



# DB Performance Tuning Roadmap

홍미희(mhhong@kr.ibm.com)  
MTS, IBM Korea



# 1. Tuning 과정 ( 문제 파악 )

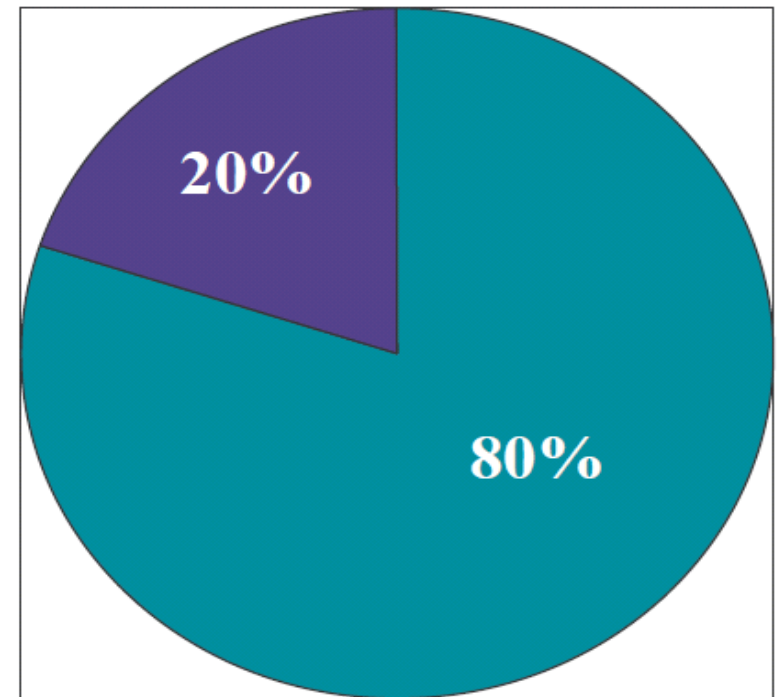
---

- Application
  - SQL
  - Host Language Code 들
- Database
  - Index 들
  - Database 와 Index 구성
  - Database 설계 ( normalization / denormalization )
- DB2 Subsystem
  - ZPARM, Pool, Locking, IRLM, DDF, 등
- 환경 요소
  - Network
  - TP Monitor ( CICS, IMS/TM )
  - Operating System



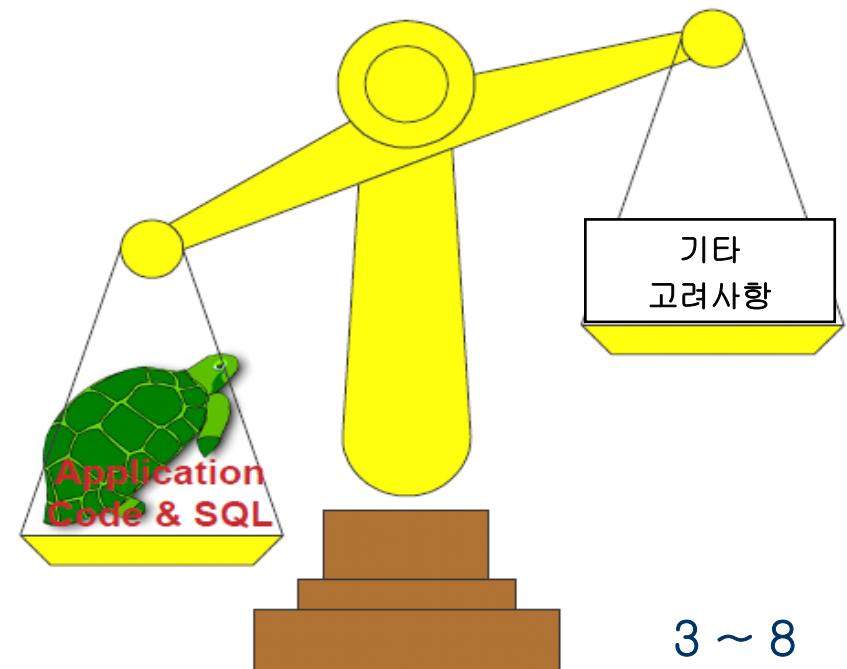
## 2. 기본 Tuning Rule 들

- 20 % 의 Tuning 노력으로  
80 % 의 Tuning 효과를 얻음
  - 20% 의 DB2 application 으로 인해  
80% 의 문제가 발생한다는 의미
- 한 번에 하나씩 Tuning 수행
  - 개별 action 이 도움이 되는지      확인  
가능
- Tuning 시 최적화 (Optimize) 대상
  - CPU, I/O 와 Concurrency



## . Application Code 와 SQL

- 대부분의 RDB Expert 들은 Performance 문제의 대부분이 RDB 사용 Application 들의 Coding 수준이 낮거나 잘못된 SQL coding 에 기인한다고 동의함
  - 문제 원인 중 70% ~ 80%

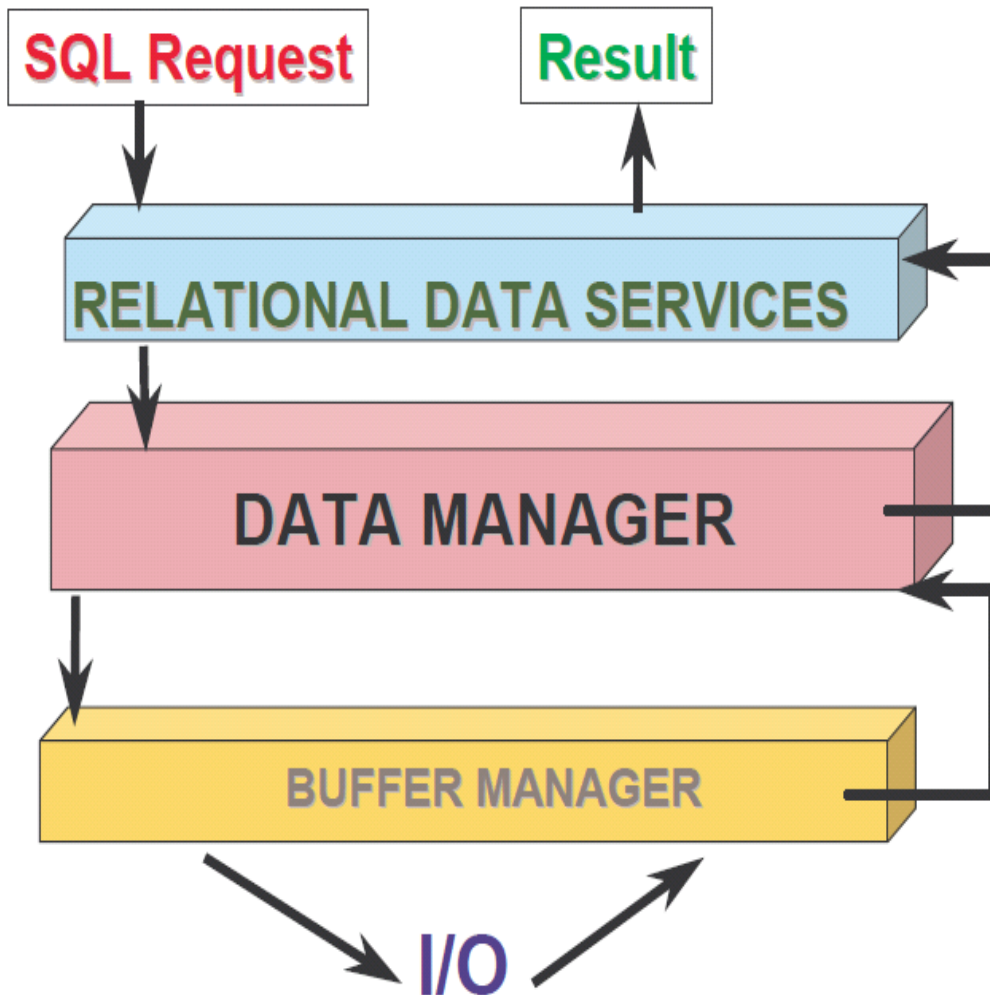


### 3. Application Tuning : SQL

---

- Coding 은 간단 하게 : Complex SQL 이 효율적 인 경우도 !
- 일반적으로는 SQL 로 수행 시키도록, Program 은 이후에 고려
- 필요한 data 의 최소 row 건수를 검색하도록
- 필요한 column 들만 검색 하도록
- 되도록 Join 조건은 사용하지 않도록
- Stage 1 과 Indexable 조건을 우선 사용
  - Host variable data type 과 column data type 이 match 되도록
- 대용량 table 에 대한 tablespace scan 이 발생하지 않도록
- 되도록 sort 가 발생하지 않도록
  - Order by 와 Group by 는 Index 지정
  - Distinct 사용 주의
  - UNION ALL 과 UNION 구분

### 3. Application Tuning : Stage 1 and Stage 2



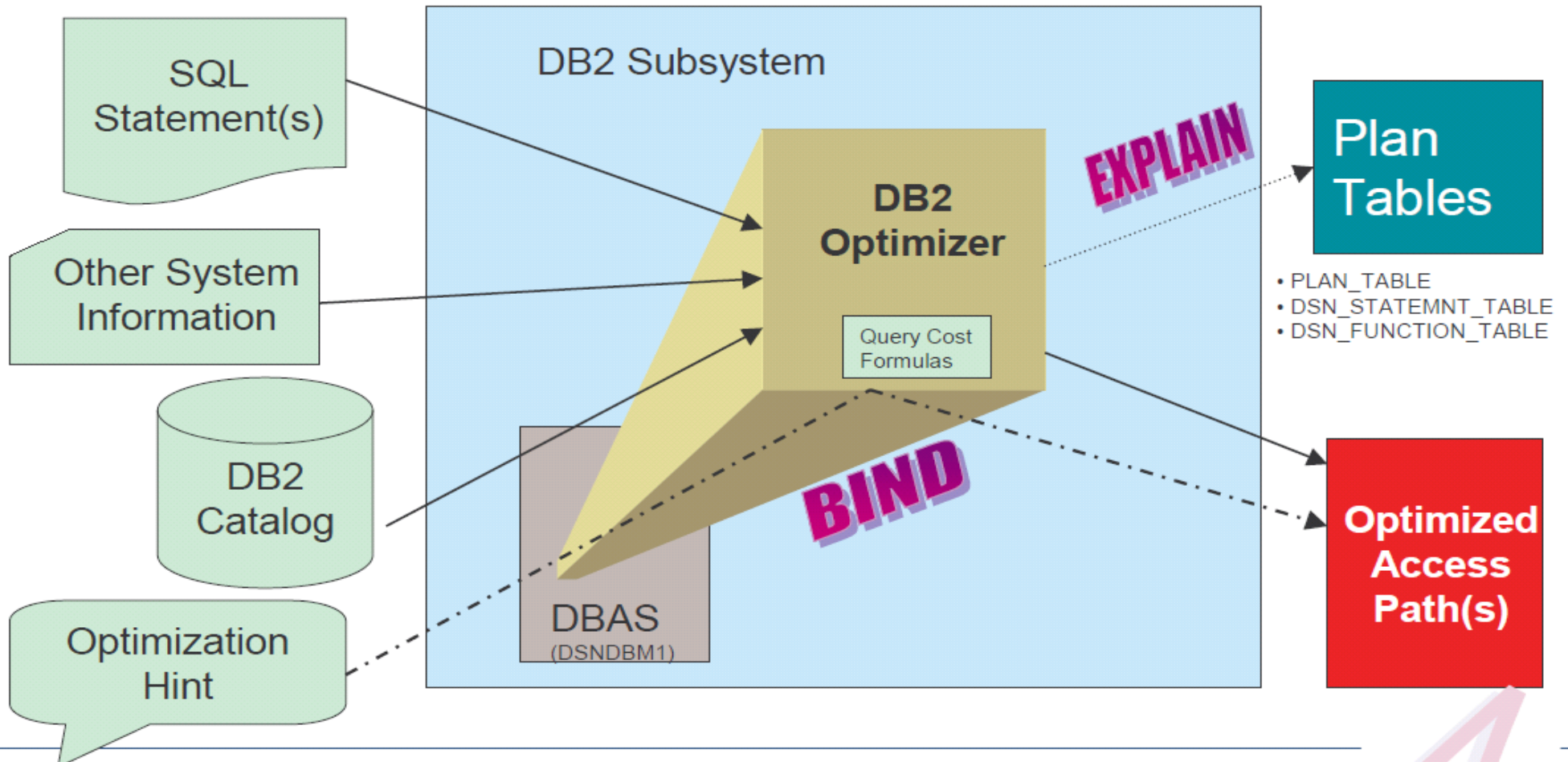
#### STAGE 2

- Data 검색 후 RDS ( Real Data System ) 가 조건 점검 ( non-sargable )
- Data Manager 보다 고가의 비용을 지불

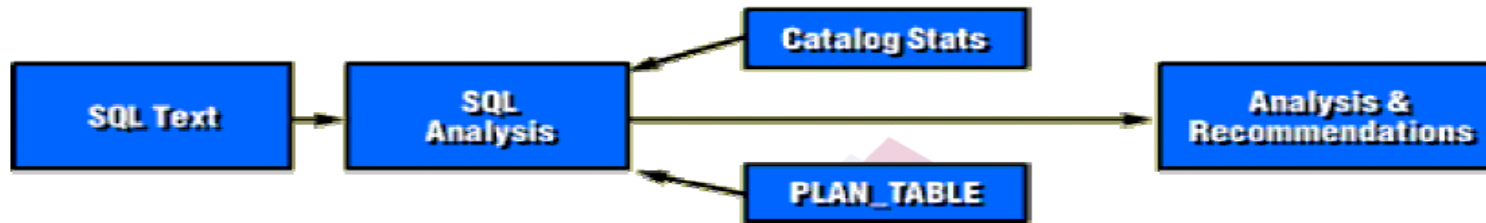
#### STAGE 1

- Data Manager 가 Data 검색 시 조건 점검 ( sargable )
- STAGE 2 보다 performance 가 우수함
- Data Manager 가 STAGE 2 에 DATA 의 량을 줄여 보낼 수 있음.

# 4. Application Tuning : Optimization



## 4. Application Tuning : *EXPLAIN* 분석



- Hint 가 사용 ?
- Index 가 사용 ?
  - Single /Multiple ?
- Matching Column 인가 ?
- Index Only 로 사용 ?
- Tablespace scan ?
- Join 형태는 ?
  - Nested Loop ?
  - Merge Scan ?
  - Hybrid join ?
- SQL Text
- Table 과 Index 정보
  - DDL
  - Stats
- Cardinality
- 기타 정보
  - Trigger ?
  - RI ?
  - Constraint ?
- Prefetch ?
  - Sequential
  - List
- Parallelism 사용 ?
  - I/O, CPU, Sysplex
  - Degree
- Sort 수행 ?
  - Join, Unique, Group by, Order by
- Locking

## 5. Application Tuning : Locking

- Update 를 동일한 순서로 coding 하여 Deadlock 발생을 예방 (Program 의 변경 없이)
- 가능하면 data 변경 용 SQL 은 UOW 의 마지막에 coding
  - 해당 uow 내에서 update 가 나중에 발생 할수록 lock 지속 기간이 짧아 짐
- Lock Avoidance 를 권장
  - ISOLATION(CS)/CURRENTDATA(NO)
  - READ ONLY CURSOR 인 경우만 적용 가능
- LOCK TABLE 구문은 주의가 필요함
- Locking 을 피하려면 ISOLATION(UR) 지정으로



## 6. Application Tuning : Commit

- Application 에서 COMMIT 을 적절히 coding 하시길 !



COMMIT

- COMMIT 의 coding 전략 을 수립하시길 !
  - TIMEOUT 이나 DEADLOCK 을 경험 하지 않도록 !



## 7. Application Tuning : Program

---

- 비효율적인 Program Logic 에 효율적인 SQL 을 coding 하면 ?
  - SQL tuning 이 잘 되어도 looping 이 1,000,000 번 발생한다면 ?
- 가능하면 SQL 로 coding 해결
  - 예: Program coding 으로 join 하는 경우 ???
- 문제 coding : 다수의 IF...ELSE 또는 CASE statement 이후에 SQL 이 coding 되는 경우
- 검색할 data 가 1 건인 경우 singleton SELECT 를 사용하도록
- RUNSTATS 수행을 잊지 말도록
  - EXPLAIN(YES) 지정으로 (RE)BIND 하시기를

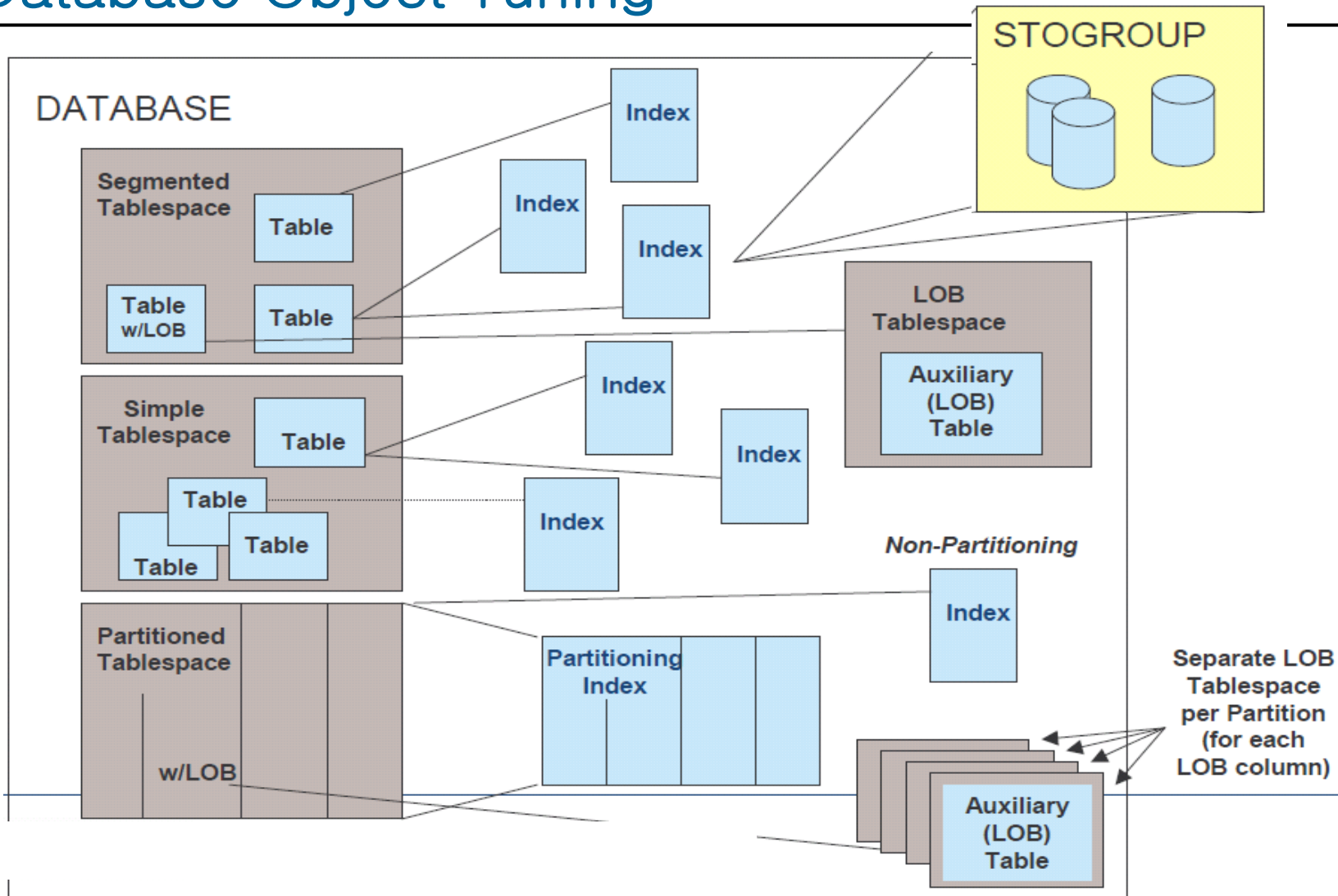
## 8. Application Tuning : *Online 과 Batch*

---

- Online transaction 설계 시 검색 대상 data 건수를 적절히 제한
  - Online 업무가 다량의 page 나 screen 을 검색 하지 않도록
- Online Sort 나 Join 의 발생 을 제한 하도록
- “OPTIMIZE FOR 1 ROW” 를 적용하여 List Prefetch 를 배제
  - LP 가 적용 되는 경우에는 DB2 가 Matching Index 에 대한 RID list 를 수집, sort 수행 후 RID List 이용하여 Data 를 access
  - 다수 page 검색 시에는 매우 효율적인 방법 임
- ROWID 를 이용한 direct access 고려
  - 동일한 row 가 여러 번 검색 될 수 있음

# Database Object Tuning

9 ~ 15



## 9. Database Organization

- 반드시 Rustats 작업을 수행
  - Database 의 Volume 증감에 의하여 신규 data structure 추가
- Data set 의 Extent 정리
- Statistics 점검 및 REORG 시점 결정
  - NEARINDREF 와 FARINDREF 점검
  - LEAFDIST, PERCDROP
  - Clustering Index 를 대상으로
    - NEAROFFPOSF 와 FAROFFPOSF
    - CLUSTERRATIOF
- Access pattern 분석 후 Reorganiz.
  - Random 대비 Sequential
  - 자동화 ( Automation ) 고려

Don't just REORG weekly  
=  
or monthly

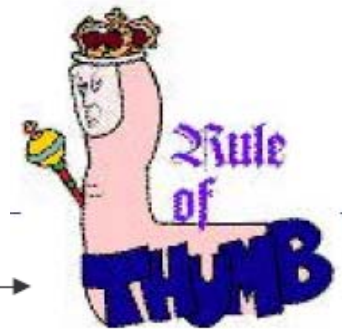
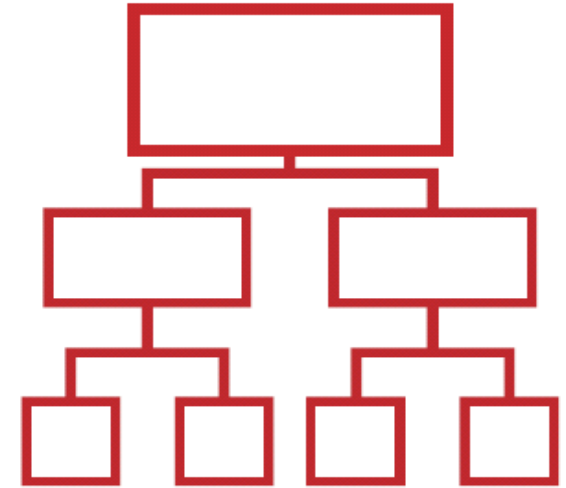
## 10. Database Design : 기본 이론

- 가능한 Normalization ( 정규화 ) 을 지키도록
  - Data Model 로 Physical Design 을 주도 하지는 않도록
- Tablespace 마다 Table 한 개 지정
- Simple Tablespace 보다는 Segmented 나 Partitioned 로
  - 반드시 대용량의 table 을 필요로 하는 경우가 아니라  
면 DSSIZE < 4GB 로 지정 하도록
- BASE TABLE 을 VIEW 로 정의 하지 않도록
- 되도록 Default 로 지정하지 않도록
  - 일반적으로 적절한 지정이 아닐 수 있음
- Free Space 의 양을 미리 결정 하도록
  - Data 의 삭제 주기를 고려



# 11. Database Design : Index 들

- Index 적정 지정 시 최적의 Performance 를 수 있음
- Unique 와 Primary Key Constraint 대상
- 이후 Foreign Key 고려
- Heavy Transaction Query 의 SQL 조건
- IXO 에 대한 Column 의 Overload
- Sorting 이 발생하지 않도록 Index 지정
  - ORDER BY, GROUP BY
- Insert/Update/Delete 고려
- Sequence column 들이 multi-column index 대상
- Index 에는 Variable Column 을 지정 하지 않도록



## 12. Database Design : Data Type 들

---

- Null 지정은 필요할 경우에 한정하여
- DB2 Data Type 을 적절히 지정하도록
  - 날짜 의 경우 CHAR 나 numeric 대신 DATE Type 을 지정
  - Numeric Data 는 numeric data type 을 지정
    - INTEGER, SMALL INTEGER, DECIMAL 등
  - INTEGER 와 DECIMAL(x,0) 구분
    - Storage 요건 등
  - “DATE 와 TIME” 과 timestamp 구분
    - 사용의 용이성 / Storage 와 Precision / 연산 비교
- Compression 과 VARCHAR
  - Compression 의 Overhead 가 적음 ( 2 byte prefix 가 없음 )
  - Compression 은 Program 의 고려사항이 없음

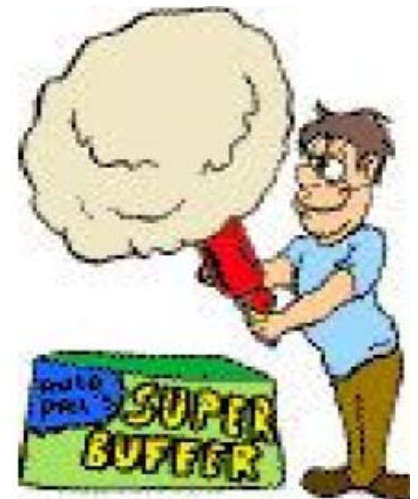
## 13. Database Design : Integrity



- Program 의 RI 관계 대신 DB2 의 RI 를 지정 하도록
  - Performance 와 사용의 용이성
  - Planned 와 Adhoc database 수정에 대한 Integrity 확인
- Look up table 에 대한 RI 는 지정하지 않도록
  - CHECK Constraint 와 lookup table 비교
- RI 가 지정 되기 어려운 경우에 Trigger 지정
  - Trigger 들이 RI 보다는 효율이 낮음
    - 그러나 Application Program 에서 점검 하는 것 보다 효율적임
- Foreign Key 에도 Index 지정

## 14. Database Design : BP

- Bufferpool Allocation
  - BP0 를 모두 경우의 Default 로 지정 하지 말 것
  - 개별 Tablespace / Index 마다 Bufferpool 을 지정하도록
- Idea
  - Catalog 는 BP0 에 지정
  - Index 와 Tablespace 는 별개 Bufferpool 에 지정
  - Heavy 하게 사용되는 Data 는 Bufferpool 분리
  - Sort Work Area 도 Bufferpool 분리
  - Data / Apple 에 대한 BP 전략 최적화
    - Sequential 인지 Random 인지
  - Bufferpool Tuning 으로 추후 조정



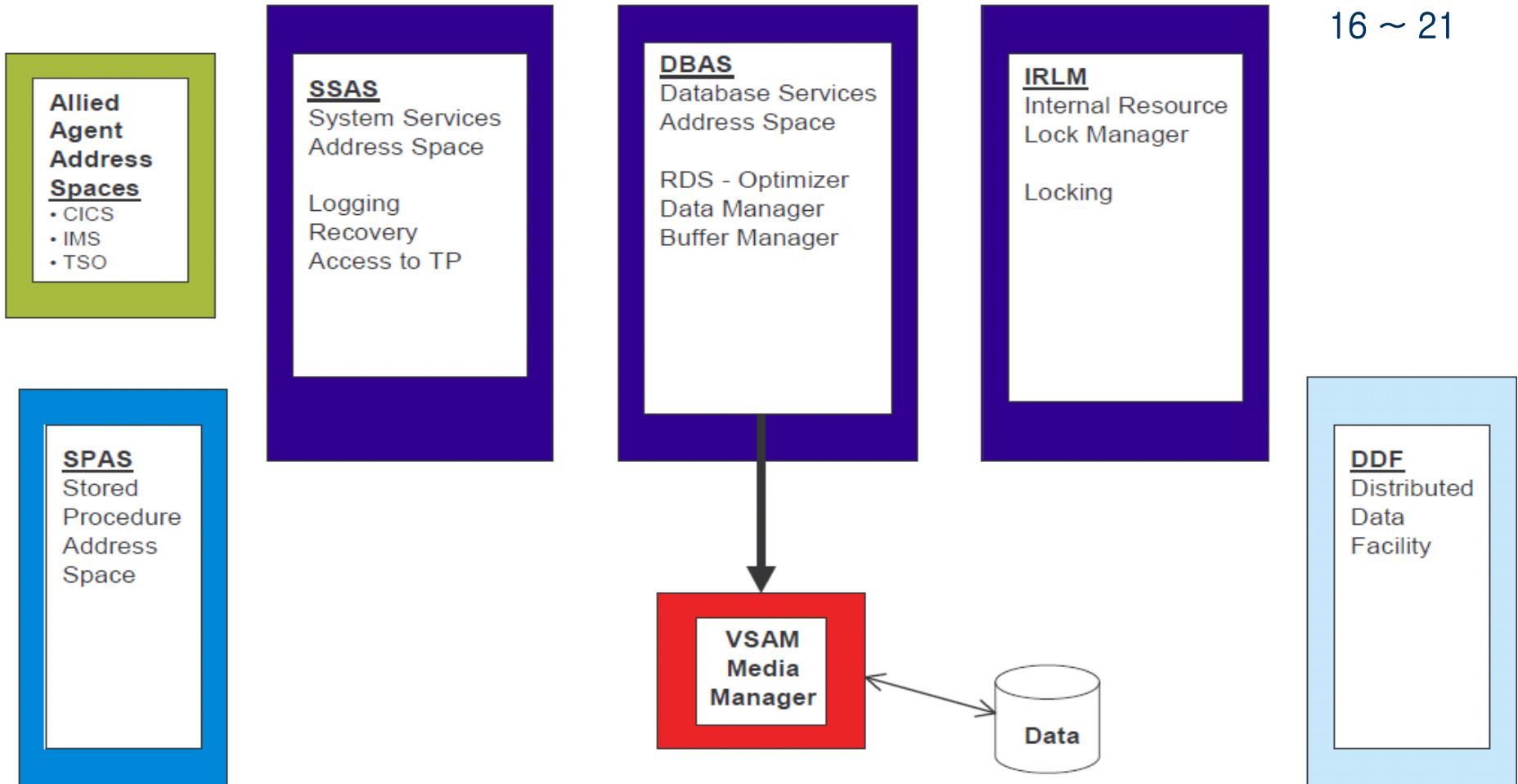
## 15. Database Design : Row 와 Column

---

- Space 의 낭비 예방
  - Row 길이 > 4056 이면 대량의 Page Size 가 필요함
  - Row 길이가 2029 ~ 4056 이면 Page 마다 Row 가 1 건
  - Row 길이 < 15 이면 Space 가 낭비됨
- Column 의 순서는 logging 고려
  - 변경이 드문 non-variable length column 을 가장 앞에 지정
  - 변경이 드문 variable length column 을 그 다음 위치에 지정
  - 변경이 자주 발생하는 column 을 가장 오른쪽에 지정
  - 그 외의 변경이 발생하는 column 들을 왼쪽에서 오른쪽으로 지정

# System 과 DB2 Subsystem Tuning

16 ~ 21



## 16. ~19. Subsystem Tuning : Pool 들

---

- DB2 는 4 가지 종류의 Pool 을 사용함 : Data 와 Information 을 임시 저장하는 Memory Structure 로 Disk I/O 의 경감 효과
  - ①⑥ – Buffer Pool 들 : Disk 에서 data 를 읽어와서 memory 내에 Data 를 임시 보관
  - ①⑦ – RID Pool : RID ( Record Identifier ) 의 Sort 시 이용 됨  
List Prefetch, Multiple Index Access,  
Hybrid Join 에 사용됨
  - ①⑧ – EDM POOL : Program Detail ( Access Path, Dynamic PREPARE, Authorization ) 와 Database Structure Information ( DBD ) 를 임시 보관함
  - ①⑨ – Sort Pool : DB2 가 Data 에 대한 Sort 수행 시 사용됨

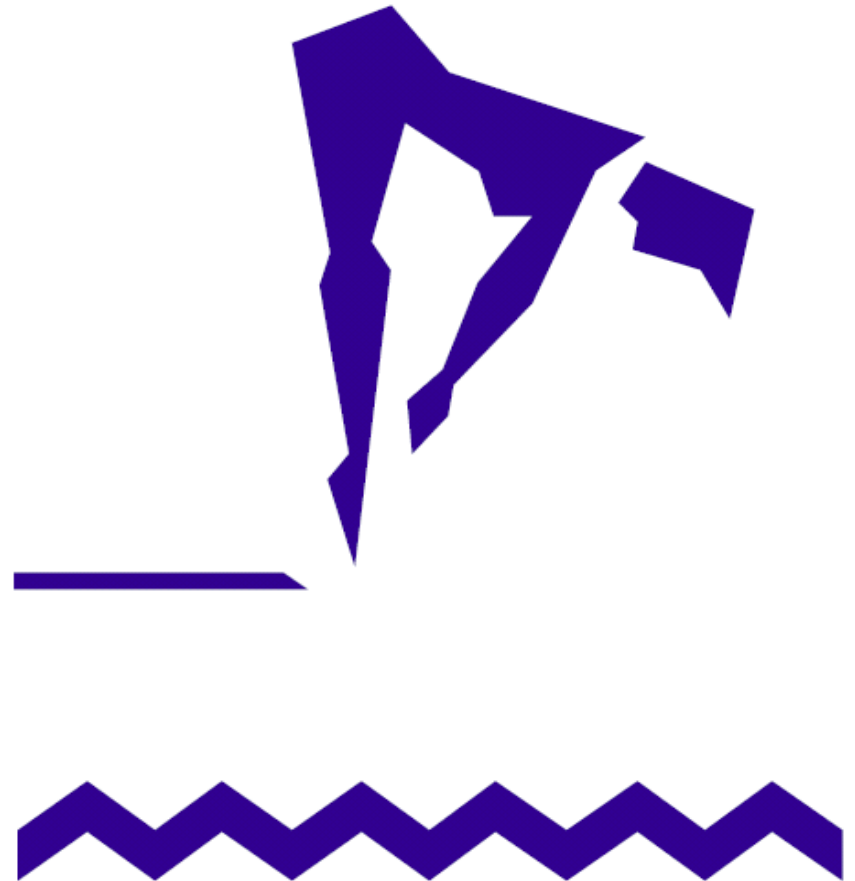
## 16. Subsystem : Buffer Pool 들

---

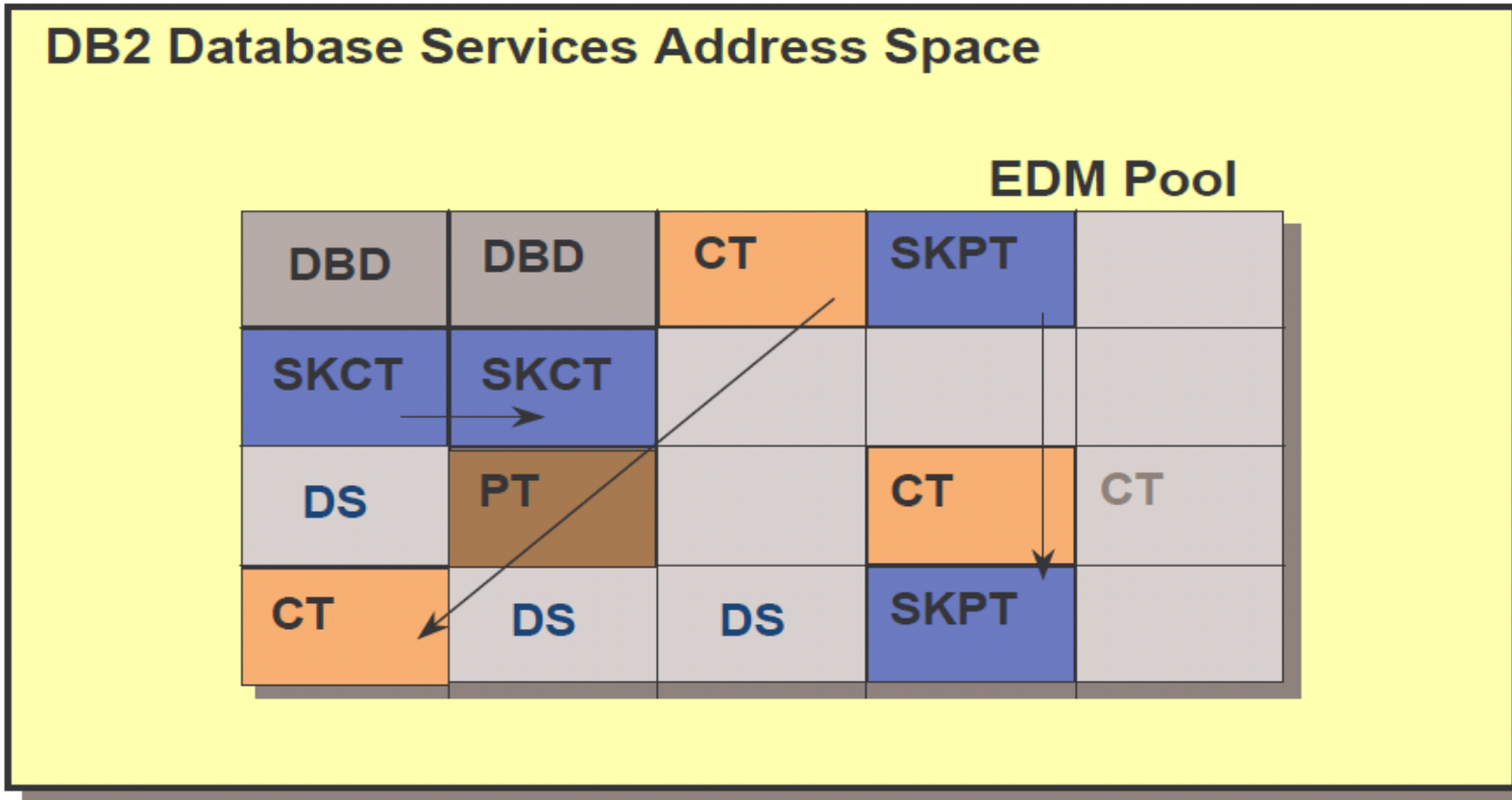
- DB2 가 제공하는 Bufferpool 은 80 가지
  - 4K : BP0 ~ BP49
  - 8K : BP8K0 ~ BP8K9
  - 16K : BP16K0 ~ BP16K9
  - 32K : BP32K0 ~ BP32K9
- Tuning 용도로 Bufferpool 을 등록
  - 문제 Object 들을 옮겨 Tuning 을 위하여 분리
- HIT Ratio 점검 : Bufferpool 내에서 대상 Page 가 발견된 비율
  - Hit Ratio 가 높을 수록 좋음
    - $HIT\ RATIO = (GETPAGES\ REQ - PAGES\ READ) / GETPAGES\ REQ$ 
      - $PAGES\ READ = SYNC\ I/O + ASYNC\ I/O$

## 17. Subsystem : RID Pool

- 모든 Processing 에 대한 RID Pool 은 하나 임
- Default RID Pool 크기는 4MB
  - 크기 변경 가능
- RID Pool 의 용도
  - Multiple row 변경 시 Unique Key 강화
  - RID 에 대한 Sorting 수행
    - List Prefetch
      - Single index list prefetch 포함
      - Multiple Index 이용한 access
      - Hybrid Join



# 18. Subsystem : EDM Pool



- EDM Pool**
- DBDs
  - SKCTs
  - CTs
  - SKPTs
  - PTs
  - Auth Cache
  - Dyn SQL Prep
  - Free pages

- General ROT : EDM Pool 은 사용률이 80% 이하를 유지 하도록

## 19. Subsystem : Sort Pool

---

- Sort Pool value 는 개별 concurrent sort user 마다 할당되는 SORT WORK 의 최대량
- Default Sort Pool 의 크기는 1MB DSNTIPC  
에서 변경 가능
- Sort Pool 의 최대 크기
  - $16000 * (12 + \text{sort key length} + \text{sort data length} + 4)$
- Sort Pool 이 클수록 Sort 가 효율적임

## 20. Subsystem : Logging

- DB2 의 처리 속도는 Log 의 속도에 제한을 받음
- Log Tuning Parameter
  - Input Buffer 크기
  - Output Buffer 크기
  - Write Threshold
- Log Configuration
  - Dual Logging 이 바람직함
  - 개별 Log 들을 별도 device / channel 에 위치시킴
- DB2 는 log data 를 log 로 부터 DASD 로 적용
  - Archive log 는 DASD 에 지정 하도록
  - 일정 기간이 지나면 archive log 를 tape 에 이행



## 20. Subsystem : System Checkpoint

---

- DB2 는 주기적으로 checkpoint 를 생성함
- Dsnzparm 의 LOGLOAD ( CHKFREQ) 에 지정됨
  - Write 되는 Log Record 수
  - 시간 ( 분 )
- Dynamic 하게 변경 가능
  - SET LOGLOAD
  - SET SYSPARM

# Summary

Application

Database

DB2 Subsystem

Environment

Do one thing at a time and;

You *can* tune DB2!

