**SQL important points to note**

[**Join vs Exists vs In (SQL)**](http://www.gregreda.com/2013/06/03/join-vs-exists-vs-in/)

*June 03, 2013 | Tags:****[sql](http://www.gregreda.com/tag/sql.html" \o "Posts tagged with )***[***databases***](http://www.gregreda.com/tag/databases.html)

Last weekend, I came across [**Jeff Atwood**](http://en.wikipedia.org/wiki/Jeff_Atwood)'s excellent [**visual explanation of SQL joins**](http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html) on Hacker News.

It reminded me of teaching SQL to the incoming batch of [**PwC FTS**](http://www.pwc.com/us/en/forensic-services/technology-solutions.jhtml) associates a few years ago. Not many of them had prior programming experience, much less SQL exposure, so it was a fun week to learn how well us instructors could teach the topic.

Most of them intuitively picked up on how the IN clause worked, but struggled with EXISTS and JOINs initially. An explanation that always seemed to help illustrate the concept was to show that often you can write the exact same query using an IN, EXISTS, or a JOIN.

As an example, let's assume the following two tables, which we'll call *tableA* and *tableB*.

id name id title

-- ---- -- ----

1 Kenny 1 Analyst

1 Rob 2 Sales

4 Molly 3 Manager

1 Greg

2 John

If we wanted to get everyone that's an Analyst, we could do the following:

**SELECT** \*

**FROM** tableA

**WHERE** tableA.id **IN** (**SELECT** tableB.id **FROM** tableB **WHERE** title = 'Analyst');

*-- Returns 3 records - Kenny, Rob, and Greg*

For those not very familiar with SQL, this should be relatively easy to understand. We have written a **[subquery](http://en.wikipedia.org/wiki/Correlated_subquery)** that will get the *id* for the *Analyst* title in *tableB*. Using IN, we can then grab all of the employees from *tableA* who have that title.

While IN statements are fairly intuitive, they're often less efficient than the same query written as a JOIN or EXISTS statement would be.

To produce the same results as above, we can do the following:

*-- EXISTS*

**SELECT** \*

**FROM** tableA

**WHERE** **EXISTS** (**SELECT** 1 **FROM** tableB **WHERE** title = 'Analyst' **AND** tableA.id = tableB.id);

*-- JOIN (INNER is the default when only JOIN is specified)*

**SELECT** \*

**FROM** tableA

**JOIN** tableB

**ON** tableA.id = tableB.id

**WHERE** tableB.title = 'Analyst';

In most cases, EXISTS or JOIN will be much more efficient (and faster) than an IN statement. Why?

When using an IN combined with a subquery, the database must process *the entire subquery* first, then process the overall query as a whole, matching up based on the relationship specified for the IN.

With an EXISTS or a JOIN, the database will return true/false while checking the relationship specified. Unless the table in the subquery is *very* small, EXISTS or JOIN will perform much better than IN.

Furthermore, writing the query as a JOIN gives us some additional flexibility to easily return all of the employees if we'd like, or to even check for employees who do not have a title (orphan records).

*-- Return employees and display their title*

**SELECT** \*

**FROM** tableA

**JOIN** tableB

**ON** tableA.id = tableB.id;

*-- 1 Kenny 1 Analyst*

*-- 1 Rob 1 Analyst*

*-- 1 Greg 1 Analyst*

*-- 2 John 2 Sales*

*-- Which employees do not have a title?*

**SELECT** \*

**FROM** tableA

**LEFT** **JOIN** tableB

**ON** tableA.id = tableB.id

**WHERE** tableB.id **IS** **NULL**;

*-- 4 Molly NULL NULL*

In the first query above, Molly falls out because she does not have a title. If we would have liked her to appear in the record set, we could simply change the JOIN to a LEFT JOIN and she would appear with NULL data from *tableB*.

If you have many IN statements littered throughout your code, you should compare the performance of these queries against an EXISTS or JOIN version of the same query - you'll likely see performance gains.

I hope this illustrated some of the subtle differences between INs, EXISTS, and JOINs. Questions and feedback in the comments are appreciated.