Denormalization in Databases

Rui Oliveira© rui.oliveira@ipleiria.pt

References

Further study required

- *Denormalization Effects on Performance of RDBMS*, Lawrence Sanders & Seungkyoon Shin, Proceedings of the 34th Hawai Inernational Conference on System Science
- *Oracle Performance Tuning (chapter 5)*, M. Gurry & P. Corrigan, O'Reilly

Normalization: remember

NORMALIZATION goals

- Store relevant data
- Eliminate redundant data
- Avoid NULL keys
- Minimize the impact of data insertion/update/removal

Denormalisation

- The **voluntary action** of creating tables while disregarding normalization goals

Rui Oliveira © rui.oliveira@ipleiria.pt

DEnormalization: when?

When we need to
reduce the response time of
database queries.

DEnormalization: how?

Reduce the number of
tables involved in the low-performance queries

AND/OR

Add redundancy to the database

Life lesson: “There’s no such thing as **free meals**”

DEnormalization secondary effects

- Redundancy has to be controlled **<u>at all times</u>** (*data integrity* has to be ensured)

The database must be programmed to control the redundancy, meaning:

- Someone (a Software Engineer) has to do it ☹☹
- The DB has to execute extra operations to ensure data integrity ☹☹☹☹

Note: check out the next chapter about integrity rules

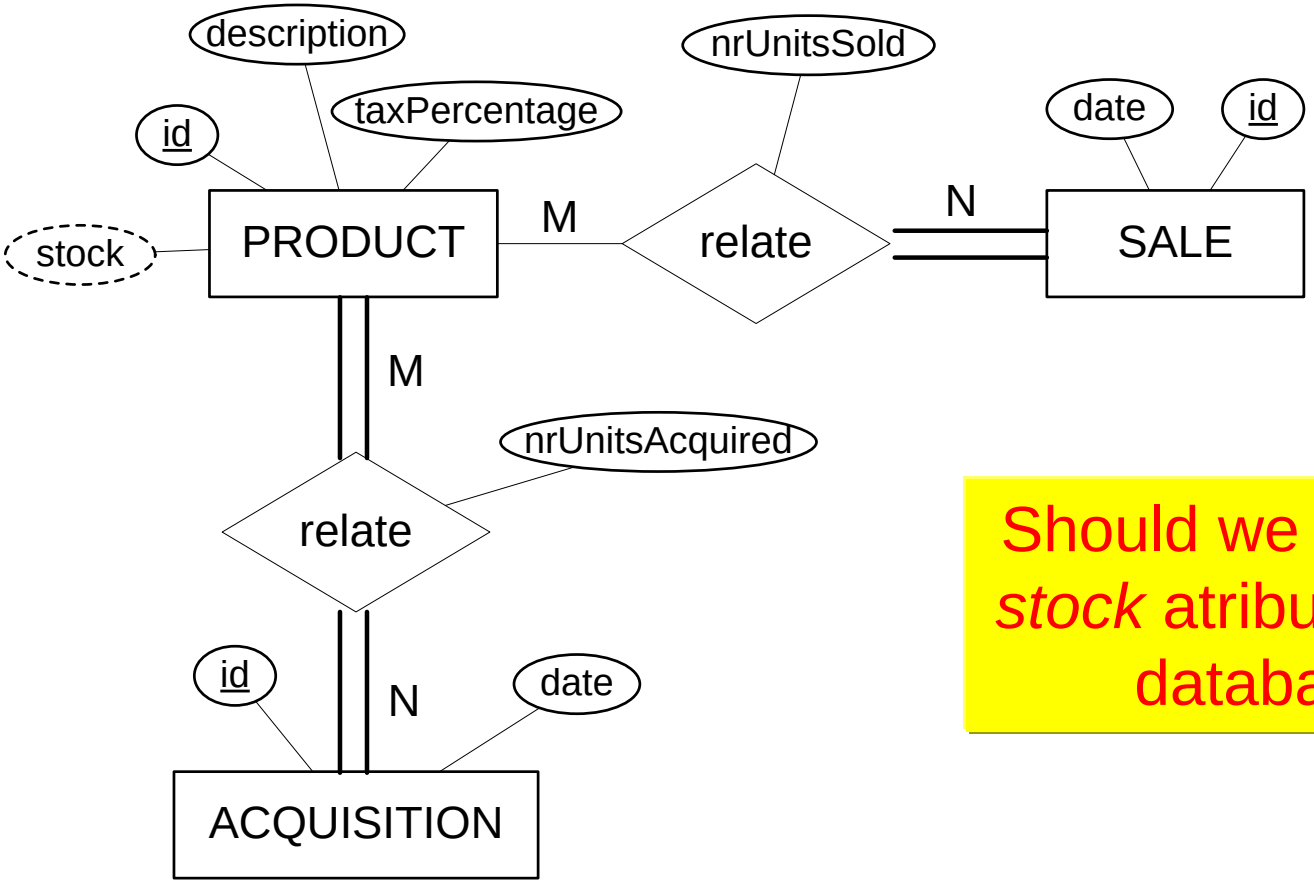
Ways of doing it

Standard ways

- Add redundancy
  - Store calculated attributes into tables
  - Add redundant columns
- Merge tables
  - Beware: NULLs will appear
- Create materialized Views

Example

A possible denormalization scenario



Should we store the *stock* atribute on the database?

Conclusions

Do you *REALLY* need to denormalize the database?

Prior to answering this question, analyze if:

- The DB well designed;
- You are using other performance improvement techniques? (like indexation);
- The performance boost will compensate the continuous effort of enforcing data consistency.