1. Problems Happen
   1. Dirty reads
   2. Unrepeatable read
   3. Phantom rows
   4. Lost updates
2. Type of Lock
   1. Shared lock
   2. Exclusive
   3. Update lock
   4. Intent Lock
   5. Schema locks
   6. Bulk update locks
3. transaction isolation level
   1. Read Uncommitted
   2. Read Committed (Default )
   3. Repeatable Read
   4. Serializable
4. Locking Hints
   1. HOLDLOCK
   2. NOLOCK
   3. PAGLOCK
   4. READCOMMITTED
   5. READPAST
   6. READUNCOMMITTED
   7. REPEATABLEREAD
   8. ROWLOCK
   9. SERIALIZABLE
   10. TABLOCK
   11. TABLOCKX
   12. UPDLOCK
   13. XLOCK
5. T
6. T

There are 4 kinds of major problems caused because of concurrency, below table shows the details of the same.

|  |  |  |
| --- | --- | --- |
| **Problems** | **Short description** | **Explanation** |
| Dirty reads | "Dirty Read" occurs when one transaction is reading a record, which is part of a half, finished work of other transaction. | * User A and user B are seeing value as "5". * User B changes the value "5" to "2". * User A is still seeing the value as "5"....Dirty read has happened. |
| Unrepeatable read | In every data read if you get different values then it's an "Unrepeatable Read" problem. | * User A is seeing value as "5". * User B changes the value"5" to "2". * User A refreshes see values "5", he is surprised....unrepeatable read has happened. |
| Phantom rows | If "UPDATE" and "DELETE" SQL statements does not affect the data then it can be "Phantom Rows" problem. | * User A updates all value "5' to "2". * User B inserts a new record with value "2". * User A selects all record with value "2' if all the values have changed, he is surprised to still find value "2" records.....Phantom rows have been inserted. |
| Lost updates | "Lost Updates" are scenario where one updates which is successfully written to database is overwritten with other updates of other transaction. | * User A updates all value form "5" to "2". * User B comes and updates all "2" values to "5". * User A has lost all his updates. |

Below is a chart which shows which transaction isolation level solves which problems of concurrency.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Read committed(S)** | **Repeatable read(I)** | **Serializable** | **Read Uncommitted** |
| Dirty reads | Solves | Solves | Solves | X |
| Lost updates | X | Solves | Solves | X |
| Non repeatable reads | X | Solves | Solves | X |
| Phantom rows | X | X | Solves | X |

Below is the list of Locks

|  |  |  |  |
| --- | --- | --- | --- |
|  | **When to use?** | **Reads Allowed** | **Writes Allowed** |
| Shared lock | When you want only to read and you do not want any other transactions to do update. | Yes | No |
| Exclusive | When you want to modify data and you do not want anyone to read the transaction, neither you want anyone to update. | No | No |
| Update lock | This is a hybrid lock. This lock is used when you want to do update operation which passes through multiple phases before the actual update happens. It first starts with shared lock in the read phase and then on the actual update it acquires an exclusive lock. |  |  |
|  | Read phase | Yes | No |
|  | Manipulating phase | Yes | No |
|  | Update phas | No | No |
| Intent Lock ( Demand locks) | Intent lock is for lock hierarchy. This lock is used when you want to lock resources down in the hierarchy. For example a shared intent lock on a table means shared locks are placed on pages and rows with the table. | NA | NA |
| Schema locks | When you are changing table structure. | No | No |
| Bulk update locks | Used when you are doing bulk updates | Table level No | Table level No |

**Locking hints** override the current transaction **isolation level** for the session. Below is the full list from MSDN

<http://msdn.microsoft.com/en-us/library/aa213026(v=sql.80).aspx>

|  |  |
| --- | --- |
| **Locking hint** | **Description** |
| HOLDLOCK | Hold a shared lock until completion of the transaction instead of releasing the lock as soon as the required table, row, or data page is no longer required. HOLDLOCK is equivalent to SERIALIZABLE. |
| NOLOCK | Do not issue shared locks and do not honor exclusive locks. When this option is in effect, it is possible to read an uncommitted transaction or a set of pages that are rolled back in the middle of a read. Dirty reads are possible. Only applies to the SELECT statement. |
| PAGLOCK | Use page locks where a single table lock would usually be taken. |
| READCOMMITTED | Perform a scan with the same locking semantics as a transaction running at the READ COMMITTED isolation level. By default, SQL Server 2000 operates at this isolation level. |
| READPAST | Skip locked rows. This option causes a transaction to skip rows locked by other transactions that would ordinarily appear in the result set, rather than block the transaction waiting for the other transactions to release their locks on these rows. The READPAST lock hint applies only to transactions operating at READ COMMITTED isolation and will read only past row-level locks. Applies only to the SELECT statement. |
| READUNCOMMITTED | Equivalent to NOLOCK. |
| REPEATABLEREAD | Perform a scan with the same locking semantics as a transaction running at the REPEATABLE READ isolation level. |
| ROWLOCK | Use row-level locks instead of the coarser-grained page- and table-level locks. |
| SERIALIZABLE | Perform a scan with the same locking semantics as a transaction running at the SERIALIZABLE isolation level. Equivalent to HOLDLOCK. |
| TABLOCK | Use a table lock instead of the finer-grained row- or page-level locks. SQL Server holds this lock until the end of the statement. However, if you also specify HOLDLOCK, the lock is held until the end of the transaction. |
| TABLOCKX | Use an exclusive lock on a table. This lock prevents others from reading or updating the table and is held until the end of the statement or transaction. |
| UPDLOCK | Use update locks instead of shared locks while reading a table, and hold locks until the end of the statement or transaction. UPDLOCK has the advantage of allowing you to read data (without blocking other readers) and update it later with the assurance that the data has not changed since you last read it. |
| XLOCK | Use an exclusive lock that will be held until the end of the transaction on all data processed by the statement. This lock can be specified with either PAGLOCK or TABLOCK, in which case the exclusive lock applies to the appropriate level of granularity. |