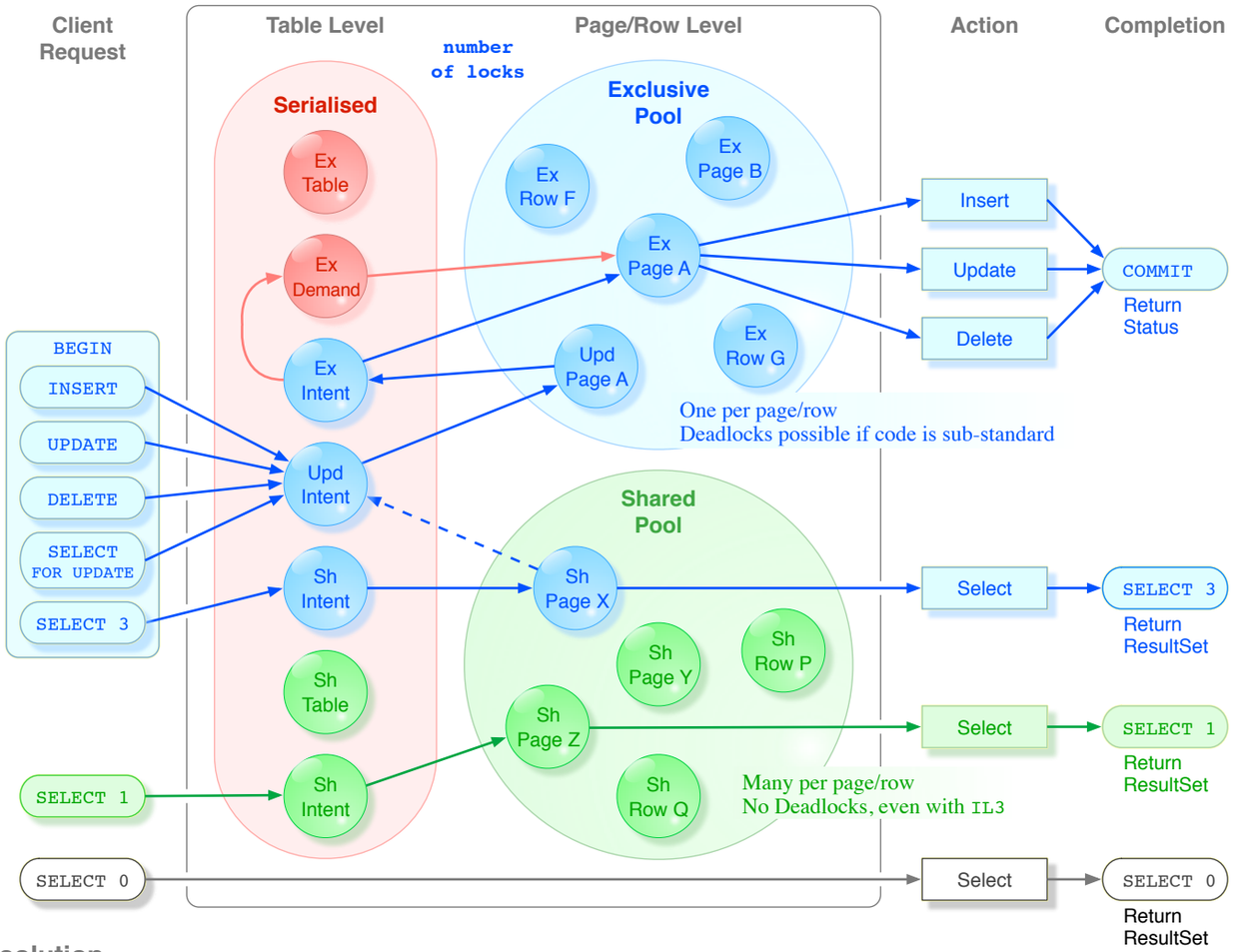
Lock Escalation



Resolution

Isolation Level, Command	Acquires	Blocked by	Not Blocked by	Blocks	Not Blocks	Duration
0 Read Uncommitted `SELECT`	*None*	*None*	*Any*	*None*	*Any*	Statement
1 Read Committed **ASE Default** `SELECT`	Shared	Exclusive	Shared	Exclusive	Shared	Statement
2 Repeatable Read (DPL/DRL Only) `SELECT`	Shared	Exclusive	Shared	Exclusive	Shared	Transaction
3 Serialisable **ANSI SQL Default** `SELECT HOLDLOCK`	Shared	Exclusive	Shared	Exclusive	Shared	Transaction
3 (15.7, DRL Only) `SELECT FOR UPDATE`	Update ... Exclusive	Shared Exclusive		Shared Exclusive		Transaction
3 `INSERT/UPDATE/DELETE`	Update ... Exclusive	Shared Exclusive		Shared Exclusive		Transaction

Note

- This is a simplified view, but it retains all the important aspects, and shows the Lock Types that are commonly visible; the entire set of Lock Types is on below
- Live objects have a shadow. Flat objects define the context in which an object is alive
- Technically, `BEGIN TRAN` and `COMMIT TRAN` define the chronological extent of a live Transaction, but here we wish to observe the extent of each constituent verb.
- Common lock types for ASE 11.9.2 to 15.7.x are shown here
- Cursor class and considerations (suh-low) not shown
- Lock Promotion is shown on the next page
- Requests granted immediately rarely appear, and are not shown
- Isolation Level 2 on APL tables executes at Isolation Level 3
- During `INSERT/UPDATE/DELETE` if a Sh-Page or Sh-Row lock is not held, it must be acquired first (`SELECT 3` is performed)
- If the database has Declarative Referential Integrity (Foreign Keys), when Exclusive locks are held for a row, ASE will acquire a Sh-Intent lock on the parent row; the efficiency in that should not be under-estimated
- An Ex-Intent lock can get skipped up to 3 times by Sh-Page; it then escalates to Ex-Demand, which is not skipped

Lock Type [66]

```
{ Ex_ | Sh_ | Update_ } { intent | page | row | table } [ -blk | -demand [ -blk | -request ] | -request ]
```

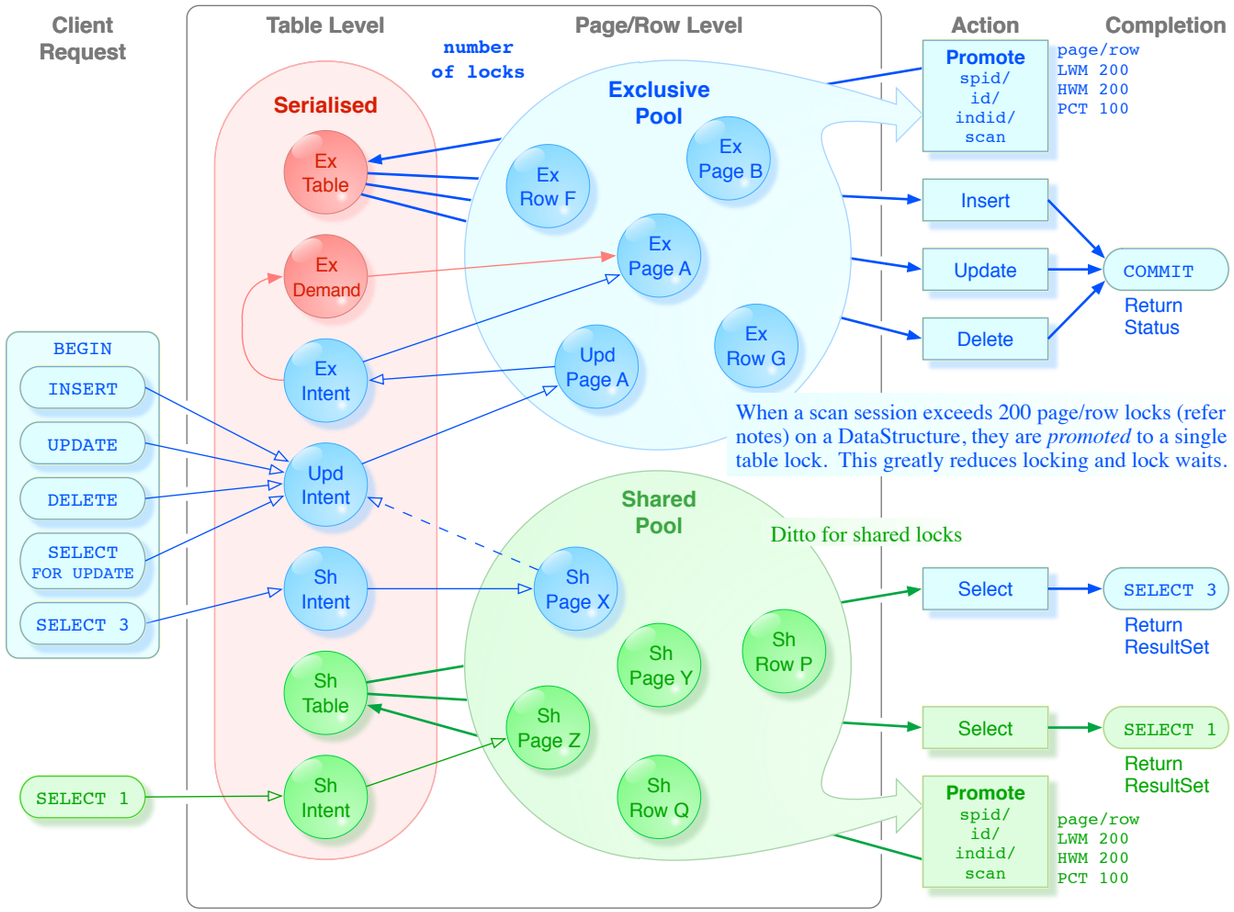
To obtain a complete list of Lock Types, execute:

```
SELECT  number,
        name
    FROM master..spt_values
    WHERE type = "L"
    ORDER BY number
```

Caveat re "Two Phased Locking (2PL)"

- The *Sybase ASE* Locking strategy is, well, ordinary, with internal Escalation (based on serialisation or hierarchy of Lock Types, this page) and Promotion (next page).
- It is not a "Two Phased Locking" strategy, there are no phases. Note sagely, the serialisation of table level locks, vs the non-serialisation of page/row level locks.
- However the "literature" in this field, which is a flood of unscientific filth, promotes the notion that "Two Phased Locking" is the **alternative** to Multi-Version Concurrency Control. It is not.
- First MVCC needs to be appreciated for what it really is: a complete failure, because it (a) breaks the first principle of a database (Single Version of the Truth), (b) appears to perform the notion of private versions (which is nothing less than schizophrenic, because such versions are stale in an online database), and (c) contrary to the label, it does not control concurrency in any way.
- This **new** "Two Phased Locking" exists only in the context of MVCC, it is the latest proposition of many, purported to "improve" the hysterical operation of MVCC.
- What it does is irrelevant to a sane person, or to Locking in general, or to an ordinary Locking strategy (with or without enhancements), because (a) its need arises only in the schizophrenic context of MVCC, and (b) it is an attempt at fixing the symptom rather than addressing the cause, which is pathetic, and guaranteed to be replaced, same as the previous "improvements" to MVCC.
- Therefore, do not view the *ASE* Locking strategy through the myopic lens of either "2PL", or insane lens of MVCC which is its context, but perceive it as the ordinary locking required to resolve resource contention (since the 1950's), with substantial advancements by *Sybase*, and unchanged since 1980. Truth is single and permanent, falsities are multiple, ever-changing.

Lock Promotion



OnLine Transaction Processing

- *Sybase ASE* is a brilliant high-volume **OLTP Server**, but that does not mean that any code that is thrown at it will be magically transformed into OLTP.
- An OLTP system is achieved only through the diligent and consistent implementation of **OLTP Standards**. That means:
  - **Smallest Possible Transactions** that are carefully designed to minimise contention
  - coded using a standard **Transaction Template** (sequenced code blocks)
  - genuine **Optimistic Locking** in the database ***and*** the application.
  - an overall **Access Sequence**
- Systems implemented using OLTP standards thrive on an OLTP Server such as *ASE*, they do not suffer locking problems.
- Naïve implementations; the lack of understanding of transactional and contention issues; the absence of standard methods to overcome them, will result in a contentious system, that suffers locking problems at every level, on any platform. Fixing one level only serves to identiy the next level of contention.
- A configuration of anything more than 50,000 locks (10,000 on a small server) means OLTP Standards have not been implemented in the system, and thus OLTP behaviour cannot be expected from it, except by faith in magic.
- Increasing the number locks merely transfers the location of the unaddressed application problem to the Lock Manager.

Keyword

Isol/Keyword	Meaning
`1 HOLDLOCK`	Hold Lock until end of Transaction
`2 NOHOLDLOCK` `3 NOHOLDLOCK`	Release Lock after Read completes (*not* end of tran)
`3 FOR UPDATE` (15.7, DRL Only)	Obtain Exclusive Lock until end of Transaction
`READPAST`	Silently skip rows with Exclusive Locks
`NOLOCK`	Table Alias *Common Mistake !*

Specifying Isolation in Context

Transaction/Session

```
SET TRANSACTION ISOLATION LEVEL {0|1|2|3}
SET LOCK {WAIT [seconds] | NOWAIT}
LOCK TABLE IN {SHARED | EXCLUSIVE} MODE
```

Query

```
SELECT ...
    FROM table_a
```

Table

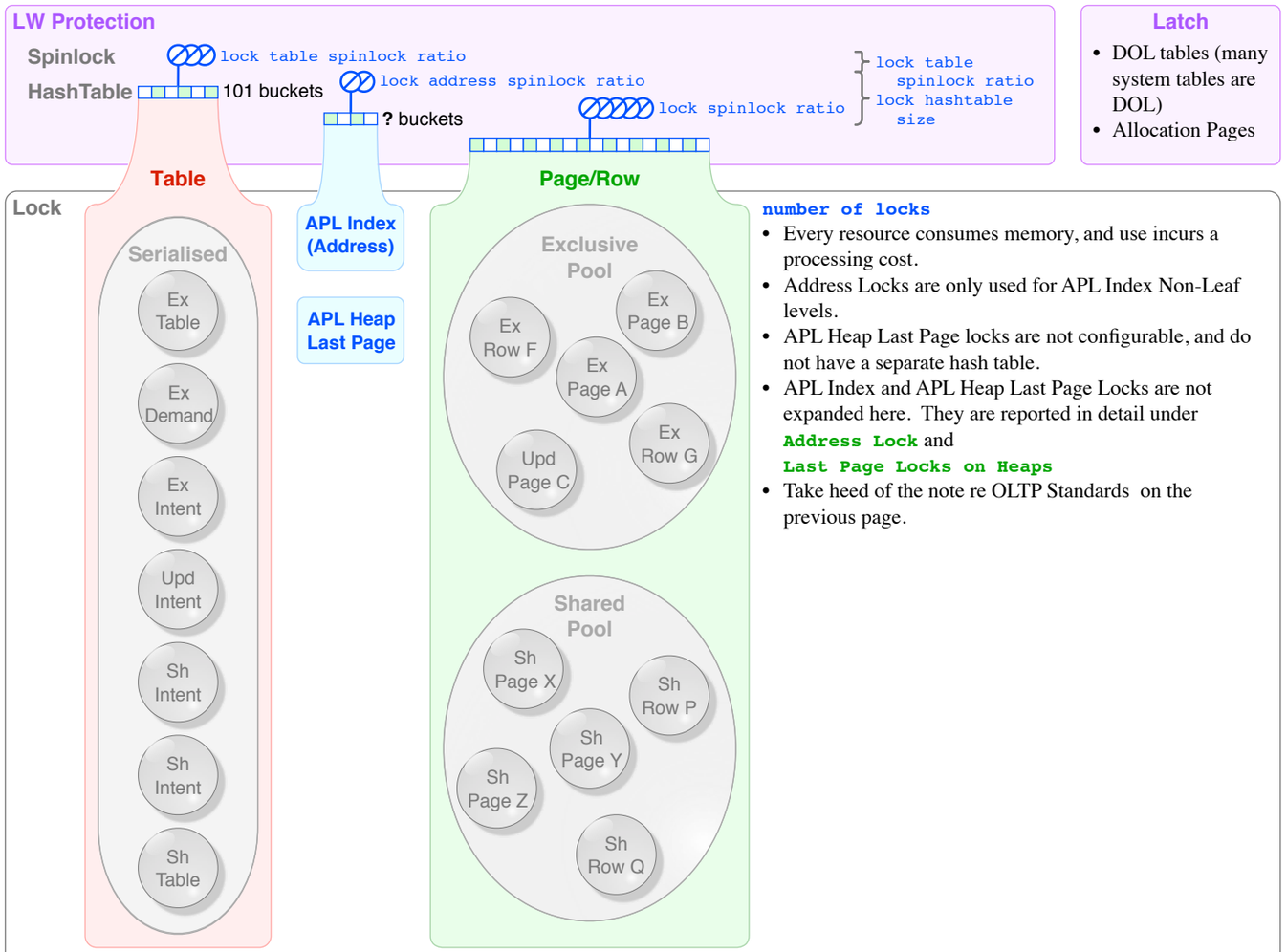
```
FROM table_b [HOLDLOCK]
```

```
READPAST
NOHOLDLOCK
FOR UPDATE
AT ISOLATION {0|1|2|3}
```

Note

- Lock Promotion for a single object is shown
- Lock Promotion is per scan session per Logical DataStructure (all partitions), but locks are accrued per spid; therefore locks per spid may exceed thresholds
- Lock Promotion may be blocked due to conflicting locks held by another spid; the spid continues without Promotion. That results in increased locks.
- Contrary to the mythology, and consistent with science, row level locking massively *increases* the number of locks in use, and therefore the opportunity for lock contention.
- Lock Promotion thresholds `{page|row} lock promotion {LWM|HWM|PCT}` may be reset at the Table or Database or Server levels.
  - However, that is not recommended as it is anti-social (imposes lock waits on other users) and it will make your server lock-bound.
- For a full-blown example, showing server level monitor metrics for 24 hours, inspect the following sections in this **server monitoring report**:
  - Kernel/Context Switch
  - Lock Manager

Configuration



number of locks

- Every resource consumes memory, and use incurs a processing cost.
- Address Locks are only used for APL Index Non-Leaf levels.
- APL Heap Last Page locks are not configurable, and do not have a separate hash table.
- APL Index and APL Heap Last Page Locks are not expanded here. They are reported in detail under **Address Lock** and **Last Page Locks on Heaps**
- Take heed of the note re OLTP Standards on the previous page.

- The number of locks are determined, and navigated by, three hash tables.
- The total hash table size is configurable, and the small Table and Address hash tables subtract a fixed number from it.
- The total hash table is configured via lock hashtable size. It must be set proportionate to number of locks. The two parameters together determine the number of locks per hash bucket. The defaults are 10,000 locks ÷ 2048 hash buckets = 5 lock chain length per hash bucket.
- Since **Avg Chain Length** should be maintained at ≤ 5, lock hashtable size should be set to number of locks ÷ 5.
- The hash tables in turn, are protected by Spinlocks, configurable via a Spinlock Ratio (no of hash buckets per Spinlock; which leaves it independent of the target, that is likely to change.

DataStructure

- For tables that suffer Allocation Page contention (mass inserts), tune des_greedyalloc
- For genuine Relation tables (Clustered Index, APL only, composite Primary Key), tune ascinserts

Contention Resolution

- In the beginning, almost all resource contention was resolved using the Locks. As ASE progressed and matured, more efficient forms of contention resolution were implemented *for specific resources*. Currently there are **four types** of contention resolution methods related to content.
- Lightweight Protection is a technical term for fast memory-only contention resolution mechanism.
- Latches and LW Protection for DOL are neither visible nor monitored, and they cannot be configured. The only metrics reported for them are under **Task Context Switches**, which serves another purpose, therefore it is indicative but indirect. Further, the metric names do not reflect the contentious resource or actual use.
- Generally Spinlocks are reported in their relevant sections. There is no section for Spinlocks that are used for DOL Index Root pages.

Mechanism	Reported As	Really Means
Lock	**Lock Management** 4 categories; full detail	(Explicit name given)
Latch	**TCS/Latch Contention**	DOL Index & OAM
LW Protection	**TCS/Modify Conflicts**	Dirty Reads & DOL sys tables
Spinlock		DOL Index/Root

Contention Object per Resource

Contention On	Is Resolved By	
	APL	DOL
AllocationPage	Latch	Latch
OAM Page	Page Lock	Latch
Index/Root	Address Lock	Spinlock
Index/Non-leaf	Address Lock	Latch
CI/Leaf Page	Page Lock	*No CI for DOL*
Heap	Page Lock	Page/Row Lock
	Last Page Lock	
Data	Page Lock	Page/Row Lock

Configuration

Lock Type	Hash Buckets	Spinlock Ratio	Spinlocks
Page/Row Default	lock hashtable size 2048	lock spinlock ratio 85	24
Table Default	101 101	lock table spinlock ratio 20	5
APL Index (Address) Default	? (Fixed) ? (Fixed)	lock address spinlock ratio 100	

DOL "No Index Lock" Myth

- DOL tables have the **exact same** contention re Index pages, and the **same** need to resolve it: the difference is, Latches are used instead of Locks.
- Declaring that "DOL tables do not use Locks on index pages" and failing to mention that they use Latches, is gross misrepresentation.
- If DOL tables are used, **TCS/Latch Contention** will be heavy (compare against **TCS/Address Locks**).