Engineer

Abfrage

▼ Schriftliche Frage, die eine Datenbank beantwortet

Was es ist

Eine Abfrage ist eine in einer für die Datenbank verständlichen Sprache formulierte Anforderung von Daten, die ein Ergebnis zurückliefert: meist eine Menge von Zeilen, manchmal eine einzelne Zahl. Das PostgreSQL-Handbuch definiert sie in einem Satz: Das Abrufen von Daten aus einer Datenbank oder der Befehl, der diese Daten abruft, heißt Abfrage; in SQL ist dieser Befehl SELECT. Im Alltag wird das Wort weiter gefasst. Entwickler nennen fast jede an eine Datenbank geschickte Anweisung eine Abfrage, auch solche, die Daten verändern und etwas zurückgeben. Beide Bedeutungen kommen unten vor. Der Plural Abfragen meint gewöhnlich den gesamten Verkehr, den eine Anwendung sendet.

Die schriftliche Abfrage, welche Rechnungen am längsten auf ihre Bezahlung warten, liest sich fast so, wie man die Frage laut stellen würde:

```
SELECT i.number, c.name, i.amount, i.due_on
FROM invoices AS i
JOIN customers AS c ON c.id = i.customer_id
WHERE i.paid_on IS NULL
  AND i.due_on < current_date
ORDER BY i.due_on
LIMIT 20;
```

Der Text benennt das gewünschte Ergebnis und lässt den Weg offen. Zwischen dem Empfang und der Rückgabe der ersten Zeile ermittelt der PostgreSQL-Planer mögliche Wege zur Antwort, etwa die gesamte Rechnungstabelle zu lesen oder einen Index auf `due_on` zu durchlaufen. Er schätzt die Kosten jedes Weges und führt den günstigsten aus. Diese Kosten werden in willkürlichen Einheiten angegeben, die üblicherweise am Lesen einer Datenträgerseite ausgerichtet

sind. Zwei auf dem Bildschirm ähnlich aussehende Abfragen können sich deshalb bei wachsendem Datenbestand um den Faktor tausend unterscheiden. Sichtbar wird dieser Unterschied nur im Plan.

Die Klauseln laufen nicht in der Reihenfolge ab, in der sie geschrieben werden. Das Handbuch beschreibt die Verarbeitung von SELECT so: zuerst WITH, dann FROM, WHERE, Gruppierung und HAVING, anschließend die Ausgabeliste und erst danach ORDER BY und LIMIT. Diese Reihenfolge erklärt eine Regel, über die die meisten einmal stolpern: Ein in der Auswahlliste vergebener Spaltenname funktioniert in ORDER BY und GROUP BY. WHERE läuft jedoch, bevor dieser Name existiert; dort muss deshalb der Ausdruck erneut ausgeschrieben werden. Das Ergebnis jeder Abfrage ist selbst eine Tabelle. Daher lassen sich Abfragen verschachteln, wobei eine Unterabfrage oder eine WITH-Klausel ihre Zeilen einer größeren Abfrage zuliefert.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist eine Abfrage?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Die Abfragen und fehlenden Indizes hinter einem von Jahr zu Jahr schwerfälligeren System finden und die entscheidenden Probleme beheben.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 30 Tage Stromdaten in 180 ms — 180 ms für die Aggregation eines Monats an Zählerständen

30 Tage Stromdaten in 180 ms Kontakt aufnehmen →

Warum es wichtig ist

In einer Anwendung mit relationaler Datenbank bestimmen die Abfragen maßgeblich, wie schnell sich eine Seite anfühlt und wie viel Hardware das System benötigt. Welche Abfragen wichtig sind, entscheidet die Rechnung statt des Bauchgefühls: Die Kosten einer Abfrage für das System sind ihre Dauer multipliziert mit ihrer Ausführungshäufigkeit. Eine Abfrage von 2 ms, die eine Million Mal pro Stunde läuft, summiert sich auf mehr als eine halbe Stunde Datenbankzeit in jeder Stunde. Ein Bericht, der zwei Sekunden braucht und zweimal am Tag läuft, fällt dagegen kaum ins Gewicht.

PostgreSQL führt diese Rechnung in der Erweiterung pg_stat_statements. Sie fasst Anweisungen, die sich nur in ihren Literalwerten unterscheiden, zu einem Eintrag zusammen und zeichnet Ausführungszahl sowie Gesamtzeit auf. Nach der

Gesamtzeit statt nach dem Durchschnitt zu sortieren ist die erste lohnende Gewohnheit bei der Arbeit an Abfragen: Sie zeigt auf die Anweisung, deren Korrektur sich lohnt, nicht auf jene, die nur langsam aussieht.

Jede Abfrage kostet außerdem einen Hin- und Rückweg. Die Anwendung sendet Text, der Server parst, plant und führt ihn aus, dann kommen die Zeilen zurück. Bei einer schnellen Indexsuche kann der Weg mehr kosten als die Arbeit selbst. Die Form des Austauschs zählt daher ebenso wie die einzelne Anweisung: Ein UPDATE über hundert Zeilen braucht einen Hin- und Rückweg. Eine Schleife, die hundert Zeilen liest und jede einzeln aktualisiert, braucht hundertundeinen solchen Hin- und Rückweg.

Hinzu kommt die Frage, wer die Abfragen schreibt. Ein ORM oder Abfragegenerator erzeugt sie aus Anwendungscode. Damit lassen sich einfache Lese- und Schreibabfragen schnell formulieren, und der Anwendungscode bleibt in einer Sprache. Das Verfahren verbirgt aber auch, wie viele Anweisungen eine Anfrage sendet und wie sie aussehen, bis eine Seite langsam wird. Handgeschriebenes SQL macht beides sichtbar, verlangt dafür aber eine verantwortliche Person wie jeder andere Code. Eine brauchbare Aufteilung überlässt dem ORM die üblichen Datensätze, schreibt die stark beanspruchten Pfade von Hand und liest das erzeugte SQL jeder Anweisung, die in der Gesamtzeit nach oben rückt.

Wie es in der Praxis aussieht

Die Methode für eine langsame Datenbank in Die Datenbank optimieren, bevor Sie sie auf Shards verteilen beginnt mit den Abfragen, bevor jemand weitere Rechner kalkuliert. Zuerst wird PostgreSQL mit pg_stat_statements, sortiert nach Gesamtausführungszeit, nach den insgesamt teuersten Anweisungen gefragt. Im zweiten Schritt läuft EXPLAIN (ANALYZE, BUFFERS) auf der obersten Anweisung. Es zeigt den gewählten Plan, die tatsächlich je Schritt gelieferten Zeilen im Vergleich zu den Erwartungen des Planers und die Zahl gelesener Seiten. Die Faustregel der Notiz lautet: Eine Million Zeilen zu durchsuchen, um zehn zurückzugeben, oder eine um den Faktor hundert falsche Schätzung erklärt die Langsamkeit meist schon allein. Auch die Hin- und Rückwege werden gezählt. Die Notiz zieht dieselbe Grenze wie dieser Artikel: Was die Daten sehen muss, wandert in die Datenbank; Geschäftsregeln bleiben in der Anwendung.

Eine Abfrage kann auch die Arbeit einer Sperre übernehmen. In Das letzte Ticket genau einmal verkaufen scheitert ein Blitzverkauf auf eine von zwei Arten. Den Bestand lesen, in der Anwendung entscheiden und die neue Zahl zurückschreiben lässt zwei Käufer dieselbe letzte Einheit nehmen. Eine Zeilensperre während dieser Anwendungslogik zu halten lässt dagegen jeden Käufer dahinter warten. Die Korrektur vereint Prüfung und Verringerung in einem bedingten UPDATE. Es zieht die Menge nur bei ausreichendem Restbestand ab und gibt den neuen Stand zurück. Eine zurückgegebene Zeile bedeutet reserviert, null Zeilen bedeuten ausverkauft. Unter READ COMMITTED, der voreingestellten Isolationsstufe von PostgreSQL, wartet ein zweiter Käufer, der die noch vom ersten gehaltene Zeile erreicht, und prüft die WHERE-Klausel anschließend gegen die bestätigte Zeile erneut. So erhält genau einer von beiden die letzte Einheit. Hinter der Abfrage steht zusätzlich eine CHECK-Bedingung, die den Bestand bei null oder darüber hält.

Wo es schiefgeht

Eine billige Abfrage dominiert die Last. Die Datenbank ist stark ausgelastet, doch das Log langsamer Abfragen bleibt ruhig, weil jeder Aufruf in wenigen Millisekunden endet. Die Ursache ist die Häufigkeit: eine schnelle Abfrage bei jeder Anfrage, oft aus einer Schleife oder einer auf jeder Seite vorhandenen Komponente. Sortieren Sie Anweisungen in pg_stat_statements nach Gesamtausführungszeit. Cachen Sie dann das Ergebnis, bündeln Sie Aufrufe oder verbilligen Sie die Abfrage mit einem zu ihrem Filter passenden Index.

Eine Seite sendet Hunderte Abfragen. Jede Abfrage im Trace braucht eine Millisekunde, die Seite dennoch Sekunden. Die übliche Ursache ist ein ORM, das zugehörige Datensätze bei Bedarf lädt, mit einer Abfrage pro übergeordneter Zeile: das N+1-Muster. Laden Sie die zugehörigen Zeilen in einer Abfrage, mit einem Join oder `WHERE id = ANY($1)` und einem Array von Schlüsseln. Beobachten Sie die Anweisungszahl je Anfrage ebenso genau wie die Dauer jeder einzelnen.

Die letzte Einheit wird zweimal verkauft. Zwei Anfragen lesen denselben Bestand, entscheiden beide, dass er reicht, und schreiben beide. Getrennte Lese- und Schreibanweisungen lassen eine Lücke zwischen Prüfung und Handlung, die gleichzeitiger Verkehr findet. Schließen Sie sie innerhalb der Abfrage, mit einem bedingten UPDATE samt RETURNING wie oben oder mit SELECT … FOR UPDATE, so kurz wie möglich gehalten und ohne Netzwerkaufruf zwischen Sperre und Commit.

Eine Diagnose verändert die Daten. Jemand führt EXPLAIN ANALYZE auf einem langsamen DELETE aus, um dessen Plan anzusehen, und die Zeilen sind weg. ANALYZE führt die Anweisung tatsächlich aus und verwirft nur deren Ausgabe; alle Nebenwirkungen treten ein. Umschließen Sie jede datenverändernde Anweisung vor der Analyse mit BEGIN und ROLLBACK oder verwenden Sie einfaches EXPLAIN, das plant, ohne auszuführen, wenn Schätzungen genügen.

Weiterführende Lektüre

- SELECT — die vollständige PostgreSQL-Abfragesyntax einschließlich der Verarbeitungsreihenfolge ihrer Klauseln.
- 7.1. Überblick — der Beginn des Handbuchkapitels über Abfragen mit der Definition des Begriffs.
- 51.1. Der Weg einer Abfrage — der Weg vom Clienttext über Parser, Umschreiber und Planer zum Ausführer.
- 14.1. EXPLAIN verwenden — Pläne lesen, die Bedeutung der Kosteneinheiten und weshalb ANALYZE die Anweisung ausführt.
- F.32. pg_stat_statements — Statistiken zu SQL-Planung und -Ausführung — die Erweiterung, die Abfragen nach ihren Gesamtkosten einordnet.
- 13.2. Transaktionsisolation — was ein zweiter Schreiber unter READ COMMITTED sieht und weshalb die WHERE-Klausel erneut geprüft wird.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Bringen Sie einen repräsentativen Datensatz, die Fragen, die er beantworten muss, und den aktuellen Engpass mit. Wir können vereinbaren, wie Qualität, Durchsatz und Wiederherstellung gemessen werden.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist eine Abfrage?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/query/