returns them to the MySQL server which evaluates the `WHERE` condition for the rows. With ICP enabled, and if parts of the `WHERE` condition can be evaluated by using only columns from the index, the MySQL server pushes this part of the `WHERE` condition down to the storage engine. The storage engine then evaluates the pushed index condition by using the index entry and only if this is satisfied is the row read from the table. ICP can reduce the number of times the storage engine must access the base table and the number of times the MySQL server must access the storage engine.

Applicability of the Index Condition Pushdown optimization is subject to these conditions:

- ICP is used for the `range`, `ref`, `eq_ref`, and `ref_or_null` access methods when there is a need to access full table rows.
- ICP can be used for `InnoDB` and `MyISAM` tables, including partitioned `InnoDB` and `MyISAM` tables.
- For `InnoDB` tables, ICP is used only for secondary indexes. The goal of ICP is to reduce the number of full-row reads and thereby reduce I/O operations. For `InnoDB` clustered indexes, the complete record is already read into the `InnoDB` buffer. Using ICP in this case does not reduce I/O.
- ICP is not supported with secondary indexes created on virtual generated columns. `InnoDB` supports secondary indexes on virtual generated columns.
- Conditions that refer to subqueries cannot be pushed down.
- Conditions that refer to stored functions cannot be pushed down. Storage engines cannot invoke stored functions.
- Triggered conditions cannot be pushed down. (For information about triggered conditions, see Section 8.2.2.3, “Optimizing Subqueries with the EXISTS Strategy”.)

To understand how this optimization works, first consider how an index scan proceeds when Index Condition Pushdown is not used:

1. Get the next row, first by reading the index tuple, and then by using the index tuple to locate and read the full table row.
2. Test the part of the `WHERE` condition that applies to this table. Accept or reject the row based on the test result.

Using Index Condition Pushdown, the scan proceeds like this instead:

1. Get the next row's index tuple (but not the full table row).
2. Test the part of the `WHERE` condition that applies to this table and can be checked using only index columns. If the condition is not satisfied, proceed to the index tuple for the next row.
3. If the condition is satisfied, use the index tuple to locate and read the full table row.
4. Test the remaining part of the `WHERE` condition that applies to this table. Accept or reject the row based on the test result.

`EXPLAIN` output shows `Using index condition` in the `Extra` column when Index Condition Pushdown is used. It does not show `Using index` because that does not apply when full table rows must be read.

Suppose that a table contains information about people and their addresses and that the table has an index defined as `INDEX (zipcode, lastname, firstname)`. If we know a person's `zipcode` value but are not sure about the last name, we can search like this:

```
SELECT * FROM people
```