ACADEMIND

Remember:
Primary keys don't have to be auto-incrementing integers

Composite Primary Keys

You must only have one primary key per table

BUT

Primary keys can span multiple columns

"`id`" is a **surrogate key**

It acts as a unique primary identifier but it isn't the real unique identification criteria

id	employee_id	project_id
1	3	2
2	1	1
3	1	2

Composite Primary Keys

You must only have one primary key per table

BUT

Primary keys can span multiple columns

"`employee_id`" + "`project_id`" form the **real key**

The unique identification criteria of each row is the combination of these two columns

employee_id	project_id
3	2
1	1
1	2

Composite Primary Keys

You must only have one primary key per table

BUT

Primary keys can span multiple columns

employee_id	project_id
3	2
1	1
1	2

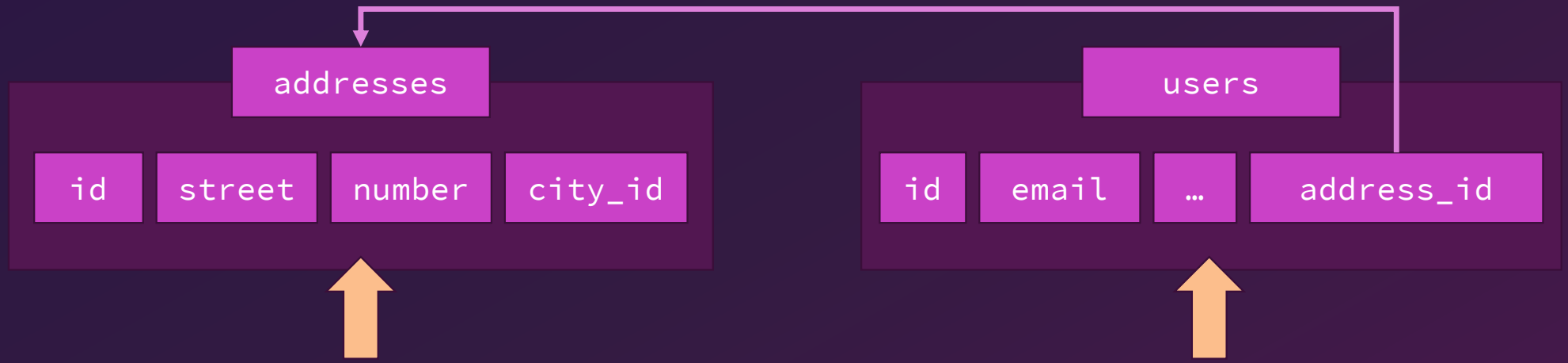
```
CREATE TABLE projects_employees (
  employee_id INT,
  project_id INT,
  PRIMARY KEY (employee_id, project_id)
);
```

Composite Primary Key

ACADEMIND

There is nothing wrong with using surrogate keys!

Surrogate Keys Are Often Very Useful



Actually, an address is uniquely identified by the **combination of all its column values** (street + house number + city + ...)

A user may have a better unique identification field – e.g. an email address

For application purposes and performance reasons, surrogate keys might still often suffice or even be preferable

Self-Referential Relationships

Example: Employees with supervisors

Self-referential

A data entity has a relationship to itself (i.e. internal relationship)

employees

id	f_name	email	supervisor_id
1	Julie	j@t.com	NULL
2	Chris	c@t.com	1
3	Max	m@t.com	2

Self-Referential Many-to-Many Relationships