术语表 【DRAFT】

> 请注意，本术语表中的定义简短而简单，旨在传达核心思想，而不是术语的完整细微之处。有关更多详细信息，请参阅正文中的参考资料。

[TOC]

异步（**asynchronous**）

不等待某些事情完成（例如，将数据发送到网络中的另一个节点），并且不会假设要花多长时间。请参阅第153页上的“同步与异步复制”，第284页上的“同步与异步网络”，以及第306页上的“系统模型与现实”。

原子（**atomic**）

1.在并发操作的上下文中：描述一个在单个时间点看起来生效的操作，所以另一个并发进程永远不会遇到处于“半完成”状态的操作。另见隔离。

2.在事务的上下文中：将一些写入操作分为一组，这组写入要么全部提交成功，要么遇到错误时全部回滚。参见第223页的“原子性”和第354页的“原子提交和两阶段提交（2PC）”。

背压（**backpressure**）

接收方接收数据速度较慢时，强制降低发送方的数据发送速度。也称为流量控制。请参阅第441页上的“消息系统”。

批处理（**batch process**）

一种计算，它将一些固定的（通常是大的）数据集作为输入，并将其他一些数据作为输出，而不修改输入。见第十章。

边界（**bounded**）

有一些已知的上限或大小。例如，网络延迟情况（请参阅“超时和未定义的延迟”在本页281）和数据集（请参阅第11章的介绍）。

拜占庭故障（**Byzantine fault**）

表现异常的节点，这种异常可能以任意方式出现，例如向其他节点发送矛盾或恶意消息。请参阅第304页上的“拜占庭故障”。

缓存（cache）

一种组件，通过存储最近使用过的数据，加快未来对相同数据的读取速度。缓存中通常存放部分数据：因此，如果缓存中缺少某些数据，则必须从某些底层较慢的数据存储系统中，获取完整的数据副本。

CAP定理（CAP theorem）

一个被广泛误解的理论结果，在实践中是没有用的。参见第336页的“CAP定理”。

因果关系（causality）

事件之间的依赖关系，当一件事发生在另一件事情之前。例如，后面的事件是对早期事件的回应，或者依赖于更早的事件，或者应该根据先前的事件来理解。请参阅第186页上的“发生之前的关系和并发性”和第339页上的“排序和因果关系”。

共识（consensus）

分布式计算的一个基本问题，就是让几个节点同意某些事情（例如，哪个节点应该是数据库集群的领导者）。问题比乍看起来要困难得多。请参阅第364页上的“容错共识”。

数据仓库（data warehouse）

一个数据库，其中来自几个不同的OLTP系统的数据已经被合并和准备用于分析目的。请参阅第91页上的“数据仓库”。

声明式（declarative）

描述某些东西应有的属性，但不知道如何实现它的确切步骤。在查询的上下文中，查询优化器采用声明性查询并决定如何最好地执行它。请参阅第42页上的“数据的查询语言”。

非规范化（denormalize）

为了加速读取，在标准数据集中引入一些冗余或重复数据，通常采用缓存或索引的形式。非规范化的值是一种预先计算的查询结果，像物化视图。请参见“单对象和多对象操作”（第228页）和“从同一事件日志中派生多个视图”（第461页）。

派生数据（derived data）

一种数据集，根据其他数据通过可重复运行的流程创建。必要时，你可以运行该流程再次创建派生数据。派生数据通常用于提高特定数据的读取速度。常见的派生数据有索引、缓存和物化视图。参见第三部分的介绍。

确定性（**deterministic**）

描述一个函数，如果给它相同的输入，则总是产生相同的输出。这意味着它不能依赖于随机数字、时间、网络通信或其他不可预测的事情。

分布式（**distributed**）

在由网络连接的多个节点上运行。对于部分节点故障，具有容错性：系统的一部分发生故障时，其他部分仍可以正常工作，通常情况下，软件无需了解故障相关的确切情况。请参阅第274页上的“故障和部分故障”。

持久（**durable**）

以某种方式存储数据，即使发生各种故障，也不会丢失数据。请参阅第226页上的“持久性”。

ETL（**Extract-Transform-Load**）

提取-转换-加载（Extract-Transform-Load）。从源数据库中提取数据，将其转换为更适合分析查询的形式，并将其加载到数据仓库或批处理系统中的过程。请参阅第91页上的“数据仓库”。

故障转移（**failover**）

在具有单一领导者的系统中，故障转移是将领导角色从一个节点转移到另一个节点的过程。请参阅第156页的“处理节点故障”。

容错（**fault-tolerant**）

如果出现问题（例如，机器崩溃或网络连接失败），可以自动恢复。请参阅第6页上的“可靠性”。

流量控制（**flow control**）

见背压（backpressure）。

追随者（**follower**）

一种数据副本，仅处理领导者发出的数据变更，不直接接受来自客户端的任何写入。也称为辅助、仆从、只读副本或热备份。请参阅第152页上的“领导和追随者”。

全文检索（full-text search）

通过任意关键字来搜索文本，通常具有附加特征，例如匹配类似的拼写词或同义词。全文索引是一种支持这种查询的次级索引。请参阅第88页上的“全文搜索和模糊索引”。

图（graph）

一种数据结构，由顶点（可以指向的东西，也称为节点或实体）和边（从一个顶点到另一个顶点的连接，也称为关系或弧）组成。请参阅第49页上的“和图相似的数据模型”。

散列（hash）

将输入转换为看起来像随机数值的函数。相同的输入会转换为相同的数值，不同的输入一般会转换为不同的数值，也可能转换为相同数值（也被称为冲突）。请参阅第203页的“根据键的散列值分隔”。

幂等（idempotent）

用于描述一种操作可以安全地重试执行，即执行多次的效果和执行一次的效果相同。请参阅第478页的“幂等”。

索引（index）

一种数据结构。通过索引，你可以根据特定字段的值，在所有数据记录中进行高效检索。请参阅第70页的“让数据库更强大的数据结构”。

隔离性（isolation）

在事务上下文中，用于描述并发执行事务的互相干扰程度。串行运行具有最强的隔离性，不过其它程度的隔离也通常被使用。请参阅第225页的“隔离”。

连接（join）

汇集有共同点的记录。在一个记录与另一个记录有关（外键，文档参考，图中的边）的情况下最常用，查询需要获取参考所指向的记录。请参阅第33页上的“多对一和多对多关系”和第393页上的“减少端连接和分组”。

领导者（leader）

当数据或服务被复制到多个节点时，由领导者分发已授权变更的数据副本。领导者可以通过某些协议选举产生，也可以由管理者手动选择。也被称为主人。请参阅第152页的“领导者和追随者”。

线性化（**linearizable**）

表现为系统中只有一份通过原子操作更新的数据副本。请参阅第324页的“线性化”。

局部性（**locality**）

一种性能优化方式，如果经常在相同的时间请求一些离散数据，把这些数据放到一个位置。请参阅第41页的“请求数据的局部性”。

锁（**lock**）

一种保证只有一个线程、节点或事务可以访问的机制，如果其它线程、节点或事务想访问相同元素，则必须等待锁被释放。请参阅第257页的“两段锁（2PL）”和301页的“领导者和锁”。

日志（**log**）

日志是一个只能以追加方式写入的文件，用于存放数据。预写式日志用于在存储引擎崩溃时恢复数据（请参阅第82页的“使二叉树更稳定”）；结构化日志存储引擎使用日志作为它的主要存储格式（请参阅第76页的“有序字符串表和日志结构的合并树”）；复制型日志用于把写入从领导者复制到追随者（请参阅第152页的“领导者和追随者”）；事件性日志可以表现为数据流（请参阅第446页的“分段日志”）。

物化（**materialize**）

急切地计算并写出结果，而不是在请求时计算。请参阅第101页的“聚合：数据立方和物化视图”和419页的“中间状态的物化”。

节点（**node**）

计算机上运行的一些软件的实例，通过网络与其他节点通信以完成某项任务。

规范化（**normalized**）

以没有冗余或重复的方式进行结构化。 在规范化数据库中，当某些数据发生变化时，您只需要在一个地方进行更改，而不是在许多不同的地方复制很多次。 请参阅第33页上的“多对一和多对多关系”。

OLAP（Online Analytic Processing）

在线分析处理。 通过对大量记录进行聚合（例如，计数，总和，平均）来表征的访问模式。 请参阅第90页上的“交易处理或分析？”。

OLTP（Online Transaction Processing）

在线事务处理。 访问模式的特点是快速查询，读取或写入少量记录，这些记录通常通过键索引。 请参阅第90页上的“交易处理或分析？”。

分区（partitioning）

将单机上的大型数据集或计算结果拆分为较小部分，并将其分布到多台机器上。 也称为分片。 见第6章。

百分位点（percentile）

通过计算有多少值高于或低于某个阈值来衡量值分布的方法。 例如，某个时间段的第95个百分位响应时间是时间t，则该时间段中，95％的请求完成时间小于t，5％的请求完成时间要比t长。 请参阅第13页上的“描述性能”。

主键（primary key）

唯一标识记录的值（通常是数字或字符串）。 在许多应用程序中，主键由系统在创建记录时生成（例如，按顺序或随机）; 它们通常不由用户设置。 另请参阅二级索引。

法定人数（quorum）

在操作完成之前，需要对操作进行投票的最少节点数量。 请参阅第179页上的“读和写的法定人数”。

再平衡（rebalance）

将数据或服务从一个节点移动到另一个节点以实现负载均衡。 请参阅第209页上的“再平衡分区”。

复制（replication）

在几个节点（副本）上保留相同数据的副本，以便在某些节点无法访问时，数据仍可访问。 请参阅第5章。

模式（schema）

一些数据结构的描述，包括其字段和数据类型。 可以在数据生命周期的不同点检查某些数据是否符合模式（请参阅第39页上的“文档模型中的模式灵活性”），模式可以随时间变化（请参阅第4章）。

次级索引（secondary index）

与主要数据存储器一起维护的附加数据结构，使您可以高效地搜索与某种条件相匹配的记录。 请参阅第85页上的“其他索引结构”和第206页上的“分区和二级索引”。

可序列化（serializable）

保证多个并发事务同时执行时，它们的行为与按顺序逐个执行事务相同。 请参阅第251页上的“可序列化”。

无共享（shared-nothing）

与共享内存或共享磁盘架构相比，独立节点（每个节点都有自己的CPU，内存和磁盘）通过传统网络连接。 见第二部分的介绍。

偏斜（skew）

1.各分区负载不平衡，例如某些分区有大量请求或数据，而其他分区则少得多。也被称为热点。请参阅第205页上的“工作负载偏斜和减轻热点”和第407页上的“处理偏斜”。

2.时间线异常导致事件以不期望的顺序出现。 请参阅第237页上的“快照隔离和可重复读取”中的关于读取偏斜的讨论，第246页上的“写入偏斜和模糊”中的写入偏斜以及第291页上的“订购事件的时间戳”中的时钟偏斜。

脑裂（split brain）

两个节点同时认为自己是领导者的情况，这种情况可能违反系统担保。 请参阅第156页的“处理节点中断”和第300页的“真相由多数定义”。

存储过程（stored procdure）

一种对事务逻辑进行编码的方式，它可以完全在数据库服务器上执行，事务执行期间无需与客户端通信。 请参阅第252页的“实际串行执行”。

流处理（stream process）

持续运行的计算。可以持续接收事件流作为输入，并得出一些输出。 见第11章。

同步（synchronous）

异步的反义词。

记录系统（system of record）

一个保存主要权威版本数据的系统，也被称为真相的来源。首先在这里写入数据变更，其他数据集可以从记录系统派生。 参见第三部分的介绍。

超时（timeout）

检测故障的最简单方法之一，即在一段时间内观察是否缺乏响应。 但是，不可能知道超时是由于远程节点的问题还是网络中的问题造成的。 请参阅第281页上的“超时和无限延迟”。

全序（total order）

一种比较事物的方法（例如时间戳），可以让您总是说出两件事中哪一件更大，哪件更小。总的来说，有些东西是无法比拟的（不能说哪个更大或更小）的顺序称为偏序。 请参见第341页的“因果顺序不是全序”。

事务（transaction）

为了简化错误处理和并发问题，将几个读写操作分组到一个逻辑单元中。 见第7章。

两阶段提交（2PC, two-phase commit）

一种确保多个数据库节点全部提交或全部中止事务的算法。 请参阅第354页上的“原子提交和两阶段提交（2PC）”。

两阶段锁定（2PL, two-phase locking）

一种用于实现可序列化隔离的算法，该算法通过事务获取对其读取或写入的所有数据的锁，直到事务结束。 请参阅第257页上的“两阶段锁定（2PL）”。

无边界（unbounded）

没有任何已知的上限或大小。 反义词是边界（bounded）。