Relational Data a

Kylie A. Bem

Northeastern Uni
Khoury College of Compu

Northeastern U

Learning go

- What is relational data?
- What is SQL?
- Basics of relational algeb
- Types of joins

RELATIONAL

Relational d

- "Relational database" coined b
- Present data to user in relation
  - The database is a collection of data tabl
  - Each table consists of rows and column
  - Each table represents an "entity" (e.g., "c
- Provide relational operators
  - Manipulate data based on relationships
  - Relationships defined on pairs of tables

Why relational

- Real-world data is ofter be stored in a single tab
- Relational data model p
  - A system for organizing such d
  - A system for computing on suc
- Other models are poss

SQL

- Standard language for u

- **S**tructured **Q**uery **L**ang

  - Define relational schemes in a
  - Create/modify/delete tables in
  - Query tables in a database

- High-level, *declarative* la
  for use with the relation

Why SQL

- **Imperative** languages lik
  allow us to specify *how*
- **Declarative** languages li
  specify *what* data manip
  - Allows SQL to be highly optim
  - We don't need to worry how
- Use SQL to query and

Tables in a relationa

- A table is a **multiset** (unordered list) of tuples (rows/records)
- Columns of the table are **attributes** of atomic data type (string, int, etc.)
- Rows/records are **tuples** of attributes specified by a schema

Some
becau

Tables in a relationa

Student

Columns a

sid	name	major	g
0001	John	CS	NU
0002	Lucy	DS	4
0003	Aiden	CS	3

Rows/records are **tuples**

Number of attributes is the **arity** of the table

Tables in a relationa

Course

crn
00234
00653
00783
01945

Student

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00
0003	Aiden	CS	3.33

A **relational database** consists of multiple related tables

Tables correspond to **entities** (e.g., students, courses, etc.)

Table schem

- A table (relation) is def
  - Table name
  - Attributes names
  - Attribute data types
- Student(<u>sid</u>: *string*, name *string*, gpa: *float*)
- <u>Keys</u> are attributes with

Keys

- A **key** is a *minimal* set of *attributes* t *uniquely* identifies a *tuple* in a table
- A **primary key** uniquely identifies tuples in *its own table*
- A **foreign key** uniquely identifies tuples in *another table*

Primary ke

- A **primary key** is a *spec* that unique identifies tu
- A **candidate key** is any could be used as a prim
- A **composite key** is a ke multiple attributes (vs. **s**
    - A **compound key** is a composi are simple *foreign keys*

Primary ke

- A **natural key** uses attri real-world meaning
- A **surrogate key** has no the database
- We may create a surro
  - There is no natural key
  - The natural key would be ineff

Primary ke

sid is

Student

sid	name	majo
0001	John	CS
0002	Lucy	DS
0003	Aiden	CS

- **sid** is a key and likely used by the sc
- **name** will certainly contain duplicate

Foreign ke

- A **foreign key** is a set of a *references* a candidate key
- May be required to refere
- Forms a constraint on allo
- Forms a relationship betw

Foreign ke

student and **course** are foreign keys

(stu

Enrolled

student	course
0002	00653
0002	01945
0003	00783

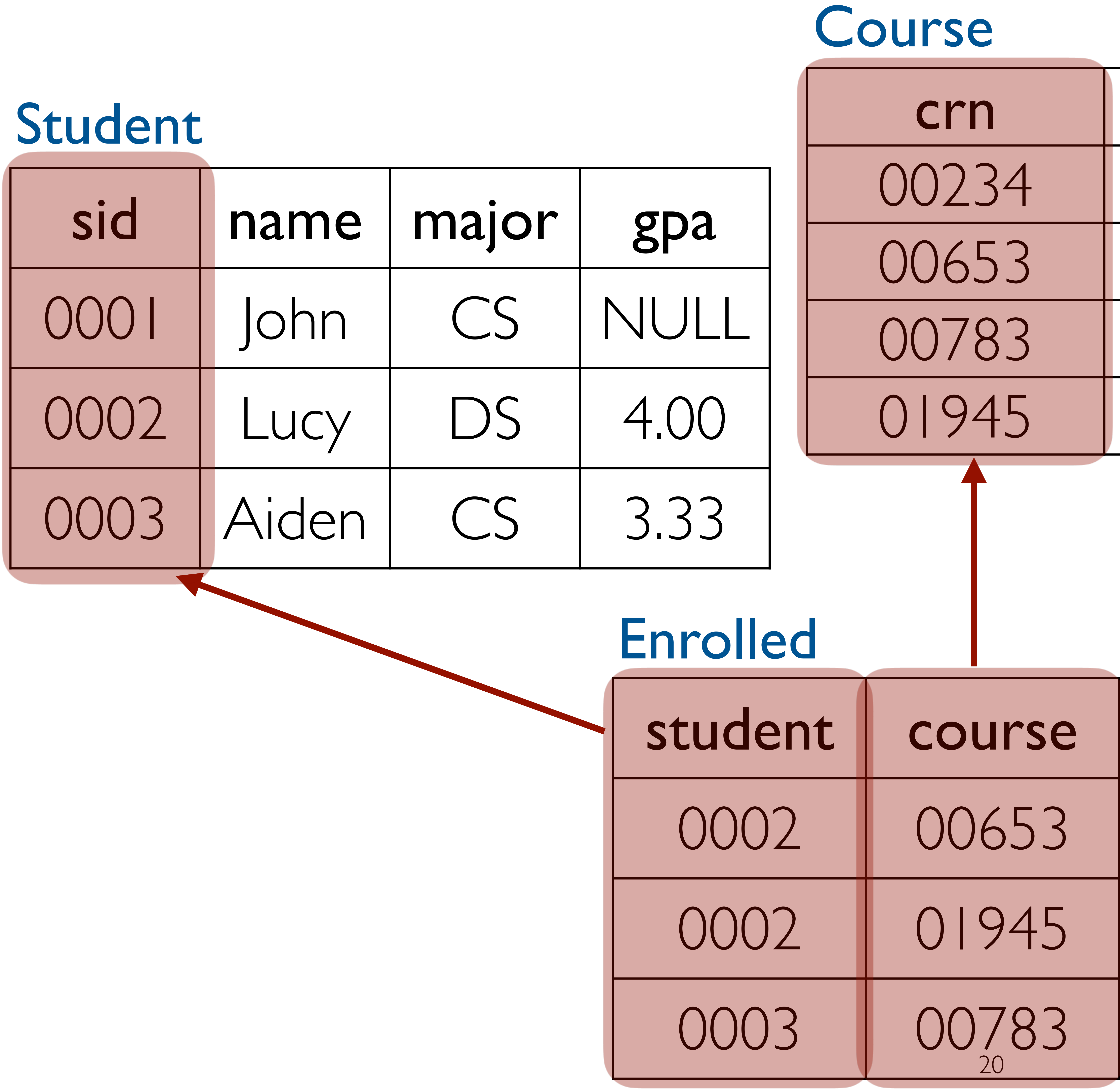
- **(student, course)** is a key with
- **student** and **course** reference

NULL

- Indicates we “don’t kno
  - Missing information
  - Attribute not applicable for tha
- Keys cannot have NULL
- Does not mean zero!
- Be careful of compariso
  - *TRUE And NULL = NULL*
  - *TRUE Or NULL = TRUE*, etc.

Relationships betw

- Relationships between t *primary key-foreign key p*
  - Define a one-to-many relations
  - Allows us to express join oper
- Indexing on keys allows
- Allows for powerful que



Student

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00
0003	Aiden	CS	3.33

Course

crn
00234
00653
00783
01945

Enrolled

student	course
0002	00653
0002	01945
0003	00783

Database norma

- Structure database tabl
  "normal forms"
- Reduce data redundanc
- Improve data integrity
- *Tidy data* follows the "th

Codd’s third norr

- A table is in 1st normal for
  - It stores information in rows and co primary key that uniquely identifies
  - Each column is unique and contains
- A table is in 2nd normal fo
  - It is in 1st normal form, and
  - All non-key columns are dependent
- A table is in 3rd normal fo
  - It is in 2nd normal form, and
  - All non-key columns depend *only* o

RELATIONAL A

Relational alg

- Theoretical foundation for
- Defines operators that tra and produce *output* relatio
- Underlies SQL implementa database management syst

Basic SQL qu

- Basic query form
- “Select” query

```
SELECT <attributes>
FROM <one or mor
WHERE <condition
```

Car

Projection (

- Selects attributes (
- Drops duplicate tu
- SQL : *SELECT*

Student

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00
0003	Aiden	CS	3.33

SELECT na
FROM stu

Student

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00
0003	Aiden	CS	3.33

```
SELECT *
FROM stu
```

Selection (

- Filter tuples (rows) base
- SQL : *WHERE*

Student

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00
0003	Aiden	CS	3.33

```
SELECT r
FROM stu
WHERE g
```

Student

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00
0003	Aiden	CS	3.33

```
SELECT n
FROM stu
WHERE g
```

Rename (ρ

- Rename attributes (colu
- SQL : *AS*

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00
0003	Aiden	CS	3.33

```
SELECT
    sid AS
    name A
FROM stu
```

Cross produc

- All combinations of all t
from both tables
- Not commonly used by

```
SELECT *
FROM student, enrolled
```

Student

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00

×

En

st

↓

sid	name	major	gpa
0001	John	CS	NULL
0001	John	CS	NULL
0002	Lucy	DS	4.00
0002	Lucy	DS	4.00

Natural join

- Combinations of tuples (ro
  equal values for common at
- Also a term for any join on
  attributes (columns)

```
SELECT DISTINCT
    S.sid, name, major, gpa, crn, grade
FROM student S, enrolled E
WHERE S.sid = E.sid
```

Student

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00
0003	Aiden	CS	3.33

⋈

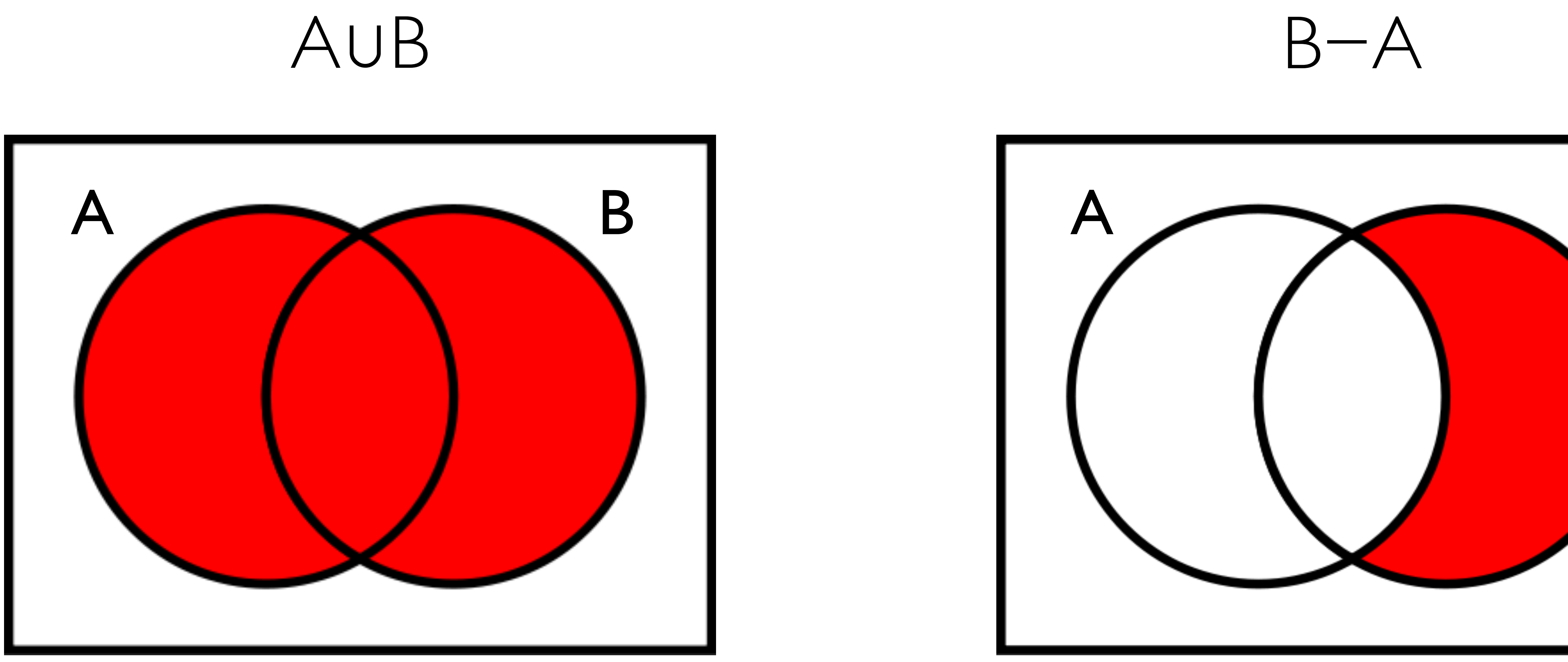
En

↓

sid	name	major	gp
0002	Lucy	DS	4.0
0002	Lucy	DS	4.0
0003	Aiden	CS	3.3

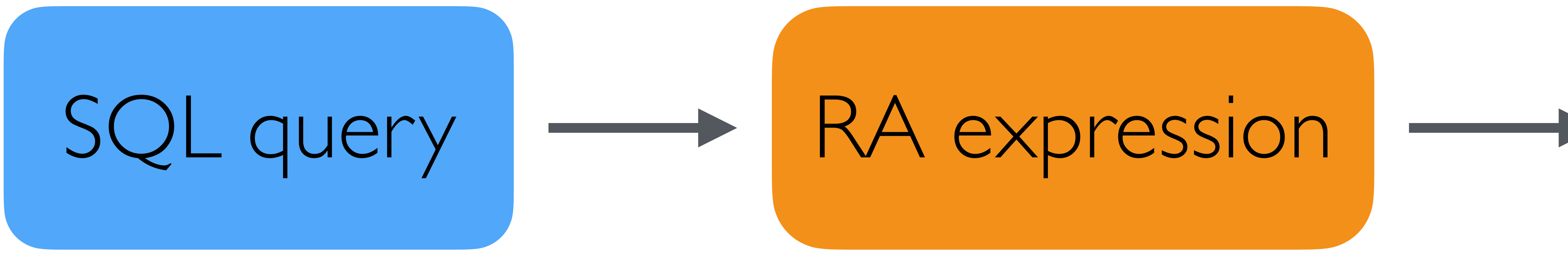
Set union (∪) and di

- Both tables must have t attributes (columns)
- Intersection (∩) definec union and difference



Relational algebra (R

- SQL queries are translat
- SQL engine then optimi
  - Search for logically-equivalent R mathematical properties (comm
  - Optimize to minimize I/O and #


SQL query

RA expression

SQL

Basic SQL qu

```
SELECT [DISTINCT] <attrib
FROM <tables> [aliases]
WHERE <conditions>
[GROUP BY <attributes>]
[HAVING <conditions>]
[ORDER BY <attributes>]
```

Basic SQL qu

SELECT selects columns
DISTINCT eliminates duplicates
AS renames columns with an al
FROM specifies which tables to
WHERE filters rows based on c
GROUP BY groups rows with s
HAVING filters the groups
ORDER BY sorts the output

SELECT nam
FROM stude
WHERE ma

Student

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00
0003	Aiden	CS	3.33

SELECT nam
FROM stude
ORDER BY

Aggregatio

COUNT() number of d
SUM() sum of values
AVG() mean of values
MAX() maximum value
MIN() minimum value

Student

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00
0003	Aiden	CS	3.33

```
SELECT AV
FROM stude
WHERE ma
```

```
SELECT maj
FROM stude
GROUP BY
HAVING gp
```

JOINS

Joins

- Join between two tables ret combinations of tuples mee
- Typically join on primary key
- Often multiple ways to expr

```
SELECT                                  SEL
    sid, name, course, grade
FROM student, enrolled          =       FRO
WHERE sid = student                     JOI
```

Student

sid	name	major	gpa
0001	John	CS	NULL
0002	Lucy	DS	4.00
0003	Aiden	CS	3.33

JOIN

sid	name	c
0002	Lucy	0
0002	Lucy	0
0003	Aiden	0

Types of jo

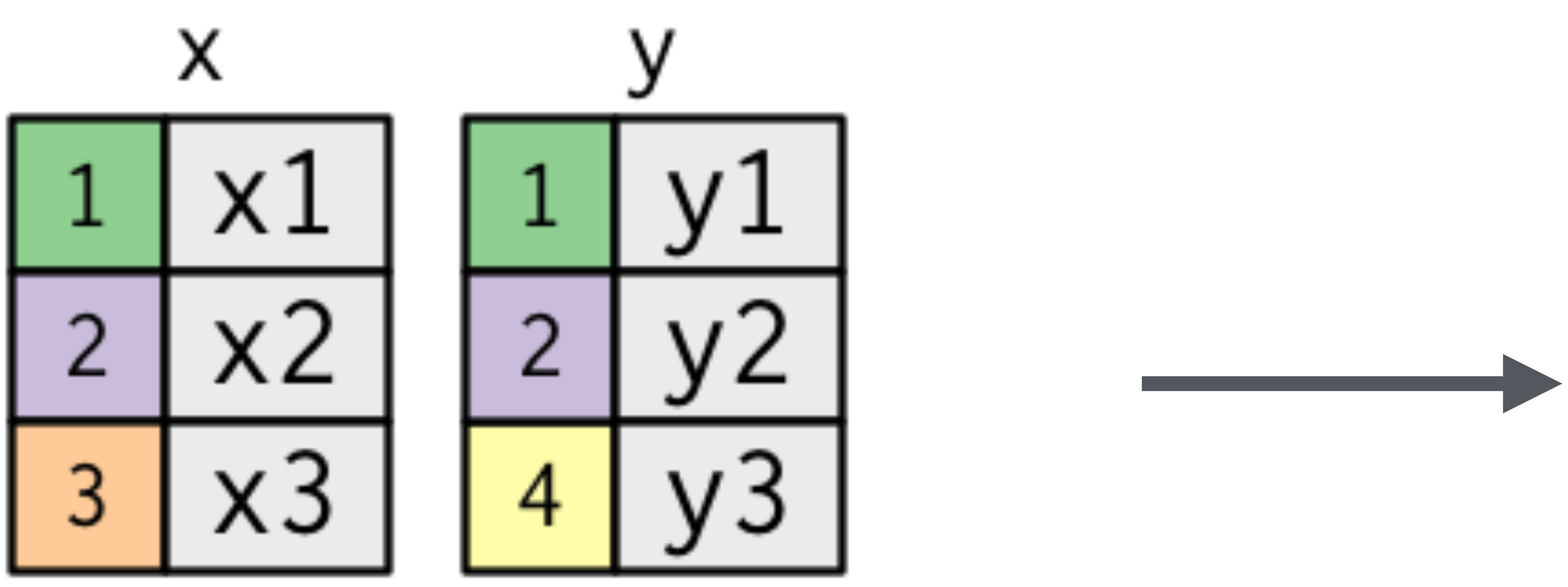
- Joins are distinguished by wh
  columns are retained in the

- Mutating joins keep columns
  - *Inner joins* keep **only** rows with equal
  - *Outer joins* keep any rows that appear

- Filtering joins keep columns
  - *Semi joins* keep only rows that appear
  - *Anti joins* drop rows that appear in a s

Visualizing jo

Look for rows with matc



x

1	x1
2	x2
3	x3

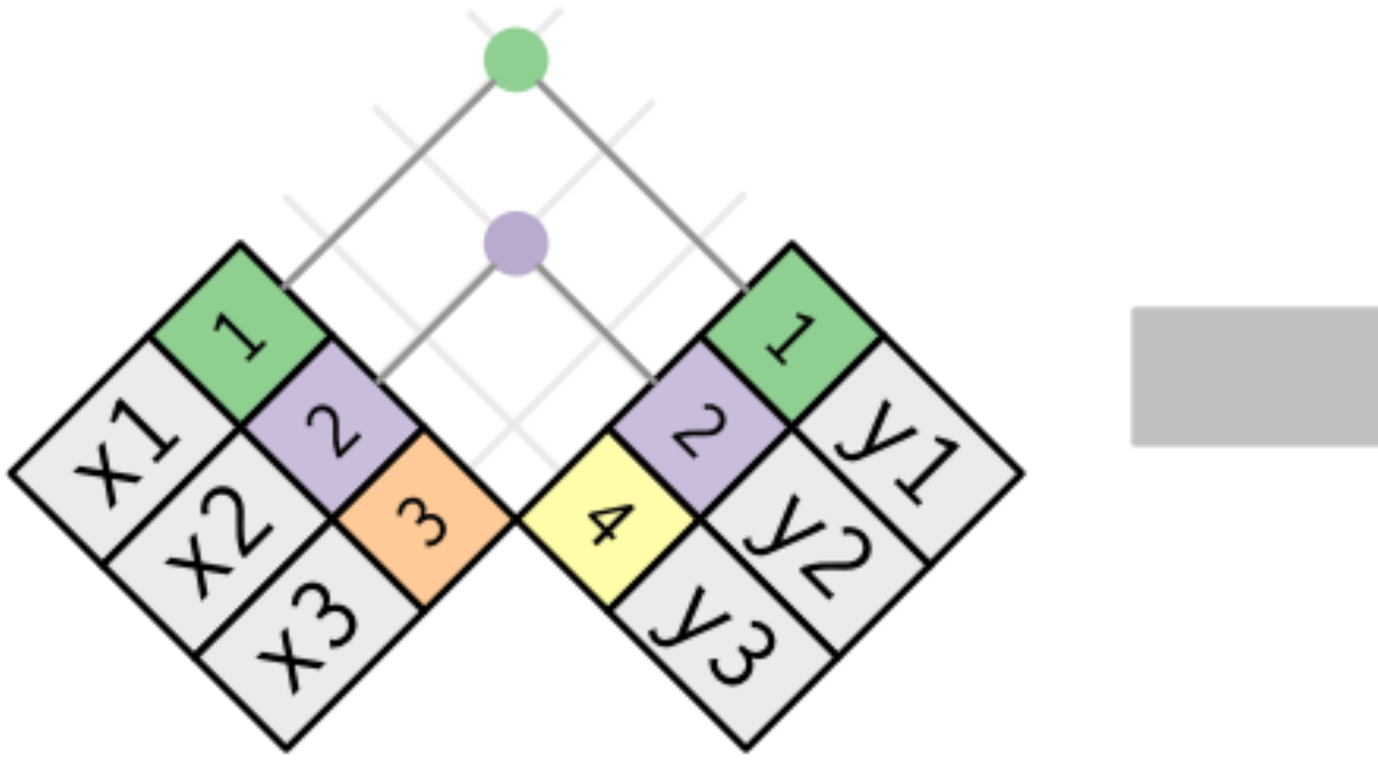
y

1	y1
2	y2
4	y3

Two tables

Inner joi

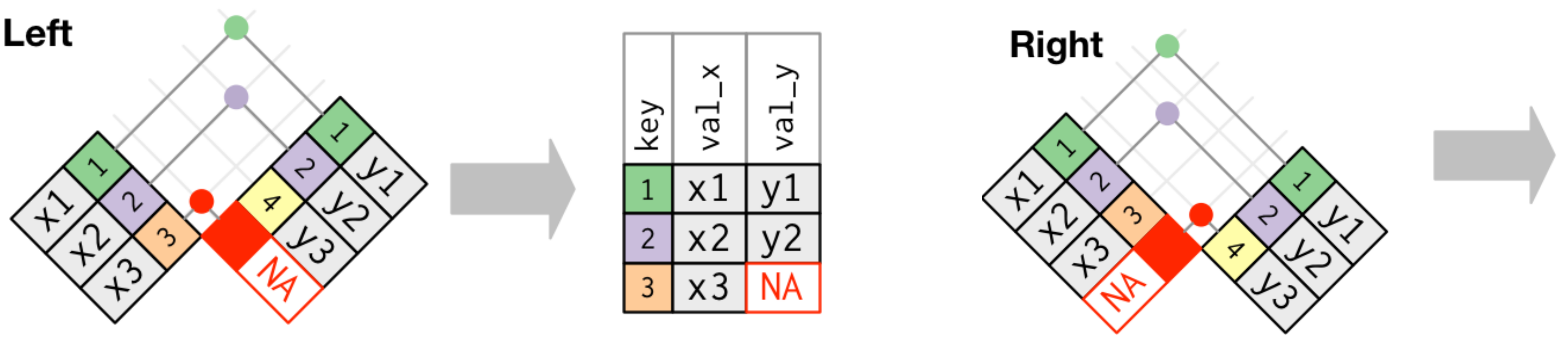
- Keep **only** rows with matchi
- Useful for retaining only com



SELECT * FROM x JOIN y

Outer joir

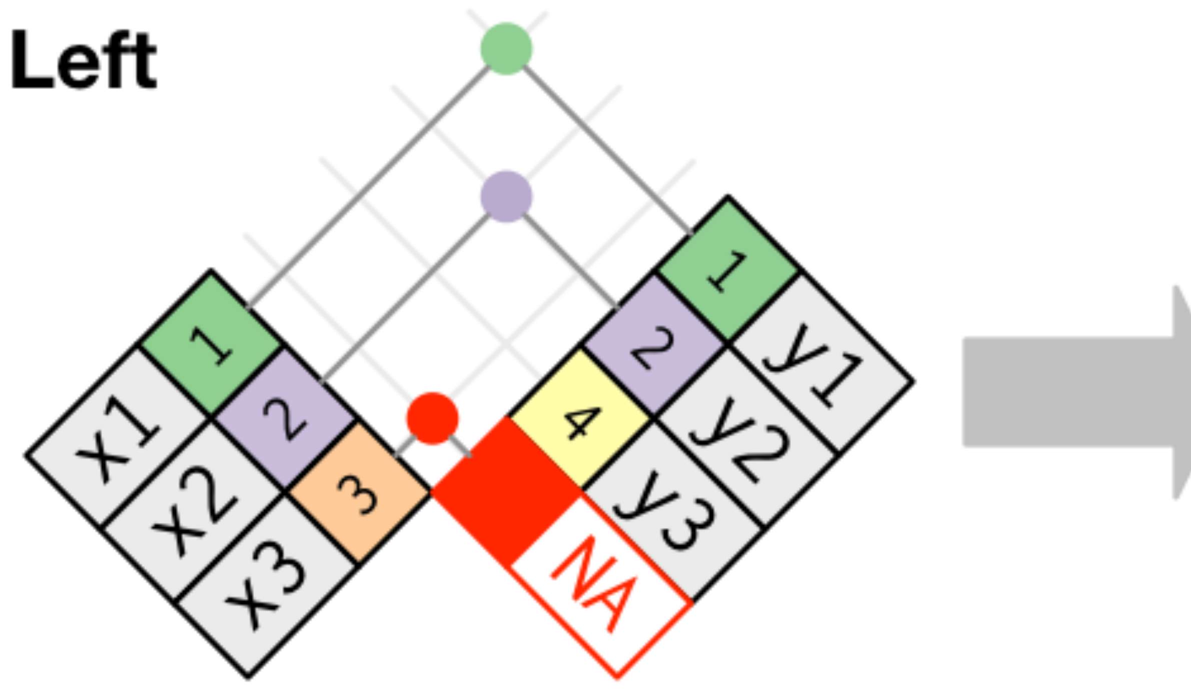
- Keep **any** rows that appear in
- Fill in non-matches with missi
- Useful for annotating one tab
another while retaining the o



key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA

Left outer j

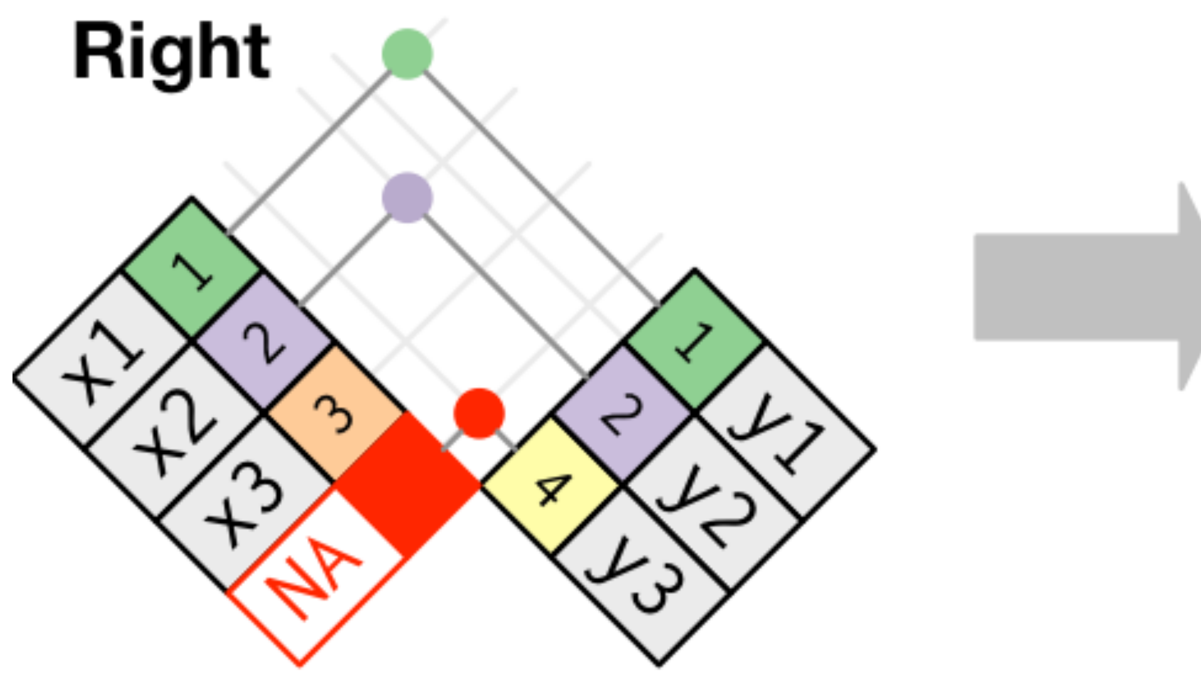
- Keep **all** rows that appear in l
- Most common type of join in



SELECT * FROM x LEFT JO

Right outer

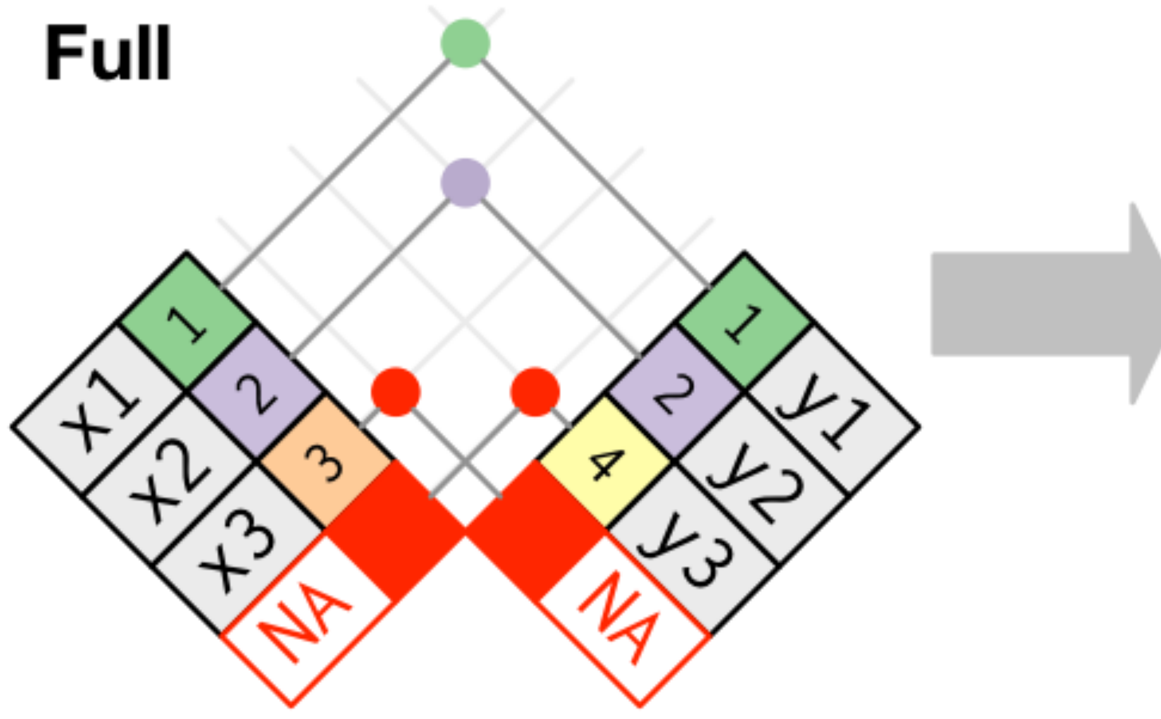
- Keep **all** rows that appear in
- Can be expressed with an eq



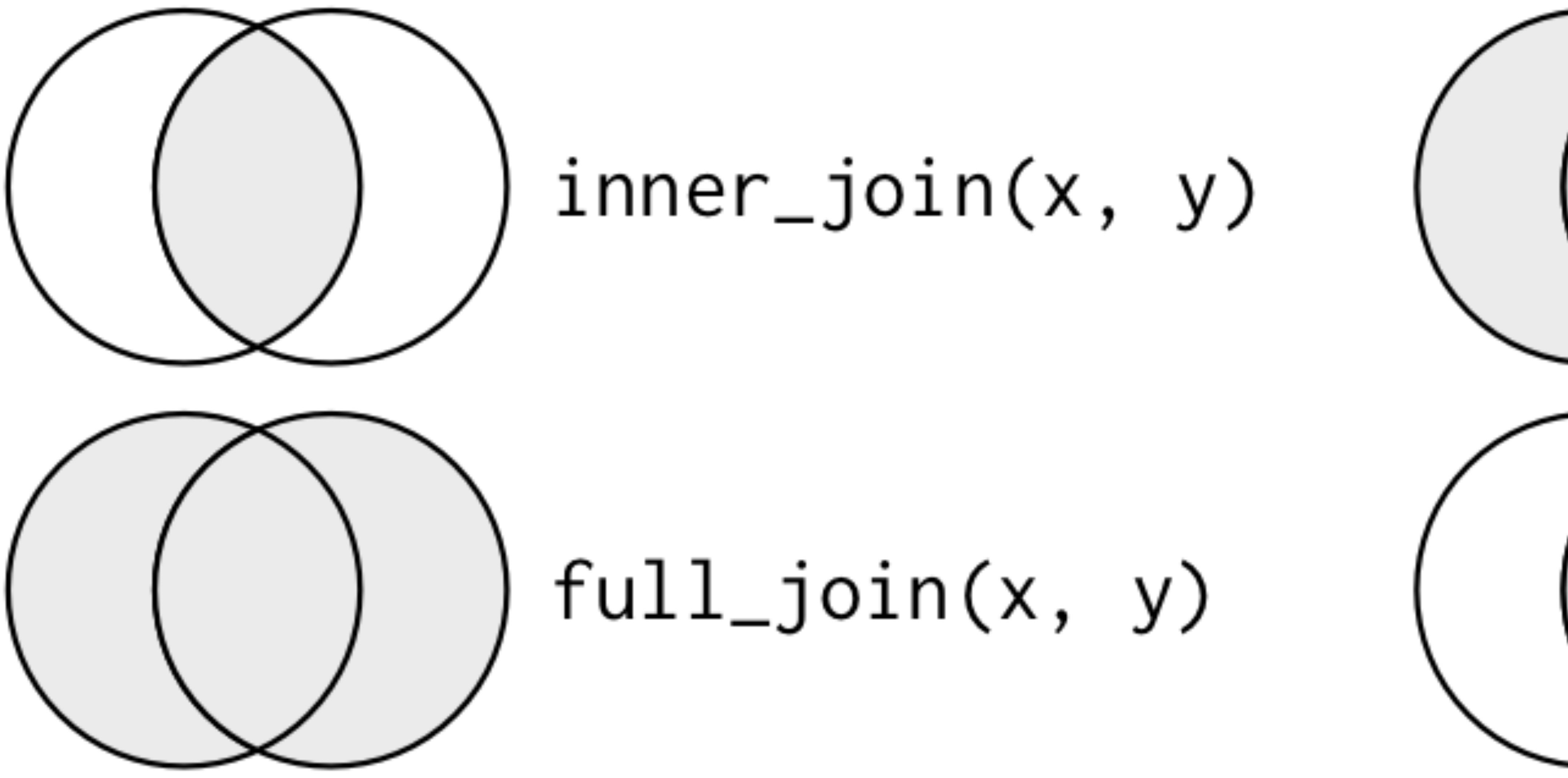
SELECT * FROM x RIGHT

Full outer jo

- Keep **all** rows that appear in
- Not often used in data analys



SELECT * FROM x FULL JO

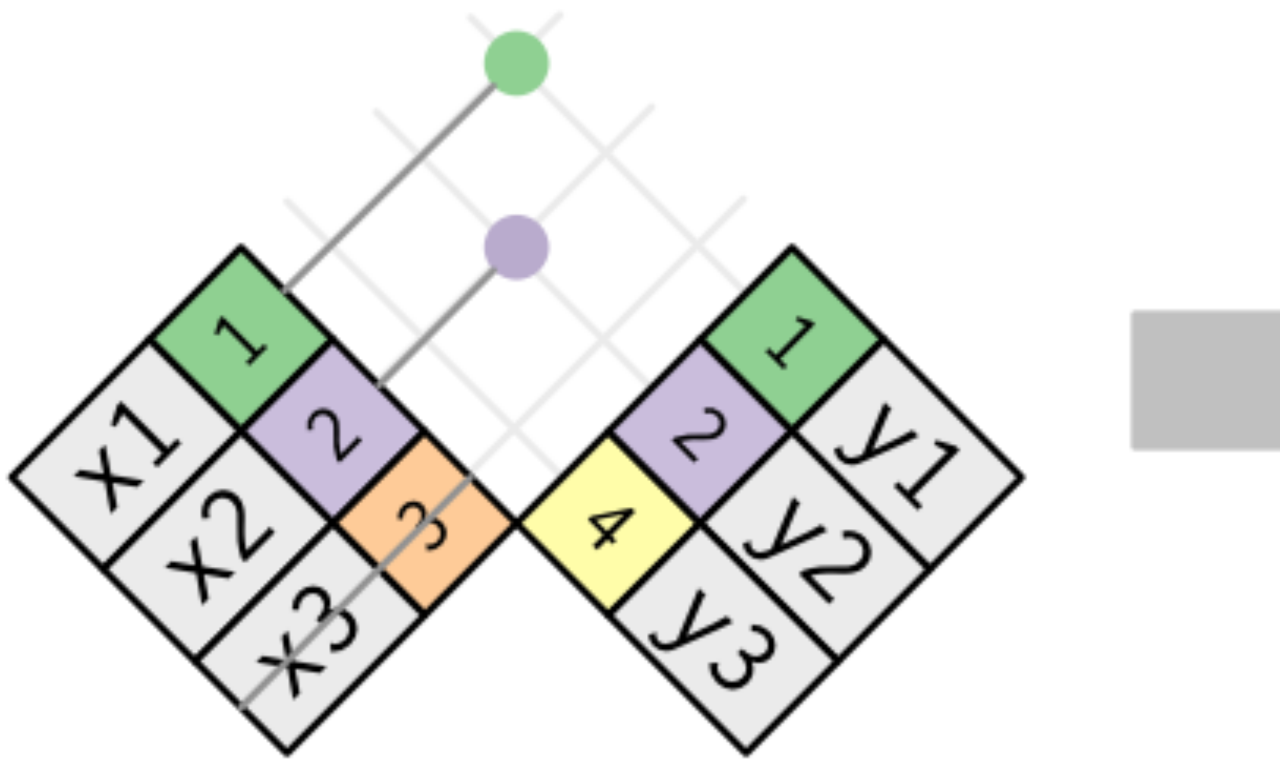


inner_join(x, y)

full_join(x, y)

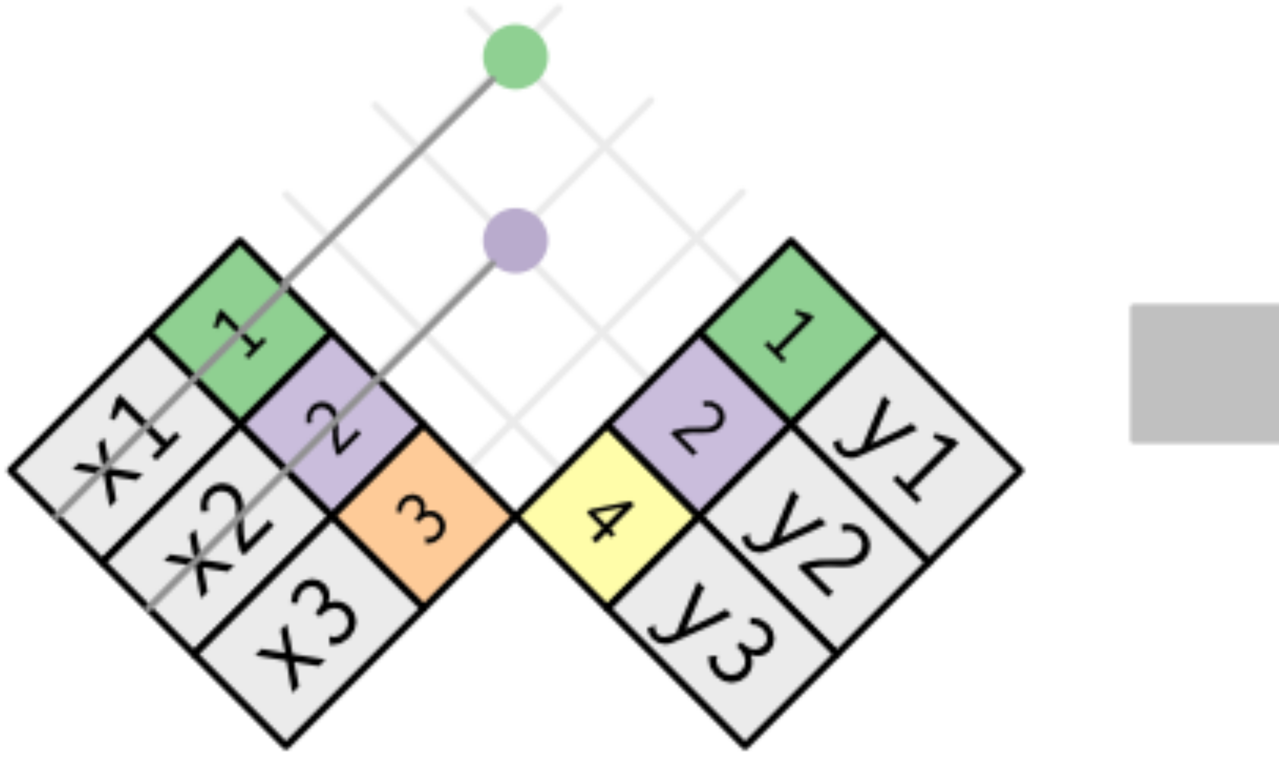
Semi join

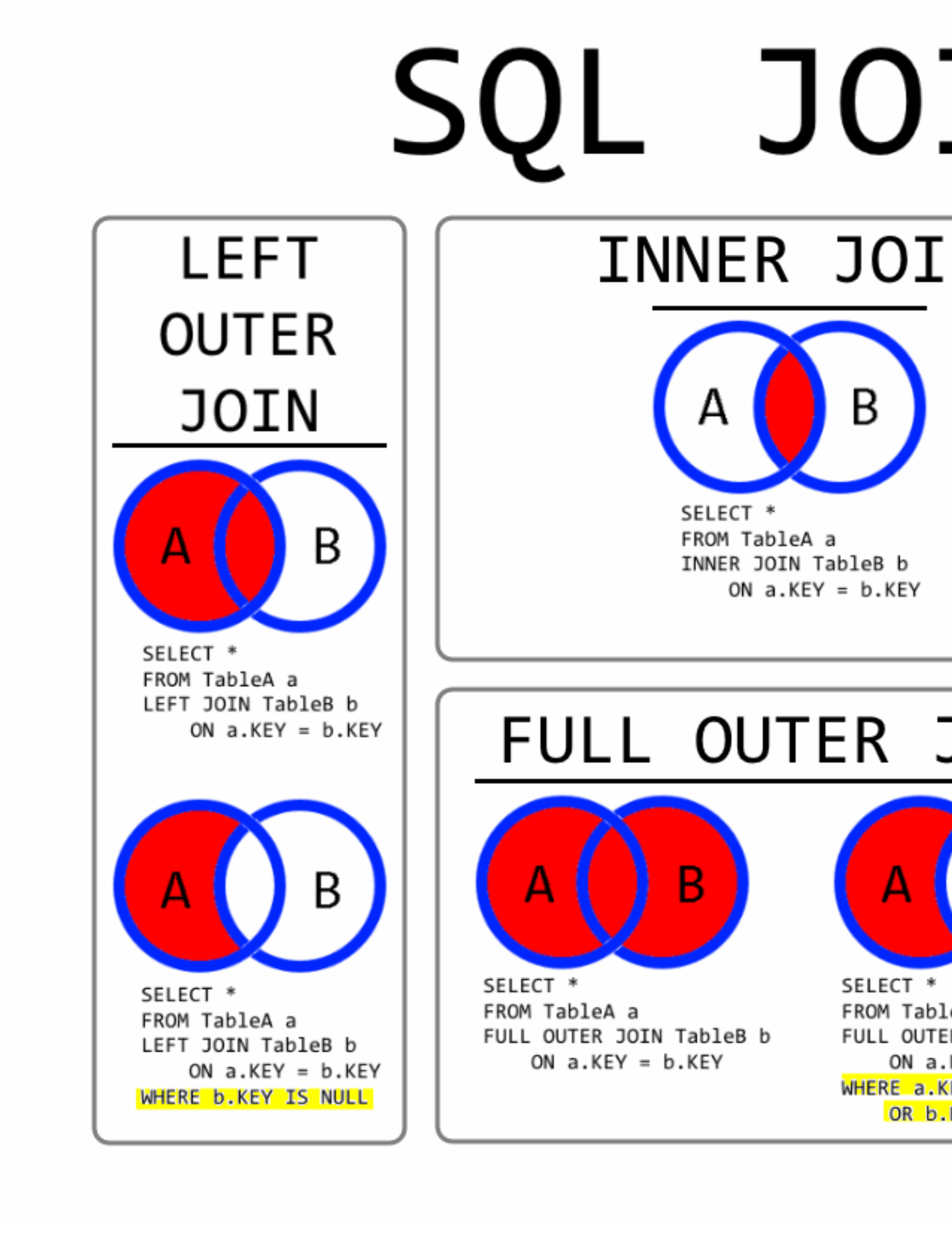
- Keep rows in one table that
- Useful for filtering



Anti join

- Drop rows in one table tha
- Useful for filtering





SQL JOI

LEFT OUTER JOIN

```
SELECT *
FROM TableA a
LEFT JOIN TableB b
    ON a.KEY = b.KEY
```

```
SELECT *
FROM TableA a
LEFT JOIN TableB b
    ON a.KEY = b.KEY
WHERE b.KEY IS NULL
```

INNER JOI

```
SELECT *
FROM TableA a
INNER JOIN TableB b
    ON a.KEY = b.KEY
```

FULL OUTER J

```
SELECT *
FROM TableA a
FULL OUTER JOIN TableB b
    ON a.KEY = b.KEY
```

```
SELECT *
FROM Tabl
FULL OUTE
    ON a.
WHERE a.K
    OR b.
```