more efficient in most cases than the block-nested loop algorithm. Joins such as those shown here can be optimized in this manner:

```
SELECT *
    FROM t1
    JOIN t2
        ON t1.c1=t2.c1;

SELECT *
    FROM t1
    JOIN t2
        ON (t1.c1 = t2.c1 AND t1.c2 < t2.c2)
    JOIN t3
        ON (t2.c1 = t3.c1)
```

Hash joins can also be used for Cartesian products—that is, when no join condition is specified.

You can see when the hash join optimization is being used for a particular query using `EXPLAIN FORMAT=TREE` or `EXPLAIN ANALYZE`. (In MySQL 8.0.20 and later, you can also use `EXPLAIN`, omitting `FORMAT=TREE`.)

The amount of memory available to a hash join is limited by the value of `join_buffer_size`. A hash join that requires more than this much memory is executed on disk; the number of disk files that can be used by an on-disk hash join is limited by `open_files_limit`.

As of MySQL 8.0.19, the `hash_join` optimizer switch which was introduced in MySQL 8.0.18 no longer supported (hash_join=on still appears as part of the value of optimizer_switch, but setting it no longer has any effect). The `HASH_JOIN` and `NO_HASH_JOIN` optimizer hints are also no longer supported. The switch and the hint are both now deprecated; expect them to be removed in a future MySQL release. In MySQL 8.0.18 and later, hash joins can be disabled using the `NO_BNL` optimizer switch.

In MySQL 8.0.20 and later, block nested loop is no longer used in the MySQL server, and a hash join is employed any time a block nested loop would have been used previously, even when the query contains no equi-join conditions. This applies to inner non-equijoins, semijoins, antijoins, left outer joins, and right outer joins. TThe `block_nested_loop` flag for the `optimizer_switch` system variable as well as the `BNL` and `NO_BNL` optimizer hints are still supported, but henceforth control use of hash joins only. In addition, both inner and outer joins (including semijoins and antijoins) can now employ batched key access (BKA), which allocates join buffer memory incrementally so that individual queries need not use up large amounts of resources that they do not actually require for resolution. BKA for inner joins only is supported starting with MySQL 8.0.18.

MySQL 8.0.20 also replaces the executor used in previous versions of MySQL with the iterator executor. This work includes replacement of the old index subquery engines that governed queries of the form `WHERE value IN (SELECT column FROM table WHERE ...)` for those `IN` queries which have not been optimized as semijoins, as well as queries materialized in the same form, which formerly depended on the old executor.

For more information and examples, see Section 8.2.1.4, “Hash Join Optimization”. See also Batched Key Access Joins.

- **EXPLAIN ANALYZE Statement.** A new form of the `EXPLAIN` statement, `EXPLAIN ANALYZE`, is implemented in MySQL 8.0.18, providing expanded information about the execution of `SELECT` statements in `TREE` format for each iterator used in processing the query, and making it possible to compare estimated cost with the actual cost of the query. This information includes startup cost, total cost, number of rows returned by this iterator, and the number of loops executed.

  In MySQL 8.0.21 and later, this statement also supports a `FORMAT=TREE` specifier. `TREE` is the only supported format.