The Mother of All Query Languages: SQL in Modern Times

SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

@MarkusWinand • @ModernSQL

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

SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

D. Chamberlin

ond F. Boyce

earch Laboratory

Jose, California

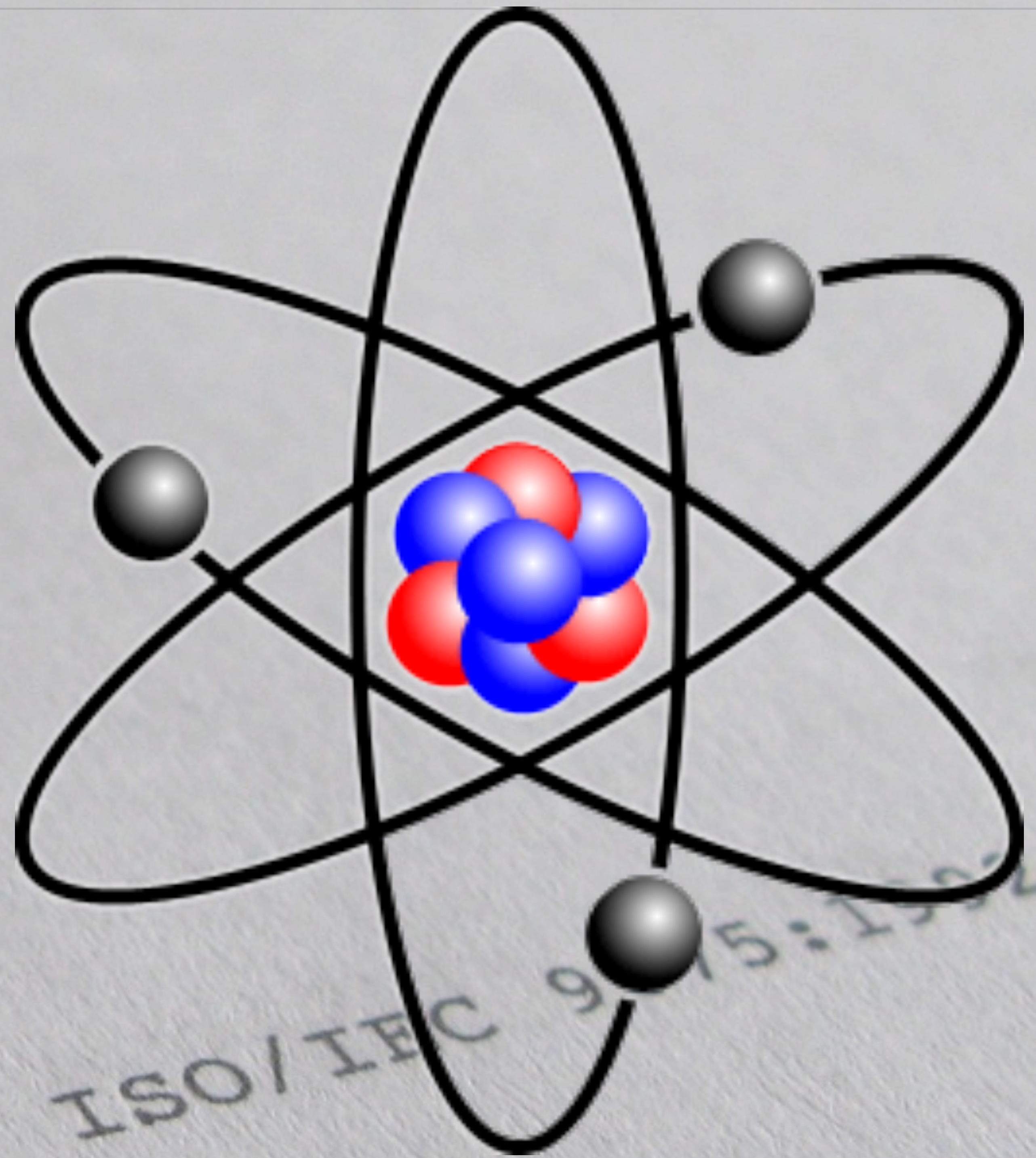
1974

1992

SQL-92 — Tied to the Relational Idea

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
⚛	⚛	⚛
⚛	⚛	
⚛	⚛	⚛

(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
⚛	⚛
⚛	⚛

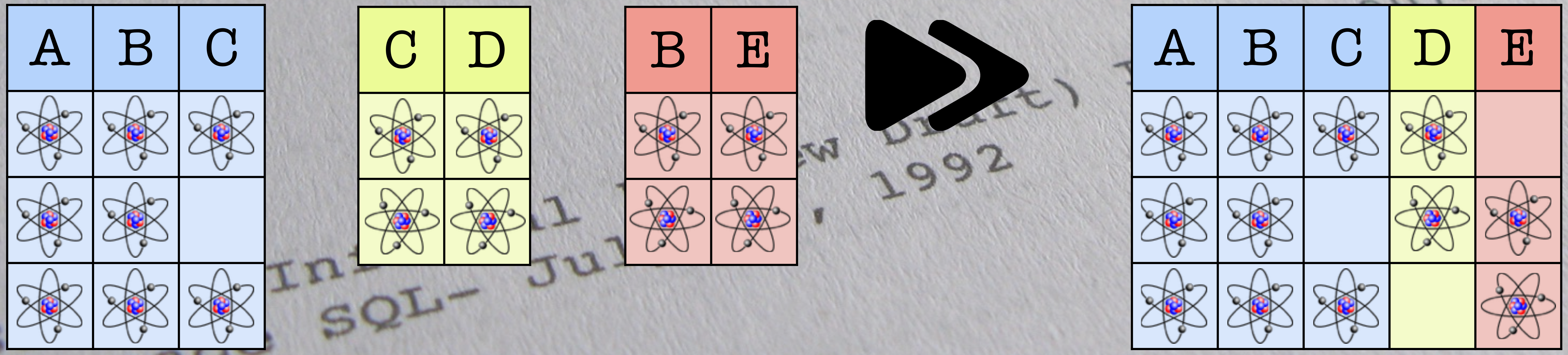
SQL-92 — Tied to the Relational Idea

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, …

A	B	C
⚛	⚛	⚛
⚛	⚛	
⚛	⚛	⚛

C	D
⚛	⚛
⚛	⚛

B	E
⚛	⚛
⚛	⚛

▶▶

A	B	C	D	E
⚛	⚛	⚛	⚛	
⚛	⚛		⚛	⚛
⚛	⚛	⚛		⚛

A	B	E
⚛	⚛	⚛
⚛	⚛	⚛

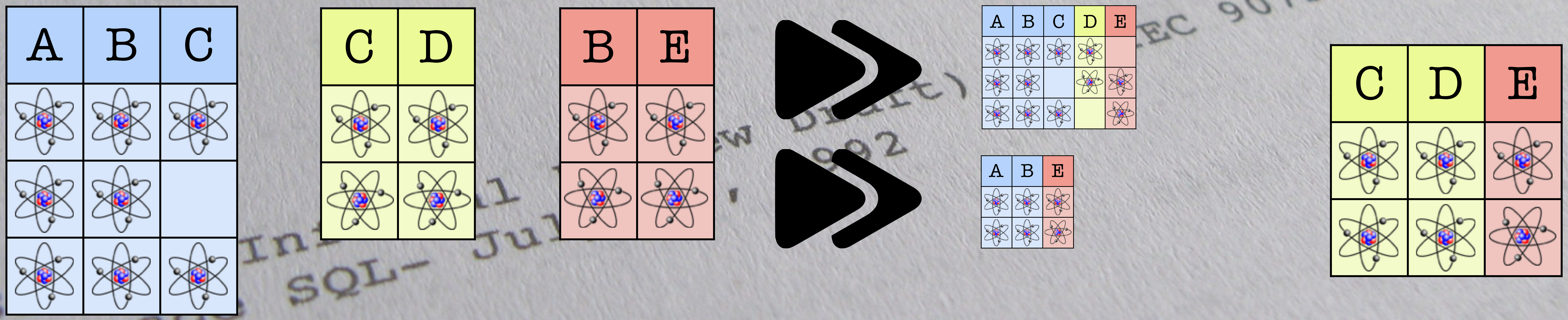
SQL-92 — Tied to the Relational Idea

Relational Data Model

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

Relational Operations

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



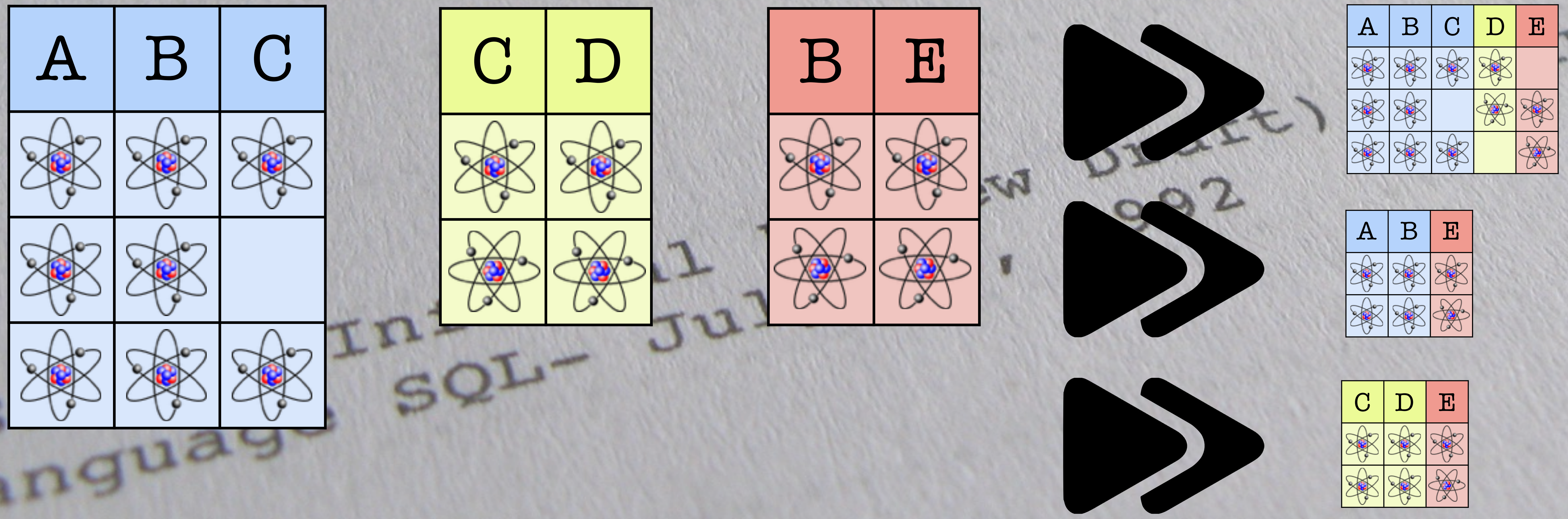
SQL-92 — Tied to the Relational Idea

Relational Data Model

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

Relational Operations

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



SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

D. Chamberlin

nd F. Boyce

earch Laboratory

Jose, California

1992

1999

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.

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.

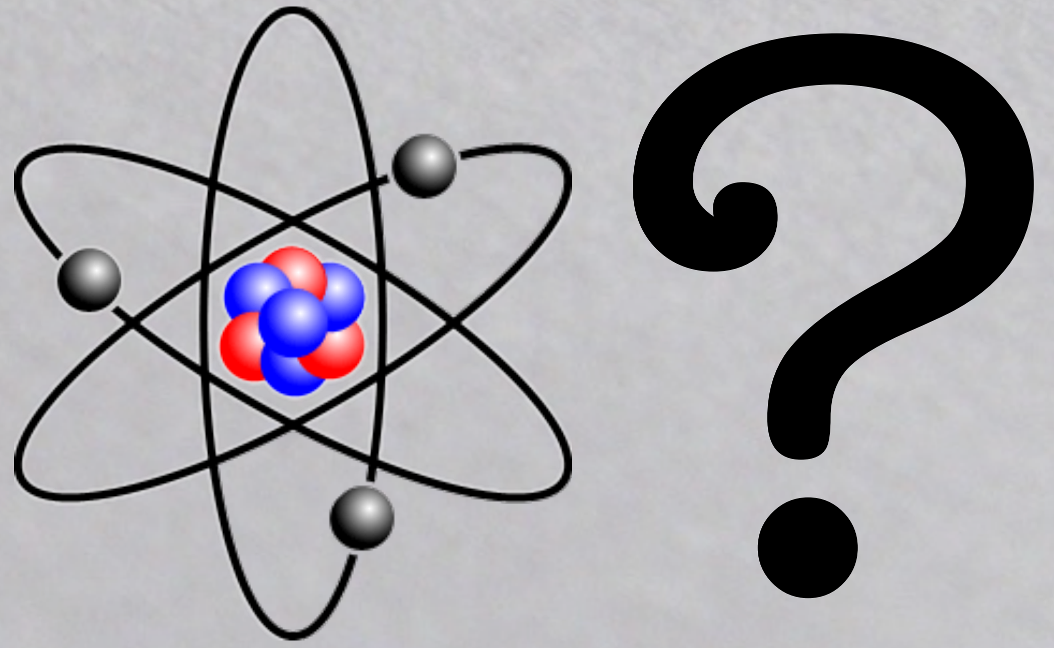
https://www.wiscorp.com/DBMS_-_GreatNews-TheRelationalModelIsDead_-_paper_-_sam.pdf

SQL:1999 — Escaping the Relational Cage

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

Relational Model?

Information Systems Corporation

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

Information Systems Corporation

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
  - arrays

A	B
⚛	[⚛,⚛]
⚛	[⚛]
⚛	[]

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

Domains Can Contain Anything!

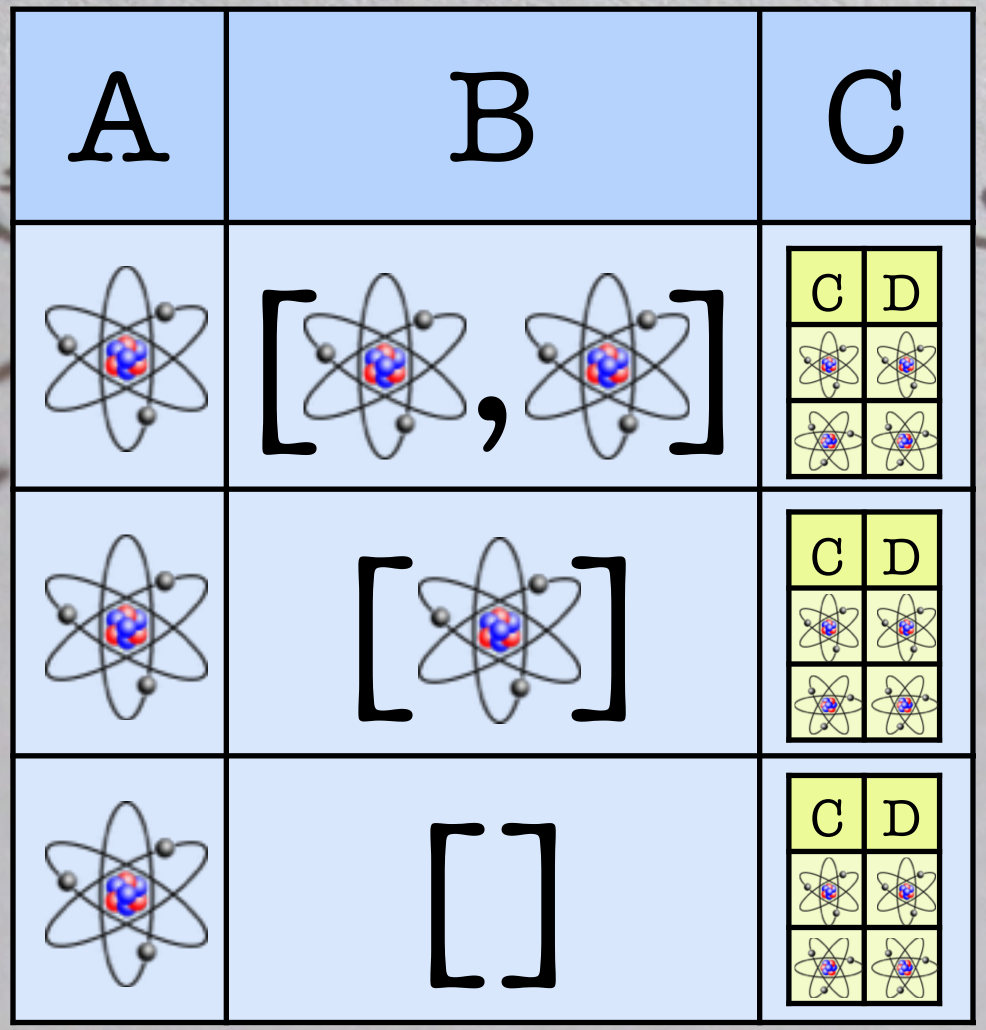
Date on Database: Writings 2000-2006

Chris Date

SQL:1999 — Escaping the Relational Cage

Relational Model?

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



A	B	C
⚛	[⚛,⚛]	C D ⚛ ⚛ ⚛ ⚛
⚛	[⚛]	C D ⚛ ⚛ ⚛ ⚛
⚛	[]	C D ⚛ ⚛ ⚛ ⚛

> 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

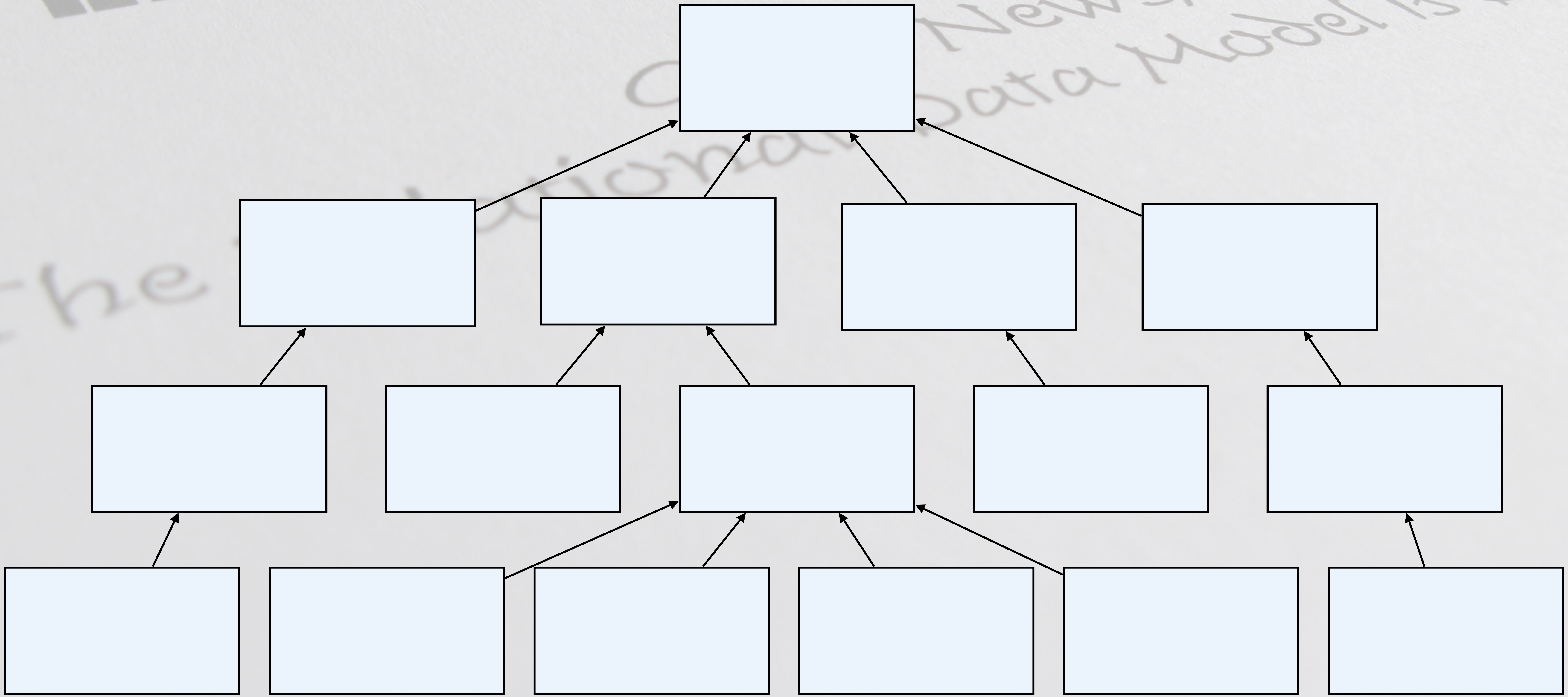
- Introduced recursive queries that process their own output
  - Transitive closure

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

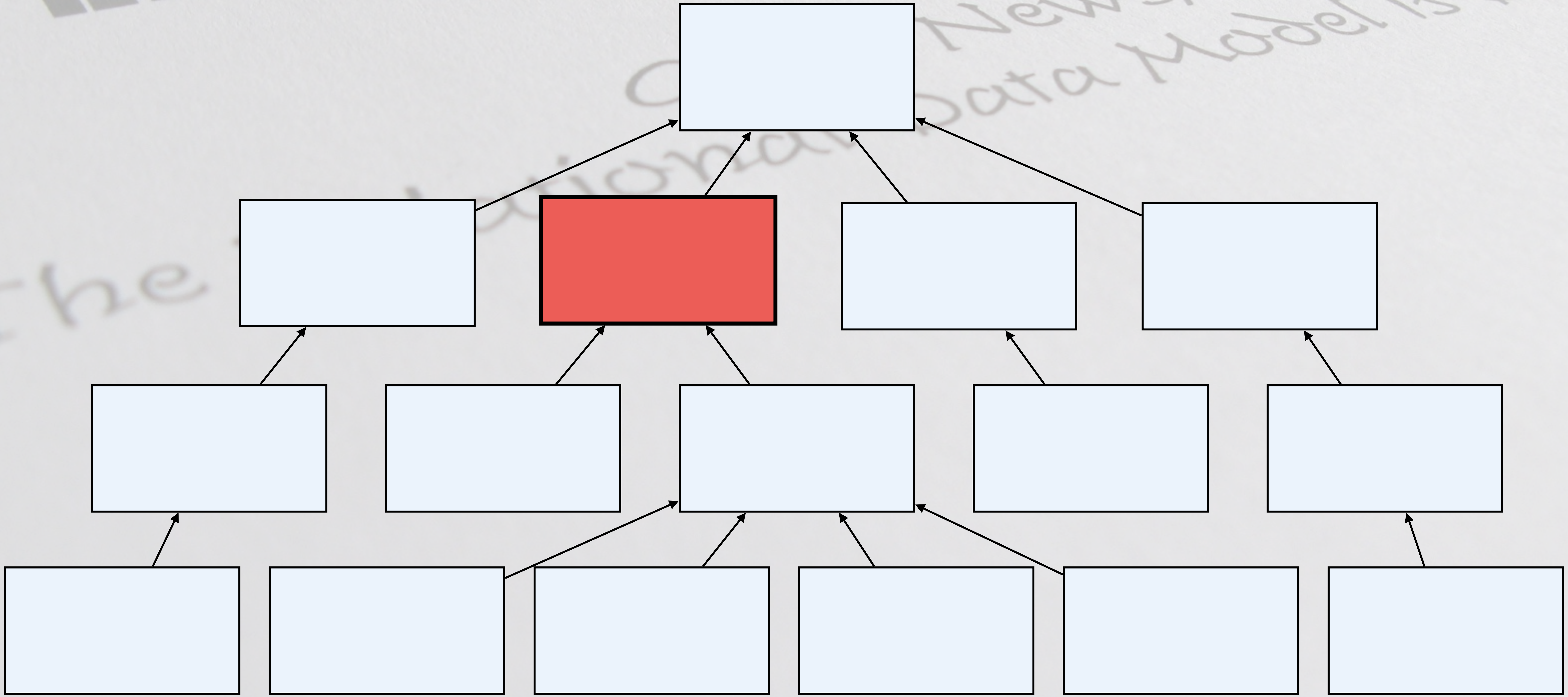
Date on Database: Writings 2000-2006

Chris Date

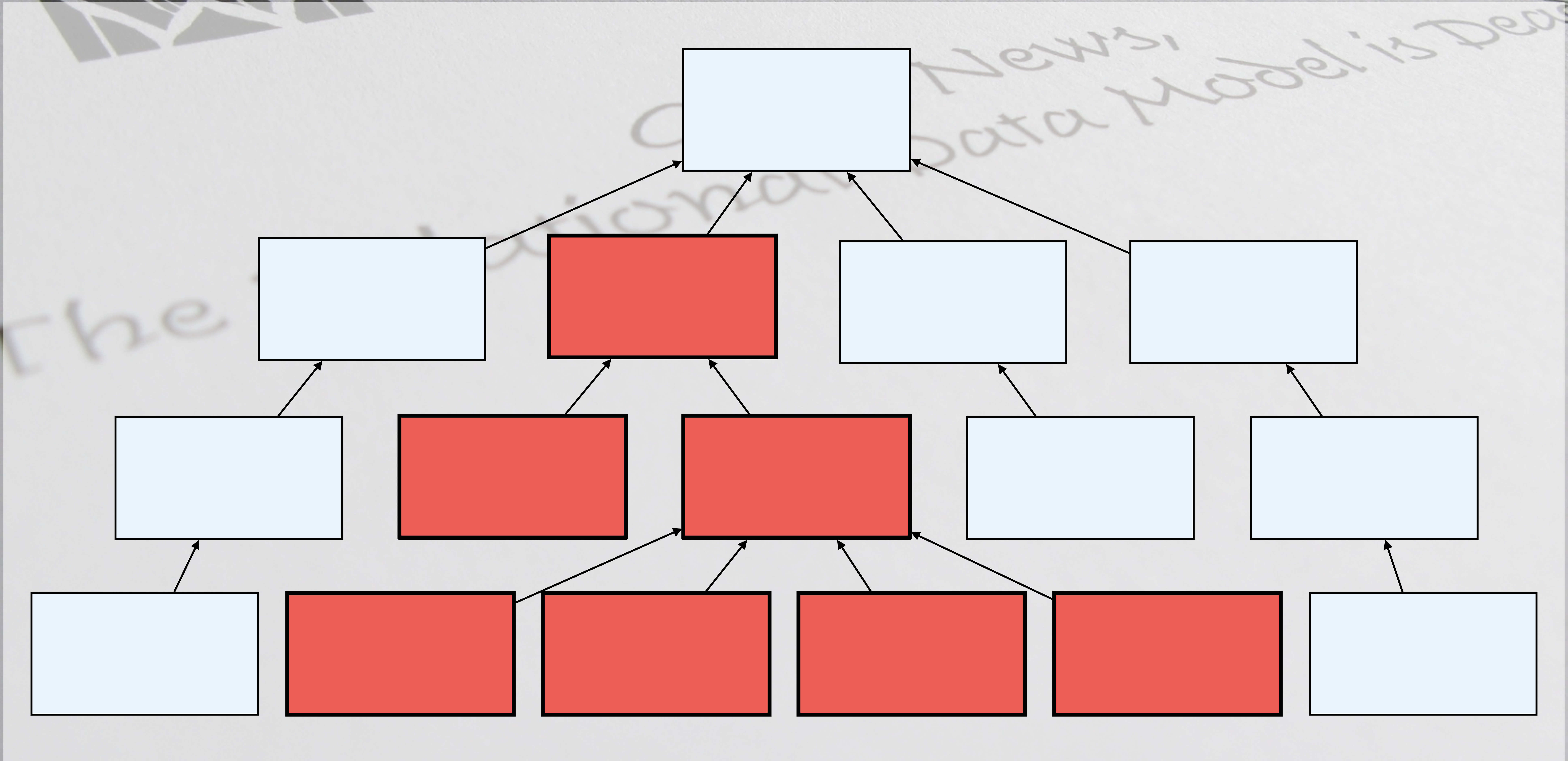
SQL:1999 — Recursion



SQL:1999 — Recursion

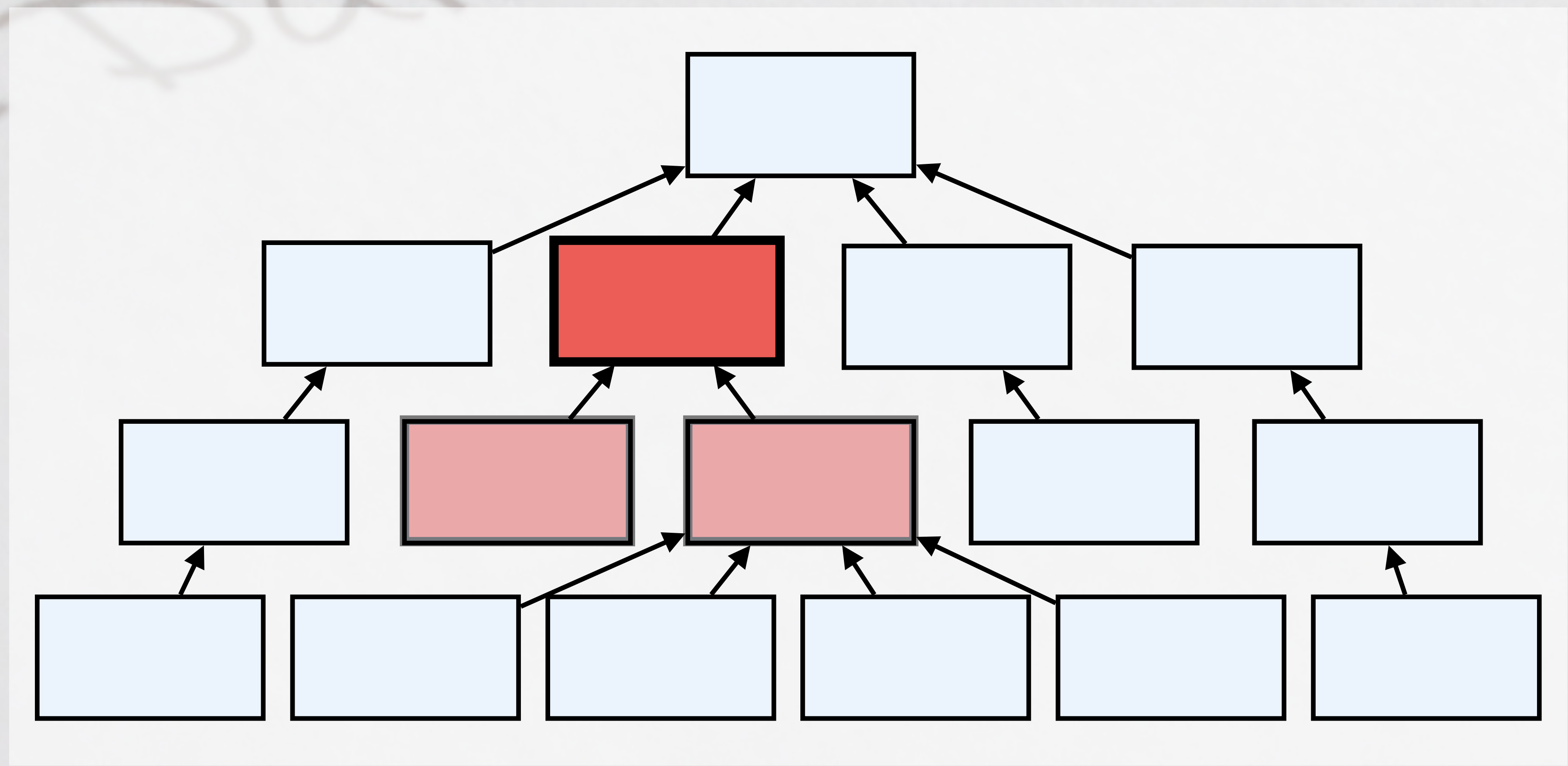


SQL:1999 — Recursion



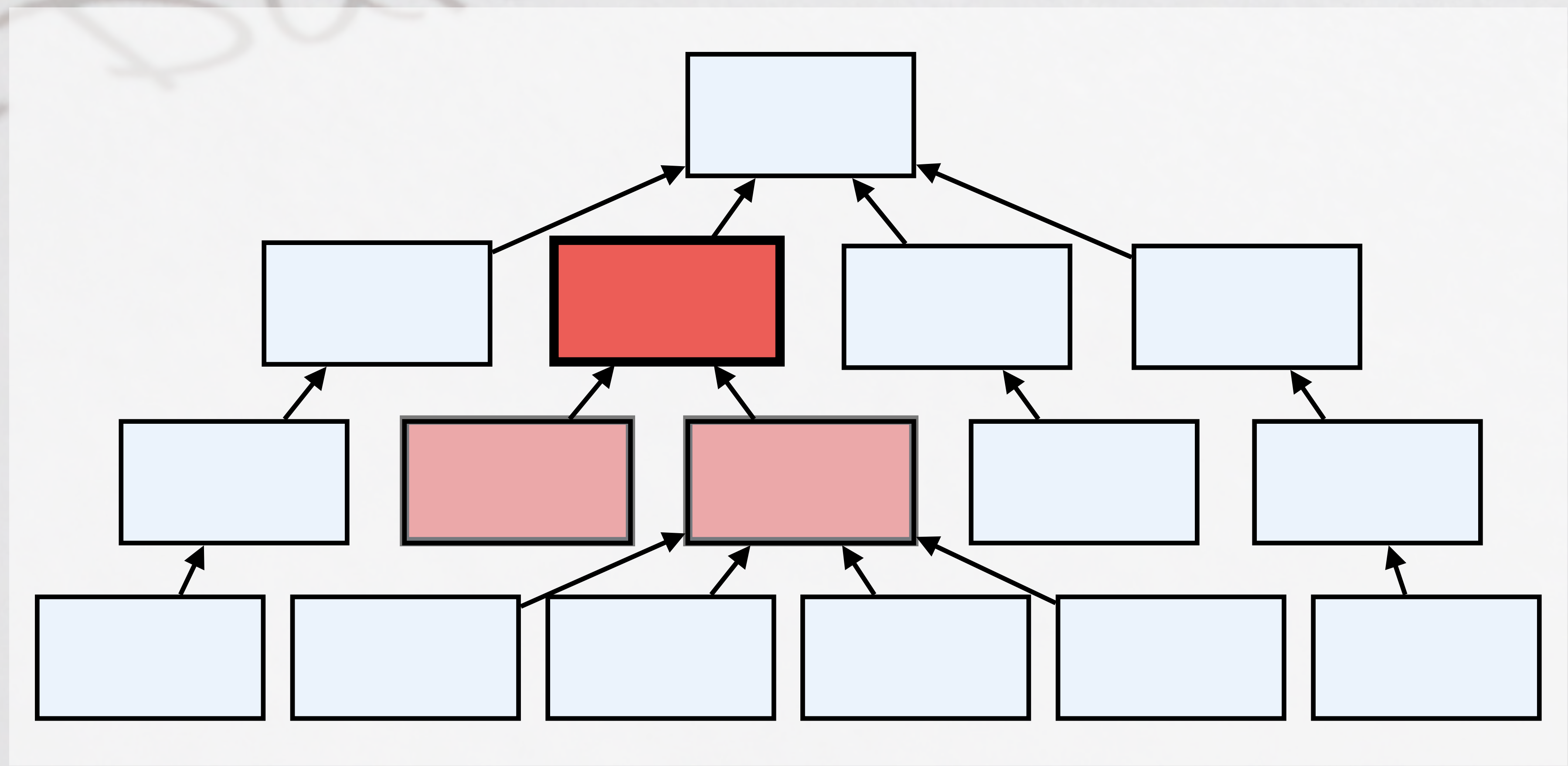
SQL:1999 — Recursion

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



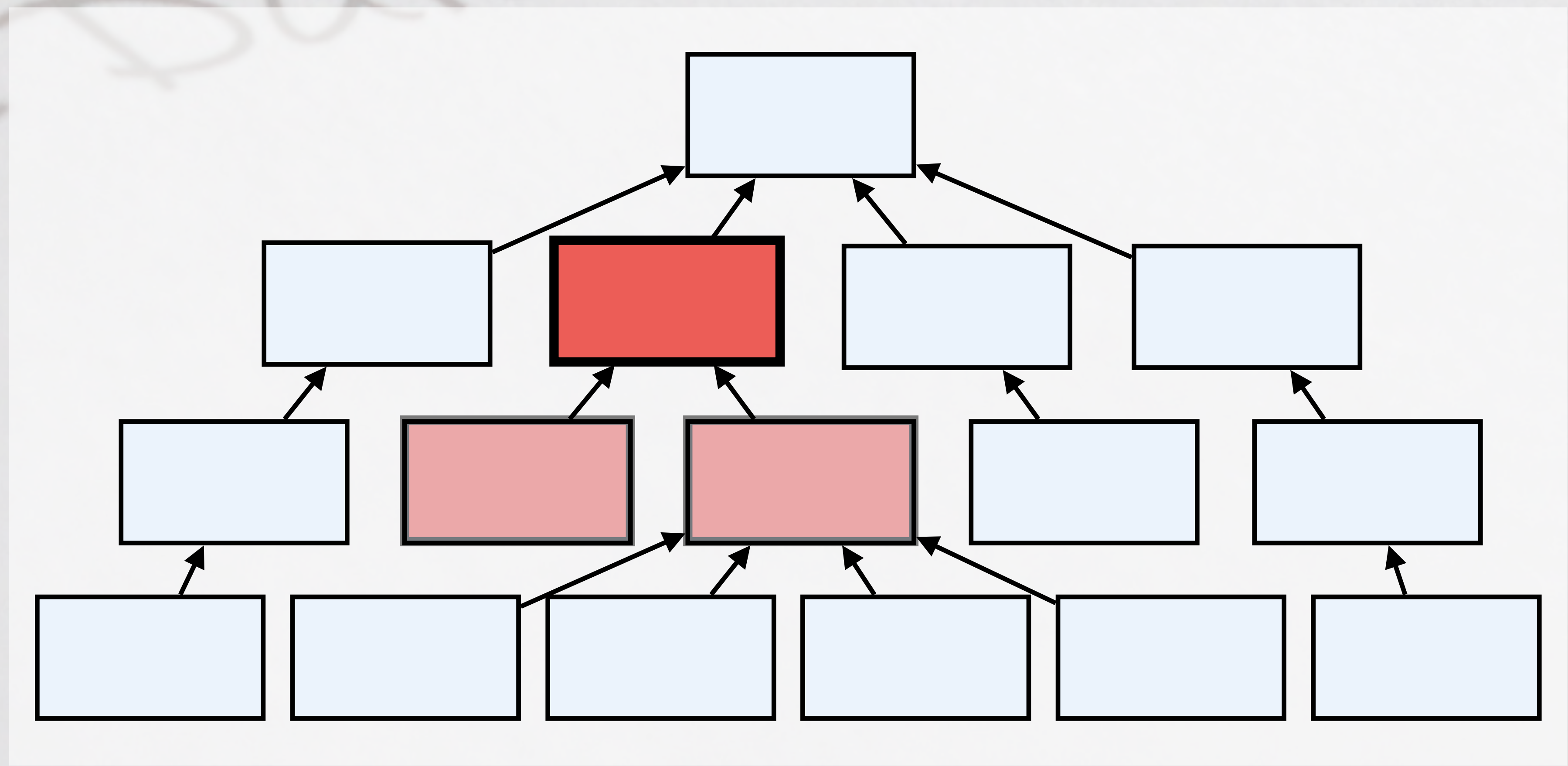
SQL:1999 — Recursion

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



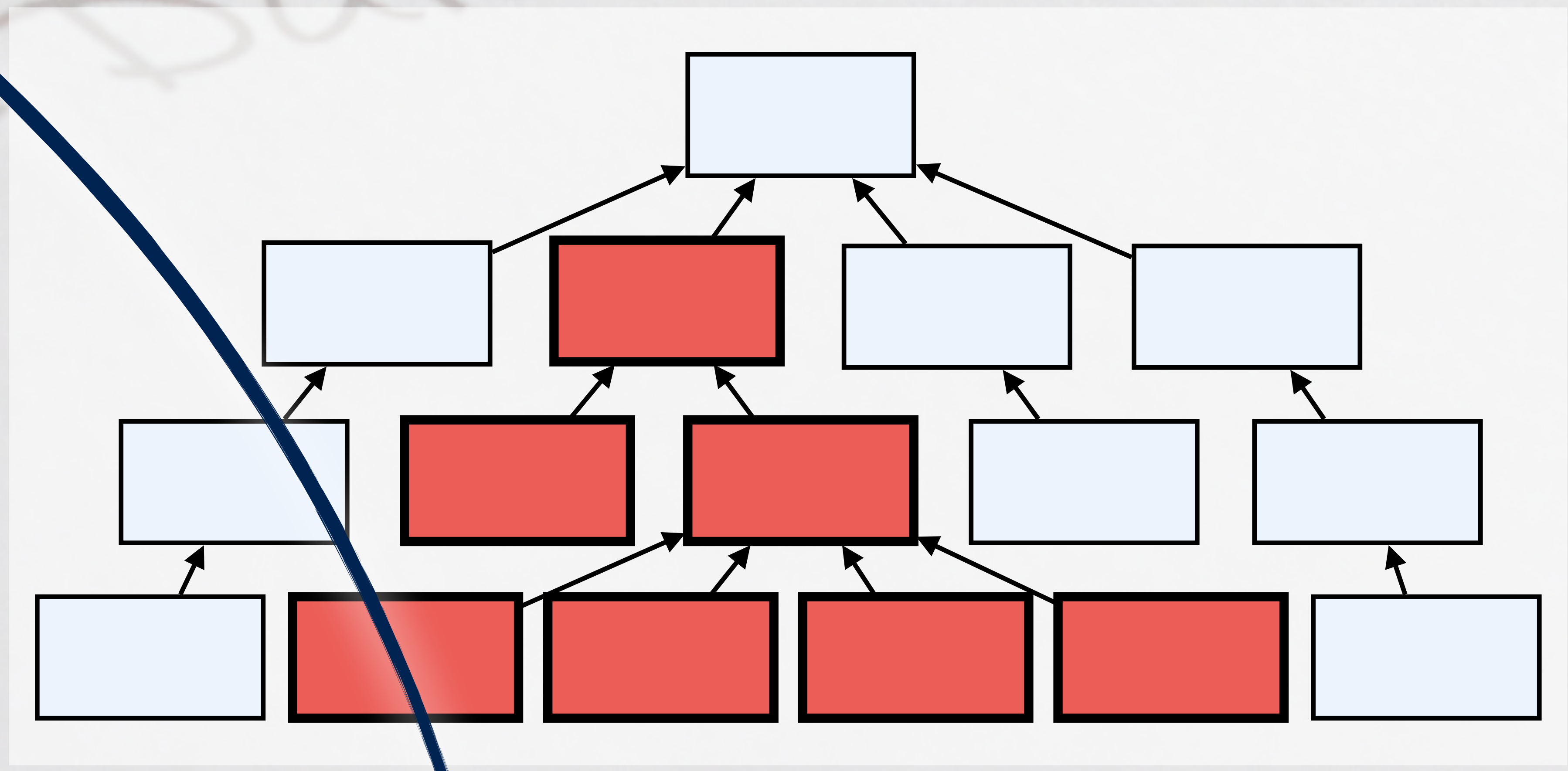
SQL:1999 — Recursion

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

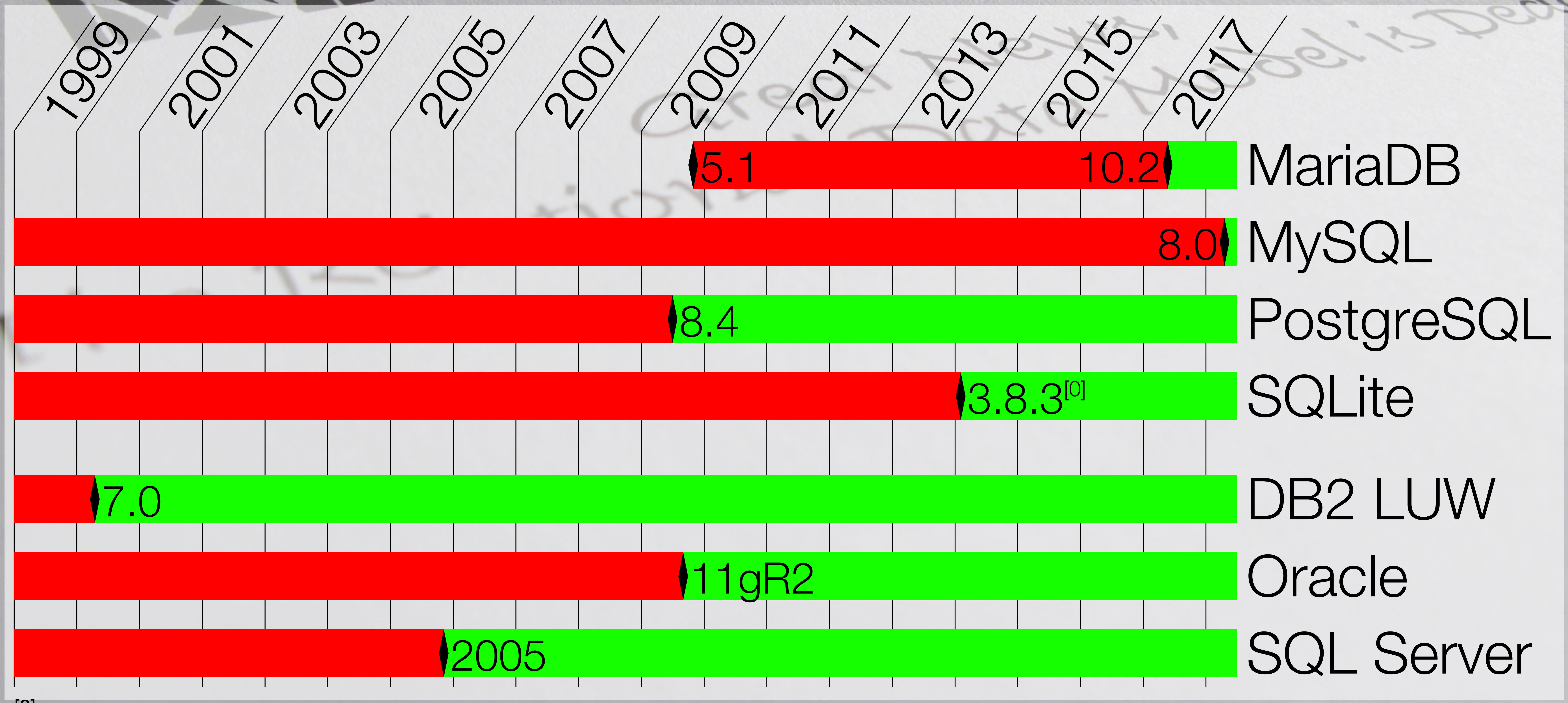


SQL:1999 — Recursion

```
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



[0]Only for top-level SELECT statements

Whitemarsh

Information Systems Corporation

G[illegible]at News,

The Rel[illegible]a Data Model is Dead!

1999 ⏩ 2016

SQL:2016 — JSON

TECHNICAL REPORT

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

techniques —

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 — Part de SQL pour JavaScript Object No

```
[
  {
    "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

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

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

id	a1
42	foo
43	bar

SQL:2016 — JSON

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

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

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

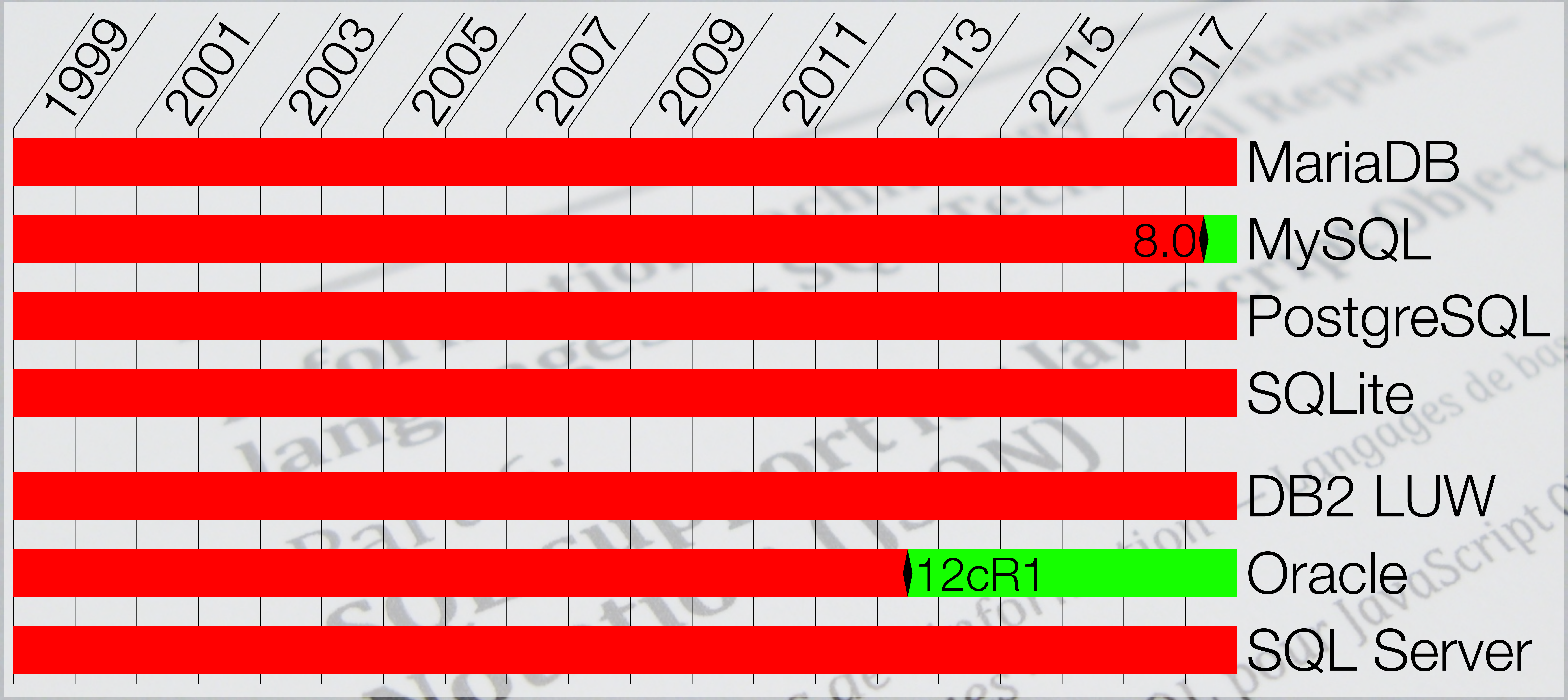
<u>SQL/JSON Path</u>

- 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



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