The hierarchy that changes underneath you

Org units, levels and groups, and the property nobody warns you about — the tree is current, not historical, so a facility reassigned in March silently rewrites last year's district totals.

Pierre Robentz Cassion

Lesson 2 of 8

Routine Data and DHIS2 — How the database is shaped

What this lesson covers

- One tree, and everything hangs off it
- Groups are not levels
- The property nobody warns you about
- What to do about it
- Reassignment, reconstructed
- Aggregating up without double-counting
- Coverage is a district-level indicator
- What comes next

One tree, and everything hangs off it — Example

```
Level 1  Country
Level 2  Region
Level 3  District
Level 4  Facility        <- our 38 units
```

One tree, and everything hangs off it — In Python

```
import pandas as pd

vax = pd.read_csv("vaccination-coverage-2024.v1.csv", parse_dates=["period"])
print(f"{vax['facility_id'].nunique()} facilities in the extract")
```

One tree, and everything hangs off it — In R

```
library(dplyr)
n_distinct(vax$facility_id)
```

Groups are not levels — In Python

```
by_type = (
    vax.groupby("facility_type")["facility_id"].nunique().sort_values(ascending=False)
)
print(by_type)
```

Groups are not levels — In R

```
vax |> summarise(facilities = n_distinct(facility_id), .by = facility_type)
```

Groups are not levels

Facility type	Facilities
Health post	20
Health centre	16
District hospital	2

Groups are not levels

- **A unit belongs to exactly one parent and any number of groups.** Health post is a group; it does not tell you where...
- **Groups can overlap.** A facility can be in "health post", "hard to reach" and "supported by partner X" at once, and...
- **Group sets are the safe version.** A group set is a set of mutually exclusive groups — facility type is usually one —...
- Ask whether the thing you are grouping by is a group set — If it is not, check that the groups partition before you sum

The property nobody warns you about

- DHIS2 stores the current hierarchy, not a historical one — A data value is attached to a facility
- A district total that differs from the same query run six months ago, with no data entry in between.
- A published annual figure that cannot be reproduced.
- Two reports of the same period disagreeing, both correct on the day they were run.

What to do about it

- Pin the hierarchy with the extract — Pull the org unit list at the same time as the data and store it beside the values

What to do about it — In Python

```
org_units = pd.read_csv("outputs/extract/2026-07/org-units.csv")
    # ea/facility id, name, parent_id, level, extracted_on

pinned = vax.merge(org_units[["facility_id", "district_id"]],
                   on="facility_id", how="left", validate="many_to_one")
assert pinned["district_id"].notna().all(), "facility with no district in the pinned tree"
```

What to do about it — In R

```r
org_units <- readr::read_csv(here::here("outputs", "extract", "2026-07", "org-units.csv"))

pinned <- vax |>
  left_join(select(org_units, facility_id, district_id), by = "facility_id",
            relationship = "many-to-one")

stopifnot(!any(is.na(pinned$district_id)))
```

What to do about it

- Aggregate from the pinned tree, not from the live one — Then a figure published in March is reproducible in September,...
- Record the extraction date on both — The next lesson but one makes that a property of the extract script rather than a...

Reassignment, reconstructed — In Python

```
reassignments = pd.DataFrame([
    {"facility_id": "FAC017", "from_district": "D01", "to_district": "D02",
     "effective": "2024-04-01", "reason": "boundary revision"},
])

def district_at(facility, period, log=reassignments):
    moves = log[(log["facility_id"] == facility)
                & (pd.to_datetime(log["effective"]) <= period)]
    return moves["to_district"].iloc[-1] if len(moves) else None
```

Reassignment, reconstructed — In R

```
reassignments <- tibble::tribble(
  ~facility_id, ~from_district, ~to_district, ~effective,     ~reason,
  "FAC017",     "D01",          "D02",        as.Date("2024-04-01"), "boundary revision"
)
```

Reassignment, reconstructed

District totals are computed on the org unit hierarchy as at 2026-07-28. One facility was reassigned from D01 to D02 in April 2024; its 2024 data appears under D02 throughout, including for periods before the reassignment.

Reassignment, reconstructed

- That sentence is the deliverable — It costs nothing and it turns an irreproducible number into a reproducible one

Aggregating up without double-counting

- A facility appearing twice in the tree — Impossible in DHIS2 — a unit has one parent — but entirely possible in the CSV...

Aggregating up without double-counting — In Python

```
duplicated = org_units["facility_id"].duplicated(keep=False)
assert not duplicated.any(), org_units.loc[duplicated, ["facility_id", "district_id"]]
```

Aggregating up without double-counting — In R

```
stopifnot(!any(duplicated(org_units$facility_id)))
```

Aggregating up without double-counting

- Summing across levels — A district total plus its facilities is the district counted twice

Aggregating up without double-counting — In Python

```
print(org_units["level"].value_counts())
```

Aggregating up without double-counting — In R

```
org_units |> count(level)
```

Aggregating up without double-counting

- Every extract should be at exactly one level — If it is not, the first thing your script does is filter to one, and say...

Coverage is a district-level indicator — In Python

```
by_facility = (
    vax[vax["report_submitted"] == True]
    .query("antigen == 'penta3'")
    .groupby("facility_id")
    .agg(doses=("doses_administered", "sum"), target=("target_population", "sum"))
)
by_facility["coverage"] = by_facility["doses"] / by_facility["target"]
print(by_facility["coverage"].describe()[["min", "max"]].round(3))
```

Coverage is a district-level indicator — In R

```
vax |>
  filter(report_submitted, antigen == "penta3") |>
  summarise(coverage = sum(doses_administered) / sum(target_population),
            .by = facility_id) |>
  summarise(min = min(coverage), max = max(coverage))
```

What comes next

- You can now place a value in space.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/routine-data-and-dhis2/dhis2-02-org-unit-hierarchy