SHOW TRACE [语句](http://doc.cockroachchina.baidu.com/#develop/sql-statements/overview/) 用于获取CockroachDB如何执行一条语句或一系列语句的详细信息。这些信息包含执行中涉及的所有节点消息与时序信息，通过这些信息可以查看到CockroachDB在软件层中执行的操作。

SHOW TRACE 语句可以用于调试查询为何没有按预期执行，增加更多的bug报告信息，或者让用户了解更多关于CockraochDB的工作原理。

使用概述

SHOW TRACE有两种不同的使用方式:

| **Statement** | **Usage** |
| --- | --- |
| SHOW TRACE FOR <stmt> | 执行一个单独的 [explainable](https://www.cockroachlabs.com/docs/stable/sql-grammar.html#explainable_stmt)语句，返回执行的跟踪信息。 |
| SHOW TRACE FOR SESSION | 返回会话期间所有执行语句的跟踪信息。 |

SHOW TRACE FOR <stmt>

使用SHOW TRACE执行一个 [explainable](http://doc.cockroachchina.baidu.com/#develop/sql-statements/EXPLAIN/#explainable-statements)语句时，会返回执行过程中涉及的所有节点的消息与时序信息，需要注意以下几点：

* SHOW TRACE FOR <stmt> 会执行目标语句，并且一旦执行完成后，会返回执行相关的操作信息。例如，跟踪 INSERT 语句会插入数据，然后返回跟踪信息；跟踪 DELETE 会删除数据，然后返回跟踪信息。这和[EXPLAIN](http://doc.cockroachchina.baidu.com/#develop/sql-statements/EXPLAIN/) 语句是不一样的，[EXPLAIN](http://doc.cockroachchina.baidu.com/#develop/sql-statements/EXPLAIN/) 语句不会执行后面的语句，只会返回一个预期的执行计划。
  + SHOW TRACE FOR <stmt>中的语句必须是一个[explainable](http://doc.cockroachchina.baidu.com/#develop/sql-statements/EXPLAIN/#explainable-statements) 语句，所有非可解释的语句都不支持。
  + 目标语句是使用本地的SQL引擎来执行的。由于已知的 [known limitation](https://github.com/cockroachdb/cockroach/issues/16562)，当不是SHOW TRACE FOR <stmt>的目标时，跟踪记录将不会反映这些语句的执行方式。
* 如果目标语句执行遇到错误时，这些错误不会返回给客户端。这些错误信息会记录到跟踪信息中。这对事务重试[transaction retries](http://doc.cockroachchina.baidu.com/#develop/transactions/#transaction-retries)有以下重要影响：
  + 正常来说，当由于争用而遇到[retryable errors](http://doc.cockroachchina.baidu.com/#develop/transactions/#error-handling)时，CockroachDB会将客户端的单个语句(隐式事务)和批量的多语句事务进行[automatically retried](http://doc.cockroachchina.baidu.com/#develop/transactions/#automatic-retries)。但是，当这些语句是 SHOW TRACE FOR <stmt>的目标语句时，CockroachDB **不会** 自动重试。
  + 当多语句事务中的每个语句单独发送时(与批量的多语句事务相反)，如果某一条语句是 SHOW TRACE <stmt>的目标语句，这条语句的重试错误不会返回到客户端。

Tips: 鉴于这些影响，当我们期望有事务重试而且可以跟踪重试信息时，使用SHOW TRACE FOR SESSION。

* 在跟踪单个语句(及隐式事务)，跟踪可能会改变语句提交其数据的方式；跟踪会禁止涉及单range事务的“单阶段”优化。所以，跟踪将不会真实反映事务的提交。而 SHOW TRACE FOR SESSION就不会有这个影响。

SHOW TRACE FOR SESSION

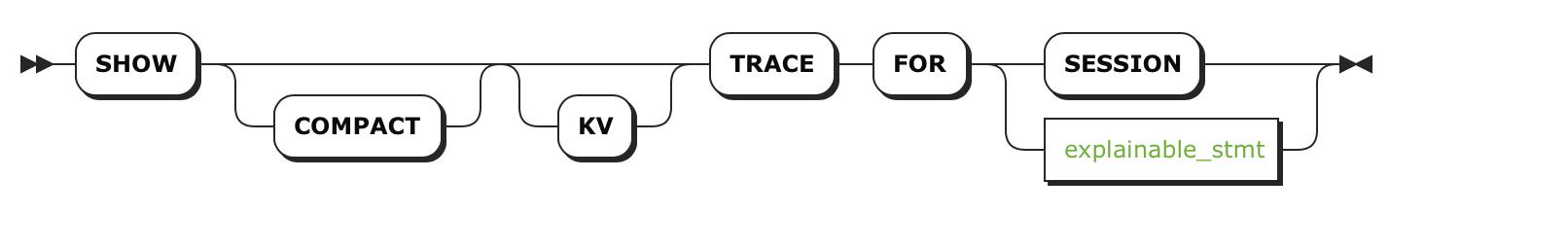
该语句会返回一个会话期间内的所有语句的消息与时序信息。需要注意以下几点：

* SHOW TRACE FOR SESSION只会返回最近的一些跟踪记录，或者返回当前活跃语句的记录。
* 如果要开启一个会话的跟踪记录，可以通过语句 [SET tracing = on;](http://doc.cockroachchina.baidu.com/#develop/sql-statements/SET(session-variable)/#set-tracing)设定。
* 如果要关闭一个会话的跟踪记录，可以通过语句[SET tracing = off;](http://doc.cockroachchina.baidu.com/#develop/sql-statements/SET(session-variable)/#set-tracing)设定。
* 与 SHOW TRACE FOR <stmt>不同，跟踪记录不会影响任何跟踪语句的执行。也就是说，在跟踪期间，语句执行错误会返回错误给客户端。当由于争用而遇到[retryable errors](http://doc.cockroachchina.baidu.com/#develop/transactions/#error-handling)时，CockroachDB会对单个语句(即隐式事务)与一条条发送的多语句事务进行自动重试 [automatically retry](http://doc.cockroachchina.baidu.com/#develop/transactions/#automatic-retries)。而且，客户端会收到要求进行[client-side retries](http://doc.cockroachchina.baidu.com/#develop/transactions/#client-side-intervention)的重试错误。也就是说，在跟踪期间，所有的事务重试记录将会记录下来。
* SHOW TRACE FOR <stmt>会覆盖最后的跟踪记录。这意味着，如果我们开启会话记录，关闭会话记录，执行SHOW TRACE FOR <stmt> ，然后执行SHOW TRACE FOR SESSION，跟踪记录返回的是SHOW TRACE FOR <stmt>记录，而不是之前的会话记录。

所需权限

对于SHOW TRACE FOR <stmt> 语句， 用户需要拥有跟踪语句相应的 [权限](http://doc.cockroachchina.baidu.com/#deploy/access-management/privileges/) 。对于SHOW TRACE FOR SESSION 语句，无权限要求。

概要



参数

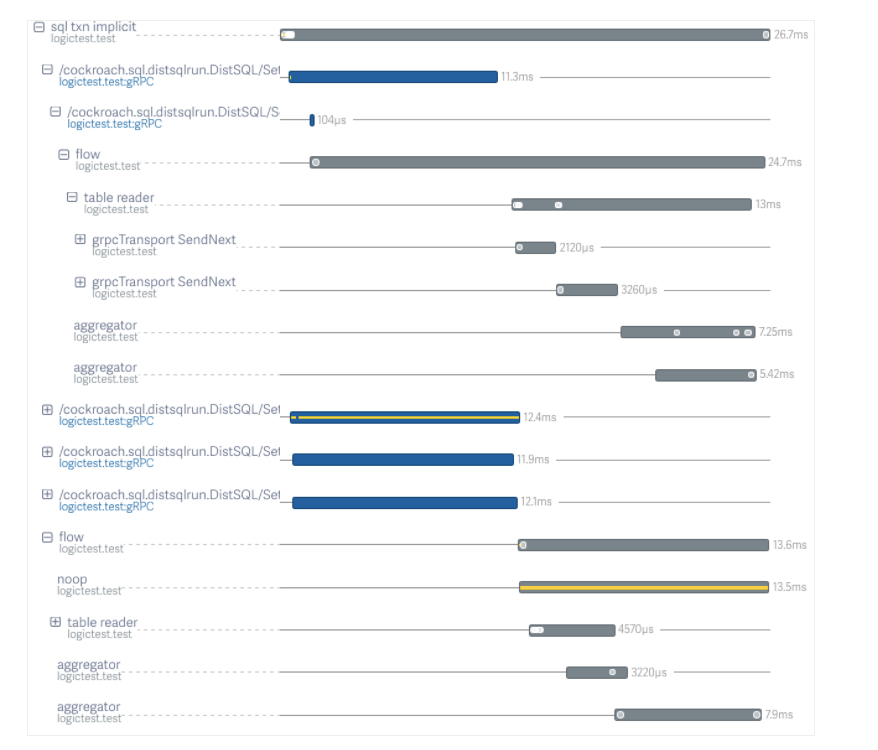
| **Parameter** | **Description** |
| --- | --- |
| KV | 如果指定该参数，返回的信息将会被限定为key-value [storage layer](https://www.cockroachlabs.com/docs/stable/architecture/storage-layer.html)中的请求和响应消息，包括per-result-row 消息。  对于 SHOW KV TRACE FOR <stmt>,包含 per-result-row 消息。  对于SHOW KV TRACE FOR SESSION, 只有在设置SET tracing = kv;后，才包含per-result-row 消息。 |
| COMPACT | v2.0版本新特点: 如果使用了该参数，会返回很少的列。更多相关信息，请看[Response](http://doc.cockroachchina.baidu.com/#develop/sql-statements/SHOW-TRACE/#response)。 |
| explainable\_stmt | 要执行和跟踪的语句。仅支持 [explainable](http://doc.cockroachchina.baidu.com/#develop/sql-statements/EXPLAIN/#explainable-statements) 语句 |

跟踪描述

CockroachDB的“跟踪”定义是 [OpenTracing's](http://opentracing.io/documentation/#what-is-a-trace) 定义的一个具体化。在内部，CockroachDB使用OpenTracing库进行跟踪，也就是说，它可以很容易的与OpenTracing兼容的收集器进行合成，例如，现在已经支持Lightstep与Zipkin。

| **Concept** | **Description** |
| --- | --- |
| **trace** | 作为高级操作(一个查询或者事务)一部分的子操作执行的相关消息。消息在内部表示为一个"spans"树，而在SHOW TRACE FOR <stmt> 中的查询执行或者SHOW TRACE FOR SESSION中的这个SQL事务执行就是一个具体的"root span"。 |
| **span** | 一种命名的定时操作，用于描述跟踪中的连续工作段。每个span下有子span，代表子操作；而这些子操作也会有子操作。  不同的span表示顺序或者并行执行的(子)操作。(可并行执行是跟踪应该描述的重要事情之一)跟踪的操作描述可能是分布式的，也就是说，不同的span可以描述由不同节点执行的操作。 |
| **message** | 带有计时信息的字符串。每个span可能包含一系列message。他们在日志记录的基础上生成的，而且可以在CockraochDB的日志文件中找到相同的消息，但是对于跟踪记录中的跨所有严重级别的消息在日志文件中是不存在的，因为日志文件默认不存储这类消息。所以，跟踪记录比日志文件更加冗长，但是跟踪记录仅仅包含一个特定操作的上下文生成的消息。 |

为了进一步阐明这些定义，我们可以看一下一条语句跟踪的可视化图。通过 [Lightstep](http://lightstep.com/)(在CockraochDB上聚合Lightstep的文档会在稍后推出)将具体的跟踪可视化。图中仅仅显示了spans，但是在工具中可以查看具体的消息。可以查看操作与子操作的名字，以及操作的父子关系与时序信息，而且可以很容易查看并行执行的操作。



返回

Note： SHOW TRACE的返回格式在未来版本中可能会改变。

CockroachDB以线性表格格式输出记录。每个结果行代表一个span开始(由=== SPAN START: <operation> ===消息标识)或者span的日志信息。行数据通常是按照时间戳的顺序列出(即他们所代表的事件的发生顺序)，但是子span的消息根据其时间在父span中交错排列。但是，来自兄弟span的消息是不会相互交错。

下图显示了不同spans的消息在跟踪实例中的交错顺序。每个盒子是一个span，盒子内部的盒子是子span。数字代表了日志信息在虚拟表中的显示顺序。

+-----------------------+

| 1 |

| +-------------------+ |

| | 2 | |

| | +----+ | |

| | | | +----+ | |

| | | 3 | | 4 | | |

| | | | | | 5 | |

| | | | | | ++ | |

| | +----+ | | | |

| | +----+ | |

| | 6 | |

| +-------------------+ |

| 7 |

+-----------------------+

Each row contains the following columns:

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| timestamp | timestamptz | 消息发生时的绝对时间。 |
| age | interval | 消息距离跟踪开始(即，SHOW TRACE FOR <stmt> 或者SHOW TRACE FOR SESSION开始执行的时间)的时间。 |
| message | string | 日志信息 |
| tag | string | 有关消息上下文的元信息。这与在方括号之间的日志文件消息的开头出现的信息相同(如，[client=[::1]:49985,user=root,n1])。 |
| loc | string | New in v2.0: 文件：生成消息的代码行位置。仅仅一些消息有这个字段，这取决于消息是如何记录下来的。传递给生成消息的 --vmodule 标志也会影响这个字段的填充。通常，如果指定了--vmodule=<file>=<level>，那么这个文件生成的消息将填充这个字段。 |
| operation | string | 记录消息操作(或者子操作)的名字。 |
| span | int | 如果按照span开始时间进行排序，一个span在所有span的虚拟列表中的span索引。 |

Note: 如果指定了 COMPACT 关键字，仅仅 age, message, tag 和 operation列会返回。此外，loc 列的值会预先添加到message中。

例子

跟踪一个简单的 SELECT语句

> **SHOW** TRACE **FOR** **SELECT** \* **FROM** foo;

+----------------------------------+---------------+-------------------------------------------------------+------------------------------------------------+-----+-----------------------------------+------+

| timestamp | age | message | tag | loc | operation | span |

+----------------------------------+---------------+-------------------------------------------------------+------------------------------------------------+-----+-----------------------------------+------+

| 2018-03-08 21:22:18.266373+00:00 | 0s | === SPAN START: sql txn === | | | sql txn | 0 |

| 2018-03-08 21:22:18.267341+00:00 | 967µs713ns | === SPAN START: session recording === | | | session recording | 5 |

| 2018-03-08 21:22:18.267343+00:00 | 969µs760ns | === SPAN START: starting plan === | | | starting plan | 1 |

| 2018-03-08 21:22:18.267367+00:00 | 993µs551ns | === SPAN START: consuming rows === | | | consuming rows | 2 |

| 2018-03-08 21:22:18.267384+00:00 | 1ms10µs504ns | Scan /Table/51/{1-2} | [n1,client=[::1]:58264,user=root] | | sql txn | 0 |

| 2018-03-08 21:22:18.267434+00:00 | 1ms60µs392ns | === SPAN START: dist sender === | | | dist sender | 3 |

| 2018-03-08 21:22:18.267444+00:00 | 1ms71µs136ns | querying next range at /Table/51/1 | [client=[::1]:58264,user=root,txn=76d25cda,n1] | | dist sender | 3 |

| 2018-03-08 21:22:18.267462+00:00 | 1ms88µs421ns | r20: sending batch 1 Scan to (n1,s1):1 | [client=[::1]:58264,user=root,txn=76d25cda,n1] | | dist sender | 3 |

| 2018-03-08 21:22:18.267465+00:00 | 1ms91µs570ns | sending request to local server | [client=[::1]:58264,user=root,txn=76d25cda,n1] | | dist sender | 3 |

| 2018-03-08 21:22:18.267467+00:00 | 1ms93µs707ns | === SPAN START: /cockroach.roachpb.Internal/Batch === | | | /cockroach.roachpb.Internal/Batch | 4 |

| 2018-03-08 21:22:18.267469+00:00 | 1ms96µs103ns | 1 Scan | [n1] | | /cockroach.roachpb.Internal/Batch | 4 |

| 2018-03-08 21:22:18.267471+00:00 | 1ms97µs437ns | read has no clock uncertainty | [n1] | | /cockroach.roachpb.Internal/Batch | 4 |

| 2018-03-08 21:22:18.267474+00:00 | 1ms101µs60ns | executing 1 requests | [n1,s1] | | /cockroach.roachpb.Internal/Batch | 4 |

| 2018-03-08 21:22:18.267479+00:00 | 1ms105µs912ns | read-only path | [n1,s1,r20/1:/Table/5{1-2}] | | /cockroach.roachpb.Internal/Batch | 4 |

| 2018-03-08 21:22:18.267483+00:00 | 1ms110µs94ns | command queue | [n1,s1,r20/1:/Table/5{1-2}] | | /cockroach.roachpb.Internal/Batch | 4 |

| 2018-03-08 21:22:18.267487+00:00 | 1ms114µs240ns | waiting for read lock | [n1,s1,r20/1:/Table/5{1-2}] | | /cockroach.roachpb.Internal/Batch | 4 |

| 2018-03-08 21:22:18.26752+00:00 | 1ms146µs596ns | read completed | [n1,s1,r20/1:/Table/5{1-2}] | | /cockroach.roachpb.Internal/Batch | 4 |

| 2018-03-08 21:22:18.267566+00:00 | 1ms192µs724ns | plan completed execution | [n1,client=[::1]:58264,user=root] | | consuming rows | 2 |

| 2018-03-08 21:22:18.267568+00:00 | 1ms195µs60ns | resources released, stopping trace | [n1,client=[::1]:58264,user=root] | | consuming rows | 2 |

+----------------------------------+---------------+-------------------------------------------------------+------------------------------------------------+-----+-----------------------------------+------+

(19 rows)

Tips: 可以使用 SHOW TRACE 的结果作为SELECT的[data source](http://doc.cockroachchina.baidu.com/#develop/sql-statements/SHOW-TRACE/table-expressions.html) ，然后使用WHERE子句进行数据过滤。 例如，如果仅仅想看message列有 spans字符串开头的行, 可以执行如下语句 SELECT \* FROM [SHOW TRACE FOR ] where message LIKE '=== SPAN START%'。

跟踪冲突的事务

在这个例子中，使用两个客户端生成冲突事务。

1. 在终端1，创建一张表：

> **CREATE** **TABLE** t (k INT);

1. 还在终端 1，开启一个事务，执行一个写操作，但是不关闭事务：

> **BEGIN**;

> **INSERT** **INTO** t **VALUES** (1);

1. 在终端 2中，执行并跟踪冲突的读取

> **SELECT** age, span, message **FROM** [**SHOW** TRACE **FOR** **SELECT** \* **FROM** t];

当前语句会被阻塞，直到终端1的事务完成。

1. 回到终端1，完成事务：

> **COMMIT**;

1. 回到终端2，会看到如下跟踪信息：

Tips: 查看以#Annotation开头的行，了解冲突是如何被跟踪的。

+*-------------------+--------+-------------------------------------------------------------------------------------------------------+*

| age | span | message |

+*-------------------+--------+-------------------------------------------------------------------------------------------------------+*

| 0s | (0,0) | === SPAN **START**: **sql** txn implicit === |

| 409µs750ns | (0,1) | === SPAN **START**: **starting** plan === |

| 417µs68ns | (0,2) | === SPAN **START**: consuming **rows** === |

| 446µs968ns | (0,0) | querying **next** range **at** /**Table**/61/1 |

| 474µs387ns | (0,0) | r42: sending batch 1 Scan **to** (n1,s1):1 |

| 491µs800ns | (0,0) | sending request **to** **local** **server** |

| 503µs260ns | (0,3) | === SPAN **START**: /cockroach.roachpb.Internal/Batch === |

| 506µs135ns | (0,3) | 1 Scan |

| 508µs385ns | (0,3) | **read** has **no** clock uncertainty |

| 512µs176ns | (0,3) | executing 1 requests |

| 518µs675ns | (0,3) | **read**-**only** path |

| 525µs357ns | (0,3) | command queue |

| 531µs990ns | (0,3) | waiting **for** **read** **lock** |

| # Annotation: The following line identifies the conflict, **and** **some** **of** the **lines** below it **describe** the conflict resolution |

| 603µs363ns | (0,3) | conflicting intents **on** /**Table**/61/1/285895906846146561/0 |

| 611µs228ns | (0,3) | replica.Send got error: conflicting intents **on** /**Table**/61/1/285895906846146561/0 |

| # Annotation: The **read** **is** **now** going **to** wait **for** the writer **to** finish **by** executing a PushTxn request. |

| 615µs680ns | (0,3) | pushing 1 **transaction**(s) |

| 630µs734ns | (0,3) | querying **next** range **at** /**Table**/61/1/285895906846146561/0 |

| 646µs292ns | (0,3) | r42: sending batch 1 PushTxn **to** (n1,s1):1 |

| 658µs613ns | (0,3) | sending request **to** **local** **server** |

| 665µs738ns | (0,4) | === SPAN **START**: /cockroach.roachpb.Internal/Batch === |

| 668µs765ns | (0,4) | 1 PushTxn |

| 671µs770ns | (0,4) | executing 1 requests |

| 677µs182ns | (0,4) | **read**-**write** path |

| 681µs909ns | (0,4) | command queue |

| 693µs718ns | (0,4) | applied **timestamp** **cache** |

| 794µs20ns | (0,4) | evaluated request |

| 807µs125ns | (0,4) | replica.Send got error: failed **to** push "sql txn" id=23fce0c4 **key**=/**Table**/61/1/285895906846146561/0 ... |

| 812µs917ns | (0,4) | 62cddd0b pushing 23fce0c4 (1 pending) |

| 4s348ms604µs506ns | (0,4) | result **of** pending push: "sql txn" id=23fce0c4 **key**=/**Table**/61/1/285895906846146561/0 rw=true pri=0 ... |

| # Annotation: The writer **is** detected **to** have finished. |

| 4s348ms609µs635ns | (0,4) | push request **is** satisfied |

| 4s348ms657µs576ns | (0,3) | 23fce0c4-1d22-4321-9779-35f0f463b2d5 **is** **now** COMMITTED |

| # Annotation: The **write** has committed. **Some** cleanup follows. |

| 4s348ms659µs899ns | (0,3) | resolving intents [wait=true] |

| 4s348ms669µs431ns | (0,17) | === SPAN **START**: storage.intentResolve: resolving intents === |

| 4s348ms726µs582ns | (0,17) | querying **next** range **at** /**Table**/61/1/285895906846146561/0 |

| 4s348ms746µs398ns | (0,17) | r42: sending batch 1 ResolveIntent **to** (n1,s1):1 |

| 4s348ms758µs761ns | (0,17) | sending request **to** **local** **server** |

| 4s348ms769µs344ns | (0,18) | === SPAN **START**: /cockroach.roachpb.Internal/Batch === |

| 4s348ms772µs713ns | (0,18) | 1 ResolveIntent |

| 4s348ms776µs159ns | (0,18) | executing 1 requests |

| 4s348ms781µs364ns | (0,18) | **read**-**write** path |

| 4s348ms786µs536ns | (0,18) | command queue |

| 4s348ms797µs901ns | (0,18) | applied **timestamp** **cache** |

| 4s348ms868µs521ns | (0,18) | evaluated request |

| 4s348ms875µs924ns | (0,18) | acquired {raft,replica}mu |

| 4s349ms150µs556ns | (0,18) | applying command |

| 4s349ms232µs373ns | (0,3) | **read**-**only** path |

| 4s349ms237µs724ns | (0,3) | command queue |

| 4s349ms241µs857ns | (0,3) | waiting **for** **read** **lock** |

| # Annotation: This **is** **where** we would have been **if** there hadn't been a conflict. |

| 4s349ms280µs702ns | (0,3) | read completed |

| 4s349ms330µs707ns | (0,2) | output row: [1] |

| 4s349ms333µs718ns | (0,2) | output row: [1] |

| 4s349ms336µs53ns | (0,2) | output row: [1] |

| 4s349ms338µs212ns | (0,2) | output row: [1] |

| 4s349ms339µs111ns | (0,2) | plan completed execution |

| 4s349ms341µs476ns | (0,2) | resources released, stopping trace |

+-------------------+--------+-------------------------------------------------------------------------------------------------------+

跟踪事务重试

在这个例子中，我们使用会话跟踪来查看 [automatic transaction retry](http://doc.cockroachchina.baidu.com/#develop/transactions/#automatic-retries)。有上面的例子一样，我们需要使用两个终端，因为重试是因为事务之间不好的交互引起的。

1. 在终端 1中，取消smart\_prompt设置，开启跟踪记录，然后在开启一个事务：

> \unset smart\_prompt

> **SET** tracing = cluster;

> **BEGIN**;

开启事务会获取一个早期时间戳，即我们会锁定事务运行时的数据快照。

1. 在终端2,执行一个读命令：

> **SELECT** \* **FROM** t;

这个终端中的读执行时刻的时间戳是大于终端1执行开启事务时的时间戳。因为我们是在[SERIALIZABLE transaction isolation level](https://www.cockroachlabs.com/docs/stable/architecture/transaction-layer.html#isolation-levels) 的事务隔离级别上操作，如果系统允许终端1的事务提交，那么必须确保在终端2之前的终端1的事务是有效的；这个将在1秒内变的相关。

1. 回到终端 1，执行和跟踪冲突写：

> **INSERT** **INTO** t **VALUES** (1);

在这个时间点，系统会探测到冲突，而且事务已经不能在提交，因为事务提交意味着会该改变终端2读取的历史记录。因此，会自动重启事务，以便在终端2的事务之后对其进行序列化。跟踪记录会反映这个重试。

1. 在终端2,执行一个读命令：

> **SET** tracing = off;

> **SELECT** age, message **FROM** [**SHOW** TRACE **FOR** **SESSION**];

Tips: 查看以#Annotation开头的行，了解冲突是如何被跟踪的。

+*--------------------+---------------------------------------------------------------------------------------------------------------+*

| age | message |

+*--------------------+---------------------------------------------------------------------------------------------------------------+*

| 0s | === SPAN **START**: **sql** txn implicit === |

| 123µs317ns | AutoCommit. err: <nil>␤ |

| | txn: "sql txn implicit" id=64d34fbc **key**=/**Min** rw=false pri=0.02500536 iso=**SERIALIZABLE** stat=COMMITTED ... |

| 1s767ms959µs448ns | === SPAN **START**: **sql** txn === |

| 1s767ms989µs448ns | executing 1/1: **BEGIN** **TRANSACTION** |

| # Annotation: **First** execution **of** **INSERT**. |

| 13s536ms79µs67ns | executing 1/1: **INSERT** **INTO** t **VALUES** (1) |

| 13s536ms134µs682ns | client.Txn did AutoCommit. err: <nil>␤ |

| | txn: "unnamed" id=329e7307 **key**=/**Min** rw=false pri=0.01354772 iso=**SERIALIZABLE** stat=COMMITTED epo=0 ... |

| 13s536ms143µs145ns | added **table** 't' **to** **table** collection |

| 13s536ms305µs103ns | **query** **not** supported **for** distSQL: mutations **not** supported |

| 13s536ms365µs919ns | querying **next** range **at** /**Table**/61/1/285904591228600321/0 |

| 13s536ms400µs155ns | r42: sending batch 1 CPut, 1 BeginTxn **to** (n1,s1):1 |

| 13s536ms422µs268ns | sending request **to** **local** **server** |

| 13s536ms434µs962ns | === SPAN **START**: /cockroach.roachpb.Internal/Batch === |

| 13s536ms439µs916ns | 1 CPut, 1 BeginTxn |

| 13s536ms442µs413ns | **read** has **no** clock uncertainty |

| 13s536ms447µs42ns | executing 2 requests |

| 13s536ms454µs413ns | **read**-**write** path |

| 13s536ms462µs456ns | command queue |

| 13s536ms497µs475ns | applied **timestamp** **cache** |

| 13s536ms637µs637ns | evaluated request |

| 13s536ms646µs468ns | acquired {raft,replica}mu |

| 13s536ms947µs970ns | applying command |

| 13s537ms34µs667ns | coordinator spawns |

| 13s537ms41µs171ns | === SPAN **START**: [async] kv.TxnCoordSender: heartbeat loop === |

| # Annotation: The conflict **is** about **to** be detected **in** the form **of** a retriable error. |

| 13s537ms77µs356ns | automatically retrying **transaction**: **sql** txn (id: b4bd1f60-30d9-4465-bdb6-6b553aa42a96) because **of** error: |

| HandledRetryableTxnError: **serializable** **transaction** **timestamp** pushed (detected **by** **SQL** Executor) |

| # Annotation: **Second** execution **of** **INSERT**. |

| 13s537ms83µs369ns | executing 1/1: **INSERT** **INTO** t **VALUES** (1) |

| 13s537ms109µs516ns | client.Txn did AutoCommit. err: <nil>␤ |

| | txn: "unnamed" id=1228171b **key**=/**Min** rw=false pri=0.02917782 iso=**SERIALIZABLE** stat=COMMITTED epo=0 |

| ts=1507321556.991937203,0 orig=1507321556.991937203,0 **max**=1507321557.491937203,0 wto=false rop=false |

| 13s537ms111µs738ns | releasing 1 **tables** |

| 13s537ms116µs944ns | added **table** 't' **to** **table** collection |

| 13s537ms163µs155ns | **query** **not** supported **for** distSQL: writing txn |

| 13s537ms192µs584ns | querying **next** range **at** /**Table**/61/1/285904591231418369/0 |

| 13s537ms209µs601ns | r42: sending batch 1 CPut **to** (n1,s1):1 |

| 13s537ms224µs219ns | sending request **to** **local** **server** |

| 13s537ms233µs350ns | === SPAN **START**: /cockroach.roachpb.Internal/Batch === |

| 13s537ms236µs572ns | 1 CPut |

| 13s537ms238µs39ns | **read** has **no** clock uncertainty |

| 13s537ms241µs255ns | executing 1 requests |

| 13s537ms245µs473ns | **read**-**write** path |

| 13s537ms248µs915ns | command queue |

| 13s537ms261µs543ns | applied **timestamp** **cache** |

| 13s537ms309µs401ns | evaluated request |

| 13s537ms315µs302ns | acquired {raft,replica}mu |

| 13s537ms580µs149ns | applying command |

| 18s378ms239µs968ns | executing 1/1: **COMMIT** **TRANSACTION** |

| 18s378ms291µs929ns | querying **next** range **at** /**Table**/61/1/285904591228600321/0 |

| 18s378ms322µs473ns | r42: sending batch 1 EndTxn **to** (n1,s1):1 |

| 18s378ms348µs650ns | sending request **to** **local** **server** |

| 18s378ms364µs928ns | === SPAN **START**: /cockroach.roachpb.Internal/Batch === |

| 18s378ms370µs772ns | 1 EndTxn |

| 18s378ms373µs902ns | **read** has **no** clock uncertainty |

| 18s378ms378µs613ns | executing 1 requests |

| 18s378ms386µs573ns | **read**-**write** path |

| 18s378ms394µs316ns | command queue |

| 18s378ms417µs576ns | applied **timestamp** **cache** |

| 18s378ms588µs396ns | evaluated request |

| 18s378ms597µs715ns | acquired {raft,replica}mu |

| 18s383ms388µs599ns | applying command |

| 18s383ms494µs709ns | coordinator stops |

| 23s169ms850µs906ns | === SPAN **START**: **sql** txn implicit === |

| 23s169ms885µs921ns | executing 1/1: **SET** tracing = off |

| 23s169ms919µs90ns | **query** **not** supported **for** distSQL: **SET** / **SET** CLUSTER SETTING should never distribute |

+*--------------------+---------------------------------------------------------------------------------------------------------------+*

See Also

* [EXPLAIN](http://doc.cockroachchina.baidu.com/#develop/sql-statements/EXPLAIN/)
* [SET (session settings)](http://doc.cockroachchina.baidu.com/#develop/sql-statements/SET(session-variable)/)