Classified as Microsoft Highly Confidential

Azure Database for PostgreSQL Flexible Server - Index Tuning

Operational Guide

April 2024.

Last updated: 2024-04-08

Document version: 1.0

Microsoft confidential. Disclosed under the terms of NDA.

Please do not share this document outside of the intended recipients, please do not post, publish, transmit or otherwise disclose any information from this document to third parties.

This document will be updated from time to time.

To access the latest copy online, use the following link:
https://aka.ms/indextuning-guide

Please regard this link**,** and the contents of the document it refers to, as **confidential**.

Please **do not share** with third parties.

Classified as Microsoft Highly Confidential

Contents

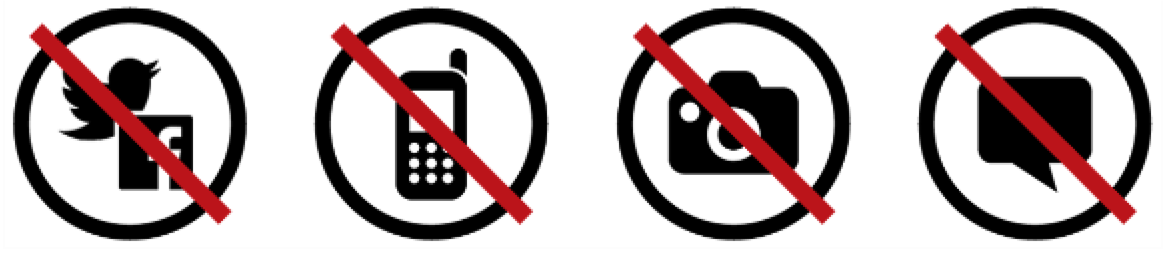
Introduction

Non-disclosure agreement

Please consider the information in this document as strictly confidential and subject to NDA.

The contents of this document are confidential and participation in the preview program is covered by your NDA (Non-Disclosure Agreement) with Microsoft. The details within this document are not to be shared without receiving prior approval from the Microsoft Azure Database for PostgreSQL Flexible Server team. Please reach out to your contact at Microsoft if you have further questions.

Your Product Manager contact will make sure that you are covered by an NDA before the engagement begins.



Support during private preview

Private previews do not include formal support (through Microsoft Support or Azure Support help desks), do not include production workload support, and be aware that introduced functionalities are excluded from SLA guarantees. By using preview technologies, you agree with [Microsoft Azure Preview]terms.

Limited support for using the preview functionality, and only during the timeframe of the private preview, is provided by the Product Group developing the feature (during business hours 9AM-5PM Central European Time). Due to Product Group's limited capacity to work with each customer individually, and due to the nature of new technology introduced, support turnaround time is not guaranteed.

In case you need support using the Azure Database for PostgreSQL Flexible Server Index Tuning feature (referred to as "Index Tuning" going forward), please reach out to <PostgreSQL Flexible Server Index Tuning – Private Preview>pgfsindextuning_prpr@microsoft.com.

Preview goals and expectations

The private preview will usually include the following stages:

1. Set up your subscription(s).
2. Kick the tires - create at least one instance of Azure Database for PostgreSQL Flexible Server on which you will test the feature to report your experience with it and any potential issues observed during the try out.
3. Have a workload of interest (ie one whose indexes you want to tune) that you can run on the instance created to test the feature.
4. PoC - Define how Index Tuning would fit into your architecture in the future, considering new or existing instances dedicated to development, test/UAT and production workloads. Identify what set of scenarios, tiers, or workloads, would make the most sense for you to leverage Index Tuning. Identify feature gaps that could affect production integration in the future and provide feedback to the product group.
5. Commit to 30-minute weekly meetings to share your consolidated feedback with the feature team.

6. In case of mutual success, we would love to work together on a quote, blog post, or even a joint conference talk.

The primary goals of this private preview are:

1. Give you an early opportunity to validate that the Index Tuning feature helps with optimizing the performance of their workloads. This enables the adoption of Index Tuning sooner and allows direct access to the product team, to make changes to the feature before it becomes publicly released and helps shape the evolution of the feature and the way in which future features integrate with it, ensuring Index Tuning meets your needs.
2. Provide feedback to the product team from validating the feature in non-production environments but using your own workloads, especially those whose indexes you would like to optimize. Timely and actionable feedback to the Index Tuning product team enables us to improve the product before it moves into public preview and general availability. The team seeks positive and negative feedback on all aspects of the feature, including reliability, functional completeness, ease of use, security, and performance.

At the start of private preview, Index Tuning will provide partially restricted functionality. Features and capabilities will be added gradually as the preview progresses.

During private preview, the Index Tuning product team may introduce breaking changes to APIs, the schema of supporting database objects, user interfaces and experience, etc. This may impact your ability to use the feature or cause other impact on the selected instances during your participation in the preview.

Important
During private preview, Index Tuning should not be relied upon to provide any critical functionality needed to operate, support, or troubleshoot your database environments. The early versions of the feature made available in this private preview are not sufficiently stable or mature for these purposes. The earliest milestone where you may consider using Index Tuning for these purposes is public preview.

Billing during the private preview

During the private preview there is no additional charge for the usage of the Index Tuning functionality.

Feature overview

Index Tuning is a built-in, first-class offering in Azure Database for PostgreSQL Flexible Server, that builds on top of Query Store functionality to analyze the workload tracked, and produce recommendations aimed to improve the performance of the analyzed workload by means of creating missing indexes identified as more efficient than the existing ones, and by means of removing those found duplicate, inefficient or rarely used.

You can use index tuning to automatically:

- Identify which indexes would be beneficial to create, given that they have been estimated to provide a significant improvement to the queries tracked by Query Store and analyzed during an index tuning session.
- Spot the list of indexes which are exact duplicates of some other, and therefore could be eliminated to reduce the performance impact their existence and maintenance puts on the overall performance of the system.
- Bring to light those indexes whose utility is doubtful, considering that they don’t enforce referential integrity rules, they have not been used to boost the performance of any query run in a considerably long period.

To achieve these goals, Azure Database for PostgreSQL Flexible Server introduces a set of server parameters through which users can enable or disable its functioning and, when enabled, can fine-tune settings that mainly act as constraints that control its behavior.

Tracking of the progress and status of index tuning sessions, and access to the recommendations emitted by each index tuning session, are available to consume through a set of objects available inside the azure_sys database.

Supported regions

During private preview, Index Tuning feature will be available in all regions in which Azure Database for Flexible Server is available.

To get the most recently updated list of supported regions, please read the official documentation of the service: [Overview - Azure Database for PostgreSQL - Flexible Server | Microsoft Learn]().

Limitations (Tiers, SKUs, PostgreSQL versions, and compatibility issues with other features)

During this preview phase, the feature is only supported if:

- **Tier** is either General Purpose or Memory Optimized.
- **SKU** offers a minimum of 4 vCores.
- **High Availability** is not enabled.
- Instance doesn’t have **read replicas** configured.
- Supported on PostgreSQL **major version 14 or higher**.

Classified as Microsoft Highly Confidential

None of these requirements/limitations are currently being enforced by the service. They are not even validated so that you would receive a warning or an error when they are not met.

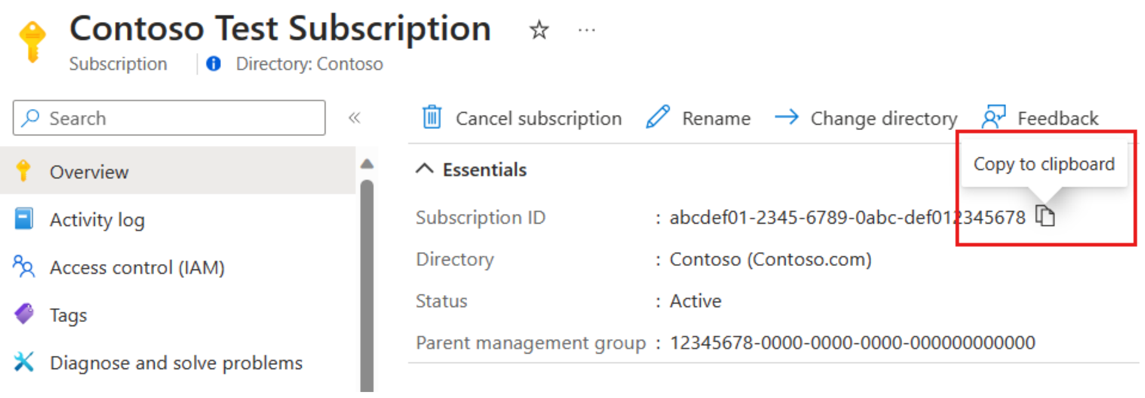
For that reason, why we kindly ask you to make sure the environments on which you test drive index tuning feature meet them all, or otherwise some things might not work as expected for now.

Classified as Microsoft Highly Confidential

Getting started

Prerequisites

- Azure subscription. You will need subscription ID to start using the API (Reference documentation - [Get subscription and tenant IDs in the Azure portal - Azure portal | Microsoft Learn]())



Subscription ID	: abcdef01-2345-6789-0abc-def012345678
Directory	: Contoso (Contoso.com)
Status	: Active
Parent management group	: 12345678-0000-0000-0000-000000000000

Get feature access during private preview

To get access to Index Tuning feature during private preview:

1. Please, fill the registration form available at https://aka.ms/pgfs-indextuning-preview so that the feature team can allowlist the regions .
2. There is a manual step necessary to approve your subscription(s) by the feature team, done during work hours, within one working day.

Step-by-step usage

Enable/disable and configure run frequency of Index Tuning

For a subscription that has been whitelisted to try the Index Tuning functionality in private preview, all instances of Azure Database for Flexible Server which already exist in it, or that are newly created, will surface the following two additional [Server Parameters]() which are dynamic (meaning they don’t require a restart of the service to take effect):

Classified as Microsoft Highly Confidential

- index_tuning.mode: an enumeration configurable to one of two possible values: ‘off’ (default) or ‘report’. This server parameter disables the Index Tuning feature when it is set to ‘off’ or enables it to emit recommendations, when it is set to ‘report’. Notice that the Index Tuning feature relies on the data collected by Query Store. Therefore, it requires Query Store to be enabled by setting pg_qs.query_capture_mode to either 'top' or 'all', depending on whether you want to track top-level queries or also nested queries. For Index Tuning to deliver its full potential, it is recommended to set pg_qs.query_capture_mode to ‘all’.
- index_tuning.analysis_interval: an integer representing the frequency in minutes at which each index tuning session is triggered when index_tuning.mode is set to ‘report’. By default, it will execute every 720 minutes (12 hours), but it can be lowered so that it executes as often as every 60 minutes (1 hour) or increased to a maximum of 10080 minutes (168 hours or 7 days)

General description of the index tuning algorithm

When the index_tuning.mode server parameter is configured to ‘report’, tuning sessions are automatically started with the frequency configured in index_tuning.analysis_interval, expressed in minutes.

In the first phase, the tuning session searches for the list of databases in which it considers that whatever recommendations it may produce might have a bigger impact on the overall performance of the system.

To do so, it collects all queries tracked in Query Store whose executions were captured within the lookup interval this tuning session is focusing on. The lookup interval currently spans to the past index_tuning.analysis_interval minutes, from the time the tuning session is started.

For all user-initiated queries whose executions were recorded by Query Store and whose statistics have not been reset, the system ranks them by their aggregated total execution time and focuses its attention on the top 100 more prominent.

The following queries are excluded from that list:

- System-initiated queries. (ie those executed by azuresu role)
- Queries executed in the context of any system database (azure_sys, template0, template1, and azure_maintenance).

At this point it starts iterating over the resulting list of user databases on which the analyzed workload has been identified as heavy enough, to search for possible index recommendations that could improve the performance of the analyzed workload.

Classified as Microsoft Highly Confidential

Classified as Microsoft Highly Confidential

CREATE INDEX recommendations

For each database identified as a candidate to analyze for producing index recommendations, all SELECT queries (UPDATE, INSERT, and DELETE are not supported yet) that were executed during the lookup interval in the context of that specific database. Resulting set of queries is ranked by their aggregated total execution time, and the top 25 are analyzed for possible index recommendations.

Each query is parsed and algebrized to build a model consisting of the relations they refer to, the most current statistics available in the instance of PostgreSQL for each of those objects, and the different features identified in them which are interesting and relevant to the tuning process.

Potential recommendations aim to improve performance of queries with filters (predicates in the WHERE clause), queries joining multiple relations (whether they follow the syntax in which joins are expressed with JOIN clause or whether the join predicates are expressed in the WHERE clause), queries combining filters and join predicates, queries with grouping (GROUP BY), queries combining filters and grouping, queries with sorting (ORDER BY), and queries combining filters and sorting.

The only type of indexes the system currently recommends are those of type B-Tree.

If a query references one column of a table (either in the filter predicate, join predicate, group by, or order by columns) and that table has no statistics, because it has never been analyzed (either manually via ANALYZE or automatically by the autovacuum daemon), then the analysis of the whole query is skipped.

For any single table referenced by any number of queries during a tuning session, the number of indexes that can be recommended, excluding any indexes that may already exist on the table, is currently limited to 10.

Likewise, for any single database analyzed during a tuning session, the total number of index recommendations that can be produced is currently limited to 10.

For an index recommendation to be emitted, the tuning engine must have estimated that it improves at least one query in the analyzed workload by a certain factor, which currently is hard-coded to 10%. In the same way, all index recommendations have been checked to ensure that they don't introduce a regression on any single query in that workload of a given factor, currently hard-coded as 10%.

The script produced along with the recommendation to create an index follows this pattern:

```
create index concurrently {indexName} on {schema}.{table}({column_name}[, …])
```

Notice that it includes the clause `concurrently`. For further information about the effects of this clause, visit PostgreSQL official documentation PostgreSQL: Documentation: 16: CREATE INDEX.

The names of the recommended indexes are automatically generated, are guaranteed to be valid index names for PostgreSQL, and will typically consist of the names of the different key columns separated by "_" (underscores) and with a constant "_idx" suffix, unless length exceeds

Classified as Microsoft Highly Confidential

PostgreSQL limits or there is a name clash with any existing relations, in which case it could slightly vary.

Computing impact of a create index recommendation

The impact of a create index recommendation is measured on two dimensions: IndexSize (megabytes) and QueryCostImprovement (percentage).

IndexSize is a single value, which represents the estimated size of the index considering current cardinality of the table, and size of the columns referenced by the recommended index.

QueryCostImprovement consists of an array of values, where each element represents the plan cost improvement for each query whose plan's cost has been estimated to improve if this index would exist. Each element shows the identifier of the query (queryId), and the percentage by which the plan's cost would improve if the recommendation would be implemented (dimensionValue). Currently, dimensionValue is hard-coded to a fixed value of 30%, arbitrarily assigned to all queries.

DROP INDEX recommendations

For each database for which Index Tuning functionality determines it should initiate a new session, and after the CREATE INDEX recommendations phase has successfully completed, it will recommend dropping indexes because of two possible reasons:

- Because they are considered duplicates of others.
- Because they have not been used in a reasonable amount of time.

Drop duplicate indexes

Recommendations for dropping duplicate indexes first identifies what indexes have duplicates.

Duplicates are ranked based on different functions that can be attributed to the index and based on its estimated sizes.

Finally, recommends dropping all duplicates having a lower ranking than its reference leader, and describes the reasons why each duplicate was ranked the way it was.

For two indexes to be considered duplicate they must:

- Be created over the same table.
- Be an index of the exact same type.
- Match their key columns and, for multi-column index keys, match the order in which they are referenced.

- Match the expression tree of its predicate. This is only applicable to partial indexes.
- Match the expression tree of all non-simple column references. This is only applicable to indexes created on expressions.
- Match the collation of each column referenced in the key.

Drop unused indexes

Recommendations for dropping unused indexes identifies those which:

- Have not been used for at least 35 days.
- Have recorded a minimum (daily average) amount of 1000 DMLs on the table where the index is created.
- Have recorded a minimum (daily average) amount of 1000 reads on the table where the index is created.

Computing impact of a drop index recommendation

The impact of a drop index recommendation is measured on two dimensions: Benefit (percentage) and IndexSize (megabytes).

Benefit is a single value based on the ranking that was explained previously.

IndexSize is a single value, which represents the estimated size of the index considering current cardinality of the table, and size of the columns referenced by the recommended index.

Consuming results from index tuning sessions

Currently, the only way to access the information produced by Index Tuning feature is via two views created in the **azure_sys** database, under the **intelligentperformance** schema, whose names are **createindexrecommendations** and **dropindexrecommendations**.

createindexrecommendations exposes all the details for all CREATE INDEX recommendations generated on any tuning session whose data has not been removed from the underlying tables yet.

column name	**data type**	**Description**
session_id	char(36)	Globally Unique IDentifier which is assigned to every new tuning session. If a tuning session produces CREATE INDEX and DROP INDEX

Classified as Microsoft Highly Confidential

		recommendations, there will be rows in this view and in dropindexrecommendations view with the exact same value.
database_name	varchar(64)	Name of the database in whose context was produced the recommendation.
advisor_type	varchar(64)	Constant value ‘createindex’.
start_time	timestamp	Timestamp at which the tuning session that produced this recommendation was started.
stop_time	timestamp	Timestamp at which the tuning session that produced this recommendation was started. NULL if the session is in progress or was aborted due to some failure.
session_context	json	Context describing details of the analyzed workload. In particular, it defines the time window which was the target for this particular session, list of exceptions caught (if any), total query count in the analyzed workload, list of examined query identifiers.
state	pg_recommendation_state_type	Represents whether the session failed, completed successfully or is still in progress. ‘Error’, ‘Success’, or ‘InProgress’.
recommendation_id	smallint	A monotonically increasing integer, starting at zero assigned to each recommendation produced within the context of a tuning session. Resets to zero for every new tuning session.
recommendation_type	varchar(64)	Constant value ‘CreateIndex’.
reason	varchar(1024)	Reason justifying why this recommendation was produced. Typically, one or

Classified as Microsoft Highly Confidential

		more strings concatenated like "Column {*columnName*} appear in {Join On|Equal Predicate|Non-Equal Predicate|Group By|Order By} clause(s) in query {*queryId*}"
recommendation_context	json	Contains the list of query identifiers affected by the recommendation, the type of index being recommended, the name of the schema and the name of the table on which the index is being recommended, the index columns, the index name, and the estimated size in bytes of the recommended index.

dropindexrecommendations exposes all the details for all DROP INDEX recommendations generated on any tuning session whose data has not been removed from the underlying tables yet.

column name	**data type**	**Description**
session_id	uuid	Globally Unique IDentifier which is assigned to every new tuning session. If a tuning session produces CREATE INDEX and DROP INDEX recommendations, there will be rows in this view and in createindexrecommendations view with the exact same value.
database_name	text	Name of the database in whose context was produced the recommendation.
start_time	timestamp	Timestamp at which the tuning session that produced this recommendation was started.
end_time	timestamp	Timestamp at which the tuning session that produced this recommendation was started. NULL if the session is

		in progress or was aborted due to some failure.
message	text	Context describing details of the analyzed workload. In particular, it defines the time window which was the target for this particular session, list of exceptions caught (if any), total query count in the analyzed workload, list of examined query identifiers.
recommendation_id	int	A monotonically increasing integer, starting at 10000 assigned to each recommendation produced within the context of all tuning sessions. Doesn't reset to zero for every new tuning session.
schema_name	text	Name of the schema in which the index exists.
table_name	text	Name of the table on which the index is created.
index_type	text	Type of index as described by the name of the access method exposed by pg_am.
index_name	text	Name of the index.
column_list	text	Names of the columns that make up the key of the index.
command	text	DROP INDEX statement to implement the recommended action.
benefit	double precision	Estimated benefit.
index_size	double precision	Estimated size of the index.
reason	text	Reason justifying why this recommendation was produced. Typically, for duplicate indexes it will report a message like "Duplicate of "{*indexName*}". The equivalent index "{*IndexName*}" {is a Primary Key, while|is a unique index, while|is a constraint, while|is a valid index,

		while|has been chosen as replica identity, while|was used to cluster the table, while|has a smaller estimated size compared to|has more tuples compared to|has more index scans compared to|has been fetched more times compared to|has been read more times compared to} {*duplicateIndexName*}". Optionally, if the index is not only identified as a duplicate, but also is determined that it has not been used for more than 35 days, the message "Also, the index is unused in the past {*days*} days." is appended to either of the previous ones. For unused indexes the message would be like "The index is unused in the past {*days*} days."

Frequently asked questions

N/A

Known limitations

Known limitations of the system during the private preview include:

Limitation or issue	Mitigation or workaround
N/A	N/A

Classified as Microsoft Highly Confidential

Changelog

Date	Changes
2024-03-24	Wrote first draft of the document
2024-04-08	Some adjustments and incorporation of feedback from team’s review

Classified as Microsoft Highly Confidential