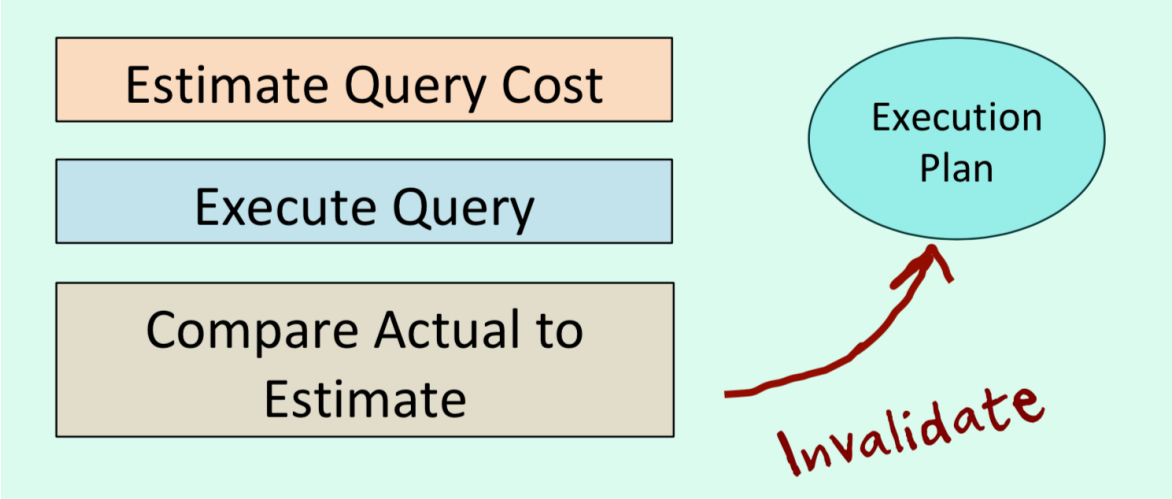
Lecture15

What can the optimizer do to improve stability?

- Feed back loop



    - One problem of course is this is a reactive, and not proactive, plan improvement. You must have a painful execution to learn about the problem.
- Adaptive Cursor Sharing
    - Oracle has introduced a mechanism in version 11. If a parameter is bound **as a search criterion**and there is a **histogram that indicates possible problems**, a query is checked. If a problem occurs, the plan will be re- evaluated when parameters change.
- SQL Server **"Recompile"** directive in SQL statement
    - SQL Server has a slightly cruder mechanism, a directive that you can add to a problem query that asks for it to be recompiled at each execution.
- Dynamic sampling
    - An interesting, and often effective, feature in Oracle is "dynamic sampling" which basically asks the optimizer to guess less and check data a bit more.

Correctly Writing a SQL Query

What is really important is what determines the magnitude of **the number of rows returned**by the query

- go for the core
- Remove what doesn't shape the result set
    - Joins to tables without any col = constant condition
    - If there is a foreign key, probably not: you know that for every row you'll find a match in the other table, so the join won't affect the number of rows returned.

Classifying Tables

```
select
distinct
cons.id, coalesce(cons.definite_code,
cons.provisional_code) dc, cons.name,
cons.supplier_code,
   cons.registration_date from weighing w
inner join production prod on prod.id = w.id
inner join process_status prst
on prst.prst_id = prod.prst_id
left outer join composition comp
on comp.formula_id = w.formula_id
inner join constituent cons on cons.id = w.id
where prod.usr_id = :userid
and prst.prst_code = 'PENDING'
```

- Tables from which data is returned with or without conditions
    - Keep as is if it belongs to the core
- Tables from which NO data is returned Conditions only
    - Turn into a subquery
- Glue Tables Join other tables
    - Useless at the end of a chain
    - Main query or subquery

Tips

- Group and sort as little as possible
- Join late
    - Keep for the very end (the top level, the outer query) all joins that change nothing (or very little) to the size of the result set.
- Loops kill performance
    - Looping for writing to a file or sending data over a network, or in a procedural language that accesses the database is perfectly justified, because you are at the border between a world that knows sets and a world that (in the best of cases) only knows collections.