NOT EXIST和NOT IN怎么选

在SQL中，”not exist”通常与子查询一起使用，用于检查子查询中是否存在至少一行数据。如果子查询没有返回任何行，则外层查询的条件结果为真。以下是一个使用`NOT EXISTS`的例子，假设我们有两个表：`employees`（员工表）和`departments`（部门表）。我们想要找出那些没有员工的部门：

```
SELECT *
FROM departments d
WHERE NOT EXISTS (
    SELECT 1
    FROM employees e
    WHERE e.department_id = d.id
);
```

这个查询的意思是：从`departments`表中选择所有记录，其中不存在`employees`表中与之对应的`department_id`。如果`employees`表中没有任何记录与`departments`表中的`id`相匹配，那么`NOT EXISTS`条件为真，这样的部门就会被选出来。

使用`NOT IN`代替`NOT EXISTS`时，你的查询会检查外层查询的某个字段是否不在子查询返回的值列表中。继续使用之前的例子，如果我们想要找出那些没有员工的部门，使用`NOT IN`的查询会是这样：

```
SELECT *
FROM departments d
WHERE d.id NOT IN (
    SELECT e.department_id
    FROM employees e
);
```

这个查询的意思是：从`departments`表中选择所有记录，其中`id`不包含在`employees`表的`department_id`列所返回的值列表中。如果`departments`表

中的某个`id`没有在`employees`表的`department_id`中出现，那么这个部门就会被选出来。
然而，需要注意的是，使用`NOT IN`时如果子查询返回的值列表中包含NULL值，可能会导致不预期的结果。因为`NULL`与任何值的比较（包括不等于`<>`）结果都是未知的，所以如果子查询返回的结果中包含NULL值，`NOT IN`子句可能不会返回任何行，即使外层查询的某些行应该被返回。这就是为什么在某些情况下，使用`NOT EXISTS`可能是更好的选择，因为它不会受到`NULL`值的影响。

当然，让我们考虑一个更复杂的例子，其中涉及到多个表和条件。假设我们有三张表：`orders`（订单表）、`order_details`（订单详情表）和`products`（产品表）。我们想要找出所有没有被订购过的产品。
表结构可能如下：

* `orders` 表有 `order_id` 和 `customer_id` 字段。
* `order_details` 表有 `order_id`、`product_id` 和 `quantity` 字段。
* `products` 表有 `product_id` 和 `product_name` 字段。

我们想要找出所有在`order_details`表中没有对应`product_id`的`products`表中的产品。以下是使用`NOT EXISTS`的查询示例：

```
SELECT p.product_id, p.product_name
FROM products p
WHERE NOT EXISTS (
    SELECT 1
    FROM order_details od
    WHERE od.product_id = p.product_id
);
```

这个查询的意思是：从`products`表中选择所有记录，其中不存在`order_details`表中与之对应的`product_id`。如果`order_details`表中没有任何记录与`products`表中的`product_id`相匹配，那么`NOT EXISTS`条件为真，这样的产品就会被选出来。
现在，让我们增加一些复杂性。假设我们只想找出那些在某个特定日期之后没有被订购过的产品。我们可以加入一个`orders`表的条件来检查订单日期：

```
SELECT p.product_id, p.product_name
FROM products p
WHERE NOT EXISTS (
```

```
    SELECT 1
    FROM order_details od
    INNER JOIN orders o ON od.order_id = o.order_id
    WHERE od.product_id = p.product_id
    AND o.order_date > '2023-01-01'
);

```

在这个查询中，我们加入了`INNER JOIN`来连接`orders`表和`order_details`表，并且增加了一个条件`o.order_date > '2023-01-01'`来过滤出在2023年1月1日之后下的订单。这样，我们就能找出所有在2023年1月1日之后没有被订购过的产品。