Relational databases (2\textsuperscript{nd} generation) were designed for traditional banking-type applications with well-structured, homogenous data elements (vertical & horizontal homogeneity) and a minimal fixed set of limited operations (e.g., set & tuple-oriented operations).

New applications (e.g., CAD, CAM, CASE, OA, and CAP), however, require concurrent modeling of both data and processes acting upon the data.

Hence, a combination of database and software-engineering disciplines lead to the 3\textsuperscript{rd} generation of database management systems: Object Database Management Systems, ODBMS.

Note that a classic debate in database community is that do we need a new model or relational model is sufficient and can be extended to support new applications.

People in favor of relational model argue that:

- New versions of SQL (e.g., SQL-92 and SQL3) are designed to incorporate functionality required by new applications (UDT, UDF, …).
- Embedded SQL can address almost all the requirements of the new applications.
- “Object people”, however, counter-argue that in the above-mentioned solutions, it is the application rather than the inherent capabilities of the model that provides the required functionality.

Object people say there is an impedance mismatch between programming languages (handling one row of data at a time)
and SQL (multiple row handling) which makes conversions inefficient.

- *Relational people* say, instead of defining new models, let’s introduce set-level functionality into programming languages.
- What do you think?
- Read “Evolution of Data Management” by Jim Gray.
- Read “Object-Relational DBMS – The Next Wave” by Michael Stonebraker. (Both members of National Academy of Engineering.)

Other problems with RDBMS:

- Short-lived transactions
- Schema changes are difficult: most organizations are locked into their existing database structures. Taylor in 1992 said: Organizations are unable to make these changes because they cannot afford the time and expense required modifying their information systems (sounds familiar? Y2K, Euro, …).
- Poor at navigational access (moving between records/objects), and strong in content-based associative access (e.g., navigate your family tree with “people” relation in SQL!).