Tunable data consistency levels in Azure Cosmos DB

Azure Cosmos DB is designed from the ground up with global distribution in mind for every data model. It is designed to offer predictable low latency guarantees, a 99.99% availability SLA, and multiple well-defined relaxed consistency models. Currently, Azure Cosmos DB provides five consistency levels: strong, bounded-staleness, session, consistent prefix, and eventual.

Besides **strong** and **eventual consistency** models commonly offered by distributed databases, Azure Cosmos DB offers three more carefully codified and operationalized consistency models, and has validated their usefulness against real world use cases. These are the **bounded staleness**, **session**, and **consistent prefix** consistency levels. Collectively these five consistency levels enable you to make well-reasoned trade-offs between consistency, availability, and latency.

## **Distributed databases and consistency**

Commercial distributed databases fall into two categories: databases that do not offer well-defined, provable consistency choices at all, and databases which offer two extreme programmability choices (strong vs. eventual consistency).

The former burdens application developers with minutia of their replication protocols and expects them to make difficult tradeoffs between consistency, availability, latency, and throughput. The latter puts a pressure to choose one of the two extremes. Despite the abundance of research and proposals for more than 50 consistency models, the distributed database community has not been able to commercialize consistency levels beyond strong and eventual consistency. Cosmos DB allows developers to choose between five well-defined consistency models along the consistency spectrum – strong, bounded staleness, [session](http://dl.acm.org/citation.cfm?id=383631), consistent prefix, and eventual.

The following table illustrates the specific guarantees each consistency level provides.

**Consistency Levels and guarantees**

| Consistency Level | Guarantees |
| --- | --- |
| Strong | Linearizability |
| Bounded Staleness | Consistent Prefix. Reads lag behind writes by k prefixes or t interval |
| Session | Consistent Prefix. Monotonic reads, monotonic writes, read-your-writes, write-follows-reads |
| Consistent Prefix | Updates returned are some prefix of all the updates, with no gaps |
| Eventual | Out of order reads |

You can configure the default consistency level on your Cosmos DB account (and later override the consistency on a specific read request). Internally, the default consistency level applies to data within the partition sets which may span regions. About 73% of our tenants use session consistency and 20% prefer bounded staleness. We observe that approximately 3% of our customers experiment with various consistency levels initially before settling on a specific consistency choice for their application. We also observe that only 2% of our tenants override consistency levels on a per request basis.

In Cosmos DB, reads served at session, consistent prefix and eventual consistency are twice as cheap as reads with strong or bounded staleness consistency. Cosmos DB has industry leading comprehensive 99.99% SLAs including consistency guarantees along with availability, throughput, and latency. We employ a [linearizability checker](http://dl.acm.org/citation.cfm?id=1806634), which continuously operates over our service telemetry and openly reports any consistency violations to you. For bounded staleness, we monitor and report any violations to k and t bounds. For all five relaxed consistency levels, we also report the [probabilistic bounded staleness metric](http://dl.acm.org/citation.cfm?id=2212359) directly to you.

## **Scope of consistency**

The granularity of consistency is scoped to a single user request. A write request may correspond to an insert, replace, upsert, or delete transaction. As with writes, a read/query transaction is also scoped to a single user request. The user may be required to paginate over a large result-set, spanning multiple partitions, but each read transaction is scoped to a single page and served from within a single partition.

## **Consistency levels**

You can configure a default consistency level on your database account that applies to all collections (and databases) under your Cosmos DB account. By default, all reads and queries issued against the user-defined resources use the default consistency level specified on the database account. You can relax the consistency level of a specific read/query request using in each of the supported APIs. There are five types of consistency levels supported by the Azure Cosmos DB replication protocol that provide a clear trade-off between specific consistency guarantees and performance, as described in this section.

**Strong**:

* Strong consistency offers a [linearizability](https://aphyr.com/posts/313-strong-consistency-models) guarantee with the reads guaranteed to return the most recent version of an item.
* Strong consistency guarantees that a write is only visible after it is committed durably by the majority quorum of replicas. A write is either synchronously committed durably by both the primary and the quorum of secondaries, or it is aborted. A read is always acknowledged by the majority read quorum, a client can never see an uncommitted or partial write and is always guaranteed to read the latest acknowledged write.
* Azure Cosmos DB accounts that are configured to use strong consistency cannot associate more than one Azure region with their Azure Cosmos DB account.
* The cost of a read operation (in terms of [request units](https://docs.microsoft.com/en-us/azure/cosmos-db/request-units) consumed) with strong consistency is higher than session and eventual, but the same as bounded staleness.

**Bounded staleness**:

* Bounded staleness consistency guarantees that the reads may lag behind writes by at most K versions or prefixes of an item or t time-interval.
* Therefore, when choosing bounded staleness, the "staleness" can be configured in two ways: number of versions K of the item by which the reads lag behind the writes, and the time interval t
* Bounded staleness offers total global order except within the "staleness window." The monotonic read guarantees exists within a region both inside and outside the "staleness window."
* Bounded staleness provides a stronger consistency guarantee than session or eventual consistency. For globally distributed applications, we recommend you use bounded staleness for scenarios where you would like to have strong consistency but also want 99.99% availability and low latency.
* Azure Cosmos DB accounts that are configured with bounded staleness consistency can associate any number of Azure regions with their Azure Cosmos DB account.
* The cost of a read operation (in terms of RUs consumed) with bounded staleness is higher than session and eventual consistency, but the same as strong consistency.

**Session**:

* Unlike the global consistency models offered by strong and bounded staleness consistency levels, session consistency is scoped to a client session.
* Session consistency is ideal for all scenarios where a device or user session is involved since it guarantees monotonic reads, monotonic writes, and read your own writes (RYW) guarantees.
* Session consistency provides predictable consistency for a session, and maximum read throughput while offering the lowest latency writes and reads.
* Azure Cosmos DB accounts that are configured with session consistency can associate any number of Azure regions with their Azure Cosmos DB account.
* The cost of a read operation (in terms of RUs consumed) with session consistency level is less than strong and bounded staleness, but more than eventual consistency

**Consistent Prefix**:

* Consistent prefix guarantees that in absence of any further writes, the replicas within the group eventually converge.
* Consistent prefix guarantees that reads never see out of order writes. If writes were performed in the order A, B, C, then a client sees either A, A,B, or A,B,C, but never out of order like A,C or B,A,C.
* Azure Cosmos DB accounts that are configured with consistent prefix consistency can associate any number of Azure regions with their Azure Cosmos DB account.

**Eventual**:

* Eventual consistency guarantees that in absence of any further writes, the replicas within the group eventually converge.
* Eventual consistency is the weakest form of consistency where a client may get the values that are older than the ones it had seen before.
* Eventual consistency provides the weakest read consistency but offers the lowest latency for both reads and writes.
* Azure Cosmos DB accounts that are configured with eventual consistency can associate any number of Azure regions with their Azure Cosmos DB account.
* The cost of a read operation (in terms of RUs consumed) with the eventual consistency level is the lowest of all the Azure Cosmos DB consistency levels

<https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels>