Datenbankadministration

13. DB2 Advisor

AG DBIS University of Kaiserslautern, Germany

Karsten Schmidt kschmidt@informatik.uni-kl.de
(Vorlage TU-Dresden)

Wintersemester 2008/2009

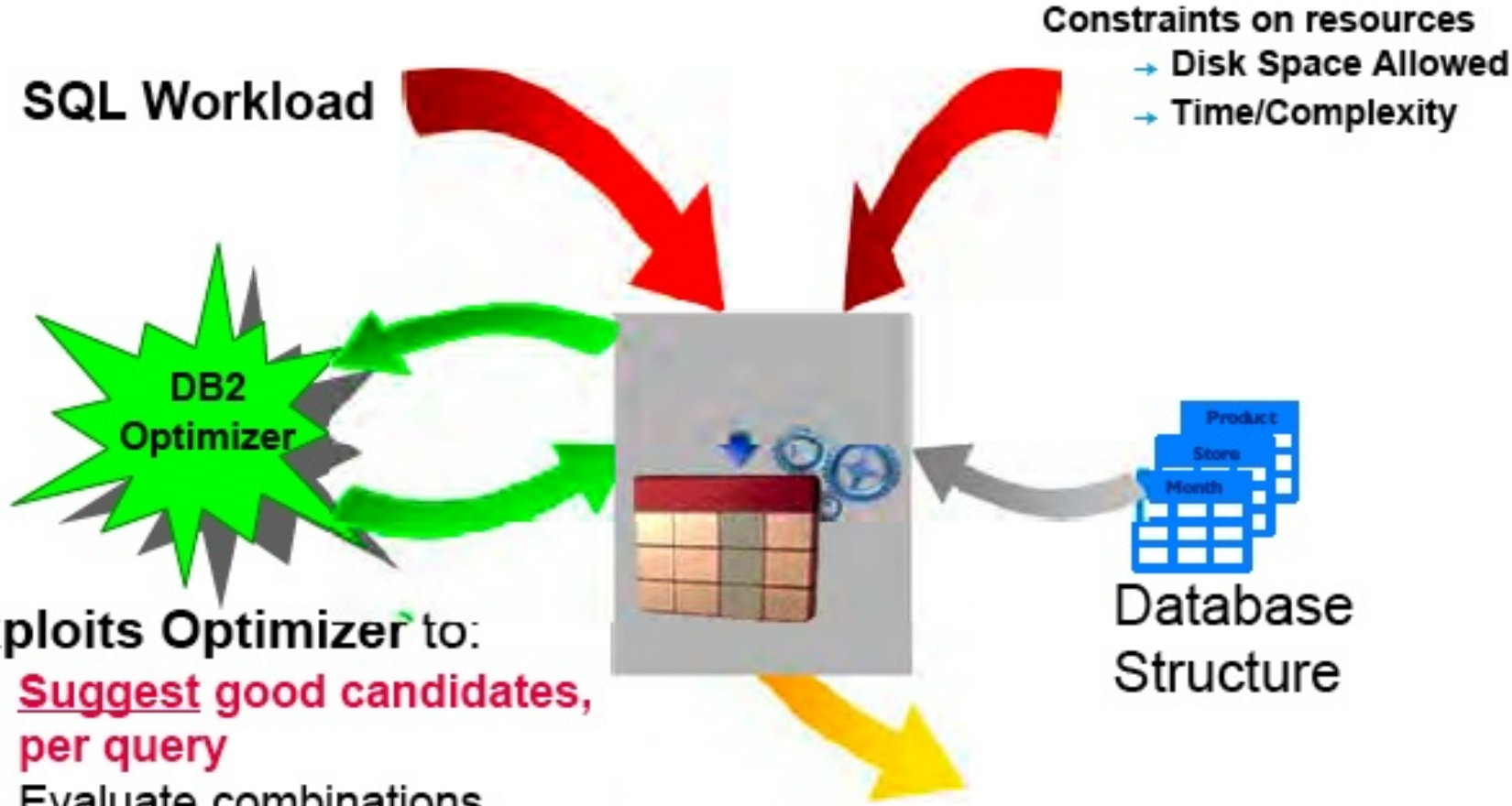
Advisor

- **Problemdefinition:**
  - Gegeben:
    - DB-Schema mit Daten
    - Menge mit SQL-Anfragen (=Workload)
  - Fragen:
    - Welche Indizes, MatViews, etc. sollen angelegt werden?
    - Wie müssen die DB-Parameter gesetzt werden?
    - Ziel: optimale Datenbank
  - Antwort:
    - DB2 Advisor (db2advis)

Index Selection: The Problem

- Huge number of possible indexes
  - Dependent upon workload (queries) anticipated
  - For each query, user has to trade off:
    - Benefits:
      - ✓ Apply predicates efficiently (save reading entire table)
      - ✓ Provide a row ordering needed by query for certain operations
      - ✓ Index-only access (avoid fetching data pages)
      - ✓ Enforce uniqueness (e.g., primary keys)
    - Costs:
      - Storage space
      - Updating
      - More plans for the optimizer to evaluate
- Time-consuming trial & error process to choose the best set of indexes
  - Create index (system sorts entire table on key of the index)
  - Collect statistics on it (system scans entire table AND all indexes)
  - Re-optimize all queries in all apps that might benefit
  - See if
    - Index was used
    - Performance improves
  - Iterate!

Solution(1): DB2 Index Advisor (V6, 1999)



Index Advisor (DB2 V6) – The Math

- Variant of well-known "Knapsack" Problem
- Greedy "bang-for-buck" solution is optimal, when integrality of objects (indexes) is relaxed

- For each query Q:
  - Baseline: Explain each query w/ existing indexes, to get cost E(Q)
  - Unconstrained: Explain each query in RECOMMEND INDEXES mode, to get cost U(Q)
  - Improvement ("benefit") B(Q) = E(Q) - U(Q)
- For each index I used by one or more queries:
  - If query Q used index I, assign "benefit" B(Q) to index I:

    B(I) = B(I) + B(Q)
  - Assign "cost" C(I) = size of index in bytes
  - Order indexes by decreasing B(I) / C(I) ("bang for buck")
  - Cut off where cumulative C(I) exceeds disk budget
- Iterative improvement: exchange handfuls of "winners" with "losers"

- **REFN: "DB2 Advisor: An Optimizer Smart Enough to Recommend its Own Indexes", ICDE 2000 (San Diego), Valentin, Zuliani, Zilio, Lohman, et al.**

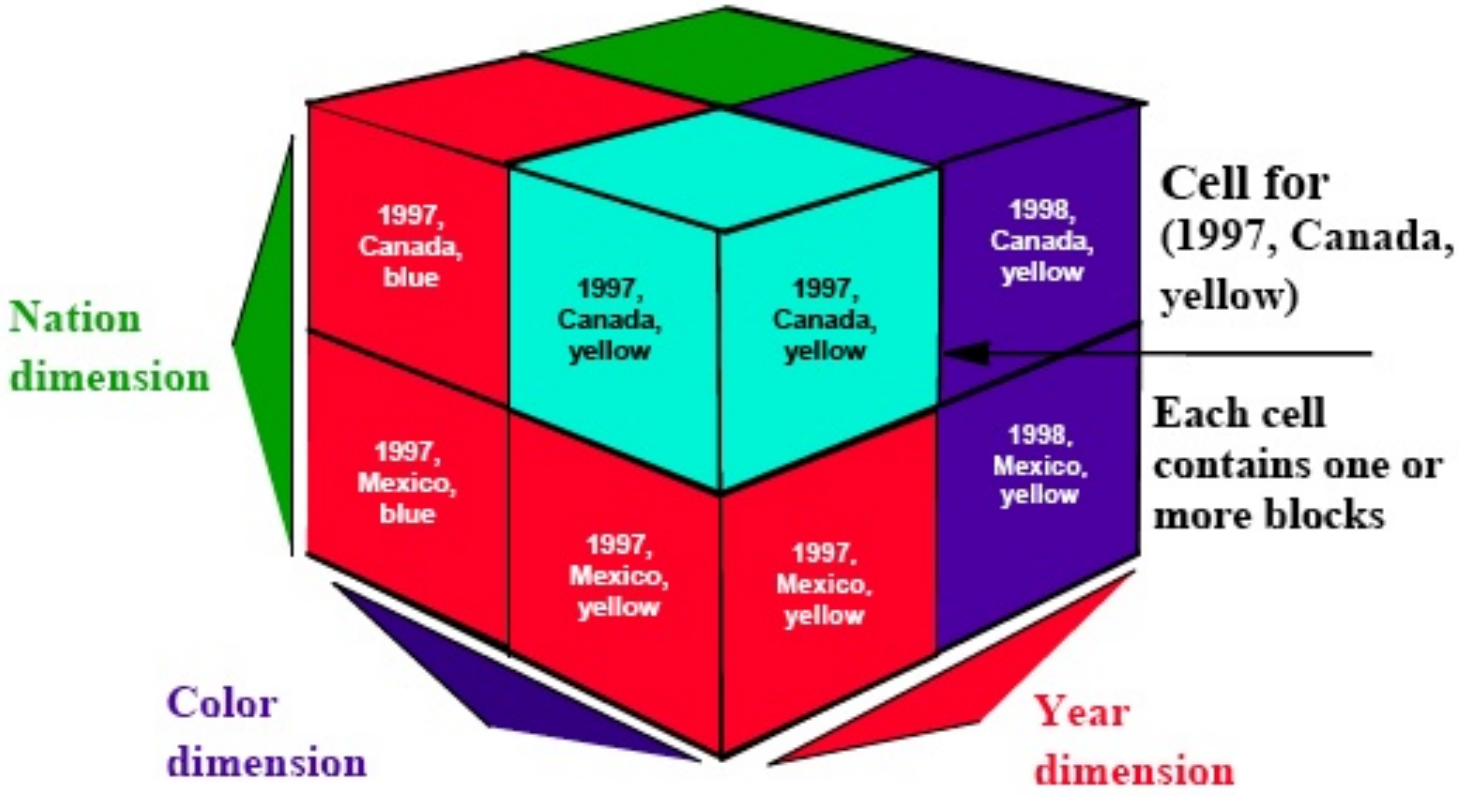
Design Advisor ("Stinger")

- An extension of existing Index Advisor (V6)
- Headquarters for all physical database design
- Recommends any combination of:
  - ✓**Indexes**
  - ✓**Materialized Views (Materialized Query Tables (MQTs))**
    - ➢**Called Automatic Summary Tables (ASTs) before V8.1**
  - ✓ **Partitioning of tables (in partitioned environment)**
  - ✓ **Multi-Dimensional Clustering (MDC) storage method (New in V8.1)**
- Takes interactions of these into consideration
- Status:
  - ✓ **Coming soon ("Stinger")!**
  - ✓ **Beta testing on customer databases now!**

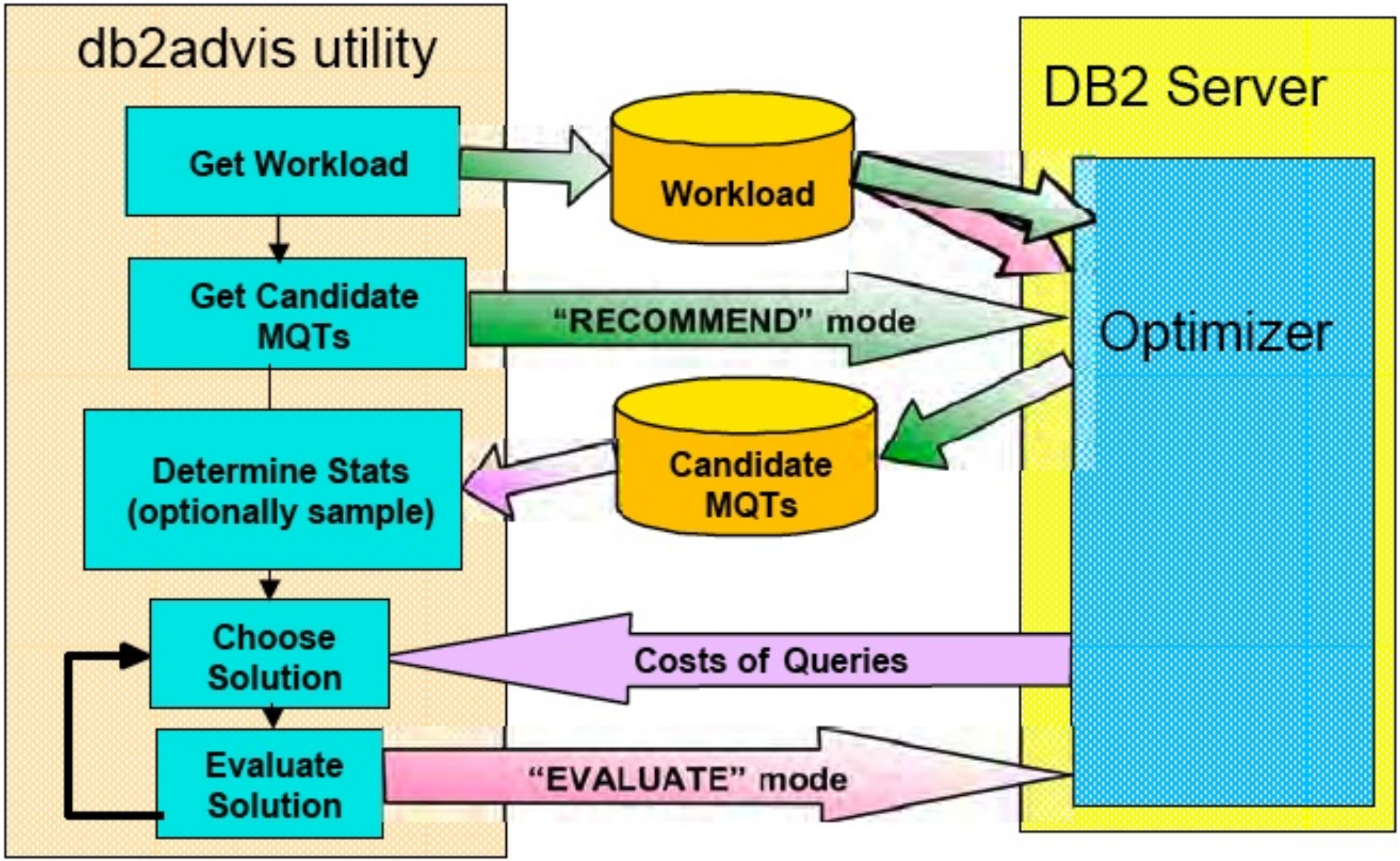
- **REFNS:**
  - "DB2 Design Advisor: Integrated Automatic Physical Database Design", **VLDB 2004**
  - "Recommending Materialized Views and Indexes with IBM's DB2 Design Advisor", **IEEE Intl. Conf. on Autonomic Computing (ICAC 2004)**
  - "Trends in Automating Database Physical Design", **IEEE 2003 Workshop on Autonomic Computing Principles and Architectures**, Banff, Alberta, August 2003

Multi-Dimensional Clustering (MDC) – V8.1

Cells are the portion of the table containing data having a unique set of dimension values; the intersection formed by taking a slice from each dimension.
Blocks are the storage units that compose each cell.



Design Advisor Architecture (MQTs only)



Design Advisor: Partition Advisor

- **Scope:**
  - DB2 "partitioned environment" (was called EEE prior to V8.1)
  - "Shared-nothing" parallelism
  - Data stored horizontally partitioned
    - In a partition group, spread across specified partitions
    - Based upon hashing of partitioning column(s)
    - May be replicated across all partitions of partition group

  - Need to co-locate similar values for joins, aggregation in queries
  - Partitioning required for a given table may be different
    - Between queries
    - Even within a query (joined on different columns)!

- **Problem:** What is optimal partitioning for each table, given:
  - Workload of queries
  - Schema, including set of partition groups & tablespaces
  - Statistics on database

Reference: "Automating Physical Database Design in a Parallel Database", ACM SIGMOD 2002 (Madison, WI, June 2002)

db2advis

- **Anwendung des DB2-Advisors**
  - Kommandozeilen-Tool
  - Nutzung

    ```
    db2advis -d <datenbank> -n <schema> -i <infile> -o <outfile>
    ```
  - - Infile
    - Menge von SQL-Anfrage
    - Häufigkeit einer Anfrage kann angegeben werden
    - `--#SET FREQUENCY x` (vor der SQL-Anfrage)
  - - Outfile
    - DDL-Skript
    - Index und MatView-Defintion

db2advis (2)

- **Beispiel (Workload.sql)**

```
--#SET FREQUENCY 50
SELECT COUNT(*) FROM tpch.region;
--#SET FREQUENCY 10
SELECT l_shipdate,SUM(l_quantity)
FROM tpch.lineitem
GROUP BY l_shipdate;
```

```
db2advis -d tpch -n db2inst1 -i Workload.sql -o index.sql
```

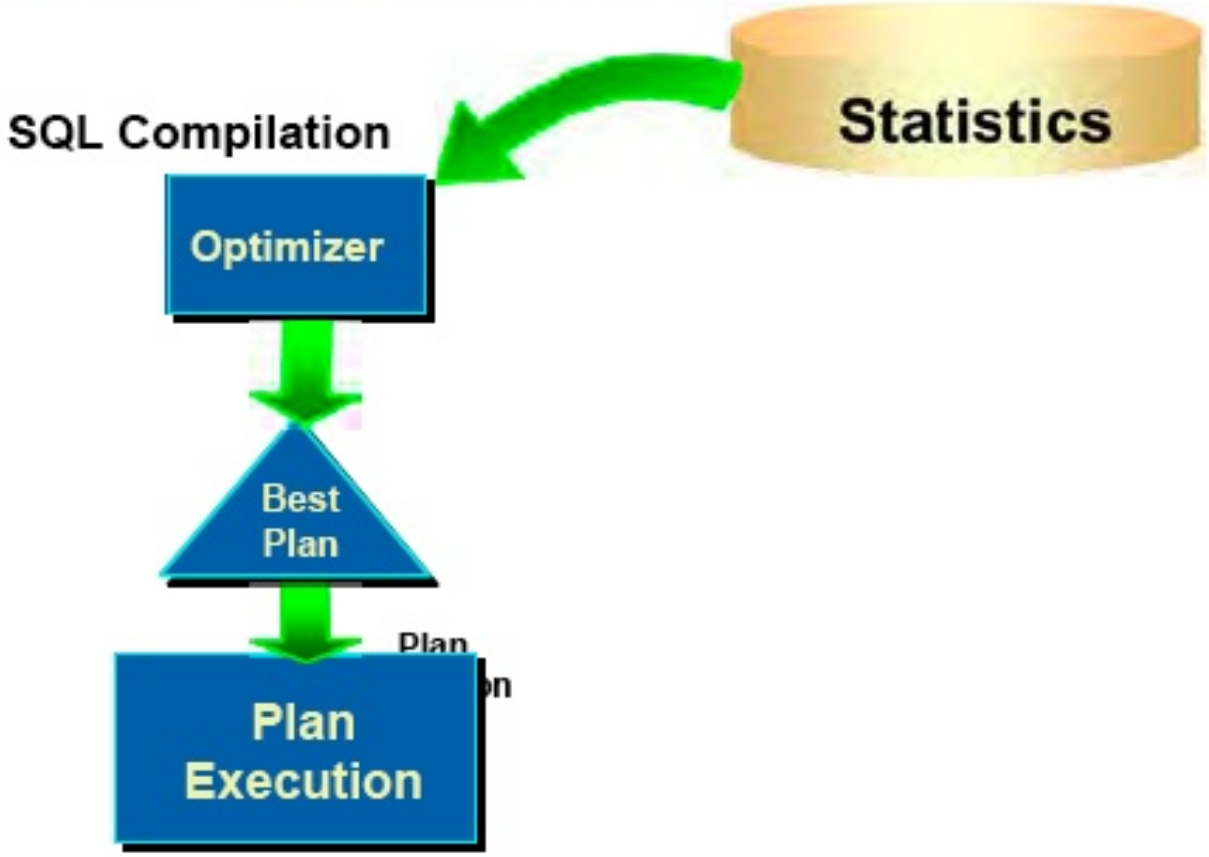
Zusätzliche Parameter

- **Advisor-Type (`-m`)**
  - I (default): nur Indizes
  - M: MatViews und Indizes
  - C: MDC
  - P: Partitionierung
- **Speicherplatzbegrenzung (`-l <disk_limit>`)**
- **Zeitbegrenzung (`-t <max_advisor_time>`)**
- **Workload**
  - SQL-File
  - bzw. über Monitor (dynamic sql monitor)

Automating Statistics Collection:

- **Problem:**
  - Optimizer requires that statistics on database be
    - Up to date (after updates)
    - Complete (multi-column)
  - User must invoke RUNSTATS
- **Solution**: Automate RUNSTATS
  - ***Invocation*** scheduled and prioritized
  - ***Run silently*** as a background daemon
    - Throttled based upon workload
  - **LEO** the **LE**arning **O**ptimizer determines which ***statistics needed***
    - Based upon learning from past queries
    - Groups of columns
      - Enables correlation detection
    - Communicated to RUNSTATS via statistical "profiles"
- Shipping in DB2 "Stinger"
- **Refn**: "Automated Statistics Collection in DB2 Stinger", **VLDB 2004**

Query Optimization -- Today



EXPLAIN gives Optimizer's Estimates

1. Monitor

New: Capture Actual Number of Rows!

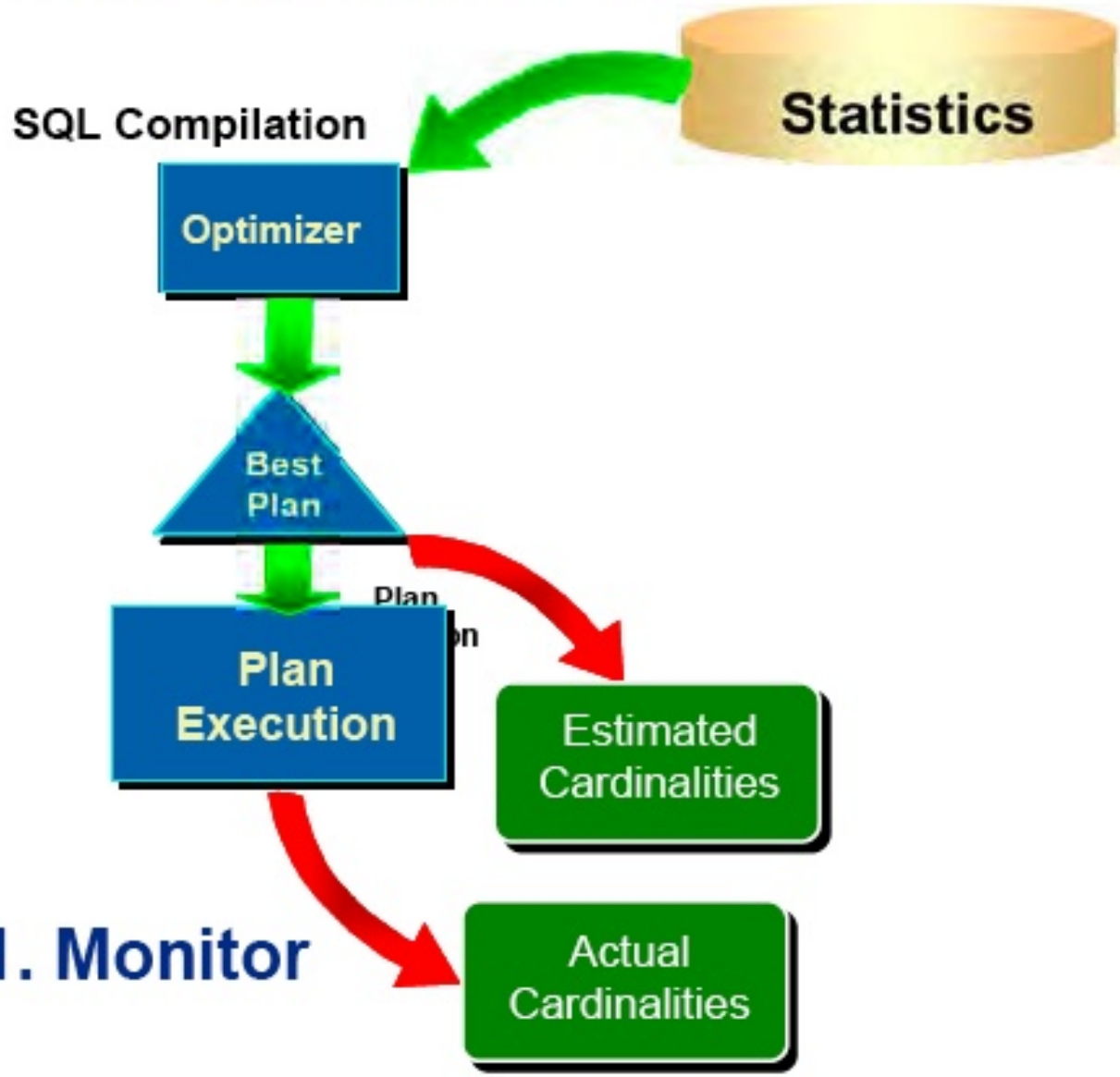
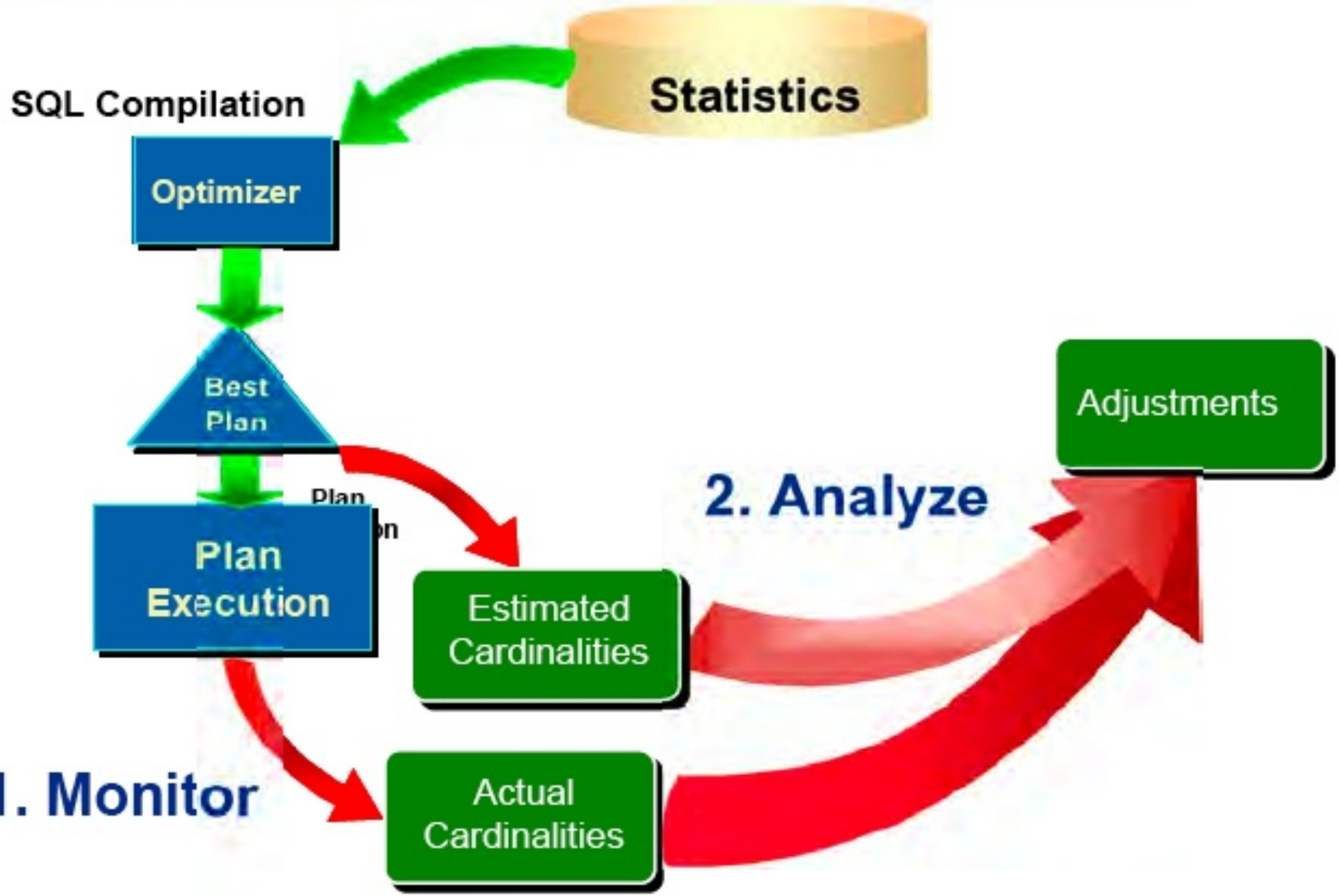
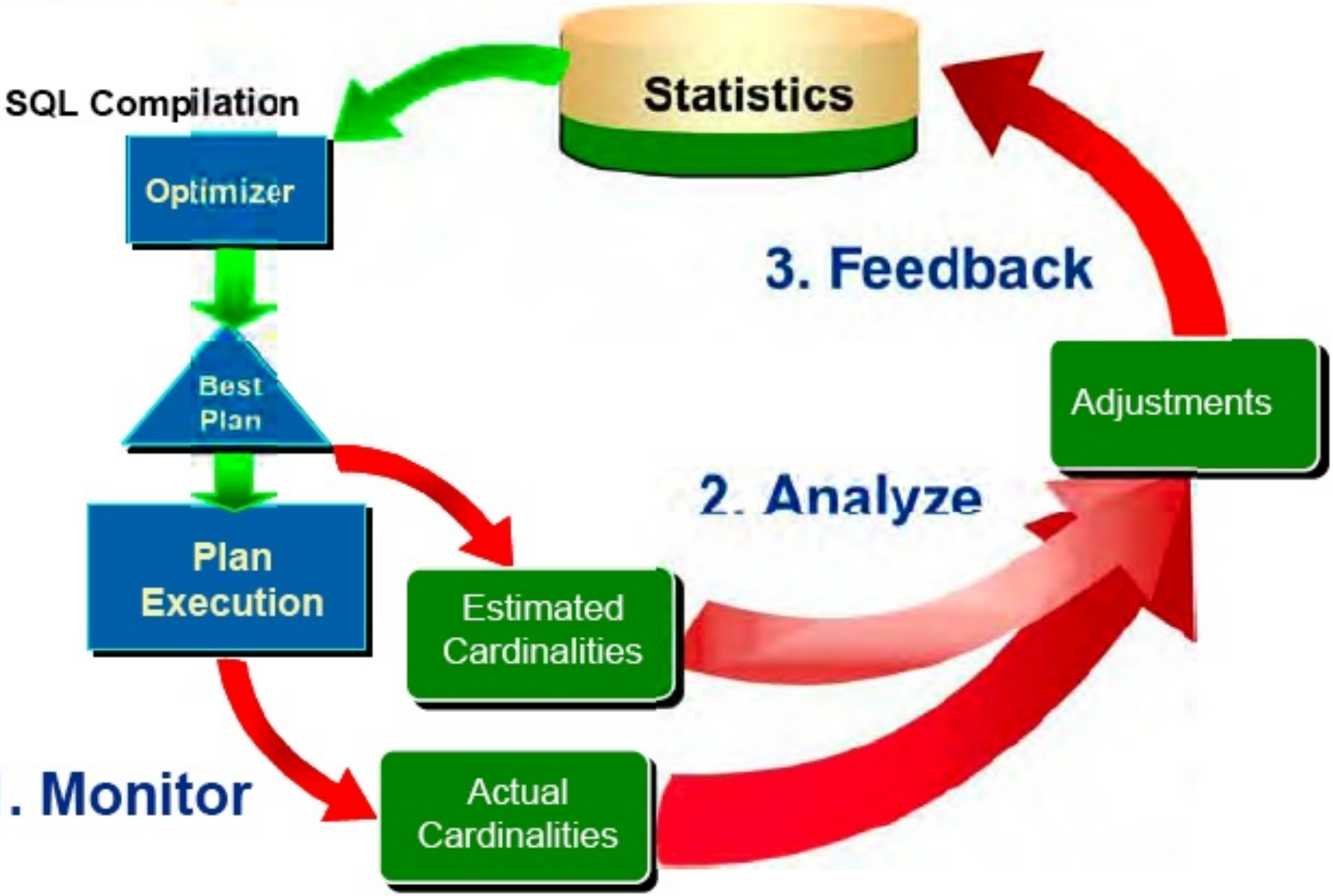


Figure Out Where the Differences Are



Augment Statistics with Adjustments



Exploit: Learning in Query Optimization!