A WALK IN THE DATA DIVESTMENT CHALLENGE

Blueprint for M&A extracts
(data to be divested)

Data Ownership Across Time: Simple? No?

1 Here’s the knotty bit, put plainly.

A rules-based extract from a shared store isn’t just “today’s permissions.” Across the timeline, who’s entitled to see what can change—sometimes twice before lunch. Classic case: an employee kicks off a transaction while at Company A, then moves to Company B before the thing settles. For audit and carve-out purposes, that record may still *belong* to A (because that’s where it was initiated), even if B now employs the person who pressed “go.” Role-based access helps, but you need time-sliced entitlements and data structures—not verbal gymnastics.

Skills travel → Entitlements don’t

Data Entitlement follows the Sale

2 How to model it (without hand-waving)?

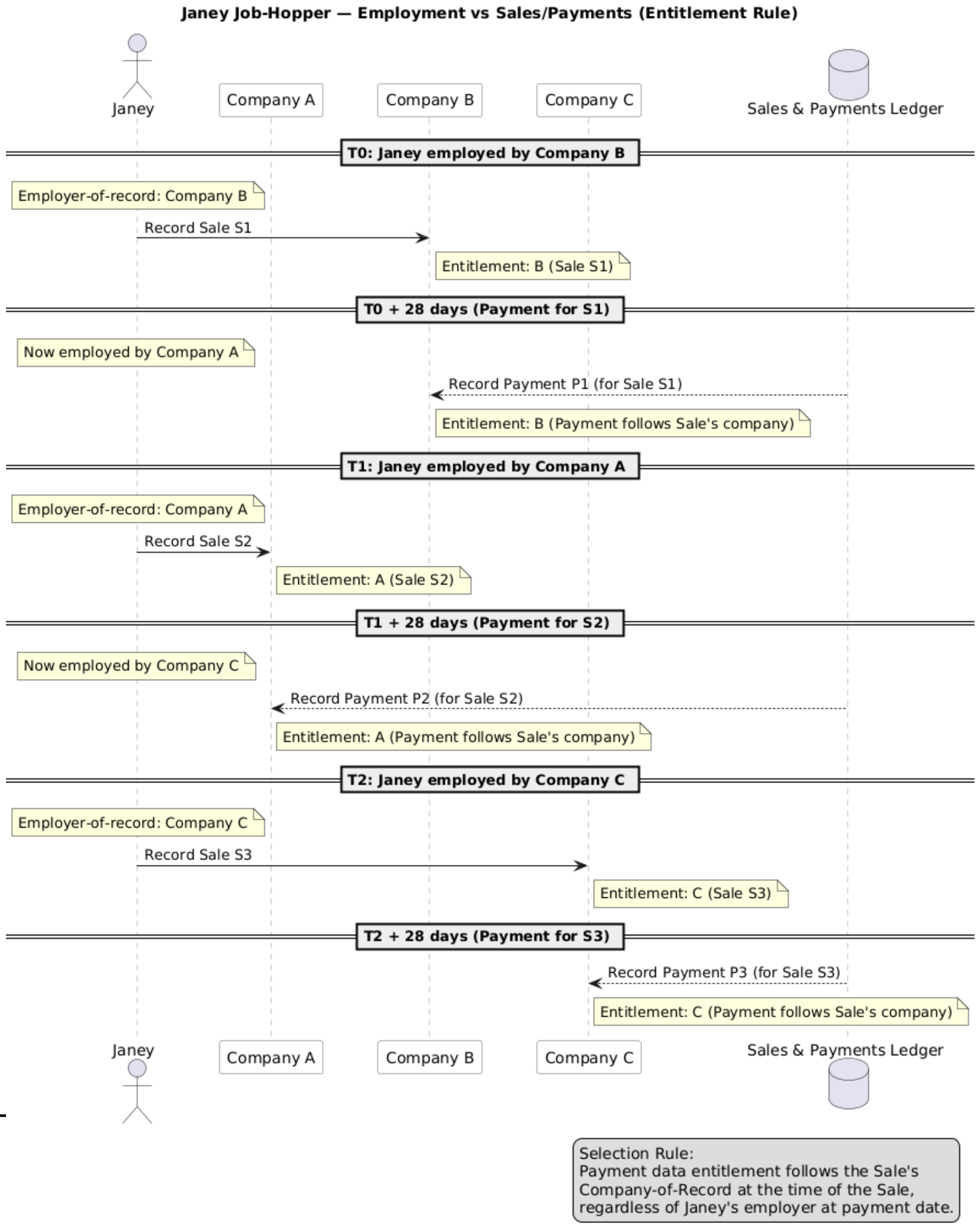
- **Effective-dated links**: employee ↔ company; transaction ↔ legal entity; processor ↔ controller.
- **Event stamps**: initiation_at, completion_at, plus **as_at** on any enquiry.
- **Bitemporal rules**: answer questions “who owned/was entitled *as-was* at T0?” and “who is entitled *as-is* today?” without rewriting history.
- **Default-deny** unless all four line up: role, purpose, time window, and policy clause (e.g., divestment perimeter).

Trainspotter’s notes This is exactly the sort of thinking AI/ML has to cope with—or, for the maths aficionados, why a humble calculator earns its keep once your linear equations wander past three variables. (Full disclosure: two variables was plenty spicy for me.)

Data Ownership Across Time: Systematic ⬄Easy to Define

3 Example: An employee Janey Job-Hopper works for a large corporate group comprised of Company A makes yellow widgets, Company B makes Blue Widgets, Company C makes Green widgets. Janey is an absolutely skilled widget sales person. When she makes a sale there is a sales transaction recorded followed by a payment transaction one month (28 days later). Janey is in so much demand that she frequently transfers between company's in the group. Big boss lady decided to split company B into a separate legal entity. Company B is entitled to the sales data made by Janey when she was employed by Company B. Company B is also entitled to the payment data for sales that Janey made when she was an employee in Company B. Express this in a diagram showing Janey's employee status changing and the transactions that relate to Janey's sales activity.

Janey's sales activity scenario is what is known as a Bi Temporal. Before you know it you will have achieved a shared business and technology stakeholder understanding. Techy diagrams to follow →



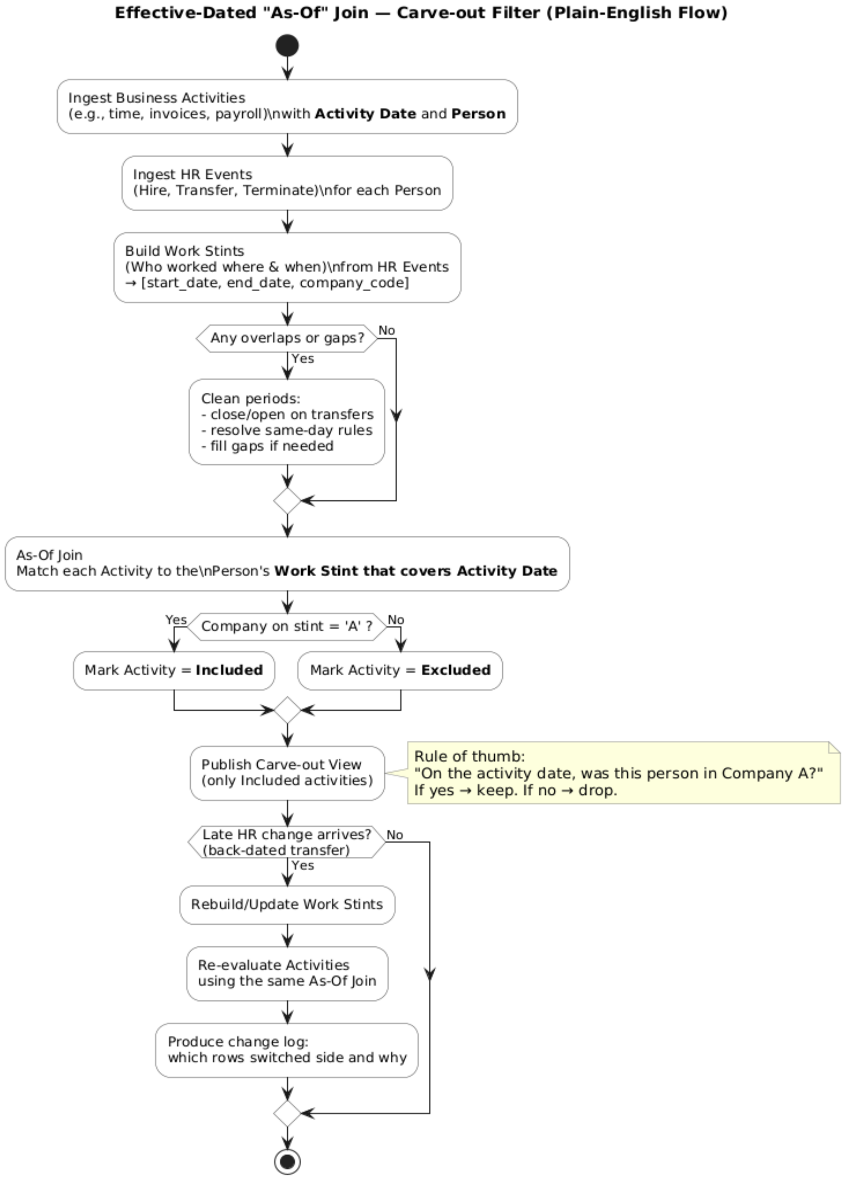
Janey Job-Hopper — Employment vs Sales/Payments (Entitlement Rule)

Janey
Company A
Company B
Company C
Sales & Payments Ledger
T0: Janey employed by Company B
Employer-of-record: Company B
Record Sale S1
Entitlement: B (Sale S1)
T0 + 28 days (Payment for S1)
Now employed by Company A
Record Payment P1 (for Sale S1)
Entitlement: B (Payment follows Sale's company)
T1: Janey employed by Company A
Employer-of-record: Company A
Record Sale S2
Entitlement: A (Sale S2)
T1 + 28 days (Payment for S2)
Now employed by Company C
Record Payment P2 (for Sale S2)
Entitlement: A (Payment follows Sale's company)
T2: Janey employed by Company C
Employer-of-record: Company C
Record Sale S3
Entitlement: C (Sale S3)
T2 + 28 days (Payment for S3)
Record Payment P3 (for Sale S3)
Entitlement: C (Payment follows Sale's company)
Janey
Company A
Company B
Company C
Sales & Payments Ledger

Selection Rule:
Payment data entitlement follows the Sale's Company-of-Record at the time of the Sale, regardless of Janey's employer at payment date.

Straight forward carve out algorithm

Carve out algorithm with two time-lines: e.g., event occurs →transfer data → event posted)



Effective-Dated "As-Of" Join — Carve-out Filter (Plain-English Flow)

Ingest Business Activities
(e.g., time, invoices, payroll)\nwith **Activity Date** and **Person**

Ingest HR Events
(Hire, Transfer, Terminate)\nfor each Person

Build Work Stints
(Who worked where & when)\nfrom HR Events
→ [start_date, end_date, company_code]

Any overlaps or gaps? — Yes / No

Clean periods:
- close/open on transfers
- resolve same-day rules
- fill gaps if needed

As-Of Join
Match each Activity to the\nPerson's **Work Stint that covers Activity Date**

Company on stint = 'A' ? — Yes / No

Mark Activity = **Included**

Mark Activity = **Excluded**

Publish Carve-out View
(only Included activities)

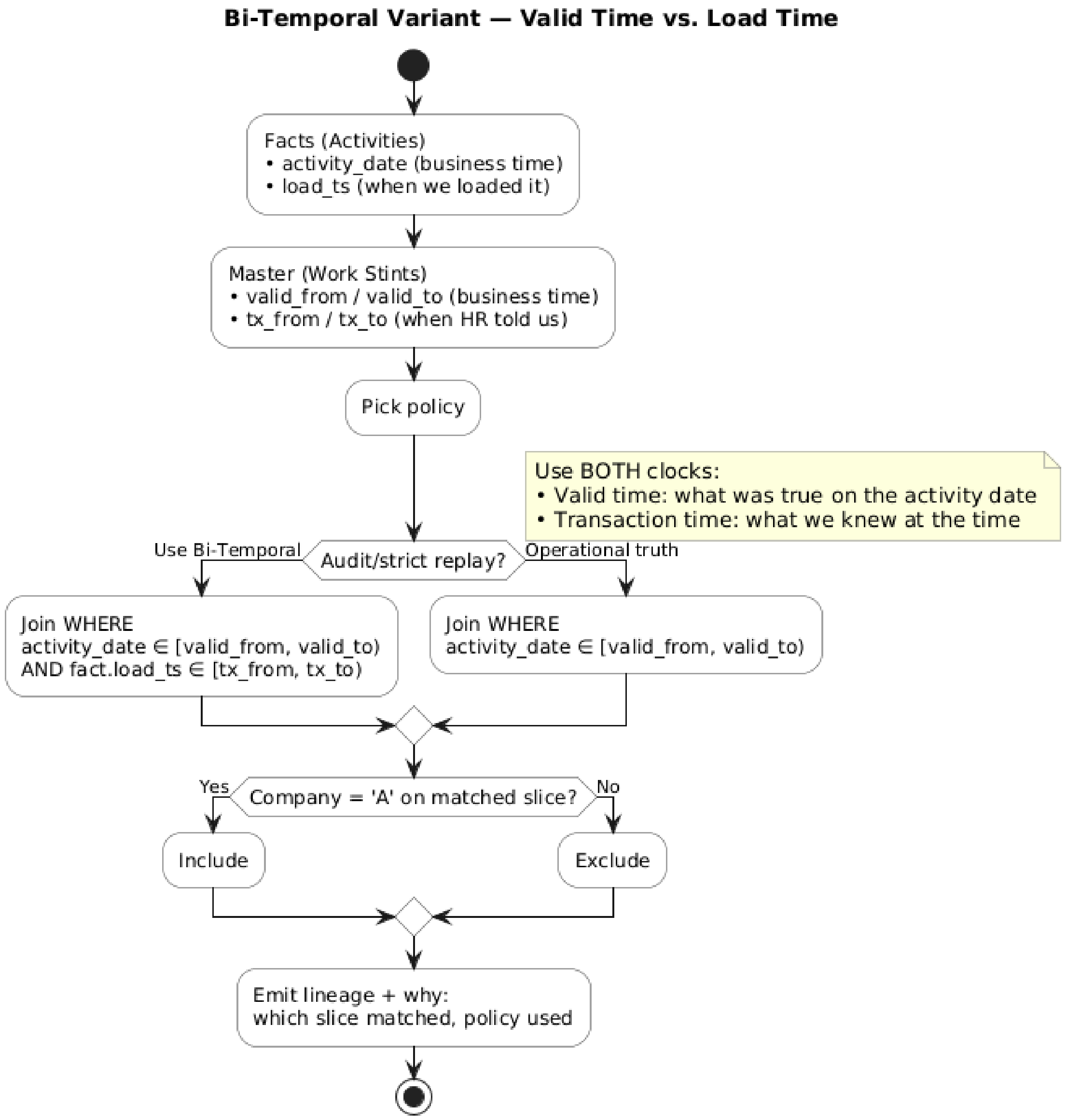
Rule of thumb:
"On the activity date, was this person in Company A?"
If yes → keep. If no → drop.

Late HR change arrives?
(back-dated transfer) — Yes / No

Rebuild/Update Work Stints

Re-evaluate Activities
using the same As-Of Join

Produce change log:
which rows switched side and why



Bi-Temporal Variant — Valid Time vs. Load Time

Facts (Activities)
- activity_date (business time)
- load_ts (when we loaded it)

Master (Work Stints)
- valid_from / valid_to (business time)
- tx_from / tx_to (when HR told us)

Pick policy

Use BOTH clocks:
- Valid time: what was true on the activity date
- Transaction time: what we knew at the time

Audit/strict replay? — Use Bi-Temporal / Operational truth

Join WHERE
activity_date ∈ [valid_from, valid_to)
AND fact.load_ts ∈ [tx_from, tx_to)

Join WHERE
activity_date ∈ [valid_from, valid_to)

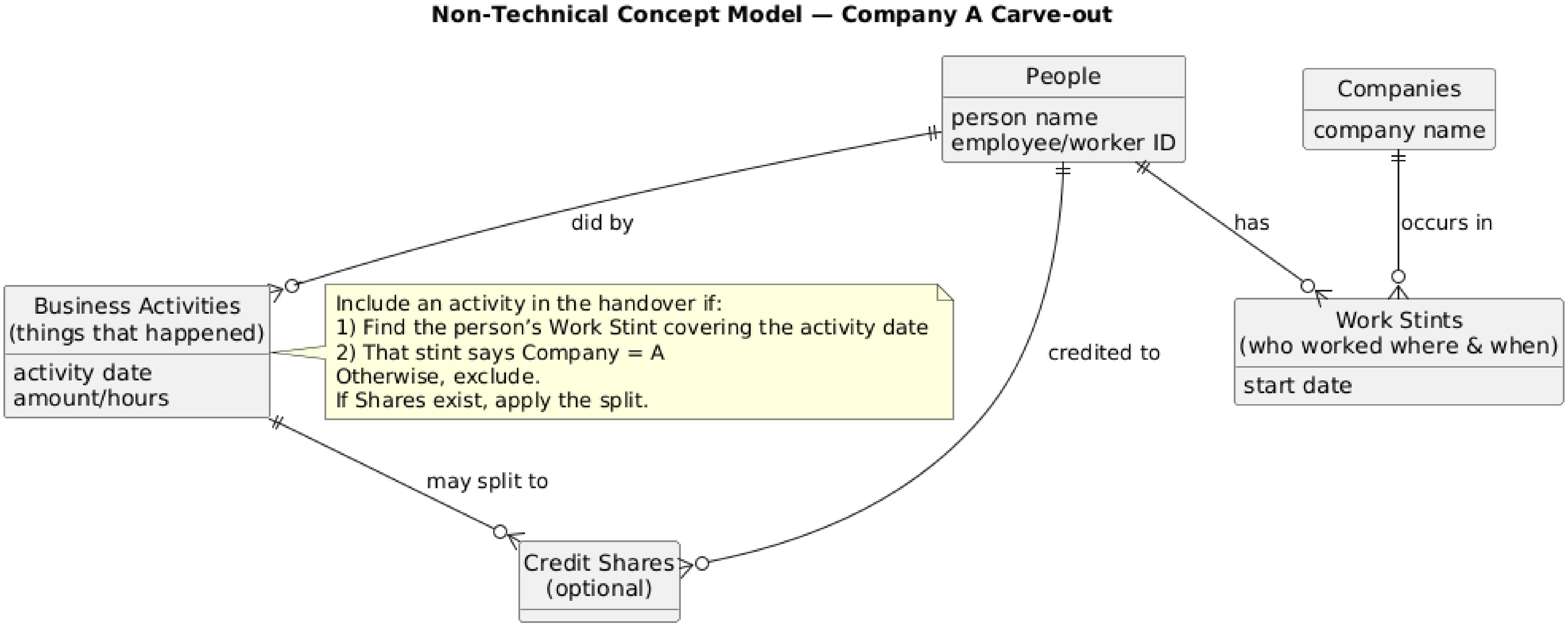
Company = 'A' on matched slice? — Yes / No

Include

Exclude

Emit lineage + why:
which slice matched, policy used

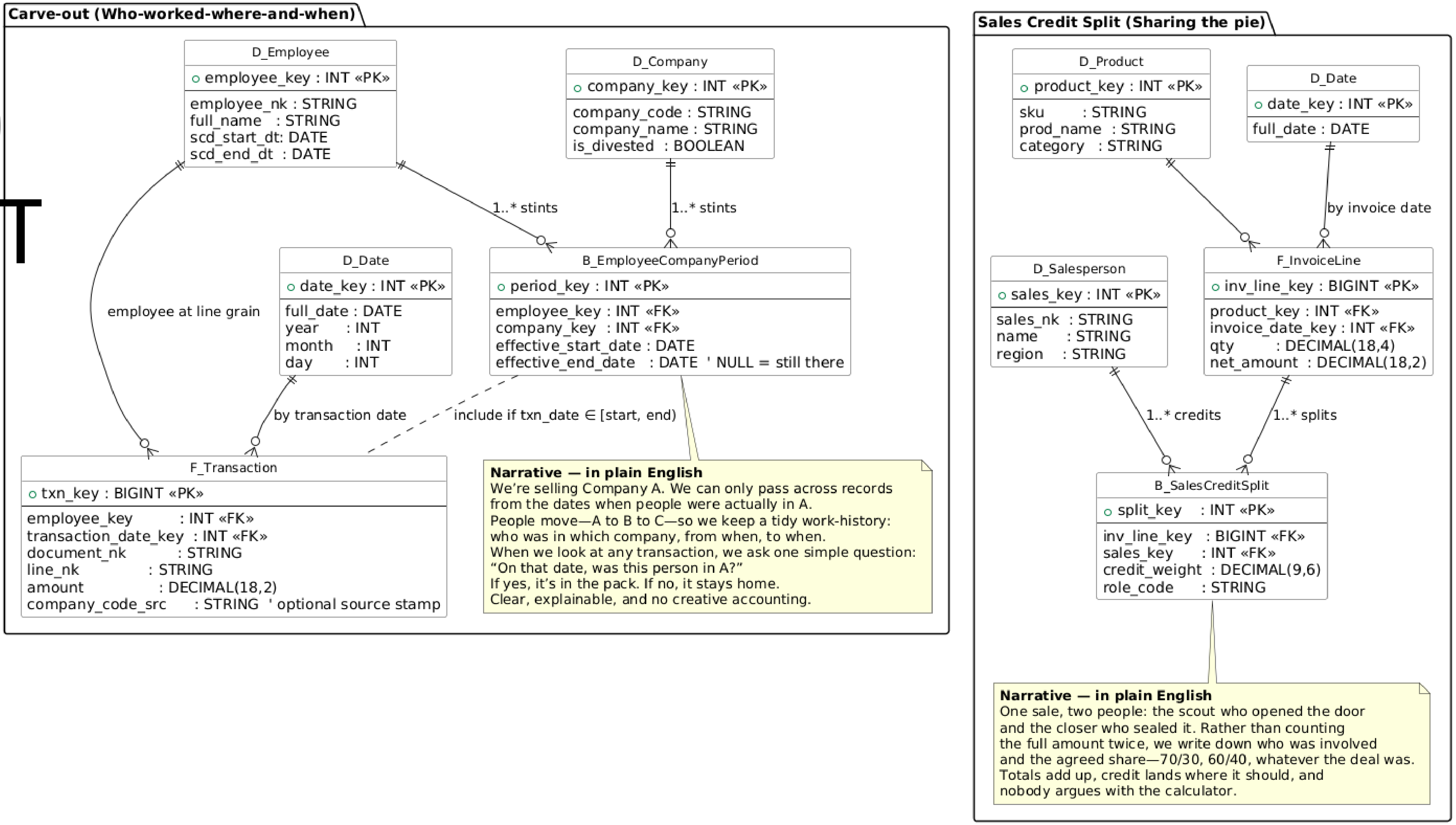
BUSINESS REQ TO SEMI TECH MODEL



Include an activity in the handover if:
1) Find the person’s Work Stint covering the activity date
2) That stint says Company = A
Otherwise, exclude.
If Shares exist, apply the split.

REAL
WORLD
DIALECT

Star Schema with Bridge Tables — Plain-English Narratives



Carve-out (Who-worked-where-and-when)

D_Employee
○ employee_key : INT «PK»
employee_nk : STRING
full_name : STRING
scd_start_dt: DATE
scd_end_dt : DATE

D_Company
○ company_key : INT «PK»
company_code : STRING
company_name : STRING
is_divested : BOOLEAN

D_Date
○ date_key : INT «PK»
full_date : DATE
year : INT
month : INT
day : INT

B_EmployeeCompanyPeriod
○ period_key : INT «PK»
employee_key : INT «FK»
company_key : INT «FK»
effective_start_date : DATE
effective_end_date : DATE ' NULL = still there

F_Transaction
○ txn_key : BIGINT «PK»
employee_key : INT «FK»
transaction_date_key : INT «FK»
document_nk : STRING
line_nk : STRING
amount : DECIMAL(18,2)
company_code_src : STRING ' optional source stamp

Narrative — in plain English
We're selling Company A. We can only pass across records
from the dates when people were actually in A.
People move—A to B to C—so we keep a tidy work-history:
who was in which company, from when, to when.
When we look at any transaction, we ask one simple question:
"On that date, was this person in A?"
If yes, it's in the pack. If no, it stays home.
Clear, explainable, and no creative accounting.

Sales Credit Split (Sharing the pie)

D_Product
○ product_key : INT «PK»
sku : STRING
prod_name : STRING
category : STRING

D_Date
○ date_key : INT «PK»
full_date : DATE

D_Salesperson
○ sales_key : INT «PK»
sales_nk : STRING
name : STRING
region : STRING

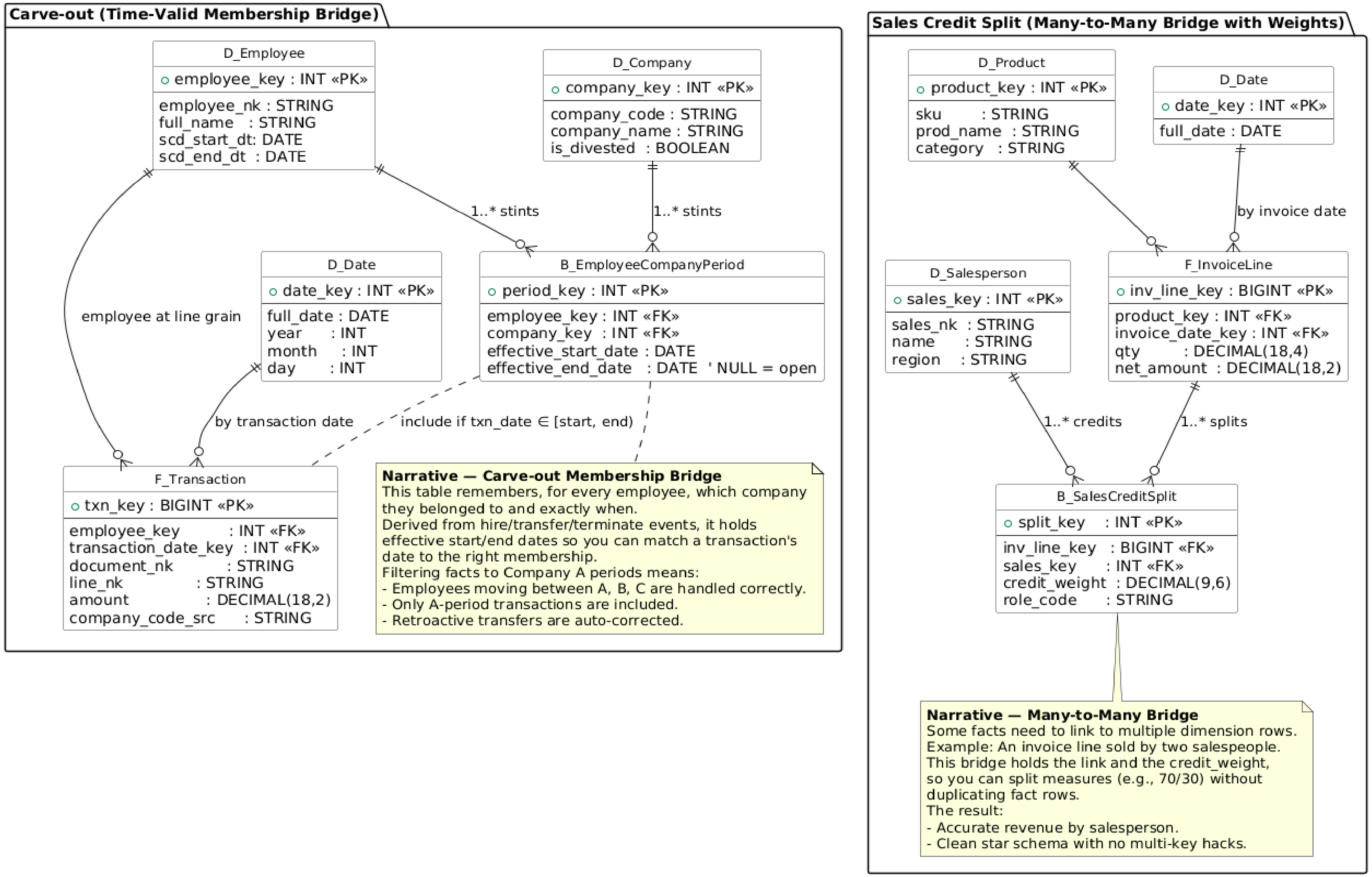
F_InvoiceLine
○ inv_line_key : BIGINT «PK»
product_key : INT «FK»
invoice_date_key : INT «FK»
qty : DECIMAL(18,4)
net_amount : DECIMAL(18,2)

B_SalesCreditSplit
○ split_key : INT «PK»
inv_line_key : BIGINT «FK»
sales_key : INT «FK»
credit_weight : DECIMAL(9,6)
role_code : STRING

Narrative — in plain English
One sale, two people: the scout who opened the door
and the closer who sealed it. Rather than counting
the full amount twice, we write down who was involved
and the agreed share—70/30, 60/40, whatever the deal was.
Totals add up, credit lands where it should, and
nobody argues with the calculator.

Bridge tables
- Who-worked-where-and-when: filters transactions to the right company by date.
- Sharing the pie: splits one sale across several people fairly and visibly.

Star Schema with Bridge Tables — Annotated



Carve-out (Time-Valid Membership Bridge)

```
D_Employee
○ employee_key : INT «PK»
employee_nk : STRING
full_name   : STRING
scd_start_dt: DATE
scd_end_dt  : DATE
```

```
D_Company
○ company_key : INT «PK»
company_code : STRING
company_name : STRING
is_divested  : BOOLEAN
```

```
D_Date
○ date_key : INT «PK»
full_date : DATE
year      : INT
month     : INT
day       : INT
```

```
B_EmployeeCompanyPeriod
○ period_key : INT «PK»
employee_key : INT «FK»
company_key  : INT «FK»
effective_start_date : DATE
effective_end_date   : DATE  ' NULL = open
```

```
F_Transaction
○ txn_key : BIGINT «PK»
employee_key          : INT «FK»
transaction_date_key  : INT «FK»
document_nk           : STRING
line_nk               : STRING
amount                : DECIMAL(18,2)
company_code_src      : STRING
```

Narrative — Carve-out Membership Bridge
This table remembers, for every employee, which company they belonged to and exactly when.
Derived from hire/transfer/terminate events, it holds effective start/end dates so you can match a transaction's date to the right membership.
Filtering facts to Company A periods means:
- Employees moving between A, B, C are handled correctly.
- Only A-period transactions are included.
- Retroactive transfers are auto-corrected.

Sales Credit Split (Many-to-Many Bridge with Weights)

```
D_Product
○ product_key : INT «PK»
sku         : STRING
prod_name   : STRING
category    : STRING
```

```
D_Date
○ date_key : INT «PK»
full_date : DATE
```

```
D_Salesperson
○ sales_key : INT «PK»
sales_nk   : STRING
name       : STRING
region     : STRING
```

```
F_InvoiceLine
○ inv_line_key : BIGINT «PK»
product_key : INT «FK»
invoice_date_key : INT «FK»
qty          : DECIMAL(18,4)
net_amount   : DECIMAL(18,2)
```

```
B_SalesCreditSplit
○ split_key    : INT «PK»
inv_line_key   : BIGINT «FK»
sales_key      : INT «FK»
credit_weight  : DECIMAL(9,6)
role_code      : STRING
```

Narrative — Many-to-Many Bridge
Some facts need to link to multiple dimension rows.
Example: An invoice line sold by two salespeople.
This bridge holds the link and the credit_weight, so you can split measures (e.g., 70/30) without duplicating fact rows.
The result:
- Accurate revenue by salesperson.
- Clean star schema with no multi-key hacks.

Bridge tables
- B_EmployeeCompanyPeriod: Resolves time-valid employee→company membership for carve-outs.
- B_SalesCreditSplit: Resolves fact→many salespeople with weighted allocation.

COMMON PATTERNS (A BIT TECHIE, BUT THEY WORK)

Pattern Name	Where it's used	How it relates to your scenario
Effective-Dated Join	HR/payroll systems, dimensional modelling	Join facts to master data based on a date range so the fact is linked to the version of the master data that was valid at the time of the event.
As-Of Join	SQL/DW, slowly changing dimensions	Filter/join to the correct dimension row "as of" the event date, regardless of current state.
Valid-Time Join	Temporal databases, bi-temporal modelling	A join that respects the valid_from / valid_to of both fact and master data.
Point-in-Time Filter	Auditing, data lakehouse design	Retrieve only records whose master data attributes were true at a specific historical point.
Effective Dating with Retroactive Adjustment	ETL/ELT change-data-capture pipelines	Handles late-arriving changes that can flip a master data record from "in" to "out" for past events.
Bi-Temporal Filtering	Financial & regulatory data	Applies both valid time (business truth) and transaction time (when the record was loaded/changed) to ensure correct historical state.