SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

Mehr als eine Abfragesprache: SQL im 21. Jahrhundert

@MarkusWinand • @ModernSQL

http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf

Mehr als eine Abfragesprache: SQL im 21. Jahrhundert

@MarkusWinand • @ModernSQL

SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf

SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

we PRESE

SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

D. Chamberlin

ond F. Boyce

earch Laboratory

Jose, California

1974

1992

SQL-92 — Tied to the Relational Idea

(Second Informal Review Draft) ISO/IEC 9075:1992, Database
Language SQL- July 30, 1992

SQL-92 — Tied to the Relational Idea

Relational Data Model

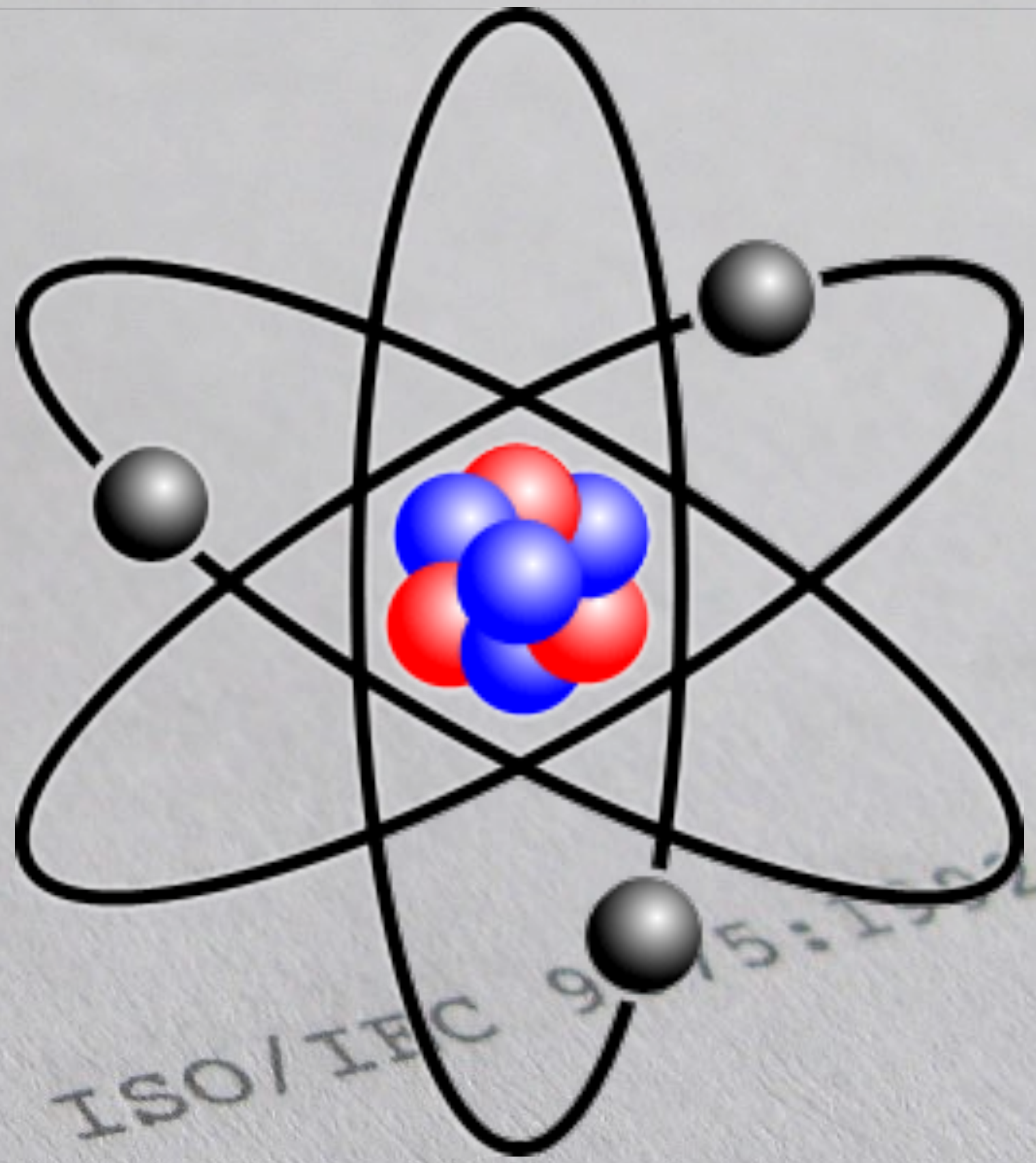
- “Atomic” types (domain)

(Second Informal Review Draft) ISO/IEC 9075:1992, Database
Language SQL- July 30, 1992

SQL-92 — Tied to the Relational Idea

Relational Data Model

- “Atomic” types (domain)



(Second Informal Review Draft) ISO/IEC 9075:1992, Database Language SQL- July 30, 1992

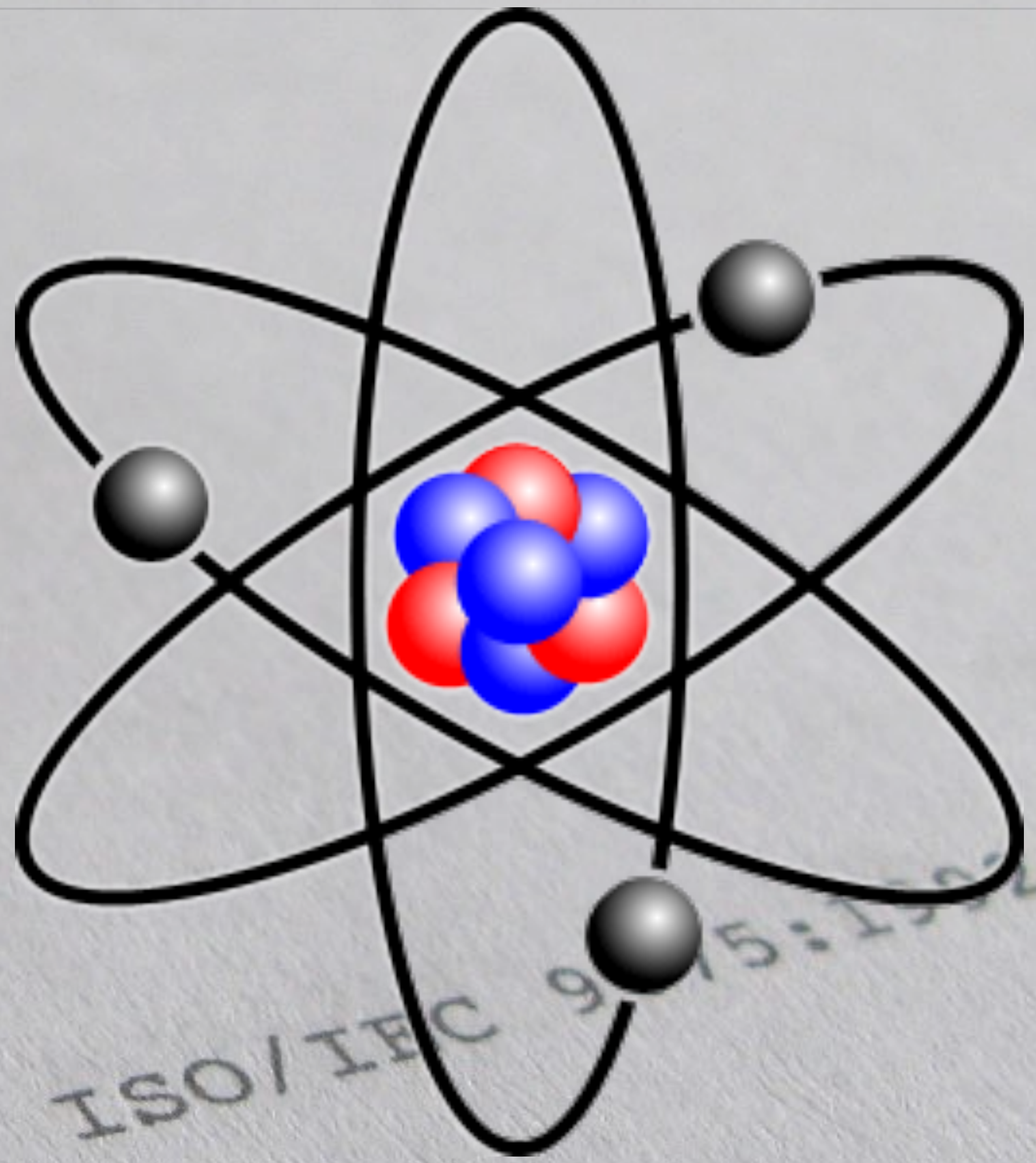
Atom image: https://commons.wikimedia.org/wiki/File:Stylised_atom_with_three_Bohr_model_orbits_and_stylised_nucleus.png

SQL-92 — Tied to the Relational Idea

Relational Data Model

- “Atomic” types (domain)

A	B	C



Atom image: https://commons.wikimedia.org/wiki/File:Stylised_atom_with_three_Bohr_model_orbits_and_stylised_nucleus.png

SQL-92 — Tied to the Relational Idea

Relational Data Model

- “Atomic” types (domain)

A	B	C
⚛	⚛	⚛
⚛	⚛	
⚛	⚛	⚛

(S Informal Review Draft) ISO/IEC 9075:1992, Database
Languag SQL- July 30, 1992

SQL-92 — Tied to the Relational Idea

Relational Data Model

- “Atomic” types (domain)
- Schema independent of processing purposes
  - “Normalization”

A	B	C
⚛	⚛	⚛
⚛	⚛	
⚛	⚛	⚛

(S Informal Review Draft) ISO/IEC 9075:1992, Database
Languag SQL- July 30, 1992

SQL-92 — Tied to the Relational Idea

Relational Data Model

- "Atomic" types (domain)
- Schema independent of processing purposes
  - "Normalization"

A	B	C
⚛	⚛	⚛
⚛	⚛	
⚛	⚛	⚛

C	D
⚛	⚛
⚛	⚛

B	E
⚛	⚛
⚛	⚛

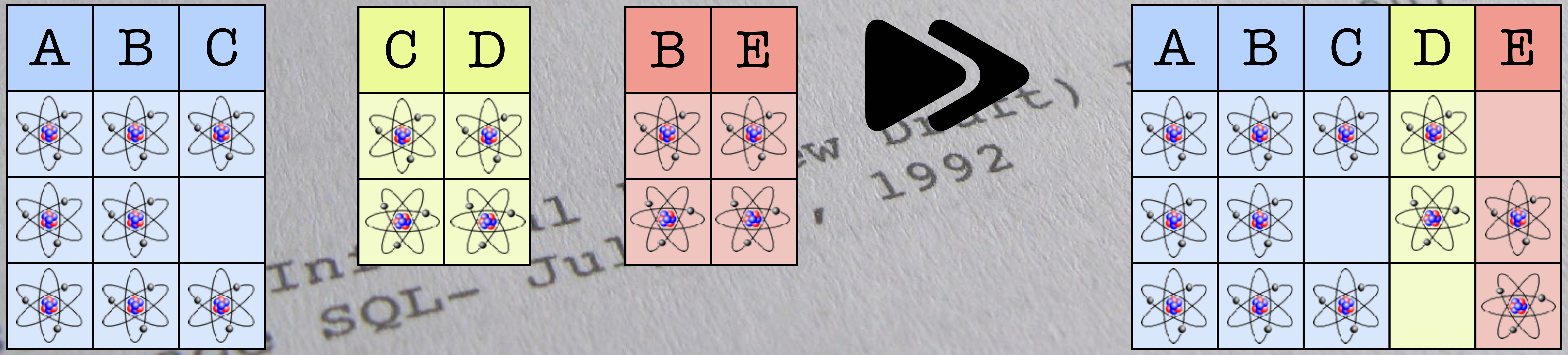
SQL-92 — Tied to the Relational Idea

Relational Data Model

- “Atomic” types (domain)
- Schema independent of processing purposes
  - “Normalization”

Relational Operations

- Transform data for each particular processing purposes
  - JOIN, UNION, nesting, …



A	B	C
⚛	⚛	⚛
⚛	⚛	
⚛	⚛	⚛

C	D
⚛	⚛
⚛	⚛

B	E
⚛	⚛
⚛	⚛

A	B	C	D	E
⚛	⚛	⚛	⚛	
⚛	⚛		⚛	⚛
⚛	⚛	⚛		⚛

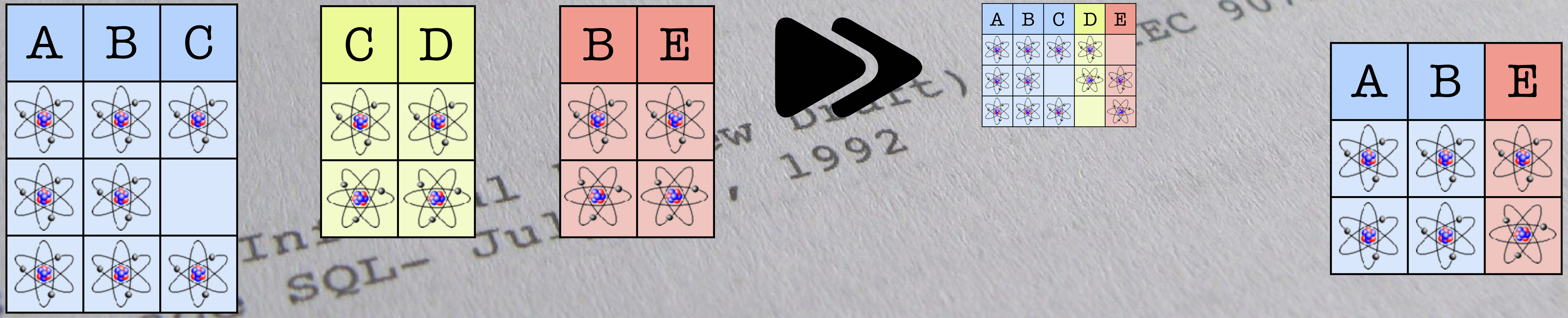
SQL-92 — Tied to the Relational Idea

Relational Data Model

- “Atomic” types (domain)
- Schema independent of processing purposes
  - “Normalization”

Relational Operations

- Transform data for each particular processing purposes
  - JOIN, UNION, nesting, …



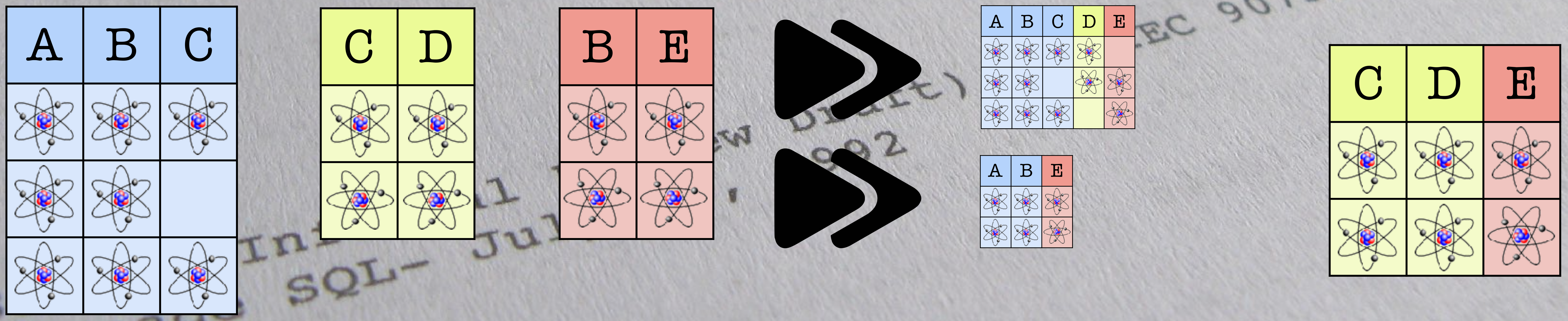
SQL-92 — Tied to the Relational Idea

Relational Data Model

- “Atomic” types (domain)
- Schema independent of processing purposes
  - “Normalization”

Relational Operations

- Transform data for each particular processing purposes
  - JOIN, UNION, nesting, …



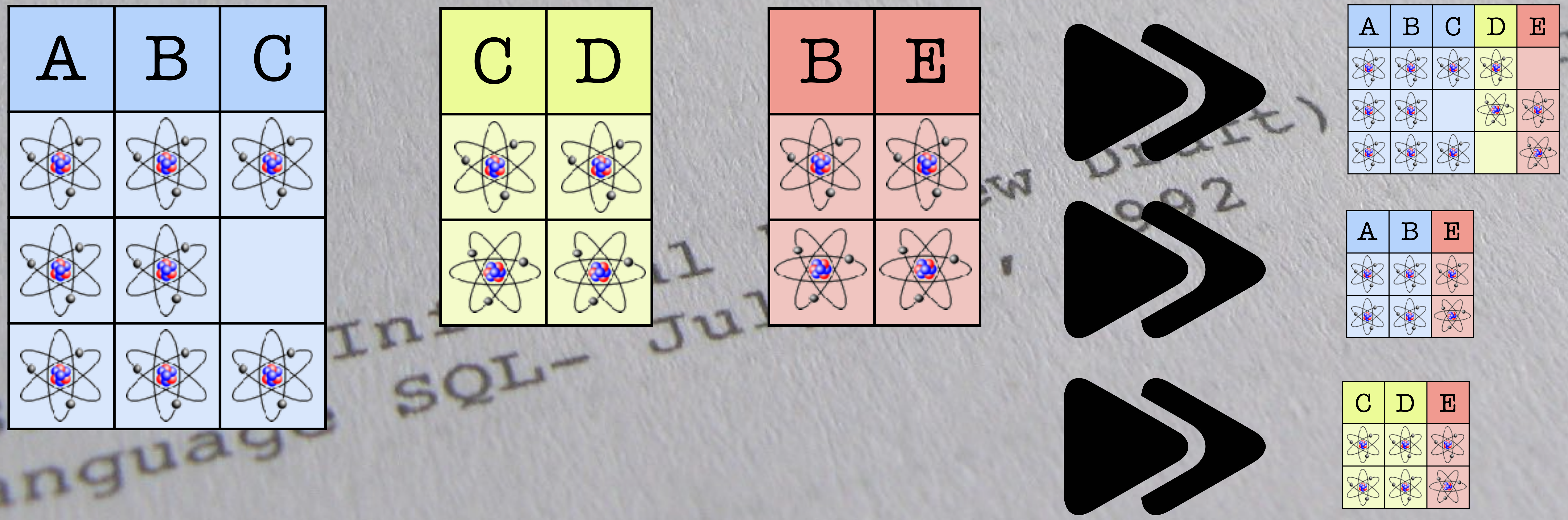
SQL-92 — Tied to the Relational Idea

Relational Data Model

- “Atomic” types (domain)
- Schema independent of processing purposes
  - “Normalization”

Relational Operations

- Transform data for each particular processing purposes
  - JOIN, UNION, nesting, …



1992 ⏩ 1999

SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

D. Chamberlin

F. Boyce

Research Laboratory

Jose, California

Whitemarsh
Information Systems Corporation

Great News,
The Relational Data Model is Dead!

https://www.wiscorp.com/DBMS_-_GreatNews-TheRelationalModelIsDead_-_paper_-_sam.pdf

SQL:1999 — Escaping the Relational Cage

> To say that these SQL:1999 extensions are mere "extended interpretations" of the relational data model is like saying that an intercontinental ballistic missile is merely an "extended interpretation" of a spear.

https://www.wiscorp.com/DBMS_-_GreatNews-TheRelationalModelIsDead_-_paper_-_sam.pdf

SQL:1999 — Escaping the Relational Cage

To say that these SQL:1999 extensions are mere "extended interpretations" of the relational data model is like saying that an intercontinental ballistic missile is merely an "extended interpretation" of a spear.

With SQL/99 you can get the best of both worlds and
of course, you can get the worst of both worlds.
It's up to the database practitioners to do the right thing.

https://www.wiscorp.com/DBMS_-_GreatNews-TheRelationalModelIsDead_-_paper_-_sam.pdf

SQL:1999 — Escaping the Relational Cage

Whitemarsh
Information Systems Corporation

Great News,
The Relational Data Model is Dead!

SQL:1999 — Escaping the Relational Cage

Relational Model?

Whitemarsh
Information Systems Corporation

Great News,
The Relational Data Model is Dead!

SQL:1999 — Escaping the Relational Cage

Whitemarsh
Information Systems Corporation

Great News,
The Relational Data Model is Dead!

Relational Model?

I was as confused as anyone else

Date on Database: Writings 2000-2006

Chris Date

SQL:1999 — Escaping the Relational Cage

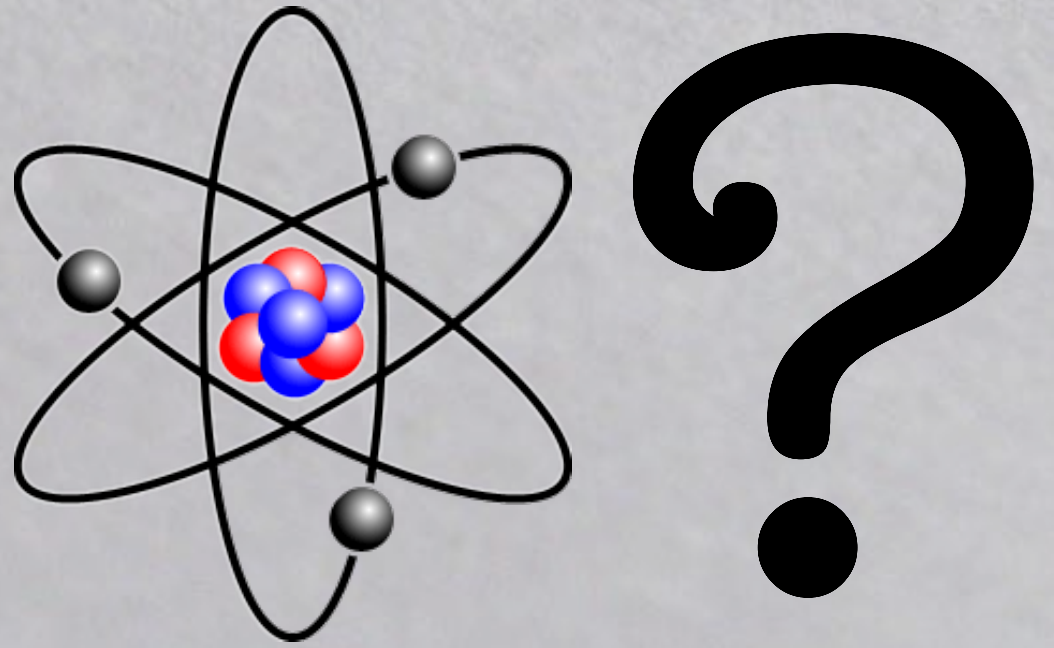
Wintermarsh

Information Systems Corporation

Great News,

The Relational Data Model is Dead!

Relational Model?



I was as confused as anyone else

Date on Database: Writings 2000-2006

Chris Date

SQL:1999 — Escaping the Relational Cage

Whitemarsh
Information Systems Corporation

Great News,
The Relational Data Model is Dead!

Relational Model?

I was as confused as anyone else ⚛?
By the early 1990s, however,
I'd seen the light

Date on Database: Writings 2000-2006

Chris Date

SQL:1999 — Escaping the Relational Cage

Great News,
The Relational Data Model is Dead!

Relational Model?

I was as confused as anyone else ⚛?
By the early 1990s, however,
I'd seen the light

Domains Can Contain Anything!

Date on Database: Writings 2000-2006

Chris Date

SQL:1999 — Escaping the Relational Cage

Relational Model?

- Introduced rich types

Great News,
The Relational Data Model is Dead!

> I was as confused as anyone else ⚛?
> By the early 1990s, however,
> I'd seen the light
>
> Domains Can Contain Anything!

Date on Database: Writings 2000-2006

Chris Date

SQL:1999 — Escaping the Relational Cage

Relational Model?

- Introduced rich types

A
⚛
⚛
⚛

I was as confused as anyone else ⚛?
By the early 1990s, however,
I'd seen the light

Domains Can Contain Anything!

Date on Database: Writings 2000-2006

Chris Date

SQL:1999 — Escaping the Relational Cage

Relational Model?

- Introduced rich types
  - arrays

A	B
⚛	[⚛,⚛]
⚛	[⚛]
⚛	[]

> I was as confused as anyone else ⚛?
> By the early 1990s, however,
> I'd seen the light
>
> Domains Can Contain Anything!

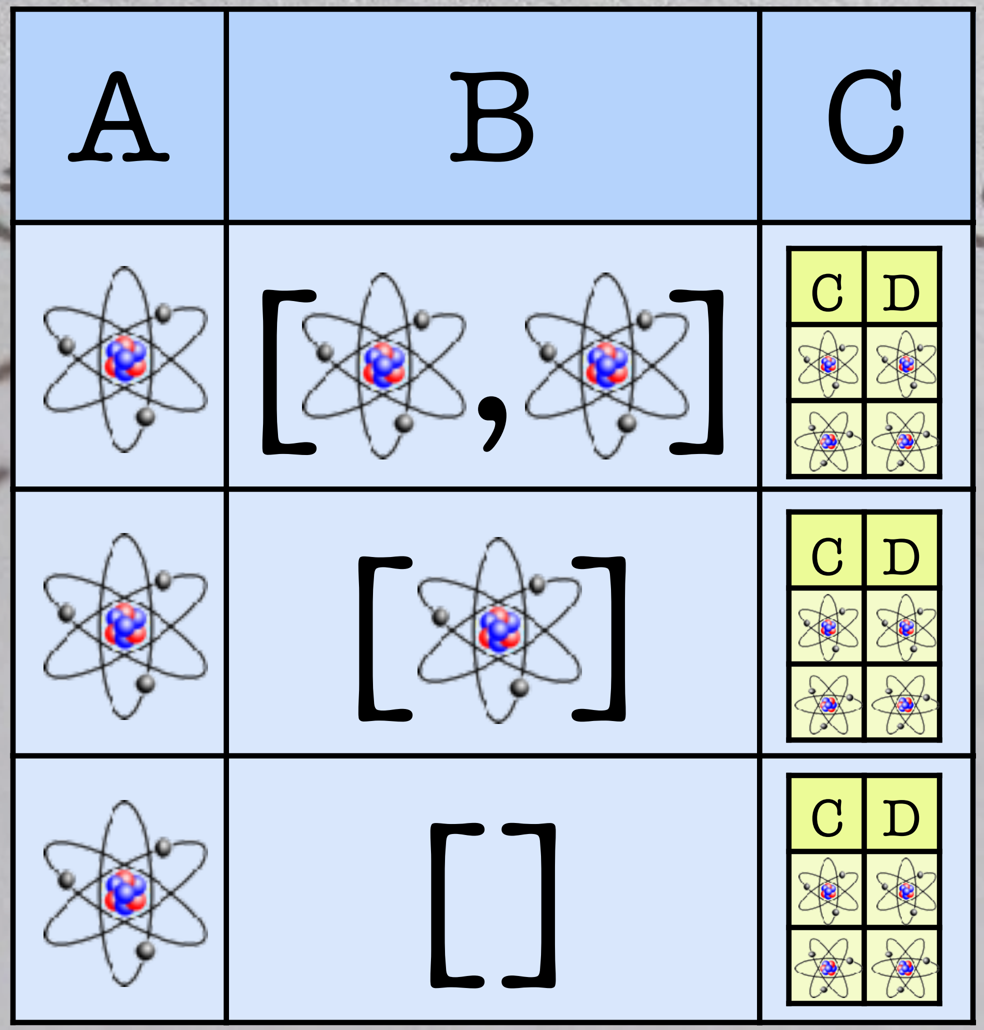
Date on Database: Writings 2000-2006

Chris Date

SQL:1999 — Escaping the Relational Cage

Relational Model?

- Introduced rich types
  - arrays
  - Nested tables (multiset)



I was as confused as anyone else ⚛?
By the early 1990s, however,
I'd seen the light

Domains Can Contain Anything!

Date on Database: Writings 2000-2006

Chris Date

SQL:1999 — Escaping the Relational Cage

Relational Model?

- Introduced rich types
  - arrays
  - Nested tables (multiset)
  - composite types (objects)

A	B	C	D
⚛	[⚛, ⚛]	C D ⚛ ⚛ ⚛ ⚛	{x: ⚛, y: ⚛}
⚛	[⚛]	C D ⚛ ⚛ ⚛ ⚛	{x: ⚛, y: ⚛}
⚛	[]	C D ⚛ ⚛ ⚛ ⚛	{x: ⚛, y: ⚛}

I was as confused as anyone else ⚛?
By the early 1990s, however,
I'd seen the light

Domains Can Contain *Anything*!

Date on Database: Writings 2000-2006

Chris Date

SQL:1999 — Escaping the Relational Cage

Relational Model?

- Introduced rich types
  - arrays
  - Nested tables (multiset)
  - composite types (objects)

Non-Relational Operations

- Introduced recursive queries that process their own output
  - Transitive closure

I was as confused as anyone else ⚛?
By the early 1990s, however,
I'd seen the light

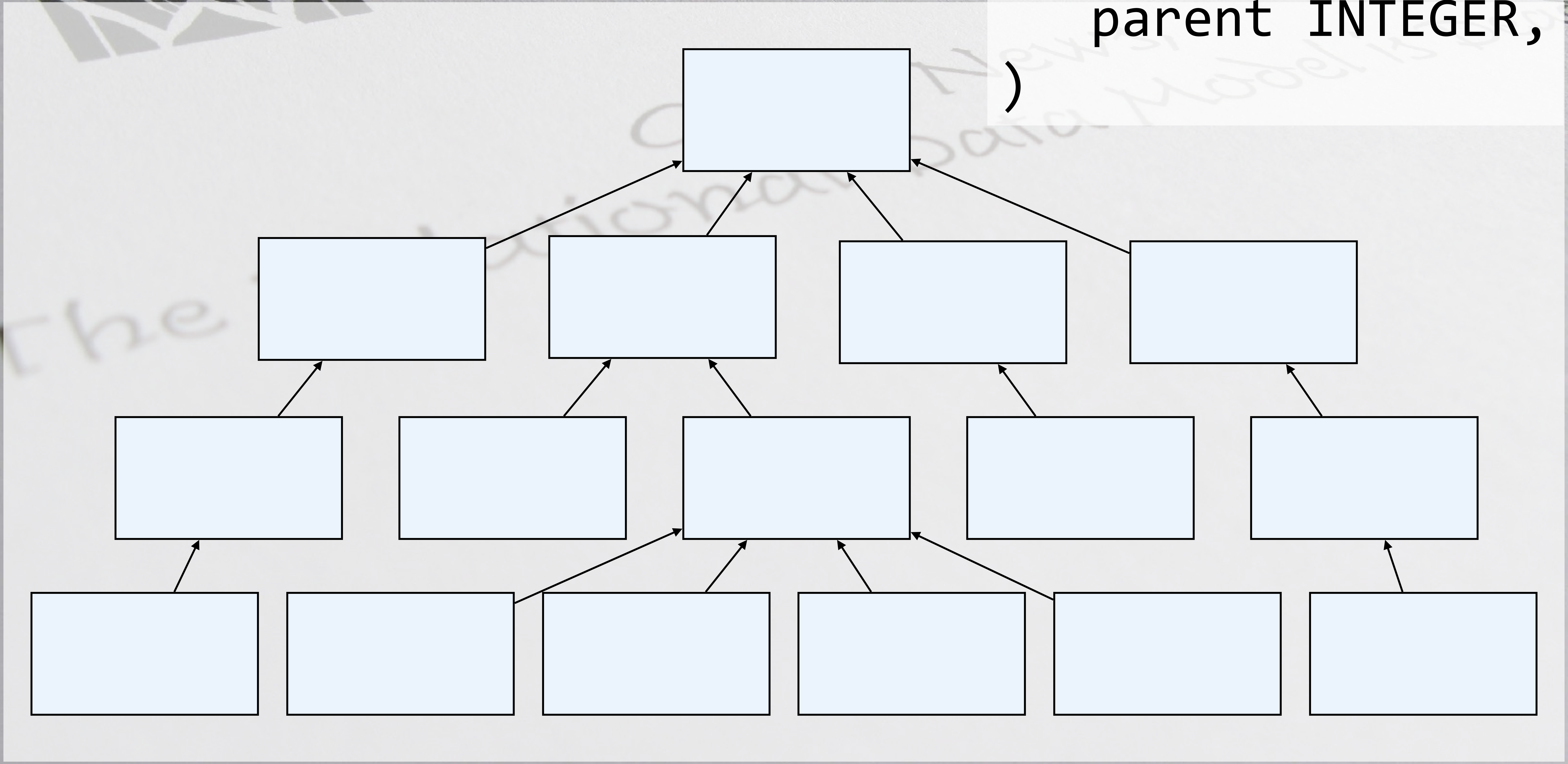
Domains Can Contain Anything!

Date on Database: Writings 2000-2006

Chris Date

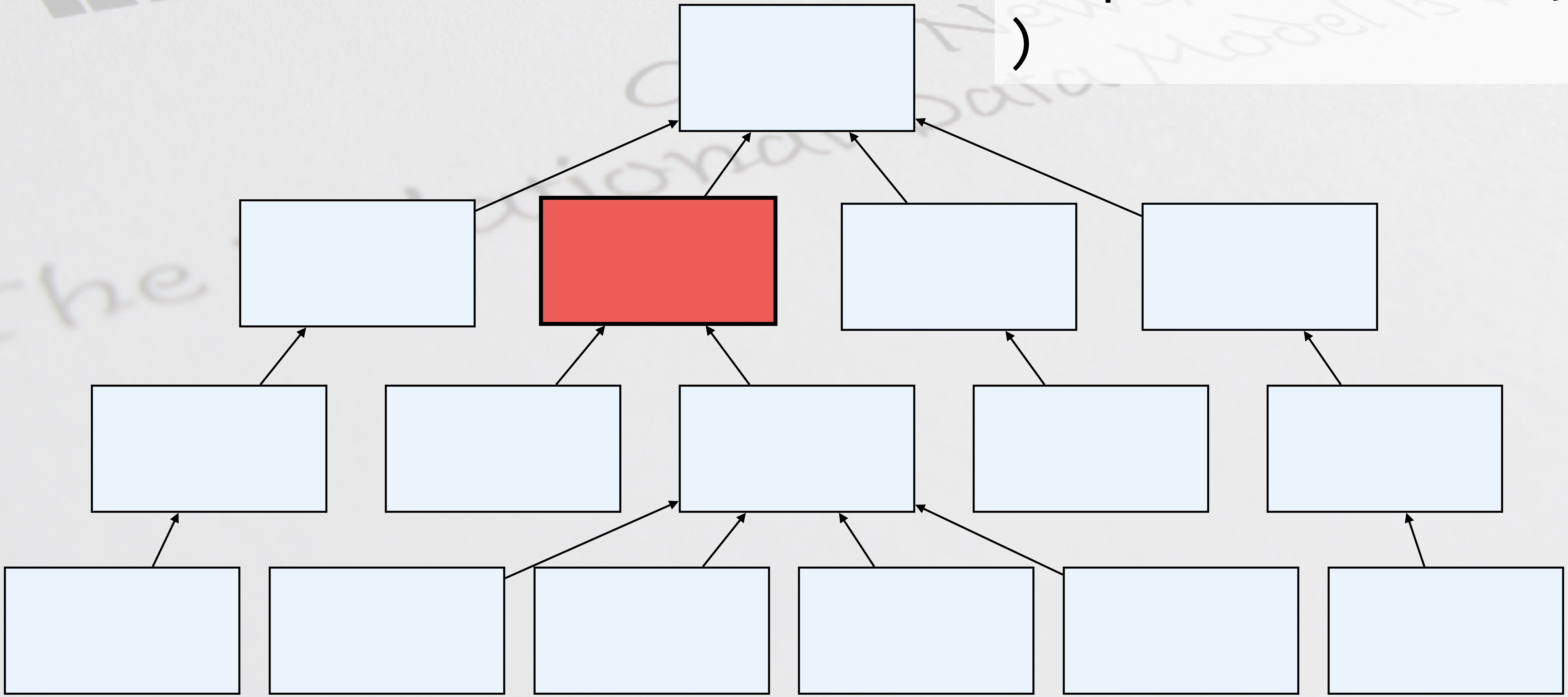
SQL:1999 — Recursion

```
CREATE TABLE t (
   id     INTEGER,
   parent INTEGER,
)
```



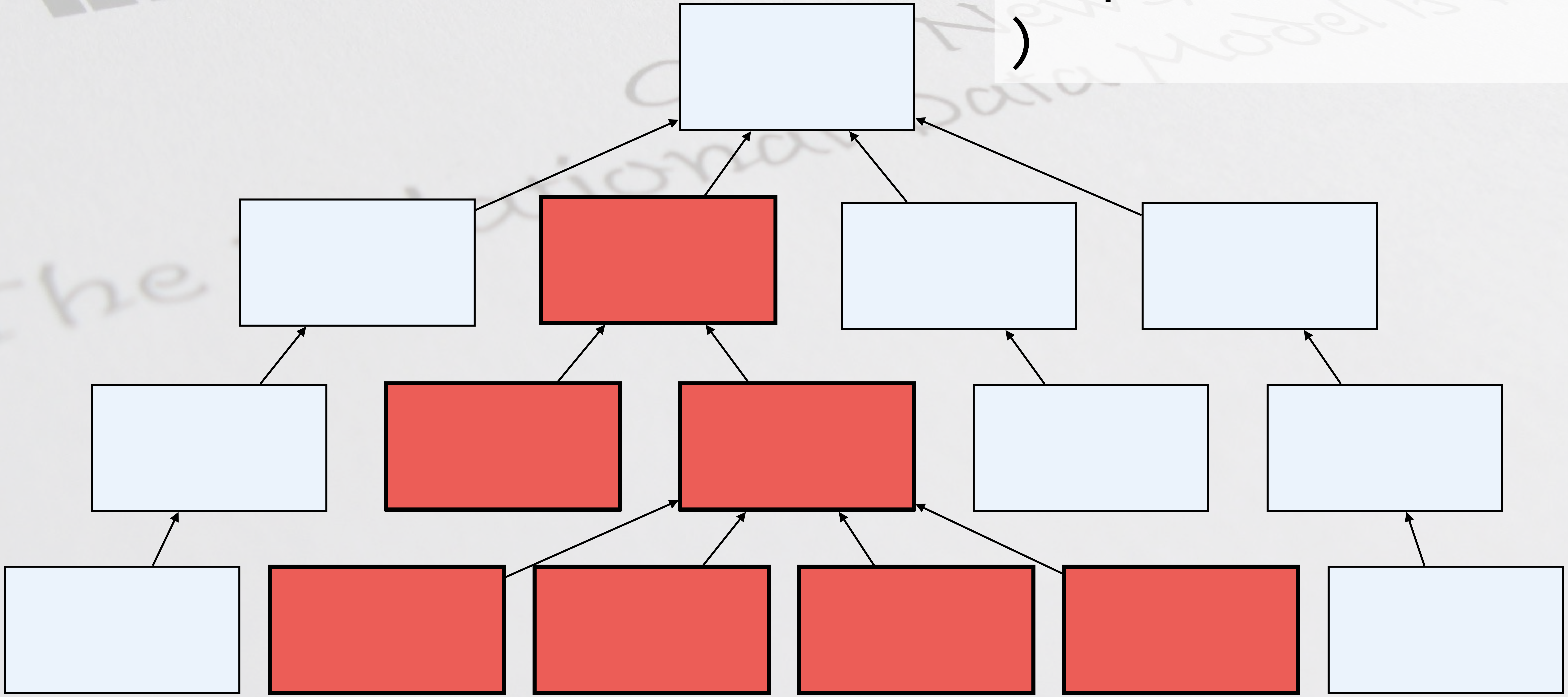
SQL:1999 — Recursion

```
CREATE TABLE t (
    id     INTEGER,
    parent INTEGER,
)
```

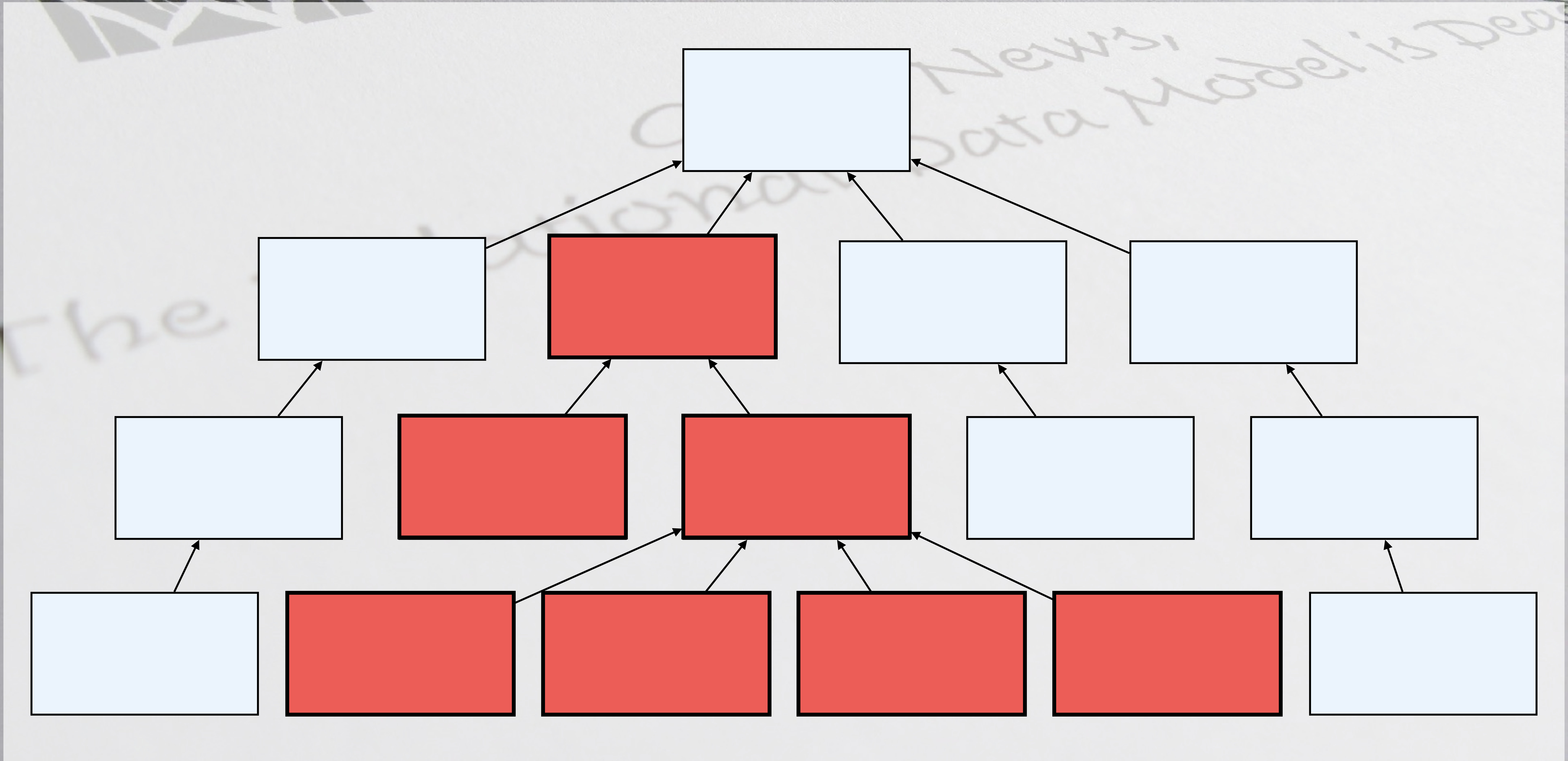


SQL:1999 — Recursion

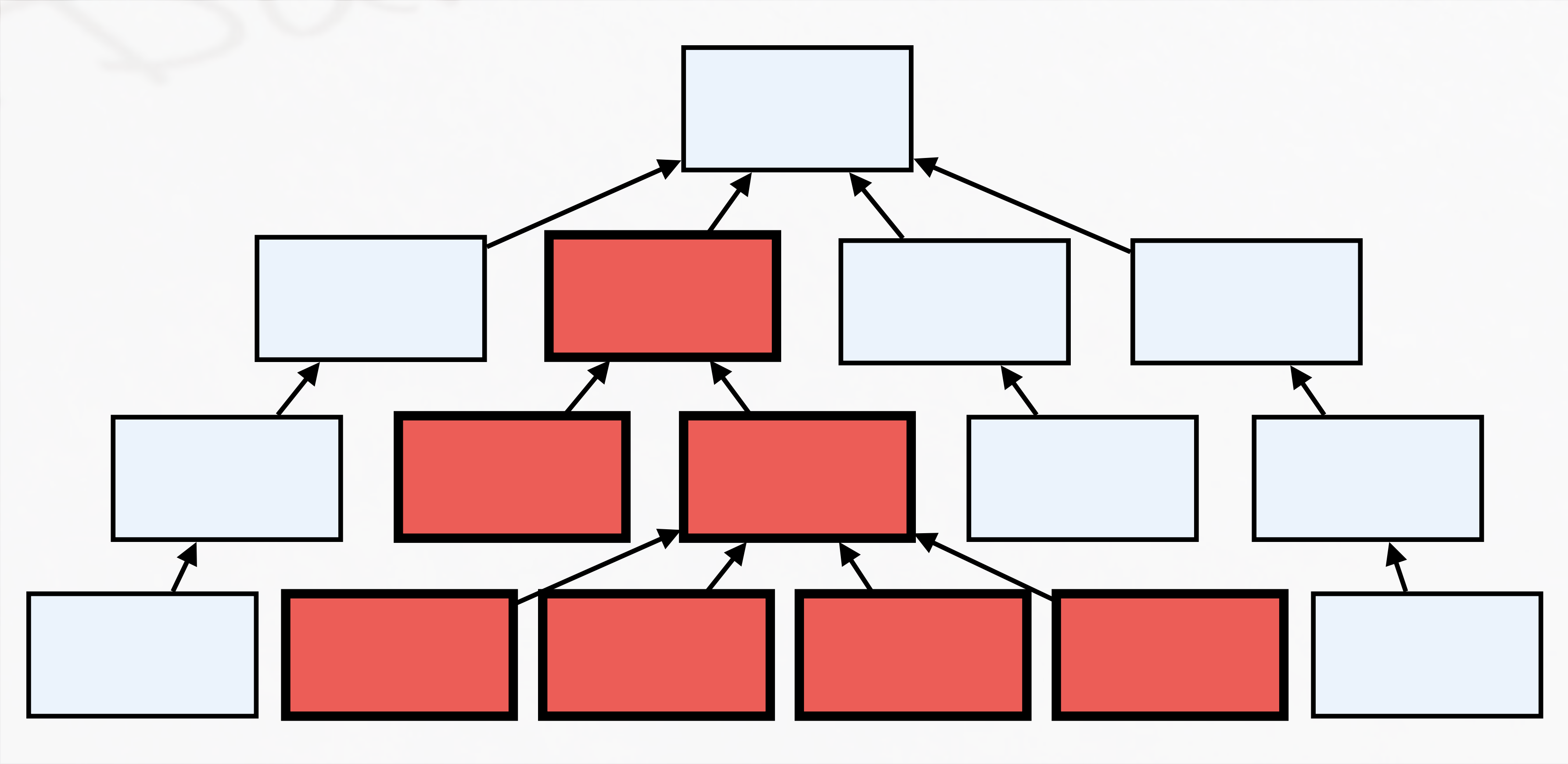
```
CREATE TABLE t (
   id     INTEGER,
   parent INTEGER,
)
```



SQL:1999 — Recursion

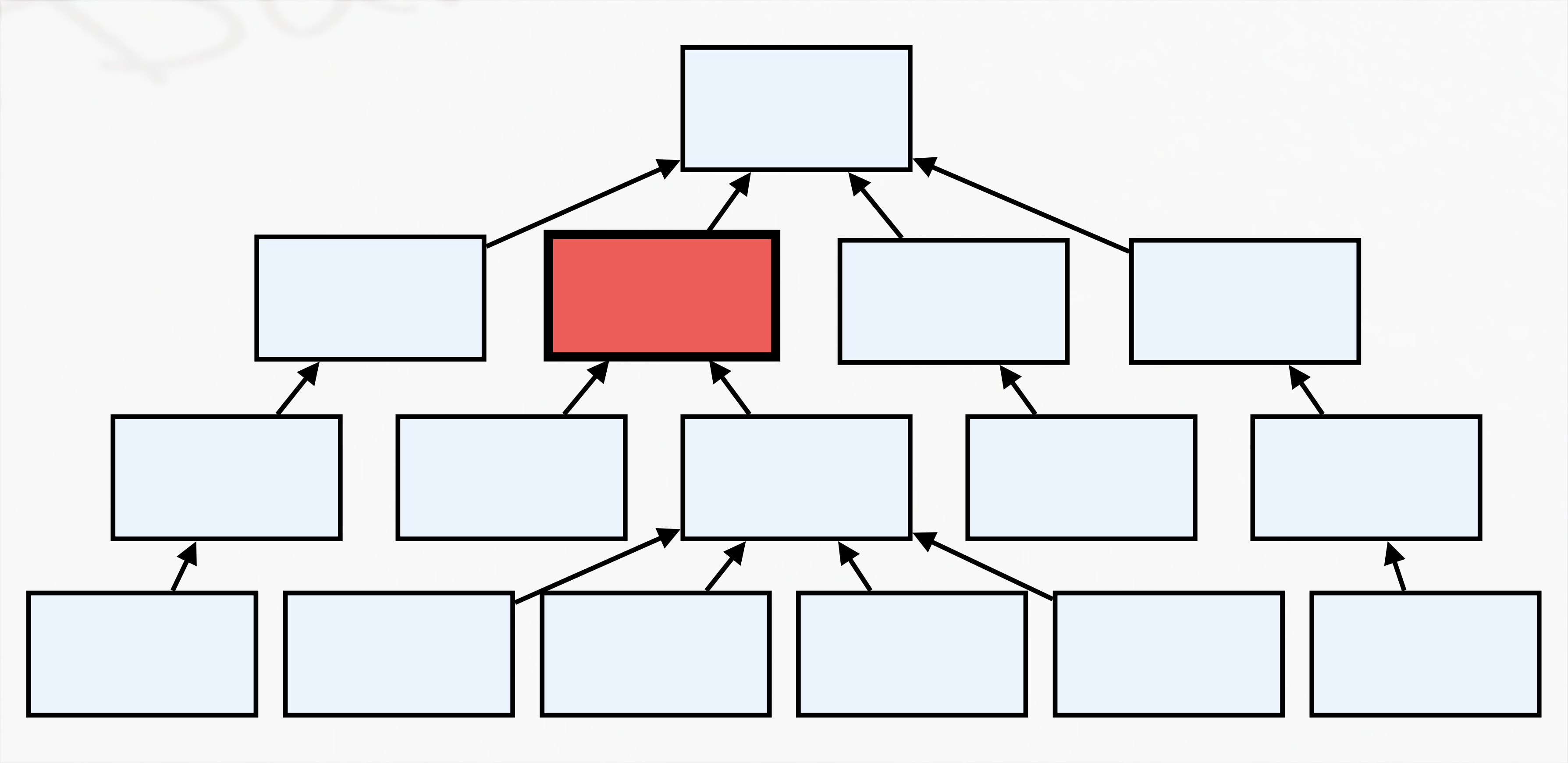


SQL:1999 — Recursion



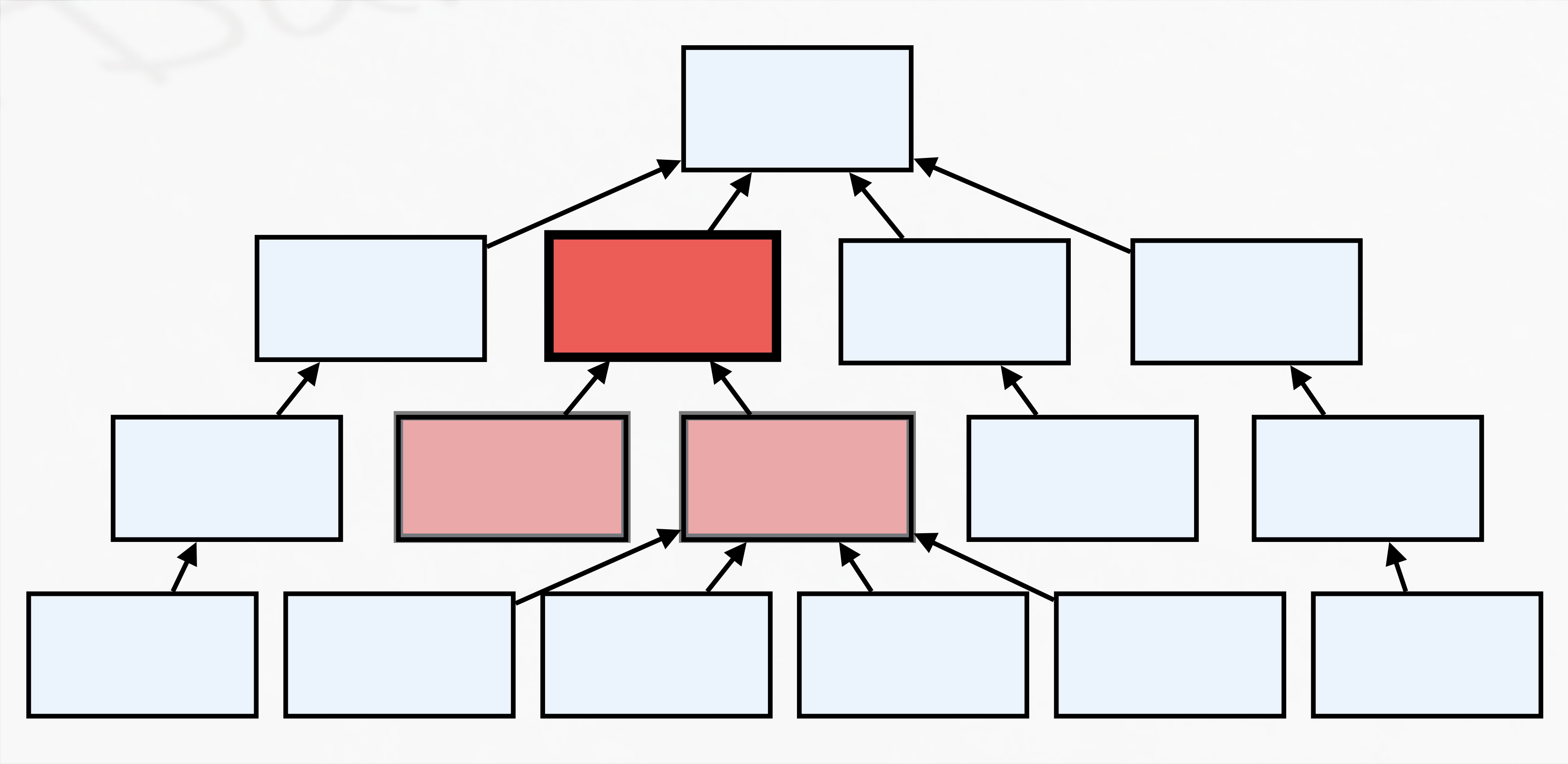
SQL:1999 — Recursion

```
SELECT t.id, t.parent
  FROM t
 WHERE t.id = ?
```



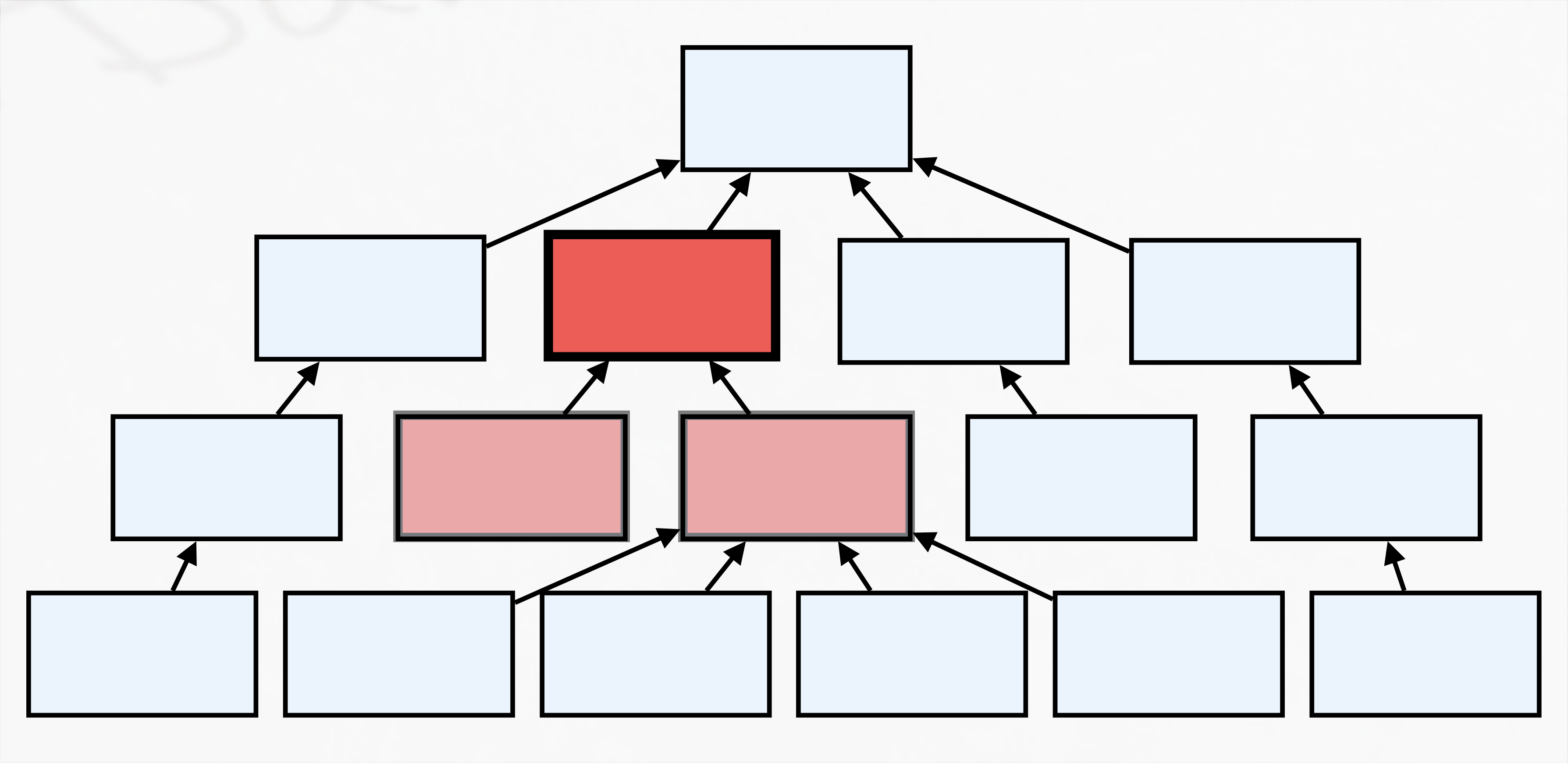
SQL:1999 — Recursion

```
  SELECT t.id, t.parent
    FROM t
   WHERE t.id = ?
UNION ALL
  SELECT t.id, t.parent
    FROM t
   WHERE t.parent = ?
```



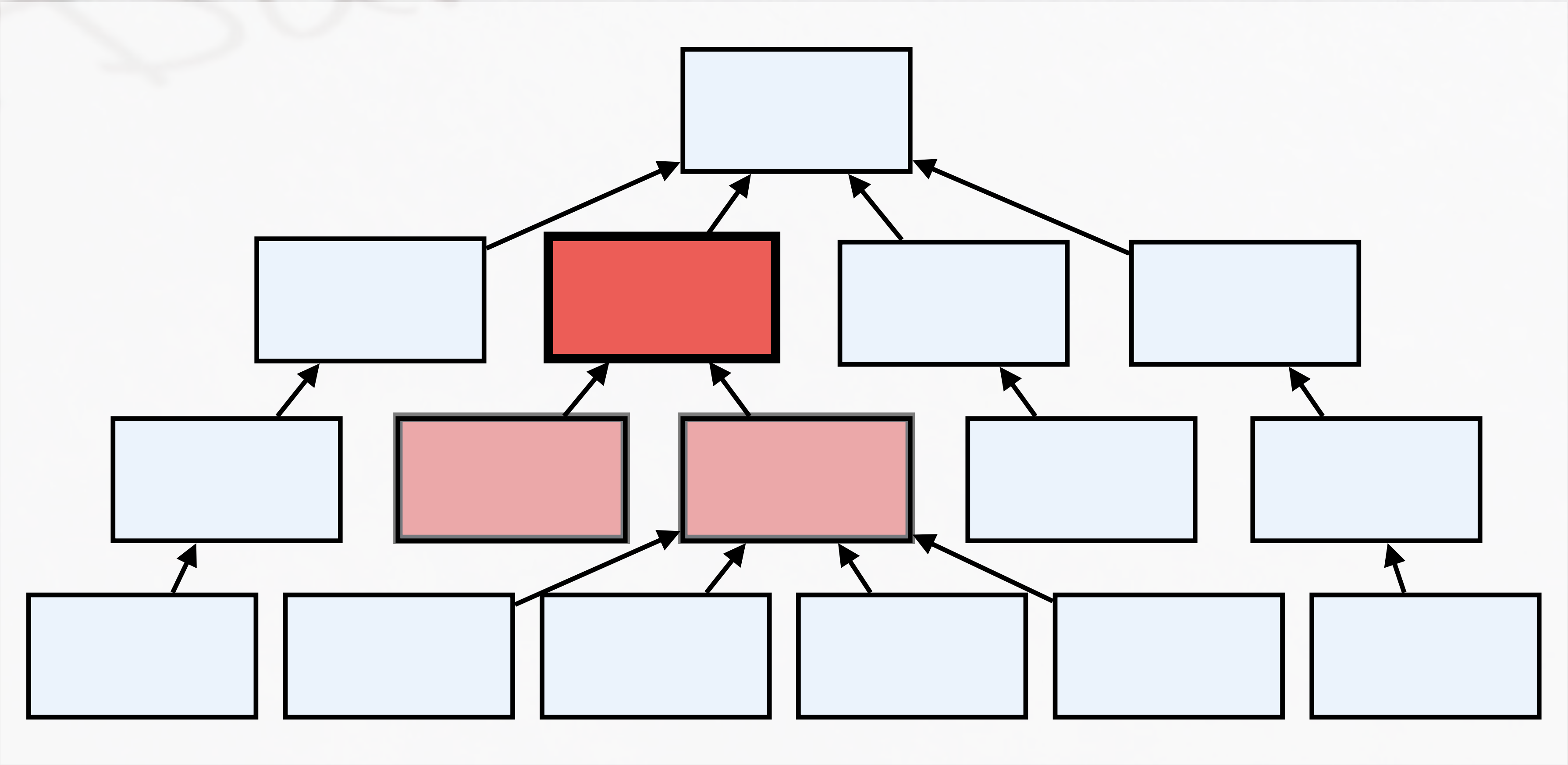
SQL:1999 — Recursion

```
  SELECT t.id, t.parent
    FROM t
   WHERE t.id = ?
UNION ALL
  SELECT t.id, t.parent
    FROM t
   WHERE t.parent = ?
```



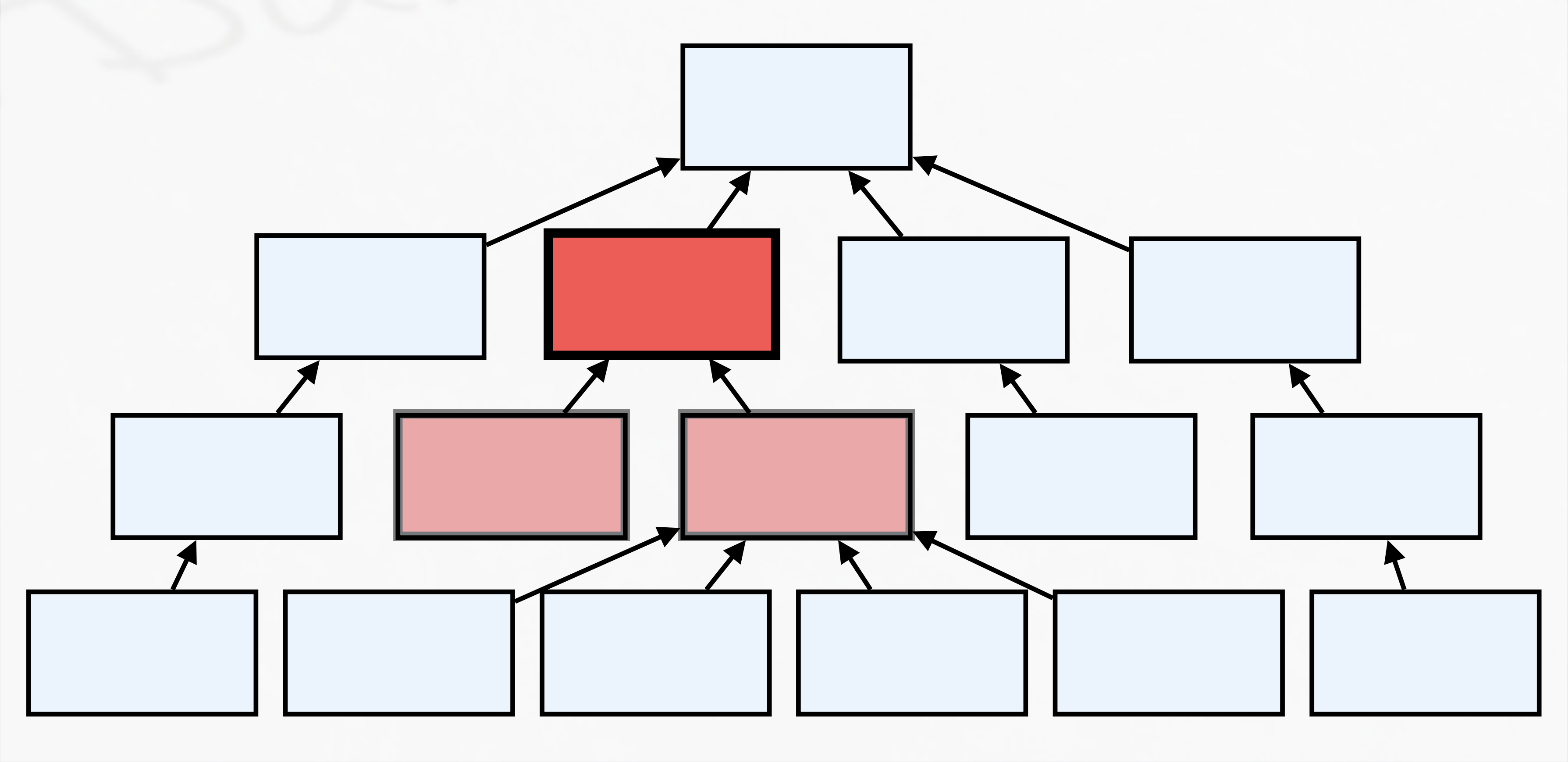
SQL:1999 — Recursion

```
  SELECT t.id, t.parent
    FROM t
   WHERE t.id = ?
UNION ALL
  SELECT t.id, t.parent
    FROM t
   WHERE t.parent = ?
```



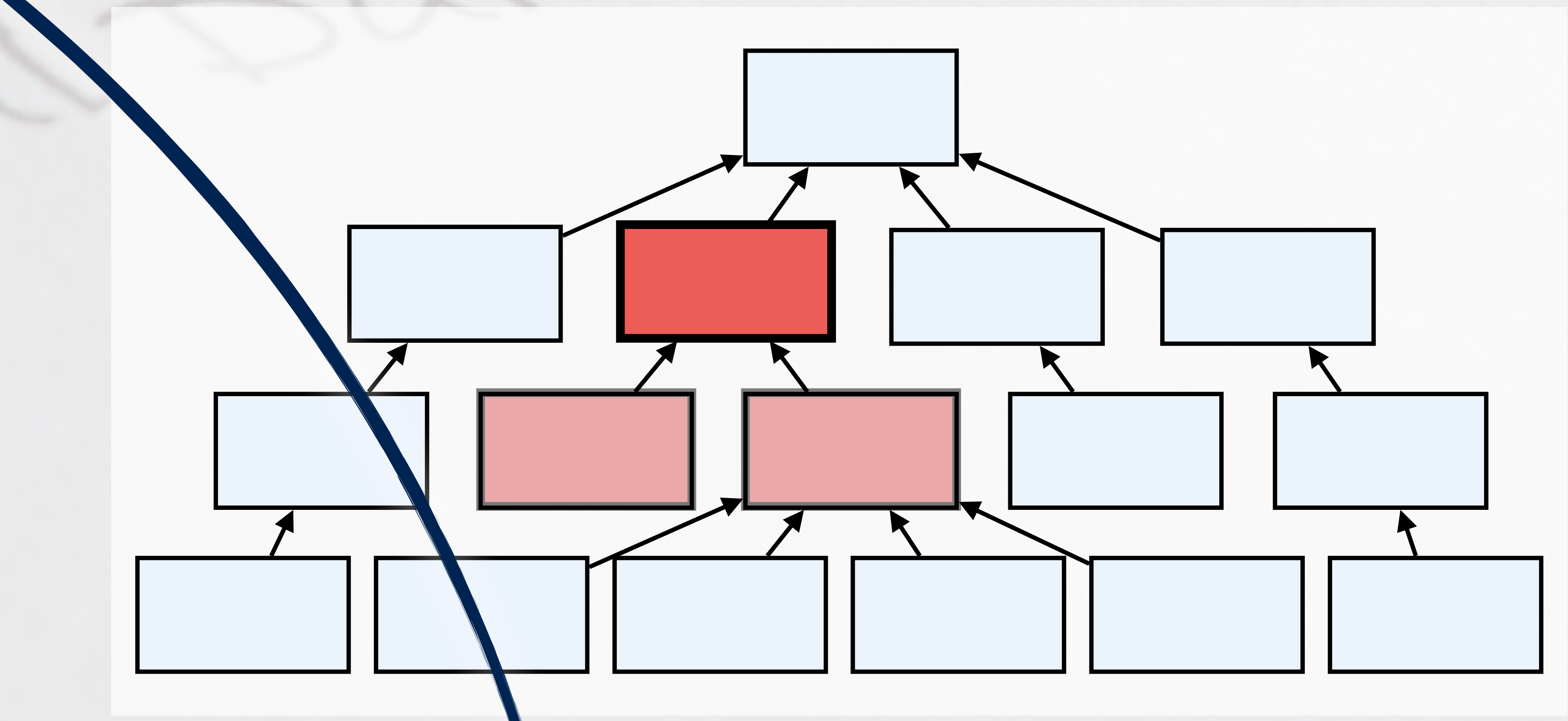
SQL:1999 — Recursion

```
WITH RECURSIVE prev (id, parent) AS (
  SELECT t.id, t.parent
    FROM t
   WHERE t.id = ?
UNION ALL
  SELECT t.id, t.parent
    FROM t
    JOIN prev ON t.parent = prev.id
)
SELECT * FROM prev
```



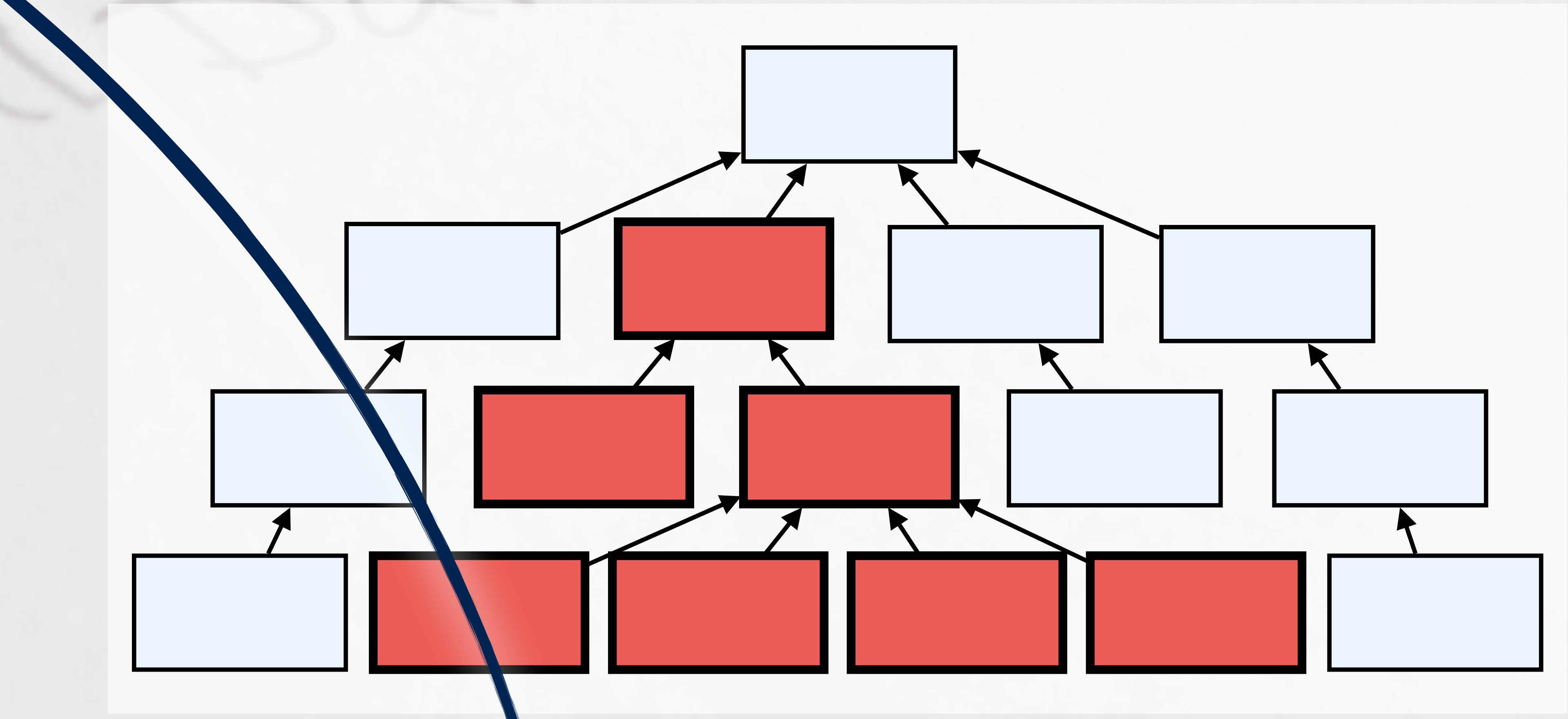
SQL:1999 — Recursion

```
WITH RECURSIVE prev (id, parent) AS (
   SELECT t.id, t.parent
      FROM t
     WHERE t.id = ?
UNION ALL
   SELECT t.id, t.parent
      FROM t
      JOIN prev ON t.parent = prev.id
)
SELECT * FROM prev
```

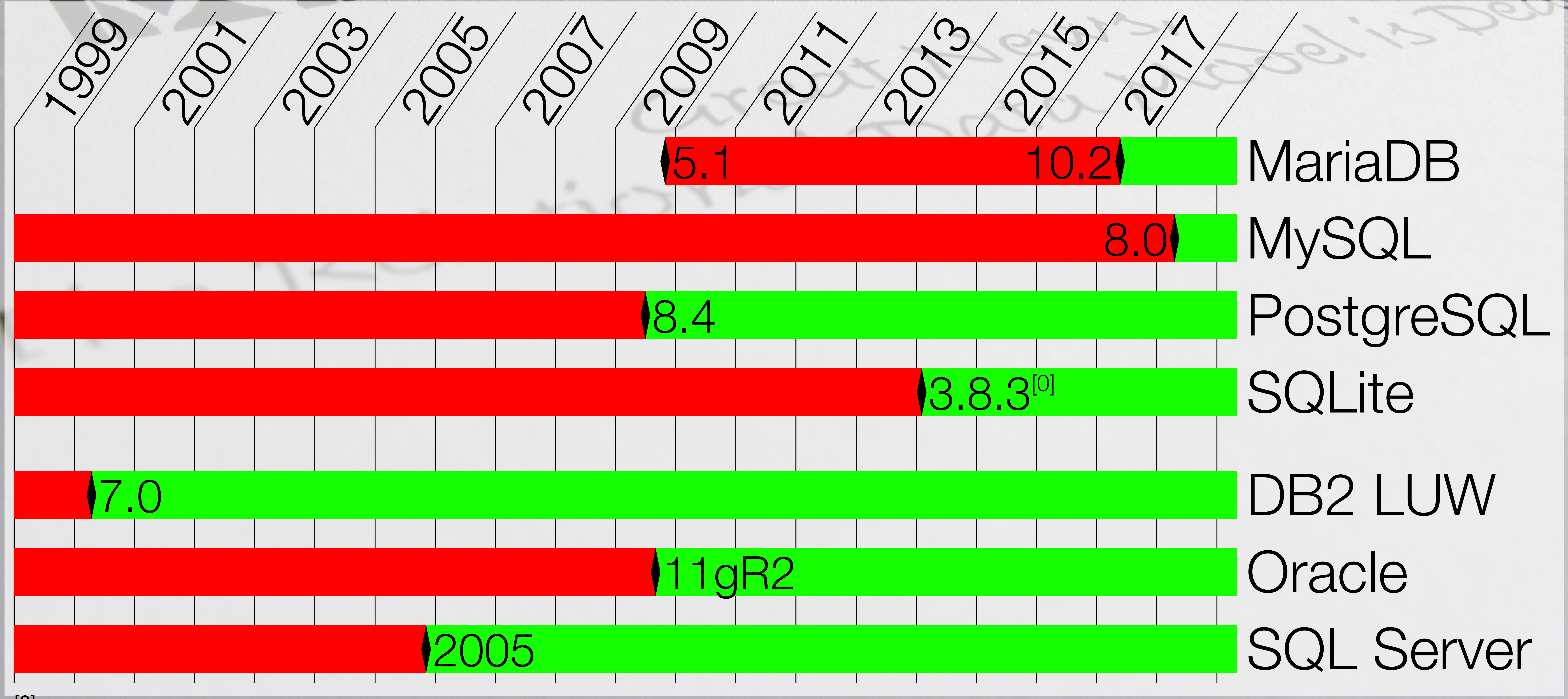


SQL:1999 — Recursion

```
WITH RECURSIVE prev (id, parent) AS (
  SELECT t.id, t.parent
    FROM t
   WHERE t.id = ?
UNION ALL
  SELECT t.id, t.parent
    FROM t
    JOIN prev ON t.parent = prev.id
)
SELECT * FROM prev
```



SQL:1999 — Recursion



[0]Only for top-level SELECT statements

Whitemarsh
Information Systems Corporation

Great News,
The Relational Data Model is Dead!

1999 ⏩ 2003

SQL/XML is Making Good Progress

Andrew Eisenberg
IBM, Westford, MA 01886
andrew.eisenberg@us.ibm.com

Jim Melton
Oracle Corp., Sandy
jim.melton

as its so
sch

the creation of a
ifications

http://www.acm.org:80/sigmod/record/issues/0206/standard.pdf (via Wayback machine)

SQL:2003 — Schemaless & Analytical

SQL/XML is Making Good Progress

Jim Melton
Oracle Corp., Sandy

Andrew Eisenberg
IBM, Westford, MA 01886
andrew.eisenberg@us.ibm.com

as its sou

the creation of a
ifications

Schemaless

- Introduced XML
  - Non-uniform documents in a single column

SQL:2003 — Schemaless & Analytical

Schemaless

- Introduced XML
  - Non-uniform documents in a single column

Later:

- JSON added with SQL:2016
- Proprietary JSON support:
  - 2012: PostgreSQL
  - 2014: Oracle
  - 2015: MySQL
  - 2016: SQL Server

SQL:2003 — Schemaless & Analytical

Schemaless

- Introduced XML
  - Non-uniform documents in a single column

Later:

- JSON added with SQL:2016
- Proprietary JSON support:
  - 2012: PostgreSQL
  - 2014: Oracle
  - 2015: MySQL
  - 2016: SQL Server

Analytical

- Introduced window functions
  - Accessing other rows of the current result

SQL:2003 — Schemaless & Analytical

Schemaless

- Introduced XML
  - Non-uniform documents in a single column

Later:

- JSON added with SQL:2016
- Proprietary JSON support:
  - 2012: PostgreSQL
  - 2014: Oracle
  - 2015: MySQL
  - 2016: SQL Server

Analytical

- Introduced window functions
  - Accessing other rows of the current result

Later:

- Extended in SQL:2011
- Popular among “New SQLs”
  - 2013: BigQuery, Hive
  - 2014: Impala
  - 2015: Spark SQL
  - 2016: NuoDB, MemSQL, Cockroach DB, VoltDB

SQL:2003 — Analytical

```
SELECT id, value




  FROM t
```

id	value
1	+10
2	+20
3	-10
4	+50
5	-30
6	-20

SQL:2003 — Analytical

```
SELECT id, value




  FROM t
```

id	value	**bal**
1	+10	
2	+20	
3	-10	
4	+50	
5	-30	
6	-20	

SQL:2003 — Analytical

```
SELECT id, value






  FROM t
```

id	value	**bal**
1	+10	**+10**
2	+20	
3	-10	
4	+50	
5	-30	
6	-20	

SQL:2003 — Analytical

```
SELECT id, value







  FROM t
```

id	value	bal
1	+10	**+10**
2	+20	**+30**
3	-10	**+20**
4	+50	**+70**
5	-30	**+40**
6	-20	**+20**

SQL:2003 — Analytical

```
SELECT id, value
     , SUM(value)
       OVER (




       ) bal
  FROM t
```

id	value	bal
1	+10	**+10**
2	+20	**+30**
3	-10	**+20**
4	+50	**+70**
5	-30	**+40**
6	-20	**+20**

SQL:2003 — Analytical

```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id




       ) bal
  FROM t
```

id	value	bal
1	+10	**+10**
2	+20	**+30**
3	-10	**+20**
4	+50	**+70**
5	-30	**+40**
6	-20	**+20**

SQL:2003 — Analytical

```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id




       ) bal
  FROM t
```

id	value	bal
1	+10	**+10**
2	+20	**+30**
3	-10	**+20**
4	+50	**+70**
5	-30	**+40**
6	-20	**+20**

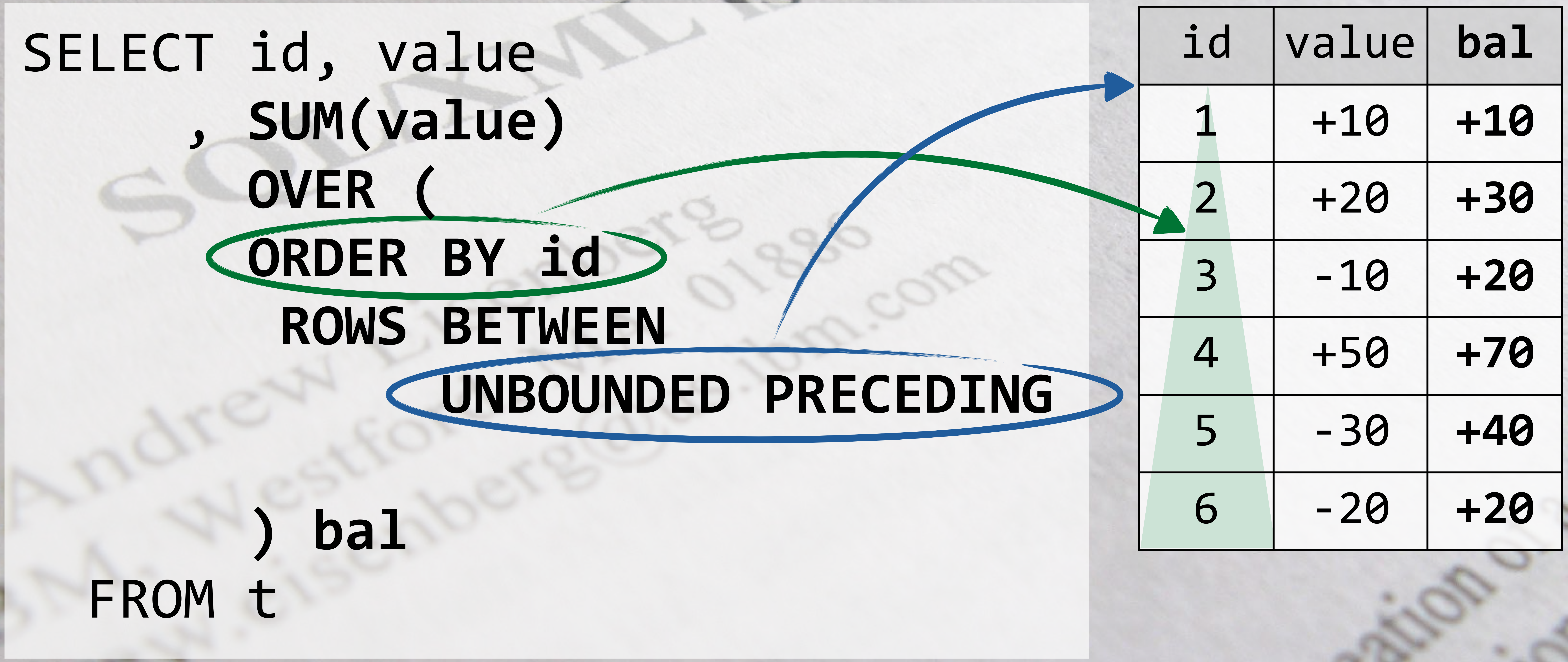
SQL:2003 — Analytical

```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
        ROWS BETWEEN
             UNBOUNDED PRECEDING

       ) bal
  FROM t
```

id	value	bal
1	+10	**+10**
2	+20	**+30**
3	-10	**+20**
4	+50	**+70**
5	-30	**+40**
6	-20	**+20**

SQL:2003 — Analytical

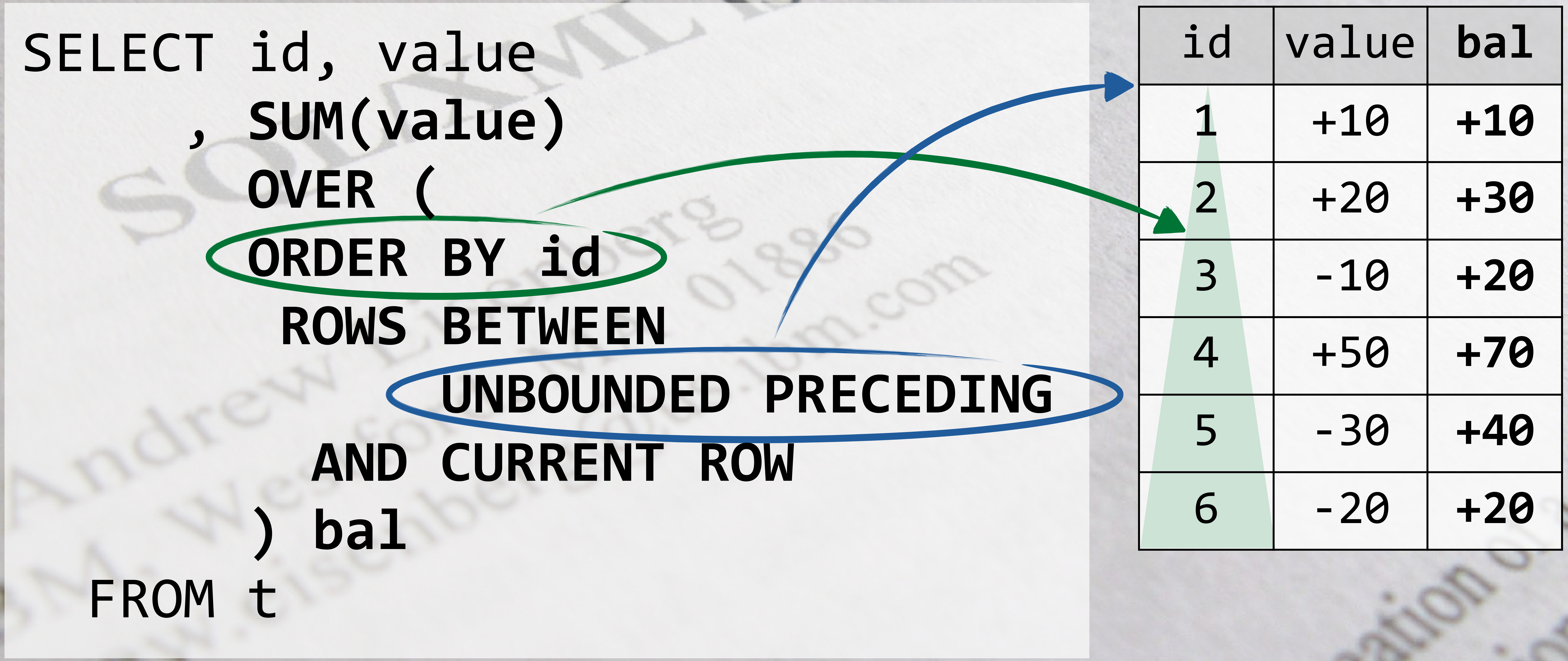


```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
        ROWS BETWEEN
             UNBOUNDED PRECEDING

       ) bal
  FROM t
```

id	value	bal
1	+10	+10
2	+20	+30
3	-10	+20
4	+50	+70
5	-30	+40
6	-20	+20

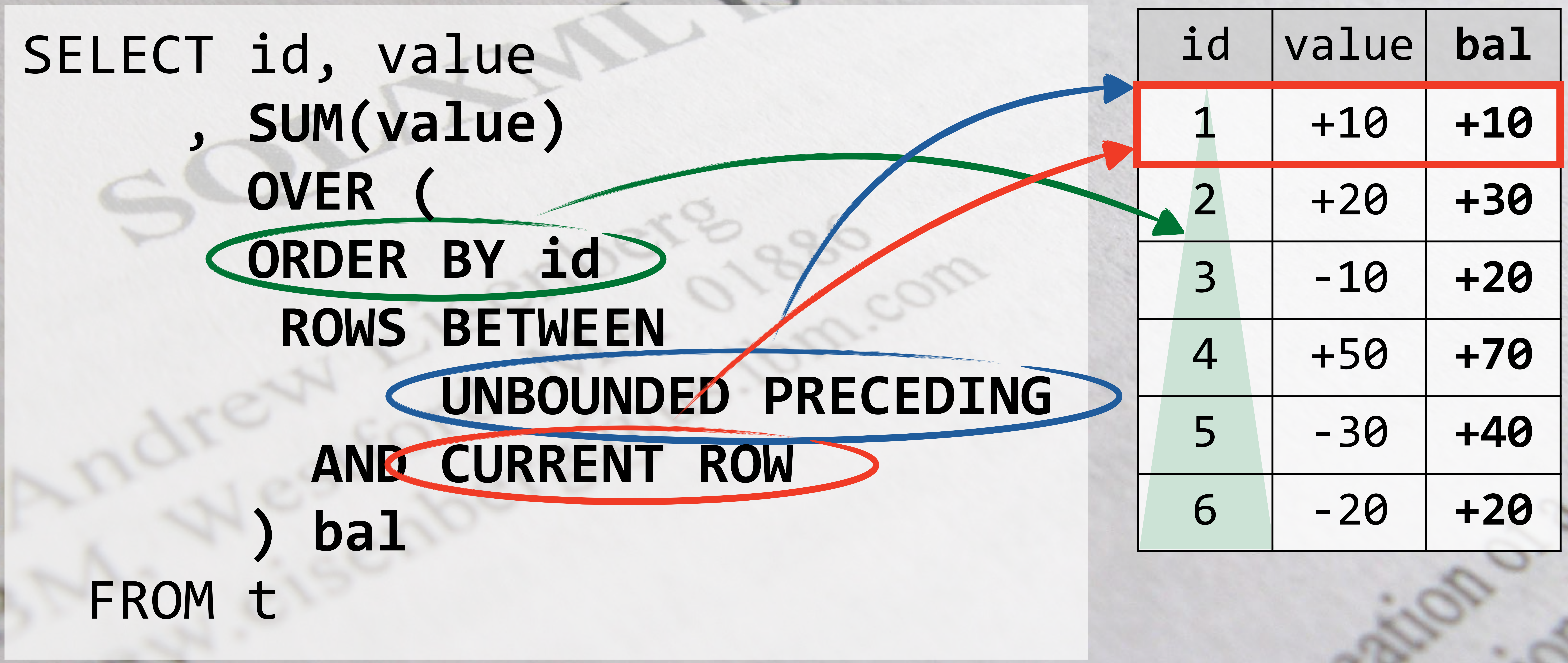
SQL:2003 — Analytical



```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
        ROWS BETWEEN
             UNBOUNDED PRECEDING
         AND CURRENT ROW
       ) bal
  FROM t
```

id	value	bal
1	+10	+10
2	+20	+30
3	-10	+20
4	+50	+70
5	-30	+40
6	-20	+20

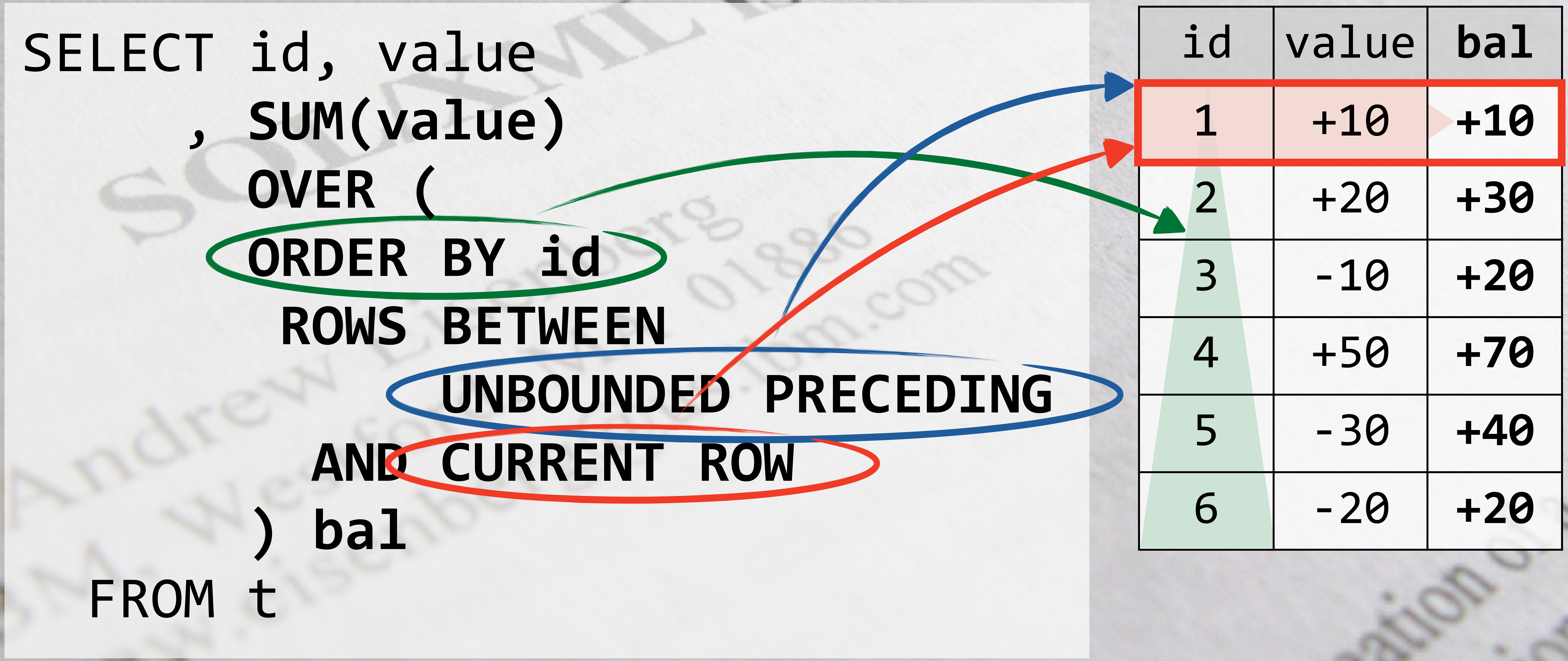
SQL:2003 — Analytical



```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
        ROWS BETWEEN
             UNBOUNDED PRECEDING
         AND CURRENT ROW
       ) bal
  FROM t
```

id	value	bal
1	+10	+10
2	+20	+30
3	-10	+20
4	+50	+70
5	-30	+40
6	-20	+20

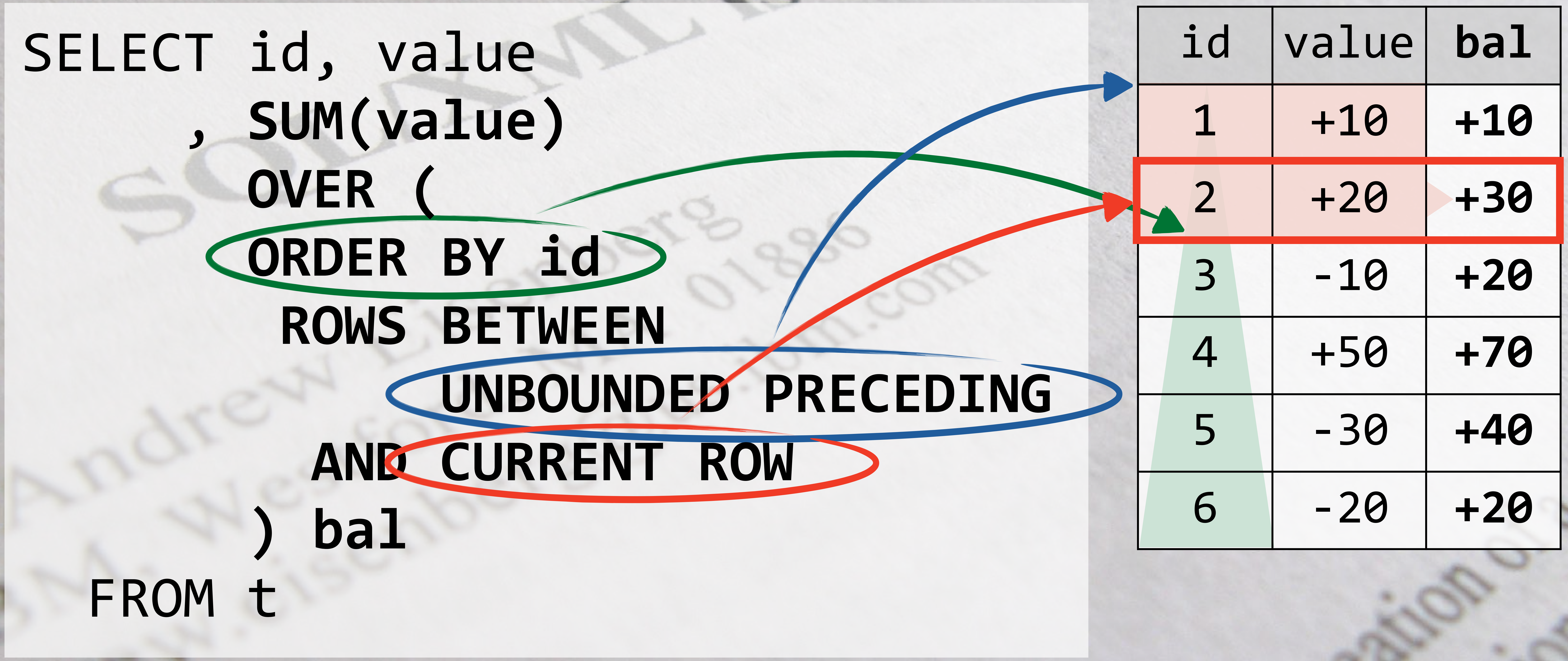
SQL:2003 — Analytical



```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
        ROWS BETWEEN
             UNBOUNDED PRECEDING
         AND CURRENT ROW
       ) bal
  FROM t
```

id	value	bal
1	+10	+10
2	+20	+30
3	-10	+20
4	+50	+70
5	-30	+40
6	-20	+20

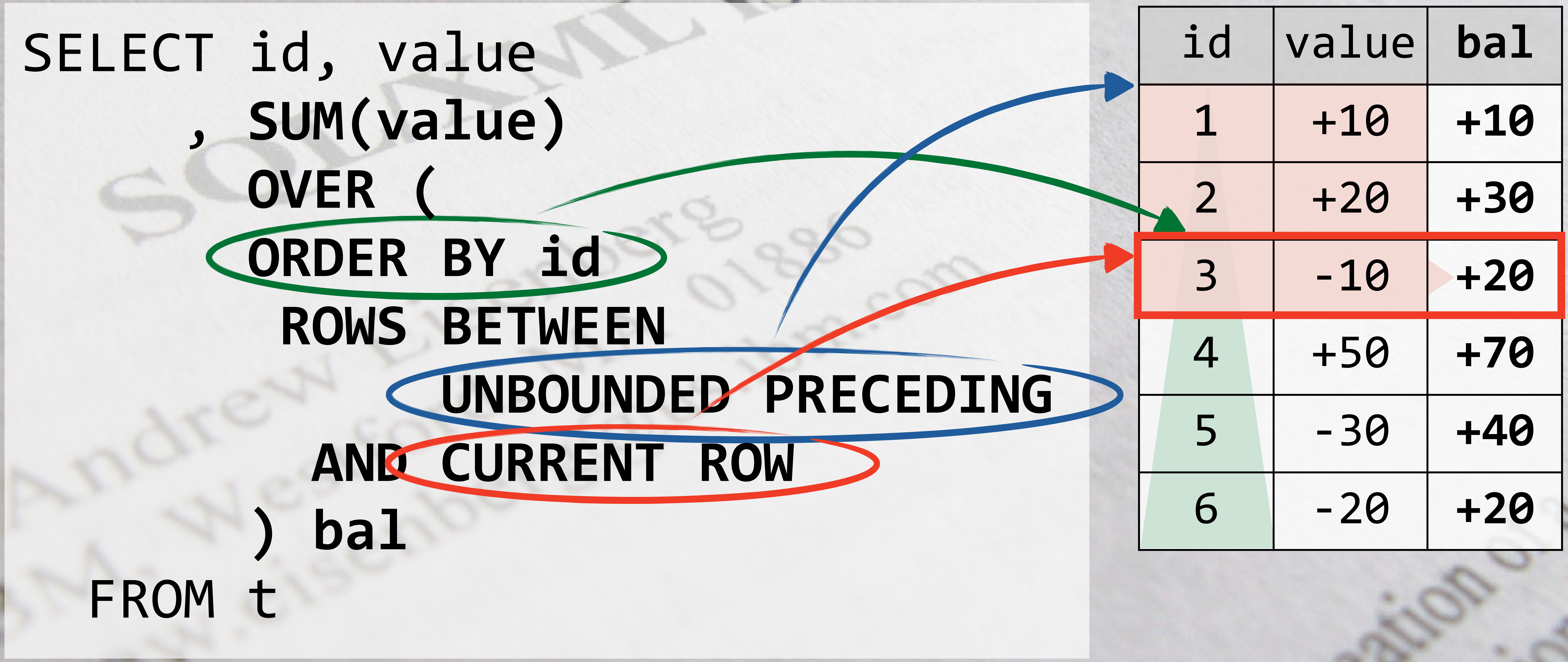
SQL:2003 — Analytical



```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
        ROWS BETWEEN
             UNBOUNDED PRECEDING
         AND CURRENT ROW
       ) bal
  FROM t
```

id	value	bal
1	+10	+10
2	+20	+30
3	-10	+20
4	+50	+70
5	-30	+40
6	-20	+20

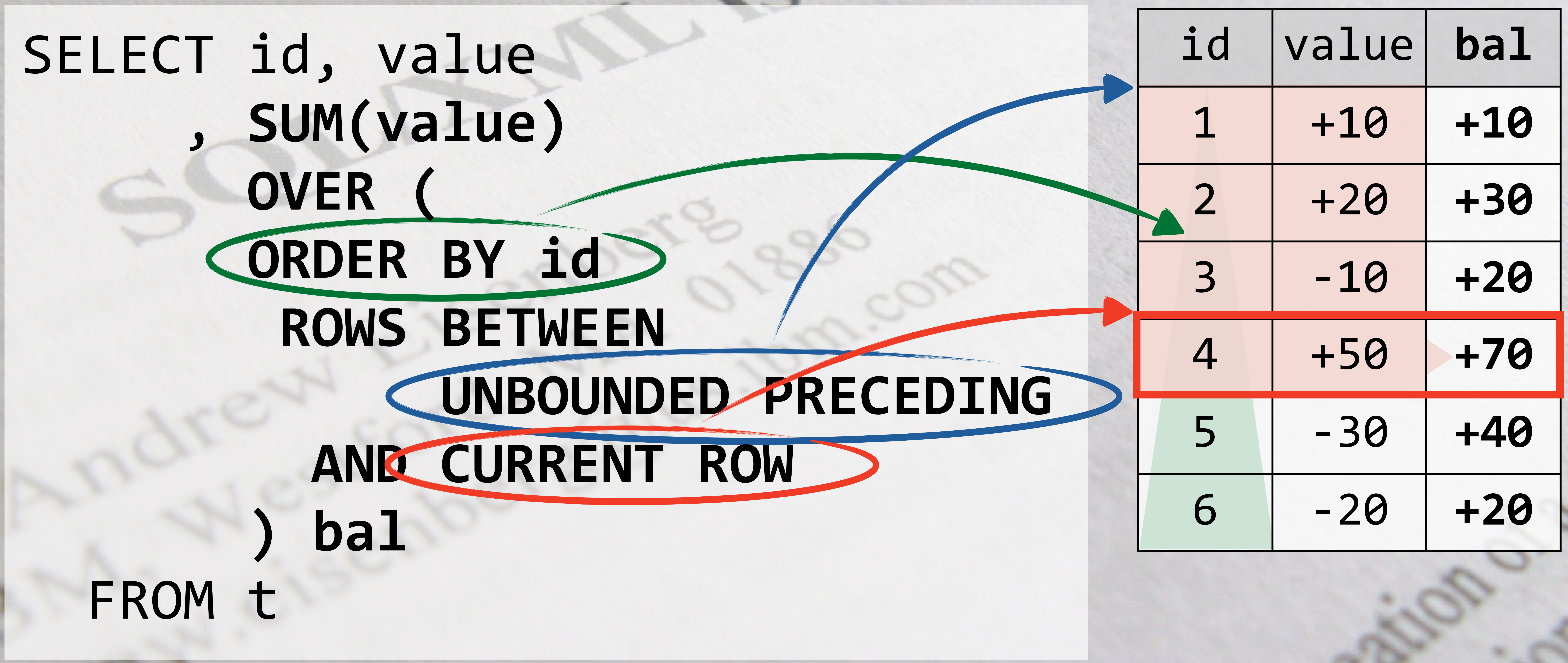
SQL:2003 — Analytical



```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
        ROWS BETWEEN
             UNBOUNDED PRECEDING
         AND CURRENT ROW
       ) bal
  FROM t
```

id	value	bal
1	+10	+10
2	+20	+30
3	-10	+20
4	+50	+70
5	-30	+40
6	-20	+20

SQL:2003 — Analytical



```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
        ROWS BETWEEN
             UNBOUNDED PRECEDING
         AND CURRENT ROW
       ) bal
  FROM t
```

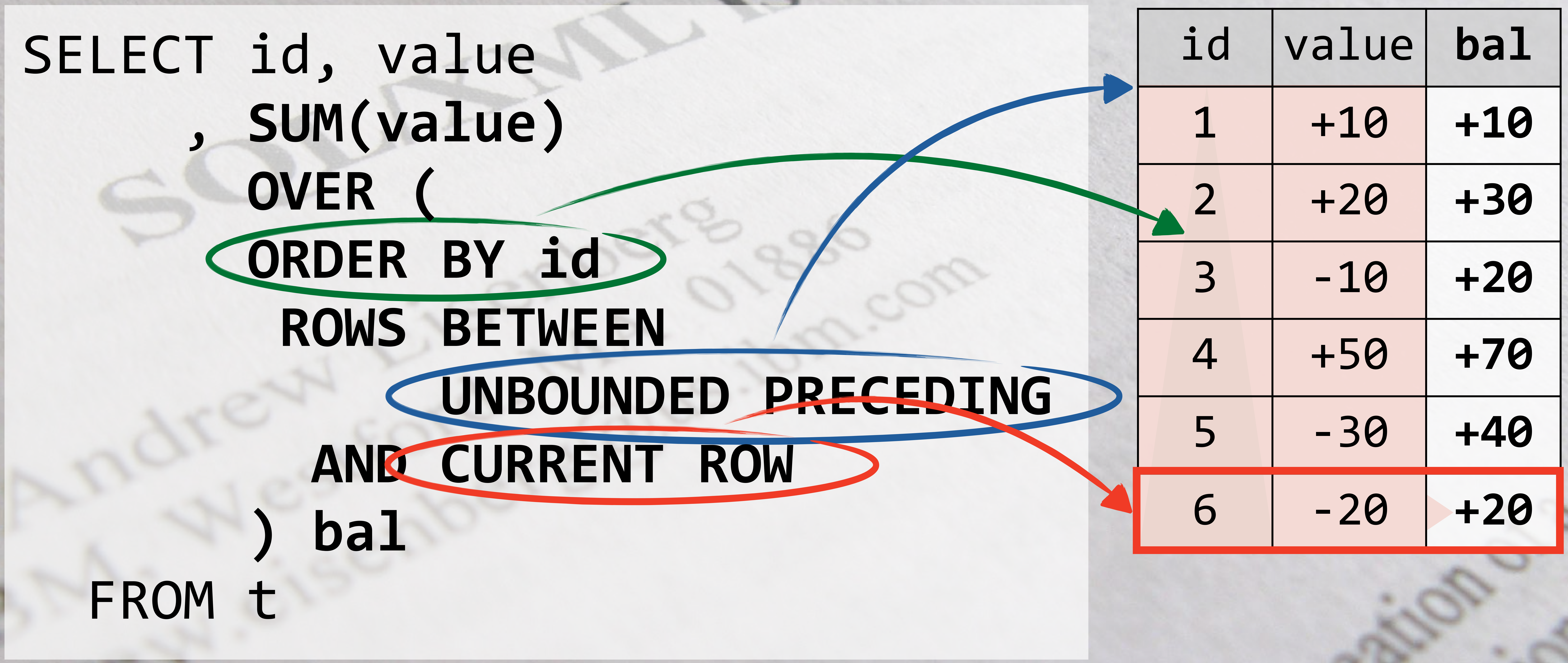
id	value	bal
1	+10	+10
2	+20	+30
3	-10	+20
4	+50	+70
5	-30	+40
6	-20	+20

SQL:2003 — Analytical

```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
        ROWS BETWEEN
            UNBOUNDED PRECEDING
         AND CURRENT ROW
       ) bal
  FROM t
```

id	value	bal
1	+10	+10
2	+20	+30
3	-10	+20
4	+50	+70
5	-30	+40
6	-20	+20

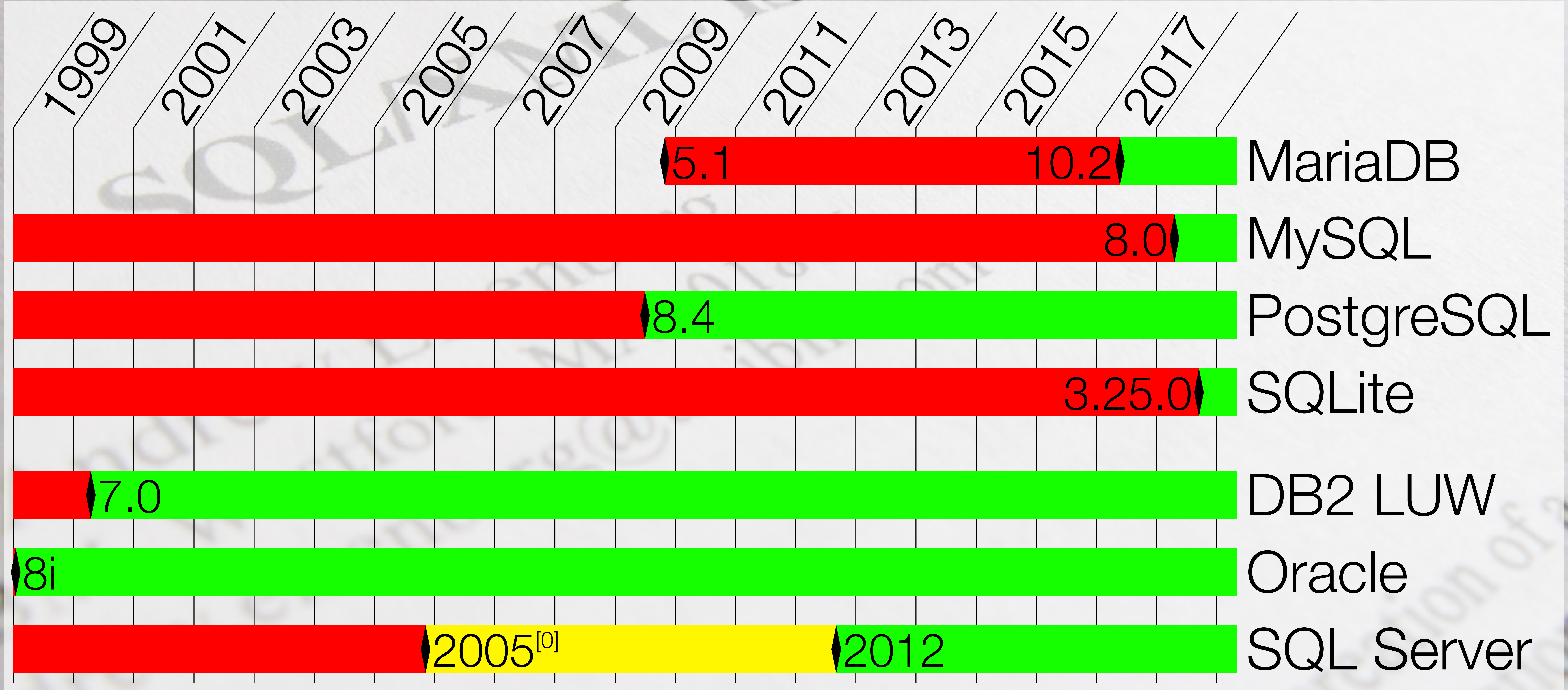
SQL:2003 — Analytical



```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
        ROWS BETWEEN
             UNBOUNDED PRECEDING
         AND CURRENT ROW
       ) bal
  FROM t
```

id	value	bal
1	+10	+10
2	+20	+30
3	-10	+20
4	+50	+70
5	-30	+40
6	-20	+20

SQL:2003 — Analytical



[0]No framing

SQL/XML is Making Good Progress

Jim Melton
Oracle Corp., Sand
jim.melton

Andrew Eis
IBM, Westford
andrew.eisenberg@us.ibm.c

as its sou
sche

the creation of a
ecifications
time, w

2003

2016

SQL:2016 — JSON

Information technology — Database languages — SQL Technical Reports —

Part 6:

SQL support for JavaScript Object Notation (JSON)

nologies de l'information — Langages de base de donn

echniques —

rt de SQL pour JavaScript Object No

http://standards.iso.org/ittf/PubliclyAvailableStandards/c067367_ISO_IEC_TR_19075-6_2017.zip

SQL:2016 — JSON

TECHNICAL REPORT

Information technology — Database languages — SQL Technical Reports —

Part 6: SQL support for JavaScript Object Notation (JSON)

Technologies de l'information — Langages de base de donn

techniques —

Partie 6: Support de SQL pour JavaScript Object No

```
[
  {
    "id": 42,
    "a1": "foo"
  },
  {
    "id": 43,
    "a1": "bar"
  }
]
```

http://standards.iso.org/ittf/PubliclyAvailableStandards/c067367_ISO_IEC_TR_19075-6_2017.zip

SQL:2016 — JSON

```
[
  {
    "id": 42,
    "a1": "foo"
  },
  {
    "id": 43,
    "a1": "bar"
  }
]
```

id	a1
42	foo
43	bar

http://standards.iso.org/ittf/PubliclyAvailableStandards/c067367_ISO_IEC_TR_19075-6_2017.zip

SQL:2016 — JSON_TABLE

```
SELECT *
  FROM JSON_TABLE
       ( ?
       , '$[*]'
         COLUMNS
         ( id INT            PATH '$.id'
         , a1 VARCHAR(…) PATH '$.a1'
         )
       ) r
```

```
[
  {
    "id": 42,
    "a1": "foo"
  },
  {
    "id": 43,
    "a1": "bar"
  }
]
```

id	a1
42	foo
43	bar

SQL:2016 — JSON_TABLE

```
SELECT *
  FROM JSON_TABLE
       ( ?
         '$[*]'
         COLUMNS
         ( id INT              PATH '$.id'
         , a1 VARCHAR(…) PATH '$.a1'
         )
       ) r
```

Bind Parameter

```
[
  {
    "id": 42,
    "a1": "foo"
  },
  {
    "id": 43,
    "a1": "bar"
  }
]
```

id	a1
42	foo
43	bar

SQL:2016 — JSON_TABLE

```
SELECT *
   FROM JSON_TABLE
        ( ?
        , '$[*]'
        COLUMNS
        ( id INT             PATH '$.id'
        , a1 VARCHAR(…) PATH '$.a1'
        )
      ) r
```

Bind Parameter

SQL/JSON Path

- Query language to select elements from a JSON document
- Defined in the SQL standard

```
[
  {
    "id": 42,
    "a1": "foo"
  },
  {
    "id": 43,
    "a1": "bar"
  }
]
```

id	a1
42	foo
43	bar

SQL:2016 — JSON_TABLE

```
SELECT *
   FROM JSON_TABLE
        ( ?
        , '$[*]'
          COLUMNS
          ( id INT            PATH '$.id'
          , a1 VARCHAR(…) PATH '$.a1'
          )
        ) r
```

Bind Parameter

SQL/JSON Path

- Query language to select elements from a JSON document
- Defined in the SQL standard

```
[
  {
     "id": 42,
     "a1": "foo"
  },
  {
     "id": 43,
     "a1": "bar"
  }
]
```

id	a1
42	foo
43	bar

SQL:2016 — JSON_TABLE

```
SELECT *
  FROM JSON_TABLE
       ( ?
       , '$[*]'
         COLUMNS
         ( id INT              PATH '$.id'
         , a1 VARCHAR(…)       PATH '$.a1'
         )
       ) r
```

Bind Parameter

SQL/JSON Path

- Query language to select elements from a JSON document
- Defined in the SQL standard

```
[
  {
     "id": 42,
     "a1": "foo"
  },
  {
     "id": 43,
     "a1": "bar"
  }
]
```

id	a1
42	foo
43	bar

SQL:2016 — JSON_TABLE — Use Case

```
SELECT *
  FROM JSON_TABLE
       ( ?
       , '$[*]'
         COLUMNS
         ( id INT            PATH '$.id'
         , a1 VARCHAR(…) PATH '$.a1'
         )
       ) r
```

```
[
  {
    "id": 42,
    "a1": "foo"
  },
  {
    "id": 43,
    "a1": "bar"
  }
]
```

id	a1
42	foo
43	bar

SQL:2016 — JSON_TABLE — Use Case

```
INSERT INTO target_table
SELECT *
  FROM JSON_TABLE
       ( ?
       , '$[*]'
         COLUMNS
         ( id INT            PATH '$.id'
         , a1 VARCHAR(…) PATH '$.a1'
         )
       ) r
```

```
[
  {
    "id": 42,
    "a1": "foo"
  },
  {
    "id": 43,
    "a1": "bar"
  }
]
```

id	a1
42	foo
43	bar

SQL:2016 — JSON_TABLE — Use Case

```
INSERT INTO target_table
SELECT *
  FROM JSON_TABLE
       ( ?
       , '$[*]'
         COLUMNS
         ( id INT            PATH '$.id'
         , a1 VARCHAR(…) PATH '$.a1'
         )
       ) r
```

```
   },
   {
     "id": 43,
     "a1": "bar"
   }
]
```

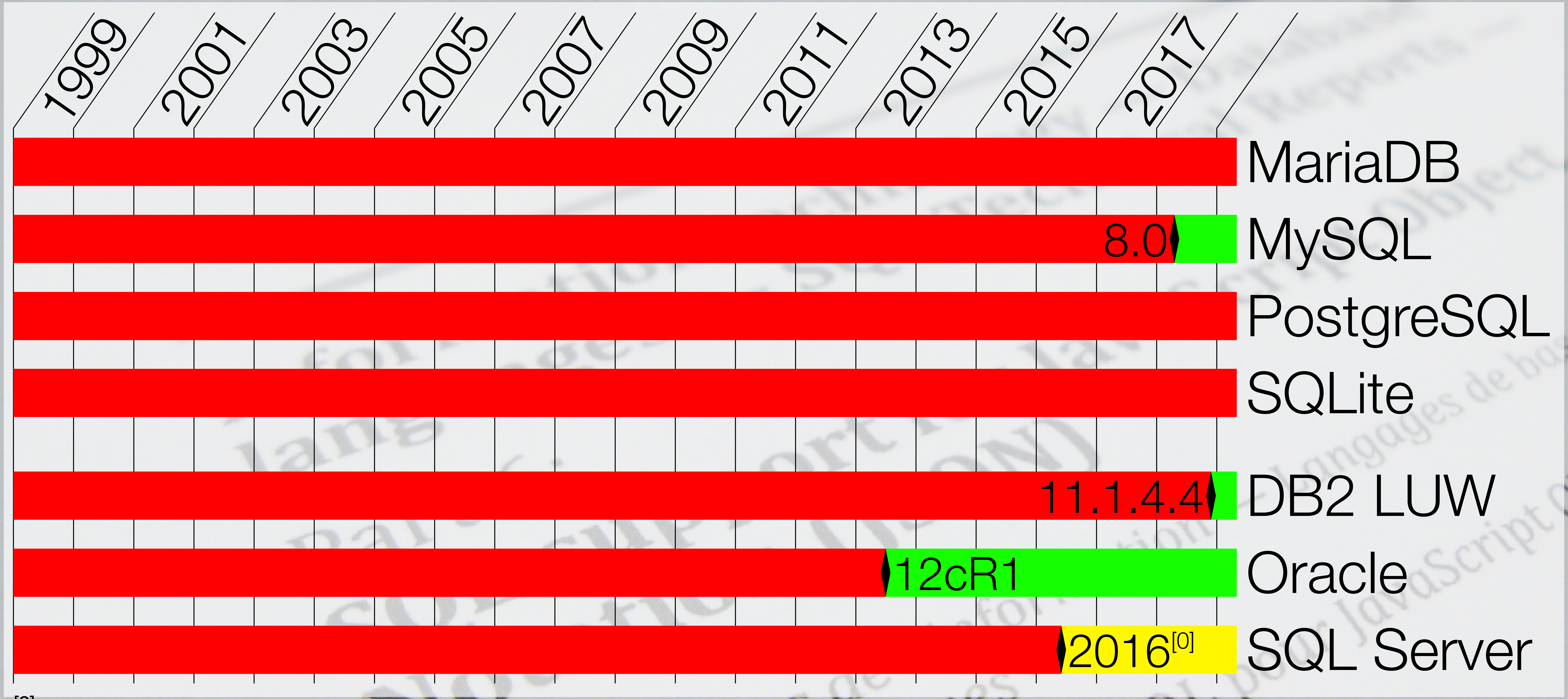
id	a1
42	foo
43	bar

Session tip:

How Well Do Relational Database Engines Support JSON?

Today 15:30!

SQL:2016 — JSON_TABLE



[0]OPENJSON provides similar functionality

TECHNICAL REPORT

Informa — Database

languag nical Reports —

Part 6:

SQL support f avaScr Object

Notation (JSON)

nologies de l'information — Langages de base de donn

techniques —

rt de SQL pour JavaScript Object No

2011

2016

SQL:2011 — Time Travelling

Temporal features in SQL:2011

Krishna Kulkarni, Jan-Eike Michels (IBM Corporation)
{krishnak, janeike}@us.ibm.com

ABSTRACT

SQL:2011 was published in December of 2011, replacing SQL:2008 as the most recent revision of the SQL standard. This paper covers the most important new functionality that is part of SQL:2011: the ability to create and manipulate temporal tables.

implement temporal support as part of the application logic, which often resulted in expensive developm cycles and complex, hard-to-maintain code. In 1995, the ISO SQL committee initi create a new part of SQL standa guage extensions for the te language extension TSQL2 [8] w time

http://cs.ulb.ac.be/public/_media/teaching/infoh415/tempfeaturessql2011.pdf

SQL:2011 — Time Travelling

Temporal features in SQL:2011

Krishna Kulkarni, Jan-Eike Michels (IBM Corporation)

{krishnak, janeike}@us.ibm.com

ABSTRACT

SQL:2011 was published in December of 2011, replacing SQL:2008 as the most recent revision of the SQL standard. This paper covers the most important new functionality that is part of SQL:2011: the ability to create and manipulate temporal tables.

implement temporal support as part of the application logic, which often resulted in expensive development cycles and complex, hard-to-maintain code. In 1995, the ISO SQL committee initiated create a new part of SQL standard guage extensions for the language extension TSQL2 [8] time

Introduction

<u>Application Versioning</u>

- Dedicated syntax added
  - When did something happen in the real world?

http://cs.ulb.ac.be/public/_media/teaching/infoh415/tempfeaturessql2011.pdf

SQL:2011 — Time Travelling

Application Versioning

- Dedicated syntax added
  - When did something happen in the real world?

New syntax (excerpt)

- `FOR PORTION OF` in `UPDATE` and `DELETE`
- `WITHOUT OVERLAPS` in `UNIQUE` constraints & `PRIMARY KEYS`
- `[IMMEDIATELY] PRECEDES`, `OVERLAPS` in `WHERE`, `HAVING`, …

SQL:2011 — Time Travelling

Application Versioning

- Dedicated syntax added
  - When did something happen in the real world?

New syntax (excerpt)

- FOR PORTION OF in UPDATE and DELETE
- WITHOUT OVERLAPS in UNIQUE constraints & PRIMARY KEYS
- [IMMEDIATELY] PRECEDES, OVERLAPS in WHERE, HAVING, …

System Versioning

- Fully automatic and (almost) transparent
  - When did we learn about something

SQL:2011 — Time Travelling

Application Versioning

- Dedicated syntax added
  - When did something happen in the real world?

New syntax (excerpt)

- `FOR PORTION OF` in `UPDATE` and `DELETE`
- `WITHOUT OVERLAPS` in `UNIQUE` constraints & `PRIMARY KEYS`
- `[IMMEDIATELY] PRECEDES`, `OVERLAPS` in `WHERE`, `HAVING`,…

System Versioning

- Fully automatic and (almost) transparent
  - When did we learn about something

Transparent changes, new syntax for queries

- `INSERT`, `UPDATE` & `DELETE` use the system time automatically
- `SELECT` can use `FOR SYSTEM_TIME AS OF`

SQL:2011 — System Versioning

Temporal Features in SQL:2011

Krishna Kulkarni, Jan-Eike Michels (IBM Corporation)
{krishnak, janeike}@us.ibm.com

ABSTRACT

SQL:2011 was published in December of 2011, replacing SQL:2008 as the most recent revision of the SQL standard. This paper covers the most important new functionality that is part of SQL:2011: the ability to create and manipulate temporal tables.

1. Introduction

implement temporal support as part of the application logic, which often resulted in expensive development cycles and complex, hard-to-maintain code.

In 1995, the ISO SQL committee initiated ... create a new part of SQL standard ... guage extensions for the ... language extensions ... TSQL2 [8] ...

SQL:2011 — System Versioning

```
CREATE TABLE t (
   ...
 , from TIMESTAMP(9) GENERATED ALWAYS
                     AS ROW START
 , till TIMESTAMP(9) GENERATED ALWAYS
                     AS ROW END

 , PERIOD FOR SYSTEM_TIME (from, till)
) WITH SYSTEM VERSIONING
```

SQL:2011 — System Versioning

Temporal features in SQL:2011

Krishna Kulkarni, Jan-Eike Michels (IBM Corporation)

{krishnak, janeike}@us.ibm.com

ABSTRACT

SQL:2011 was published in December of 2011, replacing SQL:2008 as the most recent revision of the SQL standard. This paper covers the most important new functionality that is part of SQL:2011: the ability to create and manipulate temporal tables.

1. Introduction

implement temporal support as part of the application

logic, which often resulted in expensive develop

cycles and complex, hard-to-maintain code.

In 1995, the ISO SQL committee initi

create a new part of SQL standa

guage extensions for the te

language extension

TSQL2 [8] w

SQL:2011 — System Versioning

```
INSERT INTO t (id, data)
       VALUES (1 , 'X' )
```

id	data	from	till
1	X	10:00	

SQL:2011 — System Versioning

```
INSERT INTO t (id, data)
       VALUES (1 , 'X' )
```

id	data	from	till
1	X	10:00	

```
UPDATE t
   SET data = 'Y'
 WHERE id = 1
```

id	data	from	till
1	X	10:00	11:00
1	Y	11:00	

SQL:2011 — System Versioning

```
INSERT INTO t (id, data)
       VALUES (1 , 'X' )
```

id	data	from	till
1	X	10:00	

```
UPDATE t
   SET data = 'Y'
 WHERE id = 1
```

id	data	from	till
1	X	10:00	11:00
1	Y	11:00	

```
DELETE FROM t
 WHERE id = 1
```

id	data	from	till
1	X	10:00	11:00
1	Y	11:00	12:00

SQL:2011 — System Versioning

id	data	from	till
1	X	10:00	11:00
1	Y	11:00	12:00

SQL:2011 — System Versioning

id	data	from	till
1	X	10:00	11:00
1	Y	11:00	12:00

```
SELECT *
  FROM t
```

id	data	from	till

SQL:2011 — System Versioning

id	data	from	till
1	X	10:00	11:00
1	Y	11:00	12:00

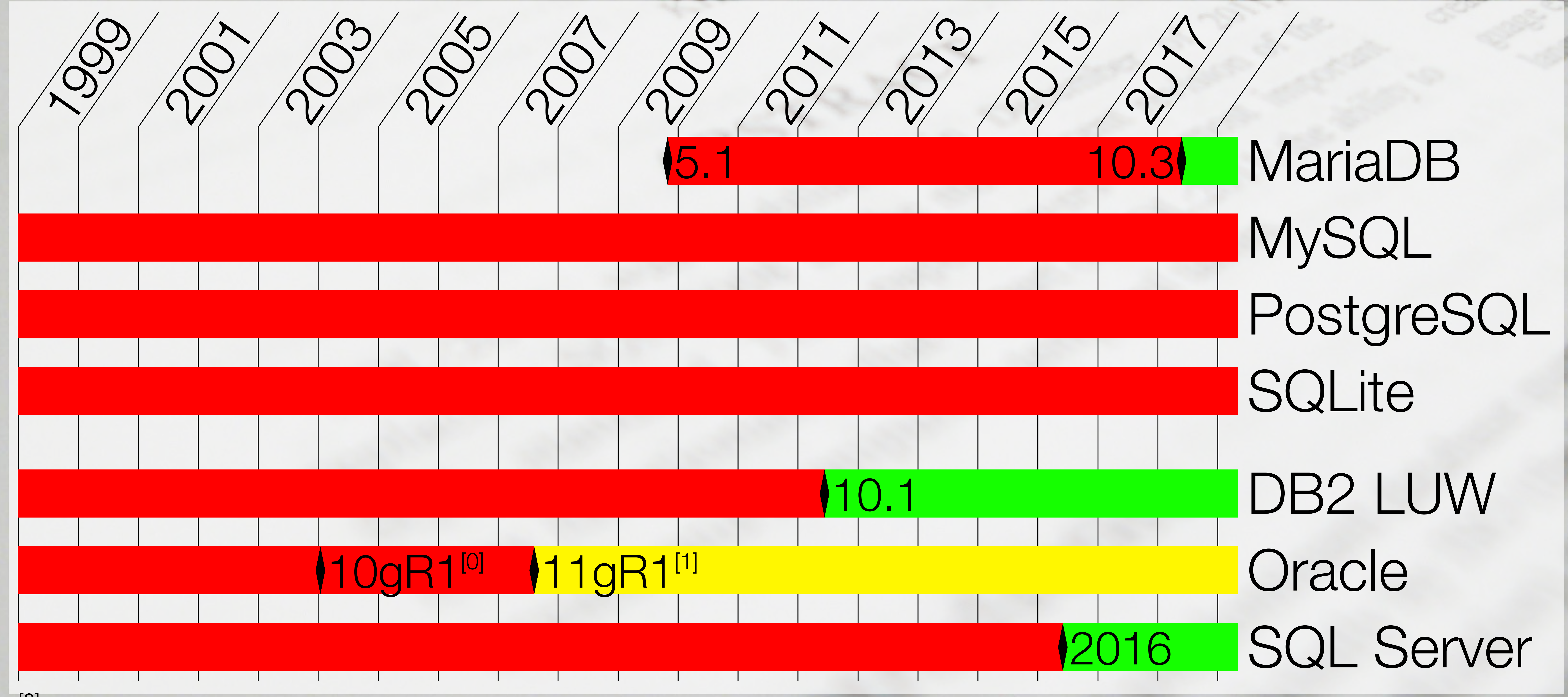
```
SELECT *
  FROM t
```

id	data	from	till

```
SELECT *
  FROM t
   FOR SYSTEM_TIME AS OF
   TIMESTAMP'…10:30:00'
```

id	data	from	till
1	X	10:00	11:00

SQL:2011 — System Versioning



[0] Short term using Flashback.
[1] Flashback Archive. Proprietery syntax.

INTERNATIONAL STANDARD

ISO/IEC 9075-2

Information technology — Database languages — SQL —

Part 2:
Foundation (SQL/Foundation)

Technologies de l'information — Langages de base de données — SQL —

Partie 2: Fondations (SQL/Fondations)

https://webstore.iec.ch/publication/59685

A **lot** has happened since SQL-92

INTERNATIONAL STANDARD

ISO/IEC 9075-2

Information technology — Database languages — SQL —

Part 2:
Foundation (SQL/Foundation)

Technologies de l'information — Langages de base de données — SQL —

Partie 2: Fondations (SQL/Fondations)

A **lot** has happened since SQL-92

SQL has evolved **beyond** the relational idea

INTERNATIONAL STANDARD

ISO/IEC 9075-2

Information technology — Database languages — SQL —

Part 2:
Foundation (SQL/Foundation)

Technologies de l'information — Langages de base de données — SQL —

Partie 2: Fondations (SQL/Fondations)

A **lot** has
happened
since SQL-92

SQL has evolved
beyond
the relational idea

If you use SQL for
CRUD operations only,
you are doing it wrong

INTERNATIONAL
STANDARD

ISO/IEC
9075-2

Information technology — Database
languages — SQL —
Part 2:
Foundation (SQL/Foundation)

Technologies de l'information — Langages de base de données —
SQL —
Partie 2: Fondations (SQL/Fondations)

A **lot** has happened since SQL-92

SQL has evolved **beyond** the relational idea

If you use SQL for **CRUD** operations only, you are doing it wrong

modern SQL

https://modern-sql.com

@ModernSQL by @MarkusWinand

Training:
https://winand.at/

SQL has evolved
beyond
the relational idea

https://modern-sql.com
@ModernSQL by @MarkusWinand

Training:
https://winand.at/