Indexing and Performance Tuning
--------------------------------

The two most important elements of Performance Tuning are indexing and buffer size manipulation. We will work on indexing first.

Let us remember what an index does. Imagine a phone book and you want to find people by FIRST NAME. You would have to read the whole book line-by-line. Or, you create a list of first names, and then for every first name a pointer [e.g., see page 209, line 38] to where a person of that first name exists in the phone book. This list of first names and pointers is called an INDEX. If you have an unsorted, unindexed column, Oracle reads the data line-by-line. This is not a good idea for fast access.

How do I even know that I improved things?

One possible approach is to measure times of executions. The table people contains 320,000 rows.

```
SELECT TO_CHAR(SYSTIMESTAMP, 'SSSSS.FF') FROM DUAL
/
SELECT * from people
/
SELECT TO_CHAR(SYSTIMESTAMP, 'SSSSS.FF') FROM DUAL
/
```

The other possibility is that we use Show execution plan in AQUA. This will create a tab at the bottom. There you can see CPU cost.

But what does CPU cost mean?

[From Donald Burleson]
CPU_COST - The CPU cost of the operation as estimated by the cost-based SQL optimizer based on a secret algorithm. The value of this column does not have any particular unit of measurement; it is merely a weighted value used to compare costs of execution plans.

Side Comments:

You can see client statistics, by using the rightmost icon.

You can also see server statistics. Those are in the DBA Tools menu for Oracle. They are very technical and hard to read. And there are many of them.
Oracle does some caching, so the second run of the same select statement is likely to be faster.

Also note that sometimes an index actually makes things slower, because you have first a disk access to get the index and then a second disk access to get the data. Disk access is the slowest operation of all, the bottle neck.

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

The importance of good query design:

Consider the table containing the gender and age of 320,000 people. If you wanted to obtain all the males aged 34 and the table had an index on gender and an index on age there would be two possibilities.

* Use the gender index to first obtain all males and then go through the result set finding the 34 year olds.

We know there are 320,000 people in the table. If we used the index to find all males it is fair to say we would return about 160,000 rows. We would then have 160,000 rows which we have to search in order (one by one) for 34 year olds.

* Use the age index to first obtain all 34 year olds and then search the result set finding the males.

Assume people are between 0 and 100. These will not be equally distributed. There are many more people aged 34 than people aged 99. Now VERY approximately we would return about 2% of the rows (about 6400) which we have to search in order (one by one) for males.

So, the second approach is obviously much better. Use the index that will give you the smallest result set.

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

Surprise: Oracle does not HAVE TO use an index, even if you create one. We have to force the system to use index structures.

You can give "hints" to the optimizer. Hints look like comments INSIDE THE select statement, starting with a + sign. The FULL hint requires a full table scan, as opposed to using an index.
These are runtimes with fresh copies of Aqua every time.

==========================================
Bad design, no index.

    SELECT  *
    FROM    (  
        SELECT  *
        FROM    PEOPLE
        WHERE   GENDER = 'M'
    )  
    WHERE   AGE = 34
/

SELECT STATEMENT 666.0 663 92775052 2618 60214
666     ALL_ROWS
TABLE ACCESS (FULL) 666.0 663 92775052 2618 60214
1 GELLER PEOPLE FULL TABLE ANALYZED 2

CPU cost: 92,775,052

==========================================
Good design, no index.

    SELECT  *
    FROM    (  
        SELECT  *
        FROM    PEOPLE
        WHERE   AGE = 34
    )  
    WHERE   GENDER = 'M'
/

SELECT STATEMENT 666.0 663 92775052 2618 60214
666     ALL_ROWS
TABLE ACCESS (FULL) 666.0 663 92775052 2618 60214
1 GELLER PEOPLE FULL TABLE ANALYZED 2

CPU cost: 92,775,052

==========================================
create index forgender on PEOPLE(GENDER)
/

Bad design with index.

    SELECT  *
FROM (  
SELECT /*+ INDEX( PEOPLE, FORGENDER ) */ *  
FROM PEOPLE  
WHERE GENDER = 'M'  
)  
WHERE AGE = 34  
/

SELECT STATEMENT 1616.0 1613 86632454 2618 60214  
1616     ALL_ROWS  
TABLE ACCESS (BY INDEX ROWID) 1616.0 1613 86632454 2618 60214 1 GELLER PEOPLE BY INDEX ROWID TABLE ANALYZED  
2  
INDEX (RANGE SCAN) 300.0 299 34801311 163359 1 GELLER FORGENDER RANGE SCAN INDEX ANALYZED  
1  
CPU cost: 86,632,454 + 34,801,311  
drop index forgender  
/

==========================================  
create index forage on PEOPLE(AGE)  
/

Good design with index.  

SELECT *  
FROM (  
SELECT /*+ INDEX( PEOPLE, FORAGE ) */ *  
FROM PEOPLE  
WHERE AGE = 34  
)  
WHERE GENDER = 'M'  
/

SELECT STATEMENT 1279.0 1279 11484191 2618 60214  
1279     ALL_ROWS  
TABLE ACCESS (BY INDEX ROWID) 1279.0 1279 11484191 2618 60214 1 GELLER PEOPLE BY INDEX ROWID TABLE ANALYZED  
2  
INDEX (RANGE SCAN) 13.0 13 1125579 5164 1 GELLER FORAGE RANGE SCAN INDEX ANALYZED  
1
CPU cost: 11,484,191 + 1,125,579

drop index forage
/

Conclusion:

Good Design with Index has a CPU cost that is 8 times smaller.

Buffer Tuning:

We have to remember that all Oracle instances are generally constrained by I/O operations. I/O is the single most dramatic component of Oracle performance, and any initialization parameters that can be used in order to reduce I/O will improve the performance of the database.

Naively speaking, you want the biggest possible buffers, so that you have to go to the disk very rarely.

However,.... if you have fixed RAM and big buffers, that means that you limit the number of buffers that you have. If you have an application that jumps around a lot in the table, getting one line from the top, one line from the bottom, one line from the middle, etc. you will soon have filled up all buffers. At the next request for which the data is NOT in any of the buffers, you need to read a whole new buffer. Now, reading a small buffer would be faster.

Example:
1 big buffer only.
Program accesses very large table as follows:

1st row,
last row
2nd row,
last-1 row
3rd row,
last-2 row etc.

EVERY TIME the one buffer will be overwritten.
That is very bad, very slow.

Now let's try a better approach:
Two small buffers, one loaded with the beginning of the table and one loaded
with the end of the table would be MUCH better in this case.

Another example:
If you have an access patterns like this:

Row Number
1
10001
50001
100001
2
10002
50002
100002
3
10003
50003
100003
.... more lines numbers like this

MANY MORE SUCH JUMPS

you would be better off to have FOUR small buffers.

Buffer 1: Starting at 1
Buffer 2: Starting at 10001
Buffer 3: Starting at 50001
Buffer 4: Starting at 100001

So, you need to look for "unusual" query patterns which might suggest to use smaller buffer sizes.

Again, Oracle helps you by keeping track of a lot of "measures" and letting you access them. For example:

Buffer hit ratio.
This metric is a measure of the effectiveness of the Oracle data block buffer. The higher the buffer hit ratio, the more frequently Oracle found a data block in memory and avoided an I/O.

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

Setable Parameters (not for US, only DBA):
[From Burleson]

With many hundreds of Oracle initialization parameters, it's important for the database administrator (DBA) to focus on those parameters that have the most impact on the performance of the Oracle database. Oracle has a huge number of parameters in the Oracle initialization files (init.ora) that control the overall configuration of the Oracle instance. While there are dozens of init.ora
parameters that affect performance, these are some of the most important:

* **shared_pool_size**  
The memory region allocated for the library cache and internal control structures

* **db_block_size**  
The size of each data block

* **db_block_buffers**  
The number of data buffers to allocate for the instance

* **sort_area_size**  
The amount of RAM reserved for each user to perform sorting operations

---

Material:

Basic Intro to Index:  
http://www.techonthenet.com/oracle/indexes.php

Tuning:  
http://www.billmagee.co.uk/oracle/sqltune/index.html

Indexing:  
http://www.dba-oracle.com/art_9i_indexing.htm

Statement Caching:  
http://www.adp-gmbh.ch/ora/concepts/library_cache.html

Buffer Tuning:  
http://www.dba-oracle.com/art_tuning_instance.htm

Bill Magee's  
http://www.billmagee.co.uk/oracle/sqltune/index.html

Holowczak also have a very good page at  
Zicklin School of Business  
http://cisnet.baruch.cuny.edu