



[SOT213] If your SQL reads like Assembler you're doing it all wrong.

Lars Breddemann

May 31st, 2020

04:00 - 05:00 UTC

How to make your solution use the database well and save mental capacity for the more important things in life.

Supporting *girls who*
code

<http://bit.ly/saponlinetrack-fundraiser>



Sponsored by **openSAP**



SAP PRESS

ESPRESSO TUTORIALS



Lars Breddemann

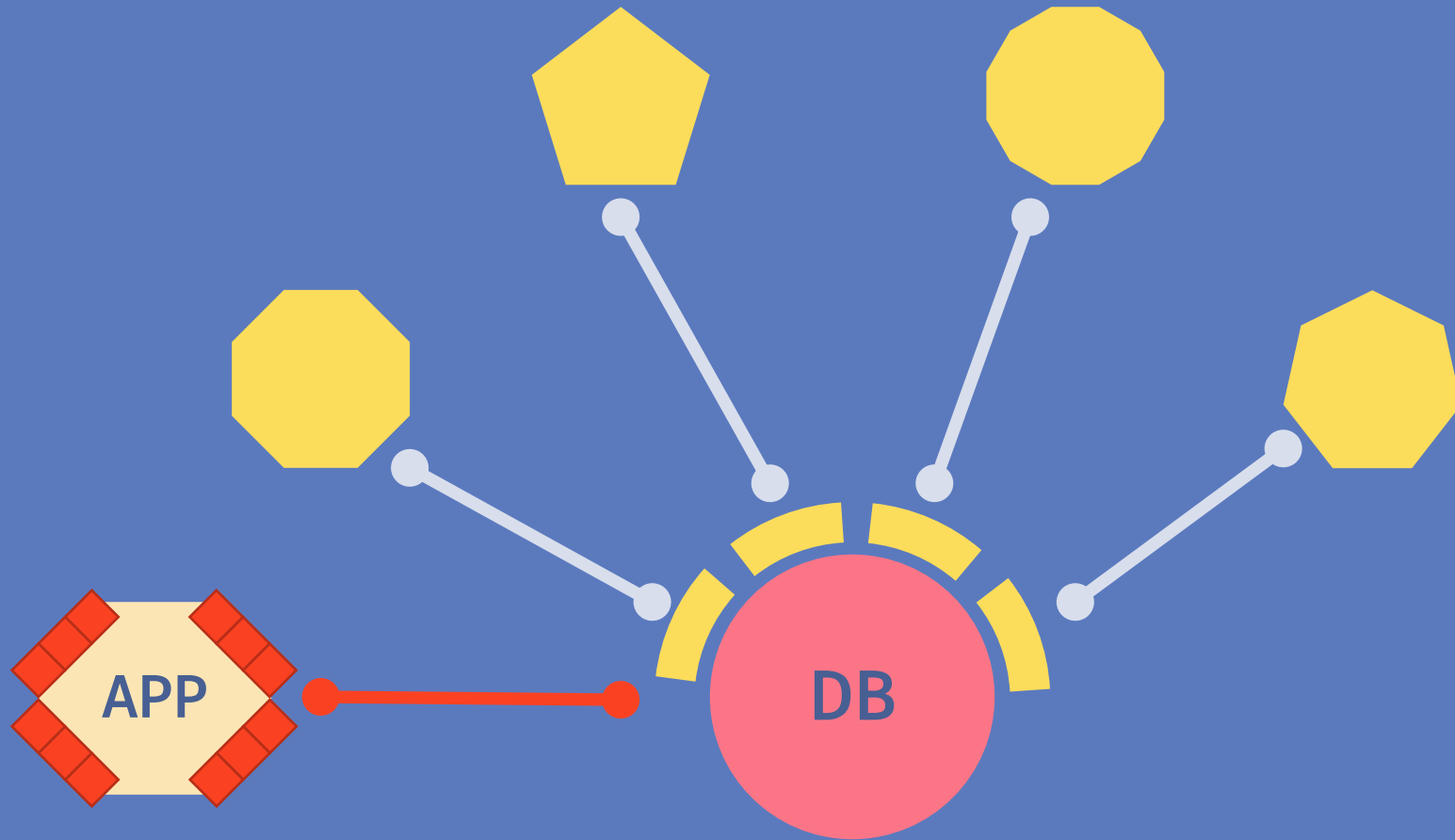
[@LBREDDemann](#)

<https://lbreddemann.org/>

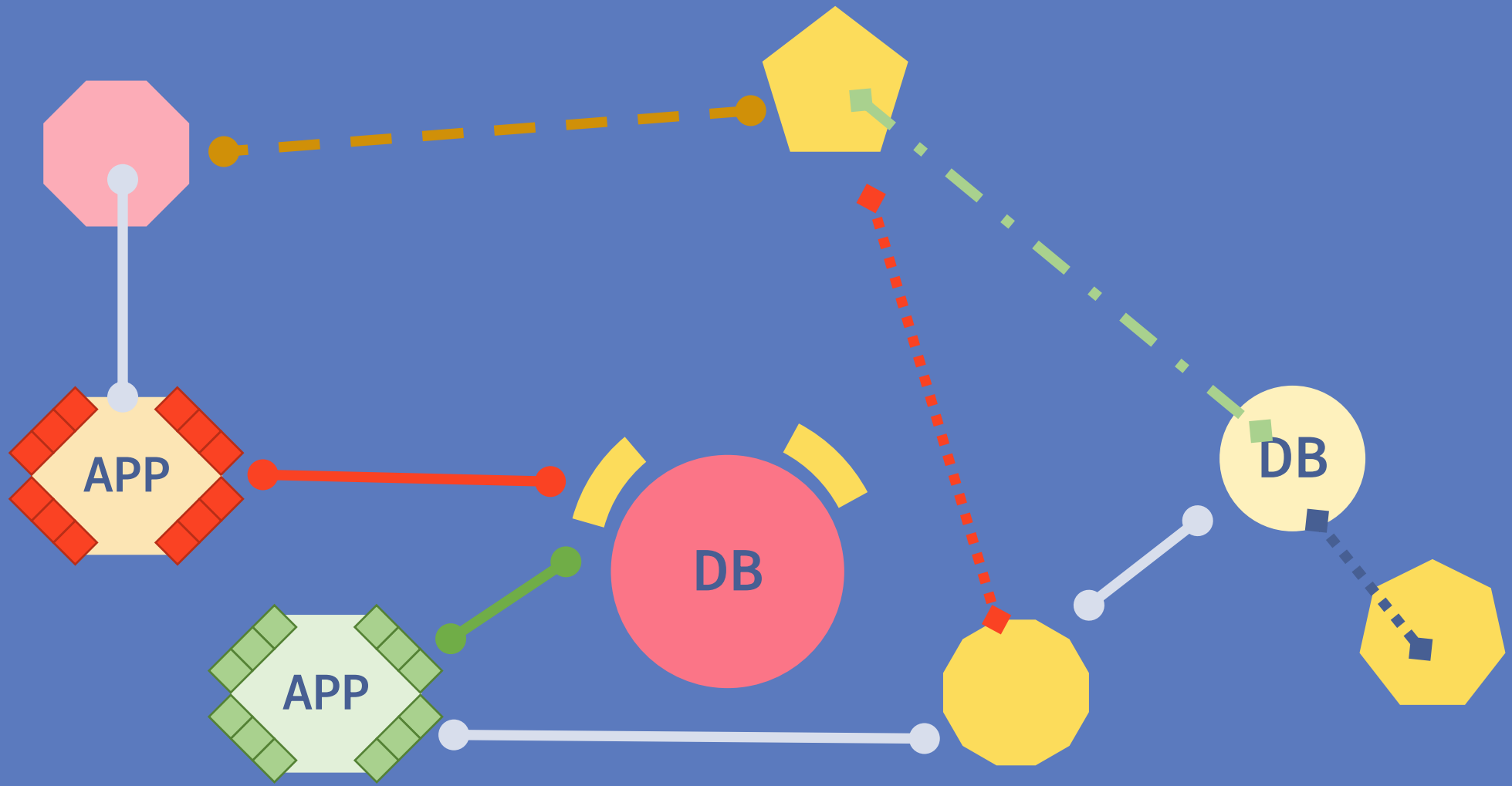
<https://dataprocessinsights.com.au/>

lars@dataprocessinsights.com.au

The idea and promise of the information organisation







An example

```

SELECT OCRD."CardCode" AS "Customer Code", OCRD."CardName" AS "Customer Name",
SUM(
    CASE
        WHEN "SYSCred" != 0 THEN "SYSCred" * -1
        ELSE "SYSDeb"
    END) AS "Balance Due", IFNULL(SUM(
    CASE
        WHEN DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) < 0
    THEN
        CASE
            WHEN JDT1."BalDueCred" <> 0 THEN JDT1."BalDueCred" * -1
            ELSE JDT1."BalDueDeb"
        END
    END), 0.00) AS "Future Remit", IFNULL(SUM(
    CASE
        WHEN (DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) >= 0
    AND
        DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) < 30)
    THEN
        CASE
            WHEN JDT1."BalDueCred" <> 0 THEN JDT1."BalDueCred" * -1
            ELSE JDT1."BalDueDeb"
        END
    END), 0.00) AS "0-30 days", IFNULL(SUM(
    CASE
        WHEN (DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) >= 30
    AND
        DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) < 60)
    THEN
        CASE
            WHEN JDT1."BalDueCred" <> 0 THEN JDT1."BalDueCred" * -1
            ELSE JDT1."BalDueDeb"
        END
    END), 0.00) AS "31 to 60 days", IFNULL(SUM(
    CASE
        WHEN (DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) >= 60
    AND
        DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) < 90)
    THEN
        CASE
            WHEN JDT1."BalDueCred" <> 0 THEN JDT1."BalDueCred" * -1
            ELSE JDT1."BalDueDeb"
        END
    END), 0.00) AS "61 to 90 days", IFNULL(SUM(

```

```

    CASE
        WHEN (DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) >= 90
    AND
        DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) < 120)
    THEN
        CASE
            WHEN JDT1."BalDueCred" <> 0 THEN JDT1."BalDueCred" * -1
            ELSE JDT1."BalDueDeb"
        END
    END), 0.00) AS "91 to 120 days", IFNULL(SUM(
    CASE
        WHEN DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) >= 120
    THEN
        CASE
            WHEN "BalDueCred" <> 0 THEN "BalDueCred" * -1
            ELSE "BalDueDeb"
        END
    END), 0.00) AS "120+ days"
FROM JDT1,
    OCRD
WHERE JDT1."ShortName" = OCRD."CardCode" AND OCRD."CardType" = 'c'
GROUP BY OCRD."CardCode", OCRD."CardName" HAVING SUM(
    CASE
        WHEN "SYSCred" <> 0 THEN "SYSCred" * -1
        ELSE "SYSDeb"
    END) > 0 OR SUM(
    CASE
        WHEN "SYSCred" <> 0 THEN "SYSCred" * -1
        ELSE "SYSDeb"
    END) < 0
ORDER BY OCRD."CardCode";

```

```

SELECT   OCRD."CardCode" AS "Customer Code",   OCRD."CardName" AS
"Customer Name", SUM(
CASE
    WHEN "SYSCred" != 0 THEN "SYSCred" * -1
    ELSE "SYSDeb"
END) AS "Balance Due",   IFNULL(SUM(
CASE
    WHEN DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) < 0 THEN
CASE
    WHEN JDT1."BalDueCred" <> 0 THEN JDT1."BalDueCred" * -1
    ELSE JDT1."BalDueDeb"
END
END), 0.00) AS "Future Remit", IFNULL(SUM(
CASE
    WHEN (DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) >= 0 AND
DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) < 30) THEN
CASE
    WHEN JDT1."BalDueCred" <> 0 THEN JDT1."BalDueCred" * -1
    ELSE JDT1."BalDueDeb"
END
END), 0.00) AS "0-30 days", IFNULL(SUM(
CASE
    WHEN (DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) >= 30 AND
DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) < 60) THEN
CASE
    WHEN JDT1."BalDueCred" <> 0 THEN JDT1."BalDueCred" * -1
    ELSE JDT1."BalDueDeb"
END
END), 0.00) AS "31 to 60 days", IFNULL(SUM(
CASE
    WHEN (DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) >= 60 AND
DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) < 90) THEN
CASE
    WHEN JDT1."BalDueCred" <> 0 THEN JDT1."BalDueCred" * -1
    ELSE JDT1."BalDueDeb"
END
END), 0.00) AS "61 to 90 days", IFNULL(SUM(
CASE
    WHEN (DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) >= 90 AND
DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) < 120) THEN
CASE
    WHEN JDT1."BalDueCred" <> 0 THEN JDT1."BalDueCred" * -1
    ELSE JDT1."BalDueDeb"

```

```

END
END), 0.00) AS "91 to 120 days", IFNULL(SUM(
CASE
    WHEN DAYS_BETWEEN(JDT1."DueDate", (SELECT NOW() "NOW" FROM DUMMY)) >= 120 THEN
CASE
    WHEN "BalDueCred" <> 0 THEN "BalDueCred" * -1
    ELSE "BalDueDeb"
END
END), 0.00) AS "120+ days"

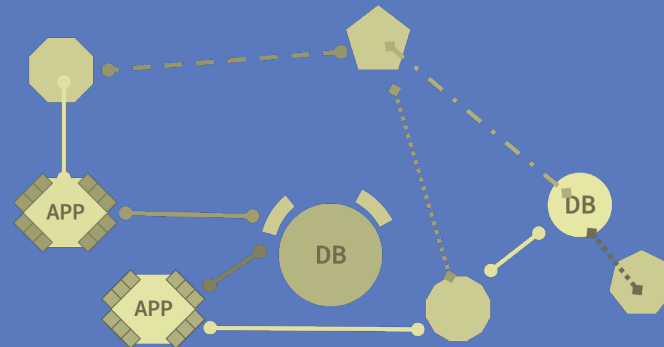
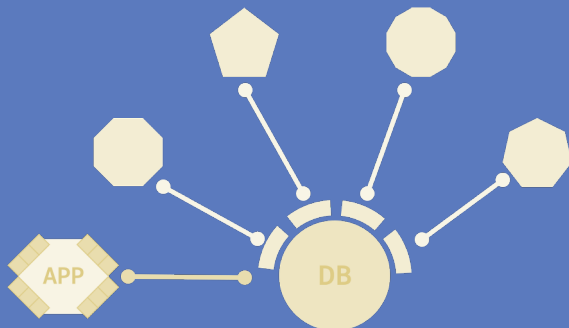
FROM JDT1,
    OCRD

WHERE JDT1."ShortName" = OCRD."CardCode" AND OCRD."CardType" = 'c'
GROUP BY OCRD."CardCode", OCRD."CardName" HAVING SUM(
CASE
    WHEN "SYSCred" <> 0 THEN "SYSCred" * -1
    ELSE "SYSDeb"
END) > 0 OR SUM(
CASE
    WHEN "SYSCred" <> 0 THEN "SYSCred" * -1
    ELSE "SYSDeb"
END) < 0
ORDER BY OCRD."CardCode";

```

Customer Code	Customer Name	Balance Due	Future Remit	0-30 days	31 to 60 days	61 to 90 days	91 to 120 days	120+ days
C00001	Robel, Hermiston and Smith	-627,931.508	-160,271.472	-36,335.124	-37,673.722	-47,190.632	-36,628.754	-307,350.523
CC0007	Hand-Murazik	-632,987.217	-169,635.295	-35,240.028	-29,631.684	-42,136.16	-34,206.392	-320,518.273
CC0013	Beatty, Kris and Wolff	-620,051.886	-157,163.081	-35,818.7	-34,185.489	-37,212.401	-37,504.447	-312,085.708
CC0019	Leffler-Koch	-635,646.453	-163,050.199	-37,458.78	-39,761.597	-37,236.948	-31,412.663	-312,695.952
CC0020	O'Conner, Swaniawski and Rogahn	-622,834.399	-162,536.634	-35,920.432	-41,185.528	-34,567.145	-35,994.354	-311,275.339
CC0022	Brakus-Braun	-620,072.744	-163,802.038	-38,761.061	-36,414.422	-38,218.068	-37,313.894	-306,977.177
CC0026	Kautzer, Wolf and Conn	-624,792.548	-167,401.11	-36,237.754	-34,729.384	-32,660.201	-39,966.022	-308,952.022
. . .								

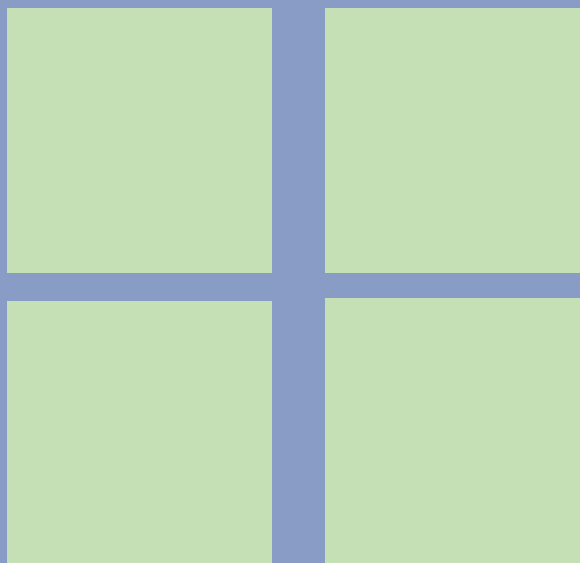
Invoice collection
Debt risk estimation
revising allowance for accounts



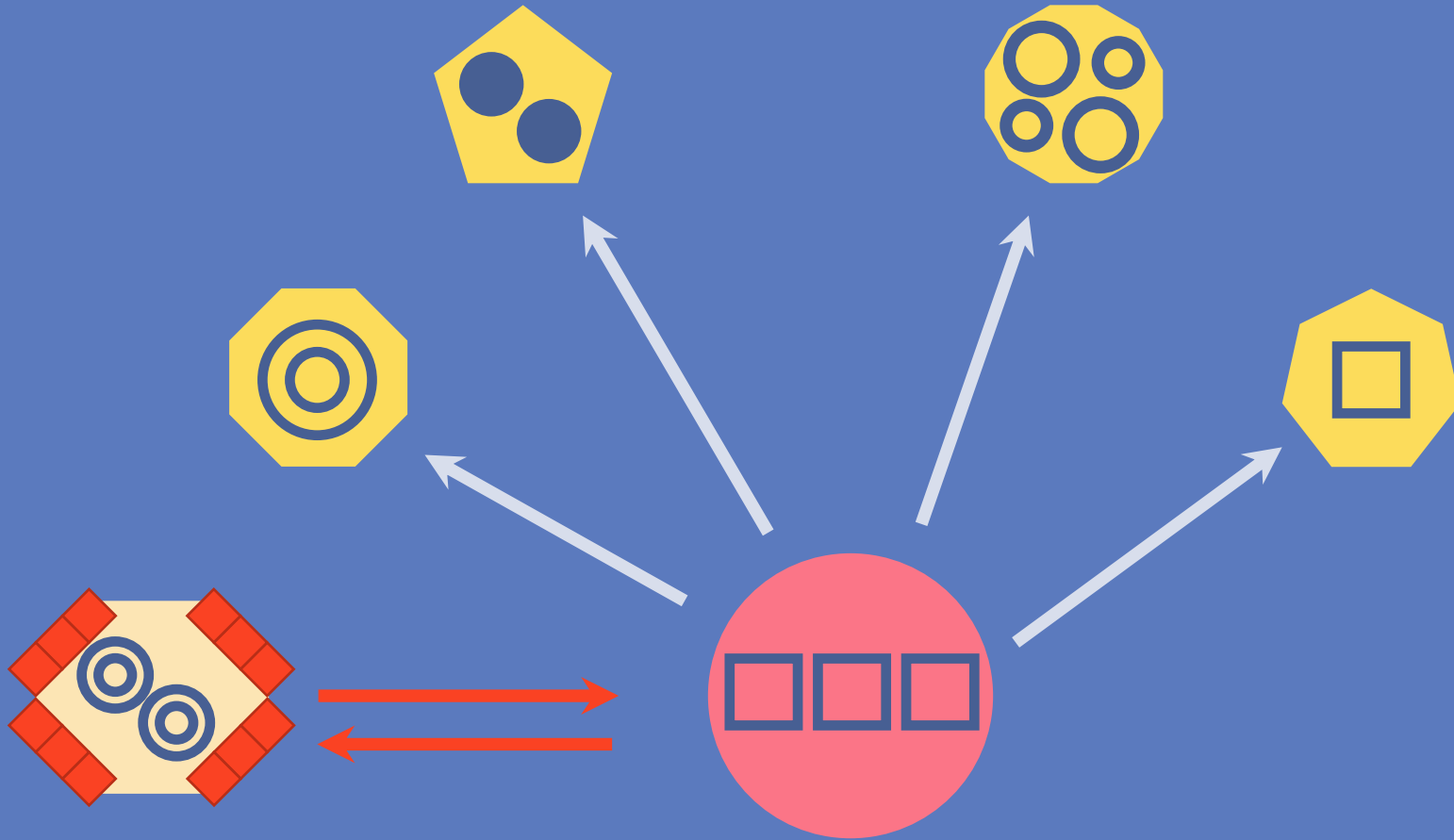
SQL

Devs/Orgs

Options



Schema-on-write / Meaning-on-read



Schema-on-write / Meaning-on-read

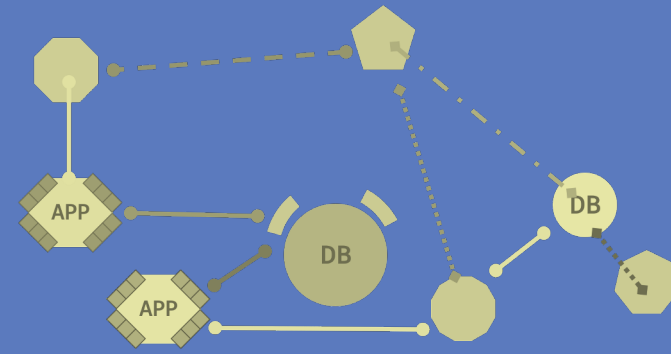
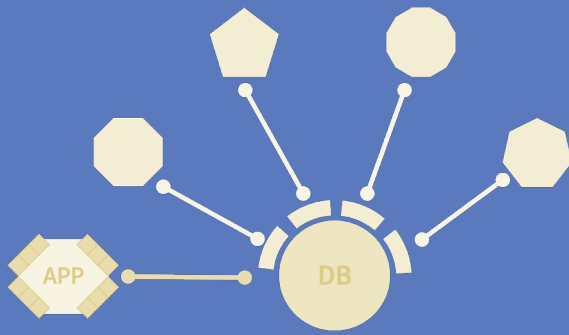
- Major benefit of SQL DBMS: data does not have to be read the same way as it was written
- Allows to compute new information from stored data
- Technical "navigation" of stored data not necessary
- **Reader** must understand what the encoded data means and how to read it
- Assigns own meaning

Conceptual gaps in modelling

- Snapshot data represents information about change
- Data modelling "laws" focus on data storage and on avoiding problems of technology
 - Lost writes, duplicate data, data non-compliant with domain
- Reads can be arbitrary
- DB constraints rule over what data can be stored, but not how it is read

SQL language landscape

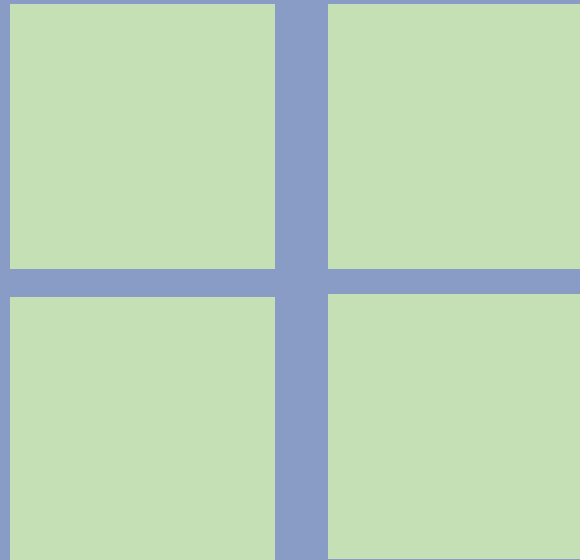
- Standardized, but not freely available
- Every DB speaks SQL – each its own dialect
- Compatibility is race to the bottom and following the big players



SQL

- Schema-on-write/Meaning-on-read
- Conceptual gaps in modelling
- SQL language

Devs/Orgs



Options

Devs/Orgs

SQL Training

Strong focus on syntax and commands

Simplistic examples (one SELECT per problem)

Anything complex used as motivation for procedural code

Lack of explanation what specific features are meant for

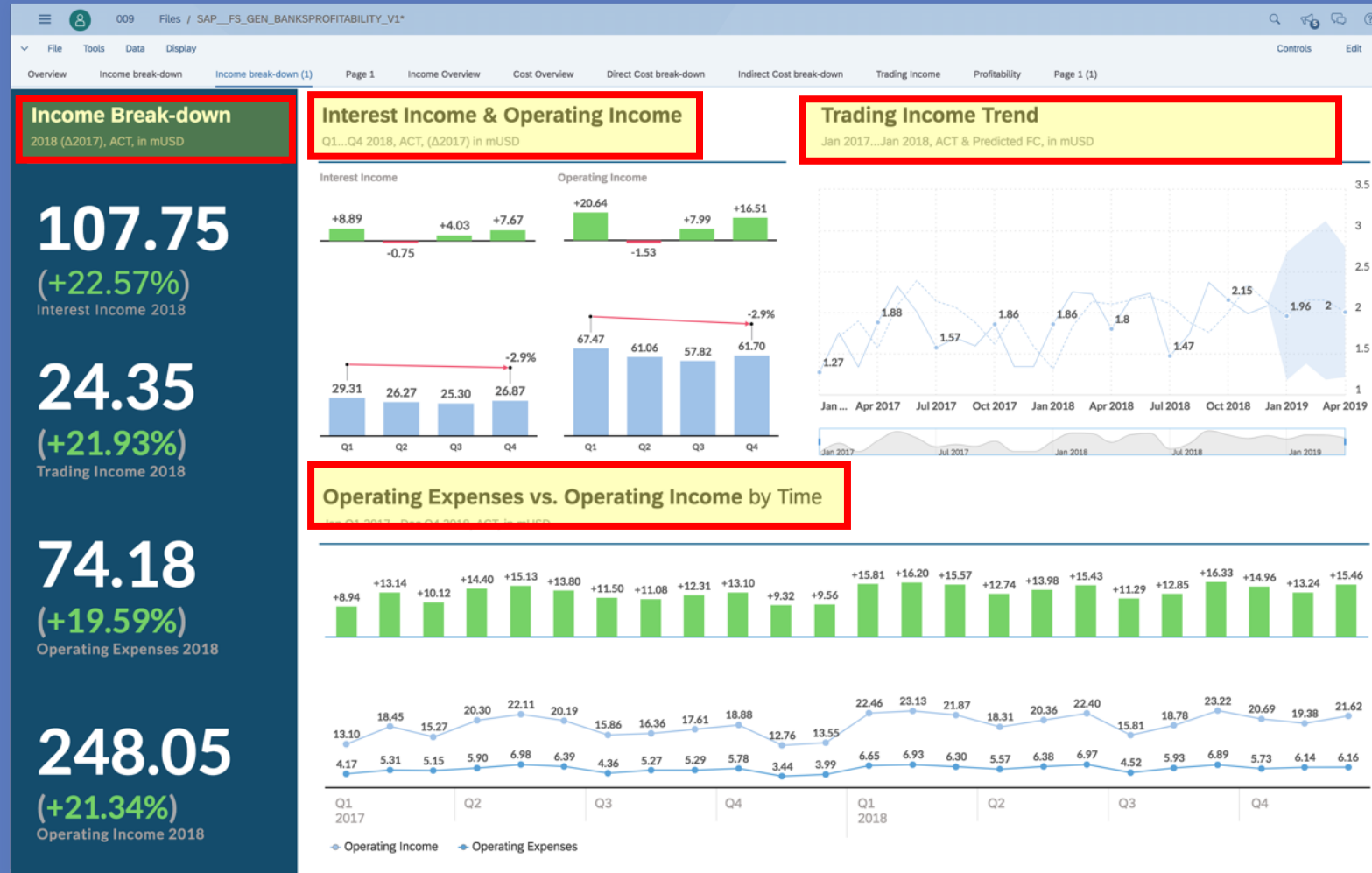
No coverage of how the database will/could be used

Legacy Systems & Culture

Tooling support

DB technology / DB usage

This is not the answer to a single question



Devs/Orgs

SQL Training

Strong focus on syntax and commands
Simplistic examples (one SELECT per problem)
Anything complex used as motivation for procedural code
Lack of explanation what specific features are meant for
No coverage of how the database will/could be used

Legacy Systems & Culture

Business logic in the application
Framework covers DB access
Getter/Setter approach to DB as bit-bucket
Control over data access with developers
Responsibility for perf./security with DBAs
Complex data access patterns hidden from developers

Tooling support

DB technology / DB usage

Complex data access pattern (selection)

ABAP Cluster/Pool tables

Multiple "tables" from ABAP point of view encoded into a single table on DB

Data with history

Valid_from/valid_to, Current-flag, just valid_to data

Default values

for items not stored in DB and "filled in" by the application

Hierarchy-Information

Parent-node (adjacency list), Path enumeration, Edge/node tables

Money-values

Integer column for value and one to store decimal shift, Decimal (fixed-point), Floating point (NO!!!!!!)

Flexible/Dynamic/Key-Value store tables

The One Table To Rule Them All

Complex data access pattern (selection)

ABAP Cluster/Pool tables

Multiple "tables" from ABAP point of view encoded into a single table on DB

Data with history

Valid_from/valid_to, Current-flag, just valid_to data

Default values

for items not stored in DB and "filled in" by the application

Hierarchy-Information

Parent-node (adjacency list), Path enumeration, Edge/node tables

Money-values

Integer column for value and one to store decimal shift, Decimal (fixed-point) or Floating point (NO!!!!!!)

Flexible/Dynamic/Key-Value store tables

The One Table To Rule Them All



Devs/Orgs

SQL Training

- Strong focus on syntax and commands
- Simplistic examples (one SELECT per problem)
- Anything complex used as motivation for procedural code
- Lack of explanation what specific features are meant for
- No coverage of how the database will/could be used

Legacy Systems & Culture

- Business logic in the application
- Framework covers DB access
- Getter/Setter approach to DB as bit-bucket
- Control over data access with developers
- Responsibility for perf./security with DBAs
- Complex data access patterns hidden from developers**

Tooling support

- Only syntax support – no support of meaningful part of the current data model
- SQL testing/checking available but not wide-spread or default
- No standard linter available
- Data modelling tools are for architects

DB technology / DB usage

Devs/Orgs

SQL Training

- Strong focus on syntax and commands
- Simplistic examples (one SELECT per problem)
- Anything complex used as motivation for procedural code
- Lack of explanation what specific features are meant for
- No coverage of how the database will/could be used

Legacy Systems & Culture

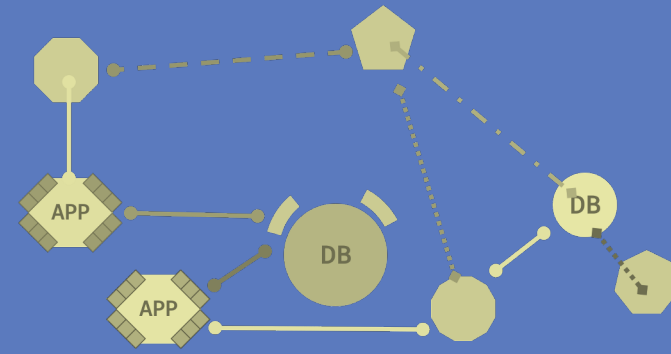
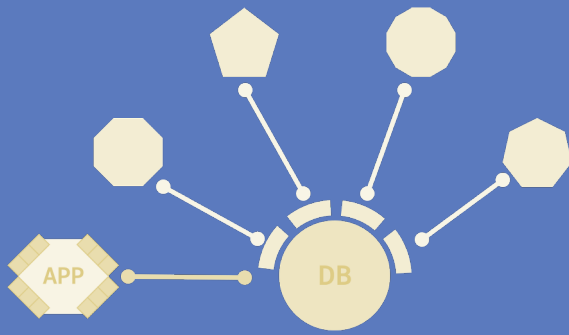
- Business logic in the application
- Framework covers DB access
- Getter/Setter approach to DB as bit-bucket
- Control over data access with developers
- Responsibility for perf./security with DBAs
- Complex data access patterns hidden from developers**

Tooling support

- Only syntax support – no support of meaningful part of the current data model
- SQL testing/checking available but not wide-spread or default
- No standard linter available
- Data modelling tools are for architects

DB technology / DB usage

- Focus on operational topics
- Reduce costs of DB
- No focus on better information
- "Look at data" and guess meaning
- Understood as equally accessible information
- Perceived Value of database a function of data volume



SQL

- Schema-on-write/Meaning-on-read
- Conceptual gaps in modelling
- SQL language

Devs/Orgs

**SQL
Training**

**Legacy
Systems
& Culture**

**Tooling
support**

**DB tech /
usage**

Options

Options



noSQL
Datalake

Leave problem to
next developer

Shared data
definitions and
convention

Top-Down semantic
modelling
(CASE, SOA)

Semantic Web

Get whole org. to
agree on meaning

Options



noSQL
Datalake

Leave problem to
next developer

Shared data
definitions and
convention

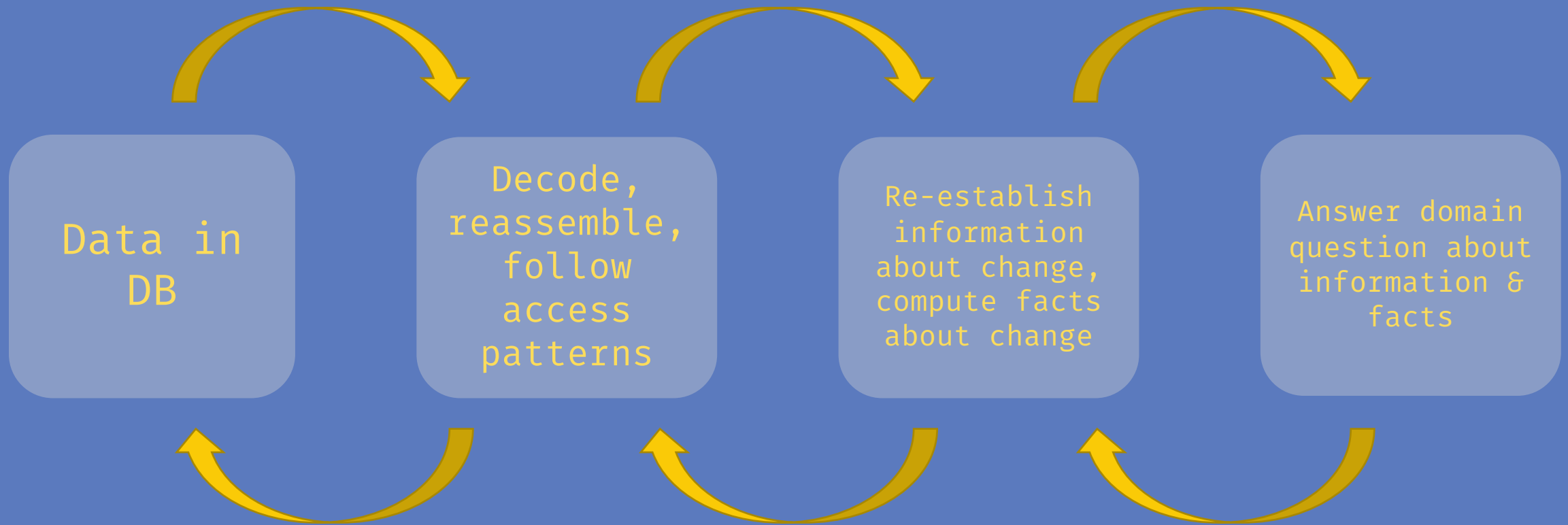
- Structured SQL code
- Functional SQL
- Domain names in code ("literate" SQL)
- Explicit meaning
- Simple (single subject) queries
- Specific solutions - don't try to build a framework

Top-Down semantic
modelling
(CASE, SOA)

Semantic Web

Get whole org. to
agree on meaning

Steps to an answer



```

WITH _customer_credits_debits
> ("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due
Days") AS
    (SELECT
        OCRD."CardCode" AS "Customer Code"
        , OCRD."CardName" AS "Customer Name"
        , (case
            when "SYSCred" != 0.0
            then "SYSCred" * -1.0
            else "SYSDeb"
            end) as "SYS Balance Due"
        , (case
            when "BalDueCred" != 0.0
            then "BalDueCred" * -1.0
            else "BalDueDeb"
            end) as "Cust Balance Due"
        , DAYS_BETWEEN(JDT1."DueDate", current_date) AS "Due Days"
    FROM
        OCRD INNER JOIN JDT1
        ON JDT1."ShortName" = OCRD."CardCode"
        AND OCRD."CardType" = 'c'
    ),
> _CUSTOMER_CREDITS_DEBITS_RANGETEXT
("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due
Range") AS
    (SELECT
        ccd."Customer Code"
        , ccd."Customer Name"
        , ccd."SYS Balance Due"
        , ccd."Cust Balance Due"
        , ddt."rangeDescription" AS "Due Range"
    FROM
        _customer_credits_debits ccd
        INNER JOIN "dueDaysText" ddt
        ON ccd."Due Days" = ddt."daysDue"
    ),
> _CUSTOMER_AGING_PIVOT
("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120
days", ">120 days") AS
    (SELECT
        "Customer Code"
        , "Customer Name"
        , SUM("SYS Balance Due") AS "Balance Due"

```

```

        , SUM(MAP ("Due Range", 'Future Remit', "Cust Balance Due", 0.0)) AS
"Future Remit"
        , SUM(MAP ("Due Range", '0-30 days', "Cust Balance Due", 0.0)) AS "0-30
days"
        , SUM(MAP ("Due Range", '31 to 60 days', "Cust Balance Due", 0.0)) AS "31
to 60 days"
        , SUM(MAP ("Due Range", '61 to 90 days', "Cust Balance Due", 0.0)) AS "61
to 90 days"
        , SUM(MAP ("Due Range", '91 to 120 days', "Cust Balance Due", 0.0)) AS
"91 to 120 days"
        , SUM(MAP ("Due Range", '121+ days', "Cust Balance Due", 0.0)) AS ">120
days"
    FROM
        _CUSTOMER_CREDITS_DEBITS_RANGETEXT
    GROUP BY
        "Customer Code", "Customer Name", "Due Range"
    HAVING
        SUM("SYS Balance Due") != 0
)
SELECT
    "Customer Code"
    , "Customer Name"
    , SUM("Balance Due") AS "Balance Due"
    , SUM("Future Remit") AS "Future Remit"
    , SUM("0-30 days") AS "0-30 days"
    , SUM("31 to 60 days") AS "31 to 60 days"
    , SUM("61 to 90 days") AS "61 to 90 days"
    , SUM("91 to 120 days") AS "91 to 120 days"
    , SUM(">120 days") AS ">120 days"
FROM
    _CUSTOMER_AGING_PIVOT
GROUP BY
    "Customer Code", "Customer Name"
ORDER BY
    "Customer Code" ;

```

```

WITH _customer_credits_debits
> ("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Days") AS

> _CUSTOMER_CREDITS_DEBITS_RANGETEXT
("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Range") AS

> _CUSTOMER_AGING_PIVOT
("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120 days", ">120 days") AS

SELECT
    "Customer Code"
  , "Customer Name"
  , SUM("Balance Due") AS "Balance Due"
  , SUM("Future Remit") AS "Future Remit"
  , SUM("0-30 days") AS "0-30 days"
  , SUM("31 to 60 days") AS "31 to 60 days"
  , SUM("61 to 90 days") AS "61 to 90 days"
  , SUM("91 to 120 days") AS "91 to 120 days"
  , SUM(">120 days") AS ">120 days"
FROM
    _CUSTOMER_AGING_PIVOT
GROUP BY
    "Customer Code", "Customer Name"
ORDER BY
    "Customer Code" ;

```

CTEs/WITH as statement local views



```

WITH _customer_credits_debits
> ("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Days") AS
(SELECT
    OCRD."CardCode" AS "Customer Code"
    , OCRD."CardName" AS "Customer Name"
    , (case
        when "SYSCred" != 0.0
        then "SYSCred" * -1.0
        else "SYSDeb"
        end) as "SYS Balance Due"
    , (case
        when "BalDueCred" != 0.0
        then "BalDueCred" * -1.0
        else "BalDueDeb"
        end) as "Cust Balance Due"
    , DAYS_BETWEEN(JDT1."DueDate", current_date) AS "Due Days"
FROM
    OCRD INNER JOIN JDT1
    ON JDT1."ShortName" = OCRD."CardCode"
    AND OCRD."CardType" = 'c'
),

> _CUSTOMER_CREDITS_DEBITS_RANGETEXT
("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Range") AS

> _CUSTOMER_AGING_PIVOT
("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120 days", ">120 days") AS

SELECT
    "Customer Code"
    , "Customer Name"
    , SUM("Balance Due") AS "Balance Due"
    , SUM("Future Remit") AS "Future Remit"
    , SUM("0-30 days") AS "0-30 days"
    , SUM("31 to 60 days") AS "31 to 60 days"
    , SUM("61 to 90 days") AS "61 to 90 days"
    , SUM("91 to 120 days") AS "91 to 120 days"
    , SUM(">120 days") AS ">120 days"
FROM
    _CUSTOMER_AGING_PIVOT
GROUP BY
    "Customer Code", "Customer Name"
ORDER BY
    "Customer Code" ;

```

CTEs/WITH as statement local views



```

WITH _customer_credits_debits
>("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Days") AS

> _CUSTOMER_CREDITS_DEBITS_RANGETEXT
("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Range") AS
    (SELECT
        ccd."Customer Code"
        , ccd."Customer Name"
        , ccd."SYS Balance Due"
        , ccd."Cust Balance Due"
        , ddt."rangeDescription" AS "Due Range"
    FROM
        _customer_credits_debits ccd
        INNER JOIN "dueDaysText" ddt
        ON ccd."Due Days" = ddt."daysDue"
    ),

> _CUSTOMER_AGING_PIVOT
("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120 days", ">120 days") AS

SELECT
    "Customer Code"
    , "Customer Name"
    , SUM("Balance Due") AS "Balance Due"
    , SUM("Future Remit") AS "Future Remit"
    , SUM("0-30 days") AS "0-30 days"
    , SUM("31 to 60 days") AS "31 to 60 days"
    , SUM("61 to 90 days") AS "61 to 90 days"
    , SUM("91 to 120 days") AS "91 to 120 days"
    , SUM(">120 days") AS ">120 days"
FROM
    _CUSTOMER_AGING_PIVOT
GROUP BY
    "Customer Code", "Customer Name"
ORDER BY
    "Customer Code" ;

```

CTEs/WITH as statement local views



```

WITH _customer_credits_debits
>("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Days") AS

>_CUSTOMER_CREDITS_DEBITS_RANGETEXT
("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Range") AS

>_CUSTOMER_AGING_PIVOT
("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120 days", ">120 days") AS
    (SELECT
        "Customer Code"
        , "Customer Name"
        , SUM("SYS Balance Due") AS "Balance Due"
        , SUM(MAP ("Due Range", 'Future Remit', "Cust Balance Due", 0.0)) AS "Future Remit"
        , SUM(MAP ("Due Range", '0-30 days', "Cust Balance Due", 0.0)) AS "0-30 days"
        , SUM(MAP ("Due Range", '31 to 60 days', "Cust Balance Due", 0.0)) AS "31 to 60 days"
        , SUM(MAP ("Due Range", '61 to 90 days', "Cust Balance Due", 0.0)) AS "61 to 90 days"
        , SUM(MAP ("Due Range", '91 to 120 days', "Cust Balance Due", 0.0)) AS "91 to 120 days"
        , SUM(MAP ("Due Range", '121+ days', "Cust Balance Due", 0.0)) AS ">120 days"
    FROM
        _CUSTOMER_CREDITS_DEBITS_RANGETEXT
    GROUP BY
        "Customer Code", "Customer Name", "Due Range"
    HAVING
        SUM("SYS Balance Due") != 0
)

SELECT
    "Customer Code"
    , "Customer Name"
    , SUM("Balance Due") AS "Balance Due"
    , SUM("Future Remit") AS "Future Remit"
    , SUM("0-30 days") AS "0-30 days"
    , SUM("31 to 60 days") AS "31 to 60 days"
    , SUM("61 to 90 days") AS "61 to 90 days"
    , SUM("91 to 120 days") AS "91 to 120 days"
    , SUM(">120 days") AS ">120 days"
FROM
    _CUSTOMER_AGING_PIVOT
GROUP BY
    "Customer Code", "Customer Name"
ORDER BY
    "Customer Code" ;

```

CTEs/WITH as statement local views



```

WITH _customer_credits_debits
> ("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Days") AS

> _CUSTOMER_CREDITS_DEBITS_RANGETEXT
("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Range") AS
    (SELECT
        ccd."Customer Code"
        , ccd."Customer Name"
        , ccd."SYS Balance Due"
        , ccd."Cust Balance Due"
        , ddt."rangeDescription" AS "Due Range"
    FROM
        _customer_credits_debits ccd
        INNER JOIN "dueDaysText" ddt
        ON ccd."Due Days" = ddt."daysDue"
    ),

> _CUSTOMER_AGING_PIVOT
("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120 days", ">120 days") AS

SELECT
    "Customer Code"
    , "Customer Name"
    , SUM("Balance Due") AS "Balance Due"
    , SUM("Future Remit") AS "Future Remit"
    , SUM("0-30 days") AS "0-30 days"
    , SUM("31 to 60 days") AS "31 to 60 days"
    , SUM("61 to 90 days") AS "61 to 90 days"
    , SUM("91 to 120 days") AS "91 to 120 days"
    , SUM(">120 days") AS ">120 days"
FROM
    _CUSTOMER_AGING_PIVOT
GROUP BY
    "Customer Code", "Customer Name"
ORDER BY
    "Customer Code" ;

```

Easy to refactor



```

WITH _customer_credits_debits
>("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Days") AS

>_CUSTOMER_CREDITS_DEBITS_RANGETEXT
("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Range") AS
    (SELECT
        "Customer Code"
      , "Customer Name"
      , "SYS Balance Due"
      , "Cust Balance Due"
      , (CASE
          WHEN ("Due Days" < 0)
          THEN 'Future Remit'
          WHEN ("Due Days" BETWEEN 0 AND 30)
          THEN '0-30 days'
          WHEN ("Due Days" BETWEEN 31 AND 60)
          THEN '31 to 90 days'
          WHEN ("Due Days" BETWEEN 61 AND 90)
          THEN '61 to 90 days'
          WHEN ("Due Days" BETWEEN 91 AND 120)
          THEN '91 to 120 days'
          WHEN ("Due Days" > 120)
          THEN '>120 days'
        END) as "Due Range"
      FROM
        _customer_credits_debits
    ),

>_CUSTOMER_AGING_PIVOT
("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120 days", ">120 days") AS

SELECT
    "Customer Code"
  , "Customer Name"
  , SUM("Balance Due") AS "Balance Due"
  , SUM("Future Remit") AS "Future Remit"
  , SUM("0-30 days") AS "0-30 days"
  , SUM("31 to 60 days") AS "31 to 60 days"
  , SUM("61 to 90 days") AS "61 to 90 days"
  , SUM("91 to 120 days") AS "91 to 120 days"
  , SUM(">120 days") AS ">120 days"
FROM
    _CUSTOMER_AGING_PIVOT
GROUP BY
    "Customer Code", "Customer Name"

```

Easy to refactor



```

WITH _customer_credits_debits
> ("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Days") AS

> _CUSTOMER_CREDITS_DEBITS_RANGETEXT
("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Range") AS
    (SELECT
        ccd."Customer Code"
        , ccd."Customer Name"
        , ccd."SYS Balance Due"
        , ccd."Cust Balance Due"
        , ddt."rangeDescription" AS "Due Range"
    FROM
        _customer_credits_debits ccd
        INNER JOIN "dueDaysText" ddt
        ON ccd."Due Days" = ddt."daysDue"
    )

select * from _CUSTOMER_CREDITS_DEBITS_RANGETEXT

> _CUSTOMER_AGING_PIVOT
("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120 days", ">120 days") AS

SELECT
    "Customer Code"
    , "Customer Name"
    , SUM("Balance Due") AS "Balance Due"
    , SUM("Future Remit") AS "Future Remit"
    , SUM("0-30 days") AS "0-30 days"
    , SUM("31 to 60 days") AS "31 to 60 days"
    , SUM("61 to 90 days") AS "61 to 90 days"
    , SUM("91 to 120 days") AS "91 to 120 days"
    , SUM(">120 days") AS ">120 days"
FROM
    _CUSTOMER_AGING_PIVOT
GROUP BY
    "Customer Code", "Customer Name"
ORDER BY
    "Customer Code" ;

```

Easy to check and mock



```

create column table _mock_customer_credits_debits_rangetext as
(
    WITH _customer_credits_debits
    >("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Days") AS
    _CUSTOMER_CREDITS_DEBITS_RANGETEXT
    ("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Range") AS
    (SELECT
        "Customer Code"
        , "Customer Name"
        , "SYS Balance Due"
        , "Cust Balance Due"
        , (CASE
            WHEN ("Due Days" < 0)
            THEN 'Future Remit'
            WHEN ("Due Days" BETWEEN 0 AND 30)
            THEN '0-30 days'
            WHEN ("Due Days" BETWEEN 31 AND 60)
            THEN '31 to 90 days'
            WHEN ("Due Days" BETWEEN 61 AND 90)
            THEN '61 to 90 days'
            WHEN ("Due Days" BETWEEN 91 AND 120)
            THEN '91 to 120 days'
            WHEN ("Due Days" > 120)
            THEN '>120 days'
            END) as "Due Range"
        FROM
            _mock_customer_credits_debits_rangetext
        )
    select * from _CUSTOMER_CREDITS_DEBITS_RANGETEXT)

```

```

> _CUSTOMER_AGING_PIVOT
("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120 days", ">120 days") AS

```

```

SELECT
    "Customer Code"
    , "Customer Name"
    , SUM("Balance Due") AS "Balance Due"
    , SUM("Future Remit") AS "Future Remit"
    , SUM("0-30 days") AS "0-30 days"
    , SUM("31 to 60 days") AS "31 to 60 days"
    , SUM("61 to 90 days") AS "61 to 90 days"
    , SUM("91 to 120 days") AS "91 to 120 days"
    , SUM(">120 days") AS ">120 days"

```

```

FROM
    CUSTOMER_AGING_PIVOT

```

Easy to check and mock



```

WITH _customer_credits_debits
> ("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Days") AS

> _CUSTOMER_CREDITS_DEBITS_RANGETEXT
("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Range") AS
    (SELECT
        "Customer Code"
      , "Customer Name"
      , "SYS Balance Due"
      , "Cust Balance Due"
      , "Due Range"
    FROM
        _mock_customer_credits_debits_rangetext
    )

select * from _CUSTOMER_CREDITS_DEBITS_RANGETEXT

> _CUSTOMER_AGING_PIVOT
("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120 days", ">120 days") AS

SELECT
    "Customer Code"
  , "Customer Name"
  , SUM("Balance Due") AS "Balance Due"
  , SUM("Future Remit") AS "Future Remit"
  , SUM("0-30 days") AS "0-30 days"
  , SUM("31 to 60 days") AS "31 to 60 days"
  , SUM("61 to 90 days") AS "61 to 90 days"
  , SUM("91 to 120 days") AS "91 to 120 days"
  , SUM(">120 days") AS ">120 days"
FROM
    _CUSTOMER_AGING_PIVOT
GROUP BY
    "Customer Code", "Customer Name"
ORDER BY
    "Customer Code" ;

```

Easy to check and mock



```

create view _customer_credits_debits
("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Days")
as
(SELECT
    OCRD."CardCode" AS "Customer Code"
    , OCRD."CardName" AS "Customer Name"
    , (case
        when "SYSCred" != 0.0
        then "SYSCred" * -1.0
        else "SYSDeb"
        end) as "SYS Balance Due"
    , (case
        when "BalDueCred" != 0.0
        then "BalDueCred" * -1.0
        else "BalDueDeb"
        end) as "Cust Balance Due"
    , DAYS_BETWEEN(JDT1."DueDate", current_date) AS "Due Days"
FROM
    OCRD INNER JOIN JDT1
    ON JDT1."ShortName" = OCRD."CardCode"
    AND OCRD."CardType" = 'c'
)

> _CUSTOMER_CREDITS_DEBITS_RANGETEXT
("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Range") AS
(SELECT
    "Customer Code"
    , "Customer Name"
    , "SYS Balance Due"
    , "Cust Balance Due"
    , "Due Range"
FROM
    _mock_customer_credits_debits_rangetext
)

> _CUSTOMER_AGING_PIVOT
("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120 days", ">120 days") AS

SELECT
    "Customer Code"
    , "Customer Name"
    , SUM("Balance Due") AS "Balance Due"
    , SUM("Future Remit") AS "Future Remit"
    , SUM("0-30 days") AS "0-30 days"
    , SUM("31 to 60 days") AS "31 to 60 days"

```

Make reusable by turning into views



```
WITH _CUSTOMER_CREDITS_DEBITS_RANGETEXT
>("Customer Code", "Customer Name", "SYS Balance Due", "Cust Balance Due", "Due Range") AS

_CUSTOMER_AGING_PIVOT
>("Customer Code", "Customer Name", "Balance Due"
, "Future Remit", "0-30 days", "31 to 60 days", "61 to 90 days", "91 to 120 days", ">120 days") AS

SELECT
    "Customer Code"
    , "Customer Name"
    , SUM("Balance Due") AS "Balance Due"
    , SUM("Future Remit") AS "Future Remit"
    , SUM("0-30 days") AS "0-30 days"
    , SUM("31 to 60 days") AS "31 to 60 days"
    , SUM("61 to 90 days") AS "61 to 90 days"
    , SUM("91 to 120 days") AS "91 to 120 days"
    , SUM(">120 days") AS ">120 days"
FROM
_CUSTOMER_AGING_PIVOT
GROUP BY
    "Customer Code", "Customer Name"
ORDER BY
    "Customer Code" ;
```

Make reusable by turning into views



MONSTER_BUSINESS_ITEMS

CLIENT	ORDER_NUMBER	ITEM_NUMBER	ITEM_INFO
007	0000000001	00000019	info for item 1/19
007	0000000001	00000018	info for item 1/18
007	0000000001	00000017	info for item 1/17
007	0000000001	00000016	info for item 1/16
007	0000000001	00000015	info for item 1/15
007	0000000001	00000014	info for item 1/14
007	0000000001	00000012	info for item 1/12
007	0000000001	00000011	info for item 1/11
007	0000000001	00000010	info for item 1/10
007	0000000001	00000009	info for item 1/9
007	0000000001	00000007	info for item 1/7
007	0000000001	00000006	info for item 1/6
007	0000000001	00000005	info for item 1/5
007	0000000001	00000004	info for item 1/4
007	0000000001	00000003	info for item 1/3
007	0000000001	00000002	info for item 1/2
007	0000000001	00000001	info for item 1/1
007	0000000001	00000000	info for item 1/0
007	0000000001	00000008	info for item 1/8
007	0000000001	00000013	info for item 1/13
[...]			

MONSTER_BUSINESS_ITEM_DATA

CLIENT	ORDER_NUMBER	ITEM_NUMBER	PAYMENT_TERMS
007	0000000001	00000000	DEFAULT1
007	0000000001	00000013	SPECIAL
007	0000000001	00000008	SPECIAL
007	0000000002	00000000	DEFAULT2
007	0000000002	00000017	SPECIAL
007	0000000003	00000000	DEFAULT0
[...]			

Access logic:

All business items use the DEFAULT payment terms (special entry in MONSTER_BUSINESS_ITEM_DATA with "magic" ITEM_NUMBER = 00000000), except if there is an entry for a specific CLIENT/ORDER_NUMBER/ITEM_NUMBER.

Package special access pattern per use case – don't recreate the access layer of the framework/app



Implicit Default Values

```

create view "MONSTER_BUSINESS_ITEMS_PAY_TERMS" as
(select
    bi."CLIENT", bi."ORDER_NUMBER", bi."ITEM_NUMBER"
    , bid."PAYMENT_TERMS"          as "PAY_TERMS_BY_SIMPLE_JOIN"
    , MAP (bid."PAYMENT_TERMS"
        , NULL
        , FIRST_VALUE(bid."PAYMENT_TERMS")
        OVER (PARTITION BY
            bi."CLIENT", bi."ORDER_NUMBER"
            ORDER BY
            bi."CLIENT", bi."ORDER_NUMBER", bi."ITEM_NUMBER" ASC
            NULLS LAST)
        , bid."PAYMENT_TERMS" ) as "PAY_TERMS_WITH_DEFAULT_FALLBACK"
from
    monster_business_items      bi
left outer join monster_business_item_data bid
on (bi."CLIENT", bi."ORDER_NUMBER", bi."ITEM_NUMBER") =
    (bid."CLIENT", bid."ORDER_NUMBER", bid."ITEM_NUMBER")
);

select
    "CLIENT", "ORDER_NUMBER", "ITEM_NUMBER"
    , "PAY_TERMS_BY_SIMPLE_JOIN"
    , "PAY_TERMS_WITH_DEFAULT_FALLBACK"
from
    monster_business_items_pay_terms
where
    "CLIENT" = '007'
and "ORDER_NUMBER" = ABAP_NUMC(1, 10)
and "ITEM_NUMBER" = ABAP_NUMC(3, 8);

```

Package special access pattern per use case – don't recreate the access layer of the framework/app



Implicit Default Values

Other options to keep code understandable and untangled

- Views on views (layered views, "VDM")
- Stackable table functions
- Late generalisation
 - Dynamic SQL is often very inefficient and hard to understand
 - Writing a SQL generator for the case (in the application) is nearly always the better option

Stackable table functions

```
create column table sales
(sales_id bigint primary key
, cust_id bigint not null
, sales_date date not null);
```

```
insert into sales values (1, 2, '20200128');
insert into sales values (2, 2, '20190112');
insert into sales values (3, 1, '20200405');
insert into sales values (4, 4, '20200315');
```

```
/* ---- */
```

```
create column table customers
(cust_id bigint primary key
, name nvarchar(256) not null)
```

```
insert into customers values (1, 'Lars');
insert into customers values (2, 'Mary');
insert into customers values (3, 'Carlos');
insert into customers values (4, 'Angela');
```

```
/* ---- */
```

```
create function active_customers
(IN _customers table (CUST_ID BIGINT))
```

```
returns table (CUST_ID BIGINT)
```

```
as
```

```
begin
```

```
    return select c.CUST_ID
           from :_customers c
           where
               exists (SELECT *
                      FROM sales s
                      WHERE s.CUST_ID = c.CUST_ID
                      AND days_between
                          (s.sales_date, current_utcdate) < 60 );
```

```
end;
```

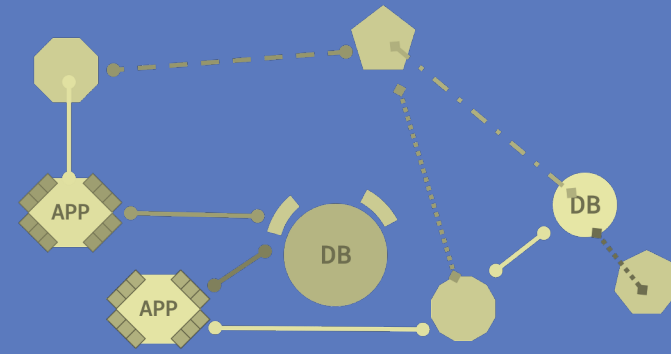
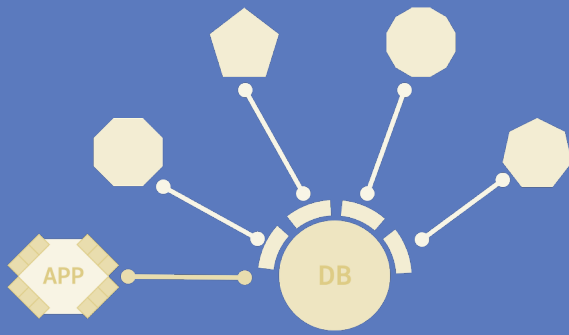
```
/* ---- */
```

```
do begin
```

```
    all_custs = select cust_id
                from customers;
```

```
    active_custs = select cust_id
                   from active_customers(:all_custs);
    select * from :active_custs;
```

```
end;
```



SQL

- Schema-on-write/Meaning-on-read
- Conceptual gaps in modelling
- SQL language

Devs/Orgs

**SQL
Training**

**Legacy
Systems
& Culture**

**Tooling
support**

**DB tech /
usage**

Options

- Structured SQL code
- Functional SQL
- Domain names in code ("literate" SQL)
- Explicit meaning
- Simple (single subject) queries
- Specific solutions - don't try to build a framework





LE FIN

Thanks for your attention.

If you liked the presentation, you might also like
my blog on <https://lbreddemann.org>

I'm available to work with you and your team.

Get in touch with me via
lars@dataprocessinsights.com.au