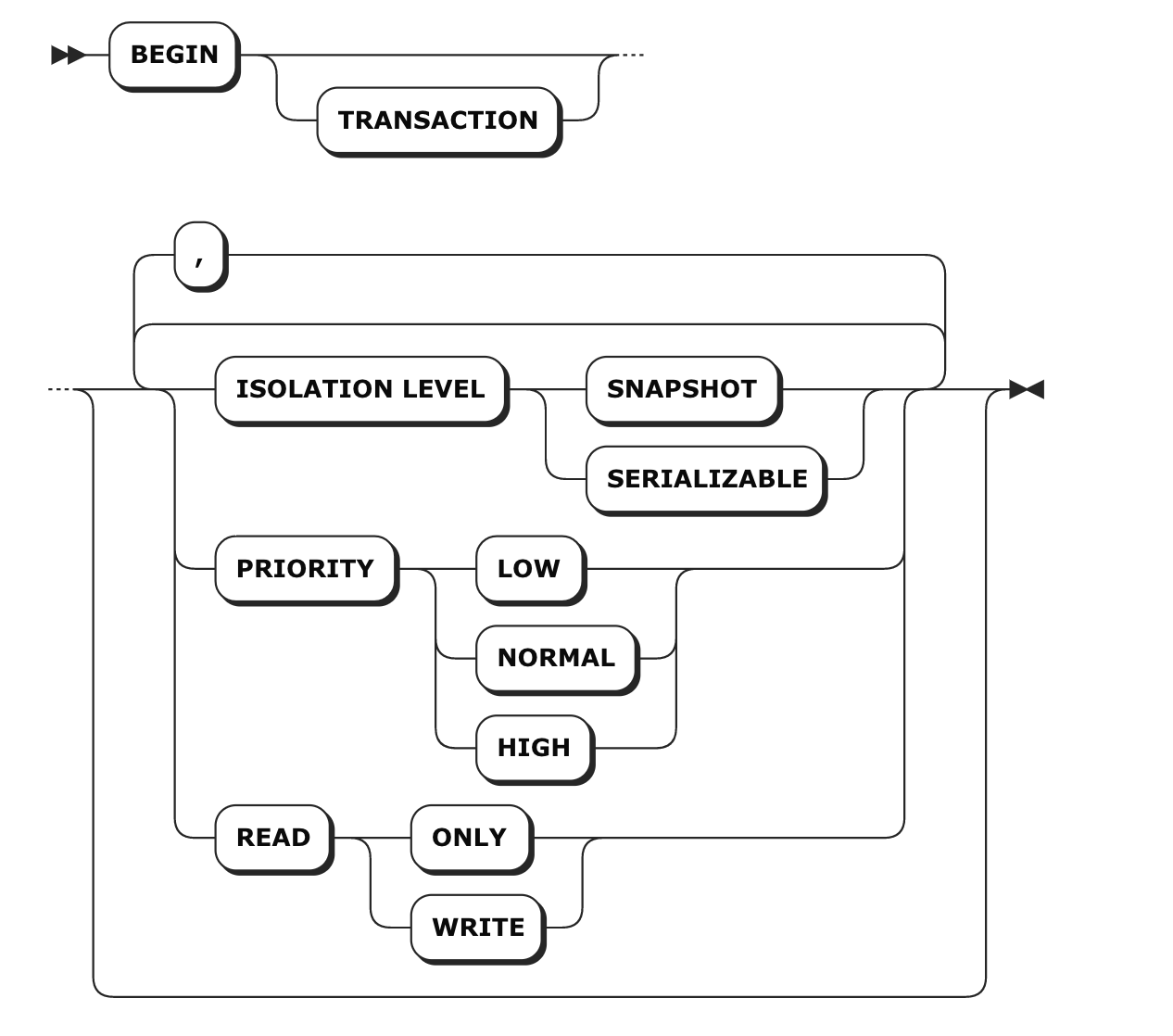
BEGIN语句启动一个事务，要么成功执行它包含的所有语句，要么根本不执行任何语句。

注意：使用事务时，应用程序应包含重试中止事务的逻辑，以中断并发事务之间的依赖关系循环。

概要



所需权限

启动事务不需要特殊权限，不过，事务中的语句都有其特定需要的权限。

别名

在CockroachDB中，以下是BEGIN语句的别名： - BEGIN TRANSACTION - START TRANSACTION

隔离级别中也存在以下别名：

* READ UNCOMMITTED, READ COMMITTED, 和 REPEATABLE READ 是 SERIALIZABLE的别名。

有关隔离级别别名的详细信息, 查看 [Comparison to ANSI SQL Isolation Levels](http://doc.cockroachchina.baidu.com/#develop/transactions/#comparison-to-ansi-sql-isolation-levels).

参数

| **Parameter** | **Description** |
| --- | --- |
| ISOLATION LEVEL | 默认情况下, CockroachDB的事务实现了最严格的ANSI隔离级别: SERIALIZABLE. 在这个隔离级别, 事务将不会导致异常. SNAPSHOT 隔离级别依然支持以保持向后兼容, 不过你应当避免使用它. 它在性能方面并无优势，并且在某些稍复杂的情况下可能会导致状态不一致。更多信息可以查看 [事务：隔离级别](http://doc.cockroachchina.baidu.com/#develop/transactions/#isolation-levels).  **默认**: SERIALIZABLE |
| PRIORITY | 如果你不想以优先级 NORMAL 执行事务, 你可以设置为LOW 或者 HIGH.  优先级越高的事务，重试的几率越低  更多细节查看[Transactions: Priorities](http://doc.cockroachchina.baidu.com/" \l "develop/transactions/#transaction-priorities).  **默认**: NORMAL |

举例

开始一个事务

使用默认设置

如果不修改BEGIN, 事务将以SERIALIZABLE 隔离级别和NORMAL 优先级。

> **BEGIN**;

> **SAVEPOINT** cockroach\_restart;

> **UPDATE** products **SET** inventory = 0 **WHERE** sku = '8675309';

> **INSERT** **INTO** orders (customer, sku, **status**) **VALUES** (1001, '8675309', 'new');

> **RELEASE** **SAVEPOINT** cockroach\_restart;

> **COMMIT**;

修改隔离级别和优先级

你可以将事务的隔离级别设置为SNAPSHOT，也可以把优先级设置为LOW 或者 HIGH。

> **BEGIN** **ISOLATION** **LEVEL** **SNAPSHOT**, PRIORITY HIGH;

> **SAVEPOINT** cockroach\_restart;

> **UPDATE** products **SET** inventory = 0 **WHERE** sku = '8675309';

> **INSERT** **INTO** orders (customer, sku, **status**) **VALUES** (1001, '8675309', 'new');

> **RELEASE** **SAVEPOINT** cockroach\_restart;

> **COMMIT**;

你也可以使用[SET TRANSACTION](https://www.cockroachlabs.com/docs/stable/set-transaction.html)来修改事务的隔离级别和优先级。

使用自动重试开始事务

CockroachDB将自动重试同一批中包含BEGIN和COMMIT的所有事务。批处理由您的驱动程序或客户端的行为控制, 但意味着CockroachDB将所有语句作为一个单元接收，而不是一些请求。

从CockroachDB的角度来看，事务作为批处理发送，如下所示：

> **BEGIN**; **DELETE** **FROM** customers **WHERE** id = 1; **DELETE** orders **WHERE** customer = 1; **COMMIT**;

但是，在应用程序的代码中，批处理事务通常只是一次发送多个语句。举个例子，在Go语言中，此事务将作为单个批次发送（并自动重试）：

db.Exec(

"BEGIN;

DELETE FROM customers WHERE id = 1;

DELETE orders WHERE customer = 1;

COMMIT;"

)

以这种方式发出语句向CockroachDB发出信号，如果事务没有一次执行成功，则不需要更改任何语句的值，因此它可以不断重试事务直到它成功执行为止。

See Also

* [Transactions](http://doc.cockroachchina.baidu.com/#develop/transactions/)
* [COMMIT](https://www.cockroachlabs.com/docs/stable/commit-transaction.html)
* [SAVEPOINT](https://www.cockroachlabs.com/docs/stable/savepoint.html)
* [RELEASE SAVEPOINT](https://www.cockroachlabs.com/docs/stable/release-savepoint.html)
* [ROLLBACK](https://www.cockroachlabs.com/docs/stable/rollback-transaction.html)