David's Drupalcon Performance Presentation

- ☐ Indexes
  - ☐ Stored as a tree
    - ☐ Trunk is first portion of index
    - ☐ Continues to branch with subsequent portions of the index
    - ☐ Leaves are pointers to rows
  - ☐ Index usage
    - ☐ The database can only use an index starting from the leftmost part with no interruptions
    - ☐ Order of index column usage
      - ☐ Satisfaction of all WHERE criteria
        - ☐ Order of WHERE criteria is irrelevant
        - ☐ Any ORs or XORs in the WHERE clause cause inefficient index usage, if any index usage is possible at all
        - ☐ If the WHERE clause consists of tautologies or nothing, the WHERE criteria are considered implicitly satisfied
      - ☐ ORDER BY criteria
        - ☐ Order must match the index
        - ☐ Must be only ASC or DESC to use an index
      - ☐ Value retrieval
        - ☐ Because indexes are trees based on the values in the rows, the database can use remaining portions of an index to retrieve values without following the index pointers to the actual row records.
        - ☐ Because indexes are generally cached (and rows are not), this is a valuable technique for reducing disk access.
      - ☐ Example
        - ☐ SELECT (5) FROM {table} WHERE (1) = value1 AND (2) = value2 ORDER BY (3), (4)
        - ☐ With index {(1), (2), (3), (4), (5)}, this query will never need to access anything other than the index, nor will it have to perform any additional processing to return results.
    - ☐ Generally, the database cannot use more than one index per table while executing a query.
      - ☐ An exception
        - ☐ SELECT column1 FROM {table} WHERE column2 = value2 AND/OR column3 = value3
        - ☐ {table} has two indexes: {column2} and {column3}
        - ☐ Index statistics indicate that obtaining the rows where column2 = value2, separately obtaining the rows column3 = value3, and finally applying the criteria to the union of the two sets is faster than a table scan.
    - ☐ Extra columns may exist as a suffix on an index with only a minor performance impact.
- ☐ Engines and performance
  - ☐ MyISAM
    - ☐ Locks
      - ☐ INSERTs are table-level (with one exception)
      - ☐ UPDATEs are table-level
      - ☐ DELETEs are table-level
      - ☐ SELECTs wait on write locks
    - ☐ Other notes
      - ☐ TRUNCATE is effectively O(1)
      - ☐ INSERT is effectively O(1) when
        - ☐ The primary key is autoincrement
        - ☐ No UNIQUE indexes
        - ☐ This special case is possible because no checking is necessary for key conflicts with existing rows
  - ☐ InnoDB
    - ☐ Locks
      - ☐ UPDATEs are row-level

David's Drupalcon Performance Presentation

- INSERTs are table-level
- DELETEs are row-level
- SELECTs use MVCC, so they never wait on locks
- Multi-version concurrency control
  - Keeps revisions around as necessary to satisfy read requests without waiting for concurrent data changes to complete
- Other notes
  - Even though INSERTs are table-level, they can buffered in the transaction system

- Replication
  - When it's the answer
    - You have SELECT queries that don't require access to perfectly fresh data
    - You want a server to run backups from without burdening the master
    - You want a hot standby in case of failure of the master
  - When it's not the answer
    - If you have high levels of lock contention and you're looking for performance benefits
      - Replication simply runs the data-modification queries from the master server on the slaves
      - So, the exact same locking behavior happens on the slaves