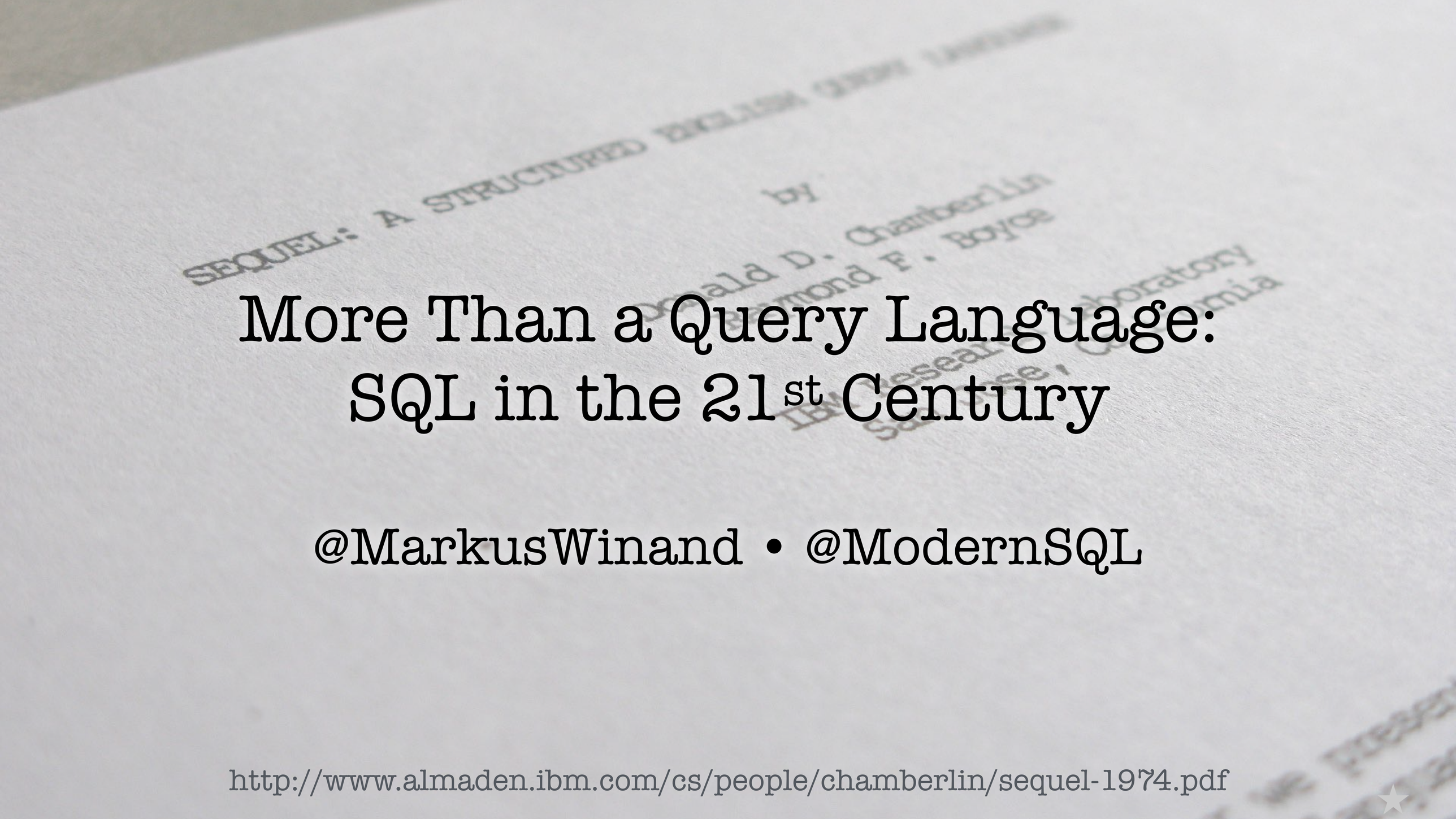
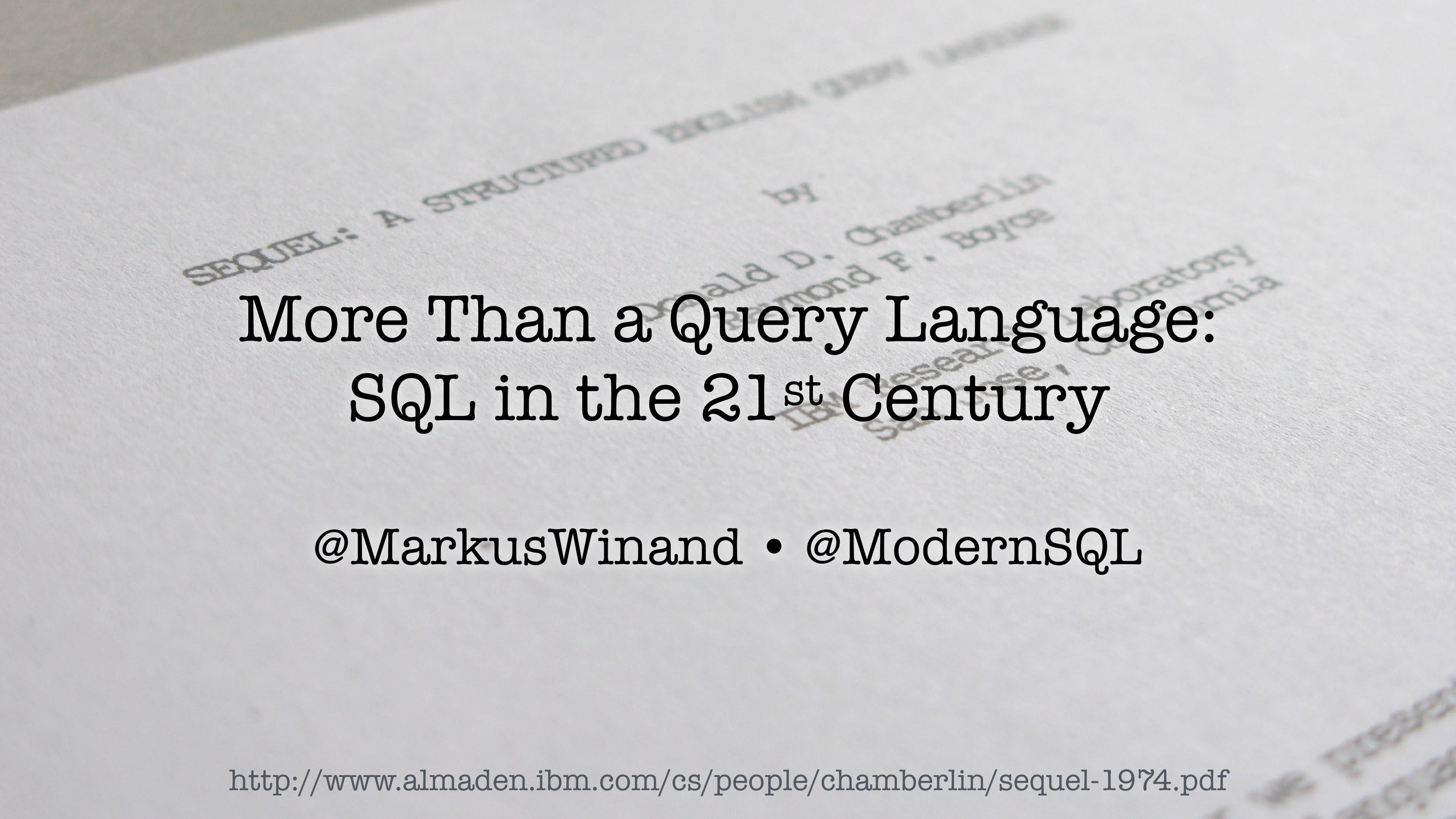



# More Than a Query Language: SQL in the 21st Century

@MarkusWinand • @ModernSQL

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



# More Than a Query Language: SQL in the 21st Century

@MarkusWinand • @ModernSQL

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

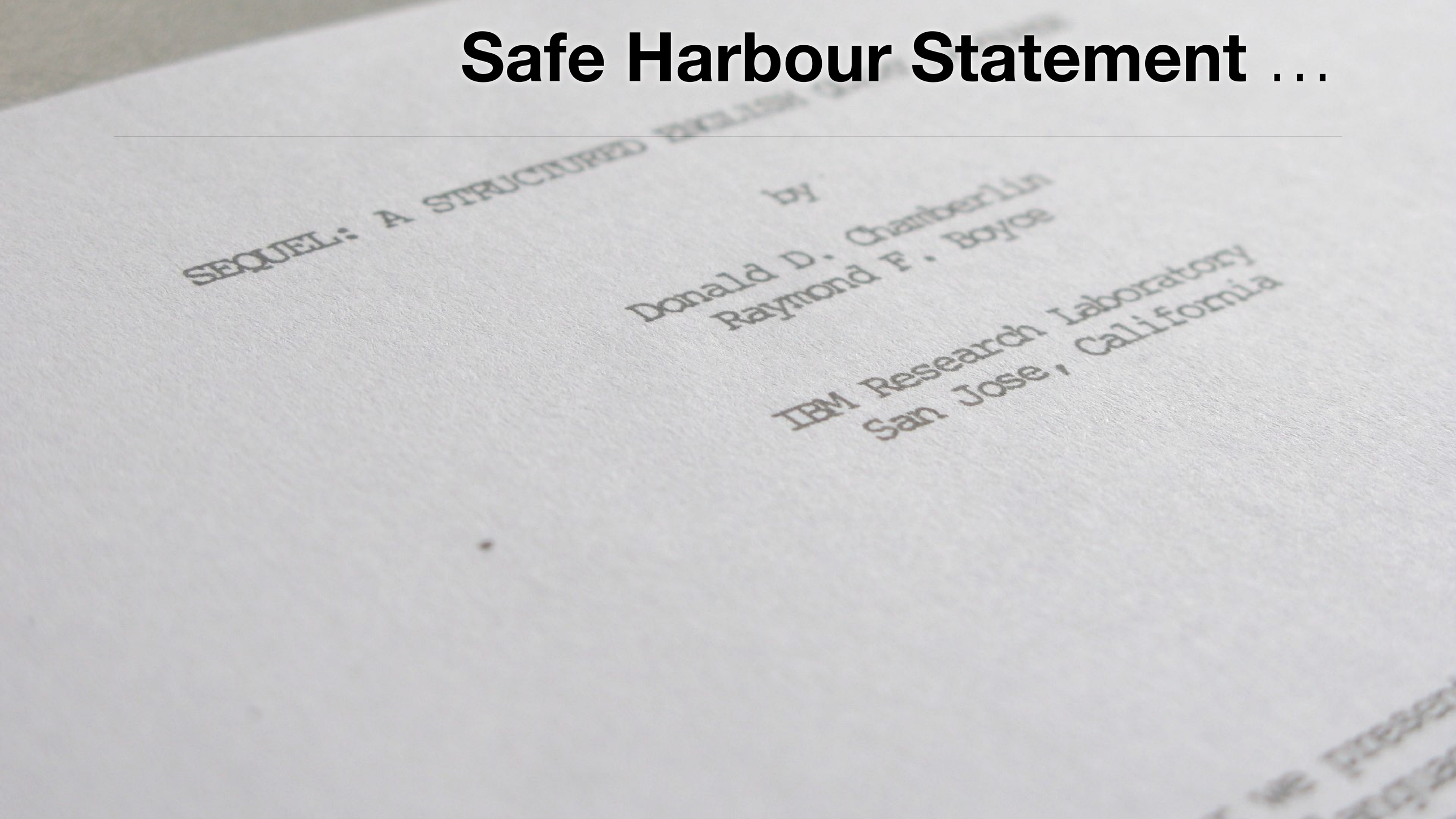
# SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

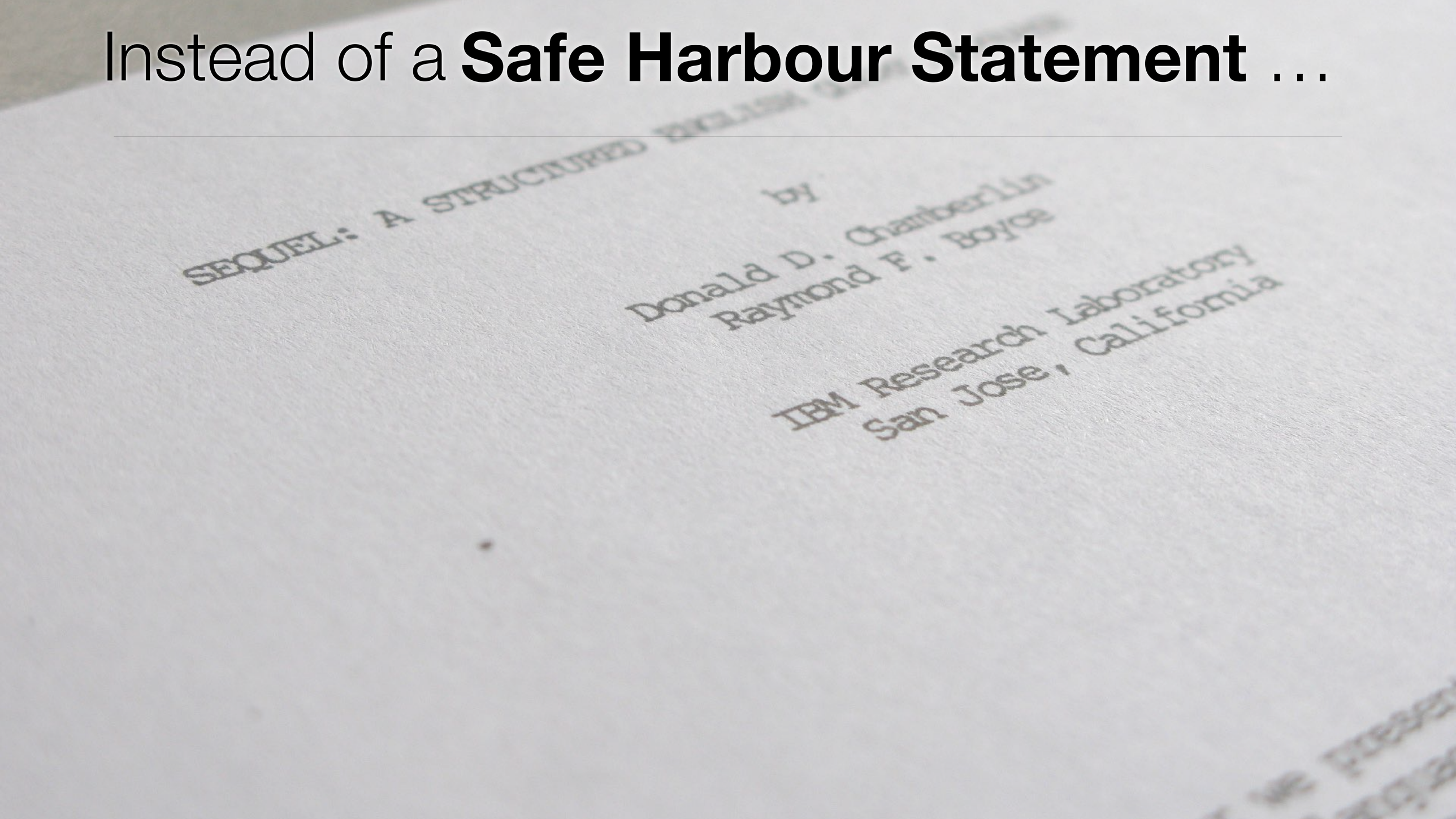
Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

# Safe Harbour Statement …



# Instead of a **Safe Harbour Statement** …



# Instead of a **Safe Harbour Statement** …

## Take this **Safe the Planet Statement:**

**I’m traveling a lot for business and cause more than my fair share of climate-damaging emissions on this planet.**

**I prefer climate-neutral means of transport**
to lower my contribution to climate change.

**If I cannot avoid flying, I offset its climate impact at a transparent climate protection company.**
(currently atmosfair.de)

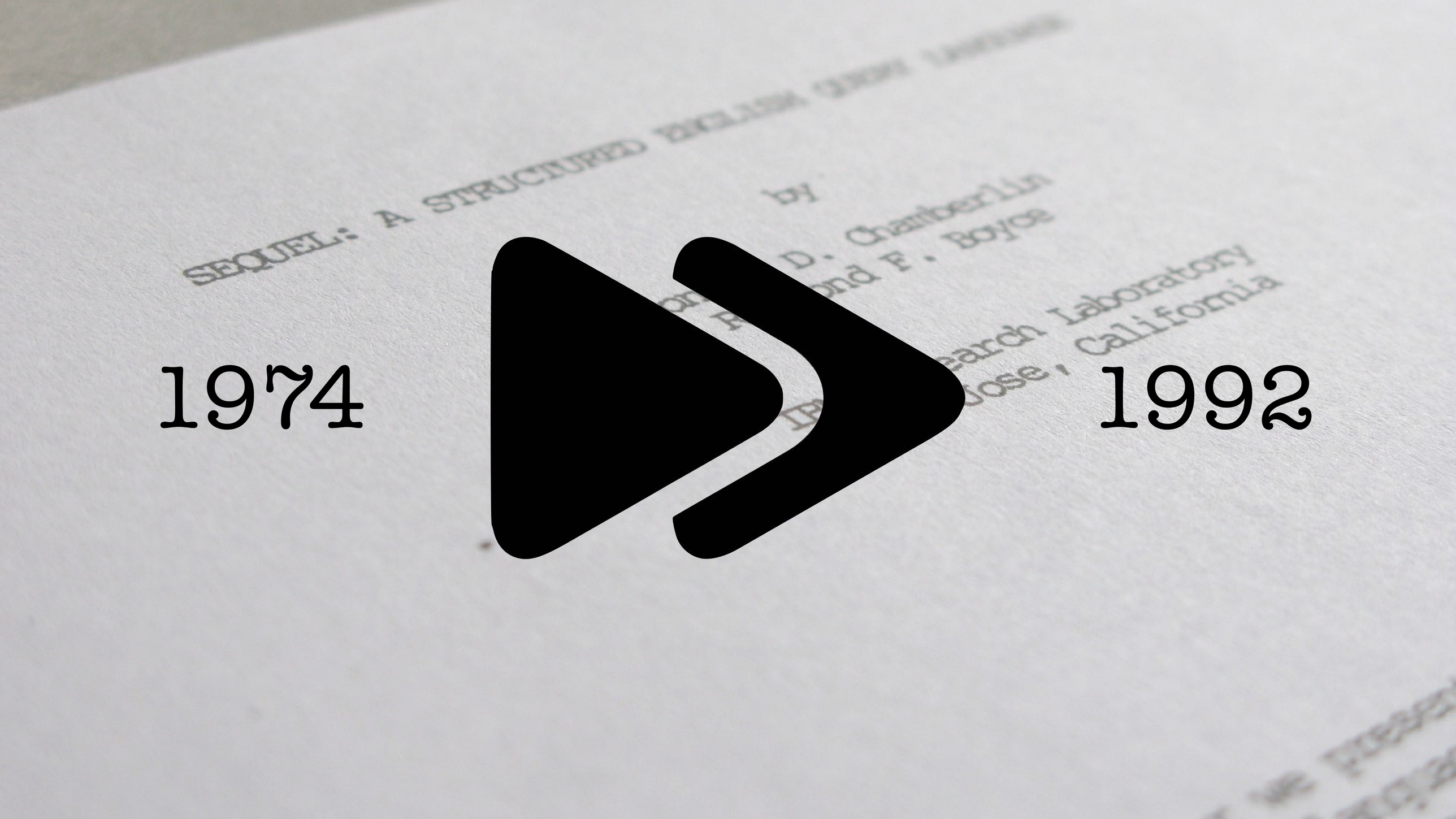
Picture: https://upload.wikimedia.org/wikipedia/commons/2/2c/North_America_from_low_orbiting_satellite_Suomi_NPP.jpg

# SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

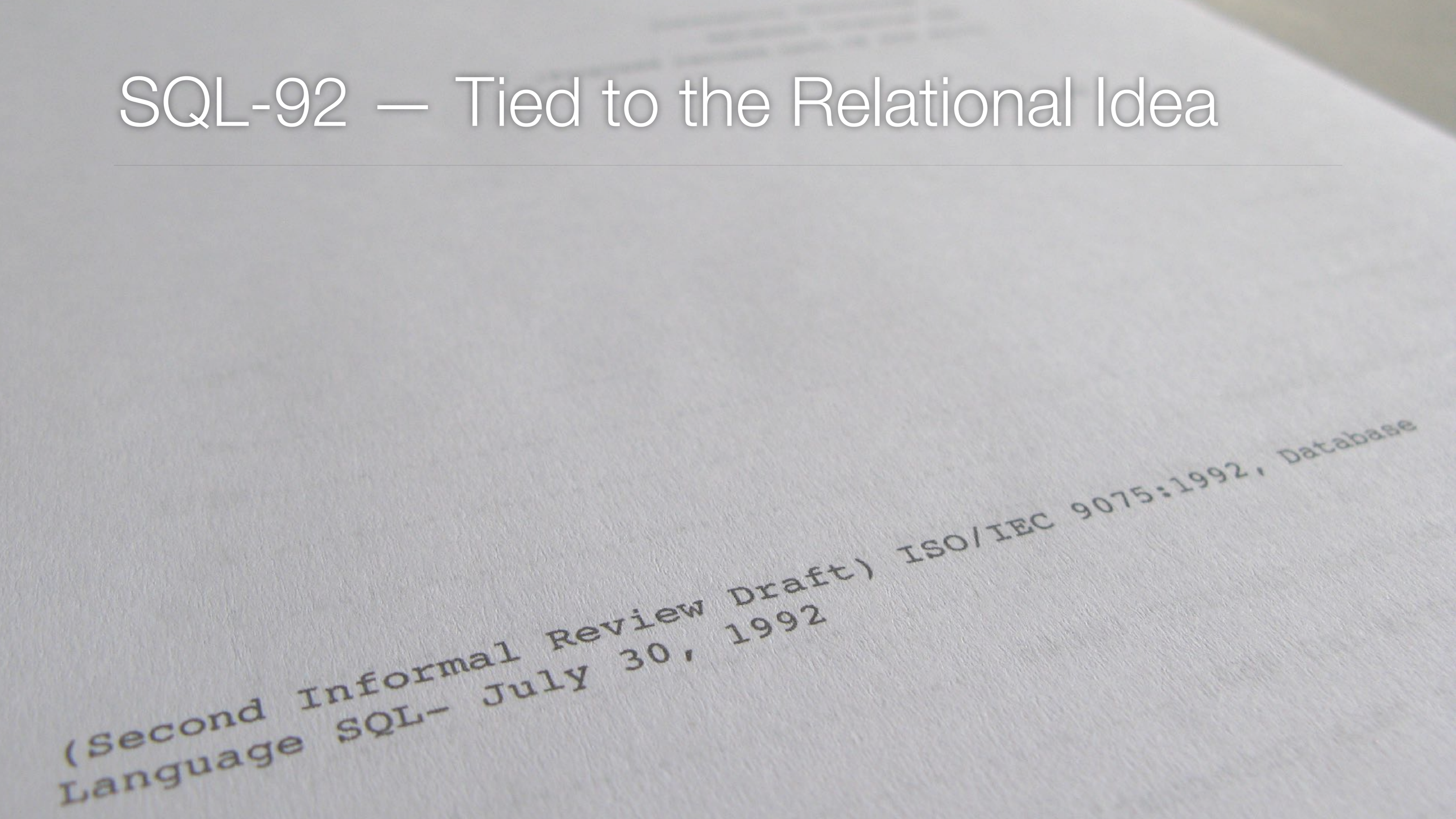
by

Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California


SEQUEL: A STRUCTURED
by
D. Chamberlin
F. Boyce
Research Laboratory
San Jose, California
1974
1992

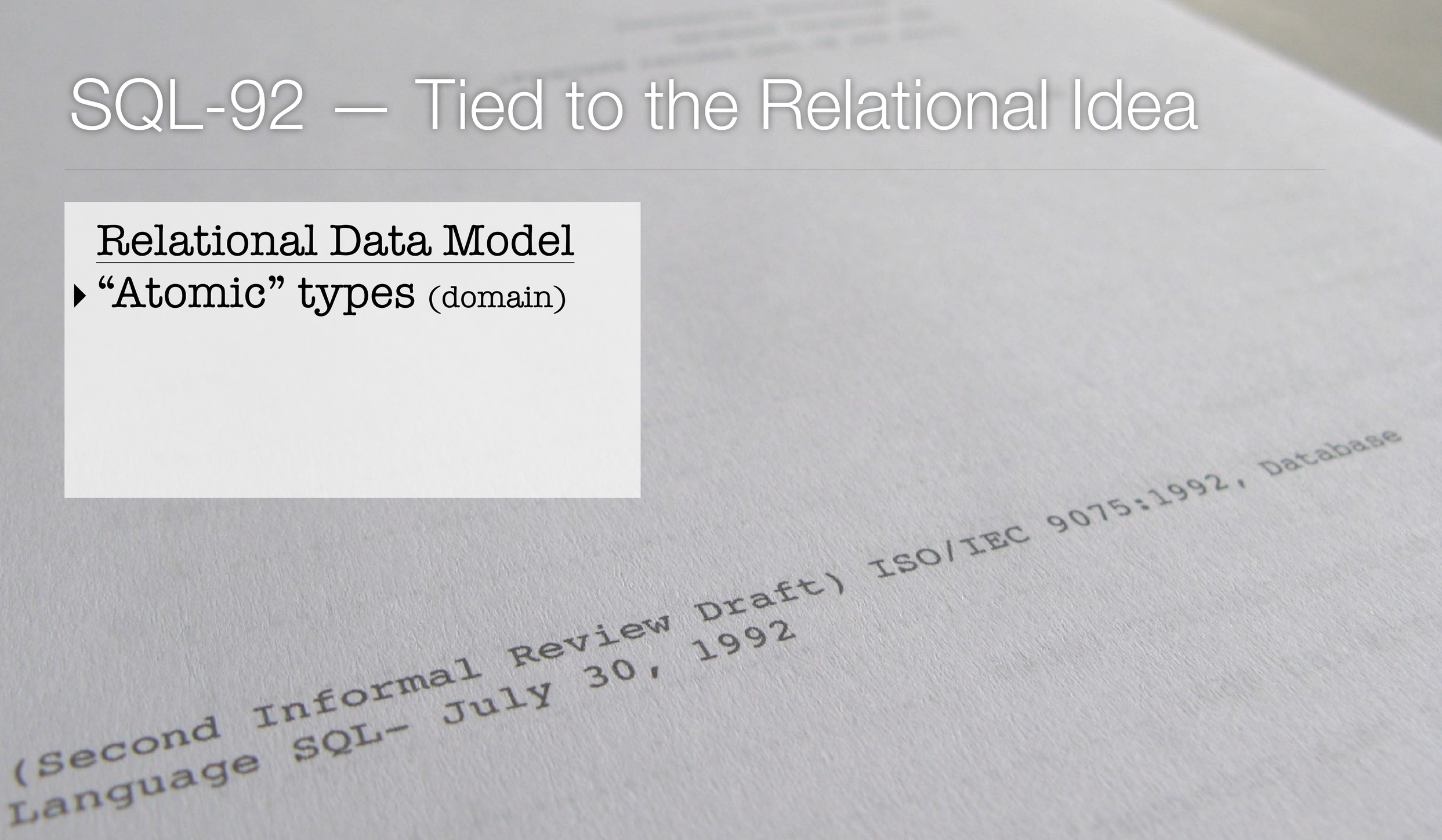
# SQL-92 — Tied to the Relational Idea



# SQL-92 — Tied to the Relational Idea

Relational Data Model

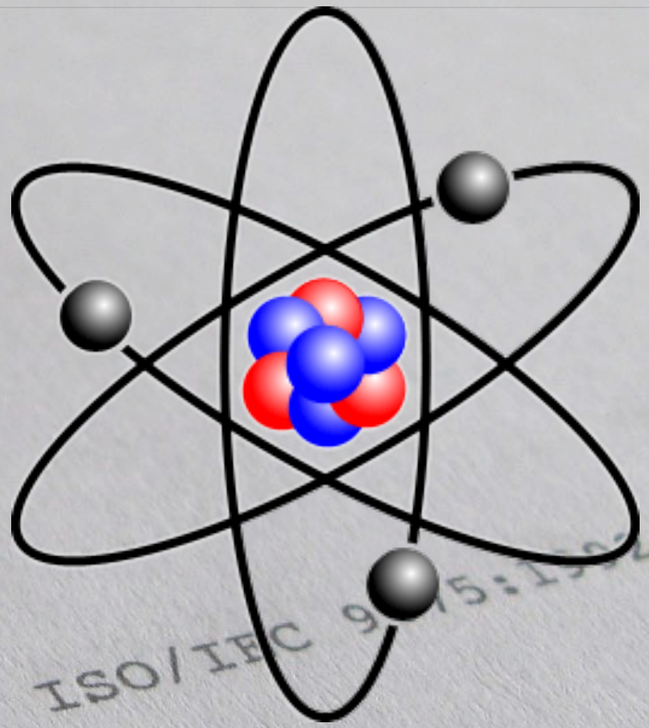
- “Atomic” types (domain)



# SQL-92 — Tied to the Relational Idea

Relational Data Model

- “Atomic” types (domain)

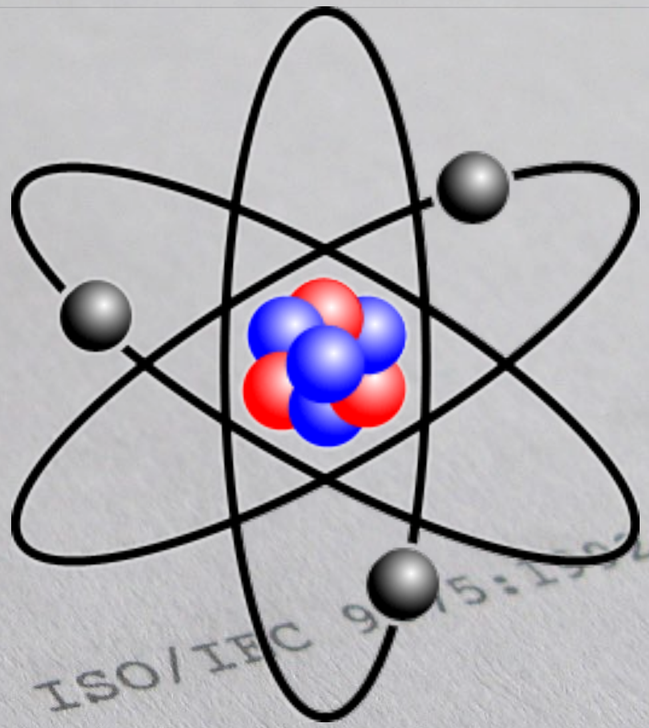


Atom image: https://commons.wikimedia.org/wiki/File:Stylised_atom_with_three_Bohr_model_orbits_and_stylised_nucleus.png

# SQL-92 — Tied to the Relational Idea

## Relational Data Model

- “Atomic” types (domain)



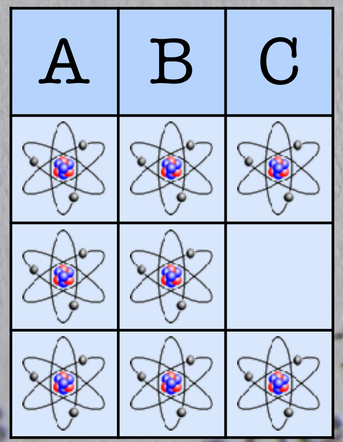
| A | B | C |
|---|---|---|
| | | |
| | | |
| | | |

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)



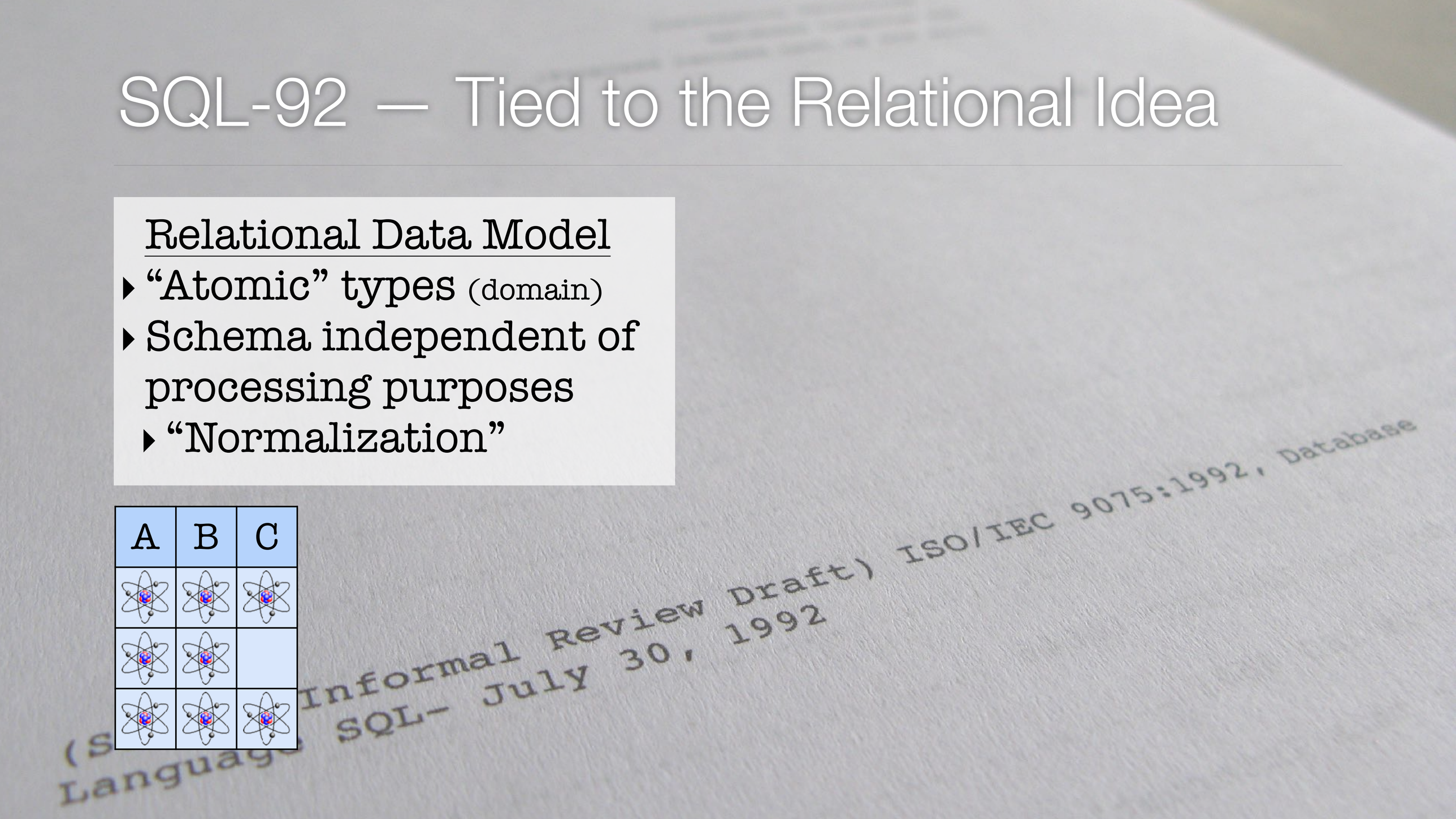

(S Informal Review Draft) ISO/IEC 9075:1992, Database Language SQL- July 30, 1992

# SQL-92 — Tied to the Relational Idea

Relational Data Model

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

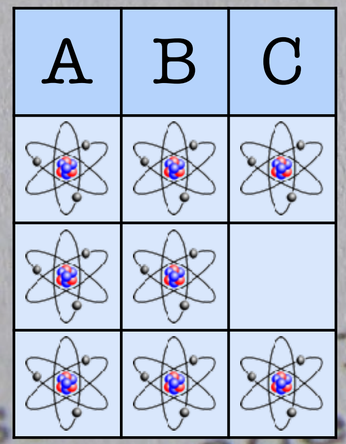
| A | B | C |
|---|---|---|
| ⚛ | ⚛ | ⚛ |
| ⚛ | ⚛ | |
| ⚛ | ⚛ | ⚛ |

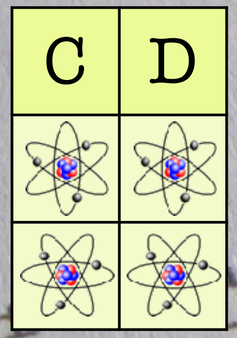


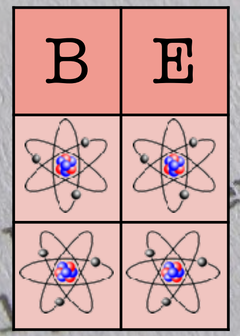
# SQL-92 — Tied to the Relational Idea

Relational Data Model

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



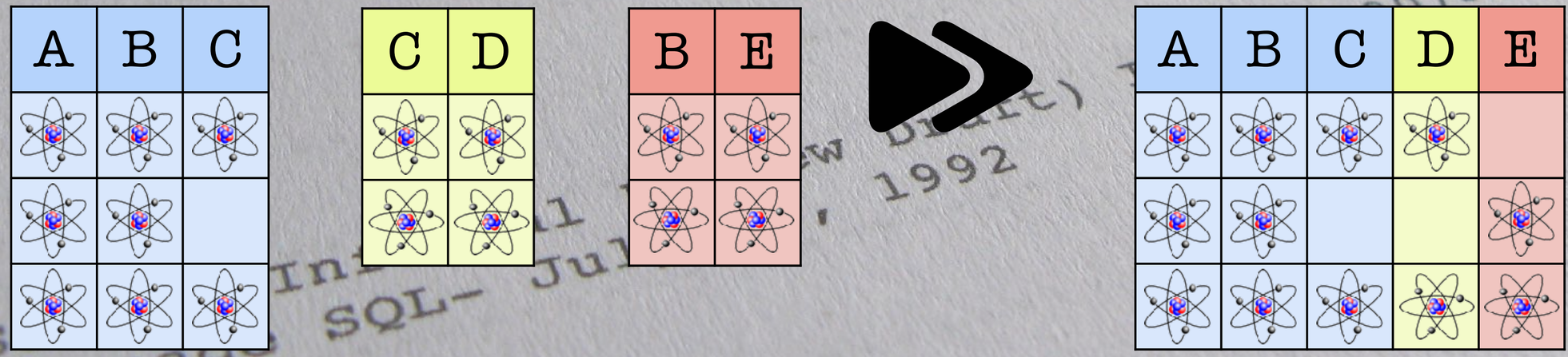
# SQL-92 — Tied to the Relational Idea

**Relational Data Model**

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

**Relational Operations**

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



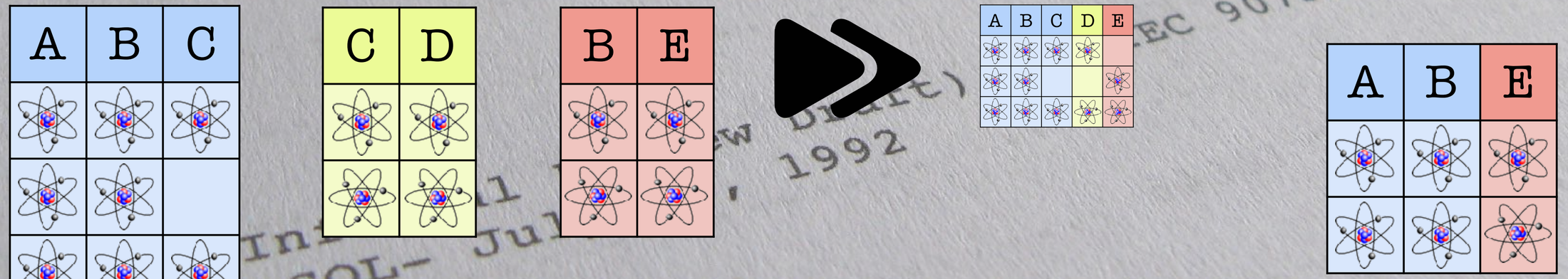
# SQL-92 — Tied to the Relational Idea

## Relational Data Model

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

## Relational Operations

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



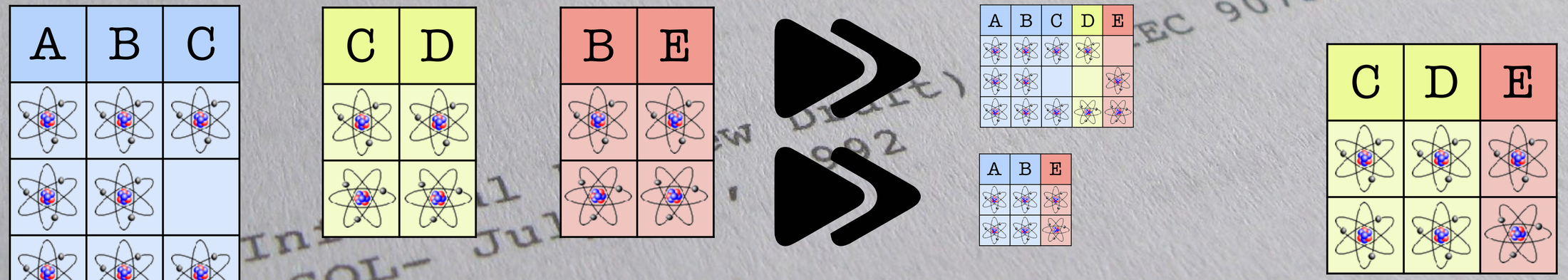
# SQL-92 — Tied to the Relational Idea

## Relational Data Model

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

## Relational Operations

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



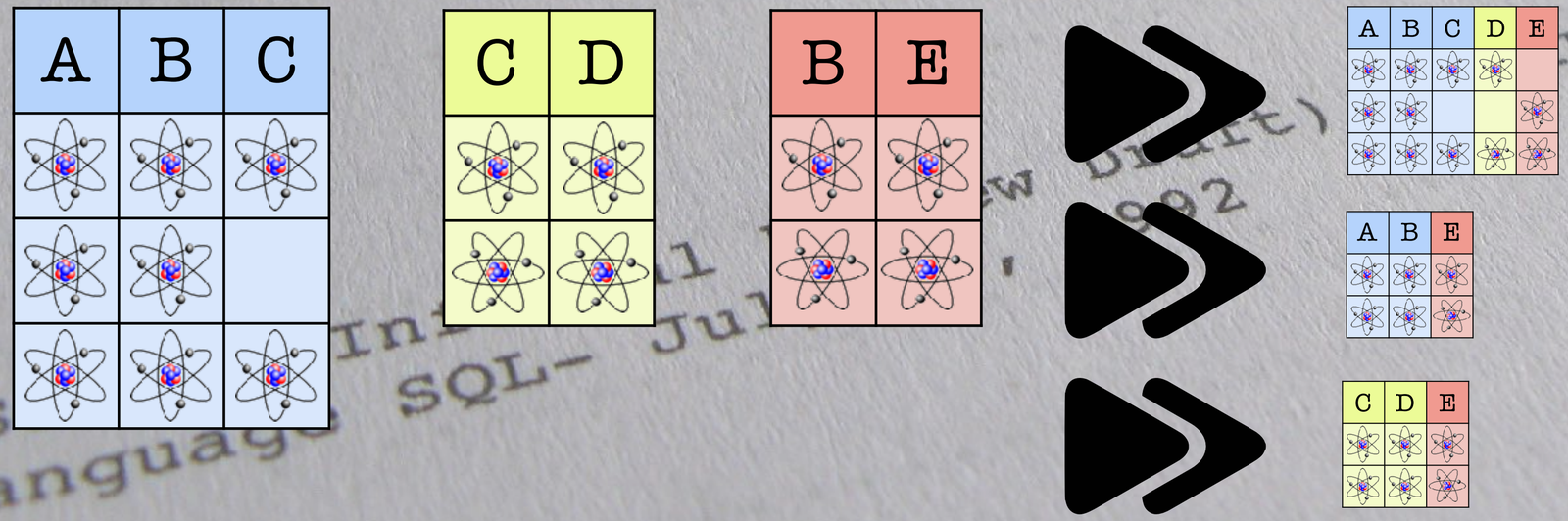
# SQL-92 — Tied to the Relational Idea

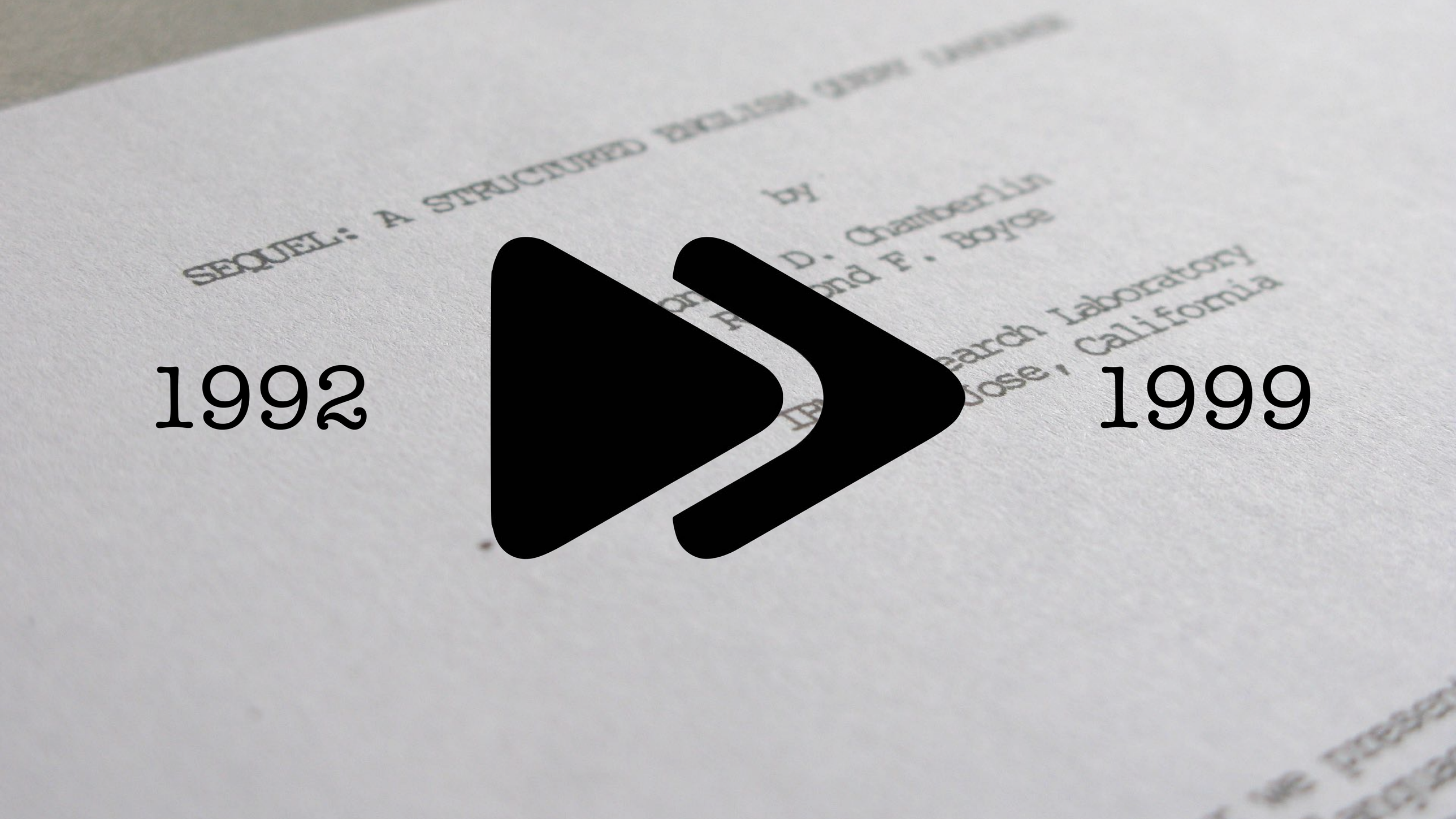
## Relational Data Model

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

## Relational Operations

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




SEQUEL: A STRUCTURED
by
D. Chamberlin
F. Boyce
Research Laboratory
Jose, California
1992
1999



Lubitemarsh
Information Systems Corporation

# The Relational Data Model is Dead!

Great News,

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.

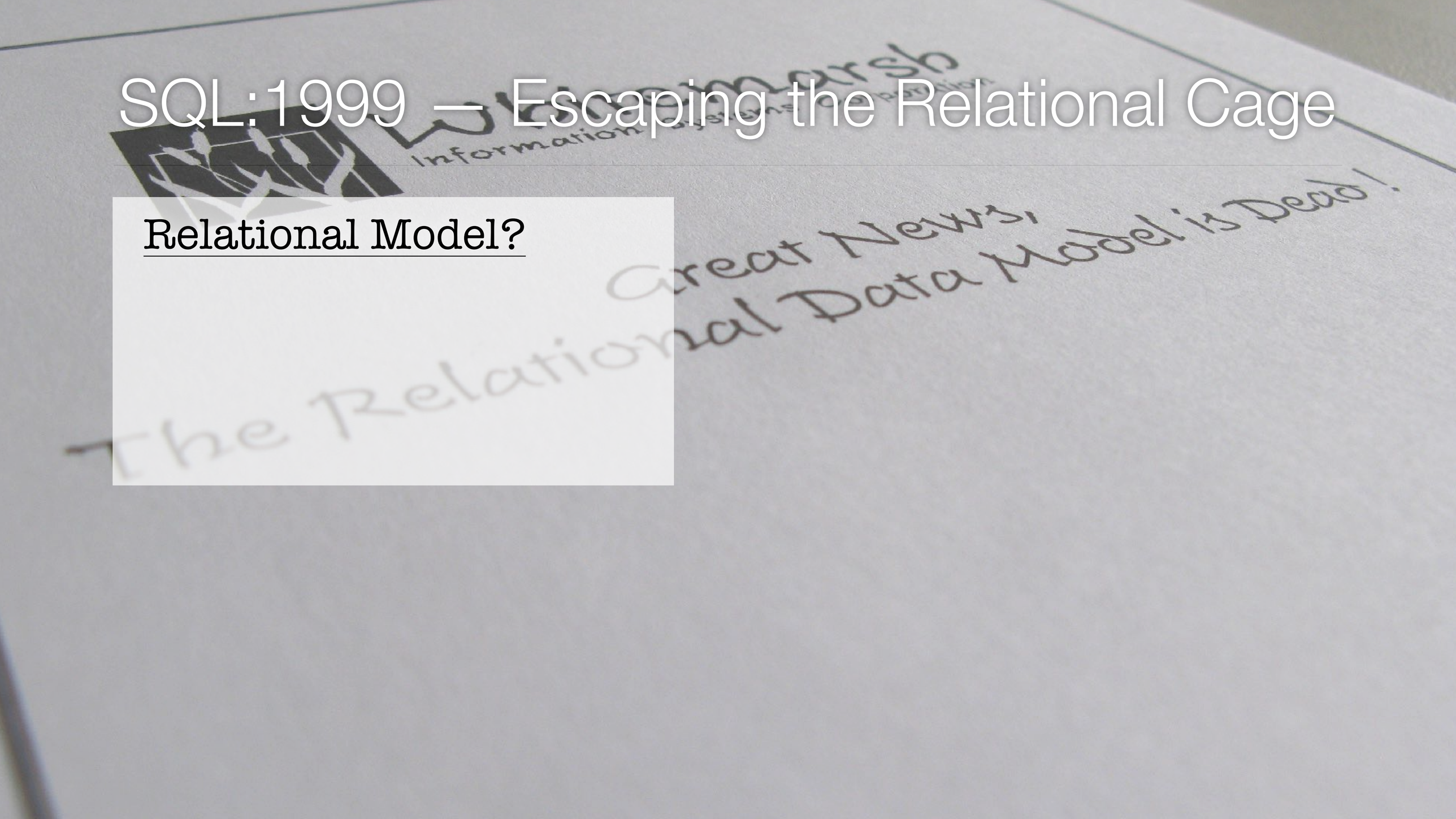
With SQL/99 you can get the best of both worlds and of course, you can get the worst of both worlds. It's up to the database practitioners to do the right thing.

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

# SQL:1999 — Escaping the Relational Cage



# SQL:1999 — Escaping the Relational Cage




## Relational Model?

# SQL:1999 — Escaping the Relational Cage

Relational Model?

I was as confused as anyone else

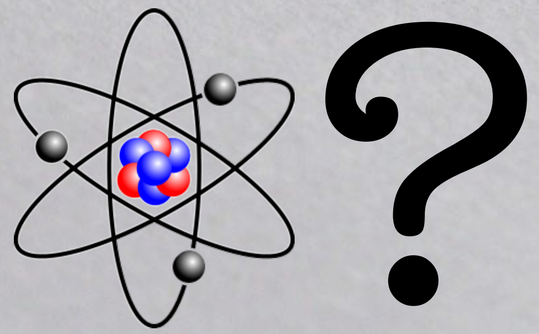


Chris Date

Date on Database: Writings 2000-2006

# SQL:1999 — Escaping the Relational Cage

Relational Model?



I was as confused as anyone else



Chris Date

Date on Database: Writings 2000-2006

# SQL:1999 — Escaping the Relational Cage

## Relational Model?

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



Chris Date

Date on Database: Writings 2000-2006

# SQL:1999 — Escaping the Relational Cage

## Relational Model?

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

Domains Can Contain *Anything!*



Chris Date

Date on Database: Writings 2000-2006

# SQL:1999 — Escaping the Relational Cage

## Relational Model?

- Introduced rich types

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



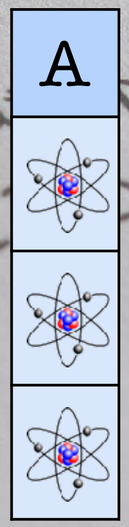
Chris Date

Date on Database: Writings 2000-2006

# SQL:1999 — Escaping the Relational Cage

Relational Model?

- Introduced rich types




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

Domains Can Contain *Anything!*



Chris Date

Date on Database: Writings 2000-2006

# 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!



Chris Date

Date on Database: Writings 2000-2006

# 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!



Chris Date

Date on Database: Writings 2000-2006

# 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!



Chris Date

Date on Database: Writings 2000-2006

# 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!*

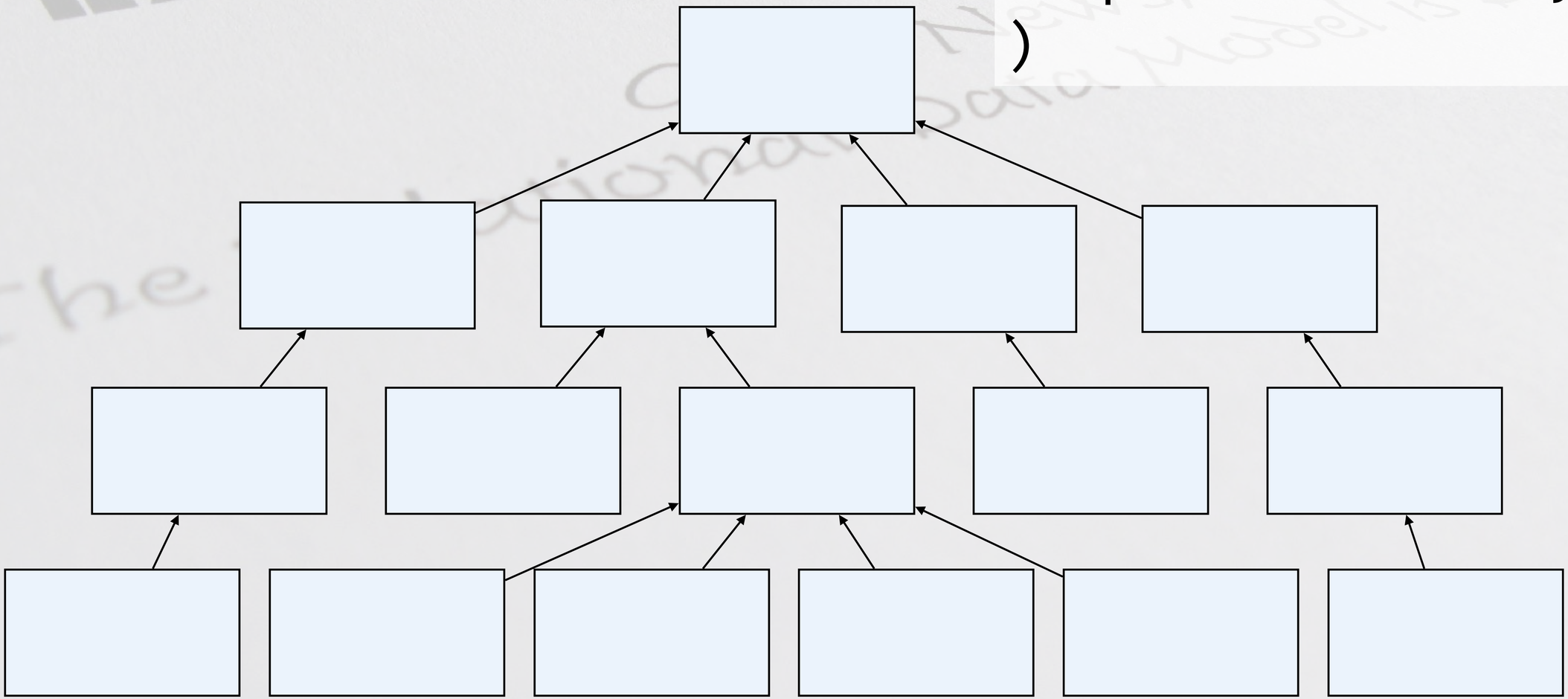


Chris Date

Date on Database: Writings 2000-2006

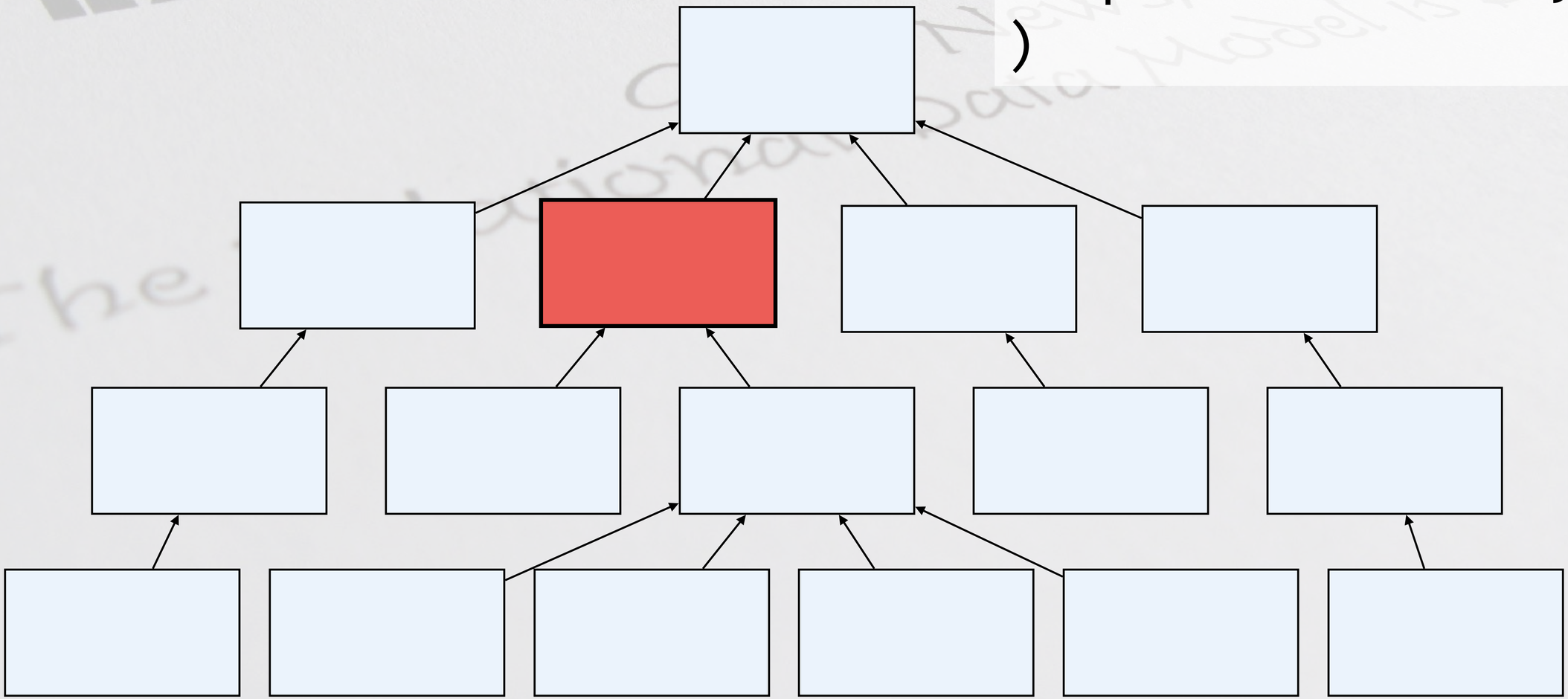
# SQL:1999 — Recursion

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



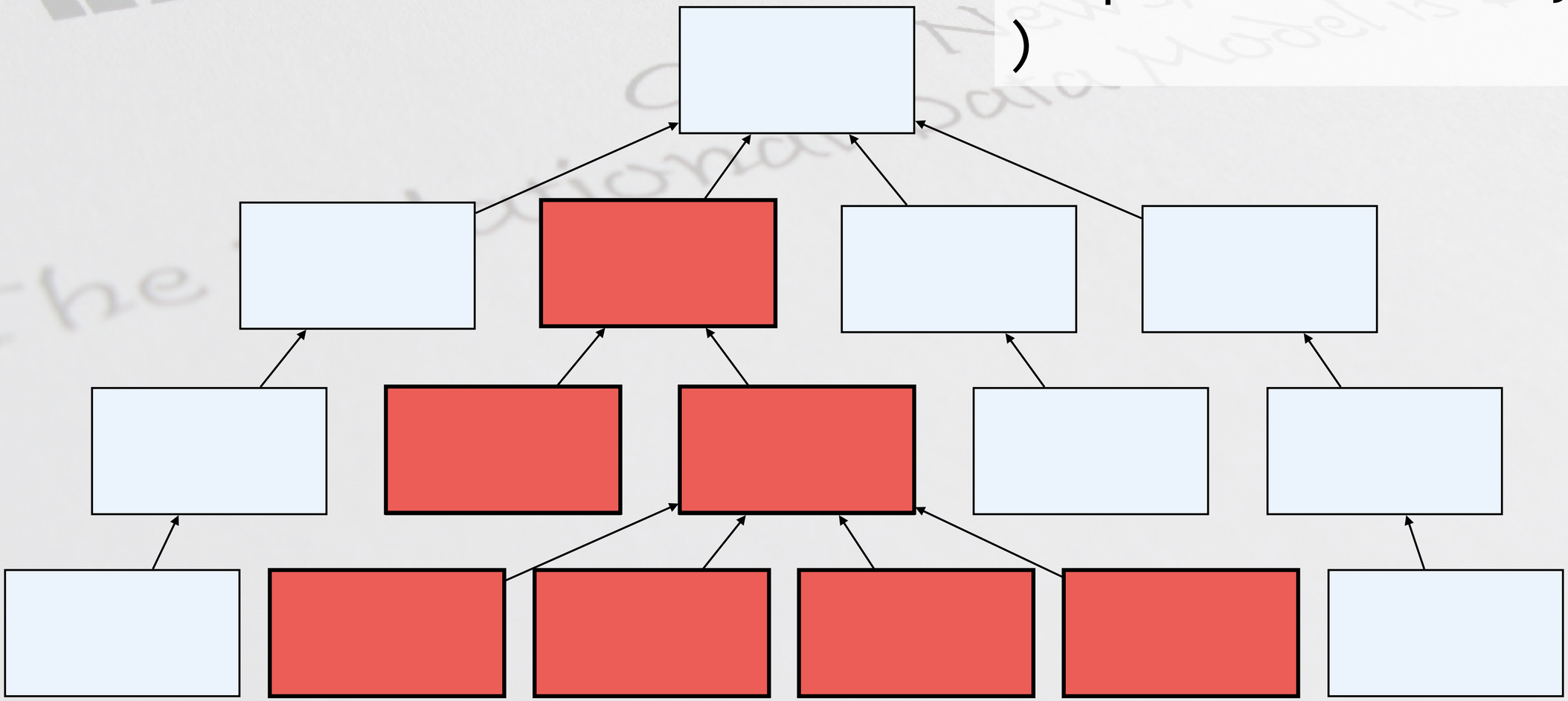
# SQL:1999 — Recursion

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

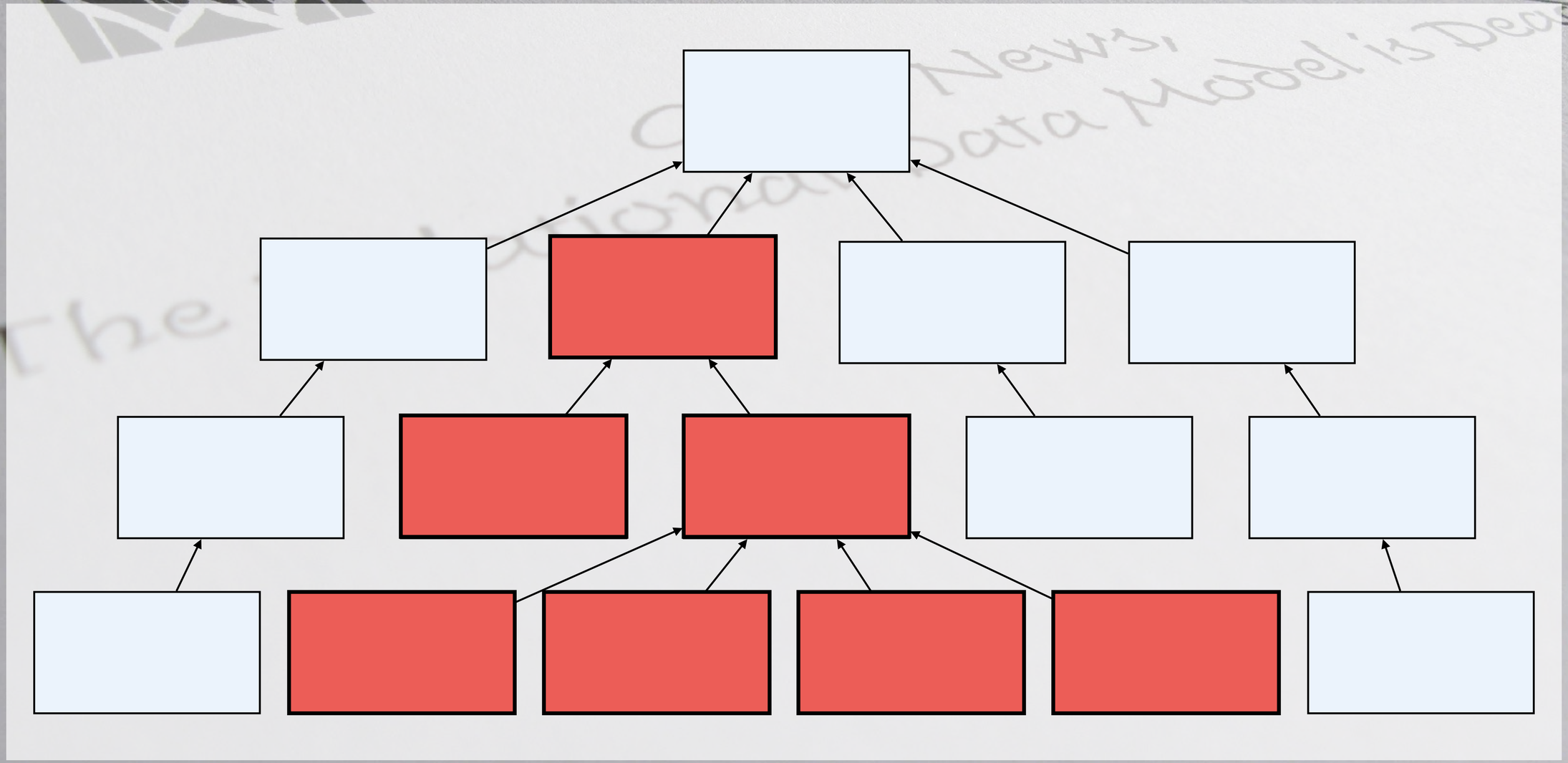


# SQL:1999 — Recursion

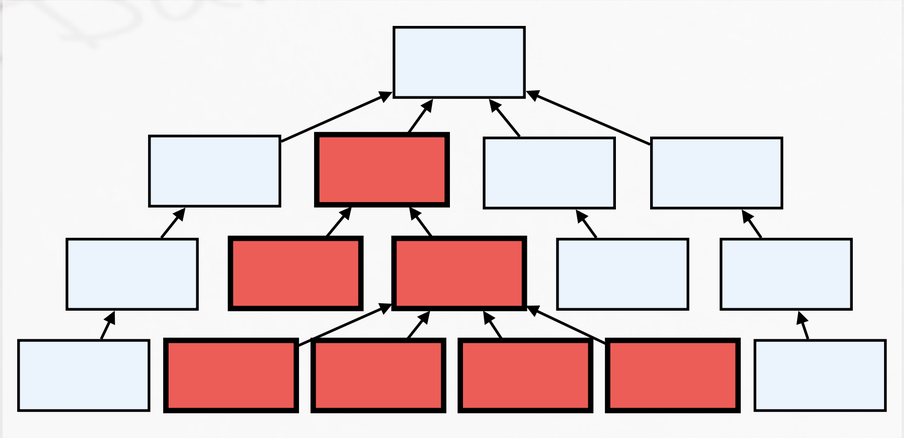
```
CREATE TABLE t (
    id      INTEGER,
    parent INTEGER,
)
```



# SQL:1999 — Recursion

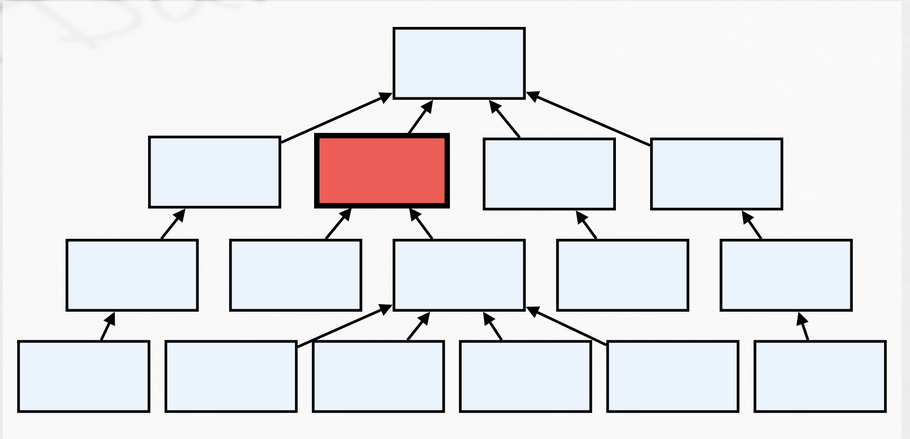


# SQL:1999 — Recursion



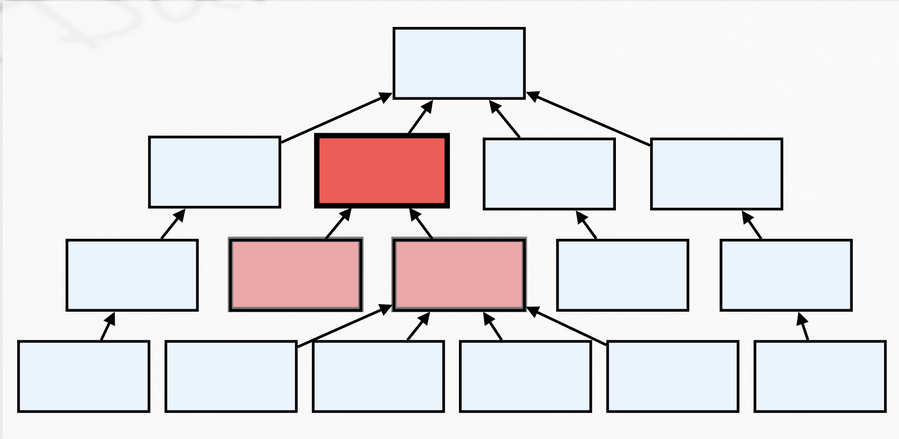
# SQL:1999 — Recursion

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



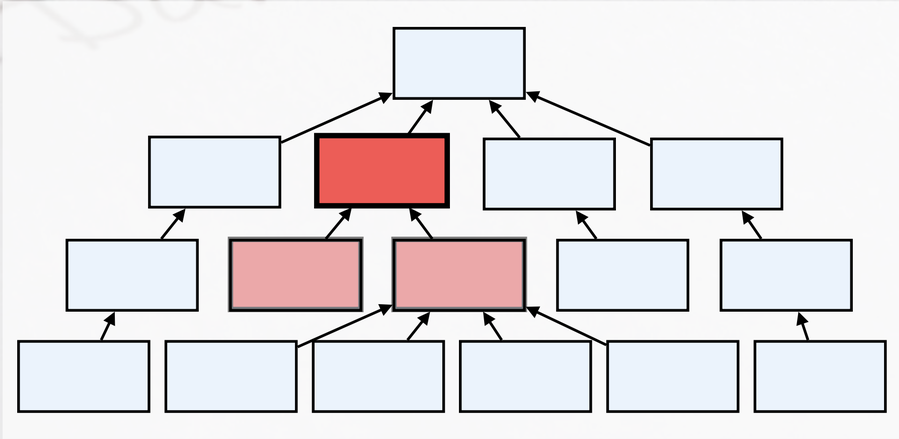
# SQL:1999 — Recursion

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



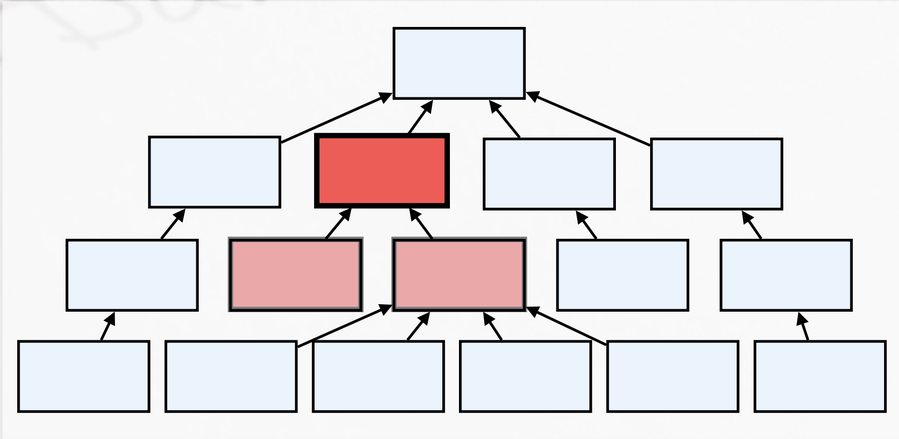
# SQL:1999 — Recursion

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



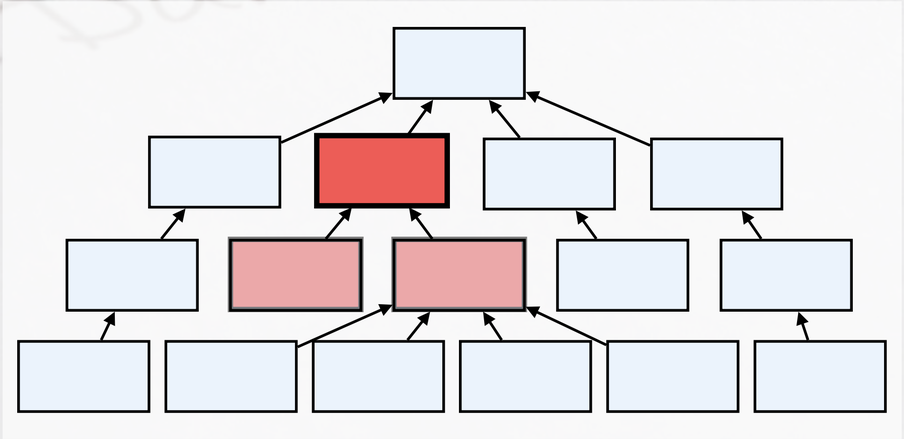
# SQL:1999 — Recursion

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



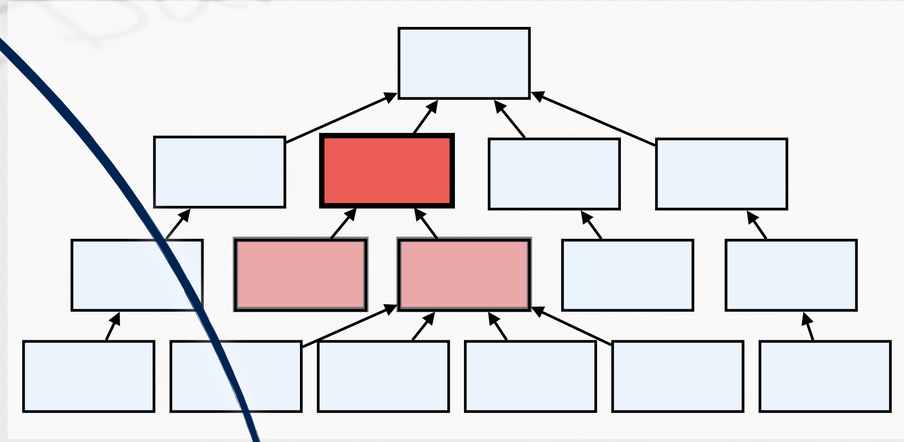
# SQL:1999 — Recursion

```
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
```



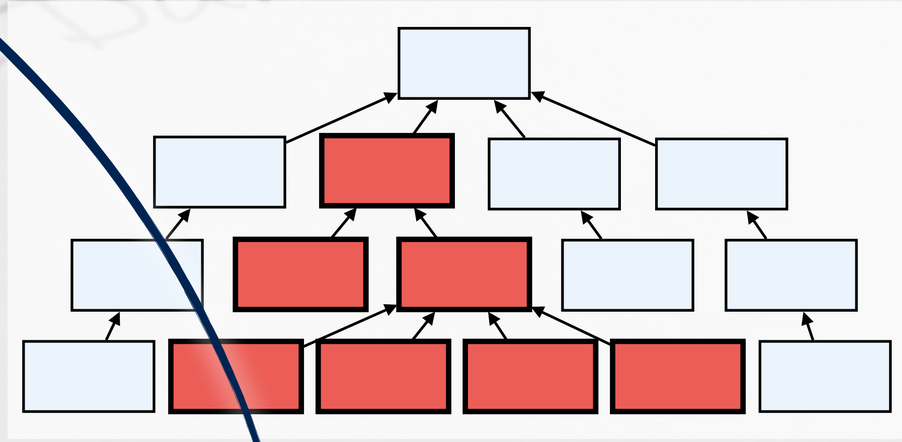
# SQL:1999 — Recursion

```
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
```

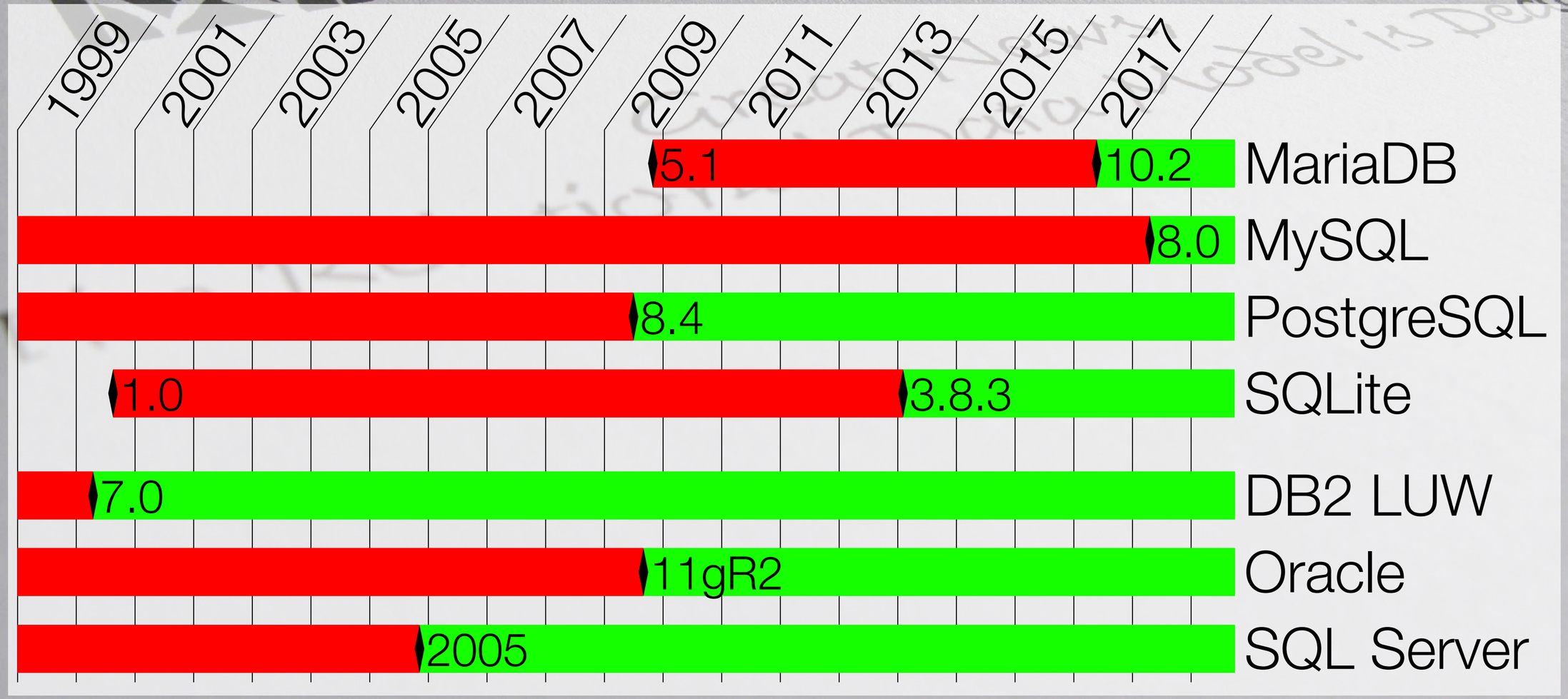


# SQL:1999 — Recursion

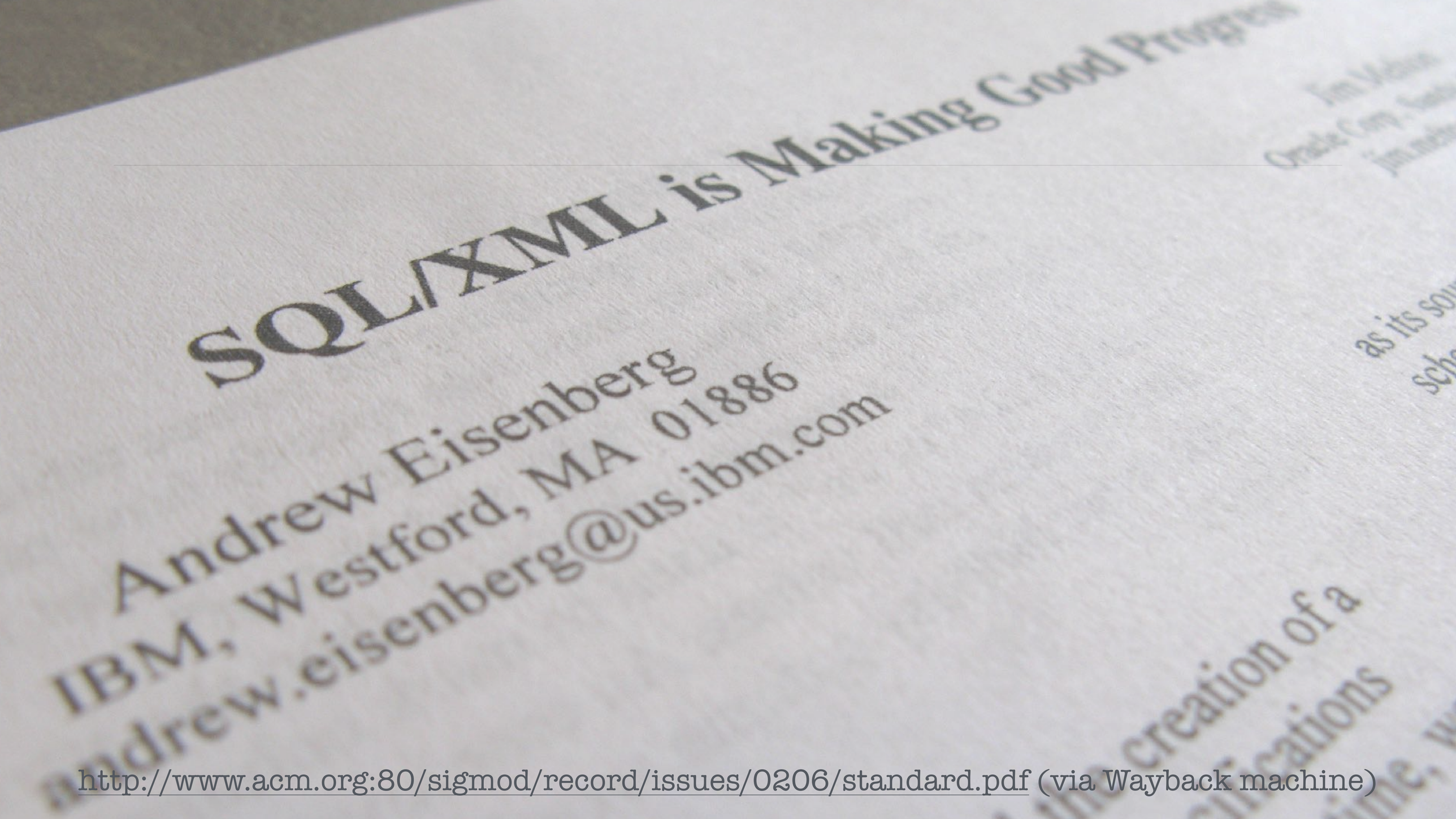
```
WITH RECURSIVE prev (id, parent) AS (
   SELECT t.id, t.parent
     FROM t
    WHERE t.id = ?
UNION ALL
   SELECT t.id, t.parent
     FROM t
     JOIN prev ON t.parent = prev.id
)
SELECT * FROM prev
```



# SQL:1999 — Recursion




Lubitemarsh
Information Systems Corporation
The Relational Great News,
Data Model is Dead!
1999
2003




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

# SQL:2003 — Schemaless & Analytical

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

Analytical

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

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

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 |

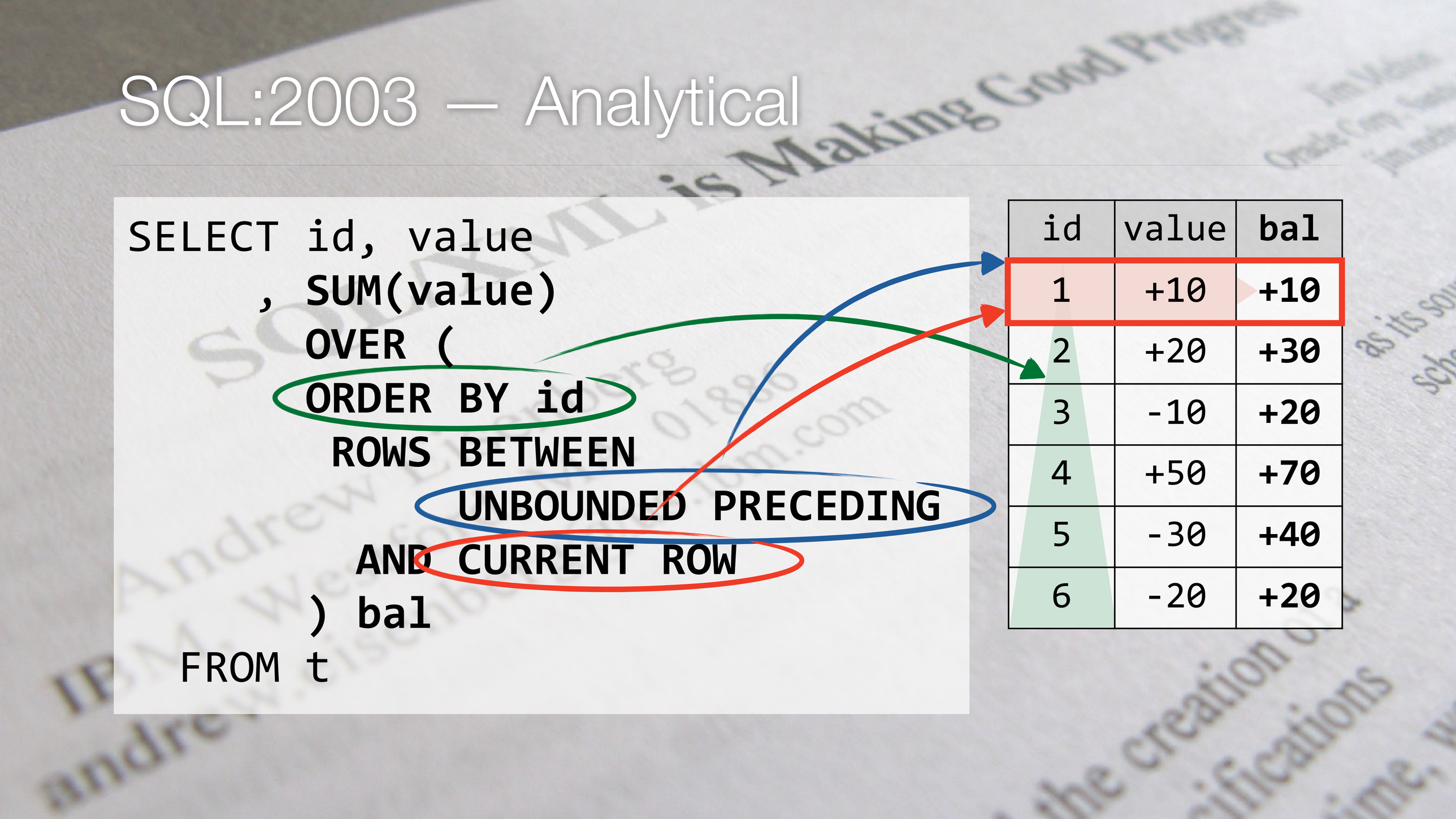
# SQL:2003 — Analytical

```
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

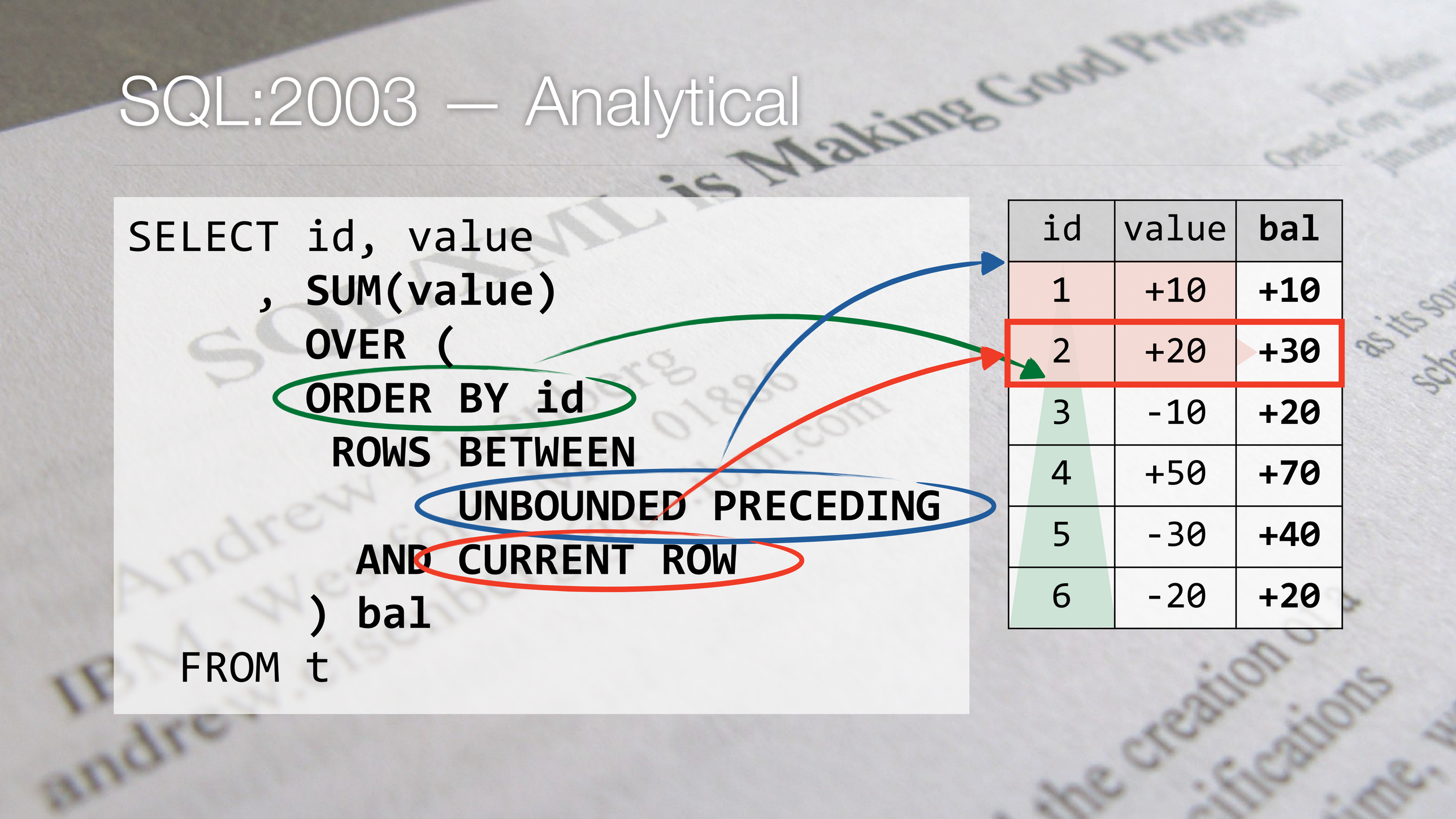
```
SELECT id, value
     , SUM(value)
       OVER (
         ORDER BY id
           ROWS BETWEEN
             UNBOUNDED PRECEDING
           AND CURRENT ROW
       ) bal
  FROM t
```



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

# SQL:2003 — Analytical

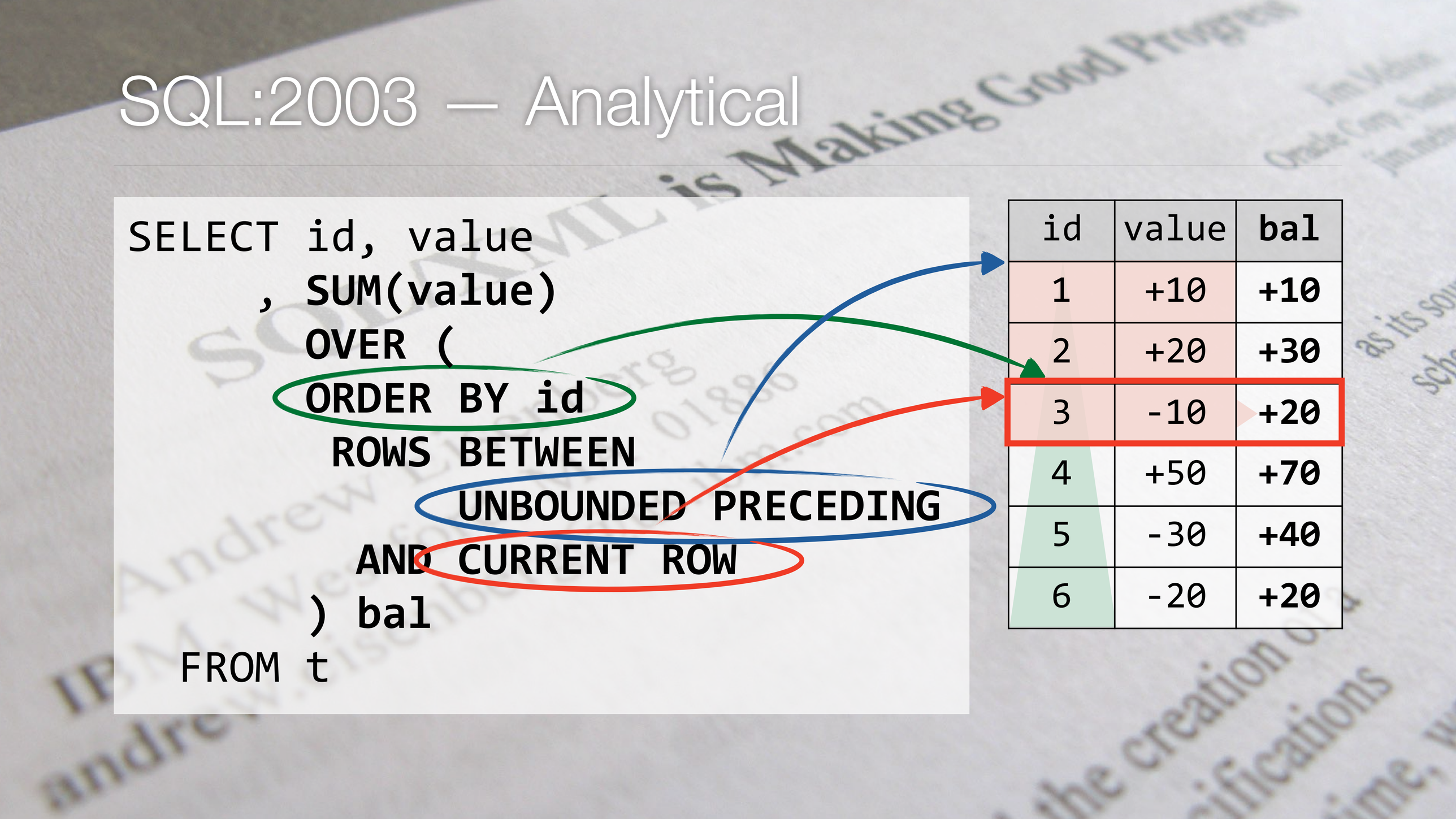
```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
         ROWS BETWEEN
           UNBOUNDED PRECEDING
         AND CURRENT ROW
       ) bal
  FROM t
```



| 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 |

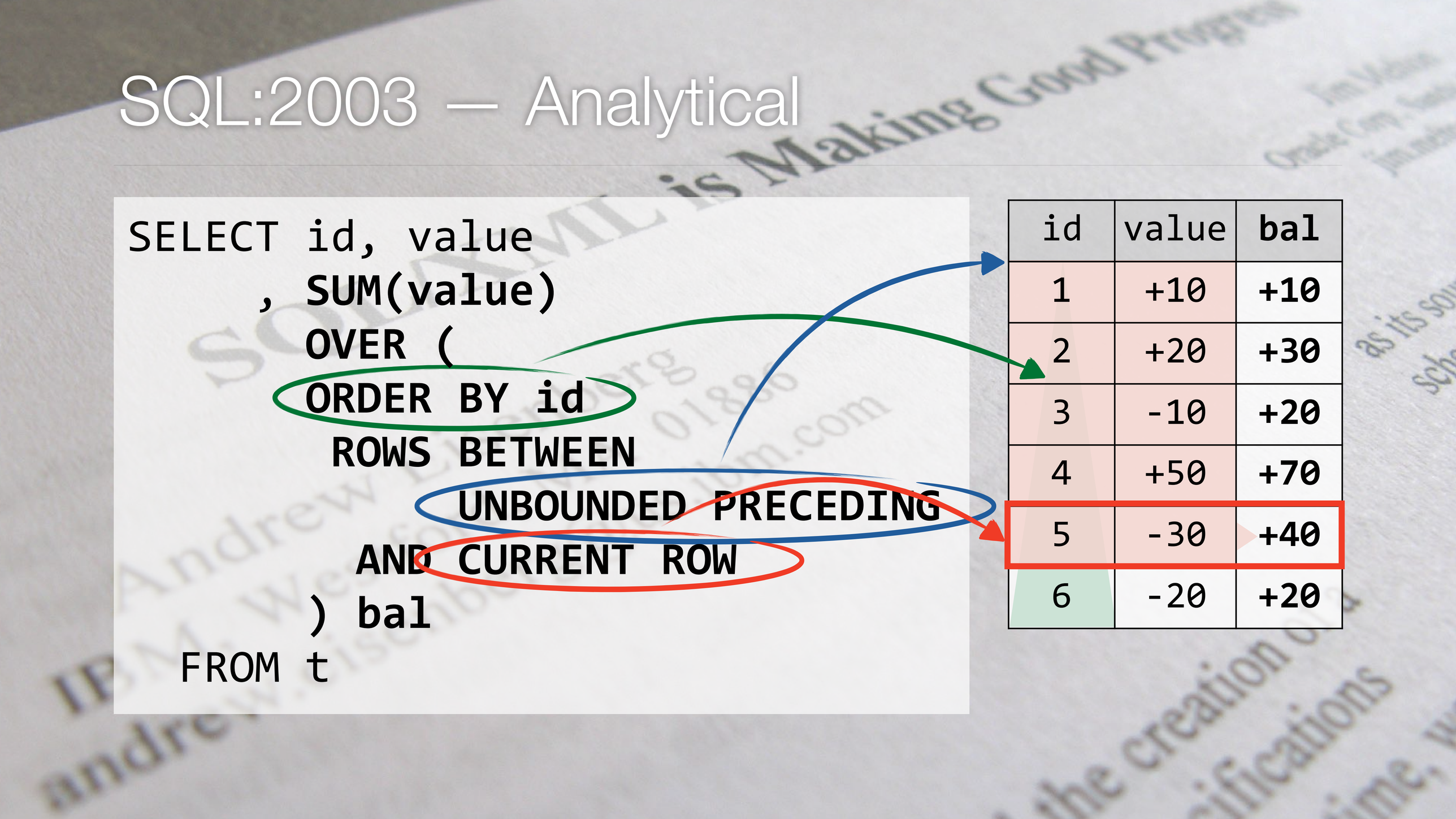
# SQL:2003 — Analytical

```
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
```



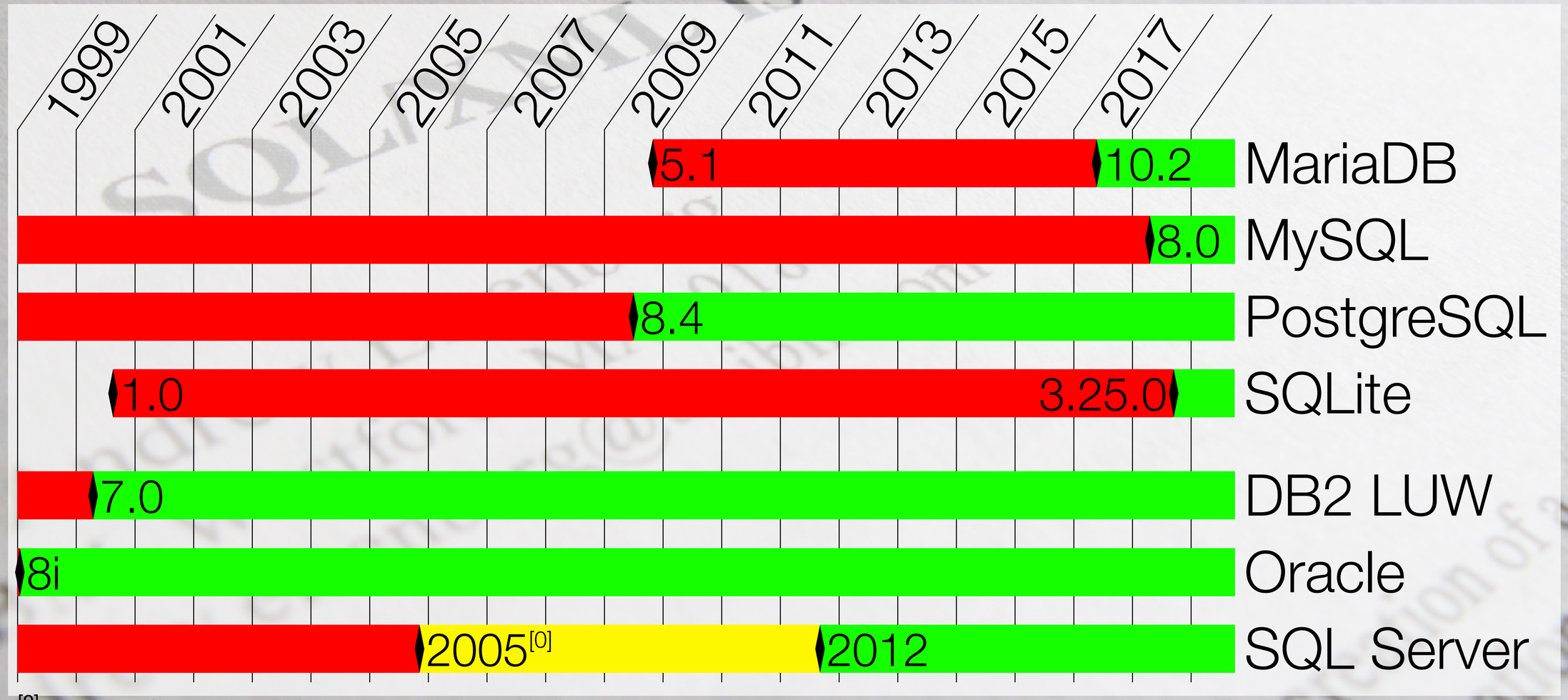
| id | value | bal |
|---|---|---|
| 1 | +10 | +10 |
| 2 | +20 | +30 |
| 3 | -10 | +20 |
| 4 | +50 | +70 |
| 5 | -30 | +40 |
| 6 | -20 | +20 |

# SQL:2003 — Analytical

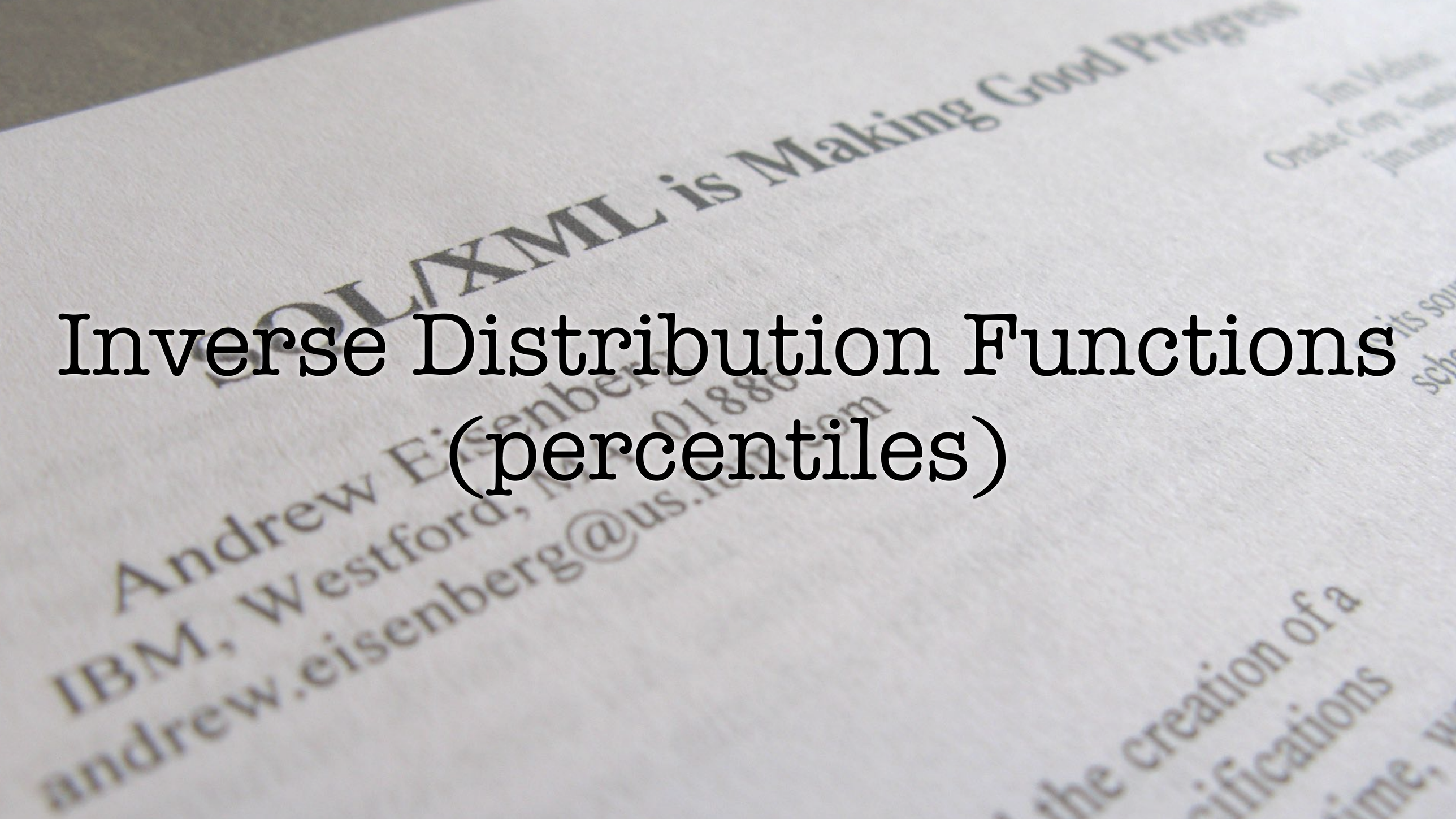
```
SELECT id, value
     , SUM(value)
       OVER (
       ORDER BY id
         ROWS BETWEEN
           UNBOUNDED PRECEDING
         AND CURRENT ROW
       ) bal
  FROM t
```

| 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]Without framing




# Inverse Distribution Functions (percentiles)

# SQL:2003 — Analytical (Median)



# SQL:2003 — Analytical (Median)

```
SELECT d1.val
  FROM data d1
  JOIN data d2
    ON (d1.val < d2.val
        OR (d1.val=d2.val AND d1.id<d2.id))
 GROUP BY d1.val
HAVING count(*) =
       (SELECT FLOOR(COUNT(*)/2)
          FROM data d3)
```

# SQL:2003 — Analytical (Median)

```
SELECT d1.val
  FROM data d1
  JOIN data d2
    ON (d1.val < d2.val
        OR (d1.val=d2.val AND d1.id<d2.id))
 GROUP BY d1.val
HAVING count(*) =
       (SELECT FLOOR(COUNT(*)/2)
          FROM data d3)
```

Number rows

# SQL:2003 — Analytical (Median)

```
SELECT d1.val
  FROM data d1
  JOIN data d2
    ON (d1.val < d2.val
        OR (d1.val=d2.val AND d1.id<d2.id))
 GROUP BY d1.val
HAVING count(*) =
       (SELECT FLOOR(COUNT(*)/2)
          FROM data d3)
```

Number rows

Pick middle one

# SQL:2003 — Analytical (Median)

```
SELECT d1.val
  FROM data d1
  JOIN data d2
    ON (d1.val < d2.val
        OR (d1.val=d2.val AND d1.id<d2.id))
 GROUP BY d1.val
HAVING count(*) =
       (SELECT FLOOR(COUNT(*)/2)
          FROM data d3)
```

Number rows

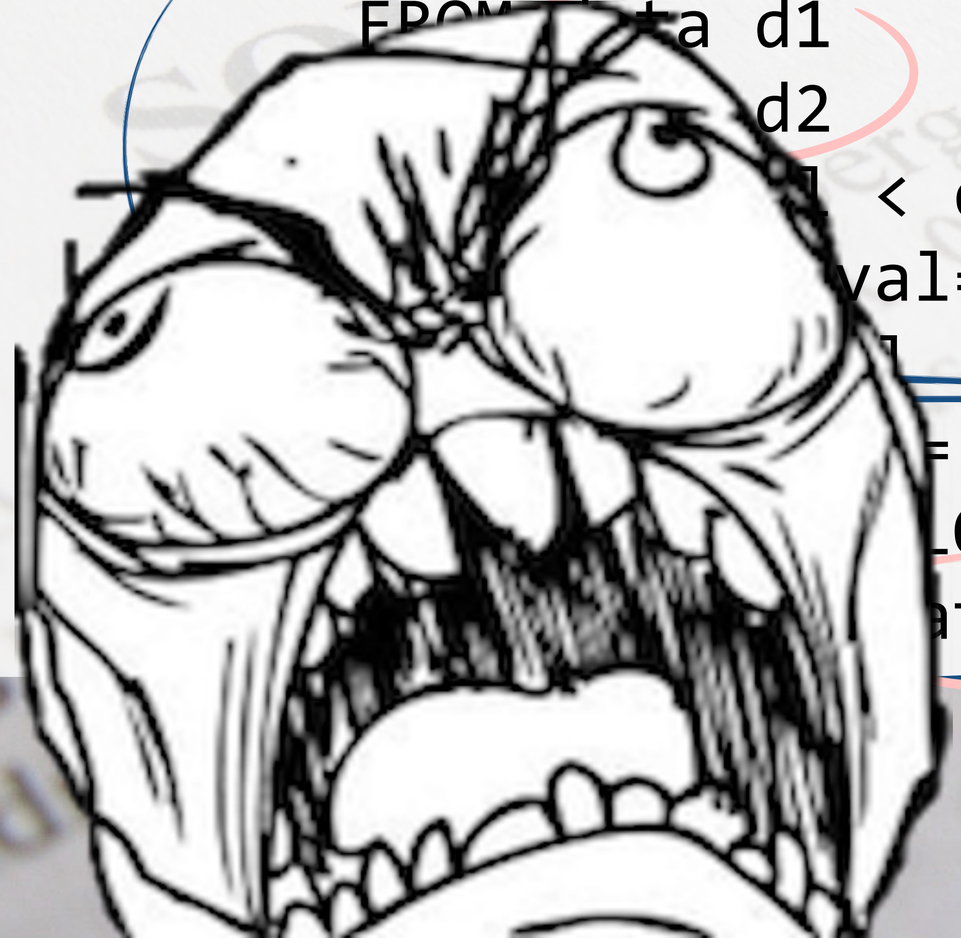
Pick middle one

# SQL:2003 — Analytical (Median)

```
SELECT d1.val
  FROM data d1
       d2
       1 < d2.val
          val=d2.val AND d1.id<d2.id))
       1
  F
  LOOR(COUNT(*)/2)
  ata d3)
```

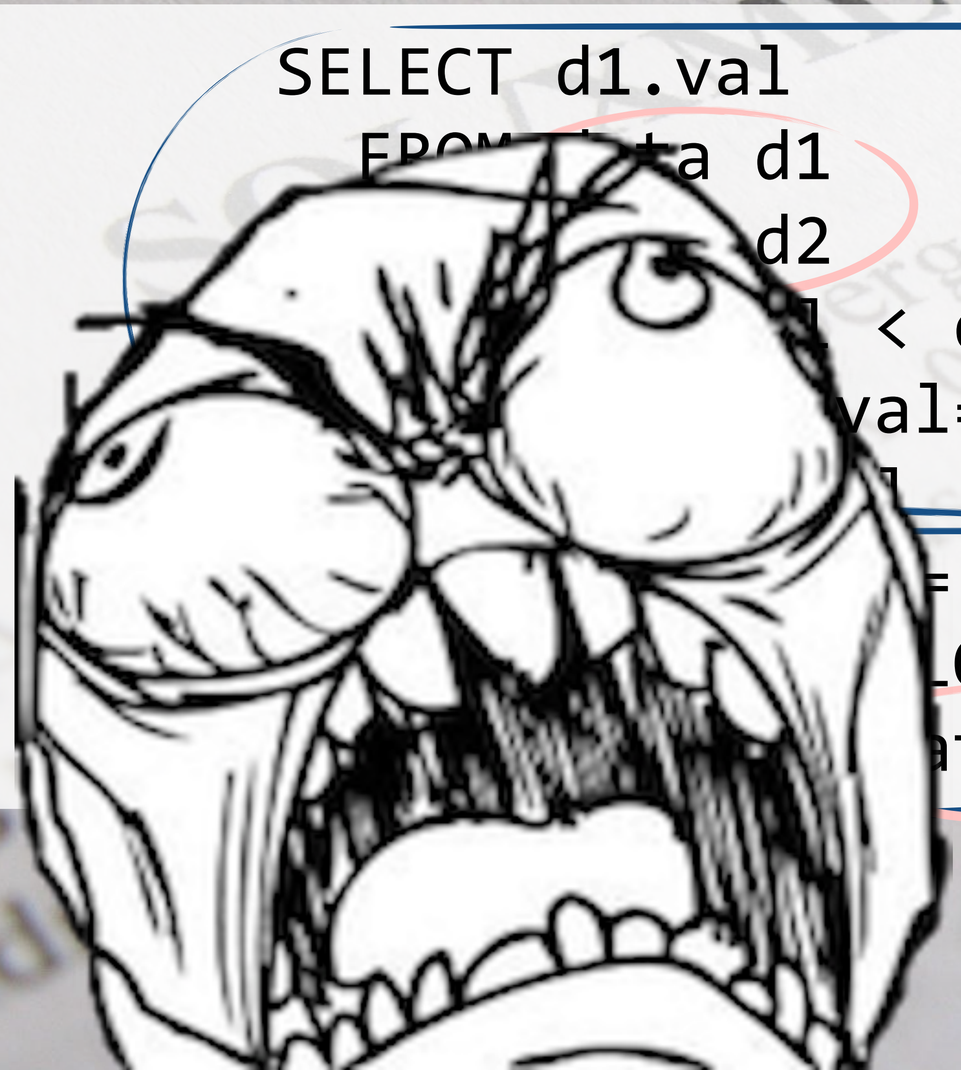
Number rows

Pick middle one



# SQL:2003 — Analytical (M

```
SELECT d1.val
   FROM ata d1
                d2
                 < d2.val
                val=d2.val AN
                 1
                 F
                LOOR(COUNT(
                ata d3)
```

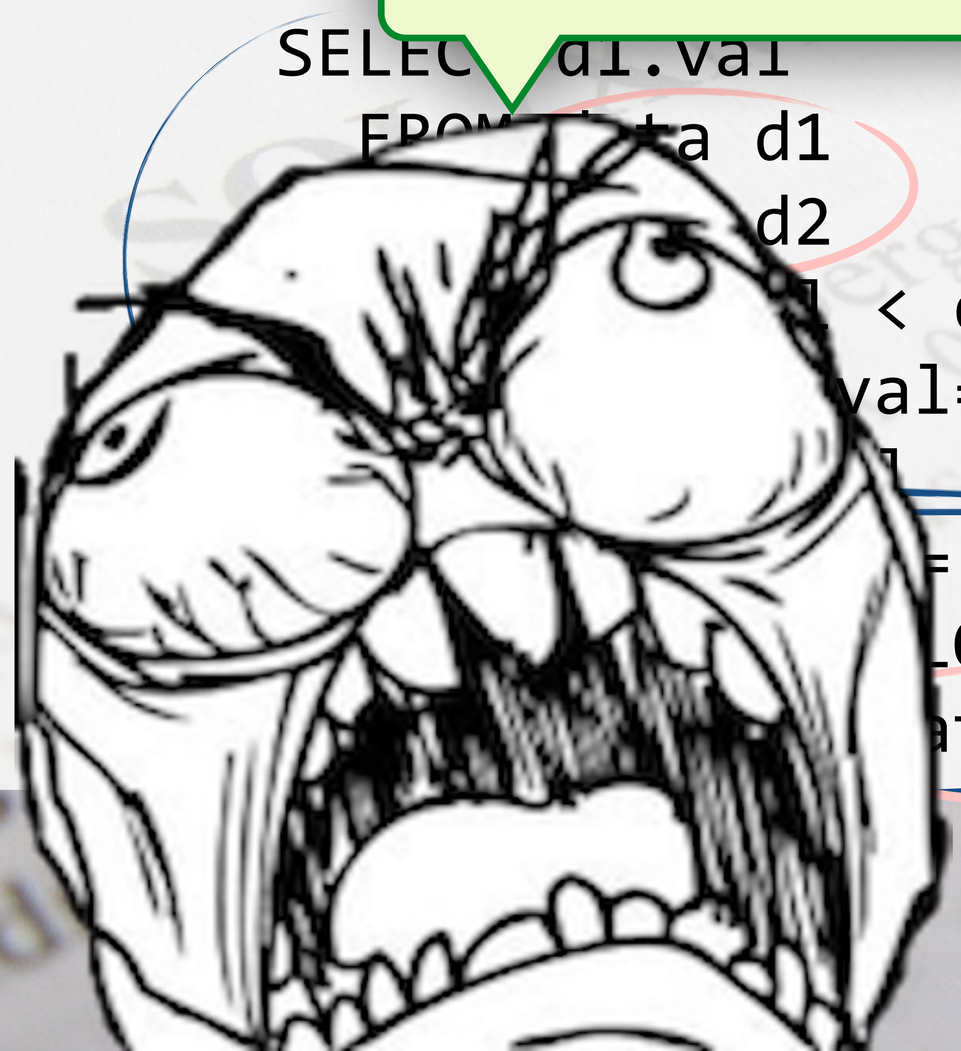




SQL:20 M

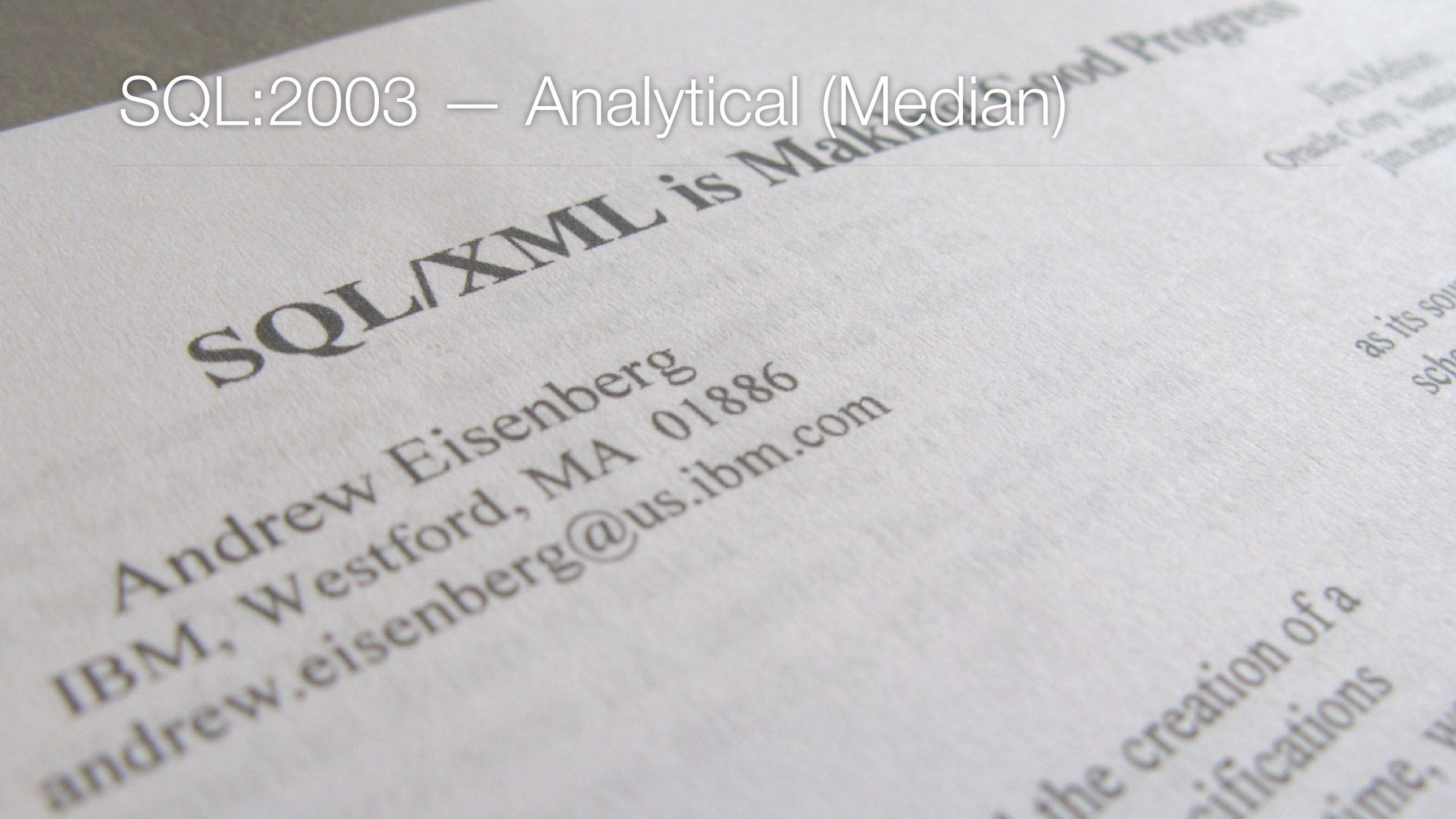
Grab stickers and coasters!

```
SELEC  d1.val
   FROM ta d1
        d2
        1 < d2.val
        val=d2.val AN
        1
F
LOOR(COUNT(
ata d3)
```





# SQL:2003 — Analytical (Median)



# SQL:2003 — Analytical (Median)

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val)
  FROM data
```

# SQL:2003 — Analytical (Median)

Median

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val)
  FROM data
```

# SQL:2003 — Analytical (Median)

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val)
  FROM data
```

Median

Which value?

# SQL:2003 — Analytical (Median)

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val)
  FROM data
```

# SQL:2003 — Analytical (Median)

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val)
  FROM data
```

Two variants:

- for discrete values (categories)
- for continuous values (linear interpolation)

# SQL:2003 — Analytical (Median)

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val)
   FROM data
```

Two variants:

- for discrete values (categories)
- for continuous values (linear interpolation)



# SQL:2003 — Analytical (Median)

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val)
   FROM data
```

Two variants:

- for discrete values (categories)
- for continuous values (linear interpolation)



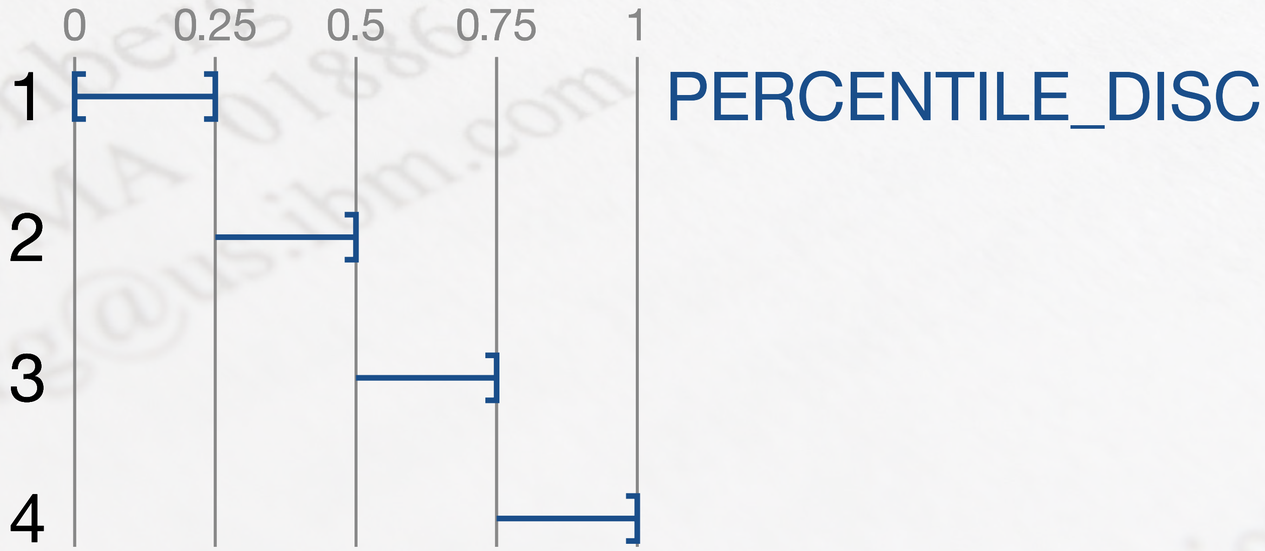

PERCENTILE_DISC

# SQL:2003 — Analytical (Median)

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val)
   FROM data
```

Two variants:

- for discrete values (categories)
- for continuous values (linear interpolation)

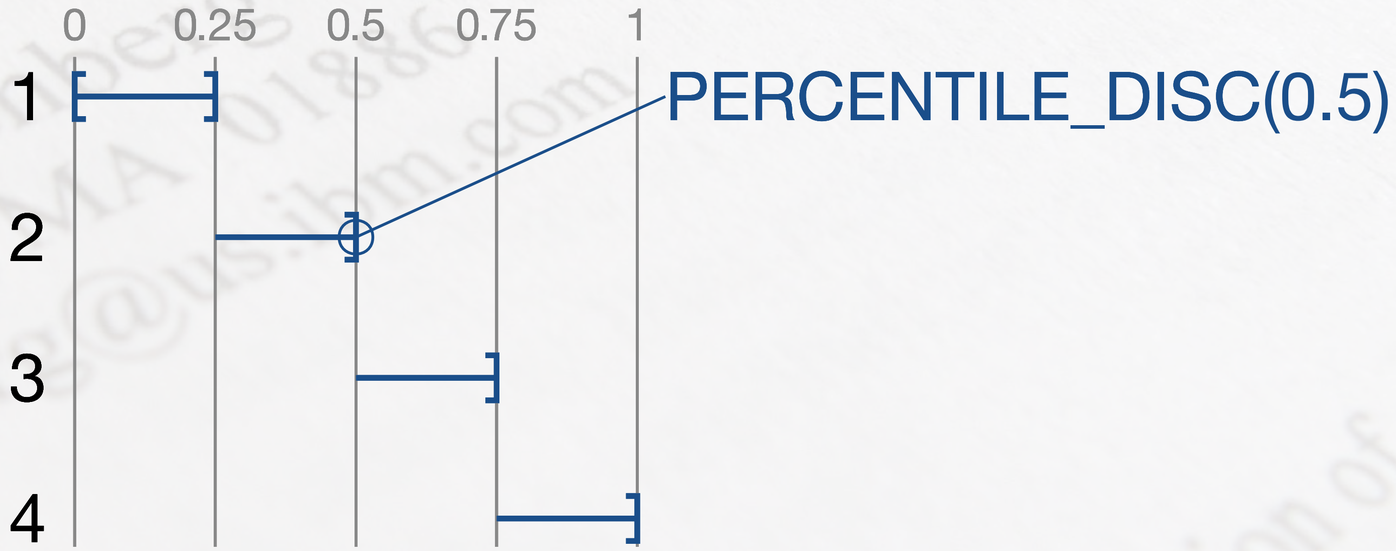


# SQL:2003 — Analytical (Median)

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val)
   FROM data
```

Two variants:

- for discrete values (categories)
- for continuous values (linear interpolation)

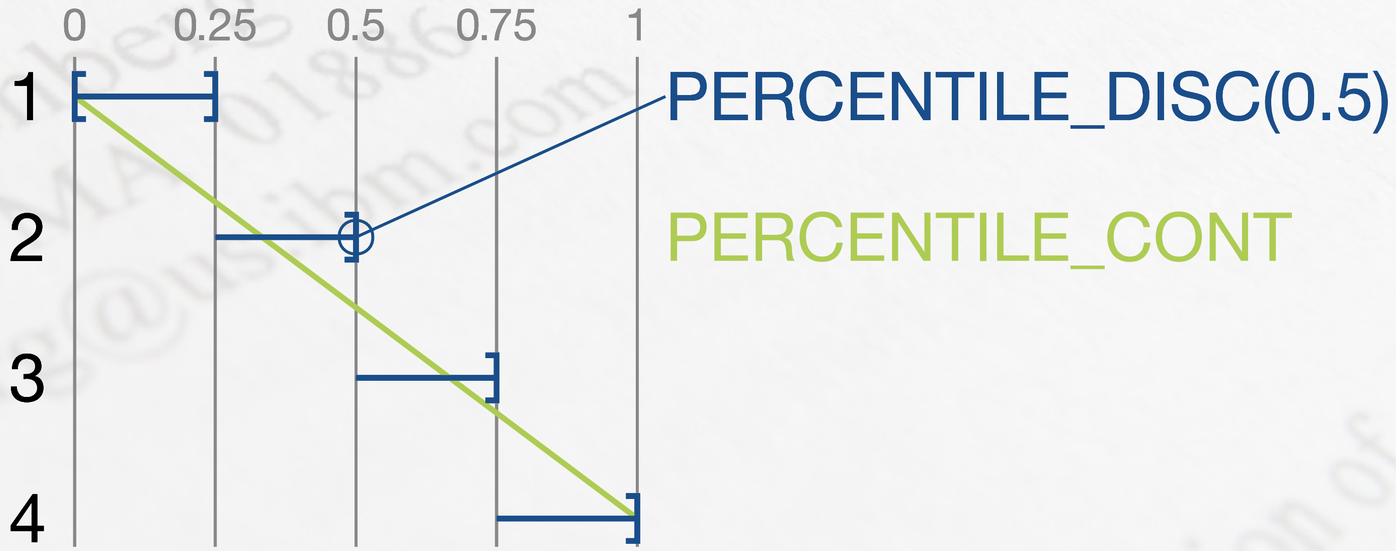


# SQL:2003 — Analytical (Median)

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val)
   FROM data
```

Two variants:

- for discrete values (categories)
- for continuous values (linear interpolation)

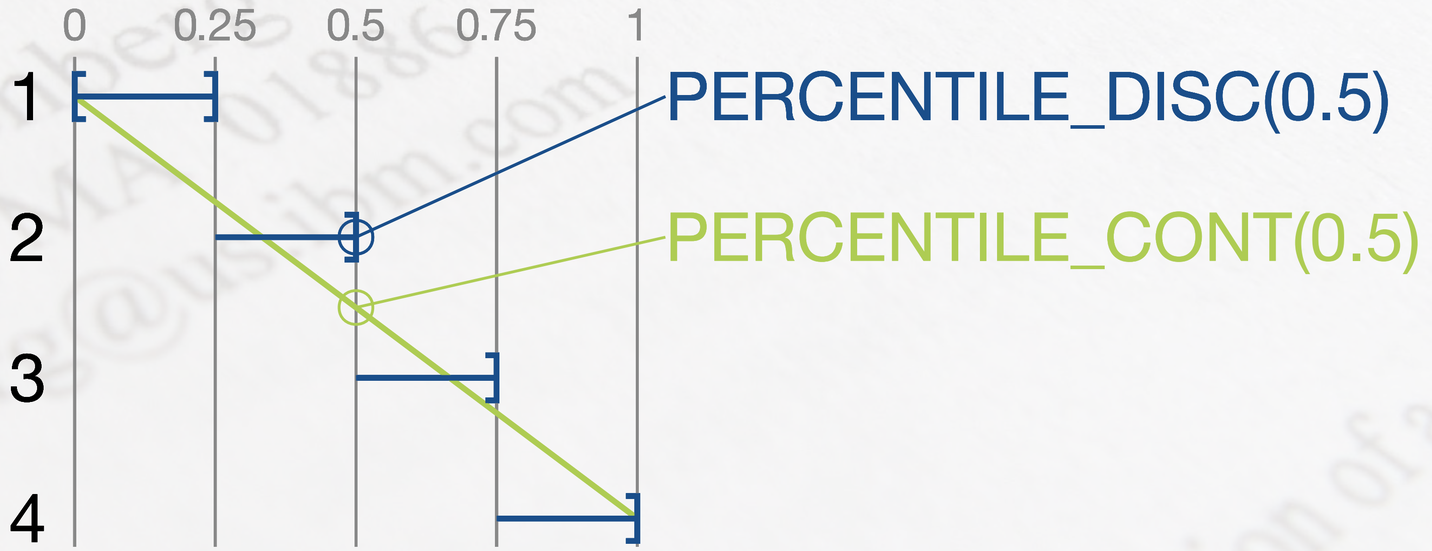


# SQL:2003 — Analytical (Median)

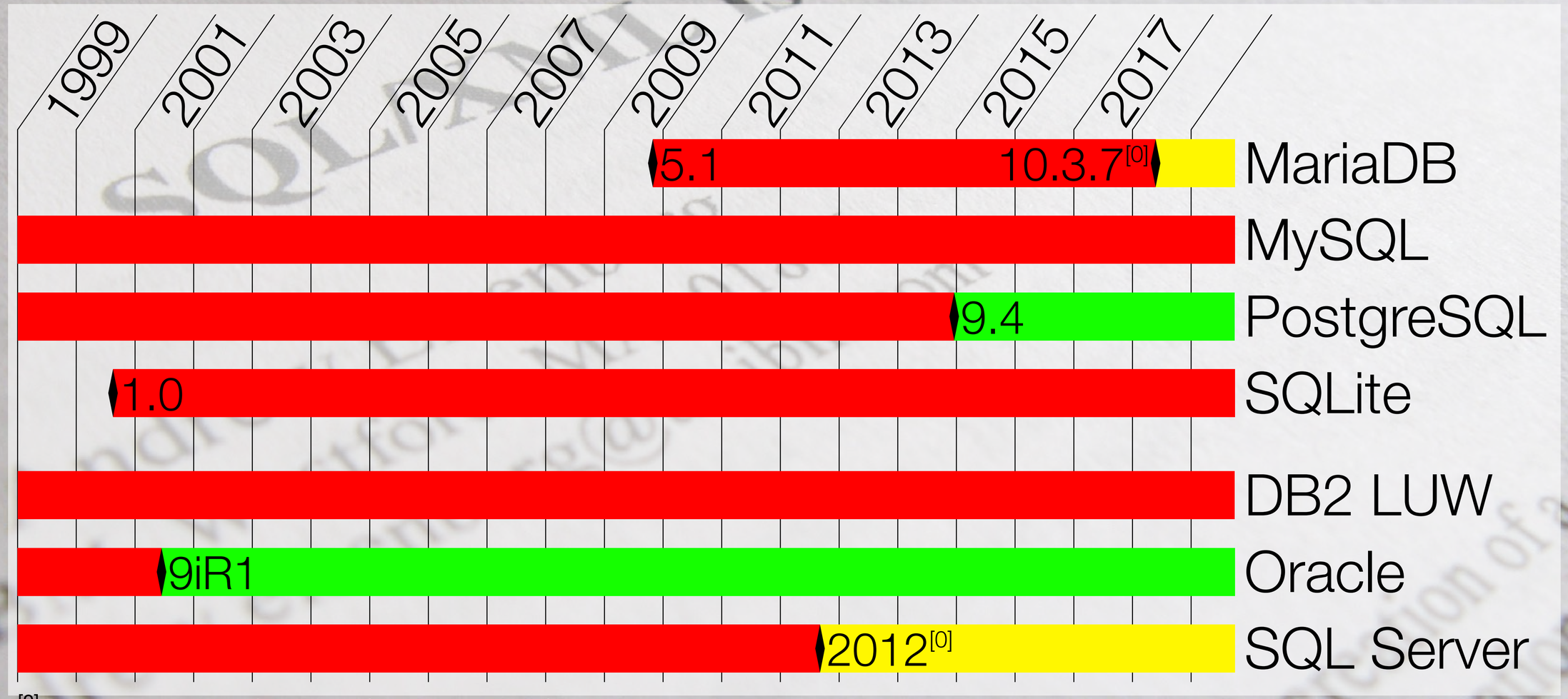
```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val)
   FROM data
```

Two variants:

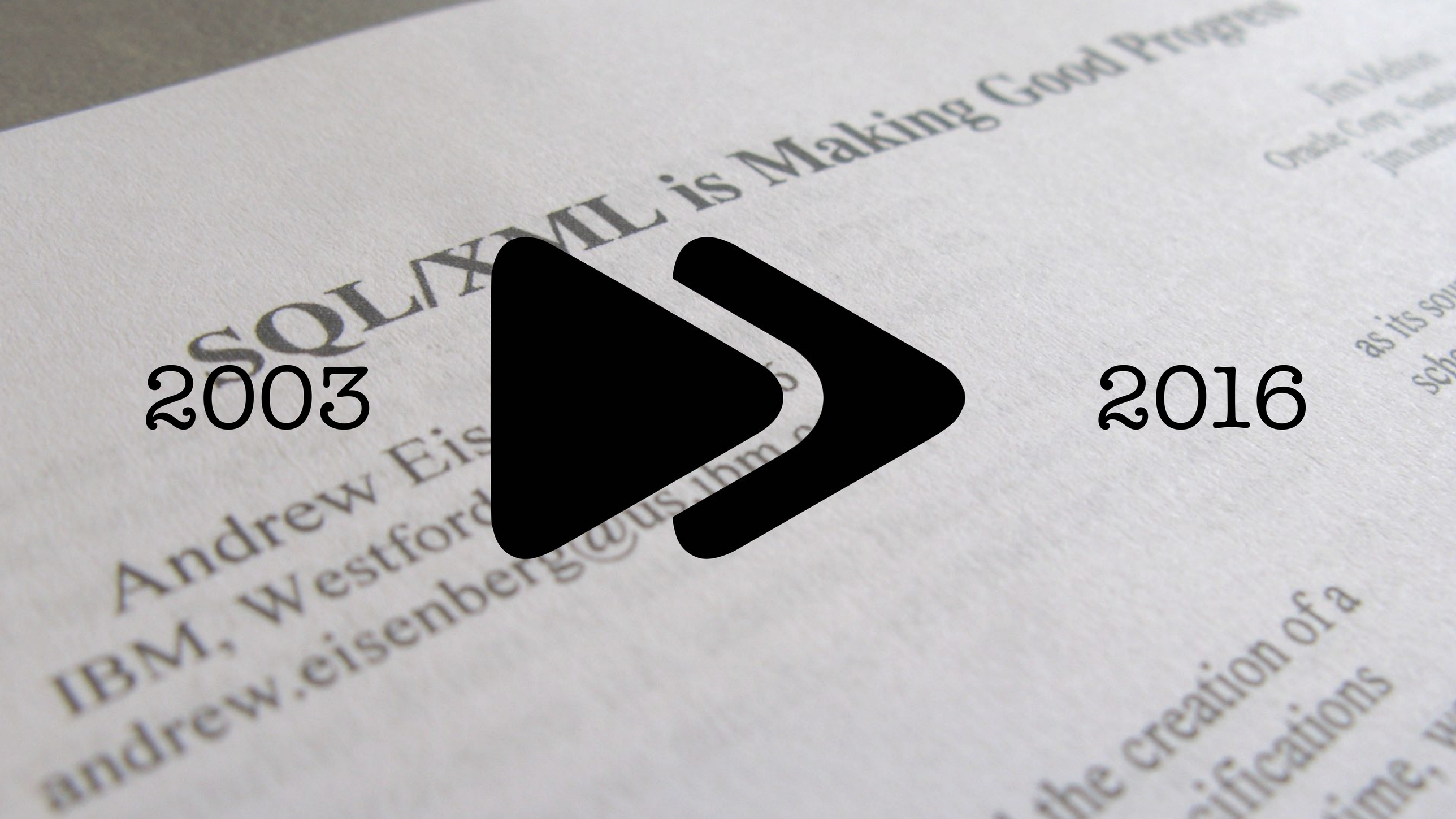
- for discrete values (categories)
- for continuous values (linear interpolation)



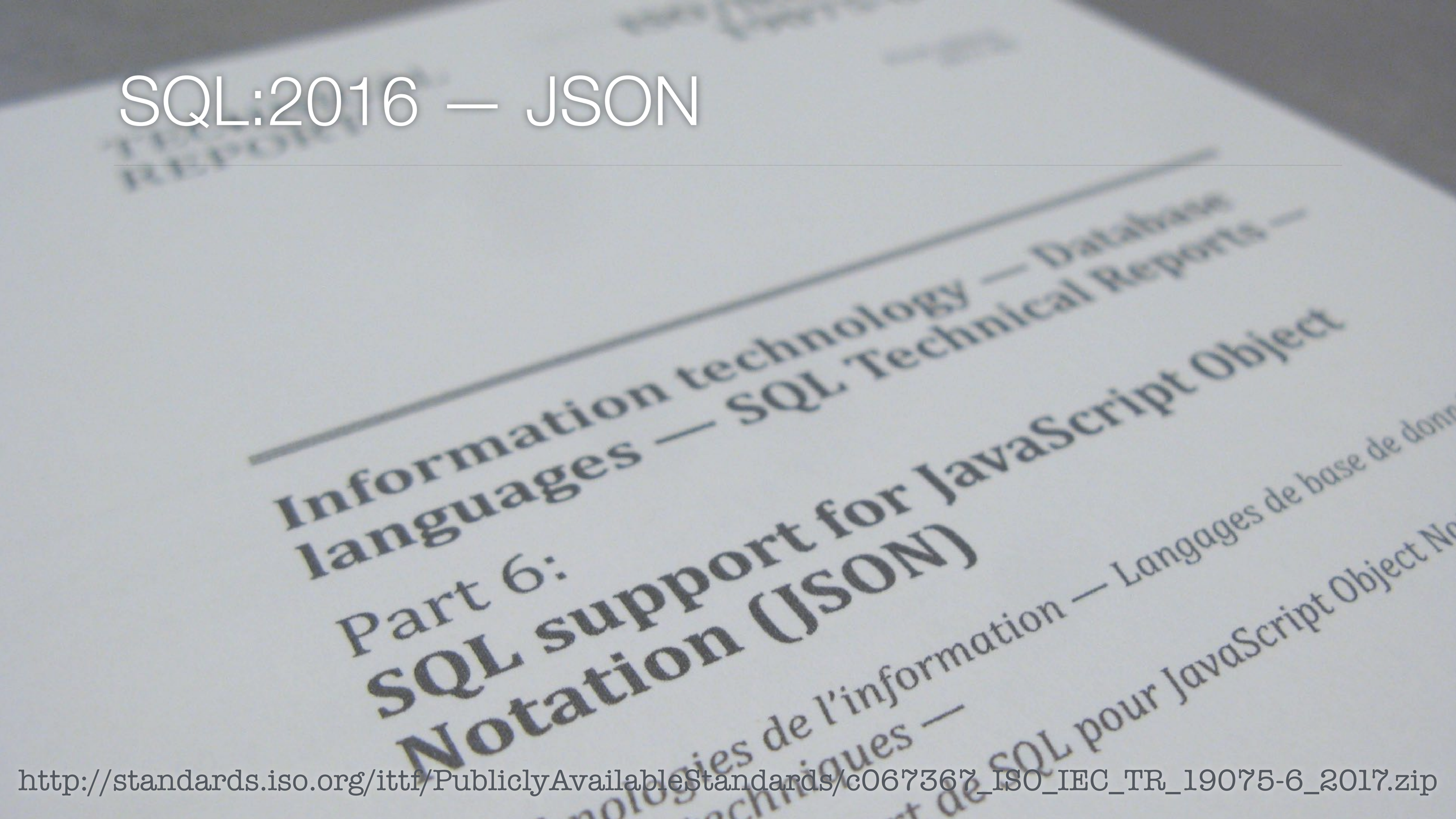
# SQL:2003 — PERCENTILE_DISC




[0]Only as window function (`OVER` required).


SQL/XML is Making Good Progress
Andrew Eisenberg
IBM, Westford
andrew.eisenberg@us.ibm.com
2003
2016

# SQL:2016 — JSON




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

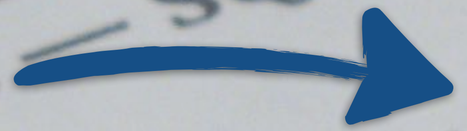
# SQL:2016 — JSON

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

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 |

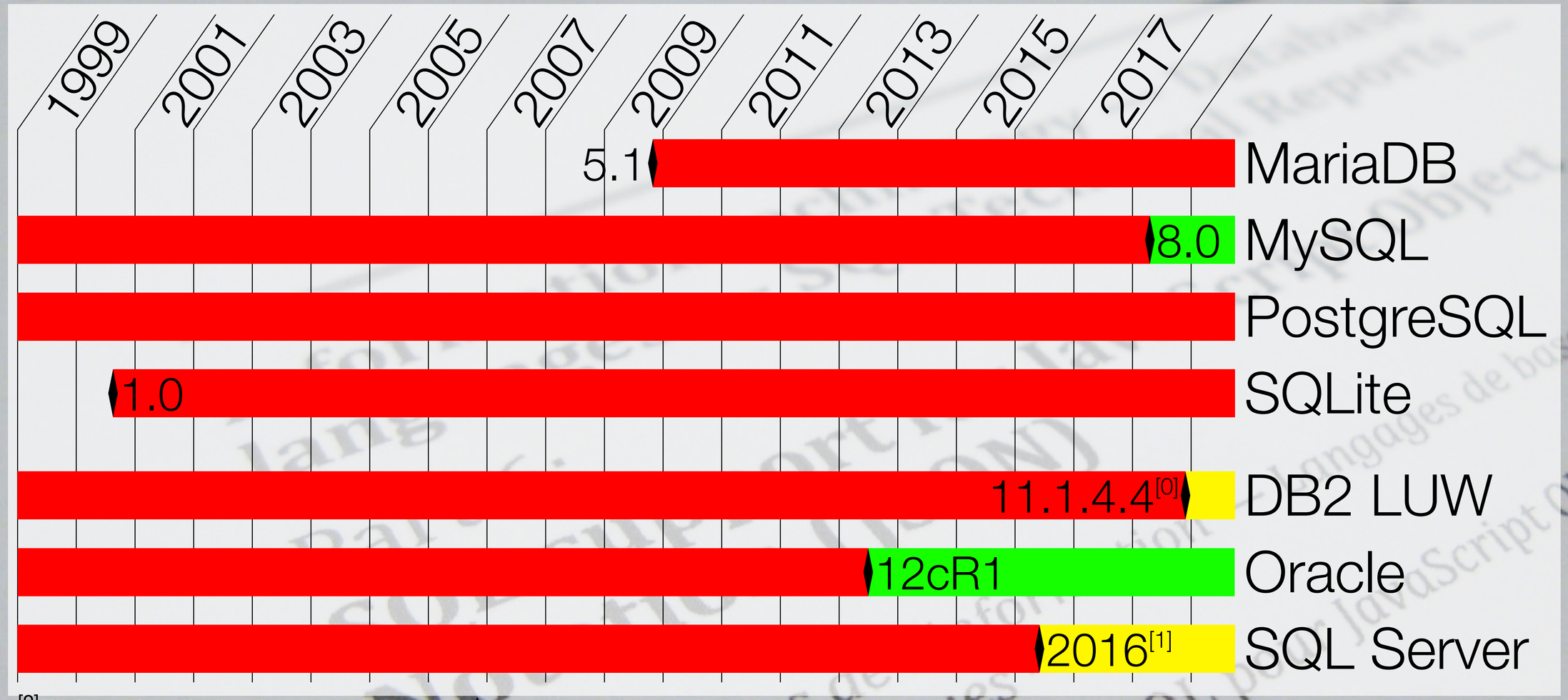
# SQL:2016 — JSON_TABLE — Use Case

```
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"
  }
]
```

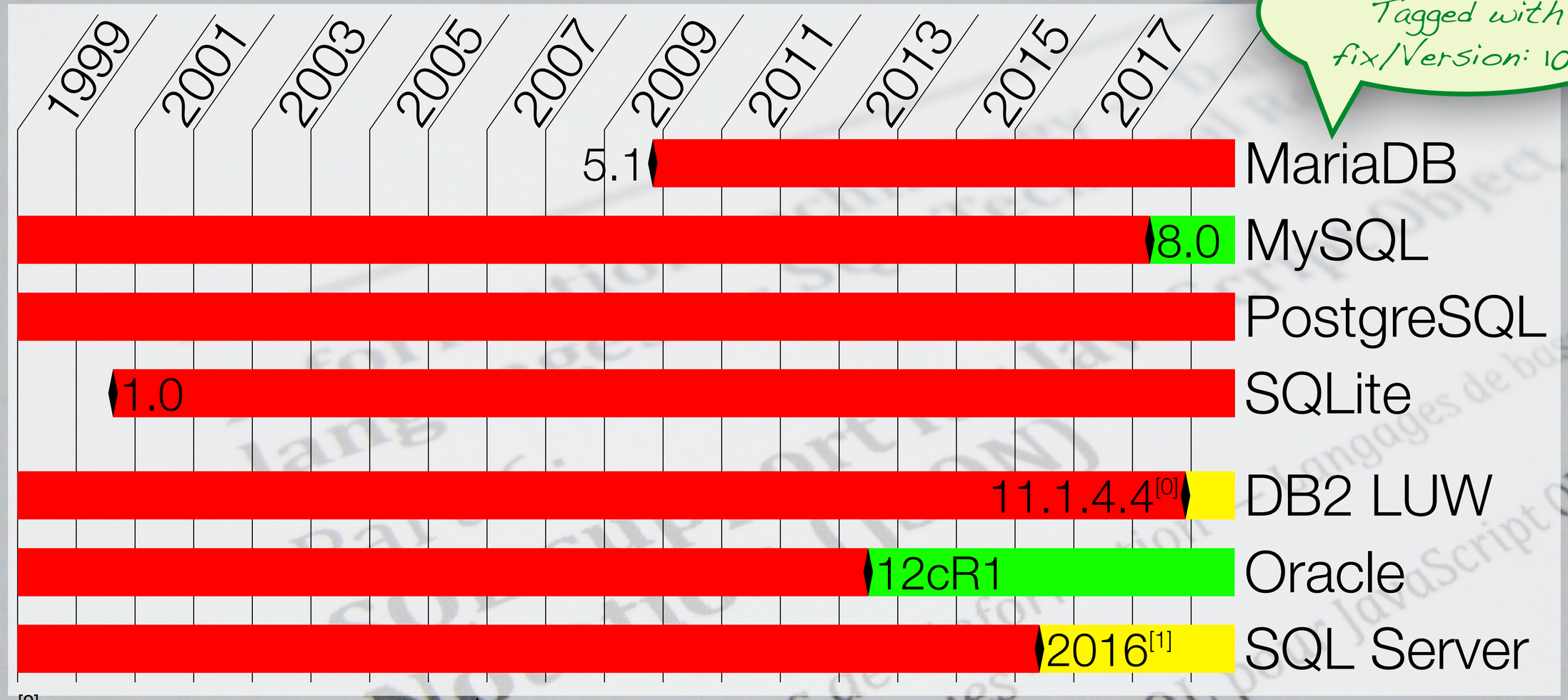
| id | a1 |
|---|---|
| 42 | foo |
| 43 | bar |

# SQL:2016 — JSON_TABLE



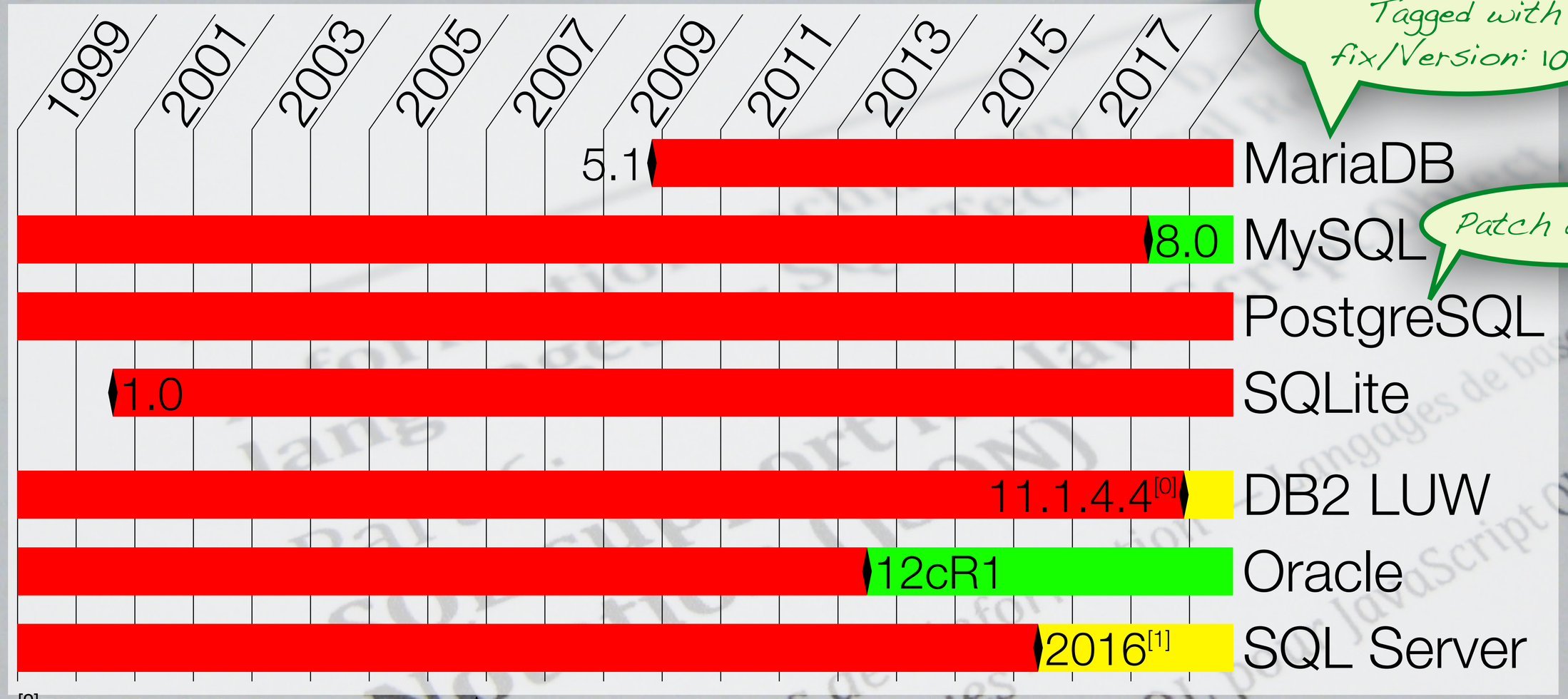

[0]Ridicoulus limitations: only 'strict $' as row expression
[1]OPENJSON provides similar functionality

# SQL:2016 — JSON_TABLE

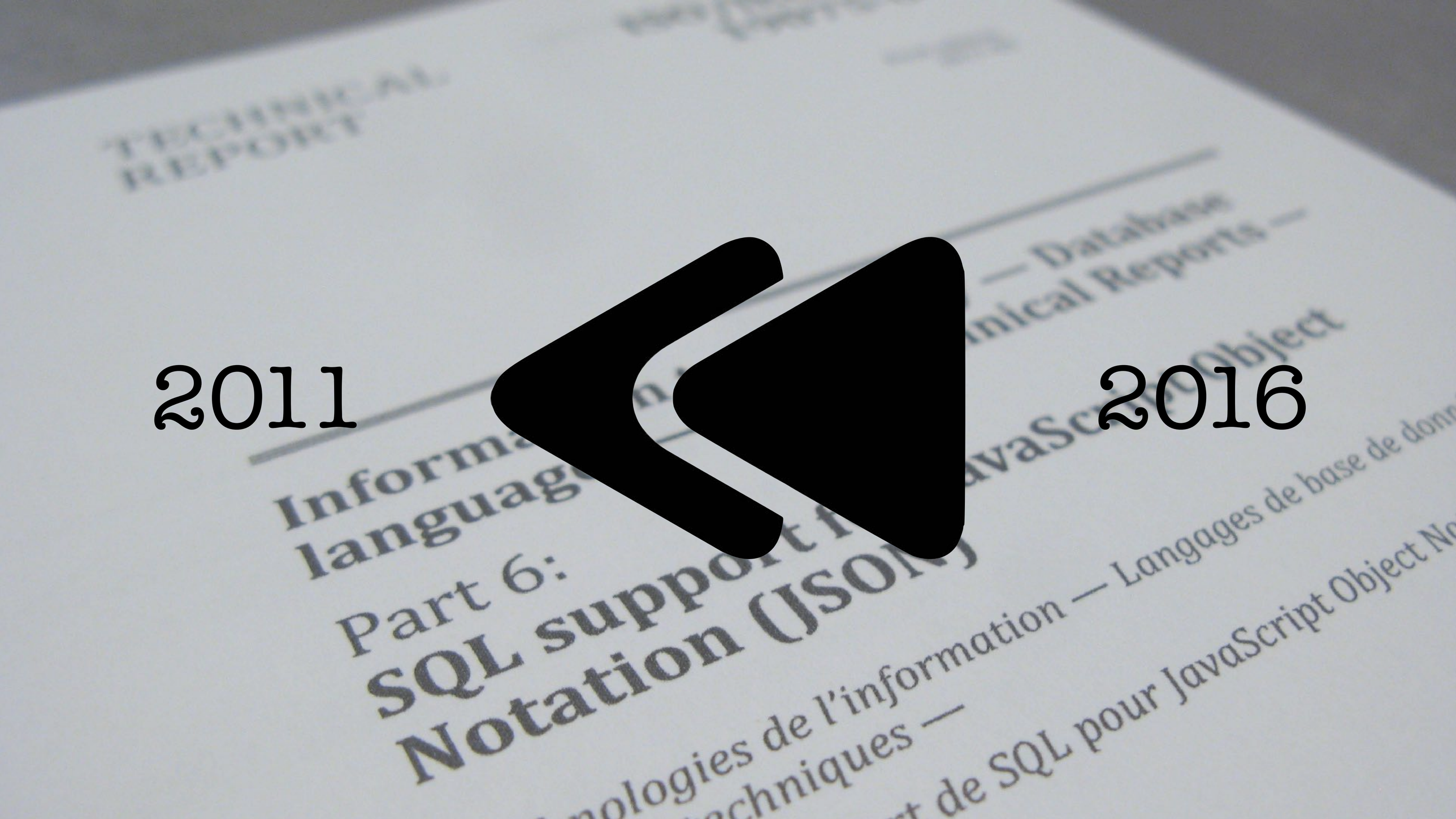



[0]Ridicoulus limitations: only 'strict $' as row expression
[1]OPENJSON provides similar functionality

# SQL:2016 — JSON_TABLE




[0]Ridicoulus limitations: only 'strict $' as row expression
[1]OPENJSON provides similar functionality


2011
2016

# SQL:2011 — Time Travelling




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?

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?

## System Versioning

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

### 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

Transparent changes, new syntax for queries

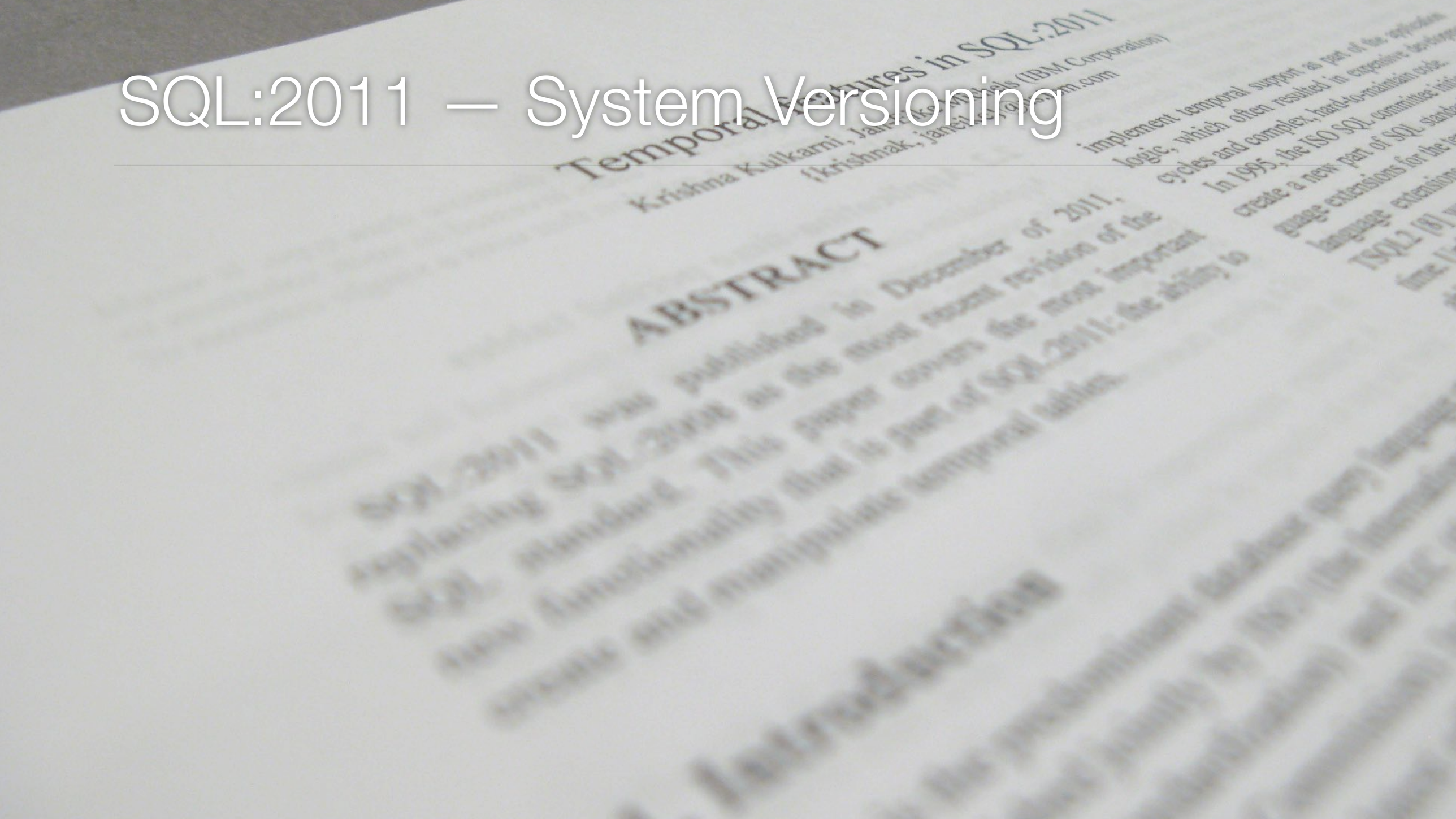
- `INSERT`, `UPDATE` & `DELETE` use the system time automatically
- `SELECT` can use `FOR SYSTEM_TIME AS OF`

# SQL:2011 — System Versioning

```
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



# 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 |

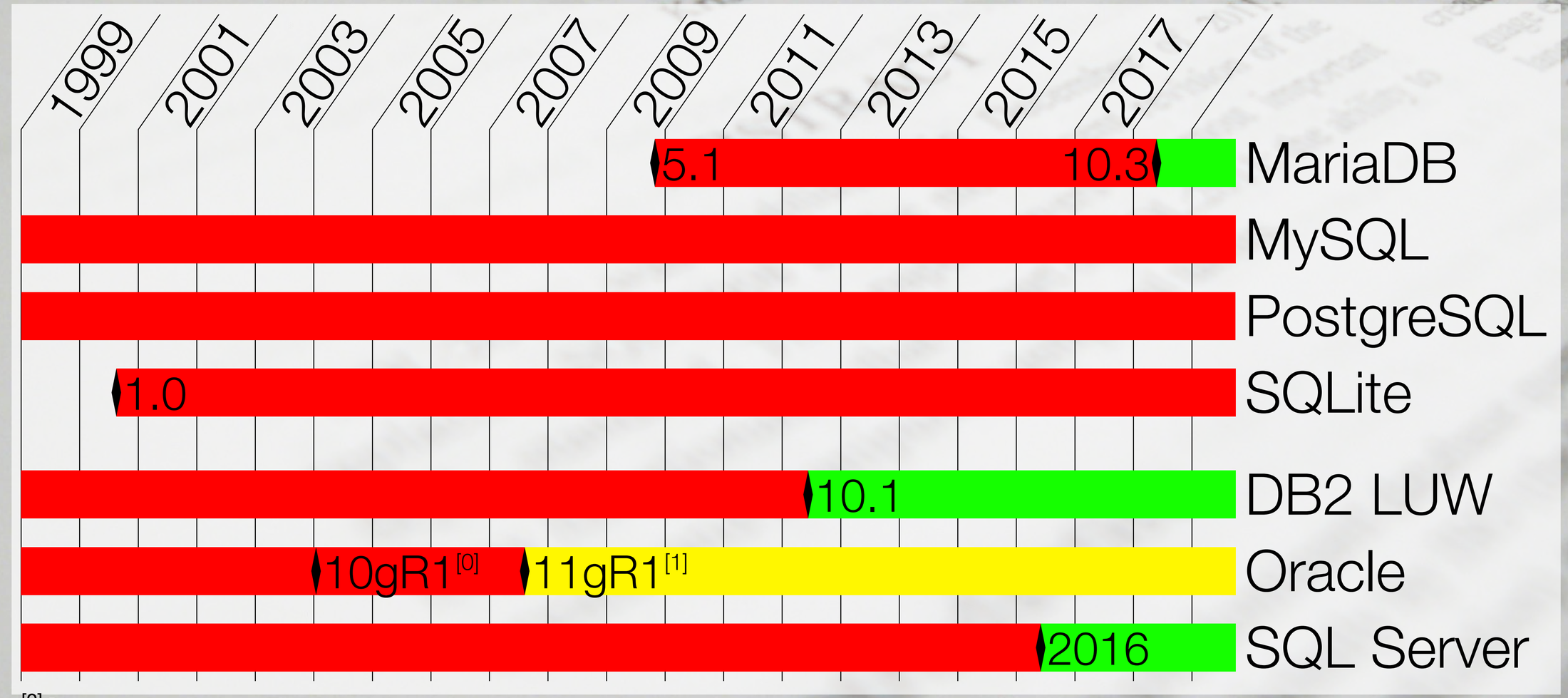
```
SELECT *
  FROM t
```

| 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 —

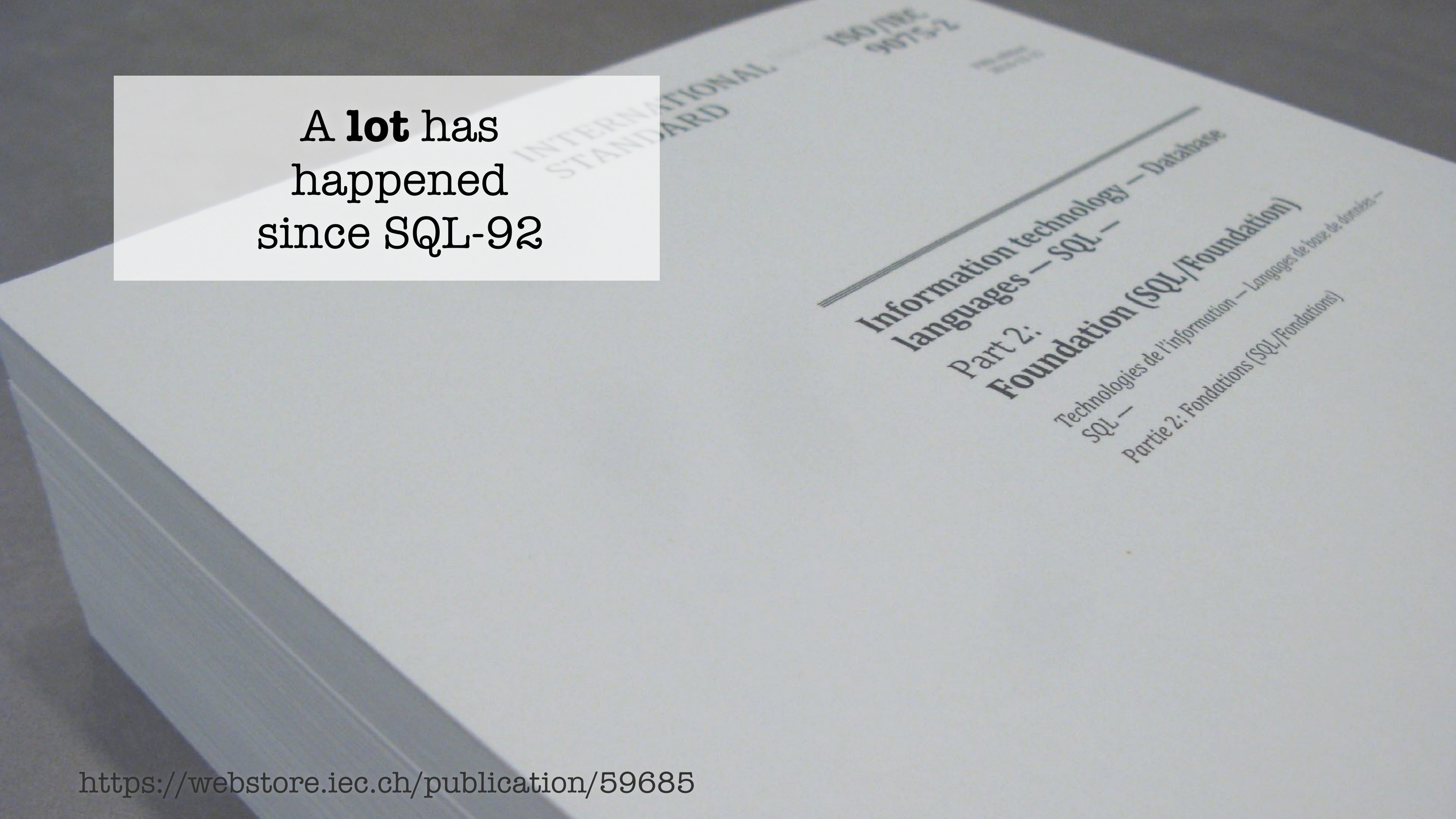
## Part 2: Foundation (SQL/Foundation)

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

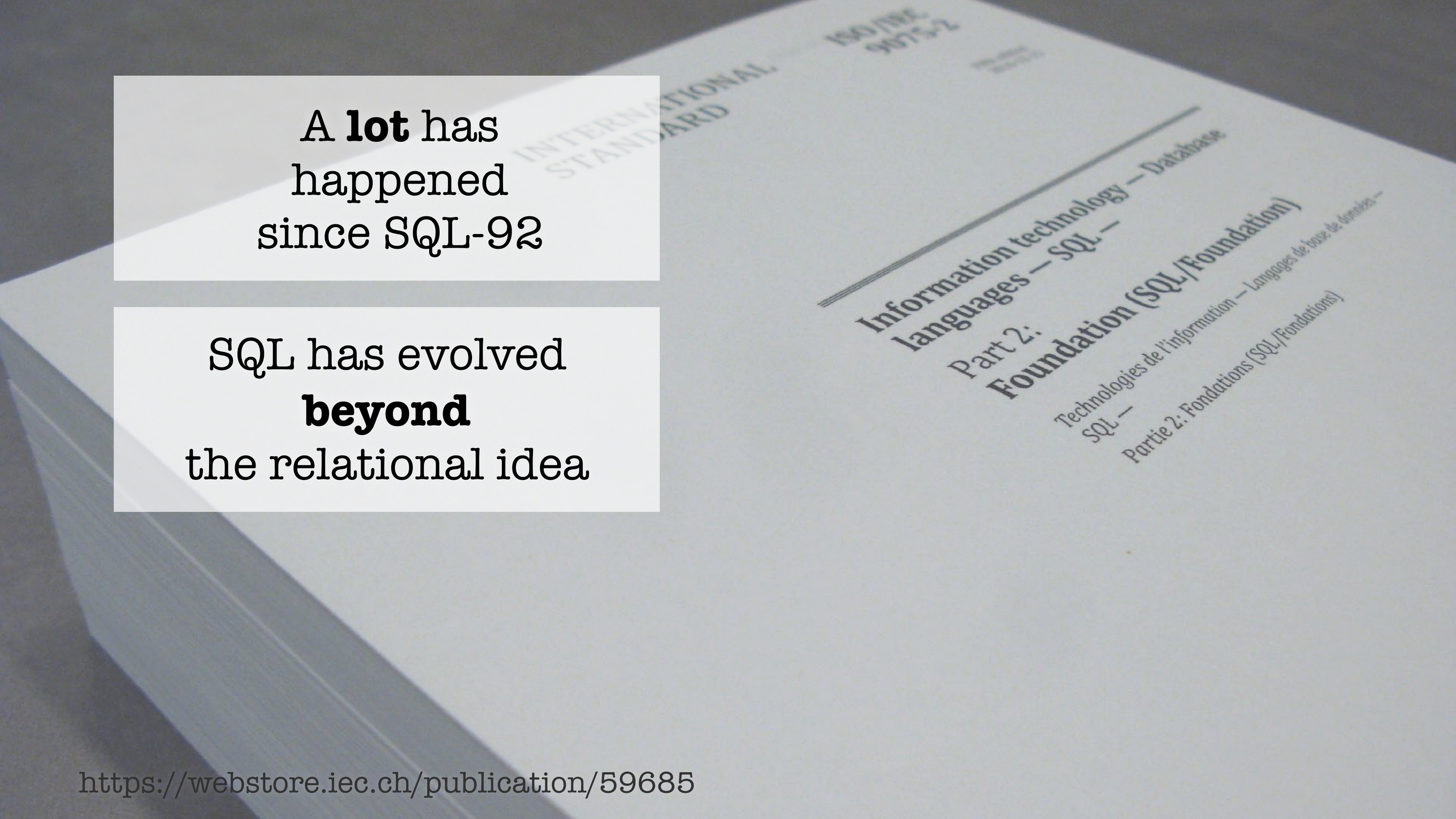
Partie 2: Fondations (SQL/Fondations)

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

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




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




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

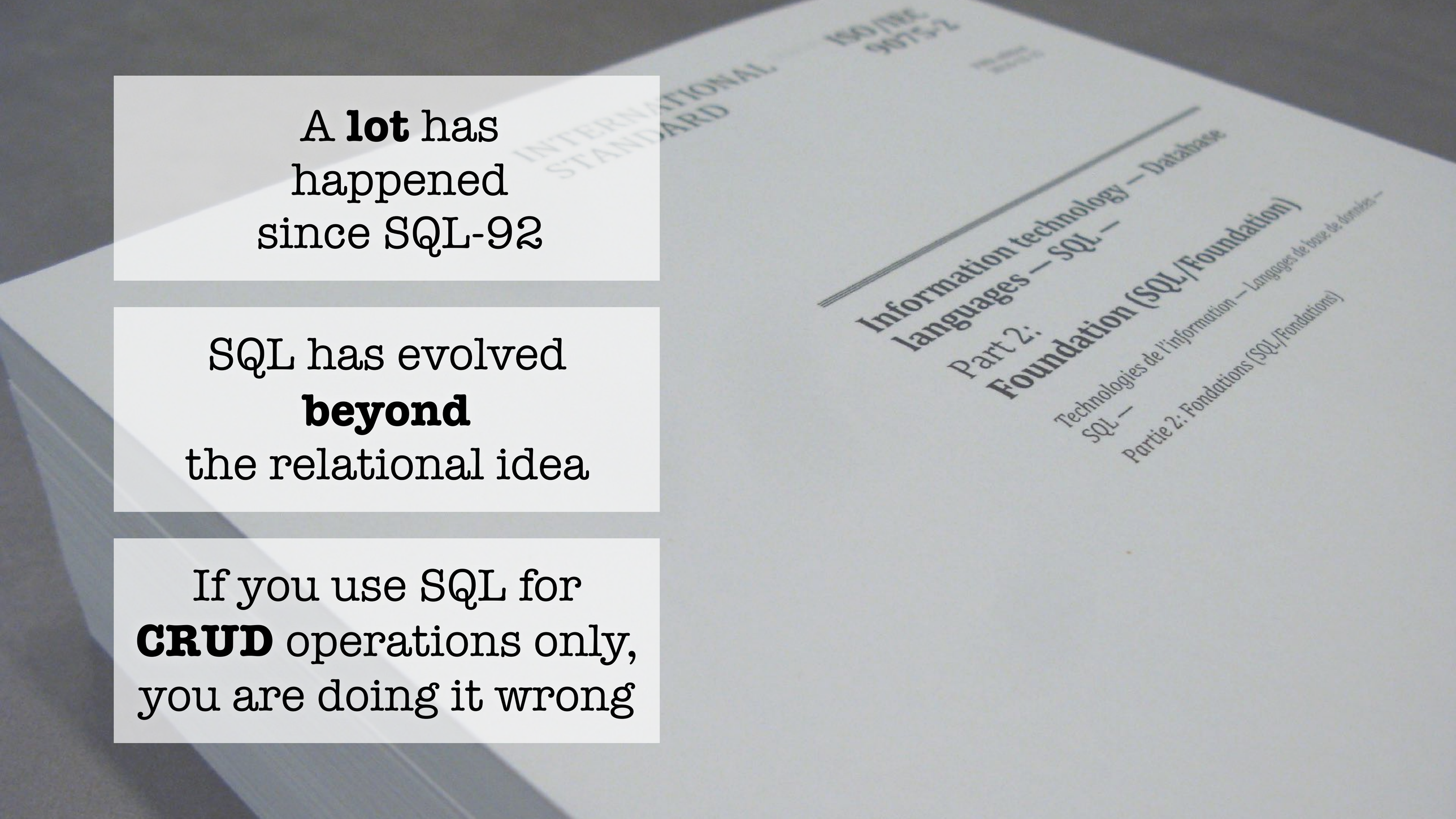
SQL has evolved
**beyond**
the relational idea

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

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



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




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

Training:
https://winand.at/