并发控制介绍

常见的并发控制有：

MVCC：Multi-version Concurrency Control （多版本并发控制）。

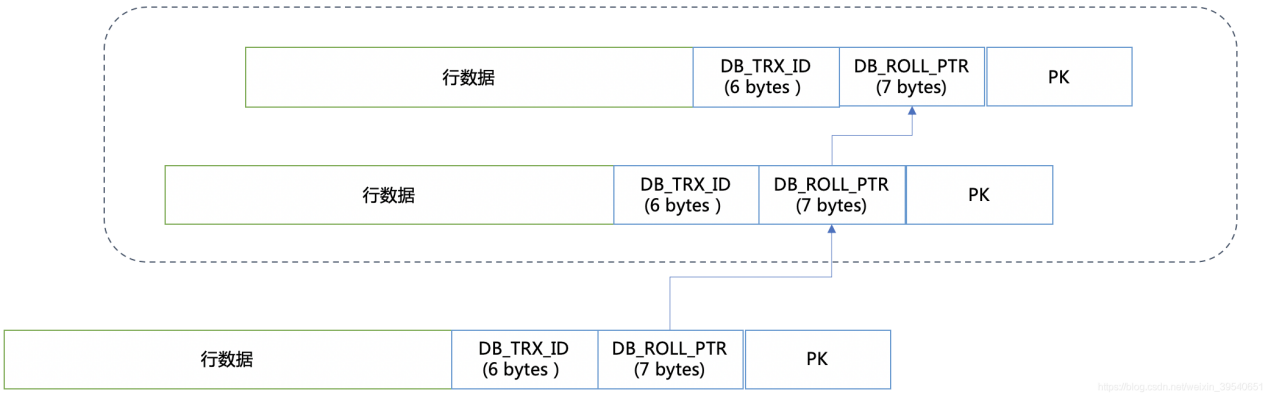
S2PL： Strict Two-Phase Locking（严格二阶段锁）：读写互斥，保证串行。

OCC：Optimistic Concurrency Control(乐观并发控制）：和悲观锁不同，乐观锁在提交前不加锁，提交时如果读取时的数据被其他事务修改并提交，则回退事务。

简单来说，在MVCC机制还没有出现之前，数据库中读写是互斥的，而通过MVCC机制使得读写不互斥。因此MVCC使关系型数据库并发读写能力得到很大的提高，而大多数OLTP系统中的85%以上是读操作。

Mysql的MVCC的实现方式

mysql innodb引擎的MVCC也是通过undo段来实现的，但是和oracle不同的是：mysql的多版本并发控制是基于记录级的，**mysql中通过undo来形成行的版本链。**



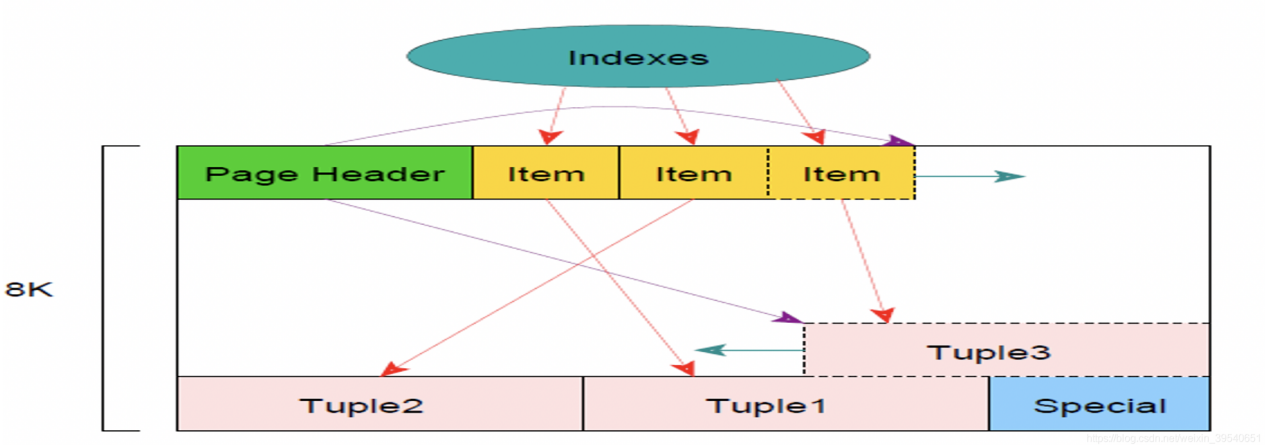
上图中的回滚指针(DB_ROLL_PTR)字段用来指写入回滚段(rollback segment)的 undo log record (撤销日志记录记录)。

而其具体做法就是通过两个隐含列来实现的：一个是db_trx_id，指出该行的事务ID，一个是db_roll_ptr，指出这条记录的pre-image数据在UNDO中的地址。

PostgreSQL的MVCC实现方式

- pg中没有undo这一概念
- **pg中的多版本并发是通过在表中数据行的多个版本来实现的**
  - 例如在一张表中我们要更新一条记录，pg并不是直接修改该数据，而是通过插入一条全新的数据，同时对老数据加以标识。

在pg中一个page页结构大致如下图所示



对应的数据结构为PageHeaderData：

而实现多版本是通过HeapTupleFields：

要想理解pg的多版本机制，首先要弄清楚几个关键的字段

- t_xmin：插入该元组的事务的txid；
- t_xmax：删除或更新该元组的事务的txid，如果尚未删除或更新该元组，则t_xmax设置为0；
- t_cid：命令ID（cid），这表示从0开始在当前事务中执行此命令之前已执行了多少个SQL命令；
- t_ctid：指向自身或新元组的元组标识符（tid），当该元组被更新时，该元组的t_ctid指向新的元组，否则，t_ctid指向自身。

关于这些字段的详细介绍见：PostgreSQL表的系统字段。

例子：

```
bill=# select xmin,xmax,ctid,* from t;
 xmin | xmax | ctid  | id | info
------+------+-------+----+------
  824 |    0 | (0,1) |  1 | a
(1 row)
```

更新该表中数据

可以看到新记录的xmin被置为823，ctid指向新的记录。

```
bill=# update t set info = 'b' where id=1;
UPDATE 1
bill=# select xmin,xmax,ctid,* from t;
 xmin | xmax | ctid  | id | info
------+------+-------+----+------
  825 |    0 | (0,2) |  1 | b
(1 row)

```

总结