

One giant table is not a database.

D8A.ACADEMY/RESOURCES/NORMAL-FORMS

THE IDEA each fact once update one place only anomalies update, insert, delete bugs normalize = split tables + keys	1NF · ATOMIC one value per cell no value lists no phone1, phone2 rows, not columns a real key every row reachable
2NF · WHOLE KEY no partial deps composite keys only (order_id, product_id) classic case product_name ← hangs off half the key.	3NF · KEY ONLY no transitive deps non-key → non-key zip → city on orders move city out the test key, whole key, nothing but
THE SMELLS name x 40k rows typo = 2 customers update hits 40 rows the fact lives 40x delete an order lose the customer too	DENORMALIZE ON PURPOSE OLTP stays 3NF writes must be safe star schema is flat reads win know which side ← the follow-up.

WORTH MEMORIZING

01 The split, in one before/after

The mega-table stores the customer name on every order. 3NF stores it once.

```

-- before: one fact, 40,000 copies
-- orders(id, customer_name, customer_city, ...)

-- after: the fact lives once
CREATE TABLE customers (
  id BIGINT PRIMARY KEY,
  name TEXT NOT NULL, city TEXT);
CREATE TABLE orders (
  id BIGINT PRIMARY KEY,
  customer_id BIGINT REFERENCES customers(id));

```

→ A rename is now UPDATE one row, not UPDATE 40,000 where you hope the spelling matched. That is the whole sales pitch of 3NF.

02 Trap: which anomaly bit you

The three failure modes normalization exists to prevent, with their tells.

ANOMALY	THE TELL	THE CURE
update	same edit, many rows	store the fact once
insert	cannot add a customer with no order	own table
delete	last order gone, customer gone	own table

→ Interview phrasing: normalize for writes, denormalize for reads. OLTP lives near 3NF, the warehouse relaxes it on purpose, and knowing which side you are on is the actual answer.