📋 Free Checklist — 12 Points

Database Migration Safety Checklist

Review every schema change before it hits production. Catch dangerous migrations, prevent outages, and ship safer SQL.

✅ The 12-Point Checklist

1 **Read the migration AND the schema it produces**

Don't just review the `ALTER TABLE` statement. Run it against a copy of production schema and inspect the result. A migration that looks harmless in isolation can break foreign key constraints or indexes.

TIP

2 **Check for destructive changes**

Dropping columns, tables, or indexes destroys data permanently. Verify the column is unused by application code. Search your entire codebase for references before approving a `DROP`.

CRITICAL

3 **Adding NOT NULL without a default value**

`ALTER TABLE ... ADD COLUMN ... NOT NULL` without a `DEFAULT` will fail on tables with existing rows. Add the default first, then apply NOT NULL in a follow-up migration.

CRITICAL

4

Renaming columns in-place

Renaming a column with `ALTER TABLE ... RENAME COLUMN` is safe in PostgreSQL but can cause downtime in MySQL (rebuilds the table). Consider adding a new column, backfilling, and deprecating the old one.

CAUTION

5

Verify index additions won't lock tables

Creating indexes on large tables can lock writes for minutes or hours. Use `CREATE INDEX CONCURRENTLY` (PostgreSQL) or `ALGORITHM=INPLACE, LOCK=NONE` (MySQL 5.6+) to avoid blocking.

CAUTION

6

Check foreign key constraint timing

Adding a foreign key with `ALTER TABLE` validates all existing rows, which can be slow. On PostgreSQL, add the constraint as `NOT VALID` first, then validate separately to avoid long locks.

CAUTION

7

Confirm type changes are backward-compatible

Widening a column (`INT` → `BIGINT`) is usually safe. Narrowing (`VARCHAR(255)` → `VARCHAR(50)`) can truncate data. Changing `TEXT` → `JSONB` may fail on invalid values.

CAUTION

8

Look for missing indexes on new foreign keys

Every foreign key should have a corresponding index for fast JOINs and cascade operations. Without it, DELETEs and UPDATEs on the parent table become full table scans on the child.

TIP

9 Test the migration on a copy of production data

A migration that runs in 10ms on your local machine with 100 rows might take 30 minutes on a 50M-row production table. Always test migrations on realistic data volumes.

CRITICAL

10 Ensure migrations are idempotent where possible

Use `IF NOT EXISTS` and `IF EXISTS` guards. If a migration fails midway and is re-run, it should not crash. This is especially important in CI/CD pipelines.

TIP

11 Review default values for correctness

A `DEFAULT '1970-01-01'` on a timestamp column or `DEFAULT 0` on a price column can silently corrupt data. Defaults should be semantically meaningful or explicitly NULL.

CAUTION

12 Have a rollback plan

Every migration should be reversible or have a documented manual rollback. In an emergency, you need to restore service in minutes, not hours. Test the rollback before deploying.

CRITICAL

Download the interactive version at **schemalens.tech/migration-checklist** · © 2026 SchemaLens