Index Design Problem

```
create table Company(
  id int primary key,
  name varchar(500) not null,
  product varchar(500)
);
create table Person(
  id int primary key,
  name varchar(200) not null unique,
  worksFor int,
  foreign key (worksFor) references Company(id)
    on update cascade on delete no action
);
create table PersonEmail(
  person int not null,
  foreign key (person) references Person(id)
    on update cascade on delete cascade,
  email varchar(200) not null,
  primary key(person, email)
);
```

The most common query is to look up the name of the company that a person works for.

Index Design Problem 1 Analysis

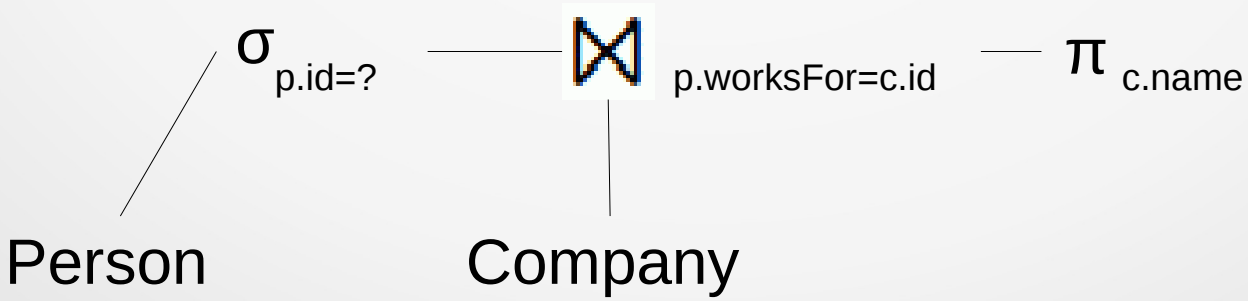
The most common query is to look up the name of the company that a person works for.

No attribute of Person was specified, so we start with the id of the person. The pipeline for processing this query consists of finding the Person using the id, then joining with Company, finally extracting the name of the company.

Index Design Problem 1 Analysis

The most common query is to look up the name of the company that a person works for.

No attribute of Person was specified, so we start with the id of the person. The pipeline for processing this query consists of finding the Person using the id, then joining with Company, finally extracting the name of the company.



Index Design Solution Part 1

The primary keys, uniqueness constraints and foreign key targets must all have indexes. These indexes do not require a range query, so we start with:

```
Person(id) hash
Person(name) hash
Company(id) hash
PersonEmail(person,email) hash
```

Index Design Solution Part 2

Now we look at the query pipeline. The selection on p.id is an exact match so Person(id) hash is sufficient. The join uses c.id to obtain the Company record. This is an exact match so Company(id) is sufficient. The solution is therefore:

```
Person(id) hash
Person(name) hash
Company(id) hash
PersonEmail(person,email) hash
```