += . Q] GG 42감72'
50 * +=: OF . Q] GG Jhdl : 답
MS
60 * +=^ EP . Q] GG Jhdl <8 *3.222 =
Twl ~ozmWUP 6: −l Jhdl 52 +
70 * +=5505' 답 2023' 55.522 .
647

= 3감5 Twl ~ozmWUP * . Q] GG. + . 6감7
0l εy×~wz 7 * + xo−mk~wz 3.222
0

637 C

=

```
[1] AnalyticDB-V (2020)
  └ "벡터 검색을 SQL에 통합하자" → VAQ 개념 제시
    하지만 카디널리티 추정은 고정값(10%) 가정
    → 미해결 문제 제기

          ↓

[13] pgvector (2021~)
  └ AnalyticDB-V 개념의 오픈소스 구현
    하지만 더 심한 고정 선택도(33.3%) 문제 드러남
    → 실무에서 가장 널리 쓰이는 구현체임에도 불구하고
      치명적 최적화 오류 발생
    → 문제의 심각성 인식

          ↓

[20] ICDE 2024 논문 (2024)
  └ pgvector의 성능 문제를 체계적으로 분석
    5가지 한계를 식별하고, 근본 vs 구현 구분
    특히 고정 선택도(33.3%)를 근본적인 것은 아니지만
    **매우 영향이 큰** 구현상 문제로 규명
    → 학술적 정당성 확보

          ↓

Exqutor (2025)
  └ [20]이 식별한 고정 선택도 문제를 직접 해결
    ECQO를 통해 **실제 카디널리티 추정**
    → pgvector에서 최대 1,000배 성능 향상 달성
```

=

30b3d = 0# - WUP # .

40b35d = 0Evhtu~dkHF/] .

50b42d = 0xo-mk~wz 7 .

60l εy×~wz = 0b42d .

---

738 C

---

736 C% PN I %

=

각V/////각V////각V/////////////////각V////각V//////각				
Evhtu~dkHF/]	4242	32'	#	# .
xo-mk~wz	4243감	55C5'	#]Y1J 315	# 1 .
N3HI 4246	4246			2023' 감<2'

737 C

6C

```
데이터: 상품 이미지 1,000,000개
```

```
쿼리: 특정 이미지와 유사한 상품 찾기 (거리 < 0.5)

실제 데이터 분포:
  - 쿼리A: 유사 상품 5개 → 선택도 0.0005%
    AnalyticDB-V 추정: 10% → 오류: 20,000배
    pgvector 추정: 33.3% → 오류: 66,600배

  - 쿼리B: 유사 상품 5,000개 → 선택도 0.5%
    AnalyticDB-V 추정: 10% → 오류: 20배
    pgvector 추정: 33.3% → 오류: 66배

  - 쿼리C: 유사 상품 500,000개 → 선택도 50%
    AnalyticDB-V 추정: 10% → 과소 추정 5배
    pgvector 추정: 33.3% → 과소 추정 1.5배
```

7C

```
데이터: 뉴스 기사 10,000,000개 (각 768차원 BERT 임베딩)
쿼리: 특정 주제와 관련된 기사 찾기

실제 분포:
  - 쿼리"기술뉴스": 유사 기사 10,000개 → 선택도 0.1%
    pgvector 추정: 33.3% → 오류: 333배

  - 쿼리"스포츠뉴스": 유사 기사 1,000,000개 → 선택도 10%
    pgvector 추정: 33.3% → 오류: 3.3배
```

8C[OPQ 7579

```
데이터셋: Wikipedia 이미지 임베딩 10,000,000개
쿼리: 벡터 유사도 < 1.0 (가장 가까운 100개 찾기)

실제 선택도: 0.001% (100 / 10,000,000)
pgvector 추정: 33.3%

오류: 33,300배 과대 추정

최종 쿼리 성능:
  - 비효율적 조인 순서로 인한 추가 시간: 850초
  - 정확한 추정으로 계획한 경우: 2초
  - 성능 차이: 425배
```

73B

Tw| ~ozmWUP . . =

```
SELECT o.order_id, o.customer_id, SUM(l.price * l.quantity) AS total
FROM orders o
JOIN lineitem l ON o.order_id = l.order_id
WHERE l.product_embedding <-> query_vec < 0.5
  AND o.order_date > '2025-01-01'
  AND l.category = 'electronics'
GROUP BY o.order_id, o.customer_id;
```

C G 536) - .1 G : 55

xo-mk~wz * +=
toymq-mu =822
=822 극5505' A 422
=Wmy Wkhv ww wzl mzl *322 +답 toymq-mu 822

=
=822 극203' A 8.222
= 8.222 답 wzl mzl

=

```
pgvector 계획:
```

```
  100만 × 600만 = 600경 불필요한 스캔 작업 → 매우 느림

최적 계획:
  6,000행 × 100만 = 60억 작업량 → 매우 빠름

비율: 최대 1,000배 차이
```

739 -Vz경

각V////각V/////////각V//////////각V//////각			
b3dEvhtμ~dqHF/]	WUP		
b35dxo-mk~wz			*55C5' +
b42dM3HI 4246	7		# #

I εy×~wz =b42d # . #
답 I εy×~wz I GUS

---

83 -f 고결~결뮌 I 공공뮌각헤몸

836

= *] mk~wz Vhvom U×mzμ+ WUP . * +
0

=

=

```
SELECT ... FROM R
WHERE σ_vec(v_col, q_vec, ε) AND σ_scalar(scalar_col)
```

=

V= *각V각A v +
-f kwt= *I +
하f-mk= * 편 +
하f|khthz=

=

|mtf-mk =

```
현재 (고정값 기반):
  sel_vec = 0.333333 (pgvector)  or  0.1 (AnalyticDB-V)
  → 실제 데이터와 무관한 상수값

필요한 것 (정확한 추정):
  sel_vec = ¦{r ∈ R : distance(r.v_col, q_vec) < ε}¦ / ¦R¦
  → 데이터와 쿼리에 의존하는 함수
```

=

=

```
잘못된 추정:
  ¦σ_vec ∩ σ_scalar¦ ≈ n × 0.333 × sel_scalar
  → 조인 순서, 조인 방식, 인덱스 사용 결정 오류

정확한 추정:
  ¦σ_vec ∩ σ_scalar¦ ≈ n × actual_sel_vec × sel_scalar
  → 최적 실행 계획 수립
```

837 I

7.3.6

Evhtu~dkHF/] 32' =

. 42

xo¬mk~wz 55.5' =

=20.223' 감<2'

55.522

7.3.7

HF * .QG] +=

```
WHERE age > 30 AND salary < 100000
```

Tw|~ozmWUP =

homB 52 = 답 감<2'

|hthzu ? 322222 = 답 감<2'

=

=

```
WHERE embedding <-> query_vec < 0.5
```

30 =98:

3H =32감322

98: H = A 32e98: * +

40 =

3 20.7 20.3'

4 20.7 72'

50 =

7.3.8

=# C#

=

*3감4u | +

# #

| εy×~wz = . 3감4u |

*3.222 +

8.3.8 C I

8.3.8.6

```
스칼라 범위 검색:
  WHERE age > 30
  결과: 결정적 (같은 데이터면 항상 같은 결과)

벡터 범위 검색:
  WHERE embedding <-> query_vec < 0.5
  결과: 쿼리 벡터 q_vec에 의존
        같은 데이터, 다른 q_vec → 완전히 다른 결과
```

833337

*G×z| mwnHqu mv| qwhtqμ+=

```
1차원: 거리 d인 점들 = 선분 (길이 2d)
2차원: 거리 d인 점들 = 원 (넓이 πd²)
3차원: 거리 d인 점들 = 구 (부피 (4/3)πd³)
768차원: 거리 d인 "초구" 부피 = d^768에 비례

문제: 벡터 공간의 대부분이 중심에서 멀리 떨어져 있음
      → 임의의 두 벡터도 매우 멀리 떨어짐
      → 거리 분포가 쿼리에 따라 극도로 다양함
```

833338

LRW^ 1N/IJJth~ =

```
HNSW 인덱스:
  - 근사 알고리즘: 모든 답을 보장하지 않음
  - ef_search 파라미터에 따라 정확도 조정 가능
  - ef_search=100 → recall 95%
  - ef_search=500 → recall 99.9%

문제:
  선택도 추정을 위해 인덱스 탐색 시,
  어떤 ef_search 값을 사용할 것인가?

  선택도 추정에서 99% 정확도 필요?
  아니면 95% 정확도면 충분?
```

833339

```
쿼리 최적화 중:
  SELECT ... FROM R WHERE vec_condition AND scalar_condition

현재 시점의 선택도: 카디널리티 K
↓
(다른 사용자가 동시에 데이터 수정)
↓
쿼리 실행 단계: 실제 카디널리티 K'
→ K ≠ K'인 경우 발생

이 동시성 문제를 어떻게 처리할 것인가?
```

---

839 Q구계마공결고

839336 QOg e -Q값값나 공판만나Oz 교 마개 경불Qz 교z 교나g 과나 교을e 경공맨 개 공판에

=# . 3감4u | #

```
최적화 단계:
  1. 벡터 조건 식별: WHERE embedding <-> query_vec < ε
  2. HNSW 인덱스에서 범위 검색 실행 (1~2ms)
     → 정확한 카디널리티 K 획득
  3. PostgreSQL 옵티마이저에 K 전달
  4. 최적 실행 계획 수립

쿼리 실행 단계:
  추가 비용 없음 (이미 최적 계획 생성)
```

839337 2

=3감4u | * +

=3.222 * +

= 322u | 답 322품

/ =3u | 답 322u | *322 +

8393 f 결론공부라 gb

xo-mk~wz =

30 =Tw| ~ozmWUP *xthvvmzpwws+
xo-mk~wz

40 =
*l q| ~hvkm? 207+
Ywx/O *SVHI V F `l q| ~hvkmPM2MM 32+

50 =Tw| ~ozmWUP
xo-mk~wz *QµWUP. WUPq-m +

93Q구계마공결고

---

936 Q구계마공결고

6Ct6vM계z 결론공배PN2o GoMg 1

l εy×~wz =Vmth~ml ^ wzs

=
]EU*] mk~wz/E×ou mv~ml Evhtu~dkhtU×mzu+
WUP * Etq| hi h +
* +

l εy×~wz =
Evhtu~dkHF/] = * -| +
l εy×~wz= * . +

=

```
Exqutor 논문:
"AnalyticDB-V는 최초로 벡터 검색을 SQL의 연산자로
통합했으나, 카디널리티 추정을 다루지 않아
옵티마이저가 고정된 선택도로 계획을 수립한다.
본 연구는 이 문제를 직접 해결한다."
```

7Ct68v경공문약-공결고G 18833)

l εy×~wz =

=
HF
5505'
*34.222- Kq-L×i A A +

l εy×~wz =
30 = HF xo-mk~wz
40 ih| mtqym=5505'
50 =Tw| ~ozmWUP

=

```
Exqutor 논문의 실험 구조:

기본 pgvector:
  - 선택도: 고정 33.3%
  - 쿼리 시간: 850초 (잘못된 계획)

Exqutor (ECQO 적용):
  - 선택도: 정확한 추정
  - 쿼리 시간: 0.85초
  - 성능 향상: 1,000배
```

=

```
Exqutor 논문:
"pgvector는 PostgreSQL 기반의 벡터 검색 확장으로,
완전한 SQL 기능을 제공하는 장점이 있다.
그러나 벡터 조건의 선택도를 항상 33.3%로
고정 추정하여, 옵티마이저가 비효율적인
실행 계획을 수립한다.
본 연구는 pgvector에 ECQO를 통합하여
정확한 카디널리티 기반 최적화를 실현한다."
```

8Ct75v[OPQ 7579 G 경우 문제 공결고 1

| εy×~wz =

=

xo¬mk~wz 7

¬

답 | εy×~wz

| εy×~wz =

MGHI 4246 =

```
"pgvector의 5가지 한계 중:
1~3번 (버퍼 풀, 튜플 오버헤드, 공간 증폭):
  → 근본적 (PostgreSQL 아키텍처 한계)
  → 개선 어려움

4번 (인덱스 구축 시간):
  → 부분 근본
  → 병렬 구축, WAL 우회로 개선 가능

5번 (고정 선택도):
  → 부분 근본
  → 적응적 샘플링, 벡터 통계, 인덱스 탐색으로 개선 가능
"
```

| εy×~wz =7

=

```
Exqutor 논문:
"Zhang et al. (ICDE 2024)은 PostgreSQL 벡터 지원의
5가지 한계를 규명하였다. 특히 카디널리티 추정 문제는
구현 개선으로 해결 가능하지만, 정확한 추정 기법은
미제시되었다. 본 연구는 ECQO를 통해
이 문제를 구체적으로 해결한다."
```

---

937 Q구계 문제 공결고 관물

C : - . 설 6 1555

| GUS =

```
# 최적화 단계 (1~2ms 추가 비용)
def estimate_vector_cardinality(query_vector, threshold):
    # pgvector의 HNSW 인덱스에서 실제 범위 검색 실행
    results = hnsw_index.range_search(query_vector, threshold)
    return len(results)  # 정확한 카디널리티

# PostgreSQL 옵티마이저 통합
def planner_hook(parse, ...)
    if has_vector_condition(parse):
        true_cardinality = estimate_vector_cardinality(...)
        set_selectivity(true_cardinality)
    return optimized_plan
```

=

	xo¬mk~wz* 55.05' +	l εy×~wz* +	
각/////////각/////////////////////각/////////////////////각/////////각			
20.223'	*: 72 +	*20.7 +	3.222
20.7'	*: 7 +	*20.7 +	392
72'	*72 +	*62 +	30.47

C 6가

3= * +

```
원인: HNSW 그래프 탐색의 무작위 메모리 접근 → 래치 경합
Exqutor의 영향: 선택도 추정으로는 해결 불가
대안: VBASE(Vector-Buffer Architecture) 같은 새로운 아키텍처 필요
```

4= * +

```
원인: 각 벡터마다 튜플 헤더 파싱, MVCC 가시성 확인
Exqutor의 영향: 선택도 추정으로는 해결 불가
대안: 벡터를 immutable로 취급하여 MVCC 간소화 필요
```

5= * +

```
원인: 8KB 페이지 단위 저장으로 인한 패딩, MVCC 메타데이터
Exqutor의 영향: 선택도 추정으로는 해결 불가
대안: 가변 크기 저장 또는 벡터 전용 저장소 필요
```

6= * +

```
원인: WAL 로깅, MVCC 메타데이터, maintenance_work_mem 부족
Exqutor의 영향: 선택도 추정으로는 해결 불가
대안: 병렬 구축, WAL 우회(인덱스 재생성) 필요
```

93.3 Q구현과 문제점

l εy×~wz =

30 *42 += Jhq|

40 *: +=

50 =

=

xo¬mk~wz =Jhq| 42.222 * +

l εy×~wz =Jhq| 42 * +

=#l εy×~wz xo¬mk~wz .Tw|~ozmWUP #

: 3 - 6: .

: 36 - 7 .

0 0

Etqihih Evhtu~qkHF/]*]PHF 4242+. HF xo-mk~wz.
NGHI 4246 Tw|~ozmWUP 0

. . . 0
lεy×~wz 0

. . lεy×~wz G#

: 37 MP게 결론PHP N2o - 8 .

Evhtu~qkHF/] 0

Evhtu~qkHF/] Etqihih 4242]PHF 0 = *WUP +
* . + WUP 0

. =
30 *Jhq| + 답 3.222
40 M1 WUP 답 W PI GY, JVSQ xzwl ×k~| ^ LI VI xzwl ×k~fq| M1 *CCD-
50 .

Evhtu~qkHF/] =

```
SELECT * FROM products
WHERE ANNS(image_embedding, query_img, 10)  -- 벡터 top-10
  AND price < 100000                        -- SQL 조건
```

WUP 0

Evhtu~qkHF/] WUP 0
) .) 0

= C

Evhtu~qkHF/] 0 # 32' # 0 . 322
32 0

0 3 . 322 . 72 0) 32')
0

Evhtu~qkHF/]) - WUP) .)
) 0 lεy×~wz G#

: 38 경영분석 공결고 - 8 .

xo-mk~wz 0

xo-mk~wz Evhtu~qkHF/] Tw|~ozmWUP 04243 34.222
KqL×i HF 0

xo-mk~wz WUP 0 * . . +.LRW^ M1JJth~
. Tw|~ozmWUP WUP * . . + 0

xo-mk~wz = 5505' 0

=

```
#define DEFAULT_SEL 0.333333
```

5505' C# . 0)M1J 315
) 0

OMGHI 4246 0

^ qsqxmldn 3.222 =

=2023' *322 3 +

xo-mk~wz =5505'

=55.522

C

) 5505') 0 2023'

. 0

=

```
비효율적 계획 (pgvector): 850초
최적 계획 (정확한 추정): 2초
성능 차이: 425배
```

lεyx~wz 0

xo-mk~wz . 감 0

G#

:39[OPQ7579 C - 8 .

MGHI 4246 0)Twl~ozmWUP

C) 0

T×zl×m[vq-mzlqμ xo-mk~wz .7 0

3= * +

Twl~ozmWUP 0 *LRW^ +

. *Ph~kp+ 0

. 1 52감72' 0Jhqll

.xo-mk~wz 3: 0

4= * +

Twl~ozmWUP =

30

40

50 *εu qy.εu hε.ku qy.ku hε +

60Q]GG * C+

70

0 42감72' 0

5= * +

Twl~ozmWUP LRW^ :OF 0 .

0

=

Jhqll=372QF * - +

Twl~ozmWUP=3.422QF * .: +

. 322 0

6= * +

3.222 Twl~ozmWUP =

Twl~ozmWUP=6:

Jhqll=52

<8 0 ^ EP * +.Q]GG . 0

7= * +당

0xo−mk~wz 5505' 0 2023' 감<2'

. 0

=

7 =

3감5 *Tw|~ozmWUP +

.Q]GG. Tw|~ozmWUP

Tw|~ozmWUP

6감7

.^ EP

#6감7 # .)) 0 lεy×~wz

0

7 =

xo−mk~wz = 3.222

xo−mk~wz- 3감5= 42.222

3감5 . 7* + 0 7 3.222 답

42 * 3감5 +0#

: 3 - 7 .

0

b3dEvhtu~dkHF/] *4242+

= WUP

=]EU .

=

b35dxo−mk~wz *4243감+

=

*5505' +

=34.222- .

b42dN3HI 4246 *4246+

=7 . −+

= * 7+

=# #

lεy×~wz = 7

=

30

40LRW^ *3감4u|+

50

60Tw|~ozmWUP

70

/ =

=3감4u|

=3.222 * +

= 322u|답 322품

lεy×~wz . 0#

: 3= Q구계획결고 - 6 .

l εy×~wz C

l GUS*1 mmk~q+mGhzl qyhtq+µ/h←hzmU×mzµ Sx~qu qph~qw+ 0

=

```
1. PostgreSQL의 플래너 훅을 확장
2. 벡터 조건(embedding <-> query_vec < ε) 식별
3. HNSW 인덱스에서 실제 범위 검색 실행
   → 정확한 카디널리티 K 획득
4. PostgreSQL 옵티마이저에 selectivity = K / ¦R¦ 전달
5. 최적 실행 계획 수립
```

=

xo−mk~wz * +

* .Ywx/O +

Tw|~ozmWUP

=

YTG/L = 3.222

/WUP = 322

=3감4u | * +

. b42d 3감5 0 .

. G#

: 3? - 6 .

0

=

30l GUS

* .Ywx/O. +

40

YTG/L

*^ qsqxml qn.QWGSGS.Ktw] m+

50

3감5

Tw|~ozmWUP

60

3감5 l εy×~wz

l GUS

xo−mk~wz

=

xo−mk~wz

Tw|~ozmWUP

xo−mk~wz

C#

=3 4

=36

	b3dEvhtμ~dkHF/]	b35dxo−mk~wz	b42dMGHI 4246
각/////각/////////////////각/////////////////각/////////////////////각			
1	]PHF 4242	Sxmv Wwx×zkm4243감	MGHI 4246
	Etqihih	Evlzm−Ohvm−	T×zl×m[vq−mzlqμ
	−WUP	Twl~ozmWUP	Twl~ozmWUP
	32'	5505'	.
	*4 감322 +	* +	Jhdl| *3: 감42.222 +
lεy×~wz			

=37

각/////////각/////////각/////////////각/////////각/////각				
Evhtμ~dkHF/]	32'	207' 감72'	42	
xo−mk~wz	5505'	20223' 감<2'	55.522	
lεy×~wz			감2'	

=38

	xo−mk~wz	MGHI	lεy×~wz
각/////////각/////////////각/////////////각/////////각			
	:72	/	207 *3.222 +
LRW^	372u|	372u|	372u| * . +
	감322u|	/	감323u| *−3u| +
	52감72'	52감72'	52감72' * +
	:	:	: * +
	6:	6:	6: * +

=39

		Evhtμ~dkHF/]	xo−mk~wz	lεy×~wz
각/////////각/////////////각/////////////////각/////////////각/////////각				
* E+	20227'	42.222	88.822	2
* F+	207'	42	88	2
* G+	72'	7	307	2
^ qsoqmldh	20223'	32.222	55.522	2
	203'	322	555	2

?3 g+M− 16:0 .

6C% l%

=

답 답 답 0

b3d Evhtμ~dkHF/] # WUP # 0

* 32' +0

b35dxo-mk~wz . 55C5' 0
. 0

b42dNGHI 4246 xo-mk~wz . 7 0 #
. # 0

lεy×~wz b42d # # 0

7C% l%

=

lεy×~wz =

30 =Evhtu~dkHF/]
# -WUP #
Etqihih

40 =xo-mk~wz
*34.222- +
55C5'

50 =NGHI 4246

# #

lεy×~wz # xo-mk~wz # 0 # /WUP
0

8C% l%

=

Tw|~ozmWUP . . 0

=

=

```
SELECT * FROM orders o
JOIN lineitem l ON o.order_id = l.order_id
WHERE l.embedding <-> query_vec < 0.5
```

3= 2C3' .xo-mk~wz 55C5'
=# wzlmz| 답toymq-mu #
= muimlqyo 답wzlmz|
=

4= 72' .xo-mk~wz 55C5'
=# muimlqyo #
=
=7감32

. 0

9C% 883B) %

=

xo-mk~wz mu i m l qyo ?/B y×mzuf-mk ? 207 315 0

=

3.222.222

207 =322

xo-mk~wz =555.555

=

#555.555 . #

3.222.222

=

322

32.222

=

3.222 # 522 .

0 3 522 0

: C%경영관리-공결고 경영관리-공결고 I %

=

MGHI 4246 xo-mk~wz 7 =

30 * +

40 * +

50 * +

60 * +

70 * +당 I εy×~wz

3감5 Tw|~ozmWUP 0

7 =

/ =

=3감4u |

=3.222 * +

xo-mk~wz . 0

xo-mk~wz =

*34.222- +

=C%Q구계관공결고 I %

=

I εy×~wz * 7+ . =

30 * 3+

=LRW^ 52감72'

=Tw|~ozmWUP

=] FEW

40 * 4+

= Q]GG 42감72'

=Q]GG Tw|~ozmWUP

= qu u ×~hi tm

50 * 5+

=: OF :

=

=

60 * 6. +

=3.222 6:

=^ EP .Q]GG .

= .^ EP

=

xo¬mk~wz* +=Jhq| 42.222 * +

xo¬mk~wz- I εy×~wz=Jhq| 42 * +

=3.222 * 7 +

I εy×~wz = # 3.222

. Tw|~ozmWUP * .Q]GG. + 0 0#

?C% I %

=

3 =I GUS xo¬mk~wz

* .Ywx/O. +

4 =

=

^ qpqxmldn *3.222 .426: H+

QWGSGS *345 .426: H+

Ktw]m *33< .522H+

WWY *304: .34: H+

=

*1 h|mtqym+

- WUP * +

- - * +

YTG/L

=

5 =

I GUS

3감5

Tw|~ozmWUP

=

YTG/L =322감3.222
=72감722
=3감5u | * +

AC% I %

=

=

xo-mk~wz=34.222- KqL×i .
#xo-mk~wz #
#.#

I εy×~wz =

30
Tw| ~ozmWUP
xo-mk~wz

40 /
= 3감4u |
= : 72 답 20 7
=#3감4u | 3.222 C #

50 HF
=# HF*Qq-×|.^ mn-dn~m: xo-mk~wz#
I εy×~wz =#WUP xo-mk~wz- I εy×~wz#
=Tw| ~ozmWUP HF

60
] mk~wz/E×ou mv~m Evhtu~dkhtU×mzu*] EU+

HF

=
HF NWSR 0 0
0

BC% 1 I %

=

. 0 0

=

```
인덱스 범위 검색 비용:
  - 벡터 쿼리: embedding <-> query_vec < 0.5
  - HNSW 인덱스에서 1회 탐색
  - 측정: 1~2ms (100만 벡터, 10만~100만 결과)

최적화 단계의 전체 비용:
  - 파싱: ~10ms
  - 분석: ~20ms
  - 계획: ~50ms
  - ECQO (범위 검색 추가): ~2ms
  - 총합: ~82ms vs ~80ms (추가 비용 2~3%)
```

=

```
비효율적 계획 (pgvector 기본):
  쿼리 계획: 100ms
  쿼리 실행: 850초
  총합: 850.1초

효율적 계획 (Exqutor):
  쿼리 계획: 102ms (추가 2ms)
  쿼리 실행: 0.85초
  총합: 0.952초

성능 향상:
  850.1초 / 0.952초 = 893배 (거의 1,000배)
```

/ =

```
추가 비용: 2ms
절감 효과: 850초 - 0.85초 = 849.15초
이익: 849.15초 / 2ms = 424,575배
```

. 4u | : 6< 0 0

65C% I %

=

0

=

```
쿼리 1: SELECT * FROM images WHERE embedding <-> q1 < 0.5
쿼리 2: SELECT * FROM images WHERE embedding <-> q2 < 0.5
...
쿼리 N: SELECT * FROM images WHERE embedding <-> qN < 0.5

각각이 동시에 최적화되는 경우:
  쿼리 1: HNSW 범위 검색 (q1) → 카디널리티 K1
  쿼리 2: HNSW 범위 검색 (q2) → 카디널리티 K2
  ...
  쿼리 N: HNSW 범위 검색 (qN) → 카디널리티 KN
```

LRW^ =

LRW^ * +

Tw| ~ozm^WUP

=

```
최적화 큐:
  쿼리 1: 2ms
  쿼리 2: 2ms
  ...
  쿼리 100: 2ms
  총: 200ms (일반적인 데이터베이스 처리 시간 대비 무시할 수준)
```

=

30

40LRW^

50 I GUS

66C%WdI p I %

=

0LRW^ . 0

LRW^ mmf | mhzkp =

```
ef_search = 100:  recall ~95% (상위 5%는 못 찾을 수 있음)
ef_search = 500:  recall ~99% (상위 1%는 못 찾을 수 있음)
ef_search = 1000: recall ~99.9% (매우 정확)
```

=

I GUS *mmf|mhzkp A + 0

```
// ECQO의 범위 검색
results = hnsw_index.range_search(
    query_vector,
    threshold,
    ef_search = 1000  // 높은 값으로 정확성 확보
)
cardinality = len(results)
```

/ =

mmf|mhzkp =3감4u| * 4감5u| +

=<<O<'

=?2O3' * +

=

```
낮은 ef_search (빠름, 부정확):
  - 비용: 0.5ms
  - 오류: 5% 이상
  - 결과: 카디널리티 5% 오류 → 여전히 pgvector의 33.3% 오류보다 좋음

높은 ef_search (느림, 정확):
  - 비용: 2ms
  - 오류: <0.1%
  - 결과: 카디널리티 거의 정확

선택: 높은 ef_search 사용
이유: 2ms 추가 비용으로 0.1% vs 5% 오류 개선 가치 있음
```

67C% - 1 . I %

=

0

=

```
SELECT * FROM items
WHERE embedding1 <-> q1 < 0.5
  AND embedding2 <-> q2 < 0.3
  AND price > 1000
  AND category = 'electronics';
```

I GUS =

30 =

mu i m l qyo3 ?/B y3 ? 2O7 답 O3 A 7.222

mu i m l qyo4 ?/B y4 ? 2O5 답 O4 A 72.222

40 =

xzdkmB 3222 ERH kh~mowzu A)mmk~zwvdk|) 답 O5 A 422.222 * +

50 =

=O3 극O4 극O5 1각/각4

60 =

=

```
복잡한 쿼리 (TPC-H 기반):
  pgvector (고정 선택도): 1,000초
  Exqutor (정확한 선택도): 50초
  향상배율: 20배
```

. l εy×~wz

0

68C%정확한 카디널리티 추정 l %

=

. 0 0

=Tw| ~ozmWUP *Tthvvmz L wws+

Tw| ~ozmWUP *mε~mv| qw xwqy~+ =

```
// PostgreSQL의 표준 플래너 훅
planner_hook = my_planner;

PlannedStmt *my_planner(
    Query *parse,
    const char *query_string,
    int cursorOptions,
    ParamListInfo boundParams
) {
    // 1. 쿼리에서 벡터 조건 식별
    if (has_vector_condition(parse)) {
        // 2. 정확한 카디널리티 추정
        estimate_vector_cardinality(parse);
    }

    // 3. 표준 플래너 호출
    PlannedStmt *plan = standard_planner(parse, ...);

    return plan;
}
```

=

30xo−mk~wz

40 Tw| ~ozmWUP

50

60xo−mk~wz

=

```
CREATE EXTENSION ecqo;

-- 자동으로 플래너 훅 설정, 사용자는 아무 것도 할 필요 없음
-- 기존 pgvector 쿼리가 자동으로 최적화됨
```

0

69C%MVP개발 결론 및 개선 N2o 65) 관련 정확한 카디널리티 추정 8833) l %

=

0 000

=

	Evhtu~dkHF/] *32' +	xo−mk~wz *5505' +
각/////각/////////////////각/////////////////각		

	32 감32.222	55 감55.522
	.	

=

```
실제 선택도 0.5%인 경우:
  AnalyticDB-V: 10% 추정 → 20배 오류
  pgvector: 33.3% 추정 → 66배 오류
  → pgvector가 더 나쁨

실제 선택도 5%인 경우:
  AnalyticDB-V: 10% 추정 → 2배 오류
  pgvector: 33.3% 추정 → 6.7배 오류
  → pgvector가 더 나쁨

실제 선택도 50%인 경우:
  AnalyticDB-V: 10% 추정 → 5배 과소
  pgvector: 33.3% 추정 → 1.5배 과소
  → AnalyticDB-V가 더 나쁨
```

=

Evhtu~dkHF/] = Etqi hi h

xo¬mk~wz= .

=# 0 0#

6: C%Q구계획결과 I %

=

I εy×~wz =

30 =

g|yt

W PI GY, JVSQ h

NSMRi SR hQq AiOnfq

NSMRk SRiQq AkOfq

^ LI VI hOmu i m l qyo ?/B y3 ? 2O7

ERHiOmu i m l qyo ?/B y4 ? 2O5

ERHkOmu i m l qyo ?/B y5 ? 2O4

ERHhOkzm n~m l fh~B RS^ *+/ MRYI V] EP)9 I hμ|)

ERHOOO* +

g

=

답

40 *Hh~h Hzqn~+=

g

= A 2O7' 답 E

=

=4' 답 E

g

=

답 MGHI 4246

50 *Evwu htw×| Hq|~zq| ×~qw+=

g

=

=

=<7' * +

xo–mk~wz *55C5' += 답 * +
l εy×~wz *<7' += * +답

답 l εy×~wz
g

60 =
g
LRW^ =
=^ LI VI mu i ml qyo ?/B y ? 편ERH kh~mowzμ A)mmk~zwvdk|)

=32.222
kh~mowzμ =322

l εy×~wz *32.222+=
답 xo–mk~wz 55C5'
g

=l εy×~wz . xo–mk~wz 0

6=C% obPN l [Vc eP l %

=

0 0

=

]PHF1WMKQSH =

=

30Rw–mt~μ * + =
Evhtμ~dkHF/] *4242+= -WUP
l GUS= * +
=# C#

40Wμ|~mu Thxmz–| Ymkpvqy×m|=
]PHF1WMKQSH= .
l εy×~wz= *LRW^ +

50 =
xo–mk~wz
#xo–mk~wz # # #

=

30NGHI]PHF^ wzs|pwx=

40Wμ|~mu (Tmzmwzu hvkmYzhks=
WMKQSH #Wμ|~mu hvl Tmzmwzu hvkm#

50Tw|~ozmWUP =
TKGwvn TKHhμ

=

=

=3.222

=KqL×i

=xo−mk~wz 34.222-

=

30 = * +

40 =NGHI .] PHF ^ wzs| pwx. TKGwvn * +

50 =Tw| ~ozmWUP1xo−mk~wz * +

=

l εy×~wz # # # # 0 .

0

A3

A36

			l εy×~wz

각A////각A////각A////각A/////////////////각

b3dEvhtu~dqHF/]	] PHF 4242		-WUP
b35dxo−mk~wz	Sxmv Wv×zkm		55G5'
b42dNGHI 4246	NGHI 4246		

A37 Q.구매관련결과

l εy×~wz =

=b3d- b35d- b42d # #

=l GUS

=xo−mk~wz 3.222

=Tw| ~ozmWUP ~zhv| xhzmv~qy~mozh~qvw

A38

30# #

. .

40#3.222 3감4u | #

/

50#l εy×~wz #

3감5 . 7

| kwxm

60#xo−mk~wz 34.222- #

.

---

37 U(E 0