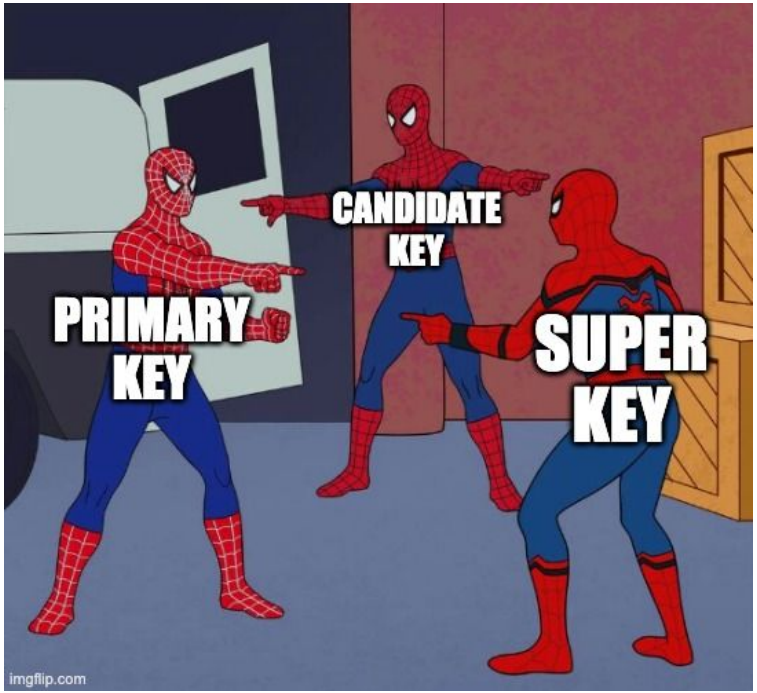
Lab 2: The Relational Model



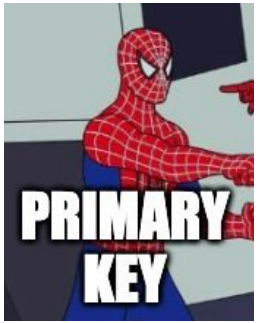
What is a key?

A key uniquely identifies each record in a table.

Remember: One key = one record

However: One record != one key

*A record is not limited to having only one key



can be

can be

- Chosen among candidate keys
- Minimal set of attributes that uniquely identifies a record
- A set of attributes that includes the candidate key as a **proper** subset

Key Example

Course_Sections (**course_id**: integer, **name**: string, **faculty_id**: integer, **section_start**: date)

What could be the candidate keys?

For each candidate key, what are all the super keys?

Candidate key 1: {course_id}

super keys = {
{course_id name}, {course_id faculty_id}, {course_id start},
{course_id name faculty_id},
{course_id name start}, {course_id faculty_id start},
{course_id name faculty_id start}
}

Candidate key 2: {name,faculty_id}

Candidate key 3: {name,start}

Creating relations with SQL

Consider relation schemas and business rules below.
Write create table statements that encode these relation schemas.
Give an example of a valid relation instance.

Stocks (***name***: string, ***price***: integer, ***exchange***: string, ***IPO_date***:date, ***industry***:string)

1) Each stock has a name, and no two stocks have the same *name*.
2) No two stocks have the same combination of *IPO_date* and *industry*. Two companies from same industry cannot IPO on same day to prevent confusion.
3) Each stock must have a *price* and an *industry*.
4) Not all stocks have an *IPO_date* or an *exchange*.

Creating relations with SQL (solution)

Stocks (***name***: string, ***price***: integer, ***exchange***: string, ***IPO_date***:date, ***industry***:string)

1) Each stock has a name, and no two stocks have the same *name*.
2) No two stock have the same combination of *IPO_date* and *industry*.
3) Each stock must have a *price* and an *industry*.
4) Not all stocks have an *IPO_date* or an *exchange*.

```
create table Stocks (
   name              varchar(128) primary key,
   price             integer              not null,
   exchange          varchar(128),
   IPO_date          date,
   industry          varchar(32)          not null,
   unique (IPO_date, industry)
);
```

Creating relations with SQL 2

Consider relation schemas and business rules below.
Write create table statements that encode these relation schemas.
Give an example of a valid relation instance.

Soft_Drinks (**id**: integer, **name**: string, **description**: string, **price**: decimal, **quantity**: integer)

1) All drinks have an *id*, and no two drinks have the same *id*
2) Each drink has a *name*, a *price,* and a quantity in stock *(quantity)*
3) Every drink is always in stock
4) No two drinks have the same name
5) Not all drinks have a description

Creating relations with SQL 2

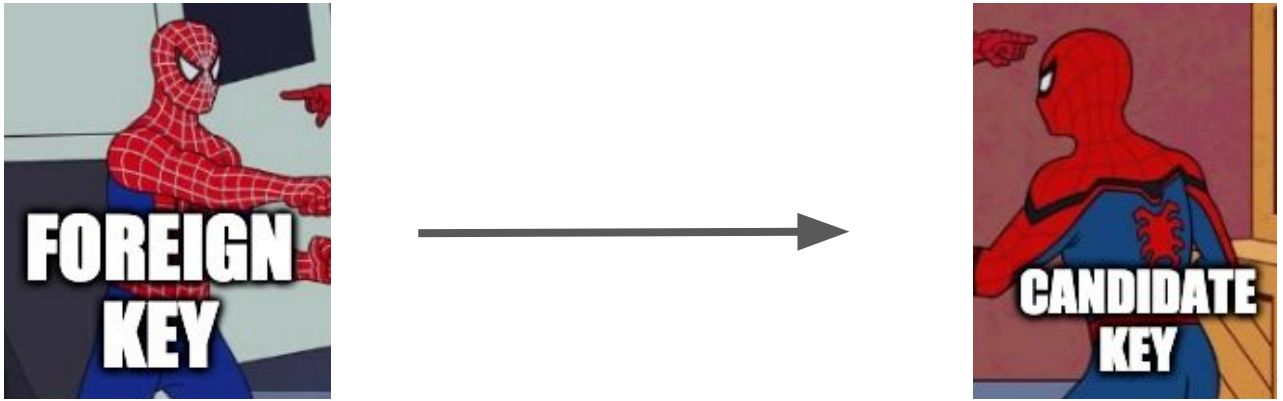
Soft_Drinks (**id**: integer, **name**: string, **description**: string, **price**: decimal, **quantity**: integer)

1) All drinks have an *id*, and no two drinks have the same *id*
2) Each drink has an id, a *name*, a *price,* and a quantity in stock *(quantity)*
3) Every drink is always in stock
4) No two drinks have the same name
5) Not all drinks have a description

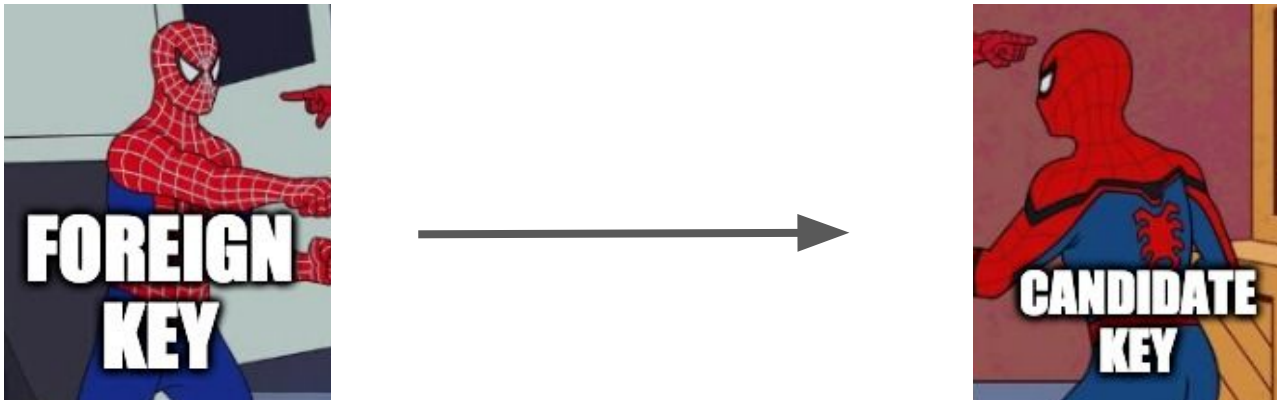
```
create table Soft_Drinks (
   id            integer              primary key,
   name          varchar(128)         not null unique,
   description   varchar(1024),
   price         decimal              not null,
   quantity      integer              not null,
   CHECK (quantity > 0)
);
```

Foreign key constraints

- A foreign key defines a (directed) link between **tuples** in different relations
- Foreign keys enforce referential integrity: a requirement that a **value** in an attribute or attributes of a tuple in one relation must also appear as a **value** in another relation
- A foreign key in a relation must reference (point to) a **candidate key** in another relation



Foreign key constraints



Enrollment(**sid**, **cid**, grade), Students(**sid**), Courses(**cid**)

Enrollment.**sid** must refer to an existing student, i.e., it must match Students.**sid** for some tuple in Students
Enrollment.**cid** must refer to an existing course, i.e., it must match Courses.**cid** for some tuple in Courses

Declaring foreign key constraints

- A foreign key in a relation must reference (point to) a **candidate key** in another relation.
- That is, the target column(s) are either a primary key or designated as UNIQUE
- Some relational systems limit this further: a foreign key must point to a primary key. For efficiency reasons, pointing to a primary key is usually a better choice.

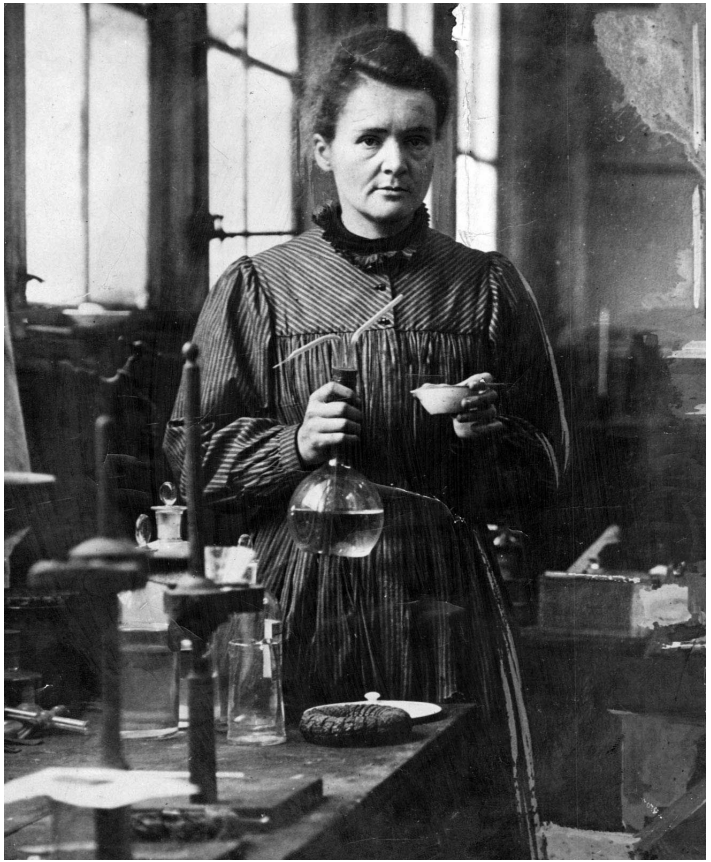
Foreign key examples

Consider relation schemas and business rules below. Primary keys are given, underlined in the relation schemas.

Write create table statements that encode these relation schemas, with the right foreign key constraints. Give an example of valid relation instances.

Scientists (*name*: string, *field*: string, *award*: string)
Books (*ISBN*: integer, *name*: string, *author*: string)

All scientists have a name, which uniquely identifies them.
Each book is written by a scientist.
If a scientist retires, the corresponding books tuples will be deleted.



Madame Curie. Nobel Prize Physics 1903. Chemistry 19011

Foreign key examples

Scientists (name: string, field: string, award: string)
Books (ISBN: integer, name: string, author:string)

All scientists have a name, which uniquely identifies them.
Each book is written by a scientist.
If a scientist retires, the corresponding books tuples will be deleted.

```
create table Scientists (
        name        varchar(128)      primary key,
        field       varchar(64),
        award       varchar(128)
);
create table Books (
        ISBN        integer       primary key,
        name        varchar(128),
        author      string        not null,
        foreign key (author) references Scientists(name) on delete cascade
);
```

Mdm Tu Youyou. Nobel Prize Medicine 2015

Foreign key examples

Consider relation schemas and business rules below. Primary keys are given, underlined in the relation schemas.

Write create table statements that encode these relation schemas, with the right foreign key constraints. Give an example of valid relation instances.

Players (*name*: string, *game*: string, *team*: string)

Teams (*name*: string, country: string, *sponsor*: string, owner: string)

Organizations(*name*: string, *country*: string)

All players have a name that uniquely identifies them, and they all belong to a pro esports team.
If a team is disbanded, its players are made to join Team Pikachu.
Teams can be owned by an organization.
If an organization is disbanded, the teams it owns are disbanded as well.

Foreign key examples

Players (*name*: string, *game*: string, *team*: string)

Teams (*name*: string, country: string, *sponsor*: string, owner: string)

Organizations(*name*: string, *country*: string)

All players have a name that uniquely identifies them, and they all belong to a pro esports team.
If a team is disbanded, its players are made to join Team Pikachu.

Teams can be owned by an organization.

If an organization is disbanded, the teams it owns are disbanded as well

```
create table Organizations (
        name            varchar(128),
        country         varchar(128),
        primary key (name, country)
);
```

```
create table Teams (
        name            varchar(128)  primary key,
        sponsor         varchar(128),
        owner           varchar(128),
        country         varchar(128),
        foreign key (owner, country) references
                Organizations(name, country) on delete cascade
);
```

```
create table Players (
        name            varchar(128) primary key,
        country         varchar(64),
        game            varchar(64),
        team            varchar(128) not null default 'Team Pikachu',
        foreign key (team) references Teams(name) on delete set default
);
```

Foreign key examples

Customers (*cid*: integer, name: string, address: string)

Products (*pid*: integer, name: string, description: string, price: decimal, qty_stock: integer)

Purchases (*cid*: integer, *pid*: integer, *datetime*: date)

Consider a small e-commerce database:

- Each customer is uniquely identified by *cid* and also has a *name* and *address*.
- Each product is uniquely identified by *pid* and has a *name*, *price* and *qty_stock,* while not all products have a description.
- Products are purchased by customers. A product can be purchased by the same customer on different dates.
- If a product is no longer sold, its *qty_stock* is updated to 0

Foreign key examples

Consider a small e-commerce database:

- Each customer is uniquely identified by *cid* and also has a *name* and *address*.
- Each product is uniquely identified by *pid* and has a *name*, *price* and *qty_stock,* while not all products have a description.
- Products are purchased by customers. A product can be purchased by the same customer on different dates.
- If a product is no longer sold, its *qty_stock* is updated to 0

```
create table Customers (
   cid          integer              primary key,
   name         varchar(128)         not null,
   address      varchar(1024)        not null
);
create table Products (
   pid                  integer      primary key
   name                 varchar(128) not null,
   description          varchar(1024),
   price                decimal      not null,
   qty_stock            integer      not null
);
```

```
create table Purchase (
        cid                     integer,
        pid                     integer,
        datetime        date,
        primary key (cid, pid, datetime),
        foreign key (cid)    references Customers (cid),
        foreign key (pid)    references Products (pid)
);
```