

Your customer moved. Your warehouse disagrees.

THE PROBLEM

dimensions change	city, plan, segment
facts point at them	which version?
slowly changing	← slowly, not never.

TYPE 1

overwrite	no history, cheap
good for	typo fixes, names
reports rewrite	last year moves too

TYPE 2

new row per change	full history
valid_from / to	+ is_current flag
surrogate key	← facts join on this one.

TYPE 3

previous_ column	one step back only
good for	a rare reorg
gets ugly	at change number two

PICK ONE

default: type 2	history is cheap now
type 1 inside it	hybrid, per column
dbt snapshots	type 2 in 10 lines

GOTCHAS

join natural key	fans out per version
no end date	9999-12-31, not NULL
late changes	backfill the versions

WORTH MEMORIZING

01 Type 2, in two statements

Close the current version, open a new one. Facts written from now on join the new surrogate key.

```
-- customer 42 moved from Paris to Lyon today
UPDATE dim_customer
  SET valid_to = '2026-10-04', is_current = false
  WHERE customer_id = 42 AND is_current;

INSERT INTO dim_customer
(customer_sk, customer_id, city,
 valid_from, valid_to, is_current)
VALUES (9871, 42, 'Lyon',
       '2026-10-04', '9999-12-31', true);
-- 2024 revenue still sits on the Paris row (sk 3310)
```

→ The natural key (customer_id) now matches two rows. Every fact table joins on customer_sk, so each order keeps the city it had when it happened.

02 Trap: the report that changed overnight

Three SCD mistakes and how each one shows up in a dashboard.

YOU DID	THE REPORT SAYS	BECAUSE
type 1 on city	Lyon revenue 2024: huge	Paris rows were overwritten
joined on customer_id	revenue doubled	two versions matched
valid_to = NULL	current filter drops rows	NULL compares to nothing

→ Interview phrasing: I default to type 2 with a surrogate key and an is_current flag, and I only use type 1 for corrections that should rewrite history.