# ORACLE OPTIMIZER

Created by Jongwon

## Rule Based Optimizer(RBO)

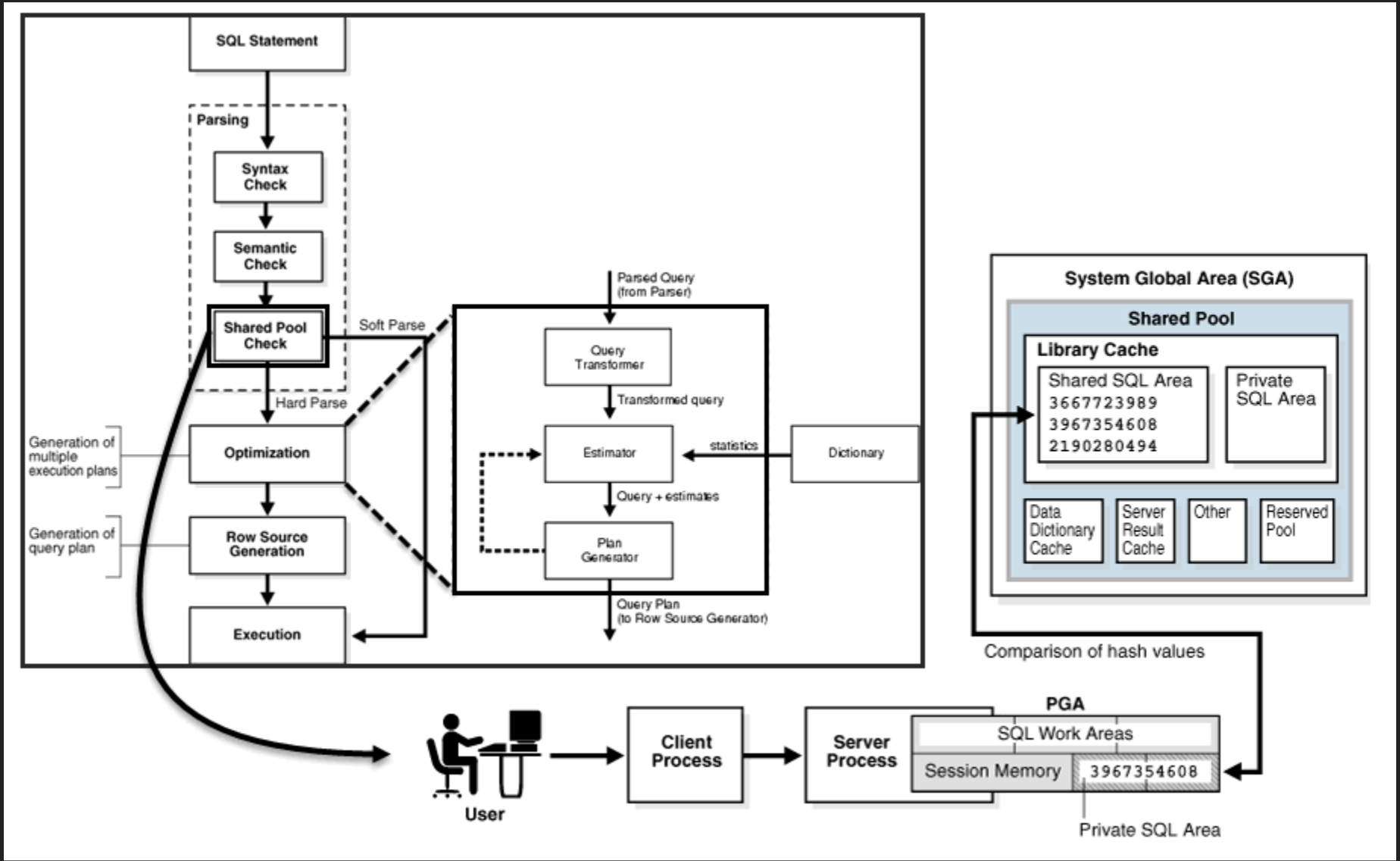
정해진 규칙에 따라 Execution Plan을 도출
Execution Plan이 거의 변하지 않는다.

---

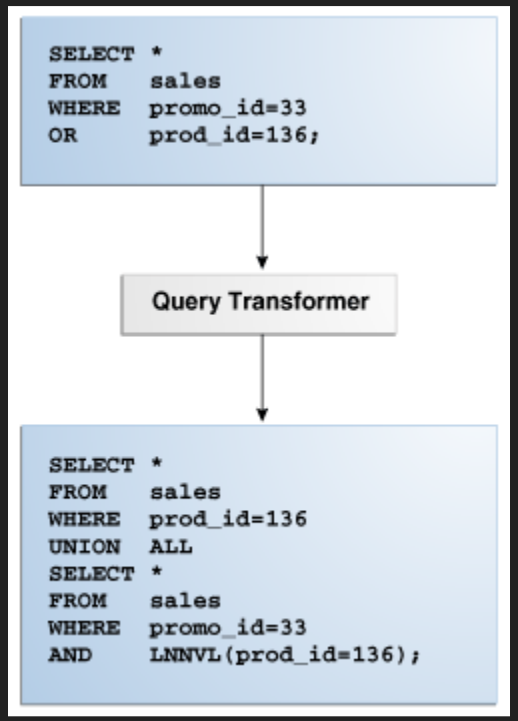
## Cost Based Optimizer(CBO)

Optimizer가 Cost를 계산하여 Execution Plan을 도출
시스템 상태 및 통계자료에 따라 Execution Plan이 변한다.

# DIAGRAMMATIC

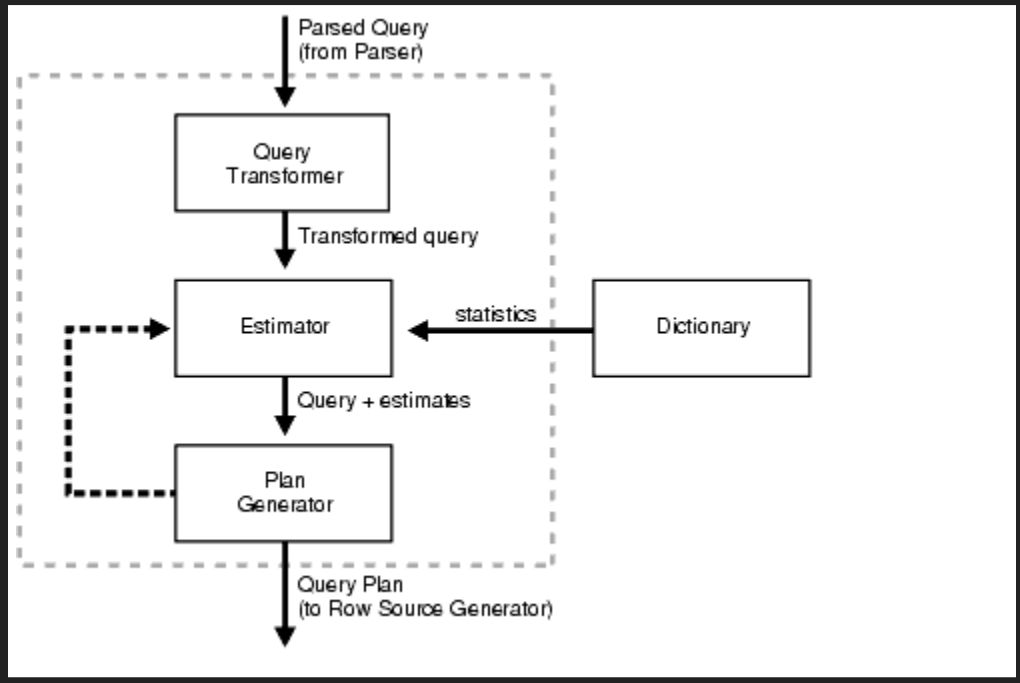


# QUERY TRANSFORMER




SQL문을 다양한 형태로 변형하여
선택 가능한 Execution Plan을 늘린다.

# ESTIMATOR




Cost를 **계산**한다.

- Selectivity
- Cardinality
- Cost(CPU, I/O)

# COST BASED OPTIMIZER

## CPU+I/O cost model(9i 기준)

Cost = (#SRds * sreadtim + #MRds * mreadtim + #CPUCycles / cpuspeed )

/ sreadtim

- sreadtim : single block을 읽는 평균 응답시간
- mreadtim : multi block을 읽는 평균 응답시간
- cpuspeed : 1초당 평균 사이클 수(오라클에서 정의)
- #SRds : single block을 읽은 수
- #MRds : multi block을 읽은 수

# EXAMPLE

```
SELECT prod_category, AVG(amount_sold)
FROM sales s, products p
WHERE p.prod_id = s.prod_id
GROUP BY prod_category;

Execution Plan
----------------------------------------------------------
Plan hash value: 1197568639

-------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name                 | Rows  | Bytes | Cost (%CPU)| Time     | P
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                      |     5 |   255 |   551   (6)| 00:00:07 |
|   1 |  HASH GROUP BY           |                      |     5 |   255 |   551   (6)| 00:00:07 |
|*  2 |   HASH JOIN              |                      |    72 |  3672 |   550   (6)| 00:00:07 |
|   3 |    VIEW                  | VW_GBC_5             |    72 |  2160 |   548   (6)| 00:00:07 |
|   4 |     HASH GROUP BY        |                      |    72 |   648 |   548   (6)| 00:00:07 |
|   5 |      PARTITION RANGE ALL |                      |   918K|  8075K|   525   (2)| 00:00:07 |
|   6 |       TABLE ACCESS FULL  | SALES                |   918K|  8075K|   525   (2)| 00:00:07 |
|   7 |    VIEW                  | index$_join$_002     |    72 |  1512 |     2   (0)| 00:00:01 |
|*  8 |     HASH JOIN            |                      |       |       |            |          |
|   9 |      INDEX FAST FULL SCAN| PRODUCTS_PK          |    72 |  1512 |     1   (0)| 00:00:01 |
|  10 |      INDEX FAST FULL SCAN| PRODUCTS_PROD_CAT_IX |    72 |  1512 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------
```

# OPTIMIZER STATISTICS

- Table Statistics
  - Row 수
  - Block 수
  - 평균 row 길이
- Column Statistics
  - 컬럼 내의 NDV(Number of Distinct Value)
  - 컬럼 내의 NULL수
  - 데이터 분포(Histogram)
- Index Statistics
  - Leaf Block 수
  - Tree의 Level
  - Index Clustering factor
- System statistics
  - I/O 성능
  - CPU 성능

# PARAMETERS

```
SELECT DISTINCT name FROM v$sql_optimizer_env
```

TEST 했던 Execution Plan이 재현되지 않는다
PARAMETER값이 다른지 우선 확인해 볼 필요가 있다.

# OPTIMIZER_MODE

## 최적화의 방향성을 지정

- 응답 중시
  - FIRST_ROW_n
  - 최초 몇 행을 반환하기까지의 시간을 최적화
- 처리량 중시
  - ALL_ROWS
  - 마지막 행을 반환하기까지의 시간을 최적화

# CACHE HIT

```
-- 기본값
OPTIMIZER_INDEX_CACHING = 0
OPTIMIZER_INDEX_COST_ADJ = 100

-- OLTP 환경
OPTIMIZER_INDEX_CACHING = 90
OPTIMIZER_INDEX_COST_ADJ = 25
```

---

- OPTIMIZER_INDEX_CACHING : 인덱스 블록이 몇 %정도 CACHE HIT 할 것인가
- OPTIMIZER_INDEX_COST_ADJ : 인덱스를 액세스하는 비용을 몇 %정도 계산할까

# HISTOGRAM

Optimizer의 계산 예외를 **예측**하기 위한 통계자료

- 빈도 분포 histogram
  - NDV(number of distinct value)
  - 각 값이 몇건이나 있는지 정확한 빈도를 보기 위해
- 높이 균형(Height Balanced)
  - 편중된 값을 검출하기 위해 사용

# HINT

- FULL
- INDEX, INDEX_SS, INDEX_FFS
- USE_NL, USE_HASH, USE_MERGE
- MERGE_AJ, HASH_AJ, MERGE_SJ, HASH_SJ
- UNNEST, NO_UNNEST

Execution Plan을 **고정**할 필요가 있을 때

# SUMMARY

- Diagrammatic
  - Query Transformer
  - Estimator
  - CBO Cost Model
  - Optimizer Statistics
- Parameters
  - OPTIMIZER_MODE
  - CACHE HIT
- Histogram
- Hint

# REFERENCE


- Query Optimizer Concepts
- Oracle cpu_cost & Oracle io_cost SQL optimizer Features
- 오다 케이지 외 6명.『 오라클 실무 테크닉 』Jpub
- Jonathan Lewis.『 비용기반의 오라클 원리 』비투엔컨설팅 역