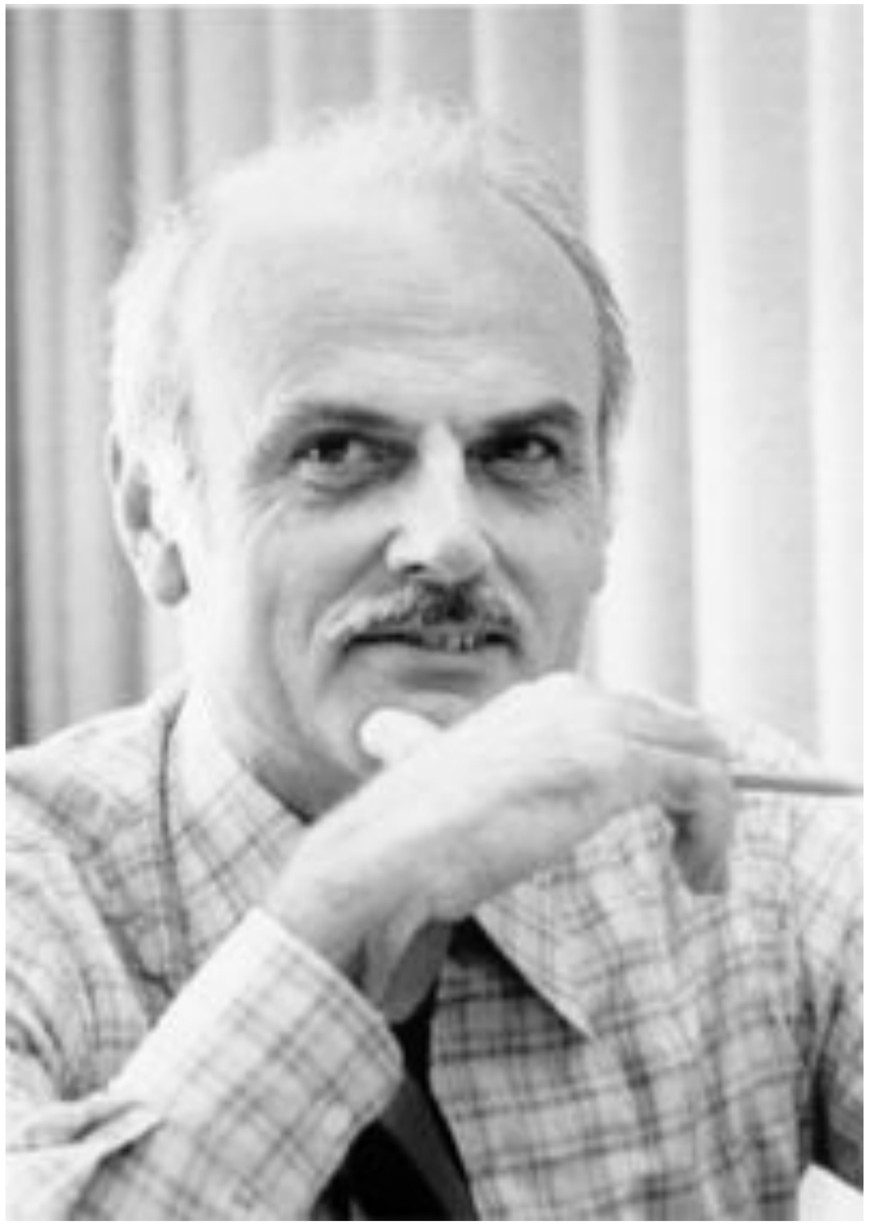
datalab

data is everywhere, value is hidden

Relational Databases

Lecturer: Азат Якупов (Azat Yakupov)
https://datalaboratory.one

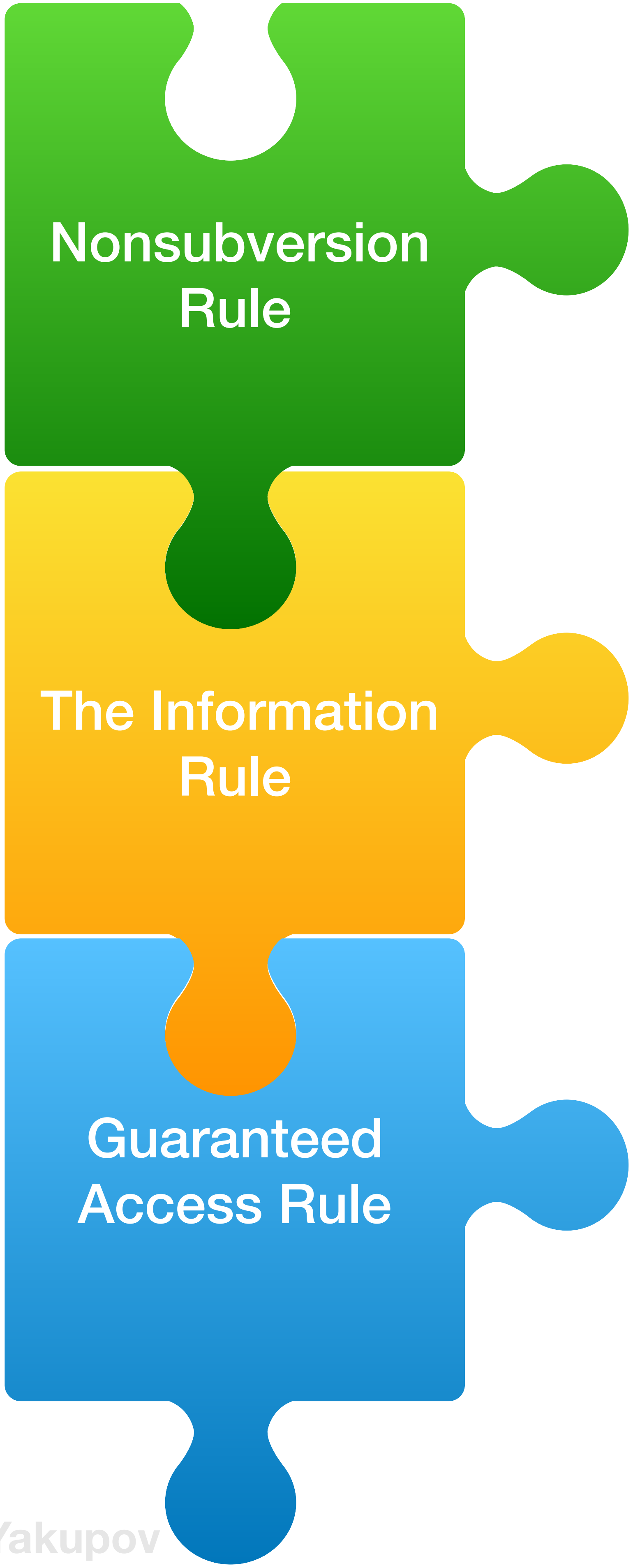


Edgar F. Codd defined a relational model in 1969.

All data is represented like ***tuples*** and grouped into ***relations***.

The purpose of the Relational Model is to provide possibility for specifying data and queries

The Relational Model was the first database model which is described in **formal mathematical terms**



Nonsubversion Rule

It’s not be possible to bypass the integrity rules defined through the database language by using lower-level languages

The Information Rule

All information in a RDBMS (including table and column names) is represented explicitly as values in tables

Guaranteed Access Rule

Every value in RDBMS is guaranteed to be accessible by using a combination of the table name, primary key value and column name

Systematic NULL value Support

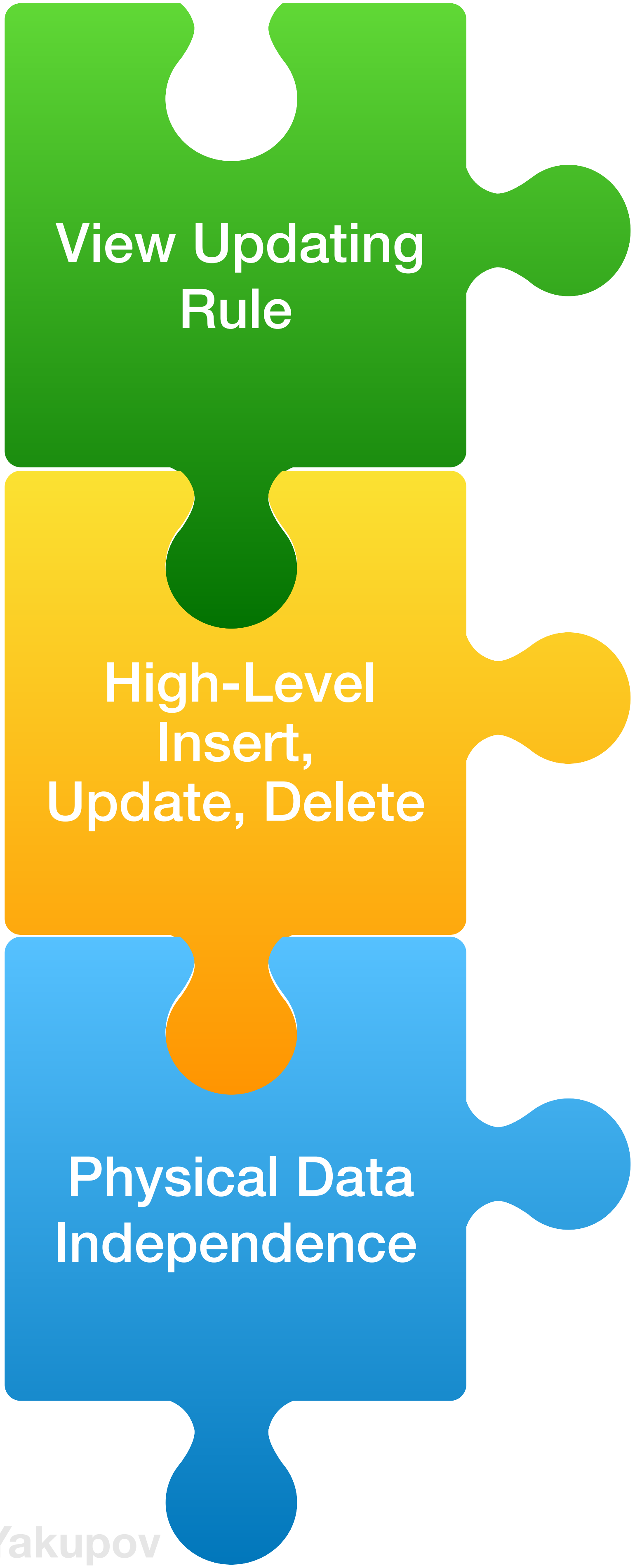
A RDBMS provides systematic support for the treatment of null values

Active Online Relational Catalog

The description of RDBMS and it’s contents is represented at the logical level as tables and can be queried using database language

Comprehensive Data Sublanguage

Must be at least one language supported with well-defined syntax.
Supports DML, DDL, integrity rules, authorisation and transactions



View Updating Rule

All views that are theoretically updatable can be updated through the system

High-Level Insert, Update, Delete

The RDBMS supports Insert, Update, Delete operations not only for one row but for any set of rows also

Physical Data Independence

Application programs are logically unaffected when physical access methods or disk storage structures are altered

Logical Data Independence

Application programs are logically unaffected, to the extent possible, when changes are made to the table structures

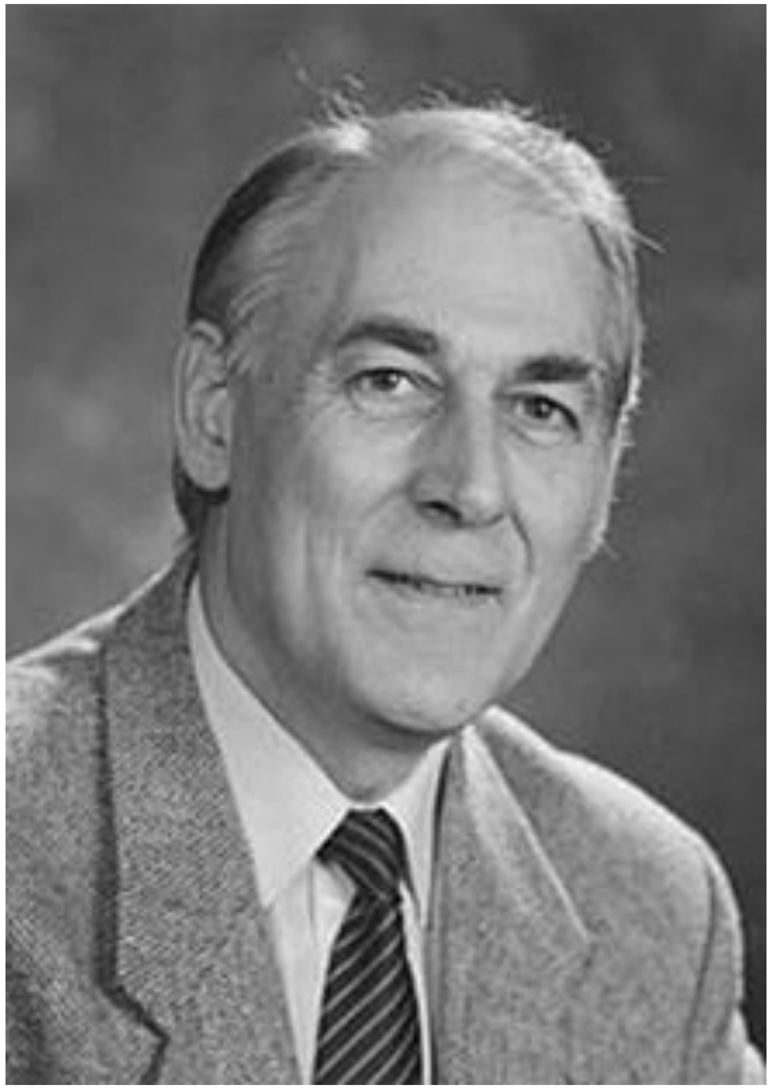
Integrity Independence

The RDBMS language must be capable of defining integrity rules. Rules must be stored in the on-line catalog and they can not be bypassed.

Distribution Independence

Application programs are logically unaffected, to the extent possible, when data is first distributed or when it is redistributed

Chris Date

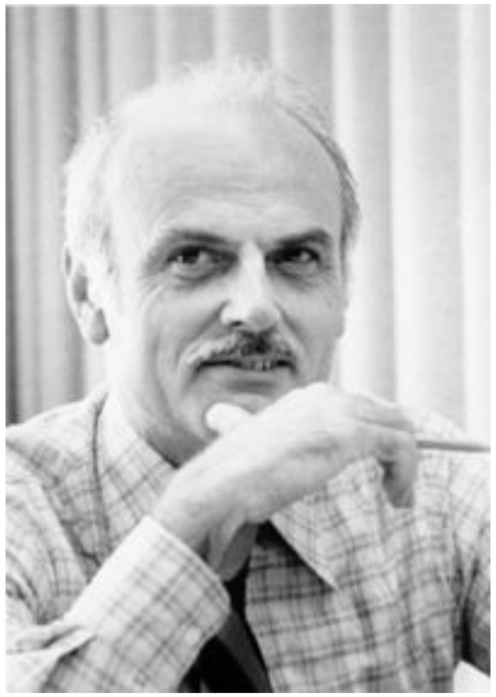


Hugh Darwen

Chris Date ,
Hugh Darwen
continued to explain an implementation of the Relational Database Model.

No one current RDBMS fully covers Relational Model design.

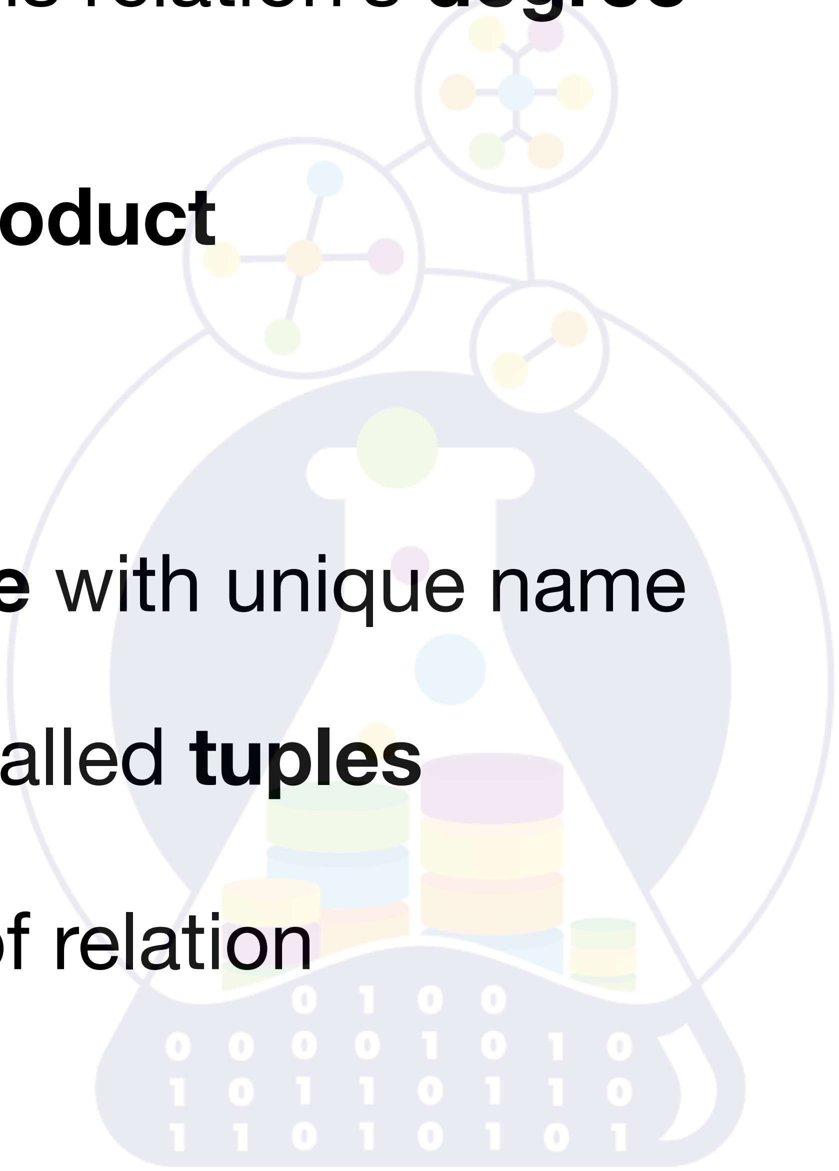
A closest physical attempt to describe **Relational Model**
is desktop database **“Rel”** completely based on **Tutorial D** language



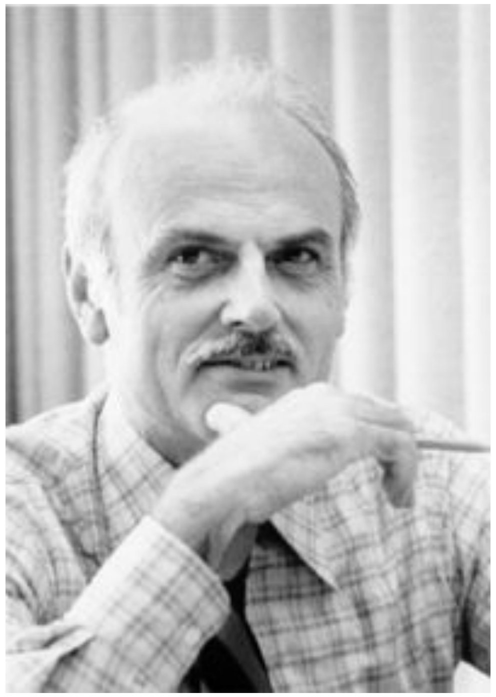
Relation R defined on sets $D_1, D_2, D_3, \ldots, D_n$

is called arbitrary subset

$R \subseteq D_1 \times D_2 \times \ldots \times D_n$, n is relation’s **degree**



- $D_1 \times D_2 \times \ldots \times D_n$ is **cartesian product**
- $D_1,\ D_2,\ \ldots,\ D_n$ are **domains**
- Named relation’s column is **attribute** with unique name
- Elements of cartesian product are called **tuples**
- Amount of all tuples is **cardinality** of relation



- $R(A, B) = R(B, A)$

A	B
1	1
1	2
3	2

=

B	A
1	1
2	1
2	3

- Relation’s **schema** is list of attributes names with **domains**

  R with attributes $A_1, A_2, \ldots, A_k$ has schema $R(A_1, A_2, \ldots, A_k)$

Relation R contains 2 **main elements**

- **Header** - is tuple’s header. Relation has the same attributes and the same degree level like defined header
- **Body** - is a set of tuples with the same header. Cardinality of relation equals amount of tuples from defined set

$$R = RELATION\{A_1 : T_1, A_2 : T_2 \ldots, A_n : T_n\}$$

$A_1, A_2, \ldots, A_n$ are **attributes**

$T_1, T_2, \ldots, T_n$ are **types**

$$R = RELATION\{A : integer, B : integer, \ldots, C : string\}$$

A : integer	B : integer	C : string
1	1	‘string #1’
1	2	‘string #1’
3	2	‘string #3’

A	B
1	1
1	2
3	2

R

×

W	Y	Z
1	1	1
3	2	1

S

=

A	B	W	Y	Z
1	1	1	1	1
1	1	3	2	1
1	2	1	1	1
1	2	3	2	1
3	2	1	1	1
3	2	3	2	1

Z

dID	dName	dCntProjects	dAvgPoint
1	Ivan	3	5
2	Peter	2	3,5

×

pName	pManagerName	pPriority
Project #1	Ivan Ivanov	high

=

dID	dName	dCntProjects	dAvgPoint	pName	pManagerName	pPriority
1	Ivan	3	5	Project #1	Ivan Ivanov	high
2	Peter	2	3,5	Project #1	Ivan Ivanov	high

Mathematical Model

The same tuple cannot appear more than once in a ***relation***

$$R(A, B, C) =$$

A	B	C
1	2	3
1	1	1
2	3	1
1	2	3

Reality

The same row can appear more than once in an ***SQL table***

```
SELECT A,B,C
FROM R;
```

A	B	C
1	2	3
1	1	1
2	3	1
1	2	3

Any ordering for tuples in a ***relation***

$R(A, B, C) =$

A	B	C
1	2	3
1	1	1
2	3	1
1	2	4

Set **ORDER BY** clause for ***table*** explicitly.

```
SELECT A,B,C
  FROM R
ORDER BY A,B,C;
```

A↓	B↓	C↓
1	1	1
1	2	3
1	2	4
2	3	1

Any ordering for attributes in a ***relation***

$$R(A, B, F, C) =$$

A	B	F	C
1	2	0	3
1	1	0	1
2	3	0	1
1	2	0	4

There is a defined ordering in metadata for columns. We can play with ordering in a **SELECT** clause

```
SELECT *
  FROM R;
```

A	B	C	F
1	2	3	0
1	1	1	0
2	3	1	0
1	2	4	0

Value of each attribute is atomic for a ***relation***

Mathematical Model

$$R(A, B, C) =$$

A	B	C
1	2	3
1	1	1
2	3	1
1	2	4

To reach the better performance we can avoid a relational model

Reality

```
SELECT *
  FROM R;
```

A	Point
1	(2, 3)
1	(1, 1)
2	(3, 1)
1	(2, 4)

Mathematical Model

No way to use unnamed attribute in a ***relation***

$R(A, B, C) =$

A	B	C
1	2	3
1	1	1
2	3	1
1	2	4

Reality

We can set unnamed column in **SQL query**.

```
SELECT A,B,A+B
  FROM R;
```

A	B	?
1	2	3
1	1	2
2	3	5
1	2	3

No way to make duplicate names for a attributes in a ***relation***

$$R(A, B, C) =$$

A	B	C
1	2	3
1	1	1
2	3	1
1	2	4

We can set column’s duplicates by **SQL query**.

```
SELECT A,
       B AS A,
       C
  FROM R;
```

A	A	C
1	2	3
1	1	1
2	3	1
1	2	4

NULL (ω) means “missing or inapplicable information”

Mathematical Model

$R(A, B, C) =$

A	B	C
ω	2	3
1	1	ω
2	ω	1
1	2	4

We can use/get a keyword **NULL**

Reality

```
SELECT *
  FROM R;
```

A	B	C
null	2	3
1	1	***null***
2	***null***	1
1	2	4

```
SELECT *
  FROM R;
```

A	B	C
null	2	3
1	1	*null*
2	*null*	1
1	2	4

```
SELECT *
  FROM R
WHERE (A = 1)
      OR (A!= 1);
```

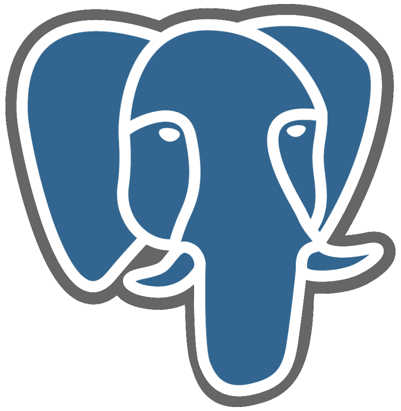
A	B	C
1	1	*null*
2	*null*	1
1	2	4

```
SELECT *
  FROM R
WHERE (A = 1) OR
          (A!= 1) OR
           A IS NULL;
```

A	B	C
null	2	3
1	1	*null*
2	*null*	1
1	2	4

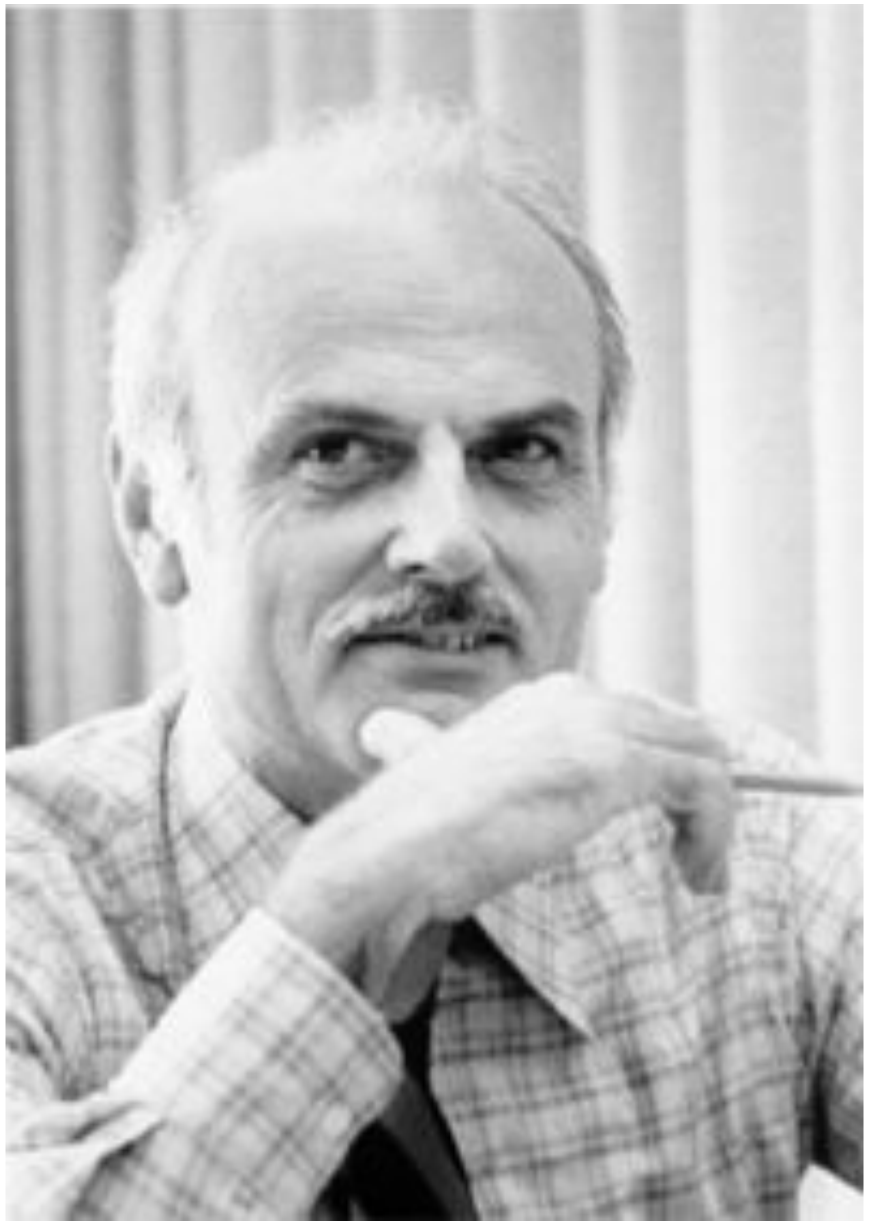
ORACLE

```
'Hello '|| NULL ||' world!' ➔ 'Hello world!'
```



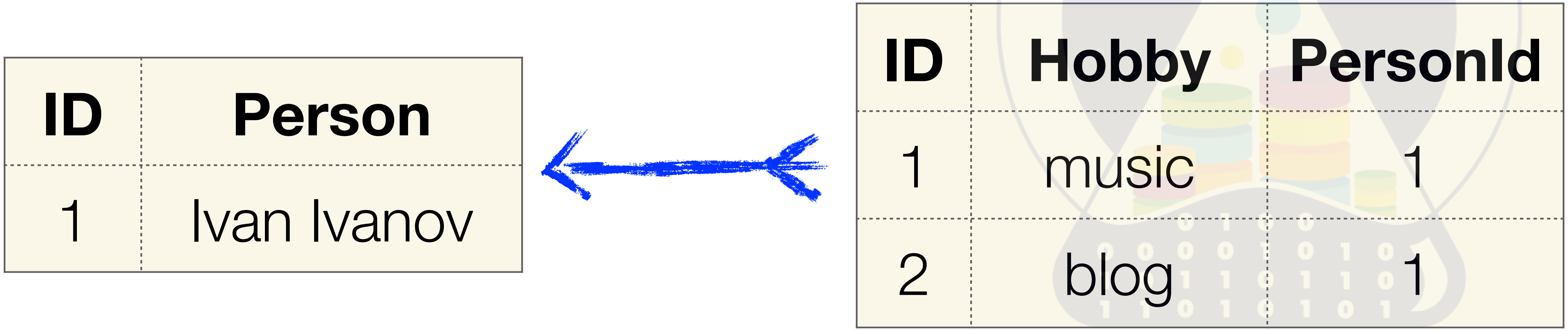
MySQL

```
'Hello '|| NULL ||' world!' ➔ NULL
```



“All information in the database must be cast explicitly in terms of values in relations and in no other way”

Information Principle



ID	Person
1	Ivan Ivanov

ID	Hobby	PersonId
1	music	1
2	blog	1

$R(\emptyset)$ means **no attributes** for R, or **relation's degree** equals 0

There are 2 **pseudonymous relations** by Hugh Darwen

TABLE_DEE (~ DEE) - a relation $R(\emptyset)$ with **one zero-tuple!**

$$RELATION\{\ \}\ \{\ TUPLE\ \{\ \}\ \}\quad \sim True$$

TABLE_DUM (~ DUM) - a relation $R(\emptyset)$ without **any tuples!**

$$RELATION\{\ \}\ \{\ \}\quad \sim False$$

- $R = S$
- $R \neq S$
- $R \subseteq S$
- $R \subset S$
- $R \supseteq S$
- $R \supset S$

R

A	B
1	1
2	0
1	2

S

A	B	C
1	4	5
2	3	3
1	2	1

$$R(A) = S(A)$$

$$R(B) \neq S(B)$$

$$R(A) \subseteq S(A)$$

$$R(A) \supseteq S(A)$$

$$IS\_EMPTY(\ <relational\_exp>\ ) = True \mid False$$

R

A	B
1	1
2	0
1	2

S

A	B	C
1	4	5
2	3	3
1	2	1

$$IS\_EMPTY(\ R(A,B)\ ) = False$$

$$IS\_EMPTY(\ R(A,B) \times S(C)\ ) = False$$

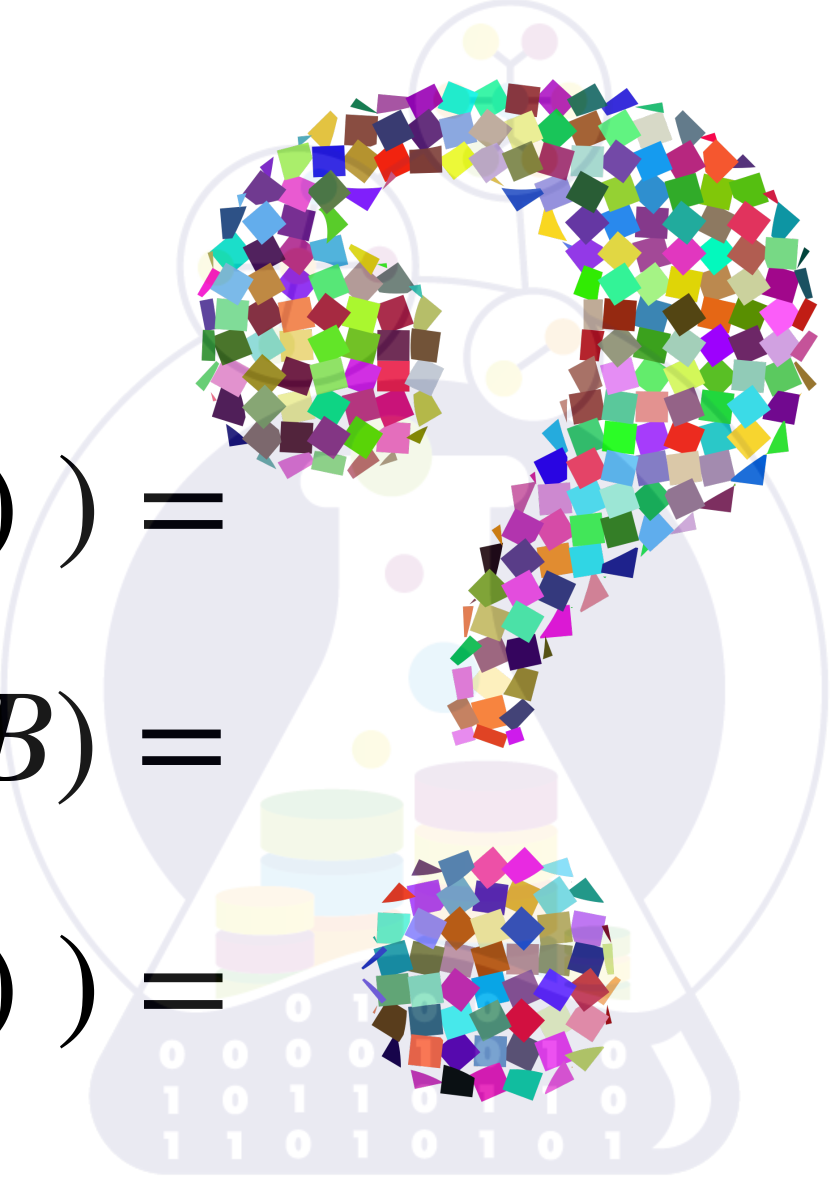
$$R(A, B) =$$

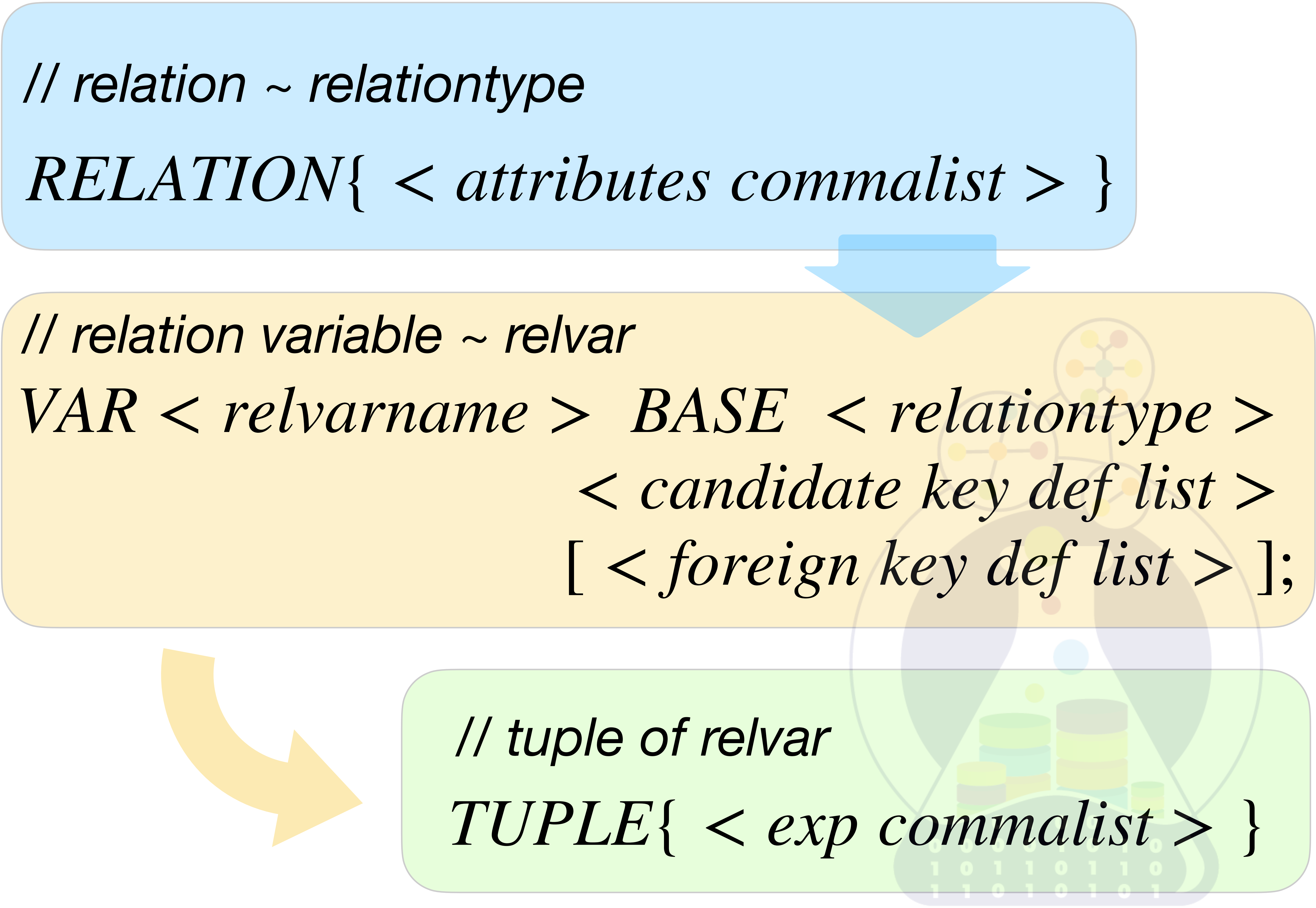
A	B
1	*null*
2	*null*
1	*null*

$$IS\_EMPTY(\ R(B)\ ) =$$

$$R(B) =$$

$$IS\_EMPTY(\ R(B) \times S(\text{Ø})\ ) =$$





// relation ~ relationtype

RELATION{ < attributes commalist > }

// relation variable ~ relvar

VAR < relvarname > BASE < relationtype >
< candidate key def list >
[< foreign key def list >];

// tuple of relvar

TUPLE{ < exp commalist > }

$$R = RELATION\{A : integer, B : integer, C : string\};$$

$$
\begin{aligned}
&VAR\ rel\ BASE\ R\\
&\qquad \{\ A\ INTEGER,\\
&\qquad\ \ B\ INTEGER,\\
&\qquad\ \ C\ STRING\ \}\\
&\qquad\ \ PRIMARY\ KEY\ \{A, B\};
\end{aligned}
$$

A : integer	B : integer	C : string
1	1	string #1
1	2	string #1
3	2	string #3

$$TUPLE\{integer(1), integer(1), string('string\ \#1')\};$$

$$TUPLE\{integer(1), integer(2), string('string\ \#1')\};$$

$$TUPLE\{integer(3), integer(2), string('string\ \#3')\};$$

- $rel' := rel$

A	B	C
1	1	string #1
1	2	string #1
3	2	string #3

- $rel' := rel$
  $WHERE\ B = 1$

A	B	C
1	1	string #1

- $rel' := rel$
  $WHERE\ C\ NOT\ ('string\ \#1')$

A	B	C
3	2	string #3

```
INSERT rel RELATION { TUPLE {
                              A INTEGER (4),
                              B INTEGER (7),
                              C STRING ('Hello')}};


rel := rel UNION RELATION { TUPLE {
                                        A INTEGER (4),
                                        B INTEGER (7),
                                        C STRING ('Hello')}};
```

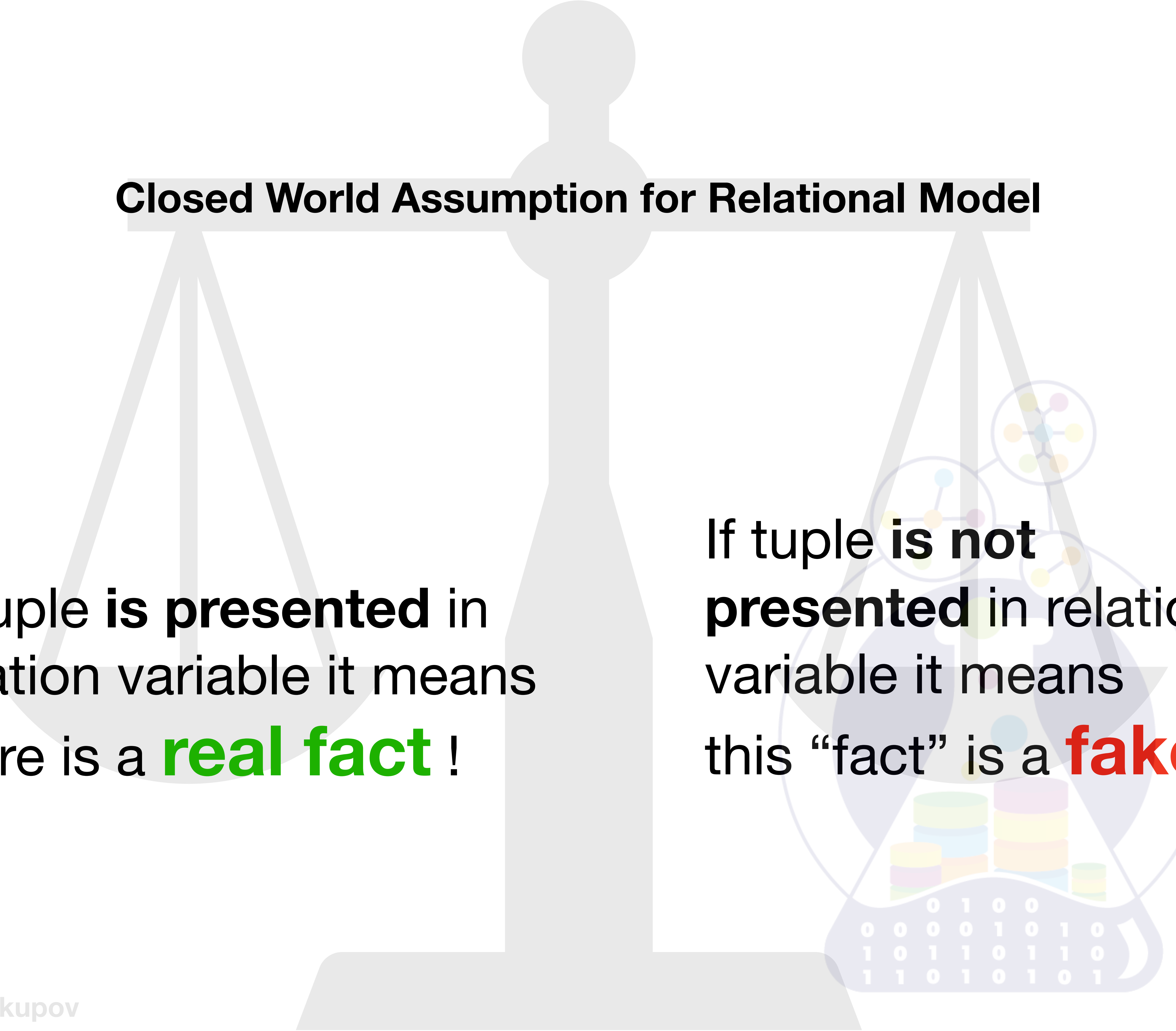
DELETE rel WHERE A = 1;

rel := *rel WHERE NOT* (*A* = 1);

$$
\begin{aligned}
&UPDATE\ rel\ WHERE\ A = 1\\
&\qquad\qquad\qquad\qquad \{B := 23 * A,\\
&\qquad\qquad\qquad\qquad\ C :='\ string\ \#4'\};
\end{aligned}
$$

$$
\begin{aligned}
&rel := WITH\ (rel\ WHERE\ A = 1)\ AS\ T1,\\
&\quad (EXTEND\ T1\ ADD\ (23 * A\ AS\ NEW\_B,\\
&\qquad\qquad\qquad 'string\ \#4'AS\ NEW\_C))\ AS\ T2,\\
&\quad T2\ \{\ ALL\ BUT\ B, C\}\ AS\ T3,\\
&(T3\ RENAME\ (NEW\_B\ AS\ B, NEW\_C\ AS\ C))\ AS\ T4 :\\
&(S\ MINUS\ T1)\ UNION\ T4;
\end{aligned}
$$

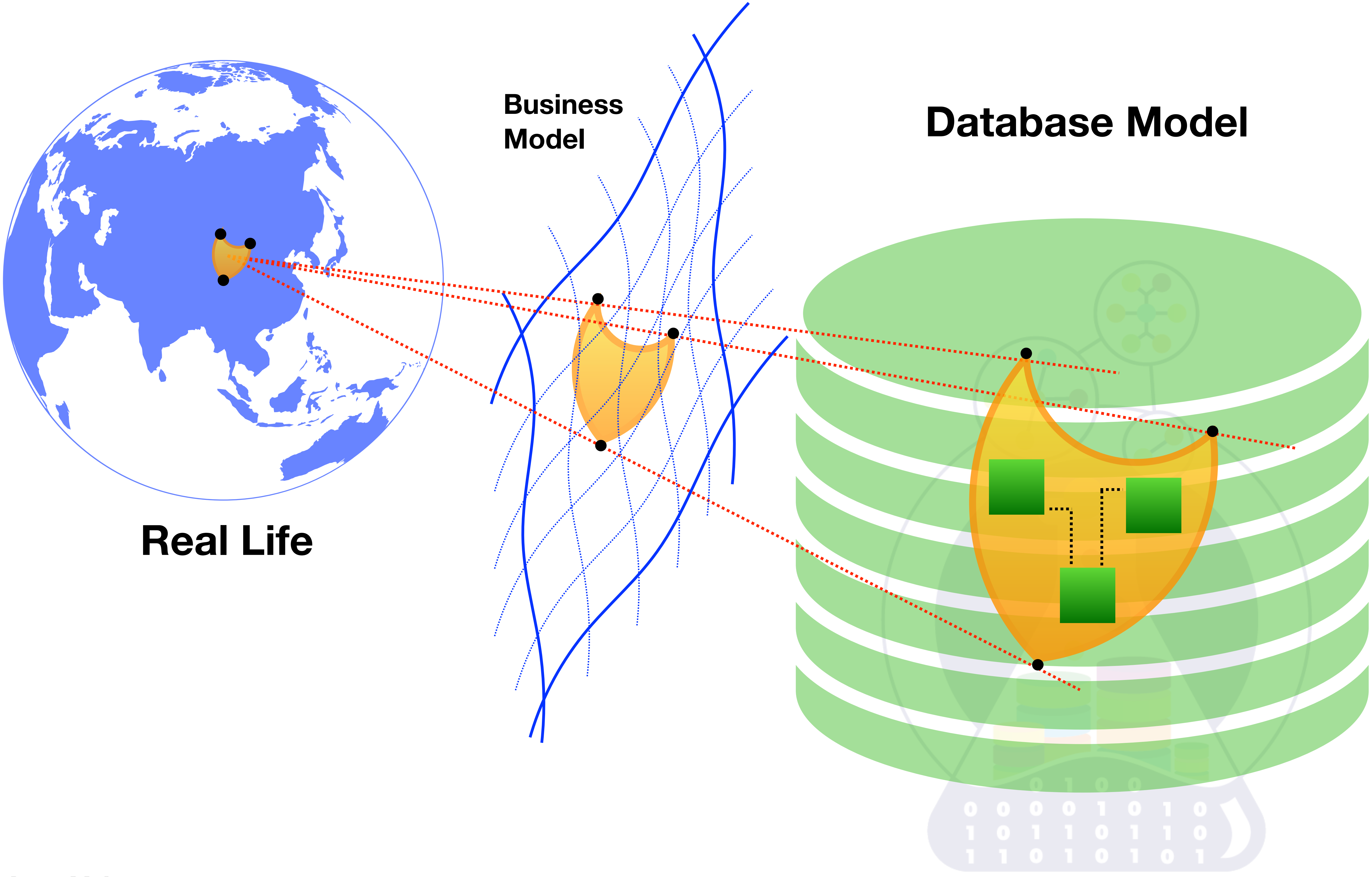
Closed World Assumption for Relational Model



If tuple **is presented** in relation variable it means there is a **real fact** !

If tuple **is not presented** in relation variable it means this “fact” is a **fake**!

Azat Yakupov



Business Model

Database Model

Real Life

Integrity by entities (~ not null for primary key)

Type's integrity

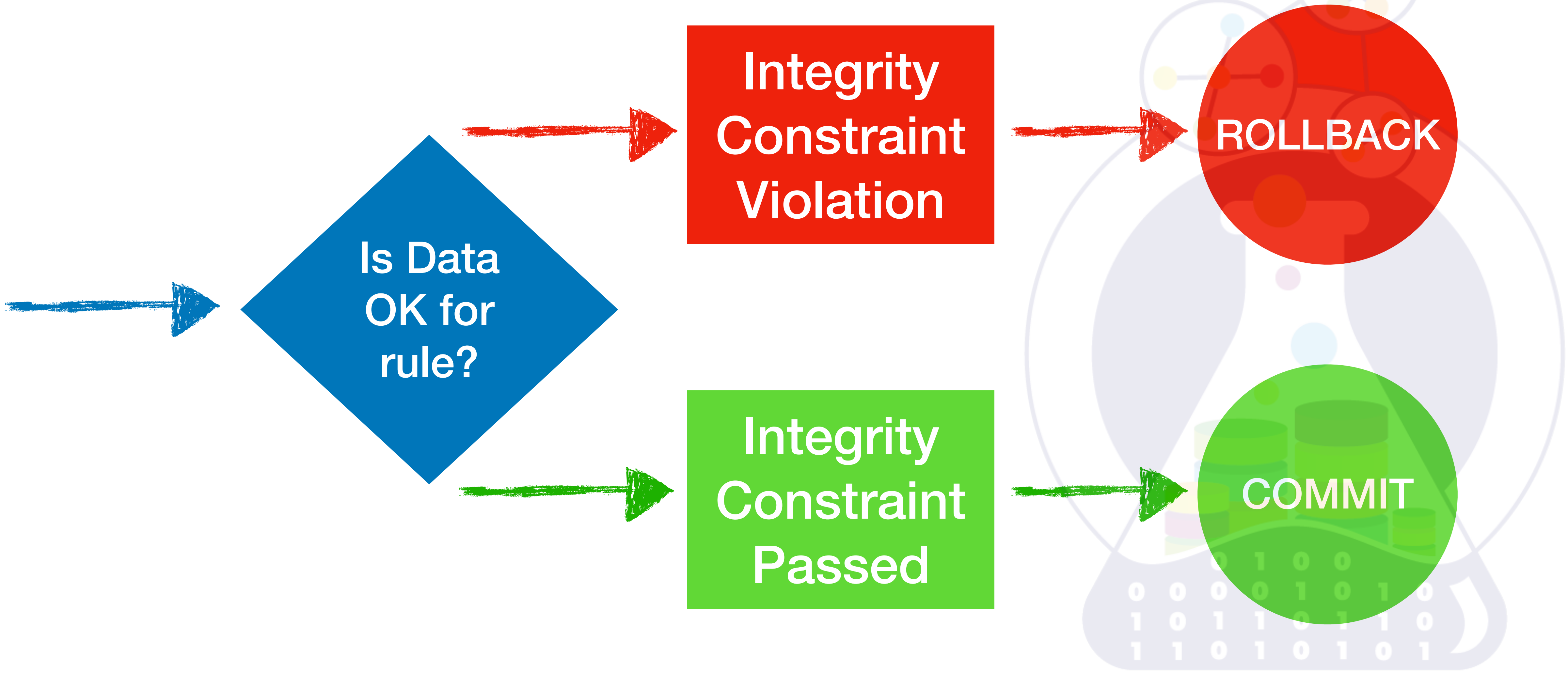
User-defined integrities

- **Unique keys**
- **Range values**
- **Foreign keys**
- **List values and RegExp**
- **Database Triggers / Database Rules**

System can control **consistency** only but **not truth** about data

C.J. Date

Integrity Constraint - logical expression is returning **TRUE** or **FALSE**



Type’s Integrity

```
TYPE weight POSSREP* {D DECIMAL (5,1)
                 CONSTRAINT D > 0.0
                 AND D < 5000.0 };
```

* **POSSREP** means **POSS**ible **REP**resentation

Attribute’s Integrity

```
VAR rel BASE R
        { A INTEGER,    1
          B INTEGER,    2
          C STRING }    3
```

Relation Variable’s Integrity

$$CONSTRAINT\ SC1$$

$$FORALL\ SX(SX.STATUS \geq 1$$

$$AND\ SX.STATUS \leq 100)$$

Relation Variable’s Integrity

$$CONSTRAINT\ SC2$$

$$FORALL\ SX\ (\ IF^{*}\ SX.CITY = '\ London'$$

$$THEN^{*}\ SX.STATUS = 20\ END\ IF\ );$$

* $IF\ p\ THEN\ q$ where p, q are logical expressions

Database Integrity

$$CONSTRAINT\ TRC2$$

$$FORALL\ PX$$

$$SUM\ (SPX_1\ WHERE\ SPX_1\,.\,P\# = PX\,.\,P\#,\ QTY) \leq$$

$$SUM\ (SPX\ WHERE\ SPX\,.\,P\# = PX\,.\,P\#,\ QTY)$$

```
CREATE TABLE Employee
(
  ID NUMBER,
  SALARY DECIMAL(9,2)
            CONSTRAINT CH_SAL
                        CHECK (SALARY>=100000),
  DNAME VARCHAR(10)
            CONSTRAINT CH_DNAME
                       CHECK (DNAME IN (‘HR’, ‘IT’)),
  BONUS DECIMAL(9,2) DEFAULT 0
);
```

```
ALTER TABLE Employee ADD CONSTRAINT
CH_NN_SALARY CHECK (SALARY IS NOT NULL);


ALTER TABLE Employee ADD CONSTRAINT
CH_NN_BONUS CHECK (BONUS IS NOT NULL);
```

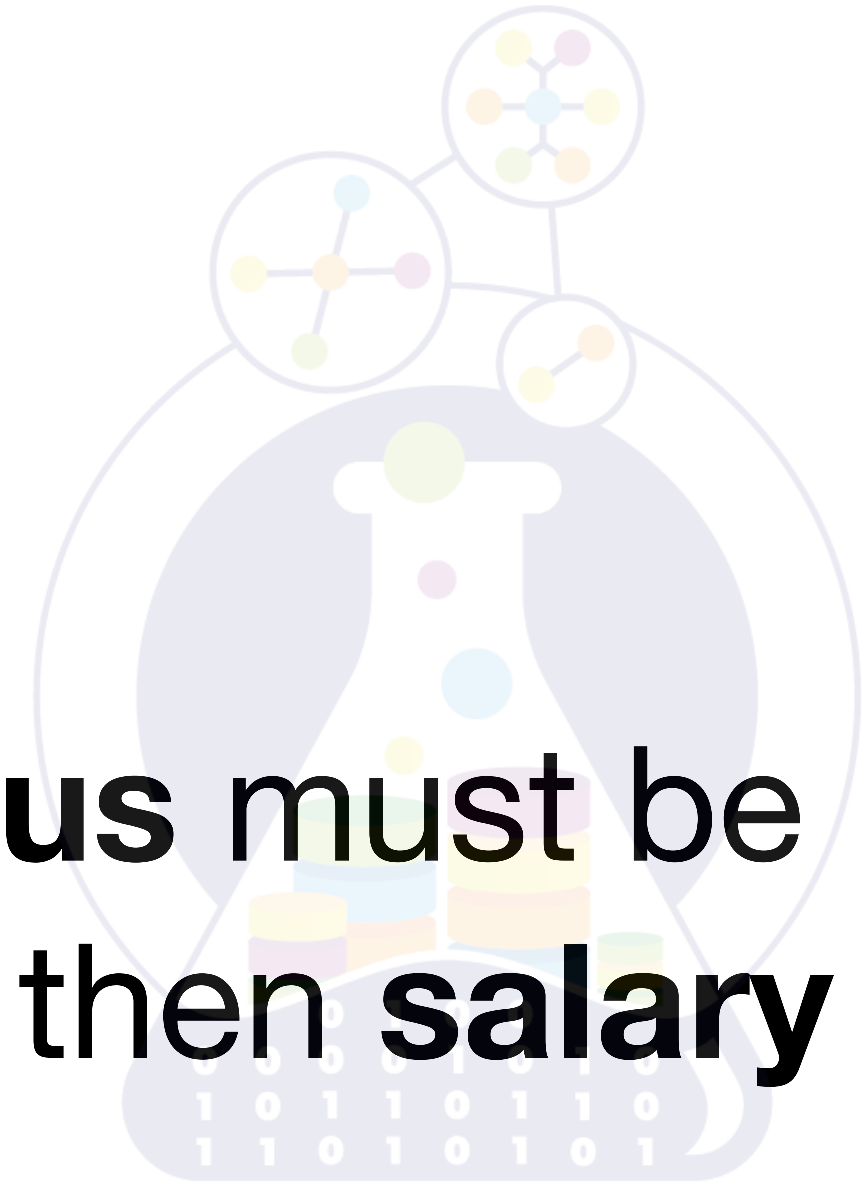
```
CREATE TABLE Employee
(
  ID NUMBER NOT NULL,
  SALARY DECIMAL(9,2)  NOT NULL
            CONSTRAINT CH_SAL
                          CHECK (SALARY>=100000),
  DNAME VARCHAR(10)
            CONSTRAINT CH_DNAME
                          CHECK (DNAME IN (‘HR’, ‘IT’)),
  BONUS DECIMAL(9,2) DEFAULT 0 NOT NULL
);
```

Add possibility to register employees from **Finance** department

2

1

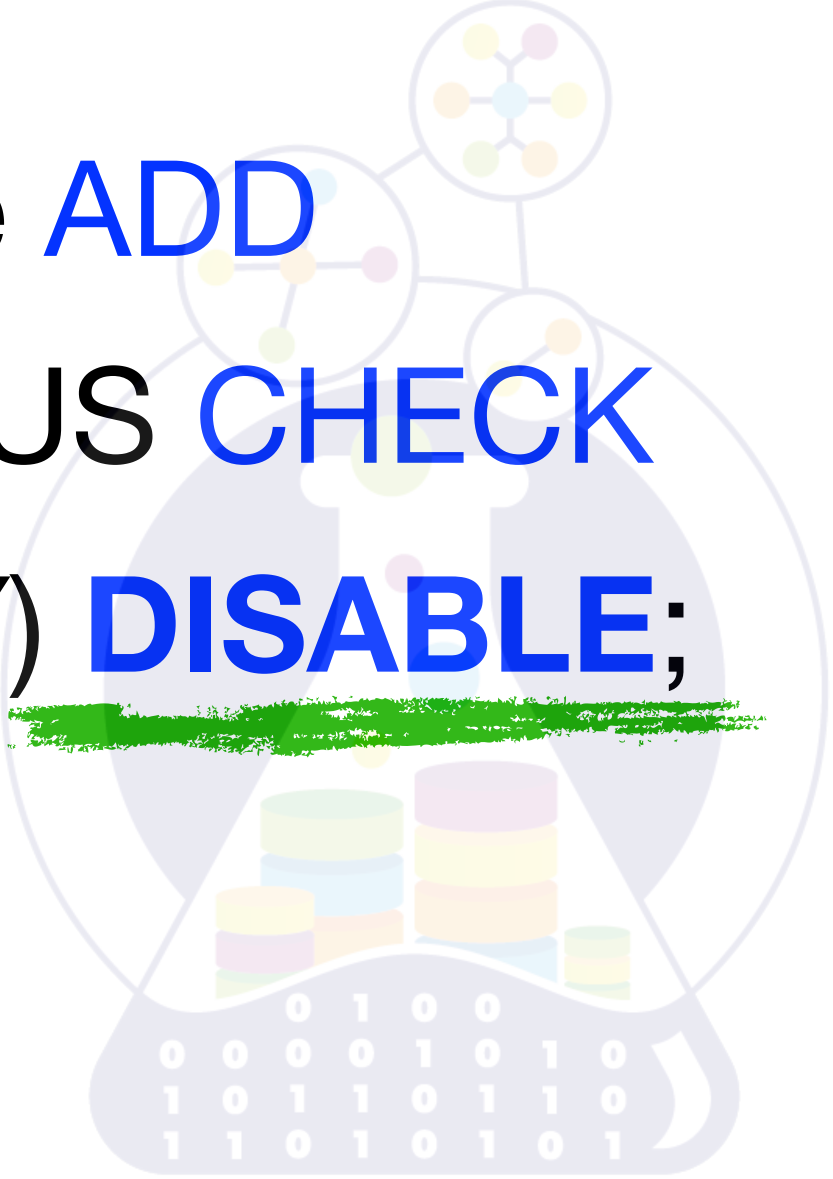
Bonus must be less then **salary**



ORACLE

1

```
ALTER TABLE Employee ADD
CONSTRAINT CH_BONUS CHECK
        (BONUS < SALARY) DISABLE;
```



ORACLE

1

```
SELECT *
  FROM Employee
WHERE BONUS >= SALARY;
```

- delete rows
- update BONUS values
- change constraint rule

ID	SALARY	DNAME	BONUS
3	100 000	IT	500 000
102	123 000	HR	500 000
34	231 000	IT	500 000

```
ALTER TABLE Employee ENABLE
CONSTRAINT CH_BONUS;
```

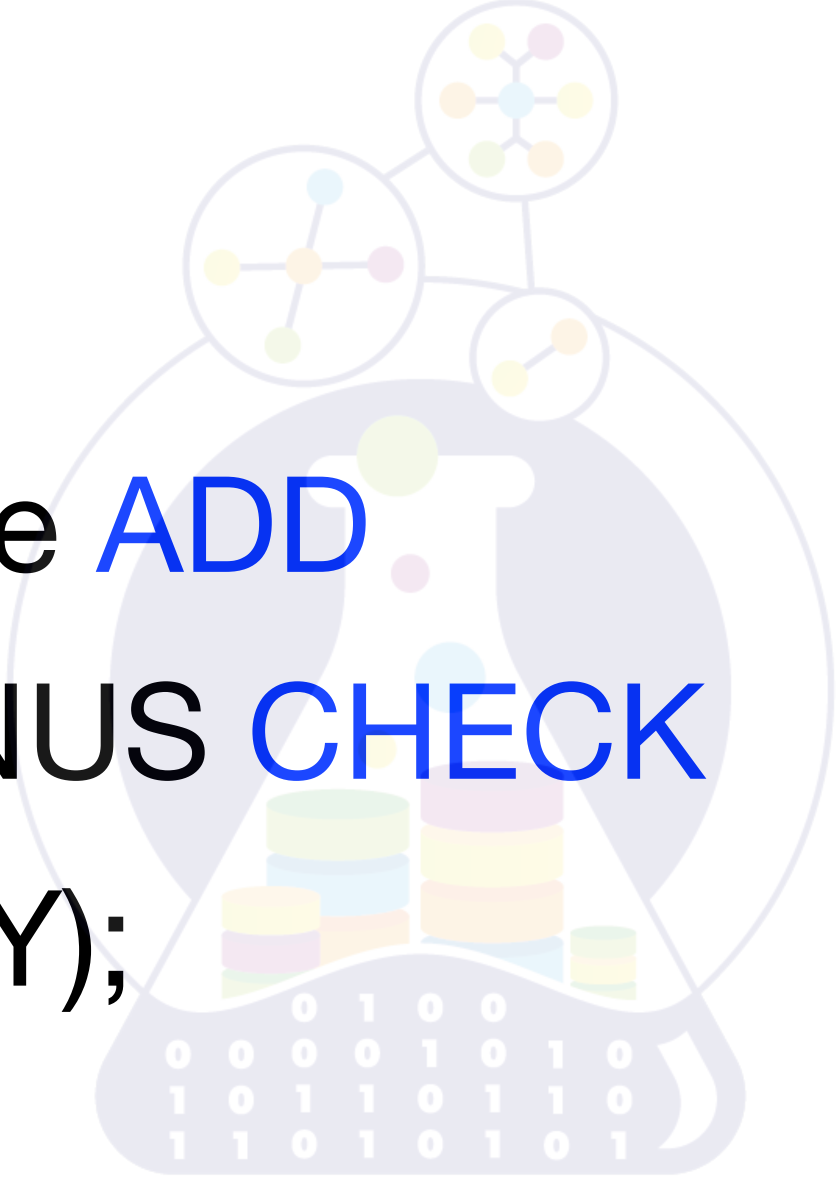
Azat Yakupov

1

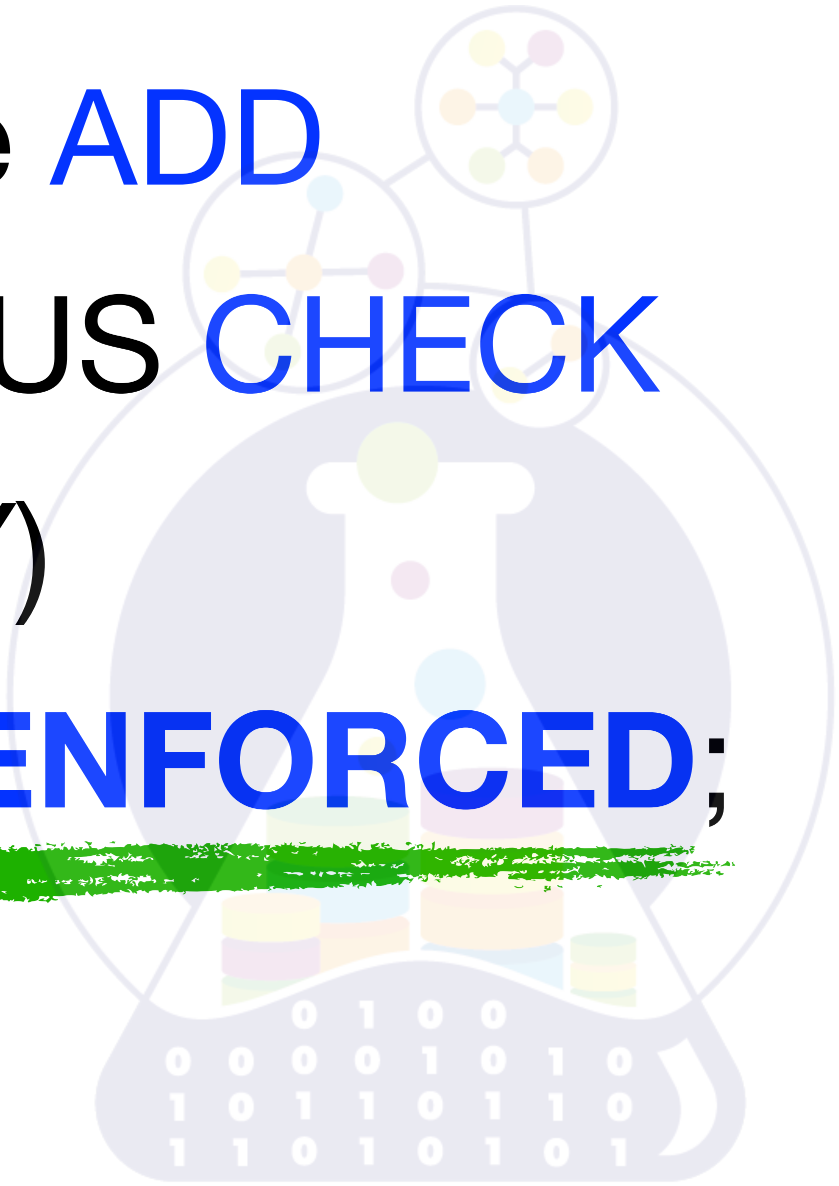
✓

```
SELECT *
  FROM Employee
WHERE BONUS >= SALARY;
```

```
ALTER TABLE Employee ADD
CONSTRAINT CH_BONUS CHECK
        (BONUS < SALARY);
```



1



```
ALTER TABLE Employee ADD
CONSTRAINT CH_BONUS CHECK
        (BONUS < SALARY)
                      NOT ENFORCED;
```

Azat Yakupov

1

```
SELECT *
   FROM Employee
WHERE BONUS >= SALARY;
```

- delete rows
- update BONUS values
- change constraint rule

ID	SALARY	DNAME	BONUS
3	100 000	IT	500 000
102	123 000	HR	500 000
34	231 000	IT	500 000

```
ALTER TABLE Employee ALTER
CHECK CH_BONUS ENFORCED;
```

2

```
ALTER TABLE Employee DROP
CONSTRAINT CH_DNAME;

ALTER TABLE Employee ADD
CONSTRAINT CH_DNAME CHECK
(DNAME IN (‘HR’, ‘IT’, ‘FINANCE’));
```

```
CREATE TABLE reservation (
    during tsrange,
    EXCLUDE USING GIST
                          (during WITH &&)
);
```

```
CREATE TABLE reservation (
    figure circle,
    EXCLUDE USING GIST (figure WITH &&)
);
```

$$Key\{ < attribute\ name\ commalist > \}$$

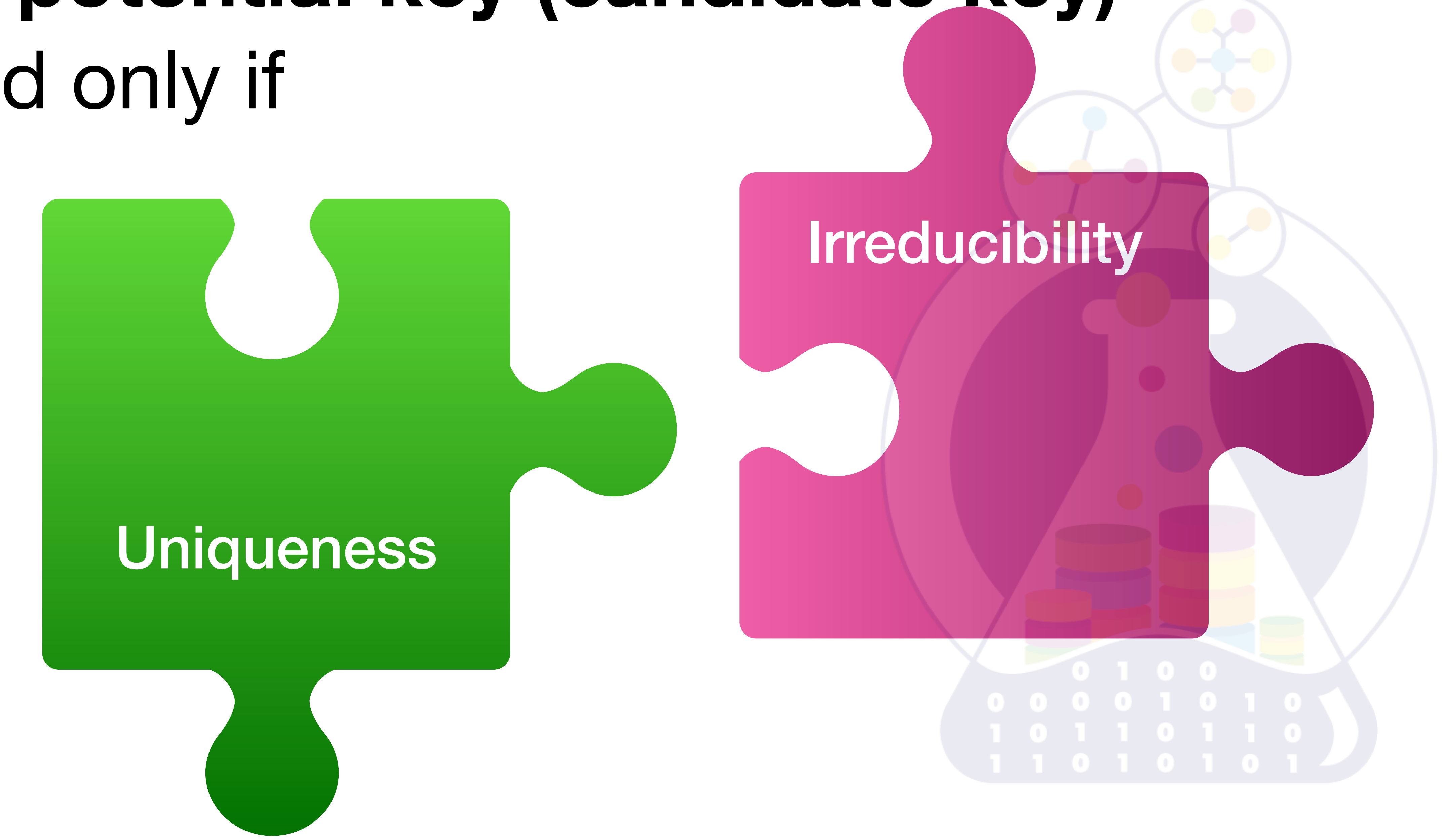
$$\begin{aligned}
&VAR\ rel\ BASE\ R\\
&\quad \{\ A\ INTEGER,\\
&\quad\ \ B\ INTEGER,\\
&\quad\ \ C\ STRING\ \}\\
&\quad\ \ KEY\ \{A, B\};
\end{aligned}$$

A : 🔑 integer	B : 🔑 integer	C : string
1	1	string #1
1	2	string #1
3	2	string #3

$$R(A_1, A_2, \ldots, A_n)$$

$$K = \{A_1, A_2, \ldots, A_m\}, m \leq n$$

K is **potential key (candidate key)** if and only if



A	B	C
1	1	2
1	2	3
3	2	4

$$K_1 = \{C\}$$

$$K_2 = \{A, B\}$$

candidates

not candidates

$$K_3 = \{A, C\}$$

$$K_4 = \{B, C\}$$

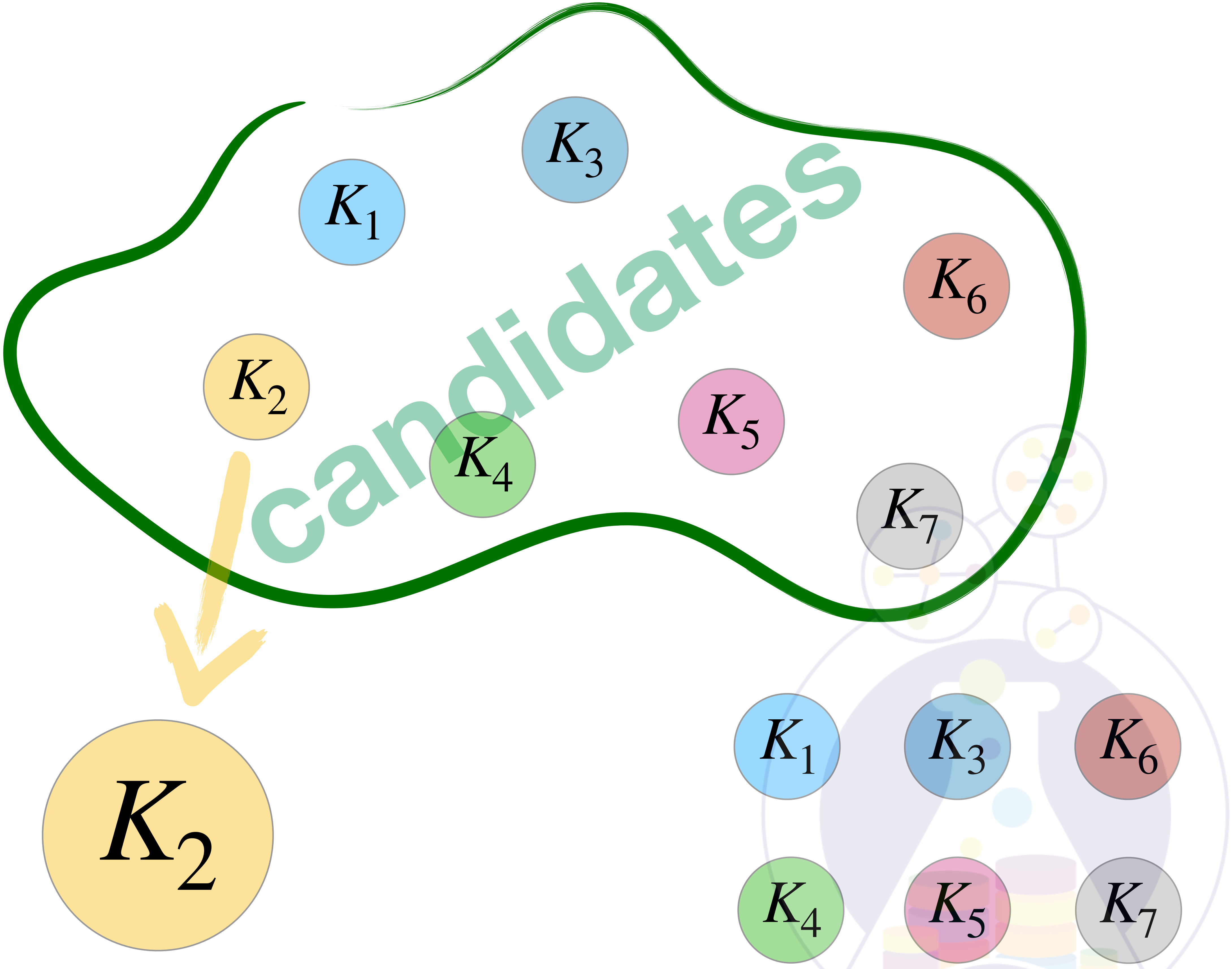
$$K_5 = \{A, B, C\}$$

K is **simple potential key (simple candidate key)** if

$K = \underline{\{A_j\}}$ has only one attribute

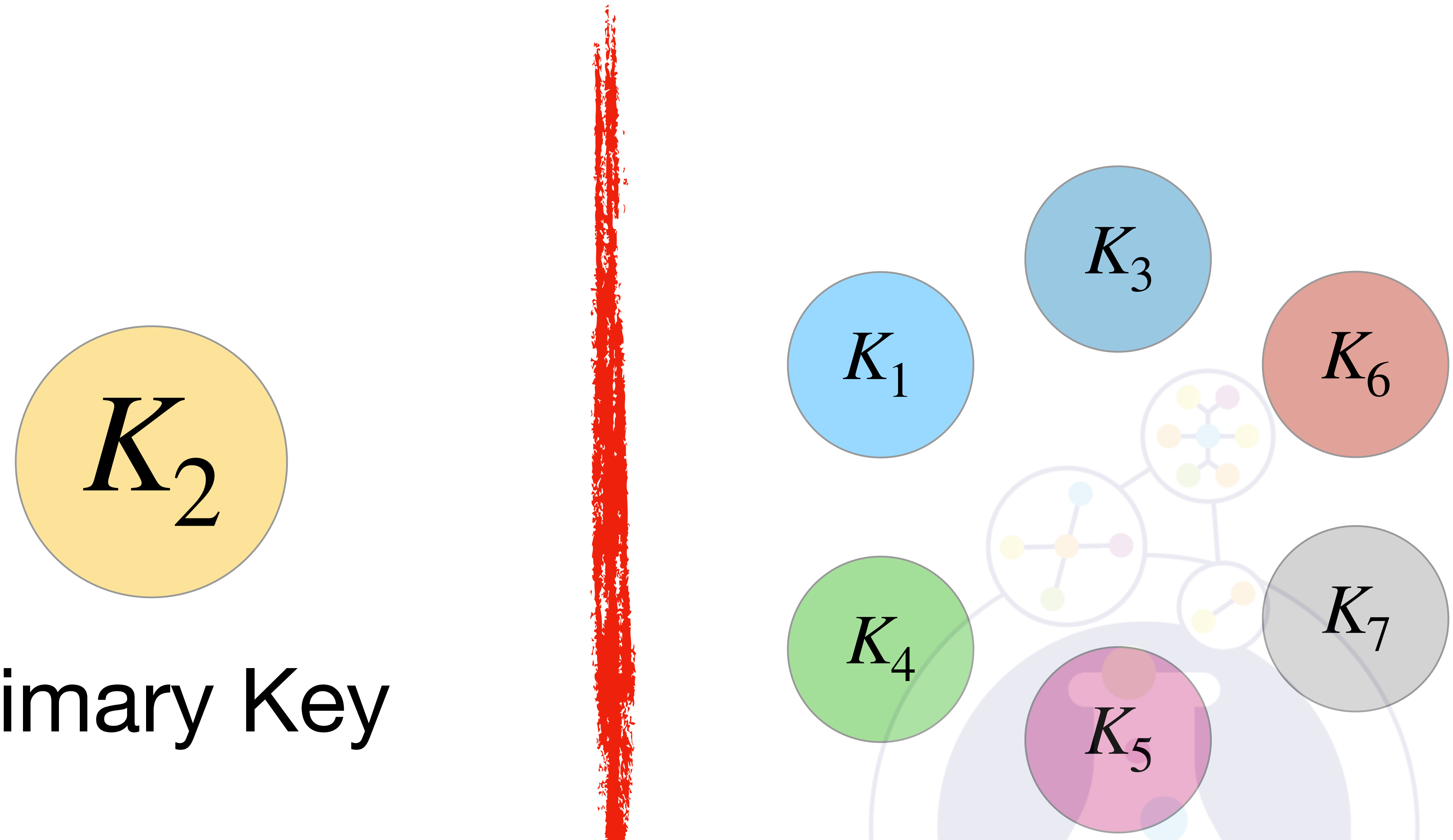
K is **compound potential key (compound candidate key)** if

$K = \underline{\{A_1, A_2, \ldots\}}$ has more then one



Primary Key

Alternative Keys



Primary Key

Unique Keys

Azat Yakupov

A	B	C
1	1	2
1	2	3
3	2	4

$$\left.\begin{aligned} K_1 &= \{C\} \\ K_2 &= \{A, B\} \end{aligned}\right] \textbf{candidates}$$

Which is a **Primary Key**

(from your point of view)

```
CREATE TABLE Employee
(
  ID NUMBER NOT NULL,
  SALARY DECIMAL(9,2)  NOT NULL
            CONSTRAINT CH_SAL CHECK (SALARY>=100000),
  DNAME VARCHAR(10)
            CONSTRAINT CH_DNAME
                            CHECK (DNAME IN (‘HR’, ‘IT’)),
  BONUS DECIMAL(9,2) DEFAULT 0 NOT NULL,
  INN VARCHAR(16),
  CONSTRAINT employee_pk PRIMARY KEY (ID),
  CONSTRAINT employee_uk UNIQUE (INN)
);
```

```
ALTER TABLE Employee ADD CONSTRAINT
employee_pk PRIMARY KEY (ID);
```

```
ALTER TABLE Employee ADD CONSTRAINT
employee_uk UNIQUE (INN);
```

$$R_2 = \{A_1, A_2, \ldots, A_n\}$$

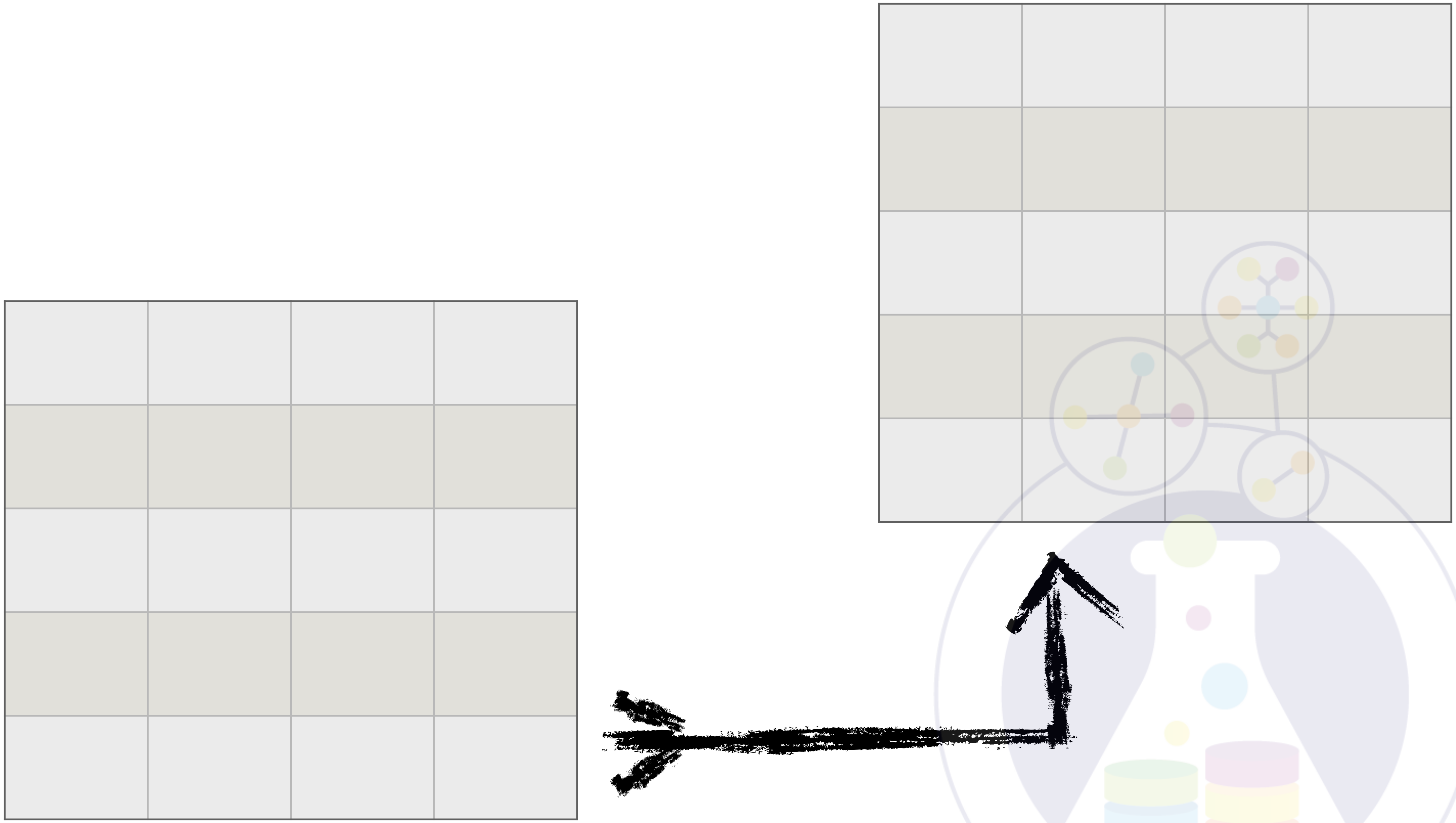
then a **Foreign Key** $FK = \{A_1, A_2, \ldots, A_m\}, m \leq n$

satisfies next rules

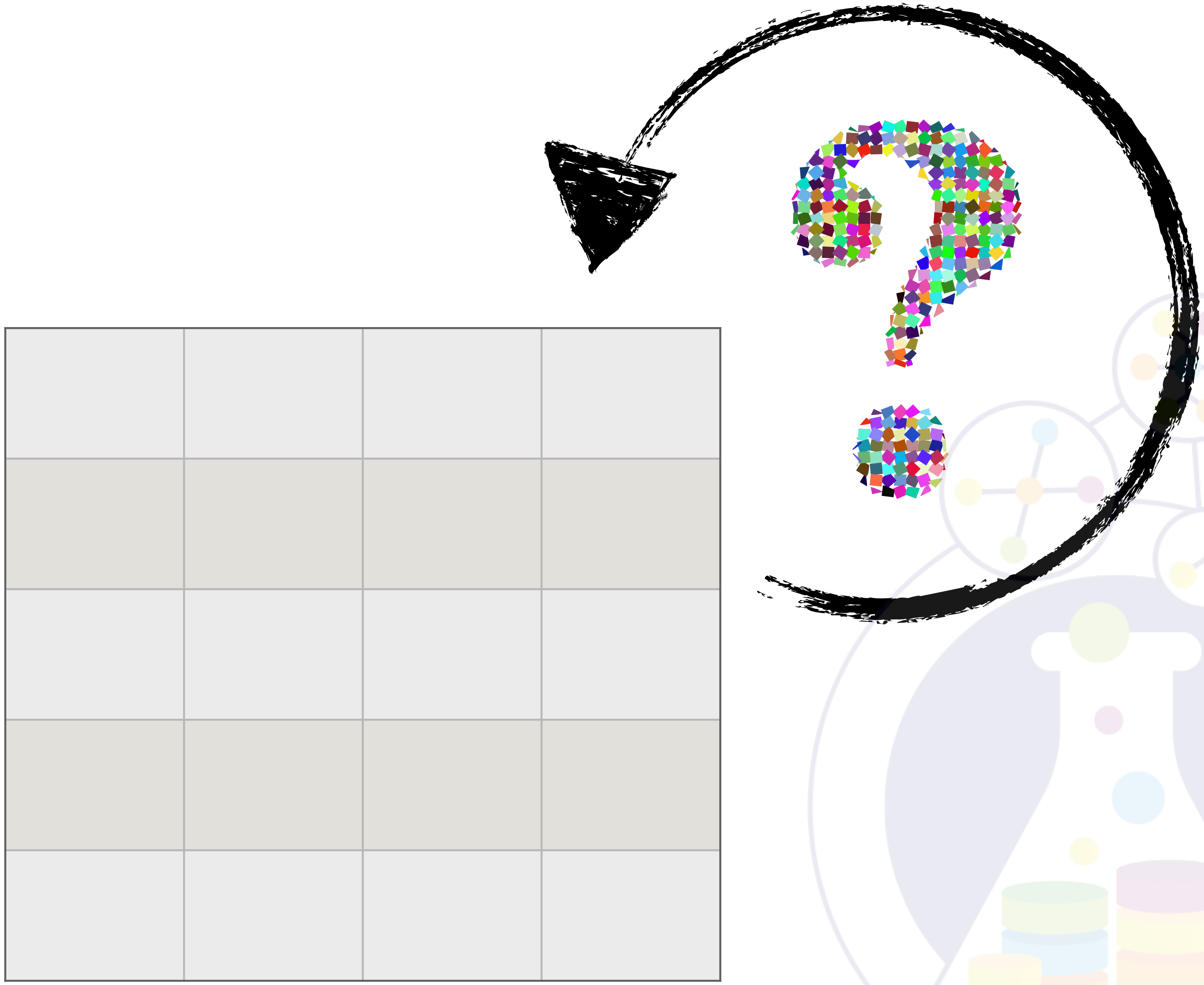
- $\exists\ R_1$ with candidate key CK. Possible $R_1 = R_2$
- $\exists\ FK^{'} \subseteq FK \Rightarrow FK^{'} = CK$
- $\forall\ value_1 \in\ FK \subseteq R_2\ \exists\ value_2 \in\ FK^{'} \subseteq FK \Rightarrow$

$$value_2 = value_3 \in CK \subseteq R_1$$

$$R_1 \neq R_2$$



$$R_1 = R_2$$



Foreign Key is **simple** when corresponding candidate key is **simple**

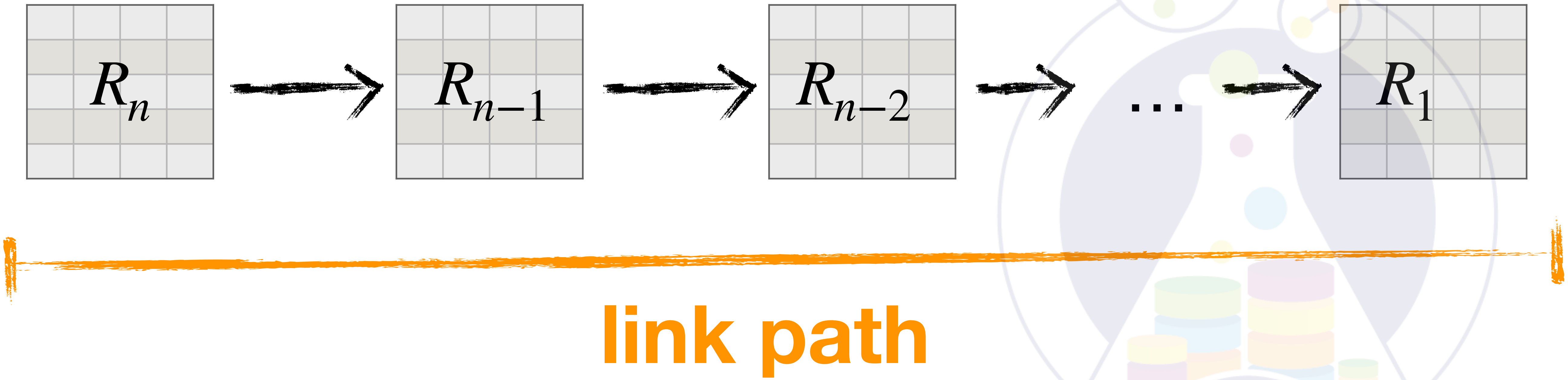
Other words a Foreign Key is based on only **one attribute**

Foreign Key is **compound** when corresponding candidate key is **compound**

Other words a Foreign Key is based on **several attributes**

The **link** is a relationship between $R_1 \to R_2$

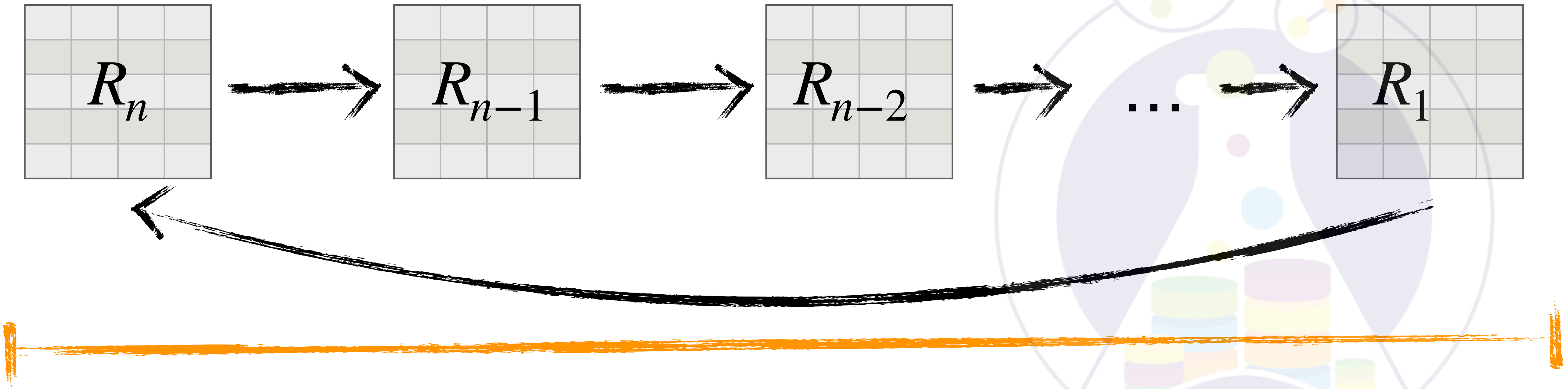
$$FOREIGN\ KEY\{ < item\ commalist > \}$$
$$REFERENCES\ < relvar\ name >$$



The **link** is a relationship between $R_1 \to R_2$

$$FOREIGN\ KEY\{ < item\ commalist > \}$$

$$REFERENCES\ < relvar\ name >$$



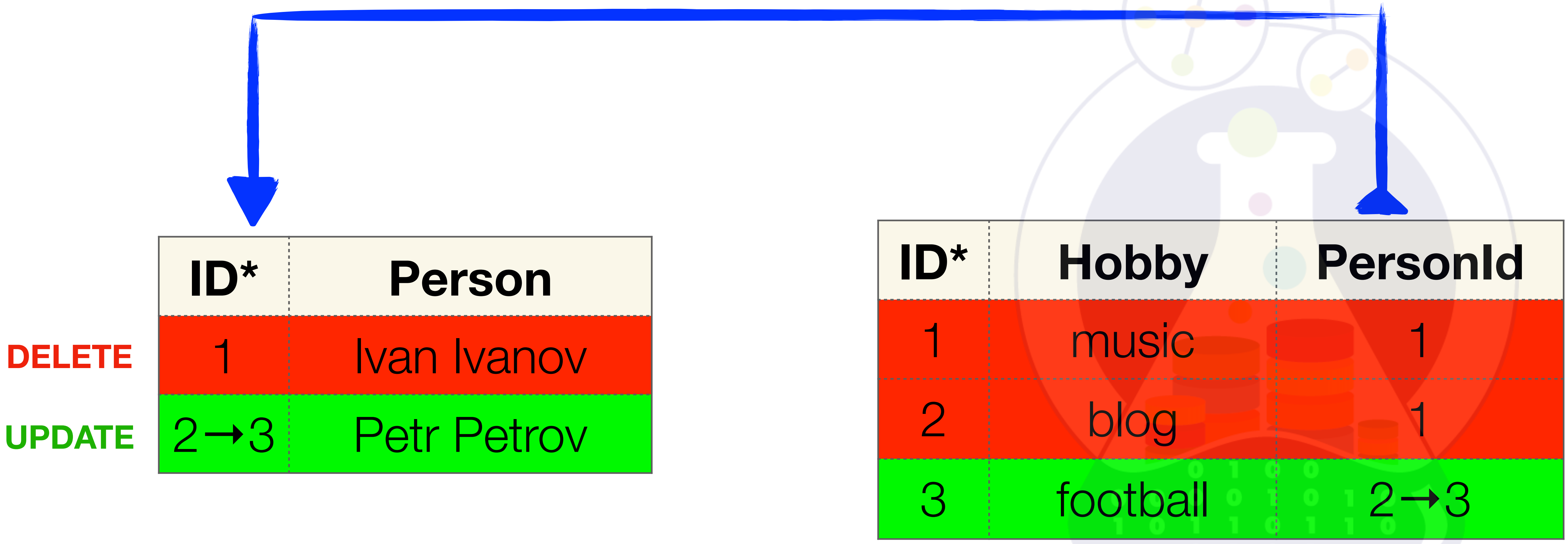
link cycle

CASCADE option

VAR rel BASE R{ . . . } . . .

FOREIGN KEY{ . . . } REFERENCES S

ON DELETE | UPDATE CASCADE;



	ID*	Person
DELETE	1	Ivan Ivanov
UPDATE	2→3	Petr Petrov

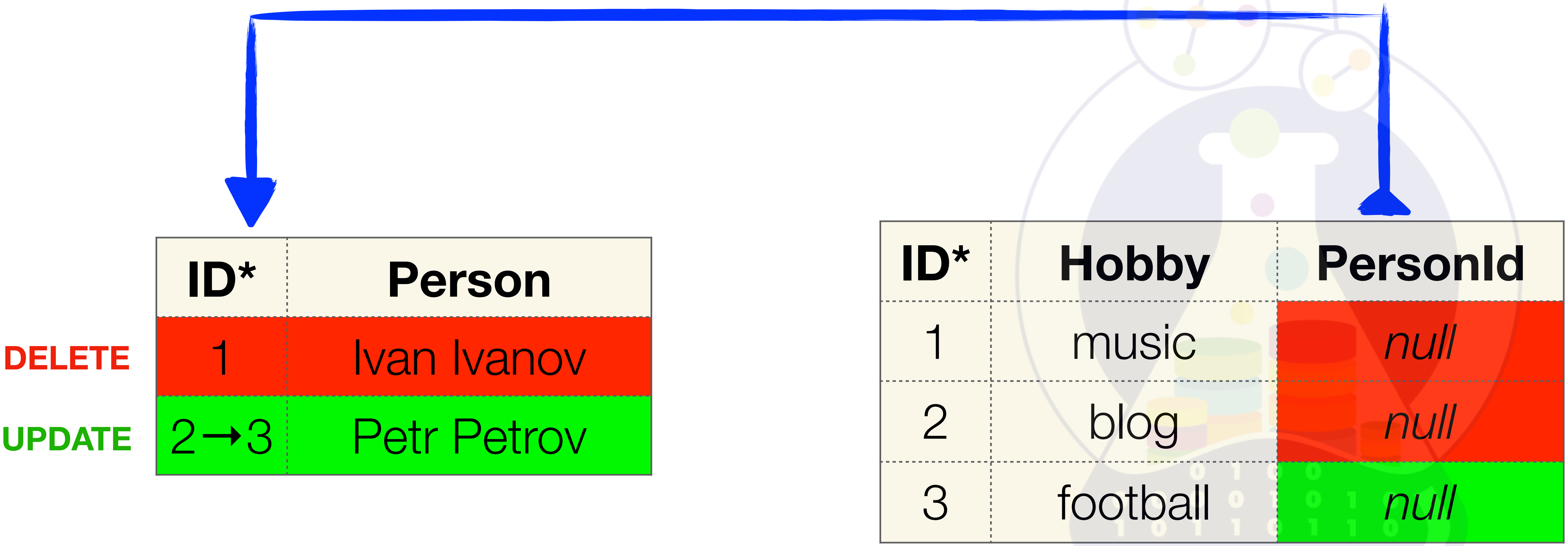
ID*	Hobby	PersonId
1	music	1
2	blog	1
3	football	2→3

SET NULL option

VAR rel BASE R{ . . . } . . .

FOREIGN KEY{ . . . } REFERENCES S

ON DELETE | UPDATE SET NULL;



	ID*	Person
DELETE	1	Ivan Ivanov
UPDATE	2→3	Petr Petrov

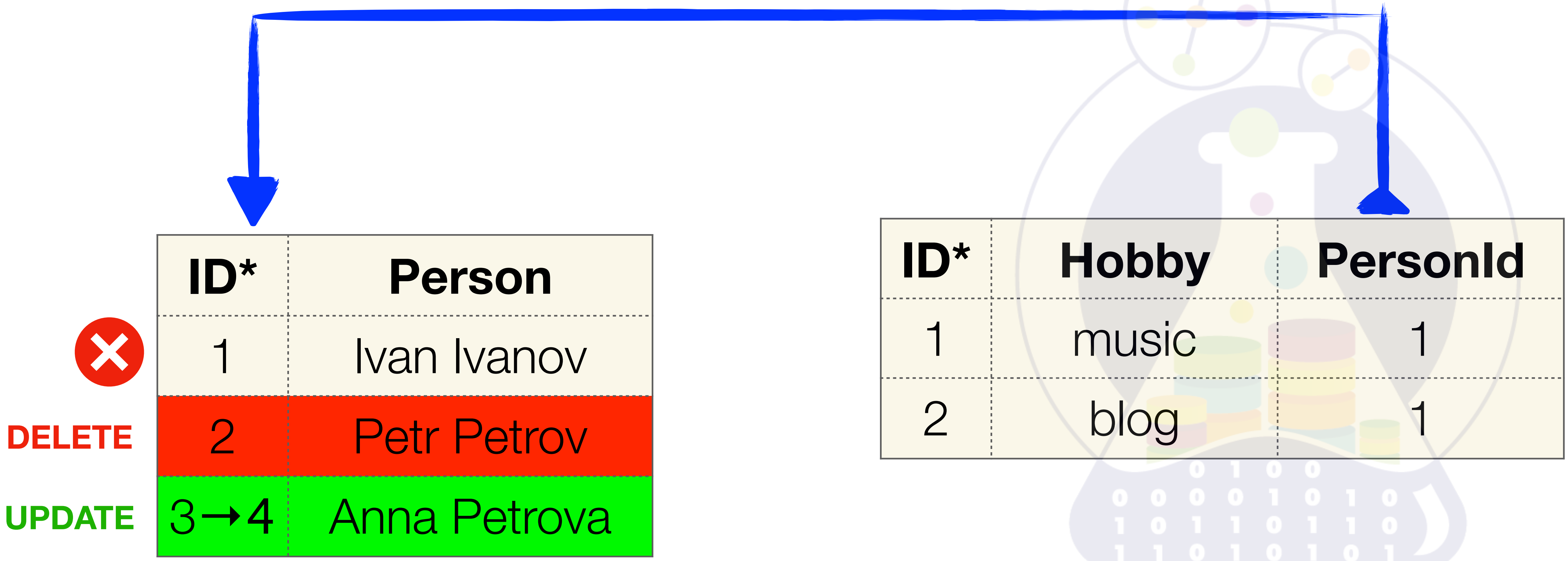
ID*	Hobby	PersonId
1	music	*null*
2	blog	*null*
3	football	*null*

RESTRICT option

VAR rel BASE R{ . . . } . . .

FOREIGN KEY{ . . . } REFERENCES S

ON DELETE | UPDATE RESTRICT;



	ID*	Person
✖	1	Ivan Ivanov
DELETE	2	Petr Petrov
UPDATE	3→4	Anna Petrova

ID*	Hobby	PersonId
1	music	1
2	blog	1

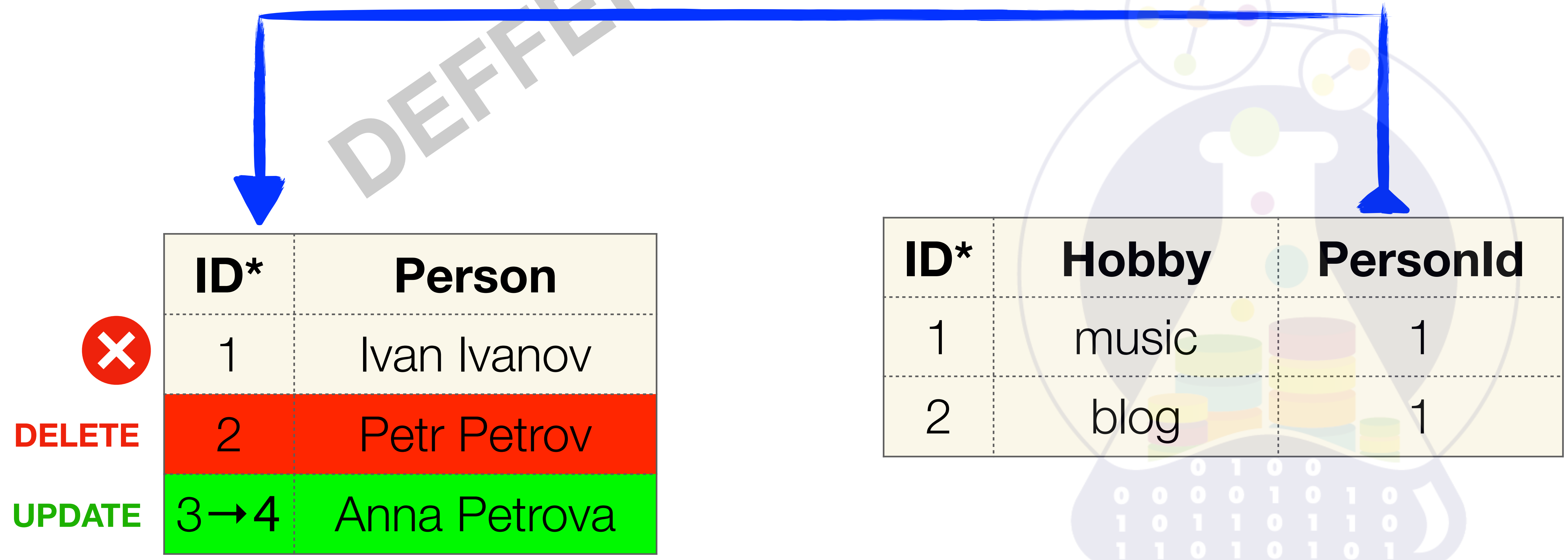
NO ACTION option

VAR rel BASE R{ . . . } . . .

FOREIGN KEY{ . . . } REFERENCES S

ON DELETE | UPDATE NO ACTION;

DEFFERED CONSTRAINT



	ID*	Person
✖	1	Ivan Ivanov
DELETE	2	Petr Petrov
UPDATE	3→4	Anna Petrova

ID*	Hobby	PersonId
1	music	1
2	blog	1

```
CREATE TABLE Employee
(
 ID NUMBER,
 INN VARCHAR2(16),
 CONSTRAINT employee_pk PRIMARY KEY (ID),
 CONSTRAINT employee_uk UNIQUE (INN)
);

CREATE TABLE Task
(
 ID NUMBER,
 EMPLOYEE_ID NUMBER,
 TASK_NAME VARCHAR2(100),
 CONSTRAINT employee_fk FOREIGN KEY (EMPLOYEE_ID)
                          REFERENCES Employee (ID)
);
```

```
CREATE TABLE Task
(
  ID NUMBER,
  EMPLOYEE_ID NUMBER,
  TASK_NAME VARCHAR2(100),
  CONSTRAINT employee_fk FOREIGN KEY (EMPLOYEE_ID)
                                                  REFERENCES Employee (ID)
            ON DELETE CASCADE ON UPDATE RESTRICT
);
```

ORACLE

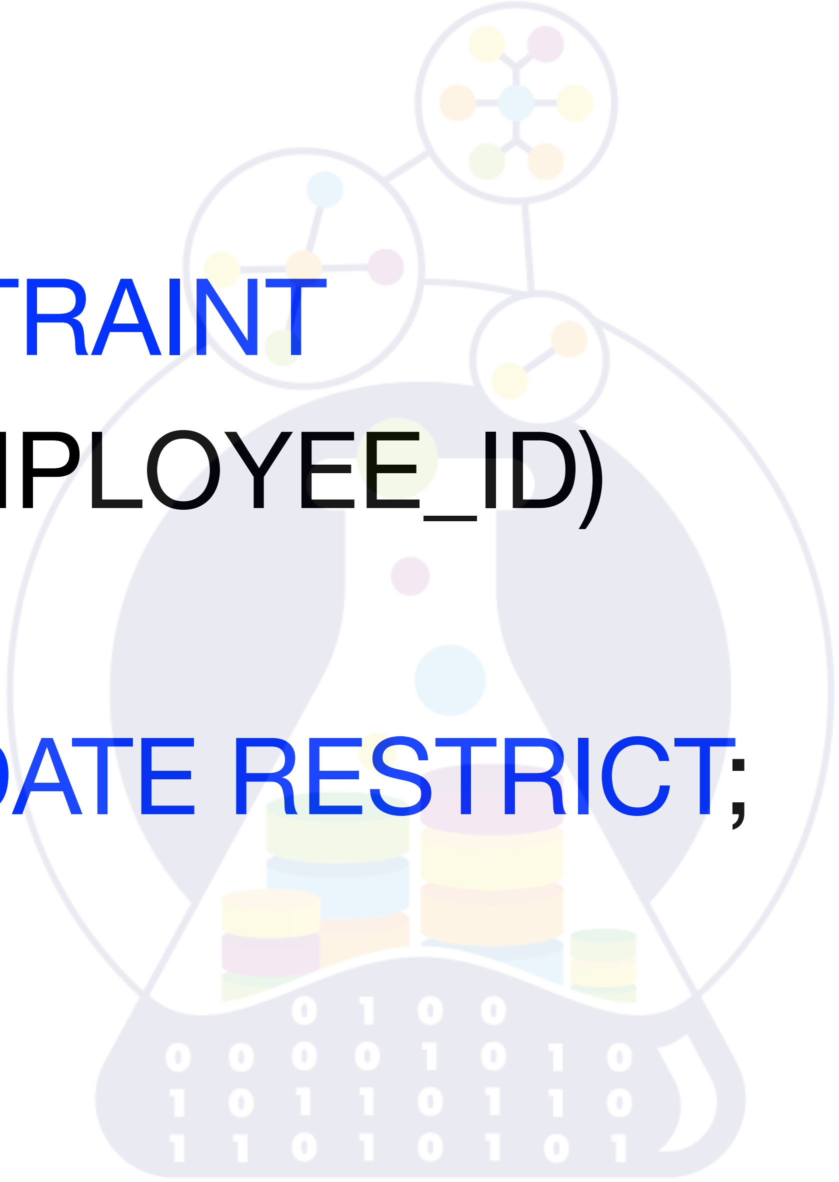
```
ALTER TABLE Task ADD CONSTRAINT
employee_fk  FOREIGN KEY (EMPLOYEE_ID)
REFERENCES Employee (ID)
ON DELETE CASCADE
ON UPDATE RESTRICT DISABLE;
```

?

```
ALTER TABLE Task ENABLE
CONSTRAINT employee_fk;
```

?

```
ALTER TABLE Task ADD CONSTRAINT
employee_fk FOREIGN KEY (EMPLOYEE_ID)
REFERENCES Employee (ID)
ON DELETE CASCADE ON UPDATE RESTRICT;
```



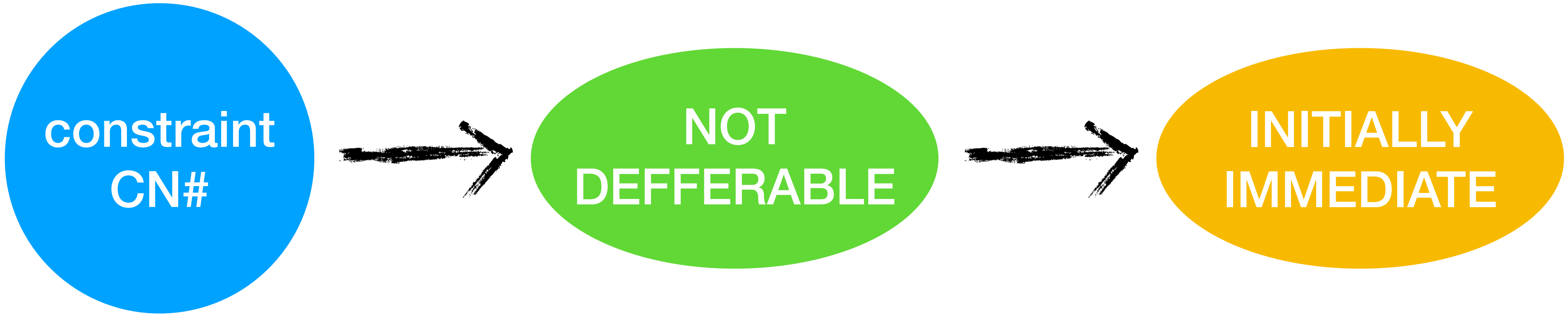
?

```
ALTER TABLE Task ADD CONSTRAINT
employee_fk FOREIGN KEY (EMPLOYEE_ID)
REFERENCES Employee (ID)
ON DELETE CASCADE ON UPDATE RESTRICT;
```

?

✓

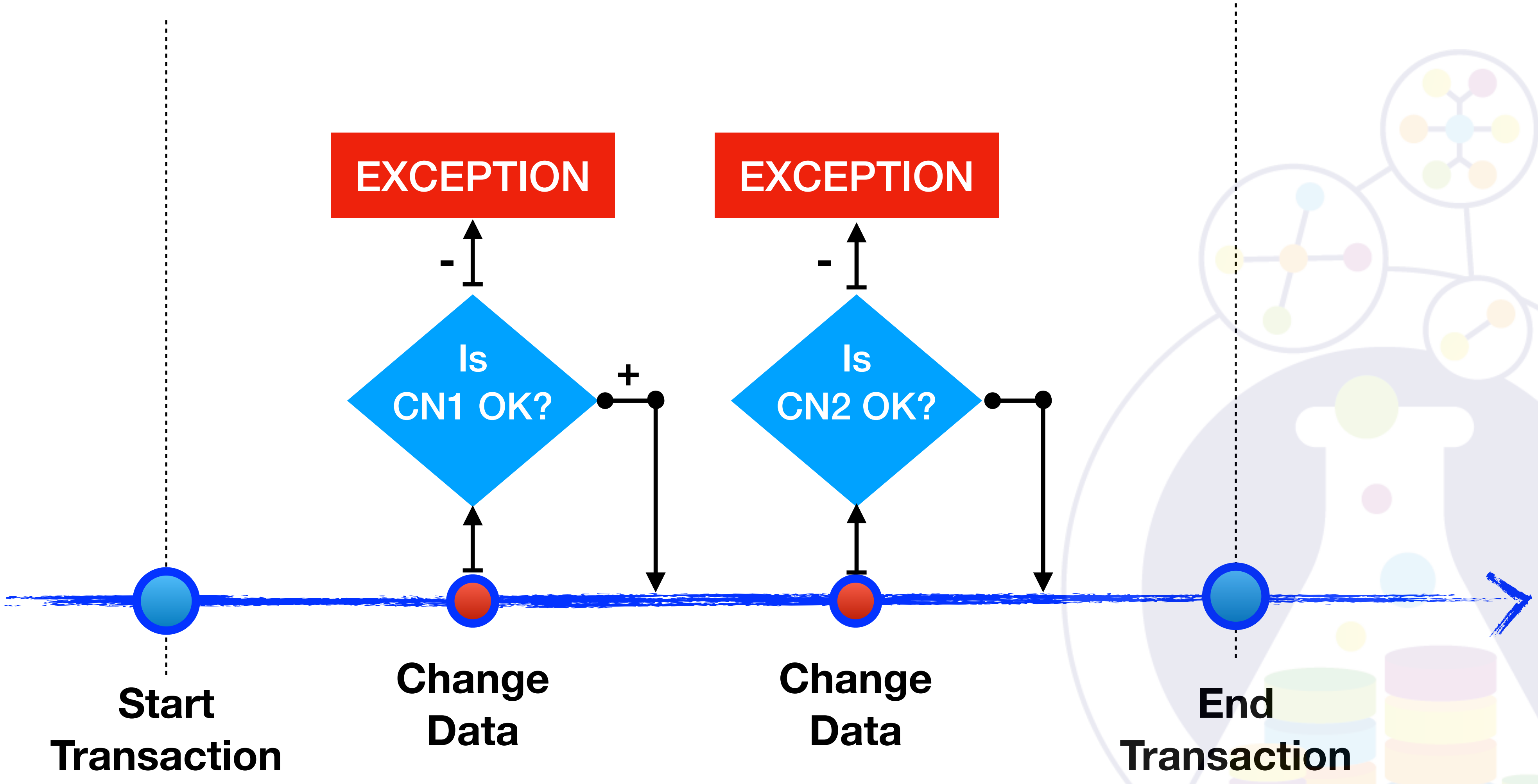
```
SELECT EMPLOYEE_ID, ID
   FROM Task AS t
WHERE NOT EXISTS
              (SELECT 1
                   FROM Employee AS e
                 WHERE e.ID = t.EMPLOYEE_ID
                     LIMIT 1);
```



constraint CN#

NOT DEFFERABLE

INITIALLY IMMEDIATE



EXCEPTION

EXCEPTION

-

-

Is CN1 OK?

+

Is CN2 OK?

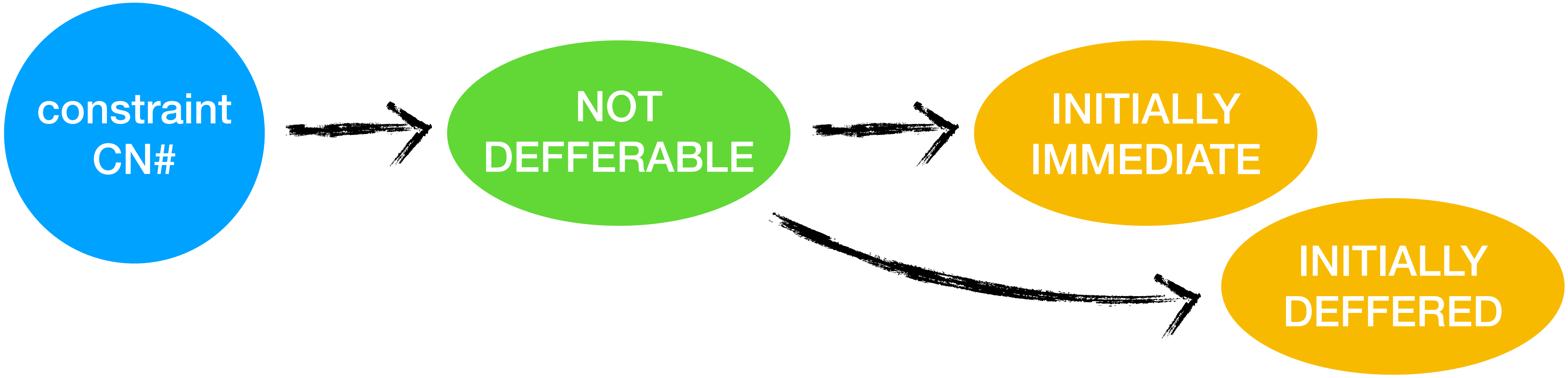
Start Transaction

Change Data

Change Data

End Transaction

Azat Yakupov

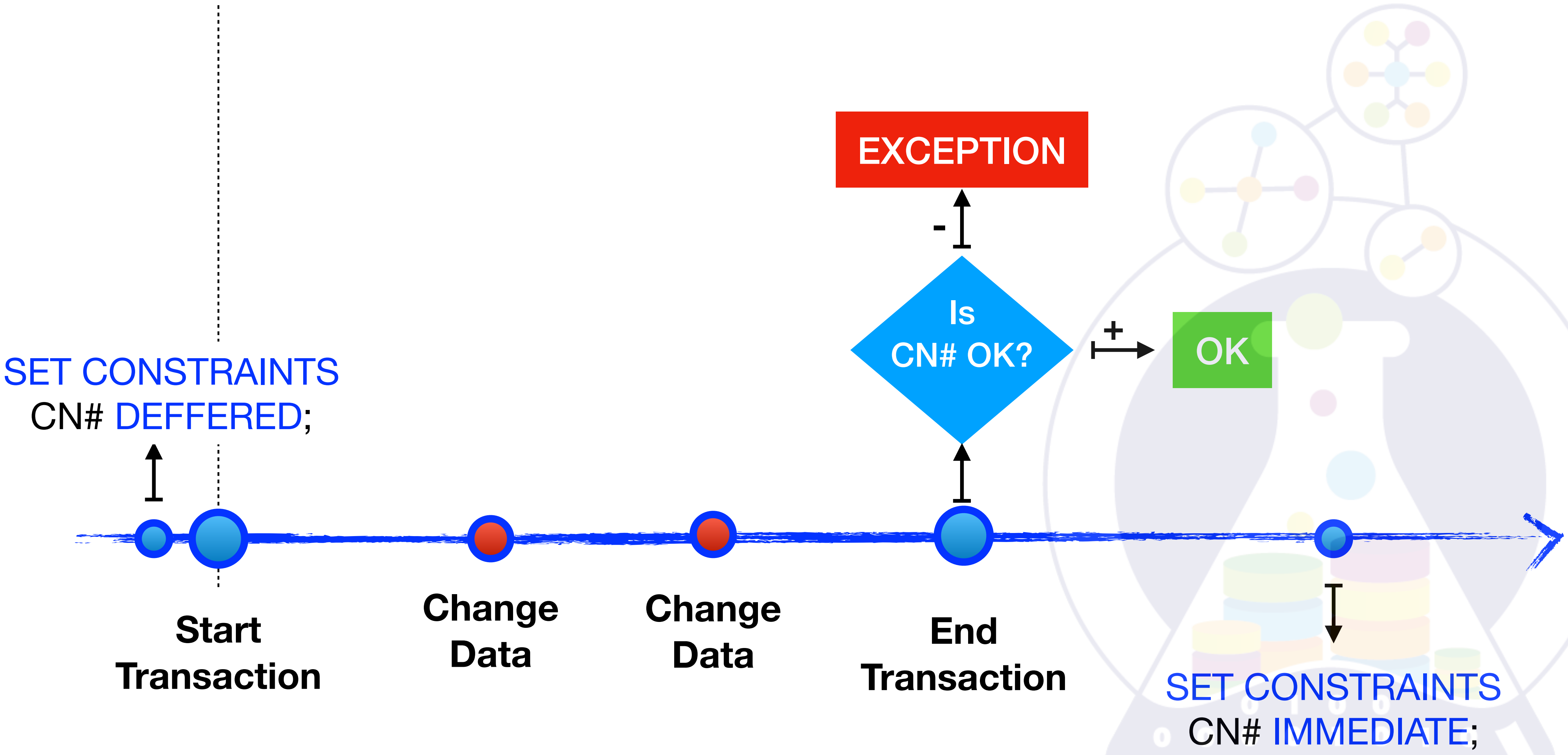


constraint CN#

NOT DEFFERABLE

INITIALLY IMMEDIATE

INITIALLY DEFFERED



EXCEPTION

-

Is CN# OK?

+

OK

SET CONSTRAINTS CN# DEFFERED;

Start Transaction

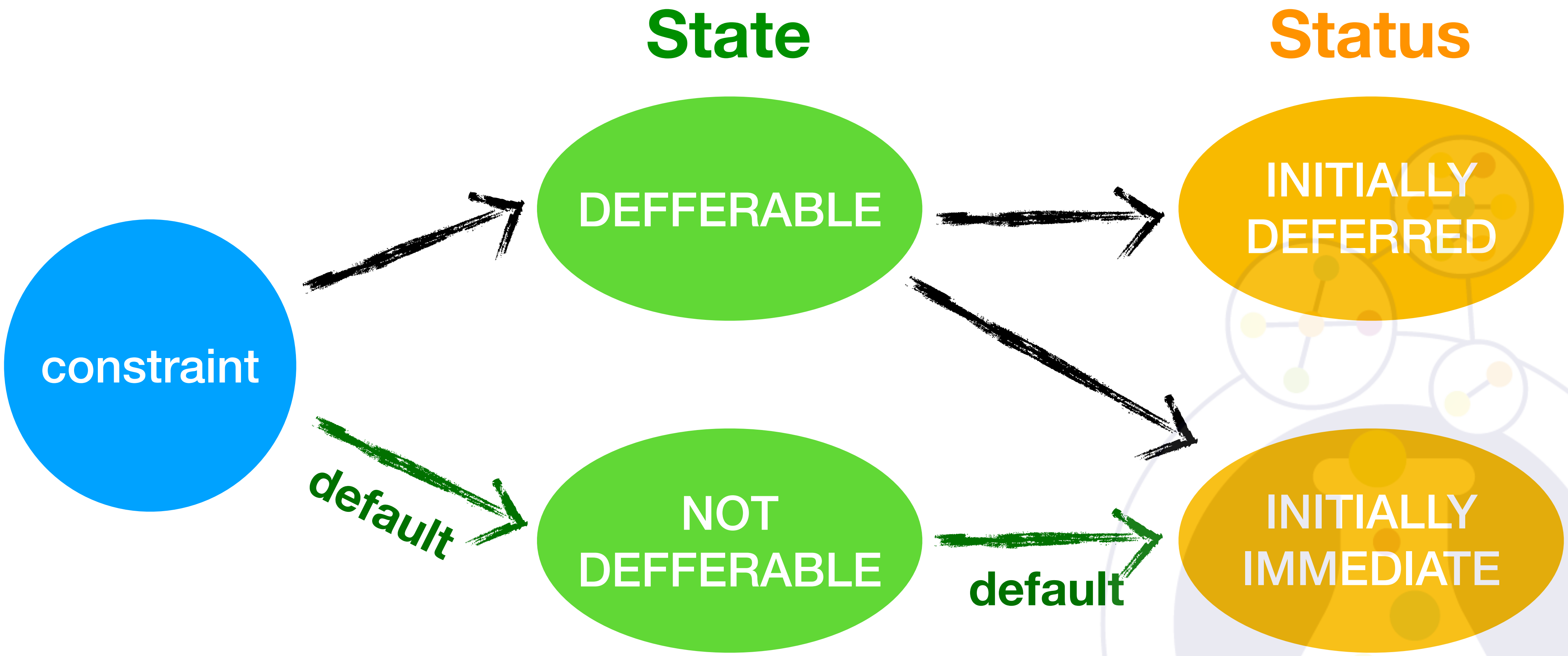
Change Data

Change Data

End Transaction

SET CONSTRAINTS CN# IMMEDIATE;

Azat Yakupov



State

Status

constraint

DEFFERABLE

NOT DEFFERABLE

default

INITIALLY DEFERRED

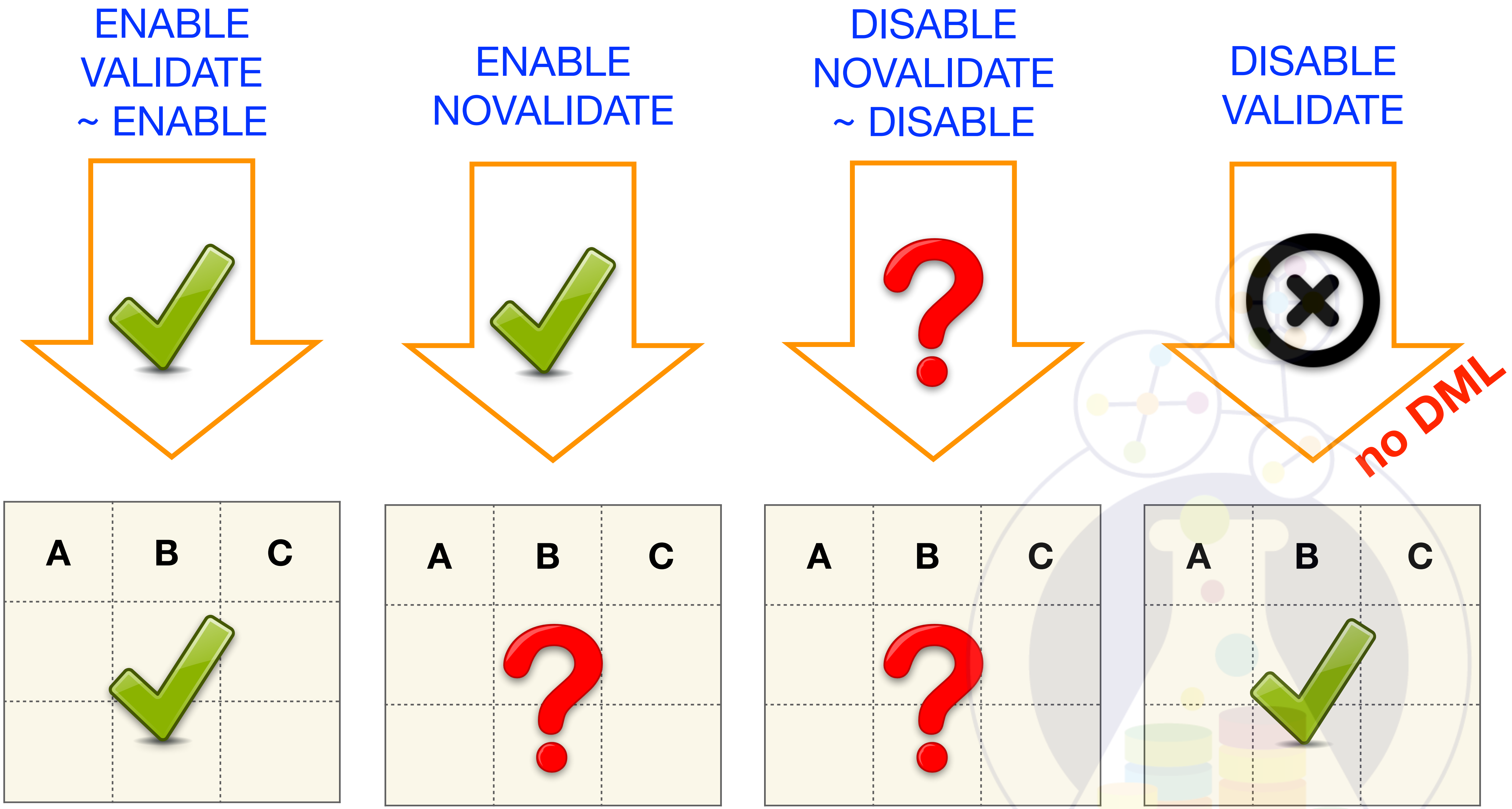
INITIALLY IMMEDIATE

default

```
ALTER TABLE Task ADD CONSTRAINT
employee_fk FOREIGN KEY (EMPLOYEE_ID)
REFERENCES Employee (ID)
ON DELETE CASCADE ON UPDATE RESTRICT
DEFERRABLE INITIALLY DEFERRED;
```

Transaction

```
SET CONSTRAINT employee_fk DEFERRED;
UPDATE Employee SET ID = 100 WHERE ID = 1;
UPDATE Task SET EMPLOYEE_ID = 100
              WHERE EMPLOYEE_ID=1;
COMMIT;
SET CONSTRAINT employee_fk IMMEDIATE;
```



ENABLE VALIDATE ~ ENABLE

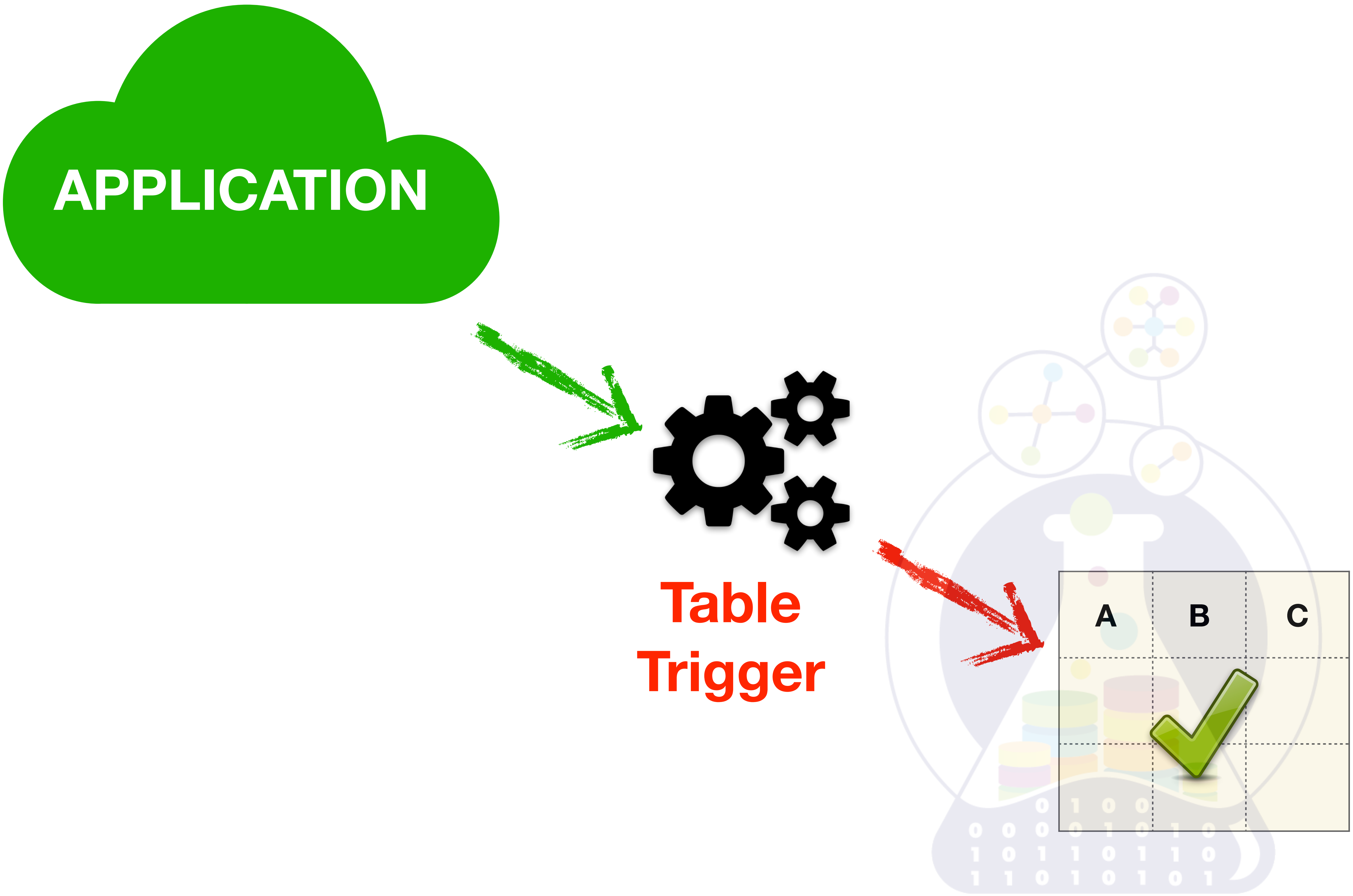
ENABLE NOVALIDATE

DISABLE NOVALIDATE ~ DISABLE

DISABLE VALIDATE

```
ALTER TABLE Task ADD CONSTRAINT employee_fk
FOREIGN KEY (EMPLOYEE_ID) REFERENCES Employee (ID)
ON DELETE CASCADE ON UPDATE RESTRICT
ENABLE NOVALIDATE;

ALTER TABLE Employee ADD CONSTRAINT
employee_uk UNIQUE (INN) ENABLE NOVALIDATE;


ALTER TABLE Employee ENABLE VALIDATE
CONSTRAINT employee_uk;
```

```
ALTER TABLE Task ADD CONSTRAINT employee_fk
FOREIGN KEY (EMPLOYEE_ID) REFERENCES Employee (ID)
ON DELETE CASCADE ON UPDATE RESTRICT
NOT VALID;

ALTER TABLE Employee VALIDATE CONSTRAINT
employee_fk;
```



APPLICATION

Table Trigger

A	B	C
	✓	

View is a *virtual continuous relation* defined by

- **name** (mandatory)
- **relation's expression** (mandatory)
- **list of candidate keys** (optional)

$$VAR \ < view\_name > \ VIEW \ < relation\_exp >$$
$$< candidate\_key\_list >$$

$$
\begin{aligned}
&VAR\ rel\ BASE\ R\\
&\qquad \{\ A\ INTEGER,\\
&\qquad\ \ B\ INTEGER,\\
&\qquad\ \ C\ STRING\ \};
\end{aligned}
$$

A : integer	B : integer	C : string
1	1	string #1
1	2	string #1
3	2	string #3

$$
\begin{aligned}
&VAR\ view\_rel\ VIEW\\
&\qquad (rel\ WHERE\ A = 1)\\
&\qquad \{A, B, C\}
\end{aligned}
$$

A	B	C
1	1	string #1
1	2	string #1

$$view\_rel \ WHERE \ B = 2$$

A	B	C
1	2	string #1

$$view\_rel \ WHERE \ B = 2 \ OR \ A + B > 0$$

A	B	C
1	1	string #1
1	2	string #1

Materialized View is a *discrete relation* defined by

- **name** (mandatory)
- **relation's expression** (mandatory)
- **list of candidate keys** (optional)
- **refresh time period** (mandatory)

$$VAR \ < mv\_name > \ SNAPSHOT \ < relation\_exp >$$

$$< candidate\_key\_list >$$

$$REFRESH \ EVERY \ < period >$$

VAR rel BASE R
{ A INTEGER,
B INTEGER,
C STRING };

A : integer	B : integer	C : string
1	1	string #1
1	2	string #1
3	2	string #3

VAR mv_rel SNAPSHOT
$(rel\ WHERE\ A = 1)$
$\{A, B, C\}$

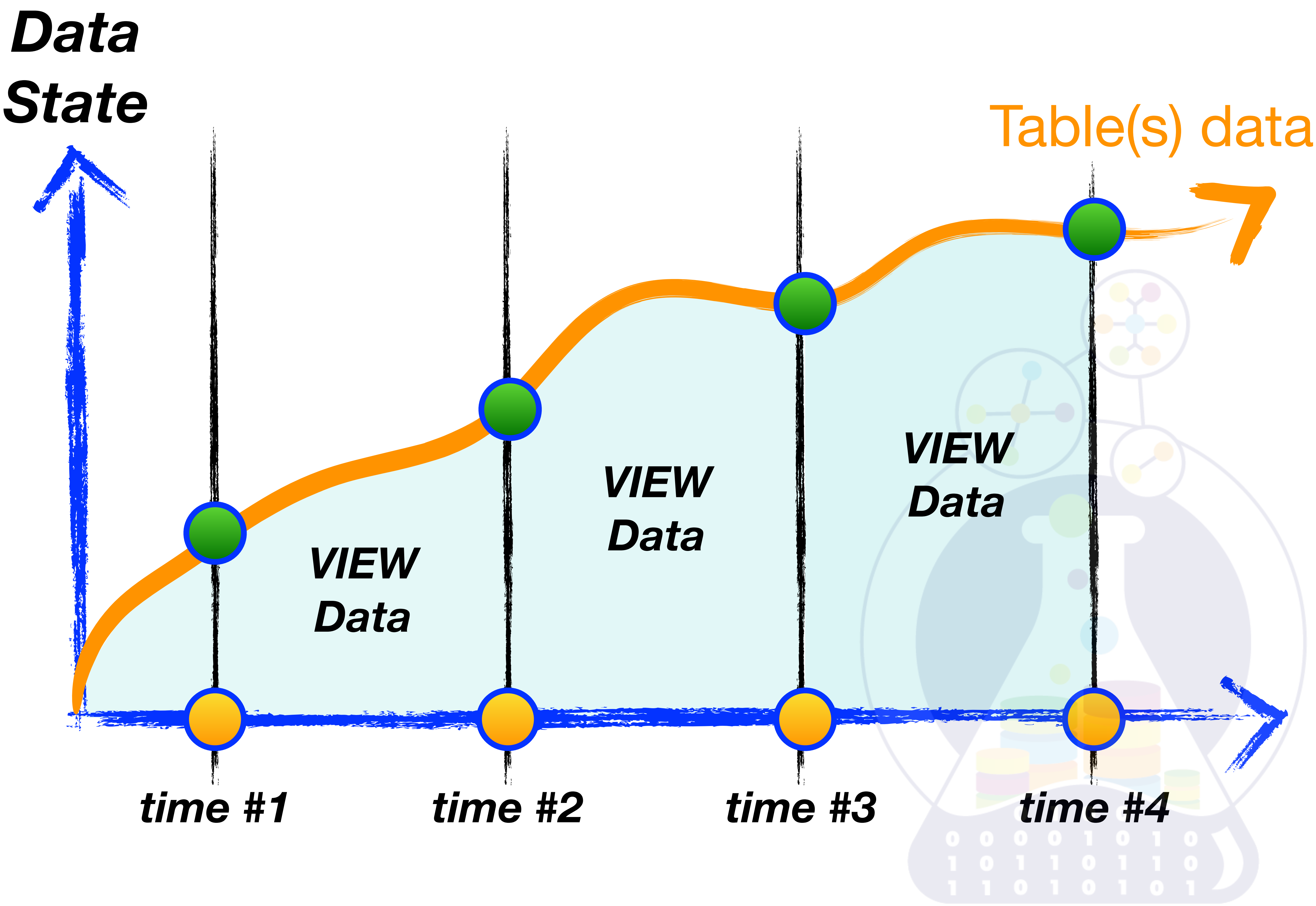
A	B	C
1	1	string #1
1	2	string #1

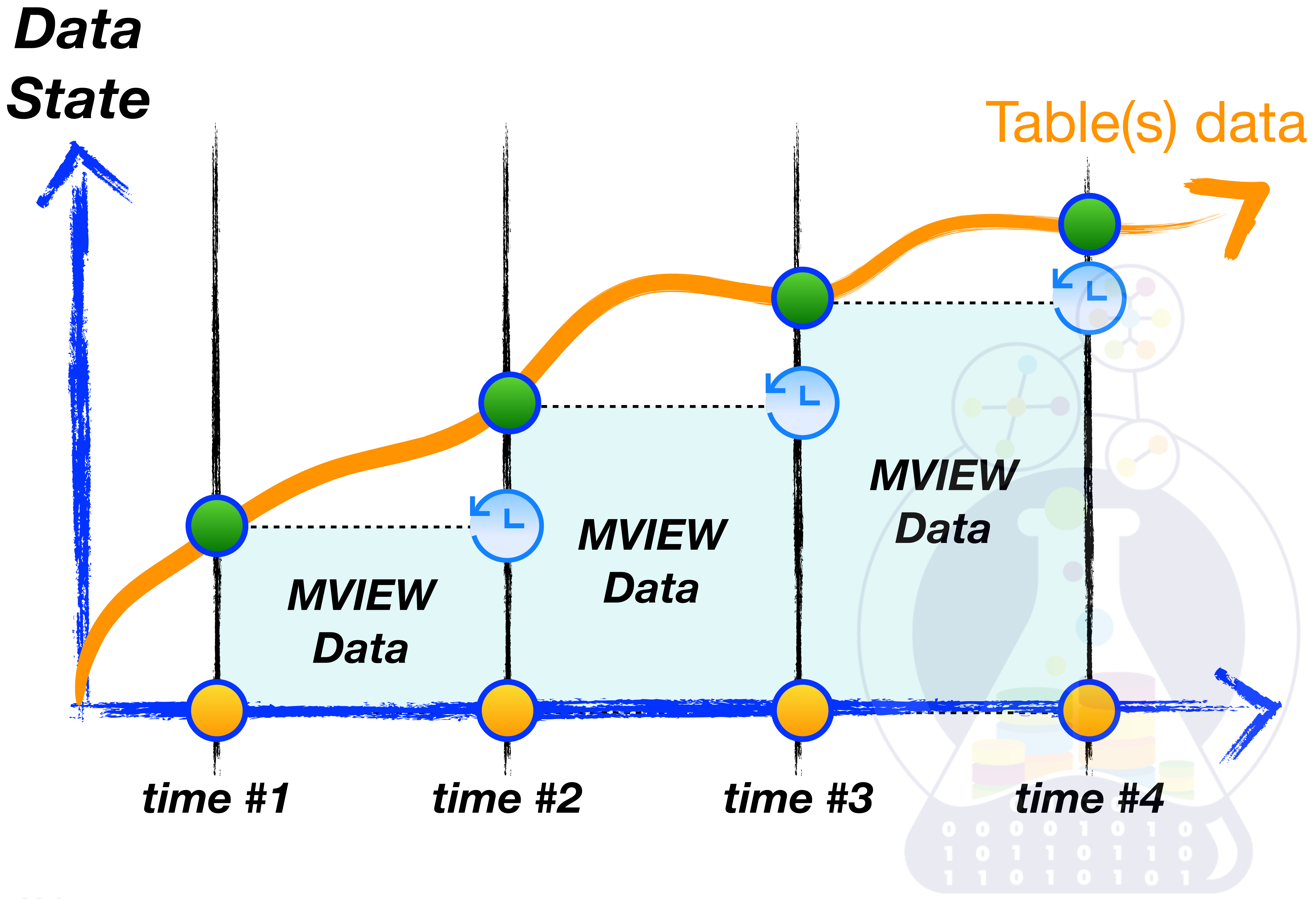
$$mv\_rel\ WHERE\ B = 2$$

A	B	C
1	2	string #1

$$mv\_rel\ WHERE\ B = 2$$
$$OR\ A + B > 0$$

A	B	C
1	1	string #1
1	2	string #1


Data
State
Table(s) data
VIEW
Data
VIEW
Data
VIEW
Data
time #1
time #2
time #3
time #4

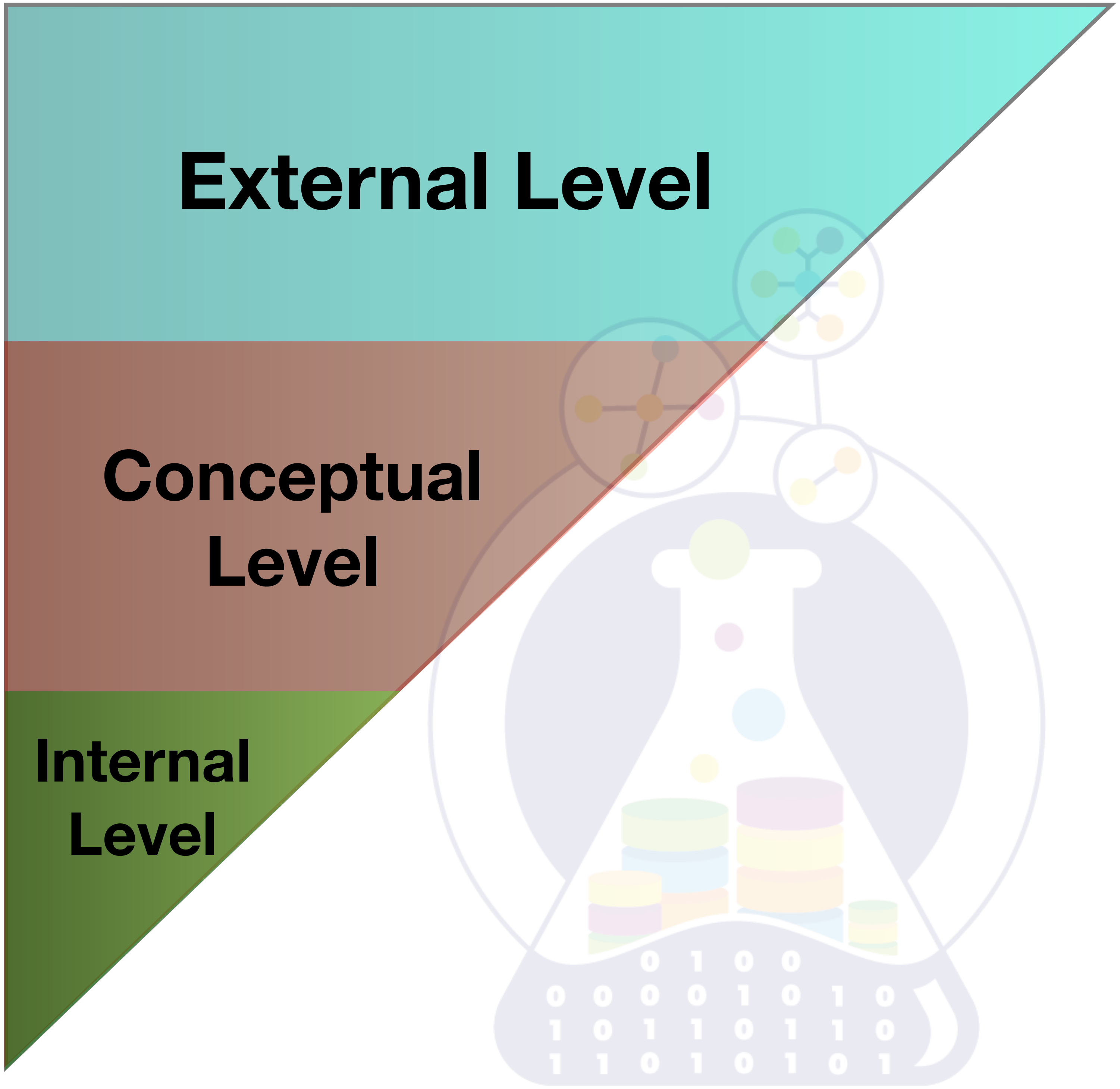

Data State
Table(s) data
MVIEW Data
MVIEW Data
MVIEW Data
time #1
time #2
time #3
time #4

ANSI/SPARK Architecture

~ user level
describes data in
database [m]views

~ logical level
describes data in
database tables

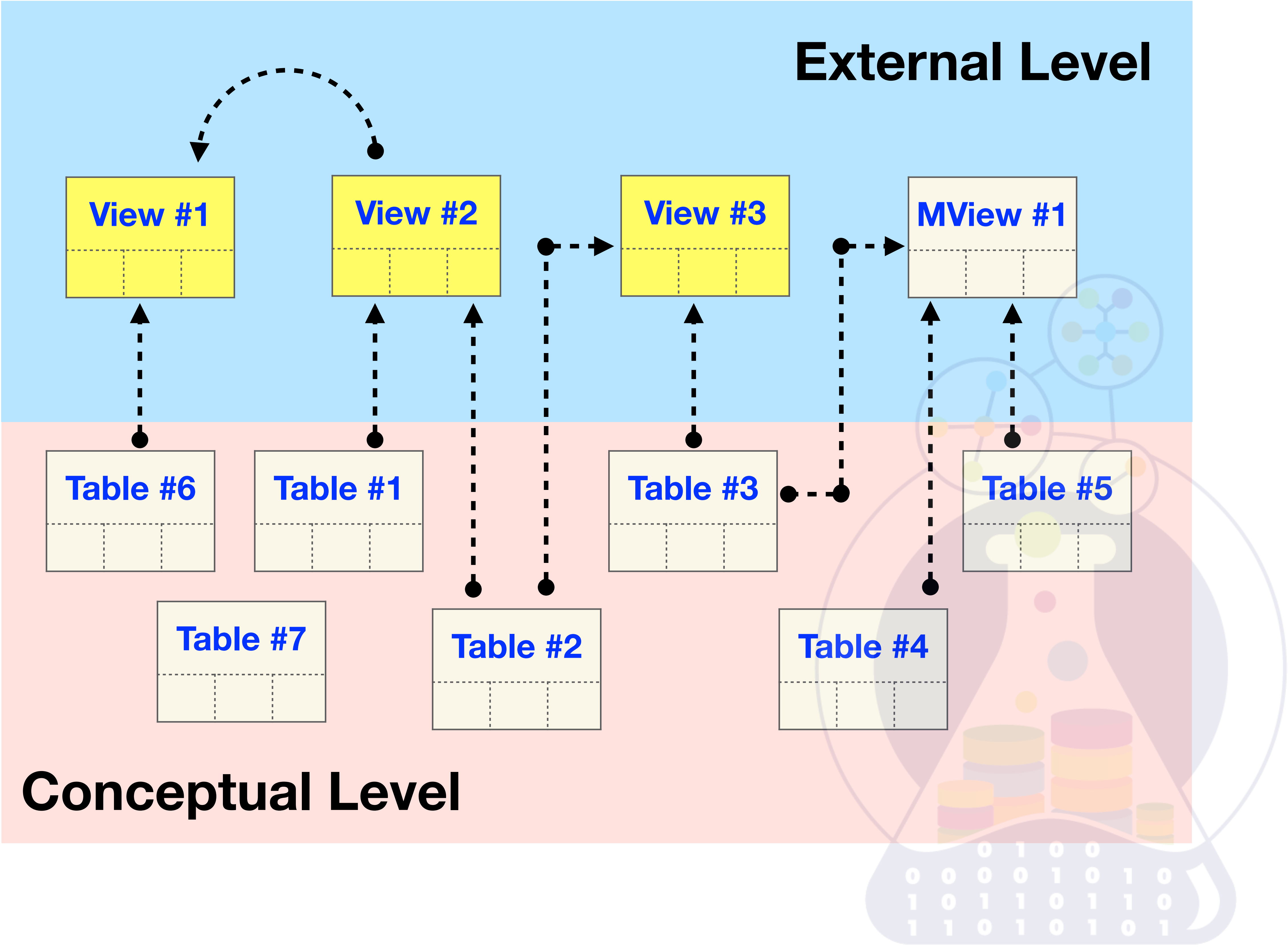
~ physical level
describes data in
database files



External Level

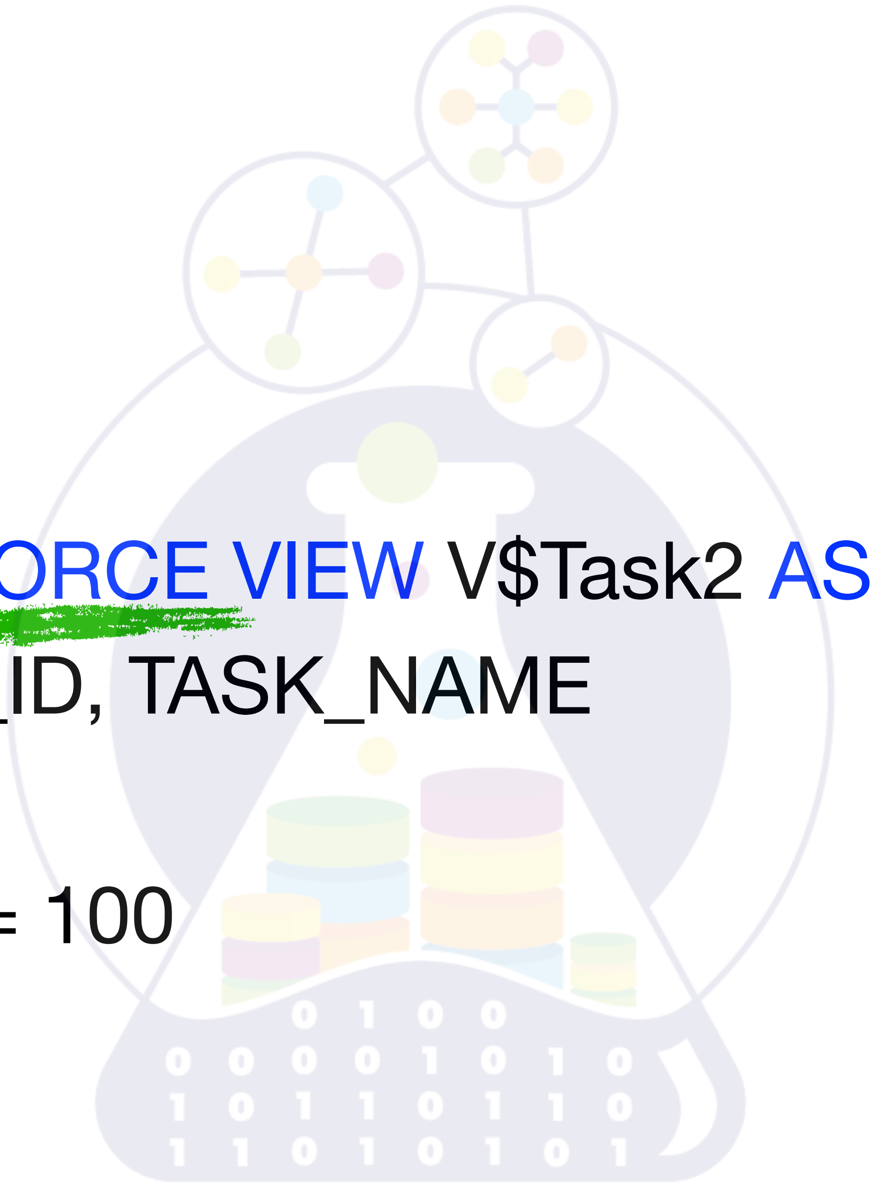
Conceptual Level

Internal Level


External Level
View #1
View #2
View #3
MView #1
Table #6
Table #1
Table #3
Table #5
Table #7
Table #2
Table #4
Conceptual Level

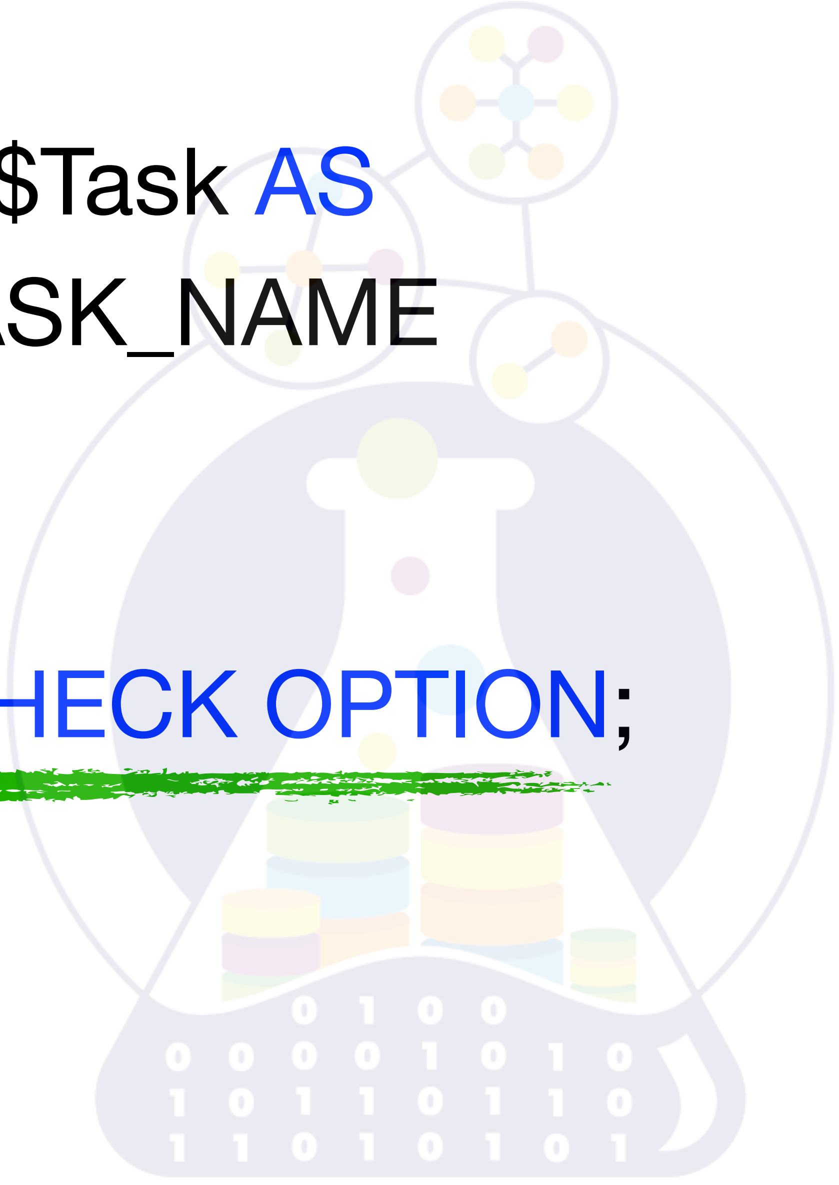
ORACLE

```
CREATE OR REPLACE FORCE VIEW V$Task1 AS
SELECT ID, EMPLOYEE_ID, TASK_NAME
  FROM Task
WHERE EMPLOYEE_ID = 100
WITH CHECK OPTION;
```

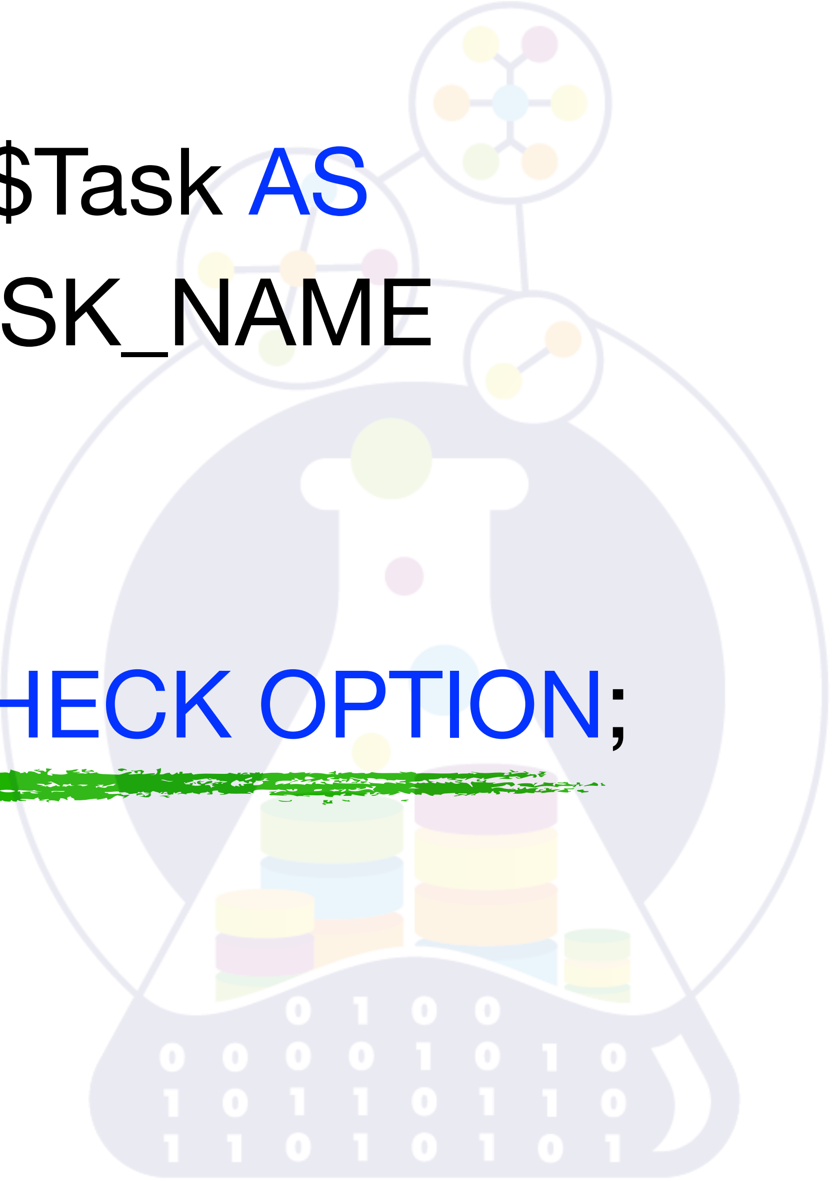


```
CREATE OR REPLACE FORCE VIEW V$Task2 AS
SELECT ID, EMPLOYEE_ID, TASK_NAME
  FROM Task
WHERE EMPLOYEE_ID = 100
WITH READ ONLY;
```

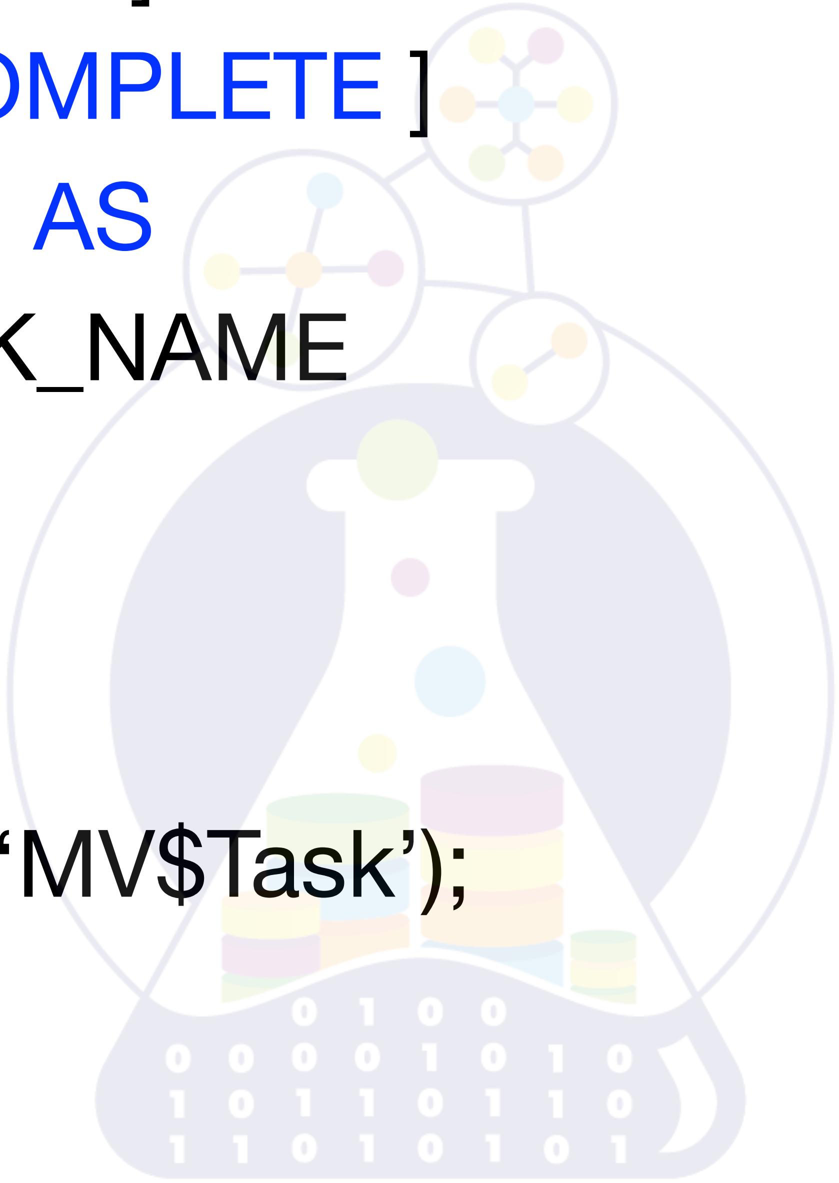
```
CREATE OR REPLACE VIEW V$Task AS
SELECT ID, EMPLOYEE_ID, TASK_NAME
   FROM Task
WHERE EMPLOYEE_ID = 100
WITH LOCAL [ CASCADED ] CHECK OPTION;
```



```
CREATE OR REPLACE VIEW V$Task AS
SELECT ID, EMPLOYEE_ID, TASK_NAME
   FROM Task
WHERE EMPLOYEE_ID = 100
WITH LOCAL [ CASCADED ] CHECK OPTION;
```

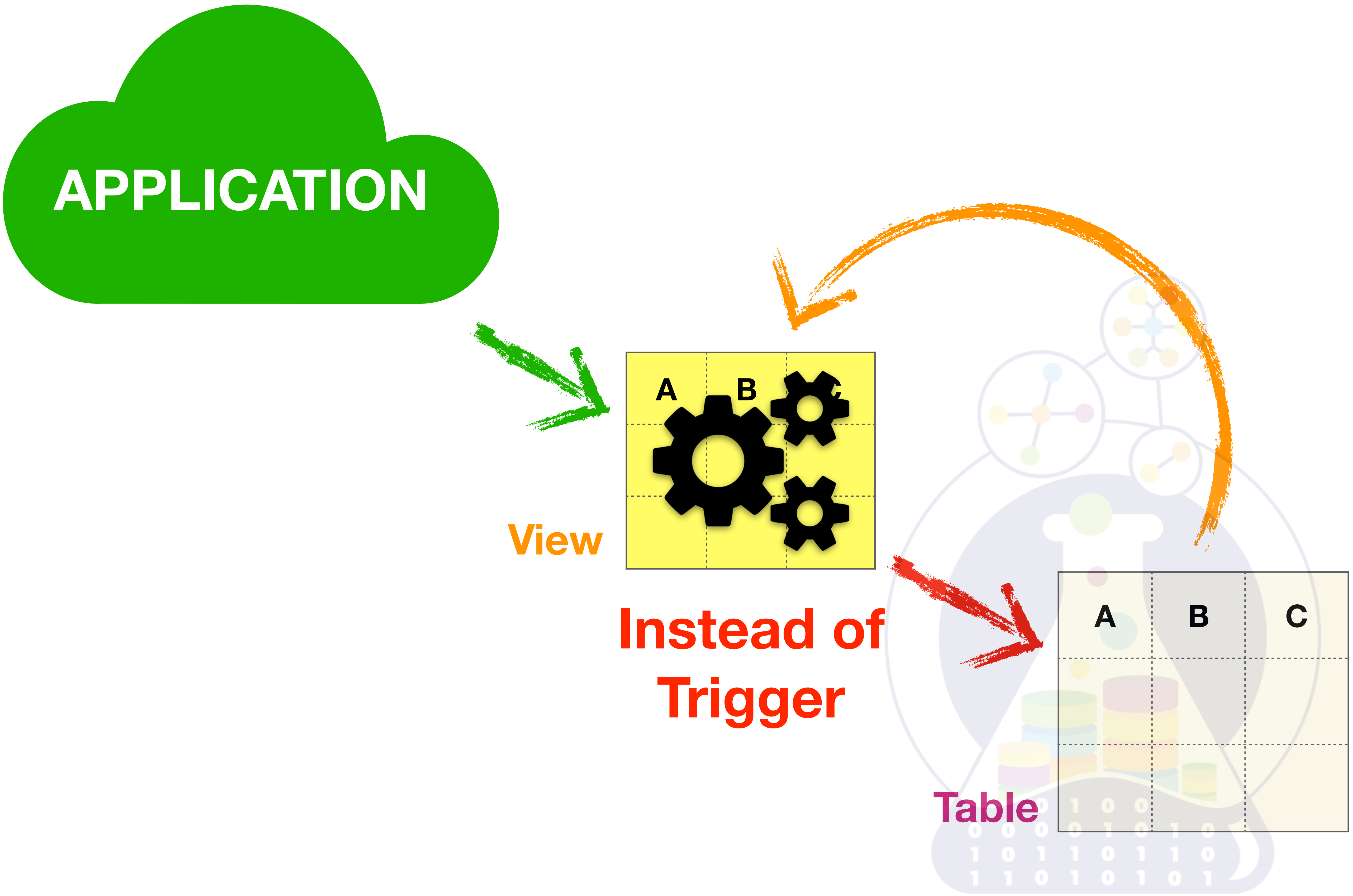


```
CREATE MATERIALIZED VIEW MV$Task
BUILD DEFERRED [ IMMEDIATE ]
REFRESH FORCE [ FAST | COMPLETE ]
ON COMMIT [ ON DEMAND ]  AS
SELECT EMPLOYEE_ID, TASK_NAME
   FROM Task;


 EXEC DBMS_MVIEW.refresh(‘MV$Task’);
```



```
CREATE MATERIALIZED VIEW MV$Task AS
SELECT EMPLOYEE_ID, TASK_NAME
  FROM Task WITH [ NO ] DATA;


 REFRESH MATERIALIZED VIEW MV$Task;
```

not implemented yet



APPLICATION

View

A B C

Instead of Trigger

Table

A B C

Azat Yakupov

COMMIT;