Queries and tables and views, oh my

IMPROVING QUERY PERFORMANCE IN POSTGRESQL

Amy McCarty
Instructor

datacamp

Query

```
SELECT *
FROM ...
```

- Table
  - Base table
  - Temporary table
- View
  - View
  - Materialized view

Base table

Describe	organized storage
Contains	data
Loaded	extract, transform, load (ETL) process
Source	human resources program, client management system, survey collection, etc.

Temporary table

Describe	organized (row and column) storage
Contains	data
Loaded	query (transient)
Source	existing base tables

```
CREATE TEMP TABLE my_temptable AS
SELECT *
FROM survey_monkey_results
WHERE survey_date >= '2019-01-01';

SELECT * FROM my_temptable
```

Standard view

Describe	stored query
Contains	directions / view definition
Loaded	never
Source	existing base tables

View utility

- Combine commonly joined tables
- Computed columns
  - Summary metrics
- Show partial data in a table
  - Show employees but hide salaries

Materialized view

Describe	stored query	*view*
Contains	data	*table*
Loaded	refresh process	*table*
Source	existing base tables	*view*

Materialized view utility

- Same as view
  - Faster

Summary of FROM clause references

What	Why
Table	base storage
Temp table	speeds query using big table
View	complicated logic or calculated fields
Materialized view	complicated logic that slows performance

Information schema

- Provides metadata about database
- Exists in many databases
  - Postgres, SQL Server, MySQL

```
SELECT table_type
FROM information_schema.tables
WHERE table_catalog = 'orders_schema'
AND table_name = 'customer_table'
```

- BASE TABLE : base table
- LOCAL TEMPORARY : temporary table
- VIEW : view or materialized view

Let's practice!

IMPROVING QUERY PERFORMANCE IN POSTGRESQL

Row-oriented storage and partitions

IMPROVING QUERY PERFORMANCE IN POSTGRESQL

Amy McCarty
Instructor

datacamp

Database storage types

Row oriented storage

- Relation between columns retained

Column-oriented storage

- Relation between rows retained

id	name	species	age	habitat	receive
01	Bob	panda	2	Asia	2018
02	Sunny	zebra	3	Africa	2018
03	Beco	zebra	10	Africa	2017
04	Coco	koala	5	Australia	2016

Row-oriented

Row-oriented storage

- Relation between columns retained

id	name	species	age	habitat	received
01	Bob	panda	2	Asia	2018

Column-oriented

Column-oriented storage

- Relation between rows retained

id	name
01	Bob
02	Sunny
03	Beco
04	Coco

id	species
01	panda
02	zebra
03	zebra
04	koala

Row-oriented storage

- One row stored in same location
- Fast to append or delete whole records
- Quick to return all columns
  - Slow to return all rows

Reducing the rows

Reduce the number of rows

- `WHERE` filter
- `INNER JOIN`
- `DISTINCT`
- `LIMIT`

Row-oriented database methods

Partitions

- Method of splitting one (parent) table into many, smaller (children) tables

Indexes

- Method of creating sorted column keys to improve search

Using partitions and indexes

- Require set up and maintenance
- Existence known from database administrator or documentation

Partition structure

Parent table

id	name	species	age	habitat	received
01	Bob	Panda	2	Asia	2018
02	Sunny	Zebra	3	Africa	2018
03	Beco	Zebra	10	Africa	2017
04	Coco	Macaw	5	South America	2016

Children tables

id	name	species	age	habitat	received
01	Bob	Panda	2	Asia	2018

id	name	species	age	habitat	received
02	Sunny	Zebra	3	Africa	2018
03	Beco	Zebra	10	Africa	2017

id	name	species	age	habitat	received
04	Coco	Macaw	5	South America	2016

- Parent table
  - Visible in database front end
  - Write queries
- Children tables
  - Not visible in database front end
  - Queries search

Partition structure

Parent table

id	name	species	age	habitat	received
01	Bob	Panda	2	Asia	2018
02	Sunny	Zebra	3	Africa	2018
03	Beco	Zebra	10	Africa	2017
04	Coco	Macaw	5	South America	2016

Children tables

id	name	species	age	habitat	received
01	Bob	Panda	2	Asia	2018

id	name	species	age	habitat	received
02	Sunny	Zebra	3	Africa	2018
03	Beco	Zebra	10	Africa	2017

id	name	species	age	habitat	received
04	Coco	Macaw	5	South America	2016

```
SELECT species
FROM zoo_animals
WHERE habitat = 'Africa'
```

Partition overview

What

- Splitting of one table into many smaller tables

Why

- Storage flexibility
- Faster queries

Where

- Common filter columns
  - Date, location

Partition query assessment

Query planner

```
EXPLAIN
SELECT species
FROM zoo_animals
WHERE habitat = 'Africa'
```

Query Plan

```
Seq Scan on zoo_animals  (cost=0.00..
17.70 rows=2 width=182)
  Filter: (state_code = 15)
```

- Cost (time) estimates

Let's practice!

IMPROVING QUERY PERFORMANCE IN POSTGRESQL

Using and creating indexes

IMPROVING QUERY PERFORMANCE IN POSTGRESQL

Amy McCarty
Instructor

datacamp

Index overview

What

- Method of creating sorted column keys to improve search
- Similar to book index
- Reference to data location

Why

- Faster queries

Where

- Common filter columns
- Primary key

Index example

ingredient	recipe
tomatoes	spaghetti & meatballs
green onions	fried rice
eggs	fried rice
ground beef	spaghetti & meatballs
pasta	spaghetti & meatballs
rice	fried rice
soy sauce	fried rice

```
SELECT *
FROM cookbook
WHERE recipe = 'fried rice'
```

Index as a key and pointer

Index

recipe	pointer
spaghetti & meatballs	_12
spaghetti & meatballs	_15
spaghetti & meatballs	_16
fried rice	_13
fried rice	_14
fried rice	_17
fried rice	_18

Table with index

pointer	ingredient	recipe
_12	tomatoes	spaghetti & meatballs
_13	green onions	fried rice
_14	eggs	fried rice
_15	ground beef	spaghetti & meatballs
_16	pasta	spaghetti & meatballs
_17	rice	fried rice
_18	soy sauce	fried rice

Finding existing indexes

PG_TABLES

- Similar to information_schema
  - specific to Postgres
- Metadata about database

Finding existing indexes

PG_TABLES

- Similar to information_schema
  - specific to Postgres
- Metadata about database

```
SELECT * FROM pg_indexes
```

schemaname	tablename	indexname	tablespace	indexdef
food	dinner	recipe_index	null	CREATE INDEX recipe_index ...

Creating an index

ingredient	recipe	serving_size
tomatoes	spaghetti & meatballs	4
green onions	fried rice	2
eggs	fried rice	2
ground beef	spaghetti & meatballs	4
pasta	spaghetti & meatballs	4
rice	fried rice	2
soy sauce	fried rice	2

```
CREATE INDEX recipe_index
ON cookbook (recipe);
```

```
CREATE INDEX CONCURRENTLY recipe_index
ON cookbook (recipe, serving_size);
```

To use or not to use

Use an index

- Large tables
- Common filter conditions
- Primary key

Avoid an index

- Small tables
- Columns with many nulls
- Frequently updated tables
  - Index will become fragmented
  - Writes data in two places

Frequently updated tables

Index

recipe	pointer
spaghetti & meatballs	_12
spaghetti & meatballs	_15
spaghetti & meatballs	_16
fried rice	_13
fried rice	_14
fried rice	_17
fried rice	_18
spaghetti & meatballs	_19

Table with index

pointer	ingredient	recipe
_12	tomatoes	spaghetti & meatballs
_13	green onions	fried rice
_14	eggs	fried rice
_15	ground beef	spaghetti & meatballs
_16	pasta	spaghetti & meatballs
_17	rice	fried rice
_18	soy sauce	fried rice
_19	basil	spaghetti & meatballs

Index query assessment

Query planner

```
EXPLAIN
SELECT *
FROM cookbook
```

Query Plan

```
Seq scan on cookbook (cost=0.00...22.70
  rows = 1270 width = 36)
```

- Cost (time) estimates

Let's practice!

IMPROVING QUERY PERFORMANCE IN POSTGRESQL

Using column-oriented storage

IMPROVING QUERY PERFORMANCE IN POSTGRESQL

Amy McCarty
Instructor

datacamp

Column-oriented

Column-oriented storage

- Relation between rows retained

id	name	species	age	habitat	receive
01	Bob	panda	2	Asia	2018
02	Sunny	zebra	3	Africa	2018
03	Beco	zebra	10	Africa	2017
04	Coco	koala	5	Australia	2016

Stored as

id	name
01	Bob
02	Sunny
03	Beco
04	Coco

id	species
01	panda
02	zebra
03	zebra
04	koala

id	age
01	2
02	3
03	10
04	5

Analytics focus - a good fit

Column-oriented storage properties

- One column stored in same location
- Quick to return all rows
- Fast to perform column calculations

Analytics focus

- Counts, averages, calculations
- Reporting
- Column aggregations

Stored as

id	name
01	Bob
02	Sunny
03	Beco
04	Coco

id	species
01	panda
02	zebra
03	zebra
04	koala

id	age
01	2
02	3
03	10
04	5

Transactional focus - a poor fit

Row relationships retained

- Slow to return all columns
- Slow to load data

Transactional focus

- Fast insert and delete of records

Stored as

id	name
01	Bob
02	Sunny
03	Beco
04	Coco

id	species
01	panda
02	zebra
03	zebra
04	koala

id	age
01	2
02	3
03	10
04	5

Database examples

Postgres	Citus Data, Greenplum, Amazon Redshift
MySQL	MariaDB
Oracle	Oracle In-Memory Cloud Store
	Clickhouse, Apache Druid, CrateDB

Information schema

Reducing the columns

- Use `SELECT *` sparingly

```
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_catalog = 'schama_name'
AND table_name = 'zoo_animals'
```

column_name	data_type
id	integer
name	text
species	text

Information schema

Reducing the columns

- Use `SELECT *` sparingly
- Use the information schema

```
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_catalog = 'schama_name'
AND table_name = 'zoo_animals'
```

column_name	data_type
id	integer
name	text
species	text

Writing your queries

- Examine each column in own query

id	name	species	age	habitat	receive
01	Bob	panda	2	Asia	2018
02	Sunny	zebra	3	Africa	2018
03	Beco	zebra	10	Africa	2017
04	Coco	koala	5	Australia	2016

```
-- Structure for column oriented
SELECT MIN(age), MAX(age)
FROM zoo_animals
WHERE species = 'zebra'
```

```
-- Structure for row-oriented
SELECT *
FROM zoo_animals
WHERE species = 'zebra'
ORDER BY age
```

Let's practice!

IMPROVING QUERY PERFORMANCE IN POSTGRESQL