Data Warehousing and Data Mining

De-Normalization Techniques

Splitting Tables

Table

ColA	ColB	ColC

Table_v1

ColA	ColB

Table_v2

ColA	ColC

Vertical Split

Table_h1

ColA	ColB	ColC

Table_h2

ColA	ColB	ColC

Horizontal split

Splitting Tables: Horizontal splitting...

Breaks a table into multiple tables based upon common column values. Example: Campus specific queries.

GOAL

- **Spreading rows for exploiting parallelism.**
- **Grouping data to avoid unnecessary query load in WHERE clause.**

Splitting Tables: Horizontal splitting

ADVANTAGE

- **Enhance security of data.**
- **Organizing tables differently for different queries.**
- **Graceful degradation of database in case of table damage.**

Splitting Tables: Vertical Splitting

- Infrequently accessed columns become extra "baggage" thus degrading performance.
- Very useful for rarely accessed large text columns with large headers.
- Header size is reduced, allowing more rows per block, thus reducing I/O.
- Splitting and distributing into separate files with repeating primary key.
- For an end user, the split appears as a single table through a view.

Performance issues: Vertical Splitting Facts

Example: Consider a 100 byte header for the member table such that 20 bytes provide complete coverage for 90% of the queries.

Split the member table into two parts as follows:

1. Frequently accessed portion of table (20 bytes), and

2. Infrequently accessed portion of table (80+ bytes).
   Why 80+?

Note that primary key (member_id) must be present in both tables for eliminating the split.

Performance issues: Vertical Splitting Good vs. Bad

Scanning the claim table for most frequently used queries will be 500% faster with vertical splitting

Ironically, for the “infrequently” accessed queries the performance will be inferior as compared to the un-split table because of the join overhead.

Pre-joining ...

- Identify frequent joins and append the tables together in the physical data model.
- Generally used for 1:M such as master-detail.
- Additional space is required as the master information is repeated in the new header table.

Pre-Joining...

normalized

Master

Sale_ID	Sale_date	Sale_person

1 M

Tx_ID	Sale_ID	Item_ID	Item_Qty	Sale_Rs

Detail

denormalized

Tx_ID	Sale_ID	Sale_date	Sale_person	Item_ID	Item_Qty	Sale_Rs

Pre-Joining: Typical Scenario

Typical of Market basket query

Join ALWAYS required

Tables could be millions of rows

Squeeze Master into Detail

Repetition of facts. How much?

Adding Redundant Columns…

Table_1

ColA	ColB

Table_1’

ColA	ColB	ColC

Table_2

ColA	ColC	ColD	…	ColZ

Table_2

ColA	ColC	ColD	…	ColZ

Adding Redundant Columns...

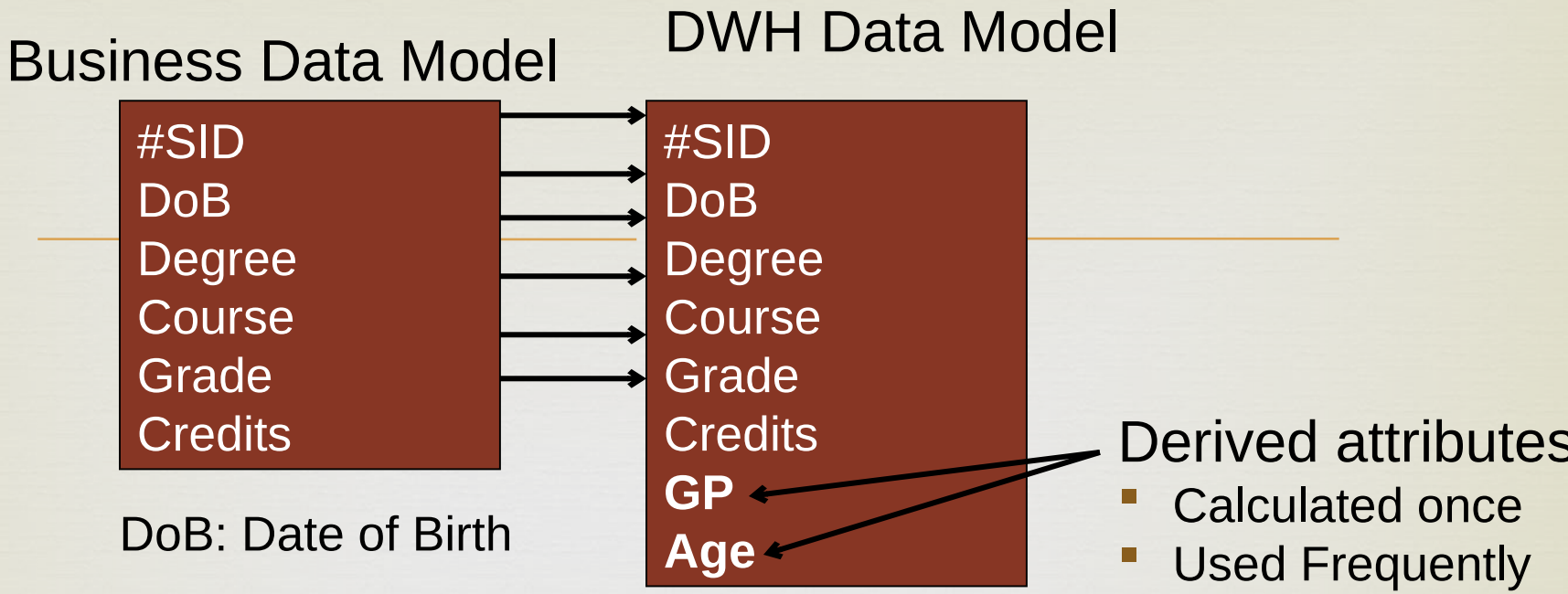
Columns can also be moved, instead of making them redundant. Very similar to pre-joining as discussed earlier.

EXAMPLE

Frequent referencing of code in one table and corresponding description in another table.

- A join is required.
- To eliminate the join, a redundant attribute added in the target entity which is functionally <u>independent</u> of the primary key.

Derived Attributes: Example



Age is also a derived attribute, calculated as Current_Date – DoB (calculated periodically).

GP (Grade Point) column in the data warehouse data model is included as a derived value. The formula for calculating this field is Grade*Credits.

Derived Attributes: Example

Advantages	Disadvantages
No need to look up source values each time a derivable value is needed	Running data manipulation language (DML) statements against the source data requires recalculation of the derivable data
No need to perform a calculation for every query or report	

Database denormalization tips

1. Instead of trying to denormalize the whole database right away, focus on particular parts that you want to speed up.

2. Do your best to learn the logical design of your application really well to understand what parts of your system are likely to be affected by denormalization.

3. Analyze how often data is changed in your application; if data changes too often, maintaining the integrity of your database after denormalization could become a real problem.