# DocumentDB 中的请求单位

本文概述了 [Microsoft Azure DocumentDB](/services/documentdb/) 中的请求单位。

阅读本文之后，你将能够回答以下问题：

* 什么是请求单位和请求费用？
* 如何指定集合的请求单位容量？
* 如何评估应用程序的请求单位需求？
* 如果超过集合的请求单位容量会发生什么情况？

## 请求单位和请求费用

DocumentDB 通过保留资源提供了快速且可预测的性能，以满足应用程序的吞吐量需求。由于应用程序加载和访问模式会随着时间推移而更改，DocumentDB 使你可以轻松增加或减少保留供你的应用程序使用的吞吐量。

通过 DocumentDB，可根据每秒处理的请求单位指定保留的吞吐量。你可以将请求单位视为吞吐量货币，因此，可以保留每秒可用于你的应用程序的定量有保障请求单位。DocumentDB 中的每个操作（编写文档、执行查询、更新文档）都会消耗 CPU、内存和 IOPS。也就是说，每个操作都会产生请求费用（用请求单位表示）。你要了解影响请求单位费用的因素，以及你的应用程序吞吐量要求，才能尽可能有效地运行你的应用程序。

## 指定请求单位容量

创建 DocumentDB 集合时，可以指定希望为集合保留的每秒请求单位的数量 (RU)。创建集合之后，将保留指定的 RU 的完整分配供集合使用。保证每个集合具有专用的和隔离的吞吐量特征。

值得注意的是，DocumentDB 在保留模型上操作，也就是说，按照你所保留的吞吐量向你计费，无论你主动使用了多少该吞吐量均是如此。但是请记住，随着应用程序的负载、数据和使用情况模式的更改，可以通过 DocumentDB SDK 或使用 [Azure 门户](https://portal.azure.cn)轻松增加或减少保留的 RU。有关扩大或缩小吞吐量的详细信息，请参阅 [DocumentDB performance levels（DocumentDB 性能级别）](/documentation/articles/documentdb-performance-levels)。

## 请求单位注意事项

在估计为 DocumentDB 集合保留的请求单位数量时，务必要考虑以下变量：

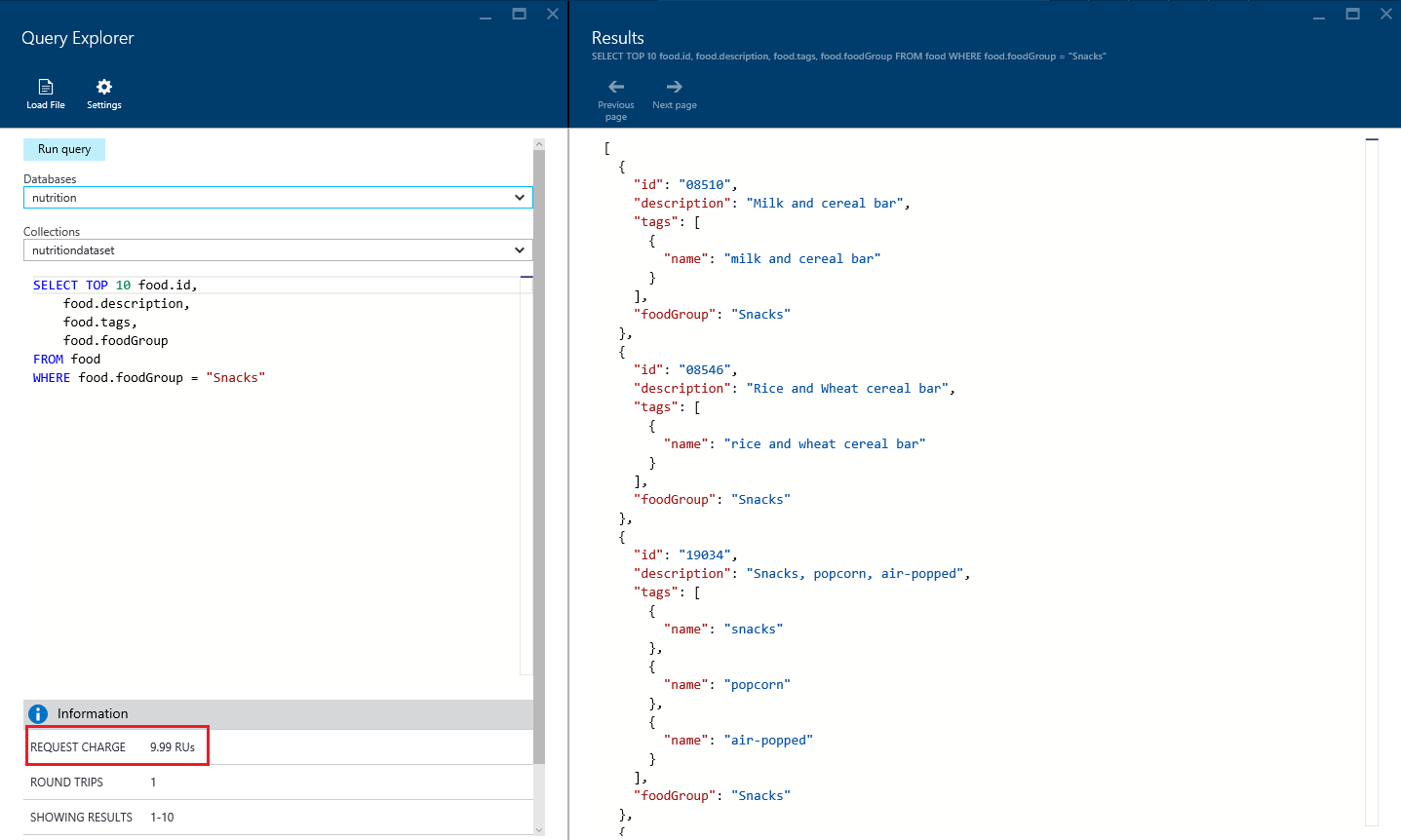
* **文档大小**。随着文档大小的增加，用来读取或写入数据的单位数也随之增加。
* **文档属性计数**。假设默认为所有属性创建索引，用于写入文档的单位数将随着属性计数的增加而增加。
* **数据一致性**。当使用“强”或“有限过时”的数据一致性级别时，将占用更多单位数来读取文档。
* **已编制索引的属性**。每个集合的索引策略都可确定默认情况下要进行索引的属性类别。通过限制已创建索引的属性数目或通过启用延迟索引编制，可以减少请求单位消耗。
* **文档索引编制**。默认情况下，将自动为每个文档创建索引，如果你选择不为其中一些文档创建索引，则将占用更少的请求单位。
* **查询模式**。查询的复杂性会影响操作使用的请求单位数量。谓词数、谓词性质、投影、UDF 数和源数据集的大小都会影响查询操作的成本。
* **脚本使用情况**。正如查询一样，存储过程和触发器也是根据所执行的操作的复杂性来使用请求单位的。在开发应用程序时，检查请求费用标头，以更好地了解每个操作消耗请求单位容量的方式。

## 估计吞吐量需求

请求单位是请求处理成本的规范化的度量。单个请求单位用于表示读取（通过自链接或 ID）一个包含 10 个唯一属性值（系统属性除外）的 1 KB JSON 文档所需的处理容量。插入、替换或删除同一文档的请求要占用服务的更多处理，因此需要更多请求单位。

[AZURE.NOTE] 用于 1 KB 文档的 1 个请求单位基线通过自链接或文档的ID 与简单的 GET 对应。

每个来自 DocumentDB 服务的响应都包含一个包含用于请求的请求单位的自定义标头 (x-ms-request-charge)。此标头也可通过 DocumentDB SDK 访问。在 .NET SDK 中，RequestCharge 是 ResourceResponse 对象的属性。对于查询，在 Azure 门户中的 DocumentDB 查询资源管理器提供了用于执行的查询的请求费用信息。



检查查询资源管理器中的 RU 费用

基于这一点，有一种方法可以估计应用程序所需的保留的吞吐量：记录与针对应用程序所使用的代表性文档运行典型操作相关联的请求单位费用，然后估计你预计每秒执行的操作数。也要确保测算并包含典型查询和 DocumentDB 脚本使用情况。

[AZURE.NOTE]如果某些文档类型在大小和已编制索引的属性方面有很大不同，那么就要记录与每个典型文档类型相关联的适用操作的请求单位费用。

例如：

1. 记录创建（插入）典型文档的请求单位费用。
2. 记录读取典型文档的请求单位费用。
3. 记录更新典型文档的请求单位费用。
4. 记录典型、常见文档查询的请求单位费用。
5. 记录应用程序使用的任何自定义脚本（存储的过程、触发器、用户自定义函数）的请求单位费用。
6. 根据给定的预计每秒运行的操作估计数计算所需的请求单位。

## 请求单位估计示例

请考虑以下 ~1 KB 文档：

{  
 "id": "08259",  
"description": "Cereals ready-to-eat, KELLOGG, KELLOGG'S CRISPIX",  
"tags": [  
 {  
 "name": "cereals ready-to-eat"  
 },  
 {  
 "name": "kellogg"  
 },  
 {  
 "name": "kellogg's crispix"  
 }  
],  
"version": 1,  
"commonName": "Includes USDA Commodity B855",  
"manufacturerName": "Kellogg, Co.",  
"isFromSurvey": false,  
"foodGroup": "Breakfast Cereals",  
"nutrients": [  
 {  
 "id": "262",  
 "description": "Caffeine",  
 "nutritionValue": 0,  
 "units": "mg"  
 },  
 {  
 "id": "307",  
 "description": "Sodium, Na",  
 "nutritionValue": 611,  
 "units": "mg"  
 },  
 {  
 "id": "309",  
 "description": "Zinc, Zn",  
 "nutritionValue": 5.2,  
 "units": "mg"  
 }  
],  
"servings": [  
 {  
 "amount": 1,  
 "description": "cup (1 NLEA serving)",  
 "weightInGrams": 29  
 }  
]  
}

[AZURE.NOTE]文档在 DocumentDB 中已缩小，所以系统计算的上述文档的大小略小于 1 KB。

下表显示了此文档的典型操作的请求单位大概费用（假定帐户一致性级别设置为“会话”且所有文档都自动编制索引）：

|  |  |
| --- | --- |
| 操作 | 请求单位费用 |
| 创建文档 | ~15 RU |
| 读取文档 | ~1 RU |
| 按 ID 查询文档 | ~2.5 RU |

此外，此表还显示了应用程序所用的请求单位大概费用：

|  |  |  |
| --- | --- | --- |
| 查询 | 请求单位费用 | 返回的文档数量 |
| 按 ID 选择食品 | ~2.5 RU | 1 |
| 按制造商选择食物 | ~7 RU | 7 |
| 按食品组选择并按重量排序 | ~70 RU | 100 |
| 选择食品组中的前 10 个食品 | ~10 RU | 10 |

[AZURE.NOTE]RU 费用取决于返回的文档数。

使用此信息，我们可以根据预计的每秒操作和查询数量来估计此应用程序的 RU 需求：

|  |  |  |
| --- | --- | --- |
| 操作/查询 | 每秒的估计数量 | 所需的 RU |
| 创建文档 | 10 | 150 |
| 读取文档 | 100 | 100 |
| 按制造商选择食物 | 25 | 175 |
| 按食品组进行选择 | 10 | 700 |
| 选择前 10 个 | 15 | 总计 150 |

在此示例中，我们认为平均吞吐量需求为 1,275 RU/s。舍入到最接近的百位数，我们会将此应用程序的集合设置为 1,300 RU/s。

## 超过保留的吞吐量限制

前面提到，请求单位消耗以每秒速率进行评估。对于超过集合设置的请求单位速率的应用程序，将限制对该集合的请求数，直到速率降低到保留级别之下。受到限制时，服务器将会抢先结束 RequestRateTooLarge（HTTP 状态代码 429）的请求并返回 x-ms-retry-after-ms 标头，该标头指示重试请求前用户必须等待的时间数量（以毫秒为单位）。

HTTP Status 429  
Status Line: RequestRateTooLarge  
x-ms-retry-after-ms :100

如果使用的是 .NET 客户端 SDK 和 LINQ 查询，则大多数情况下都不需要处理此异常，因为 .NET 客户端 SDK 的当前版本会隐式捕获此响应，并会遵循服务器指定的 retry-after 标头，然后重试请求。除非多个客户端同时访问你的帐户，否则下次重试就会成功。

如果存在多个高于请求速率的请求操作，则默认重试行为可能无法满足需要，这时客户端就会向应用程序引发 DocumentClientException，其状态代码为 429。在这种情况下，你可以考虑处理重试行为和应用程序错误处理例程中的逻辑，或为集合增加保留的吞吐量。

## 后续步骤

若要了解有关 Azure DocumentDB 的保留吞吐量的详细信息，请浏览以下资源：

* [DocumentDB 定价](/pricing/details/documentdb/)
* [管理 DocumentDB 容量](/documentation/articles/documentdb-manage)
* [对 DocumentDB 中的数据进行建模](/documentation/articles/documentdb-modeling-data)
* [DocumentDB 性能级别](/documentation/articles/documentdb-partition-data)

若要了解有关 DocumentDB 的详细信息，请参阅 Azure DocumentDB [文档](/documentation/services/documentdb/)。