Data Warehousing : *What is it?* & related Stanford DB research

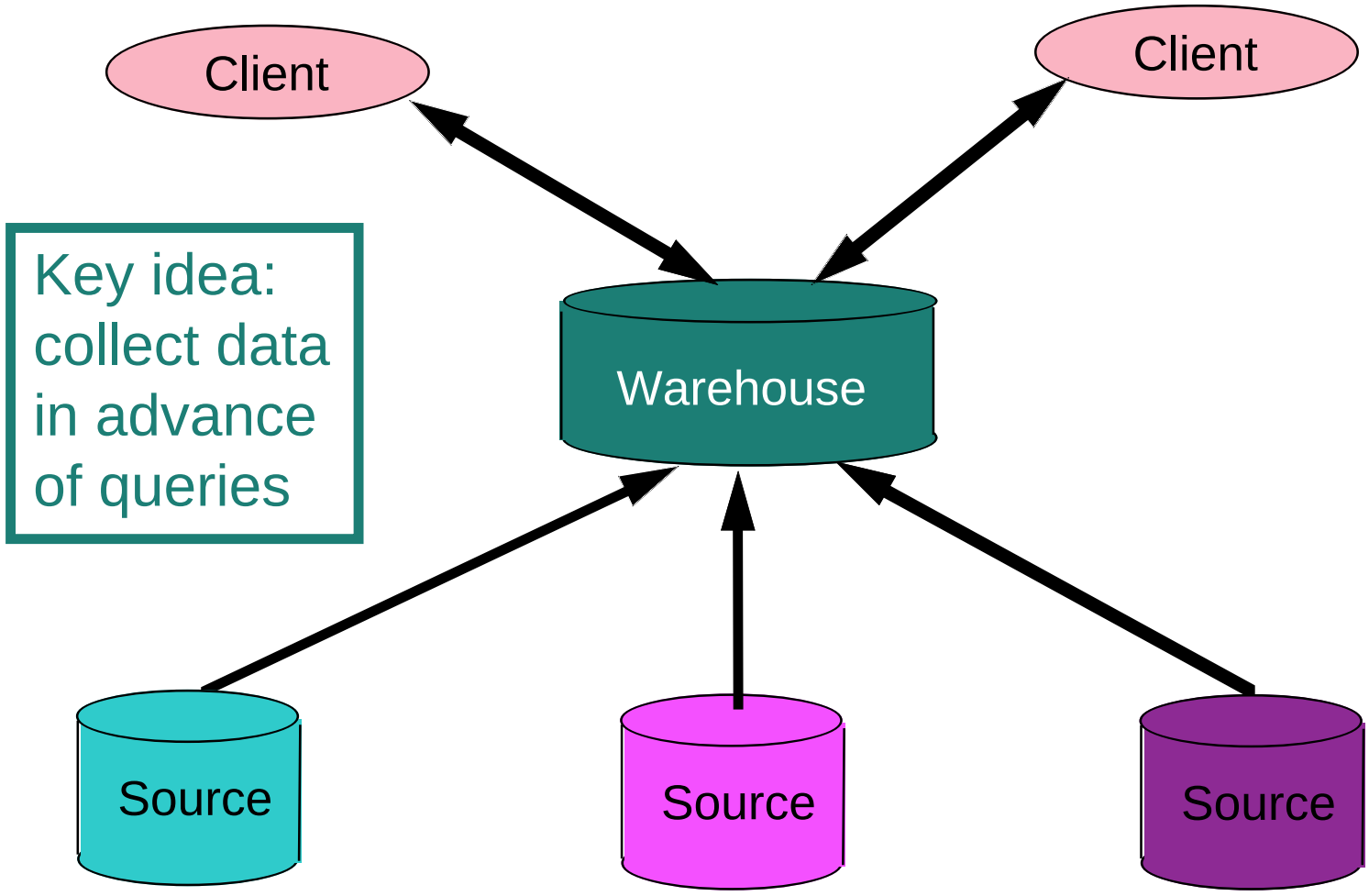
Janet L. Wiener

Stanford University

Outline

- What is a data warehouse?
- Key warehouse issues
  - Warehouse design
  - Querying and analysis
    - What to do with data once it's there
  - Creation and maintenance
    - Getting data into warehouse
- Research & current state of industry

Warehouse idea



What is a warehouse?

- Stored collection of diverse data
  - Solution to data integration problem
  - Single repository of information
- Subject-oriented
  - Organized by subject, not by application
  - Used for analysis, data mining
- Optimized differently from transaction db
- User interface aimed at executive

What is a warehouse (2)?

- Large volume (Gb, Tb) of data
- Non-volatile
  - Contents are stable for long periods of time
  - Enables long analysis transactions
  - Often updates are append-only
- Time variant kept
  - History: set of snapshots
  - Time attributes important

Warehousing & industry

- Warehousing is big business
  - $3.5 billion in early 1997
  - $8 billion in 1998 [Metagroup survey]
- Wal-mart is largest warehouse
  - 4 Tb
  - 200-300 Mb per day
- Lots of new startups with tools

Advantages of Warehousing

- High query performance
- Accessible anytime
  - Even if sources not available
- Local processing at sources unaffected
- Extra information at warehouse
  - Summarize (store aggregates)
  - Add historical information
- Queries not visible outside warehouse

Disadvantages? of warehousing

- Decide what to store in advance
  - But still run ad-hoc queries
- Can only query data stored at warehouse
  - Data gets stale
  - But stays consistent while creating report
- Must detect source changes and update warehouse

Motivating examples

- Grocery store chain
  - Cashier sales
  - Inventory invoices
  - Promotions history
- Insurance company
  - Policy info
  - Claims processing

Warehouse is specialized DB

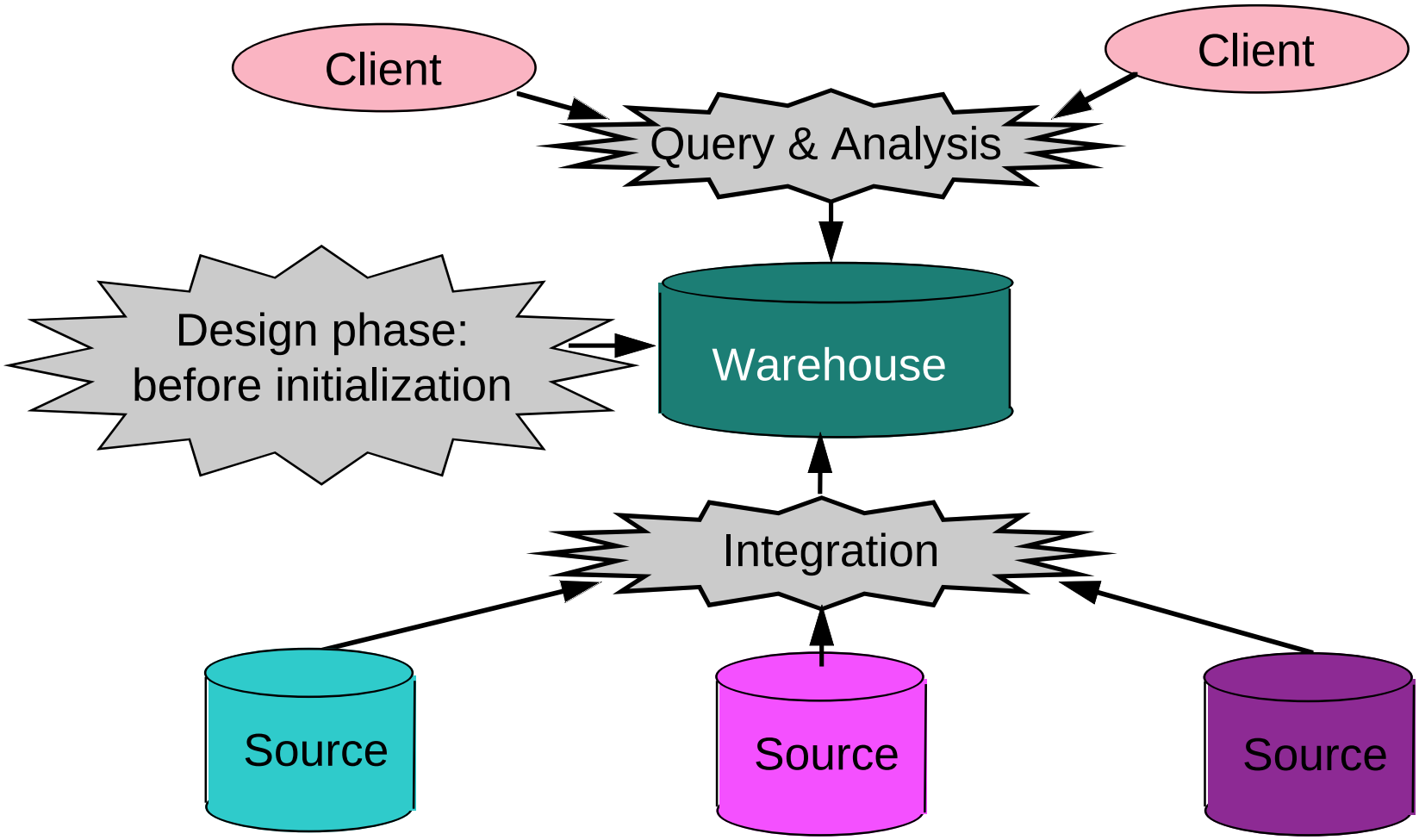
Standard DB

- Mostly updates
- Many small transactions
- Mb-Tb of data
- Current snapshot
- Raw data
- Clerical users

Warehouse

- Mostly reads
- Queries are long, complex
- Gb-Tb of data
- History
- Summarized, consolidated data
- Decision-makers, analysts as users

Warehouse architecture



Designing a warehouse

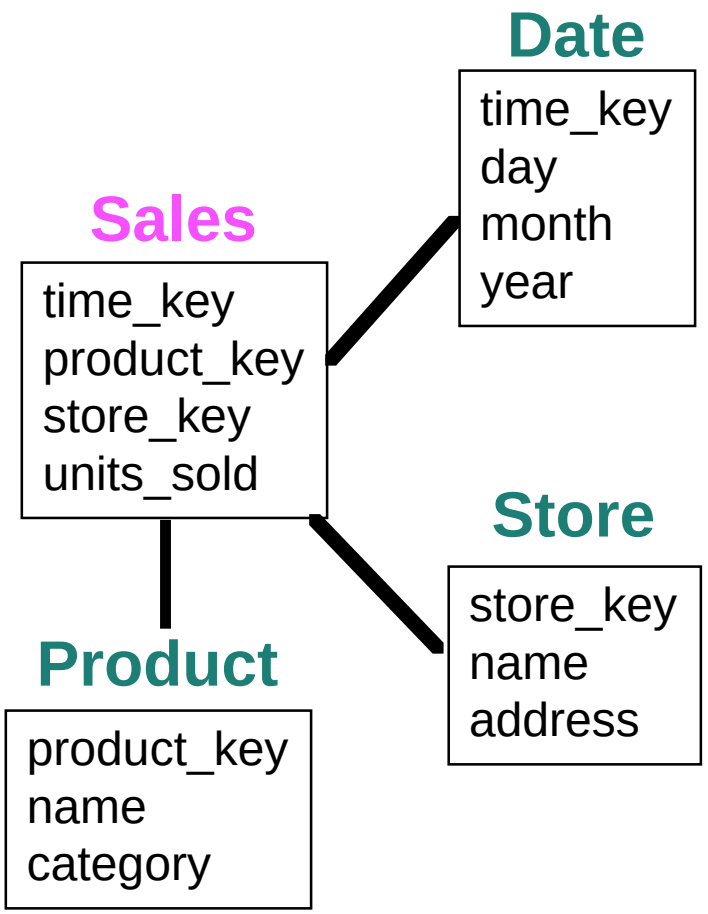
- Design options
  - Logical layout of data: ROLAP vs MDDB
- Design steps
  - Identify warehouse data
  - Identify source data needed
  - Choose hardware and software

Warehouse design: ROLAP

- ROLAP = Relational OnLine Analytical Processing
- Relational DB
  - Special indices: bitmap, multi-table join
  - Special tuning: maximize query throughput
  - Special schema design: star, snowflake
- Products
  - Oracle, Sybase IQ, RedBrick, DB2

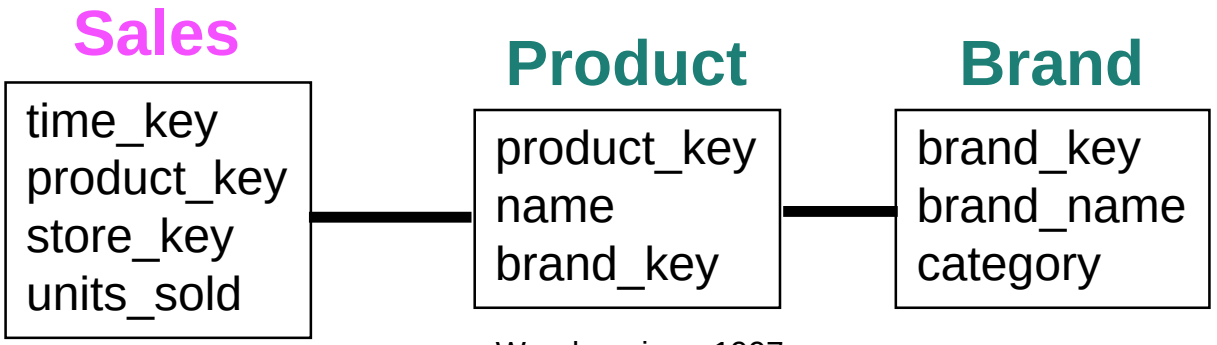
Star schema

- Central fact table
  - Numerical measurements, usually additive
  - E.g., number sold
- Many dimension tables
  - Textual descriptions
  - E.g., product specification
  - Each dimension joins with fact table



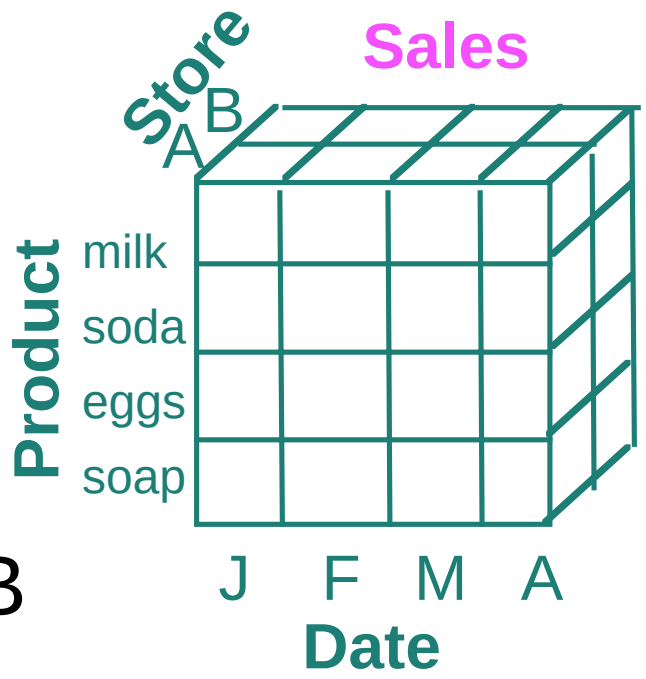
Star schema (2)

- May have several “stars”
- Dimension tables not normalized
- Use time for seasons, day of week, etc
- Snowflake schema normalizes dimensions
  - E.g., Separate common brand info from product



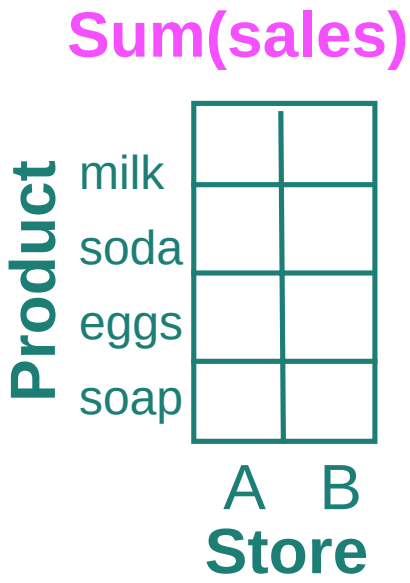
Warehouse design: MDDB

- MultiDimensional DataBase =MOLAP
- Dimensions used to index array
- "Facts" stored in array cells
- Often on top of relational DB
- Products: Pilot, Essbase, Gentia



Datacubes and aggregation

- Summary of data
  - All possible groupings for aggregation operator
    - Sum(sales) by product by city
  - Materialized or virtual (or combination)
  - Roll-up: remove dimension
    - Roll-up by product for all dates
  - Drill-down: add dimension
    - Drill-down by year



Data Mart

- Slice of data in warehouse
  - Usually by locus of control
  - E.g., only data for store A
- Summary along some dimension
  - Could be datacube minus some dimensions

Identify warehouse data

- Choose central facts, dimensions
  - Identify attributes & their granularity
  - Find source for each attribute
- Choose auxiliary data to store
  - Aggregations to precompute
  - Indices needed for queries
  - Additional data to help with maintenance
- How to choose auxiliary data is still research

Choosing aggregates

- Which materialized aggregates will minimize response time?
  - Given set of queries, storage constraint
  - Which points in aggregate datacube to materialize
- Greedy algorithm [HRU 1996]
  - Loop: Choose “best” aggregate
  - Until run out of storage
- Similar alg to choose indices [GHRU 1997]

What about maintenance?

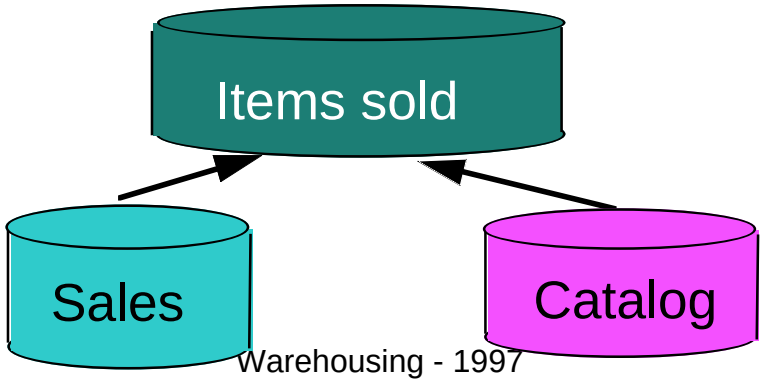
- Storage is cheap
- Updating warehouse is not
  - Usually takes warehouse off-line
  - Time-consuming to propagate updates to aggregates
- Algorithms should balance update cost against query reponse time

Choosing tables, indices (2)

- Which additional tables & indices will minimize update cost
  - Given set of tables that satisfy all queries
- A* search + heuristics [LQA 1997]
  - Prune set of choices during exhaustive search
  - Add table that is smaller than its base tables
  - Add table that won't be updated
- Ignores aggregates

Make warehouse self-maintainable

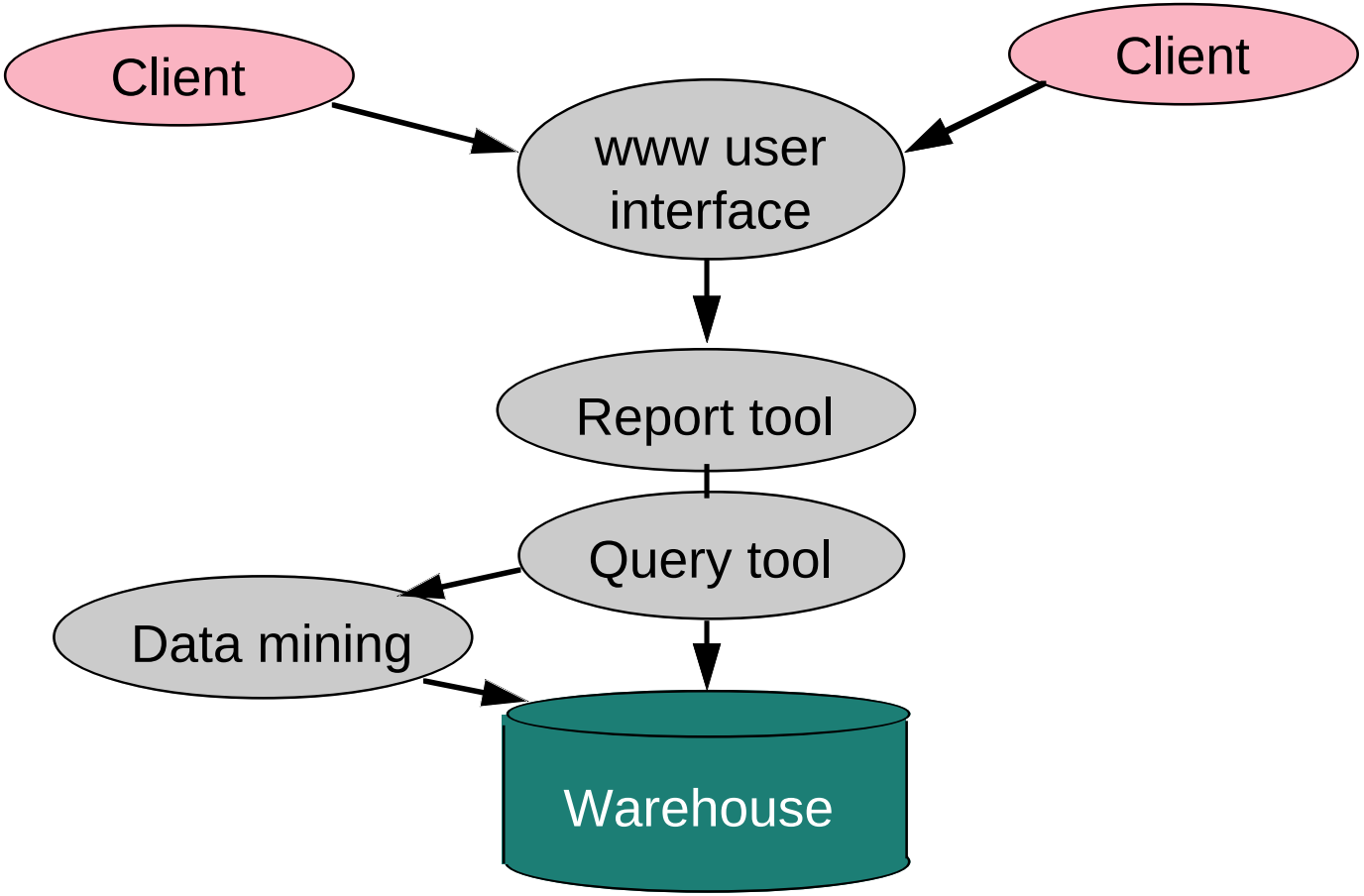
- Add auxiliary tables to minimize update cost
- Original + auxiliary are self-maintainable
  - E.g., auxiliary table of all unsold catalog items
- Some updates may still be self-maintainable
  - E.g., insert into catalog if item (the join attribute) is a key



Detection of self-maintainability

- Most algorithms are at table level
- Most algorithms are compile-time
- Tuple level at runtime [Huyn 1996, 1997]
  - Use state of tables and update to determine if self-maintainable
  - E.g., check whether sale is for item previously sold

Warehouse use



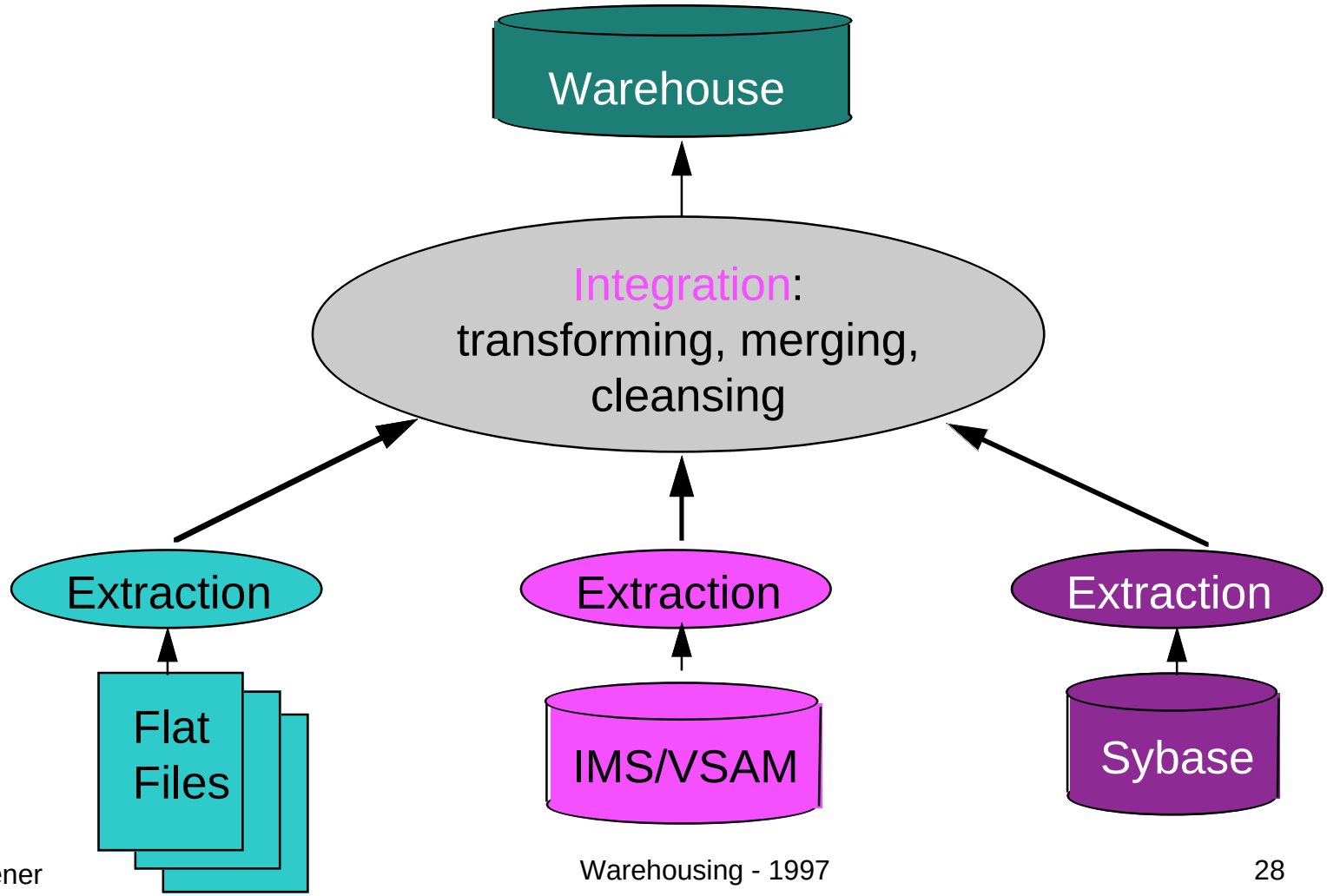
Warehouse access

- Querying the warehouse
- Comparing data
  - Data mining
- Presenting the data
- Sending reports to other users

Special query functions

- Rank
- Moving average or sum
- Cumulative total
- Median
- Time window

Warehouse loading



What is loading?

- Extraction
- Transformation
- Merging
- Cleansing
- Computation of additional data
  - Aggregates, indices

Warehouse maintenance

- Warehouse is materialized view over sources
- How often to propagate new data
  - At night, 1x a week/month, continuously
- Off-line or on-line
  - Current products take warehouse off-line
- Recomputation vs incremental maintenance
  - Reload entire warehouse or
  - Only propagate changes to warehouse

Incremental maintenance

- Must detect changes at sources
- Propagate changes into consistent view
- Same steps for changes: transforming, merging, cleansing
- Avoid going to sources if possible
- Recomputation of indices, aggregates
  - Total vs partial
- Focus of Stanford's Whips project

Data extraction

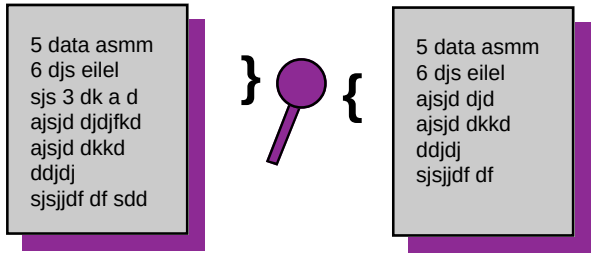
- Source types
  - Relational, flat file, IMS, VSAM, IDMS, www
- How to get source data out?
  - Dump file
  - Create report
  - Send ODBC
  - Extract tool (3rd party)

Change detection

- Detect & send changes to integrator
- Different classes of sources
  - Cooperative
  - Queryable
  - Logged
  - Snapshot/dump

Snapshot change detection

- Compare old & new snapshots
- Join-based algorithms
  - Hash old data, probe with new
- Window algorithm
  - Sliding window over snapshots
  - Good for local changes



Data transformation

- Convert data to uniform format
  - Byte ordering, string termination
  - Internal layout
- Remove, add, & reorder attributes
  - Add (regeneratable) key
  - Add date to get history
- Sort tuples

Data integration

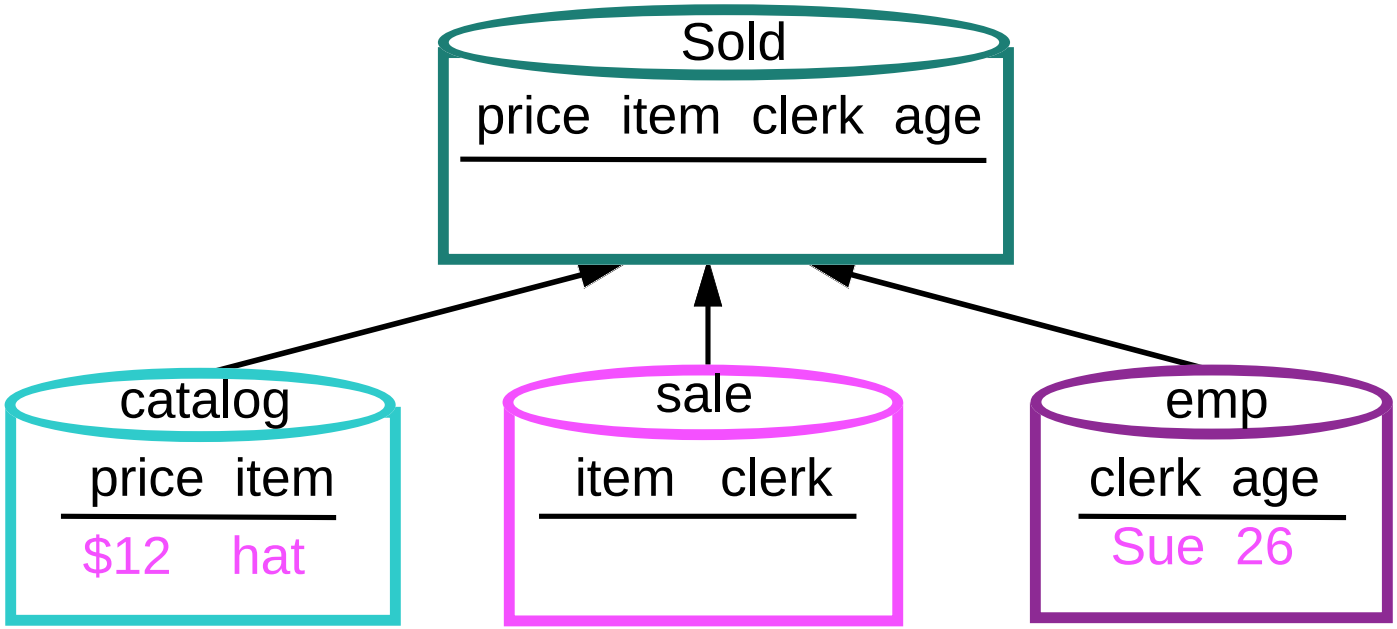
- Rules for matching data from different sources
- Build composite view of data
- Eliminate duplicate, unneeded attributes

Integrated data consistency

- Conventional maintenance inadequate
  - Sources report changes but:
  - No locking, no global transactions (sources don’t communicate, coordinate with each other)
- Inconsistencies caused by interleaving of updates

Example anomaly

- table *Sold = catalog x sale x emp*
- *insert into sale* [hat, Sue]
- *delete from catalog* [$12, hat]

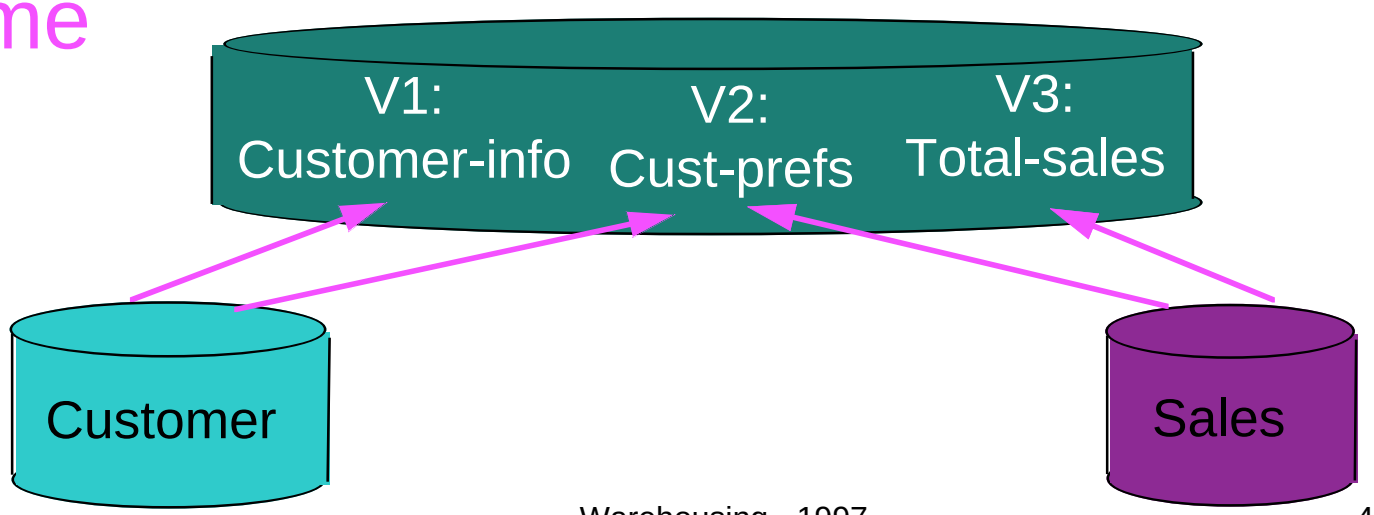


Strobe algorithm ideas

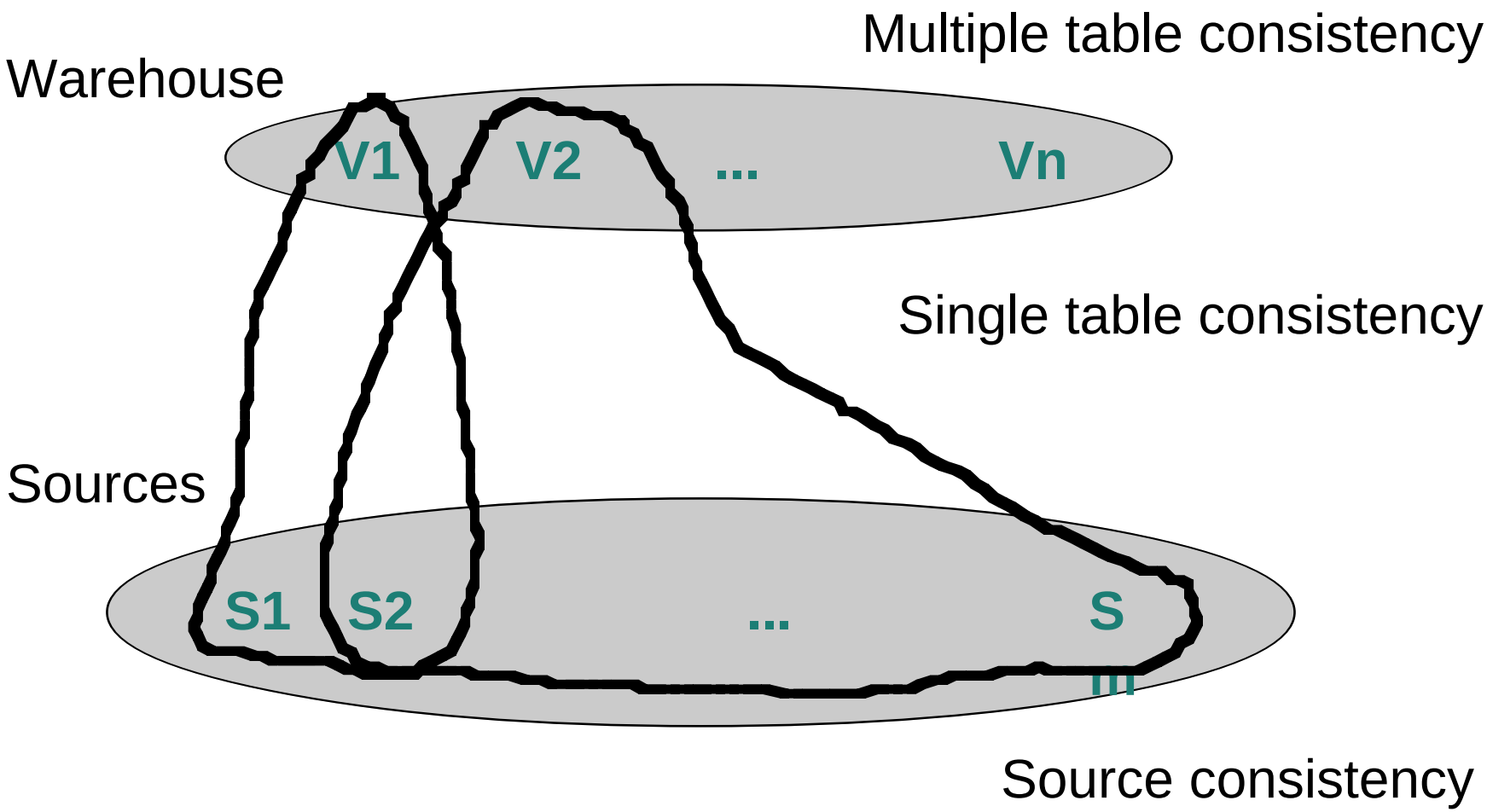
- Apply actions only after a set of interleaving updates are all processed
  - Wait for sources to quiesce
- Compensate effects of interleaved updates
  - Subtract effects of later updates before installing changes
- Can combine these ideas

Multiple table consistency

- More than 1 table at warehouse
- Multiple tables share source data
- Updates at source should be reflected in all warehouse tables at the same time



Multiple table consistency



Painting algorithm

- Use merge process (MP) to coordinate sending updates to warehouse
- MP holds update actions for each table
- MP charts potential table states arising from each set of update actions
- MP sends batch of update actions together when tables will be consistent

Data cleansing

- Find (& remove) duplicate tuples
  - E.g., Jane Doe & Jane Q. Doe
- Detect inconsistent, wrong data
  - Attributes that don't match
  - E.g., city, state and zipcode
- Patch missing, unreadable data
- Want to "backflush" clean data
  - Notify sources of errors found

Aggregate functions

- "Group" many different source tuples
  - *E.g., find average sales for each region*
- Different maintenance algorithms
  - Insert of source tuple can cause increment or decrement of aggregate value
- Detect when batch of updates does (not!) affect aggregate
  - *E.g., new sales in California do not affect East Coast total*

Aggregates (2)

- Can compute change to one aggregate based on another
  - *E.g., use total sales for each store to get total sales for region*
- Watch out for round-off errors

Current state of industry

- Extraction and integration done off-line
  - Usually in large, time-consuming, batches
  - Assume a “night” or “weekend”
- Everything copied at warehouse
  - Not selective about what is stored
  - Query benefit vs storage & update cost
- Often recompute rather than detect changes
  - Change detection is hard for legacy sources

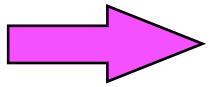
Expiring data from warehouse

- How to “remove” data from warehouse
  - When data is old
  - When data is no longer relevant
  - When storage space is no longer available

Options: deletion vs expiration

Fulltable:

company	branch	sales
ibm	east	100
ibm	west	200
ge	north	50
ge	south	350
ge	europe	200



Summary table:

company	sum-sales
ibm	300
ge	600

Options: deletion vs expiration

Fulltable:

company	branch	sales
ibm	east	100
ibm	west	200
ge	north	50
ge	south	350
ge	europe	200

Delete

Summary table:

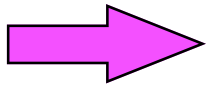
company	sum-sales
ibm	300
ge	~~600~~ 400

Options: deletion vs expiration

Fulltable:

company	branch	sales
ibm	east	100
ibm	west	200
ge	north	50
ge	south	350
ge	europe	200

Expire

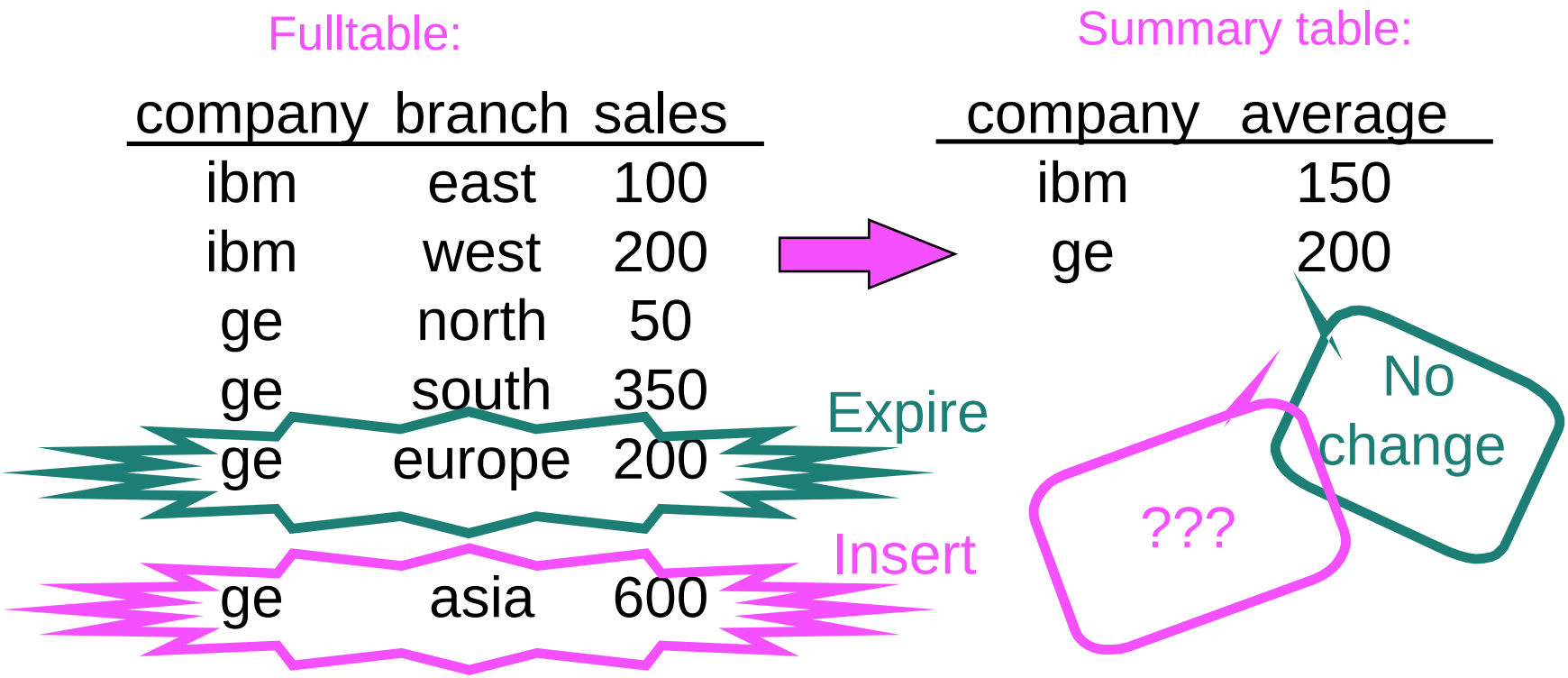


Summary table:

company	sum-sales
ibm	300
ge	600

No change

Auxiliary data



Auxiliary data

Fulltable:

company	branch	sales
ibm	east	100
ibm	west	200
ge	north	50
ge	south	350
ge	europe	200
ge	asia	600

Expire

Insert

Summary table:

company	average
ibm	150
ge	~~200~~ 300

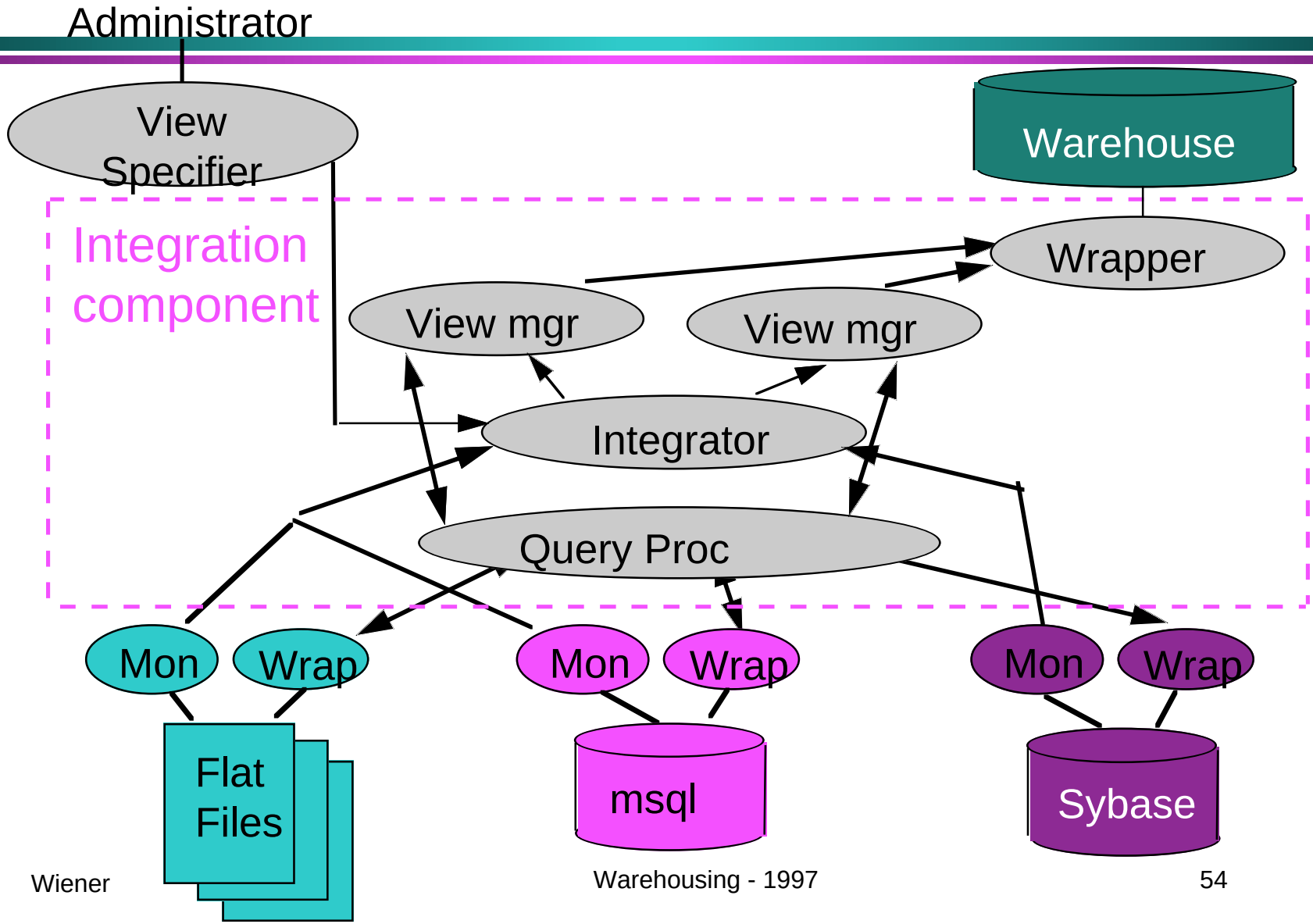
company	count
ibm	2
ge	~~3~~ 4

Now : $\frac{\text{old avg} * \text{old count} + \text{new sales}}{\text{new count}}$ = new avg

Managing tables and expirations

- Design system with variety of user choices
- On expiration of data:
  - Freeze dependent table?
  - Create auxiliary data?
  - Expire necessary data from dependent tables?
  - Archive expired data?
- Goal: flexible incremental maintenance
  - Current warehouses delete & replace whole tables
  - We want to incrementally add & expire tuples!

Whips prototype



New Whips prototype