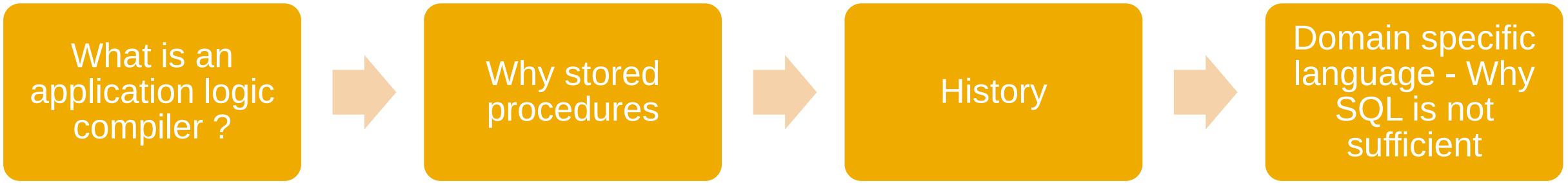
The Emergence of Application Logic Compilers

Stefan Dipper, SAP BW Development

Sept, 2013

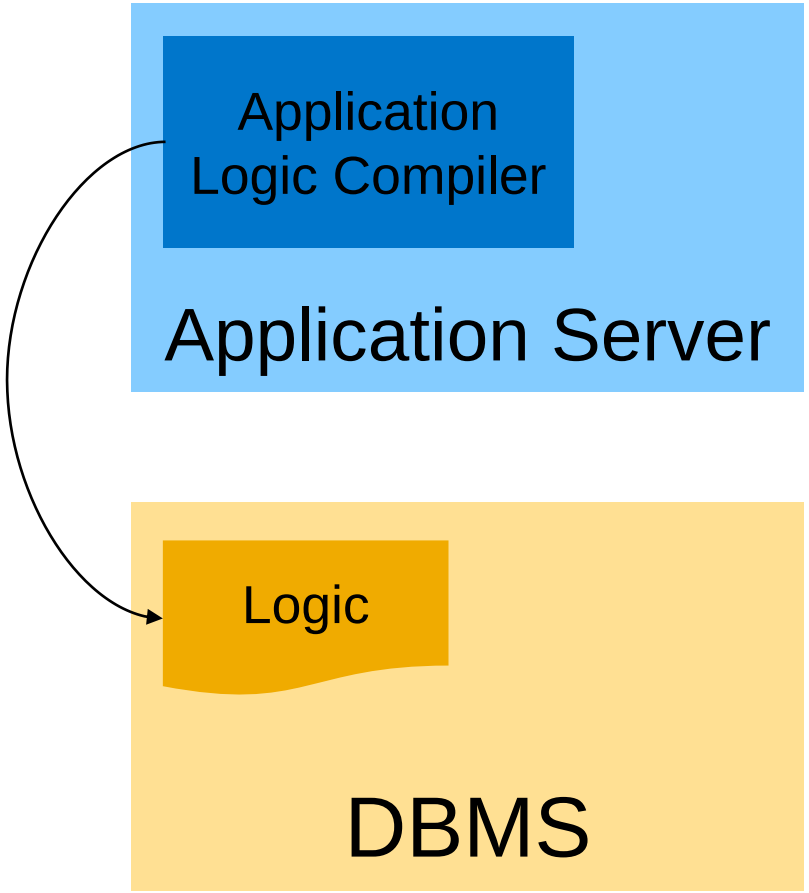
Public

Agenda



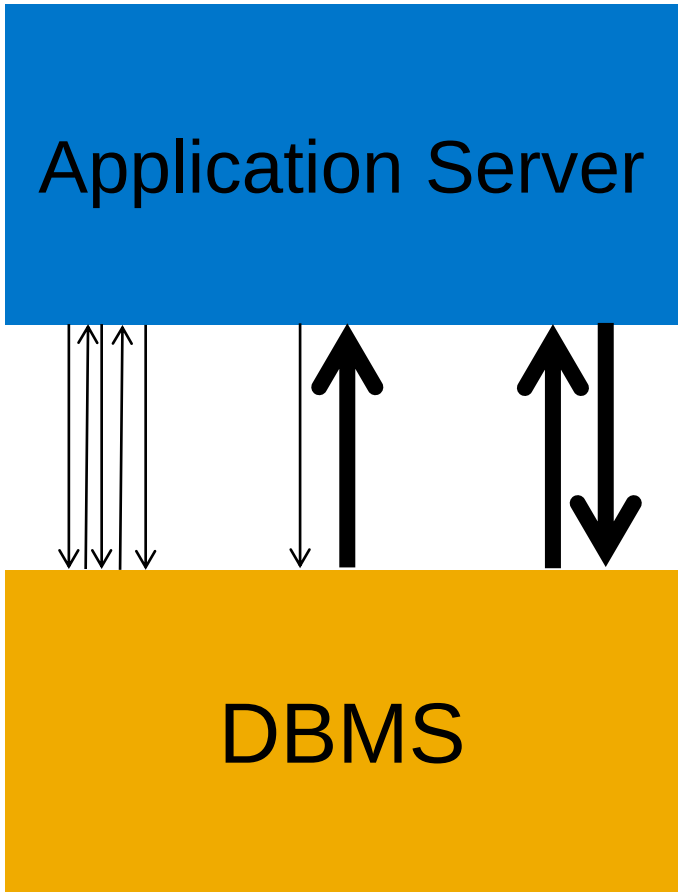
What is an application logic compiler ?

- Application logic is translated/compiled into a format that can be executed in the DB engine – Stored Procedure / View
- Compilation may take place on demand due to specific application logic. Stored procedure code often is only temporary.
- Because of specific application logic, stored procedure should be written using a domain specific language



Why stored procedures ?

- Performance
- Reduce interaction between Application and DB → code to data
  - reduce amount of data transferred between APP and DB
  - aggregate data wherever possible before returning to application
  - avoid mass data manipulation in app server
- but: single record operations based on user interaction (e.g. maintain customer contact data) are no problem
- Aggregation delivers biggest performance gain
  - Push down everything aggregation depends on

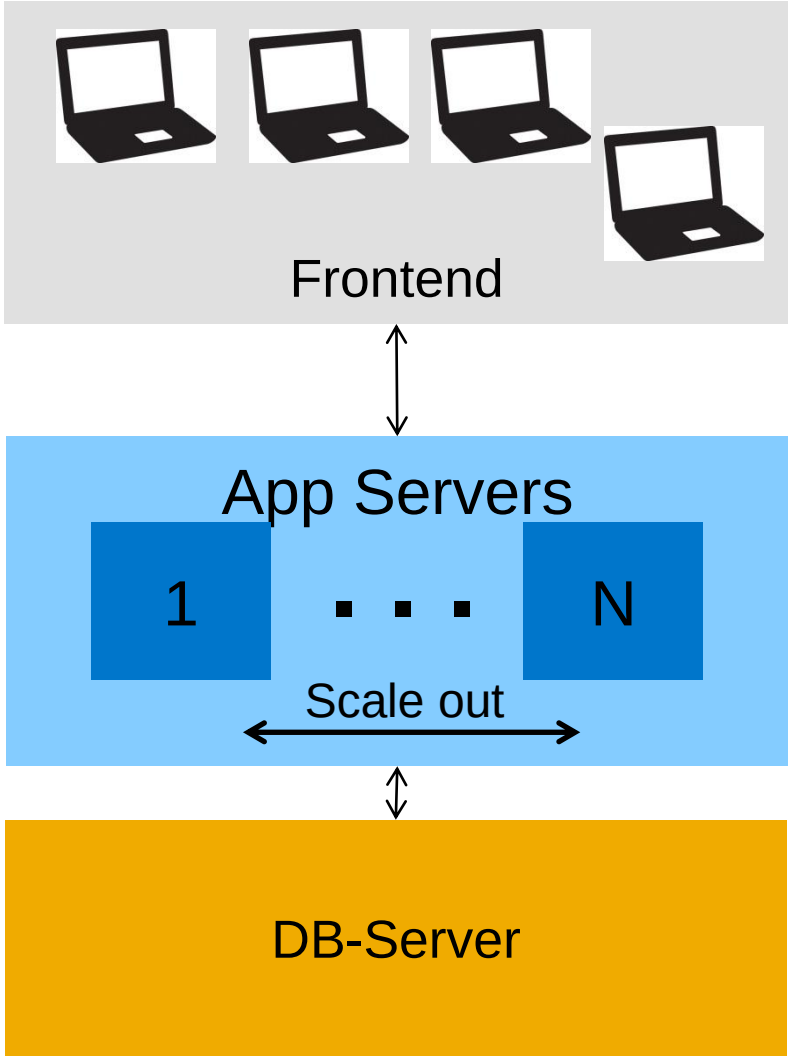


History – R/3 Programming model

1992

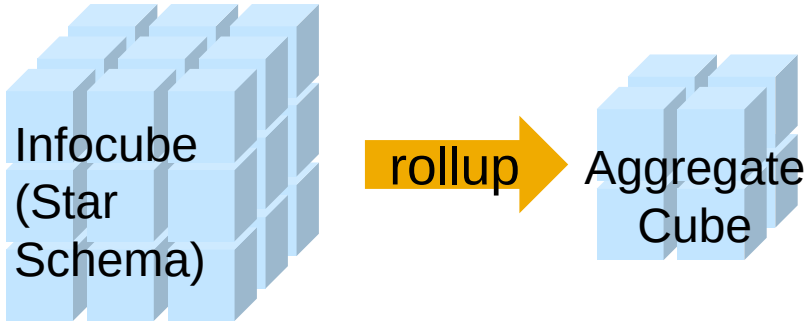
- 3 – Tier model: 1 DB Server, 1..n application servers, Frontend
  - Avoid load on db server whenever possible
  - Scale-out possible via application servers
  - Special transaction model lets DB run in cheap isolation level
- Platform independence – do not depend on special features of a DBMS
- Special programming language:
  - ABAP evolved from a reporting language
  - OPEN-SQL (SAPs SQL language abstraction) deeply integrated into ABAP
  - ABAP as a procedural extension to OPEN SQL (similar to PL/SQL), OO came afterwards

=>DBMS mainly as data storage, all logic via „do it yourself“ in application server

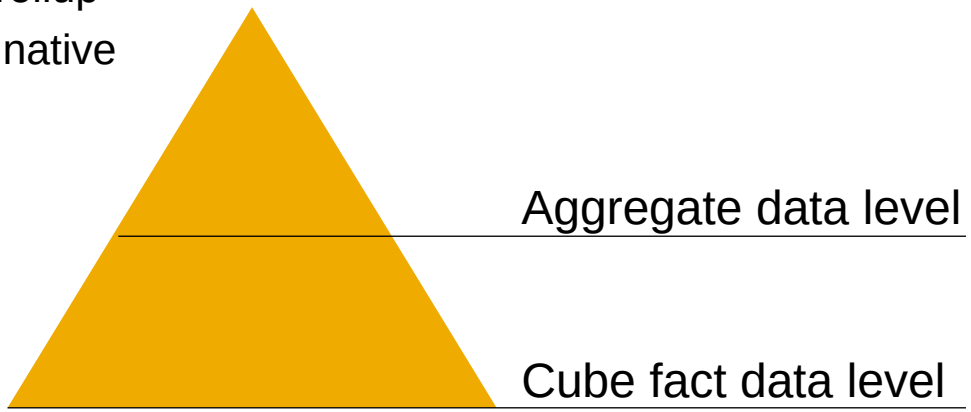


History – BW

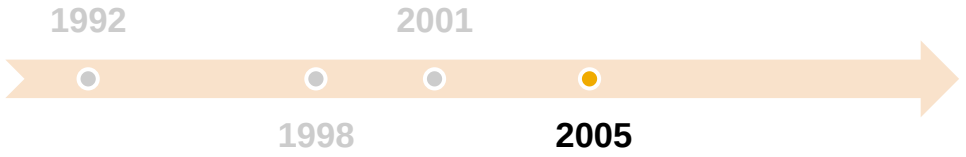
- BW (SAPs data warehouse product) introduced platform specific coding on application level, leveraging database-specific features
- SAP tried to optimize performance in a platform-independent way where appropriate
  - BW: Aggregate Cubes platform independent
  - BW Aggregates better than DB native technology (materialized views, aggregated summary tables)
  - Application-controlled data flow and application enforced referential integrity on fact data enables cheap delta rollup
  - Use of Parent-Child – Hierarchies not possible using native technology



=>more load and dependency on DBMS



History – BWA



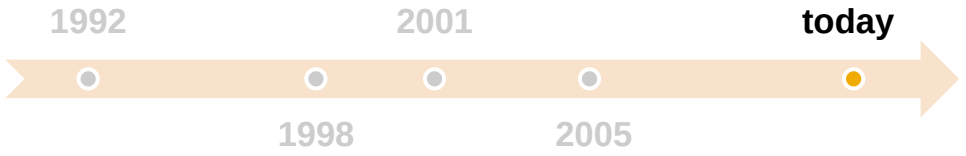
- BWA: Business Warehouse Accelerator, Predecessor of Hana
- Columnar Storage, Compression, Massive Parallelism
- Reduction of DB response time & reduction of DB result size based on ability to process restricted key figures (“measures”)
  -> first time application logic entered DB interface
- Pseudo-SQL:

```
Select customer_group,
       SUM( amount in 2004 ),
       SUM( amount in US )
from cube_a
where …
group by customer group
```

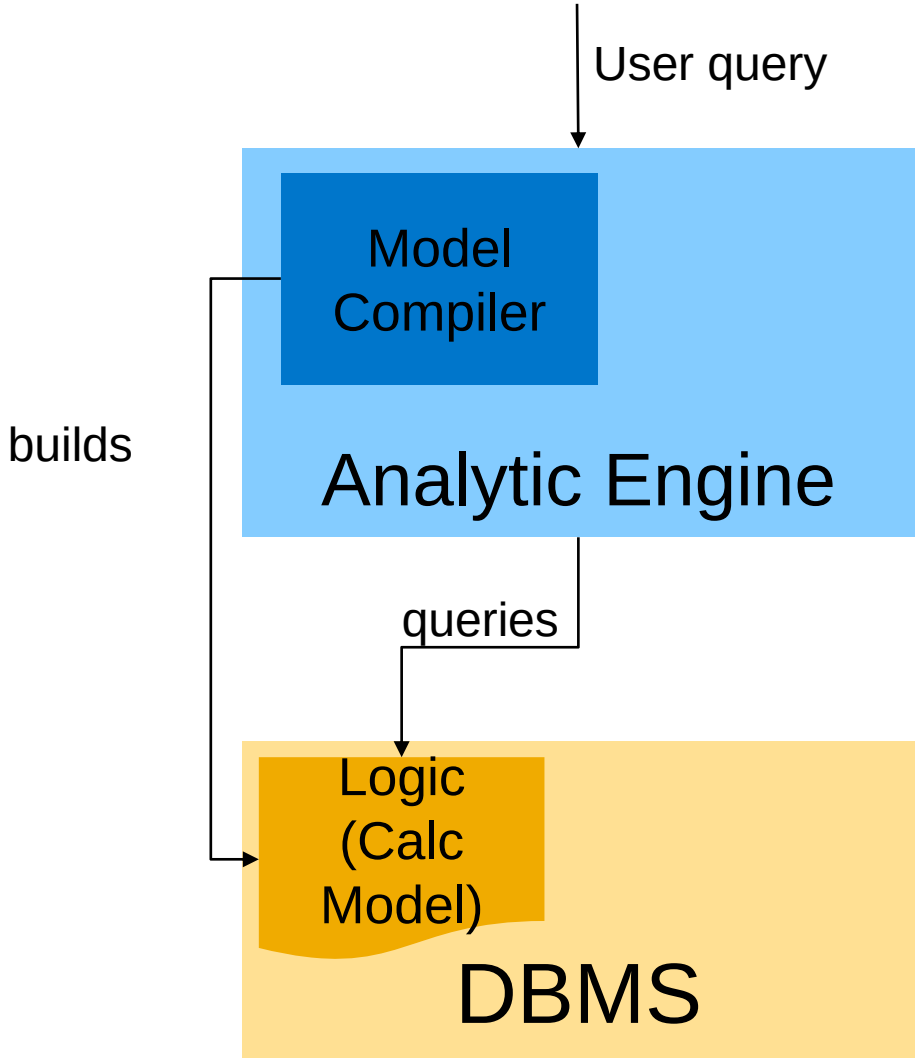
=>Application Logic processed in DB layer

- „predictible“ or „stable“ response times

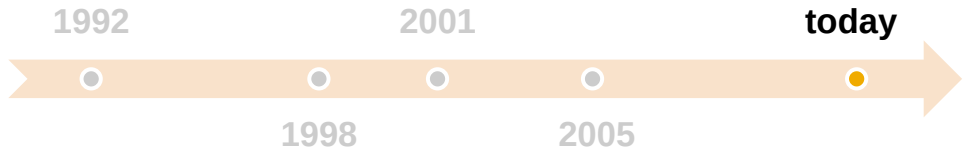
Today: OLAP

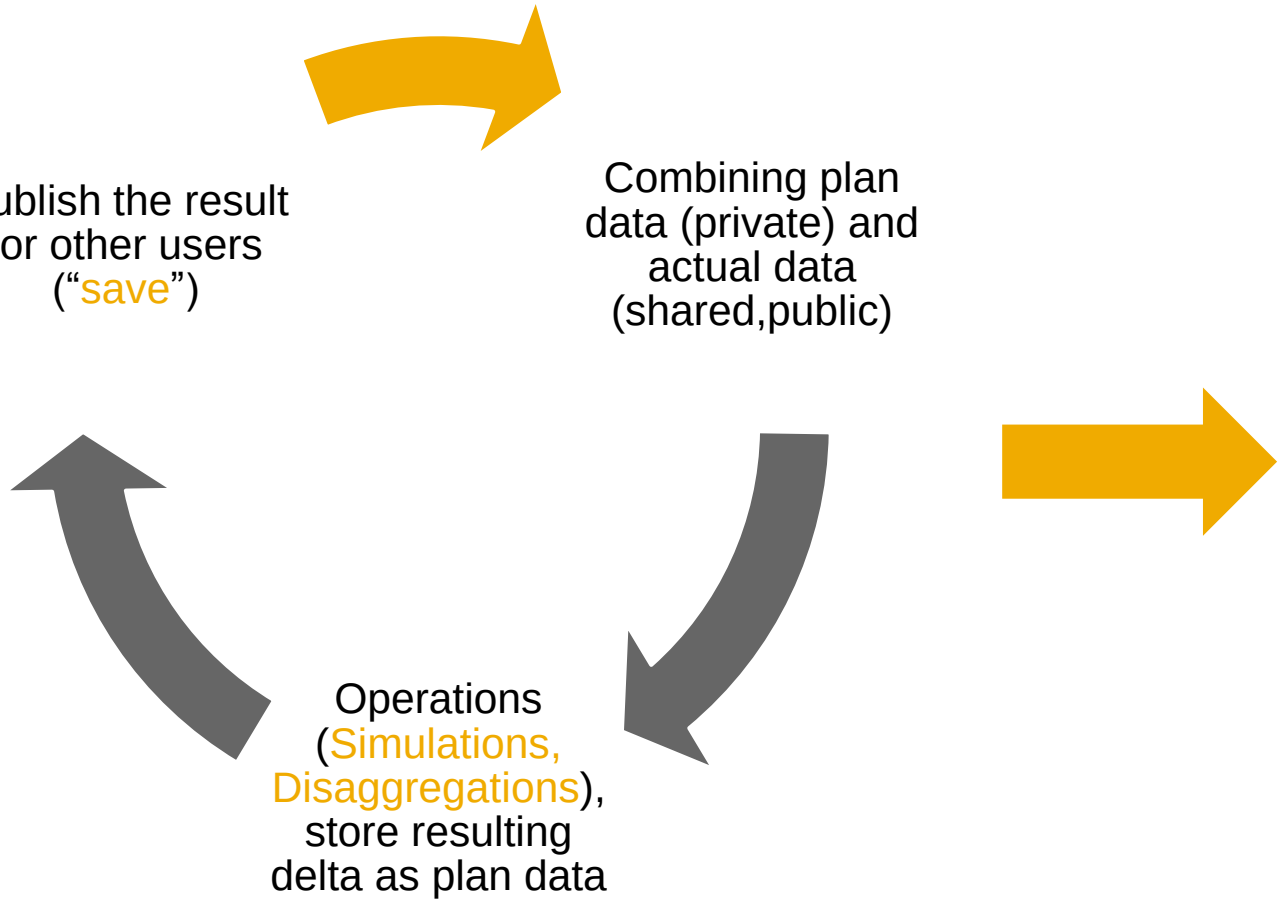


- Stored procedure on demand
- Original Idea: separation of concerns: application semantics in application layer, DBMS agnostic to semantics
- “stored procedure” is a data flow graph: nesting of aggregation, formula logic, currency conversion, hierarchy handling…
- But: huge “stored procedure” code (MBs !!) → primitives are too fine granular, so more reusable business logic building blocks needed



Today: In Memory Planning Engine





- Uses a data flow graph that is interactively built and queried
- Data flow graph can contain procedural nodes (planning functions)
- DB Planning interface abstracts the graph from the user.

Domain specific language – why SQL is not sufficient

- Application logic is very customer specific
  - different laws, tax regulations, …→ all leads to generalization
  - Part of customers intellectual property
  - parametrization or even customer exits
  - Structural differences in data models prevent simple functional reuse
- Example: Hierarchies (parent-child)
  - SQL does not really consider hierarchies
  - Common table expressions allow some processing
  - MDX offers more functionality

Aggregations and Arithmetics

- Special Aggregation Types (NOP, Median, …)
- Arithmetics: Application specific calculation rules/ special values – null handling, overflows, division by zero, zero, …
  - Special data types (decfloat), special handling of rounding, zero, …
  - 20 + NULL <> SUM(20, NULL) ?
  - Error in one cell must not break the query

Product Group	**Product**	**Price**
Soft Drinks	Cola	1.50 $/bottle
	Orange	1.50 $/bottle
	Total	1.50 $/bottle
Beer	Becks	2.00 $/bottle
	Budweiser	1.75 $/bottle
	Total	X
Total		X

Currencies and Units in Aggregation and Calculation

- Application deals with units –
  - DB does not know about measure/unit pairs at all
  - Calculations and aggregations considering units
- Product specific units cannot be made homogeneous
- currencies/units are context specific
- data driven currency conversion at query runtime allows simulations, “what if” (at constant currency…)
- Quantity conversion (e.g. for stock reports, logistics, …)

Product Group	Product	Price	Quantity	Amount
Beverages	Cola	1.50 $/bottle	5 bottles	7.50 $
	Milk	3 $/can	0 cans	0.00 $
	Total	X	5 bottles	7.50 $
Bread	Toast	4.00 $/kg	1 kg	4.00 $
Vegetables	Tomatoes	2.00 $/kg	1 kg	2.00 $
	Total	X	*	21.00 $
Total		X	*	21.00 $

Thank you