Information Management

IBM

Data Management

Informix in a DSS environment

Alfatec Conference 2010
Joe ‘Celo’ Baric – jbaric@us.ibm.com

Agenda

- **Know Your Environment**
- **Configuration**
- **Tuning**
- **Fragmentation**
- **PDQ**
- **Optimizations**

Agenda

- **Know Your Environment**
  - OLTP
  - DSS
  - DSS+OLTP
  - ETL
- **Configuration and Tuning**
- **Fragmentation**
- **PDQ**
- **Otimizations**

Know Your Environment - OLTP

- OLTP Characteristics
  - Small number of rows returned
  - Quick response times of queries
  - High read and write buffer cache rates
  - Short checkpoint durations
  - Maximum I/O throughput
    - Eliminating I/O bottlenecks
    - Optimizing index utilization
    - Optimizing fragmentation strategy
  - Known / limited “windows” for periodic tasks
  - Short “fast recovery” times

Know Your Environment - DSS

- DSS Characteristics
  - Large number of rows processed and returned
  - Summations/Aggregations common
  - Optimum memory utilization
  - Parallel data queries (PDQ)
  - Light scans
  - Maximum I/O throughput
  - Tend to be queried by front-end tools

Know Your Environment – DSS Considerations

- Important to focus on specific areas that will have the greatest impact on performance.
- Disk I/O and specifically disk **reads** are most important.
- Optimum **memory** utilization will have the greatest impact.
- Fragmenting tables and indexes intelligently will generally produce the most significant improvements.
- Focus on how data is used, utilize fragmentation
- Review SQL for faster query timings
- Utilize **PDQPRIORITY**
- Utilize temp tables for improved performance
- Have enough temporary dbspaces for sorts

DSS Tuning Considerations

- Optimum memory utilization
- Parallel data queries (PDQ)
- Light scans/Light Append
- Maximum I/O throughput

DSS Best Practices

- Focus on how data is used, utilize fragmentation
- Review SQL for faster query timings
- Utilize PDQPRIORITY
- Utilize temp tables for improved performance
- Have enough temporary dbspaces for sorts
- Consider using PSORT_DBTEMP w/ PSORT_NPROCS also (may need to do some benchmarking to find which performs better in your specific environment)

Know Your Environment – OLTP + DSS

- OLTP + DSS Characteristics
  - Combination of DSS Queries with OLTP transaction activity
  - Number of records returned depends on type of activity
  - Maximizing I/O throughput still important
  - Memory resources allocated with primarily DSS queries in mind.

DSS SQL Strategies

- Utilize “UNIONS” when you have “OR” in where clause
- Utilize temp tables in optimizing queries by splitting a query into multiple subqueries
- Utilize PDQPRIORITY
- Utilize DS_NONPDQ_QUERY_MEM (V 9.40/10.00 and above)
- Fragment tables (Understand the use of the data) to eliminate fragments from selection of the data
- Utilize external directives (V 10.00 and above)

Know Your Environment – OLTP + DSS Considerations

- Important to balance resources and loads on the system.
- Important to continuously monitor resources.
- Need to control users by preventing execution of poorly written queries which could monopolize all server resources.
- Try to run DSS queries and batch jobs in non-peak hours.
- Minimize amount of **DS_TOTAL_MEMORY** / **MAX_PDQPRIORITY** used during peak time, increase during off hours to run DSS type queries
- Dynamically adjust **DS_TOTAL_MEMORY** / **MAX_PDQPRIORITY** during peak and off hours

Environment: ETL – Bulk Loads of Data

- Use Raw Tables
- PDQ
- External tables when possible
- HPL (heterogeneous data)
- Light Appends
- Minimum or No indices during the load

Agenda

- Know Your Environment
- **Configuration**
  - Table Layout
  - Page Size
  - Disk Considerations
    - JABOD
    - RAID
    - Software Based RAID
    - IDS Mirroring
    - Coked Files
- Tuning
- FRAGMENTATION
- PDQ
- Optimizations

Table Layout

- **Large extents to minimize disk latency**
- **Consider using larger page sizes (dependent on row size)**
- **Consider putting large tables in their own dbspaces**
- **Indexes should also be offset into their own dbspaces**
- **Consider fragmenting tables when possible**

Page Size

- **IDS can configure page sizes that are multiples of the default page size**
- **Depending on OS can be 2k, 4k, 8k, or 16k**
- **Each page size will necessitate its own buffer pool (in the virtual portion of memory)**
- **Can be useful if OS supports larger pagesizes for buffer transfers.**
  - Ensure adequate memory exists

Disks Considerations

- **JABOD**
- **RAID**
  - Level 0
  - Level 1
  - Level 10
  - Level 5
  - Level 6
- **Software Based Raid**
- **Mirroring**
- **Cooked Files**

JABOD

- **Just A Bunch Of Disks**
- **Allows DBA to work with SA to place disks for minimum latency and contention**
  - Align disk layout with fragmentation
- **Most versatile**

RAID overview

- **Redundant Array of Inexpensive Disks**
- **Disk layout is removed from DBMS control**
- **Not optimal for IDS**
- **RAID Levels**
  - 0 – Data is striped across disks
  - 1 – Simple mirroring implemented
  - 10 – Striped and mirrored
  - 5 – Uses parity on the same disk to check consistency
  - 6 – Uses parity on other disks in the array to check consistency

Software Based RAID

- **Not recommended, can hurt performance**
- **Work handled by CPU**
- **Can impact performing other processes for IDS**

- **Recommendations**
  - Level 10 is recommended
  - Avoid Levels 5 or 6 for performance (though disk usage is optimized)
  - If possible, use a buffering factor that reflects the page size

IDS Mirroring

- **Description**
  - Writes are done to the primary
  - Asynchronously copied to the secondary
  - Reads are then split between primary and secondary
- **Advantages**
  - Improved query performance, especially for OLAP
- **Disadvantages**
  - Potential performance hit (with writes)
  - Cost can be more as entire dbspaces must be mirrored, so potentially unimportant data is also mirrored.

Cooked Files

- **AIO vps**
- **Not optimal,**
  - Administration is handled by CPUs (can impact iDS)
  - Disk layout is outside of DBA control
- **Direct I/O**
  - Introduced with 11.10
  - Performance approaches KAIO (raw devices)

Agenda

- Know Your Environment
- Configuration
- **Tuning**
  - Memory Tuning
  - Light Scans
  - Light Appends
  - Sorting
- FRAGMENTATION
- PDQ
- Optimizations

DSS Tuning: Memory Utilization

- BUFFERS Minimize.
- SHMVIRTSIZE Maximize.

75% or more of avail mem

- SHMADD 64000*
- SHMTOTAL Maximize
- RA_PAGES Set to 128.

#use formula in next slide to see effectiveness

- RA_THRESHOLD Set to 120.
- DS_TOTAL_MEMORY Set to 90% of SHMVIRTSIZE

Read Ahead Utilization

- ixda-RA + idx-RA + da-RA should equal RA-pgsused.

OLTP + DSS Tuning

- Need to balance resources/loads on the system
- Run DSS type queries in off hours
- Minimize amount of DS_TOTAL_MEMORY / MAX_PDQPRIORITY used during peak time, increaseduring off hours to run DSS type queries
- Dynamically adjust DS_TOTAL_MEMORY /MAX_PDQPRIORITY during peak and off hours
- Consider using DS_NONPDQ_QUERY_MEM for OLTP loads

DS_NONPDQ_QUERY_MEM

- **Limits amount of memory that is available for DSS type queries (OLTP loads)**
- **Can also be used to ensure maximum aount of memory is allocated for DSS (by setting to minimum 128k)**
- **Range 128k to approx 25% of DS_TOTAL_MEMORY**
- **Can help performance with small sort tasks**

Light Scans

- **Bypasses buffer pool**
- **Used for sequential scans**
- **Monitored with onstat –g lsc**

- **Advantages:**
  - Transfers larger blocks of data in one I/O operation
  - Bypasses the overhead of the buffer pool when many pages are read
  - Prevents frequently accessed pages from being forced out of the buffer pool when many sequential pages are read for a single DSS query
  - Uses big buffers

Light Scan scenarios

- **The optimizer chooses a sequential scan of the table.**
- **The number of pages in the table is greater than the number of buffers in the buffer pool.**
- **The isolation level obtains no lock or a shared lock on the table:**
  - Dirty Read (including nonlogging databases) isolation level
  - Repeatable Read isolation level if the table has a shared or exclusive lock
  - Committed Read isolation if the table has a shared lock

Monitoring Light Scans

- **Use onstat –g lsc**

Light Append

- **There is no attempt to buffer the data in the buffer pool,**
- **Data is written directly to new pages.**
- **At the end of a successful load the new pages are appended to the existing table.**

Sorting Considerations

- **Temporary DBSpaces – uses internal threads**
- **PSORT_DBTEMP – Uses OS files with external processes -- can be faster than temporary dbspaces**

PSORT_DBTEMP

- **Environment Variable**
- **Used in conjunction with PSORT_NPROCS**
- **Set to a directory**
- **Example: export PSORT_DBTEMP=/tmp**

PSORT_NPROCS

- **Environment Variable**
- **Used in conjunction with PSORT_DBTEMP**
- **Used to define the number of external processes for sorting**
- **General rule is approximately 2 procs per CPU VP**

Sorting Priority

- **Order of precedence for Sort settings with Informix**

1. **PSORT_DBTEMP/PSORT_NPROCS**
2. **DBSPACETEMP environment variable**
3. **DBSPACETEMP configuration parameter**
4. **DUMPDIR configuration parameter**
5. **$INFORMIXDIR/tmp**

Agenda

- Know Your Environment
- Configuration
- Tuning
- **FRAGMENTATION**
  - Round Robin
  - Fragment by Expression
  - Fragment Elimination
  - Data Layout
- PDQ
- Optimizations

Fragmentation Strategies

- **Round Robin**
- **Expression**

Round Robin

- **Rows are inserted in alternating partitions, switching to the next partition for each row**

Advantages:

- **Optimal for Bulk Load**

Disadvantage

- **Not optimal for indexing (and subsequent searching)**
- **Cannot be used with fragment elimination**

Fragment by Expression

- **Expression-based fragmentation scheme**
  - **Range Rule**

FRAGMENT BY EXPRESSION c1 < 100 IN dbsp1, c1 >= 100 AND c1 < 200 IN dbsp2, c1 >= 200 IN dbsp3;

  - **Arbitrary Rule**

FRAGMENT BY EXPRESSION zip_num = 95228 OR zip_num = 95443 IN dbsp2, zip_num = 91120 OR zip_num = 92310 IN dbsp4, REMAINDER IN dbsp5;

Advantages:

Useful for fragmentation elimination

Data immediately avail for queries that match FRAGMENT BY criteria

Disadvantages:

Slower initial load

Fragmentation Elimination

- **Eliminates fragments from being considered for processing and searching, so no resources are wasted searching in fragments with no rows that match the search criteria**

- **Eligible operators:**
  - IN, =, < >, <= ,>=, AND ,OR, NOT, MATCH, LIKE, >, <
- **INELIGIBLE operators:**
  - !=, IS NULL, IS NOT NULL

Fragment Elimination Guidelines

- **Nonoverlapping fragments on a single column:**
  - A fragmentation rule that creates nonoverlapping fragments on a single column is the preferred fragmentation rule from a fragment-elimination standpoint.
- **Overlapping fragments on a single column**
  - The fragments on a single column can be overlapping and noncontiguous. You can use any range, MOD function, or arbitrary rule that is based on a single column.
- **Nonoverlapping Fragments, Multiple columns**
  - The database server uses an arbitrary rule to define nonoverlapping fragments based on multiple columns

Fragment Elimination on Equality Expression

- **Fragments can be eliminated on Nonoverlapping fragments on a single column**
- **Fragments can be eliminated on overlapping or non-contiguous fragments on a single column**
- **Fragments can be eliminated on nonoverlapping fragments on multiple columns**

Fragment Elimination Range Expression

- **Fragments can be eliminated on Nonoverlapping fragments on a single column**
- **Fragments cannot be eliminated on overlapping or non-contiguous fragments on a single column**
- **Fragments cannot be eliminated on nonoverlapping fragments on multiple columns**

Data Layout

- Consider reorganizing tables into fewer, larger extents
- Purge or offload unused data to an “archival site”
- Create periodic jobs to purge offload unused records
- Consider using “Roll-out” partitions, which can be detached regularly (ex. Month by month)
- Consider recreating and/or redefining indexes
- Add new indexes for improved selection
- Consider Compression for both Tables and Indices
- If multiple tables lie in the same dbspace, monitor extent usage and layout (oncheck –pe). Consider using repack to keep extents contiguous

SQL Strategies

- Utilize “UNIONS” when you have “OR” in where clause
- Utilize temp tables in optimizing queries by splitting a query into multiple subqueries
- Utilize PDQPRIORITY
- Utilize DS_NONPDQ_QUERY_MEM (V 9.40/10.00)
- Fragment tables (Understand the use of the data) to eliminate fragments from selection of the data
- Utilize external directives (V 10.00)
- Index Self-Join (V11.10)

Adding and Detaching a Fragment

- **ALTER FRAGMENT ON TABLE table ATTACH fragment**

- **ALTER FRAGMENT ON TABLE table DETACH fragment**
  - FRAGMENT can be a dbspace OR a partition

Alter Table Attach Fragment Performance

- **Formulate appropriate distribution schemes for your table and index fragments.**
- **Ensure that no data movement occurs between the resultant partitions due to fragment expressions.**
- **Update statistics for all participating tables.**
- **Make indexes on the attached tables unique if the index on the original table is unique.**

Agenda

- Know Your Environment
- Configuration
- Tuning
- FRAGMENTATION
- **PDQ**
  - MAXPDQPRIORITY
  - PDQPRIORITY
  - DS_MAX_SCANS
  - DS_MAX_QUERIES
  - DS_TOTAL_MEMORY
  - Memory Quantum
  - Scan Quantum
- Optimization

PDQ Settings

MAX_PDQPRIORITY

PDQPRIORITY

DS_MAX_SCANS

DS_MAX_QUERIES

DS_TOTAL_MEMORY

MAX_PDQPRIORITY

- **ONCONFIG parameter**
- **Limits the PDQ resources that can be allocated to any one DSS query**
- **Used to scale resources allocated by users (PDQPRIORITY)**
- **Use onmode –D to change MAX_PDQPRIORITY dynamically (not saved to ONCONFIG)**

Values:

- **0 -- off**
- **1 -- Fetches data from fragmented tables in parallel (parallel scans) but uses no other form of parallelism.**
- **100 -- A PDQ query can use all available resources to process a query**
- **2-99 – Percentage of PDQ resources allocated to any one PDQ query**

PDQPRIORITY

- **SET parameter**
- **Determines resources dedicated to any PDQ query (as a percentage of MAXPDQPRIORITY)**
- **SET parameter overrides environment variable**

PDQPRIORITY Settings

- **OFF --** PDQ is turned off. The database server uses no parallelism. OFF is the default if you use neither the PDQPRIORITY environment variable nor the SET PDQPRIORITY statement.
- **LOW --** Data values are fetched from fragmented tables in parallel. (In Dynamic Server, when you specify LOW, the database server uses no other forms of parallelism.)
- **1—100** – A percentage of available resources (subject to MAX_PDQPRIORITY)
- **HIGH --** The database server determines an appropriate PDQPRIORITY value, based on factors that include the number of available processors, the fragmentation of the tables being queried, the complexity of the query, and others. IBM® reserves the right to change the performance behavior of queries when HIGH is specified in future releases.

DS_MAX_SCANS

- **ONCONFIG Parameter**
- **Use onmode –S to change ONCOFNIG dynamically (Not saved to ONCONFIG)**
- **Number of PDQ Scan threads**
- **Range – 10 to 1048576**

- **For any PDQ query, the database server calculates the number of scans to apportion, dependent on the following values:**
  - **The value of PDQ priority (set by the environment variable PDQPRIORITY or the SQL statement SET PDQPRIORITY)**
  - **The ceiling that you set with DS_MAX_SCANS**
  - **The factor that you set with MAX_PDQPRIORITY**
  - **The number of fragments in the table to scan (*nfrags* in the formula)**

DS_MAX_QUERIES

- **ONCONFIG Parameter**
- **Use onmode –Q to set dynamically (Not saved to ONCONFIG)**
- **Range: 1 - 1,388,608**
- **Used to specify the number of PDQ queries that can run concurrently**

- **If not set, the value of the DS_MAX_QUERIES configuration parameter is dependent on the setting for the DS_TOTAL_MEMORY configuration parameter:**
  - **If DS_TOTAL_MEMORY configuration parameter set, then the caclulated value of the DS_MAX_QUERIES is DS_TOTAL_MEMORY / 128, rounded down to the nearest integer value.**
  - **If the DS_TOTAL_MEMORY configuration parameter is not set, then the calculated value of the DS_MAX_QUERIES configuration parameter is 2 * num, where num is the number of CPUs specified in the VPCLASS configuration parameter**.

DS_TOTAL_MEMORY

- **ONCONFIG parameter**
- **Specifies amount of memory available for PDQ**
- **Set in units of Kilobytes**
- **Use onmode –M to change DS_TOTAL_MEMORY dynamically (not saved to ONCONFIG)**
- **Range:**
  - **If DS_MAX_QUERY is set, the minimum value is DS_MAX_QUERIES * 128.**
  - **If DS_MAX_QUERIES is not set, the minimum value is num_cpu_vps * 2 * 128.**
- **Maximum value for 32-bit platform: 2 gigabytes**
- **Maximum value for 64-bit platform: 4 gigabytes**

Monitoring PDQ: onstat –g mgm

- **mgm stands for “Memory Grant Manager”**
- **Fist part, shows the current PDQ settings (ie)**

MAX_PDQPRIORITY: 100

DS_MAX_QUERIES: 31

DS_MAX_SCANS: 1048576

DS_NONPDQ_QUERY_MEM: 128 KB

DS_TOTAL_MEMORY: 4000 KB

Monitoring PDQ Resources: onstat –g mgm part 2

- **The second part contains information about the gates**

Queries: Active Ready Maximum

0 0 31 Memory:

Total Free Quantum (KB)

4000 4000 128

Scans: Total Free Quantum

1048576 1048576 1

Load Control: (Memory) (Scans) (Priority) (Max Queries) (Reinit)

Gate 1 Gate 2 Gate 3 Gate 4 Gate 5 (Queue Length)

0 0 0 0 0

MGM – The 5 Gates

1. **Memory**
2. **Scans**
3. **Priority**
4. **Max Queries**
5. **Initialization State (MGM is being initialized)**

- **A query will wait at the respective gate for the appropriate resource.**

The Quantum

- **memory quantum = DS_TOTAL_MEMORY / DS_MAX_QUERIES**

- **Scan quantum – number of scan threads in a quantum (scan threads are allocated in blocks)**

Monitoring PDQ: onstat –g mgm part 3

Query status

Active Queries: None
Ready Queries: None
Free Resource Average # Minimum #
--- --- ---
Memory 0.0 +- 0.0 500
Scans 0.0 +- 0.0 1048576
Queries Average # Maximum # Total #
--- --- --- ---
Active 0.0 +- 0.0 0 0
Ready 0.0 +- 0.0 0 0
Resource/Lock Cycle Prevention count: 0

Optimizations

- **Light Scans for Varchars**
- **Raw Tables**
- **External Directives**

Agenda

- Know Your Environment
- Configuration
- Tuning
- FRAGMENTATION
- PDQ
- **Optimizations**
  - Light Scans for Varchars
  - Raw Tables
  - External Directives

Light Scans for Varchars

- **Improves performance by**
  - Enabling light scans on tables with rows that can span pages, which includes:
    - Varchar, lvarchar, nvarchar
    - Compressed tables
    - Any table with rows longer than a page
  - Code path optimizations:
    - Internally pass rows in batches instead of individually
    - Some filtering pushed down closer to the data
    - Streamlined code

Light Scans for Varchars: Enabling

- **Onconfig file:**
  - BATCHEDREAD_TABLE 1
- **Shell environment (set in IDS shell before starting):**
  - Csh family: IFX_BATCHEDREAD_TABLE 1
  - Ksh family: export IFX_BATCHEDREAD_TABLE=1
- **Single session:**
  - set environment IFX_BATCHEDREAD_TABLE ‘1’;

Raw Tables

- **Non-logged tables**
- **Indexes and referential constraints**
- **Unique constraints cannot be defined**
- **Light appends not supported, except with HPL**
- **Intended for iniital loading of data**
- **If load fails, data is loaded up to failpoint**
- **After loading, RAW tables can be converted to STANDARD**

Using RAW tables

1. Creation of the RAW table

CREATE RAW TABLE tablename(collist…)…;

2. Altering to Standard:

ALTER TABLE rawtabname TYPE (STANDARD);

External Directives

- External Directives allow you to use directives on static SQL (SQL that cannot be modified or changed).
- For example, you have an application that you cannot change the SQL statements in it, but are having an issue with the performance of a specific SQL statement.
- With the use of external directives you can override how the optimizer handles the SQL statement by forcing it to use directives.
- Usually used to utilize a different query plan than the optimizer chooses.

External directives syntax

```
SAVE EXTERNAL DIRECTIVES /*+ AVOID_INDEX
(table1 index1)*/, /*+ FULL(table1) */
ACTIVE FOR
SELECT /*+ INDEX( table1 index1 ) */ col1, col2
FROM table1, table2
WHERE table1.col1 = table2.col1
```

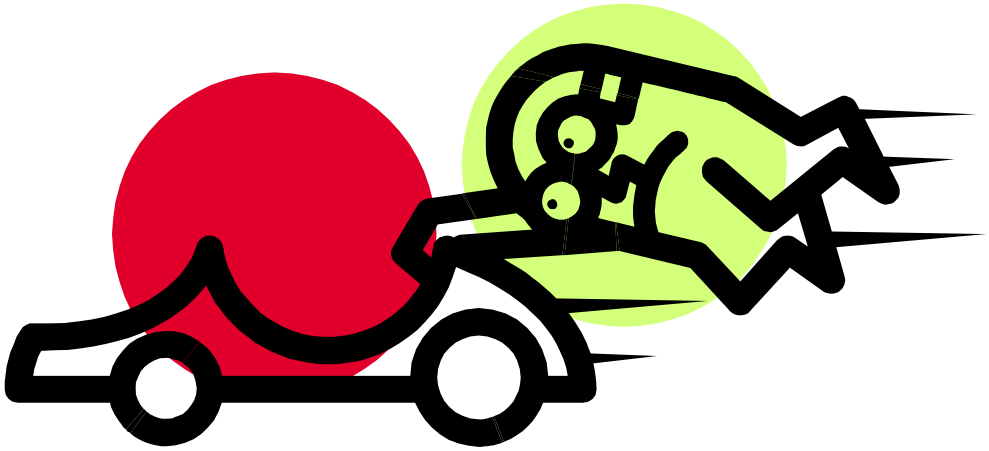
External Directives Enabling and Disabling

ONCONFIG:

- EX_DIRECTIVES (0 – OFF, 1 – ON, 2 – ON)

- Individual sessions external directives can be enabled with the following, all other combinations will have external directives OFF:
  - IFX_EXTDIRECTIVES
  - NOT SET/EX_DIRECTIVES = 2
  - 1 / EX_DIRECTIVES = 1 or 2
  - 0 “NO” External directives no matter what EX_DIRECTIVES is set to

Index Self Join



Purpose of Index Self-join

- **Improve performance on range based indexes scans which have highly duplicate lead keys**
- **Improve use of subsequent parts of composite index when used with range expression**

Scenario

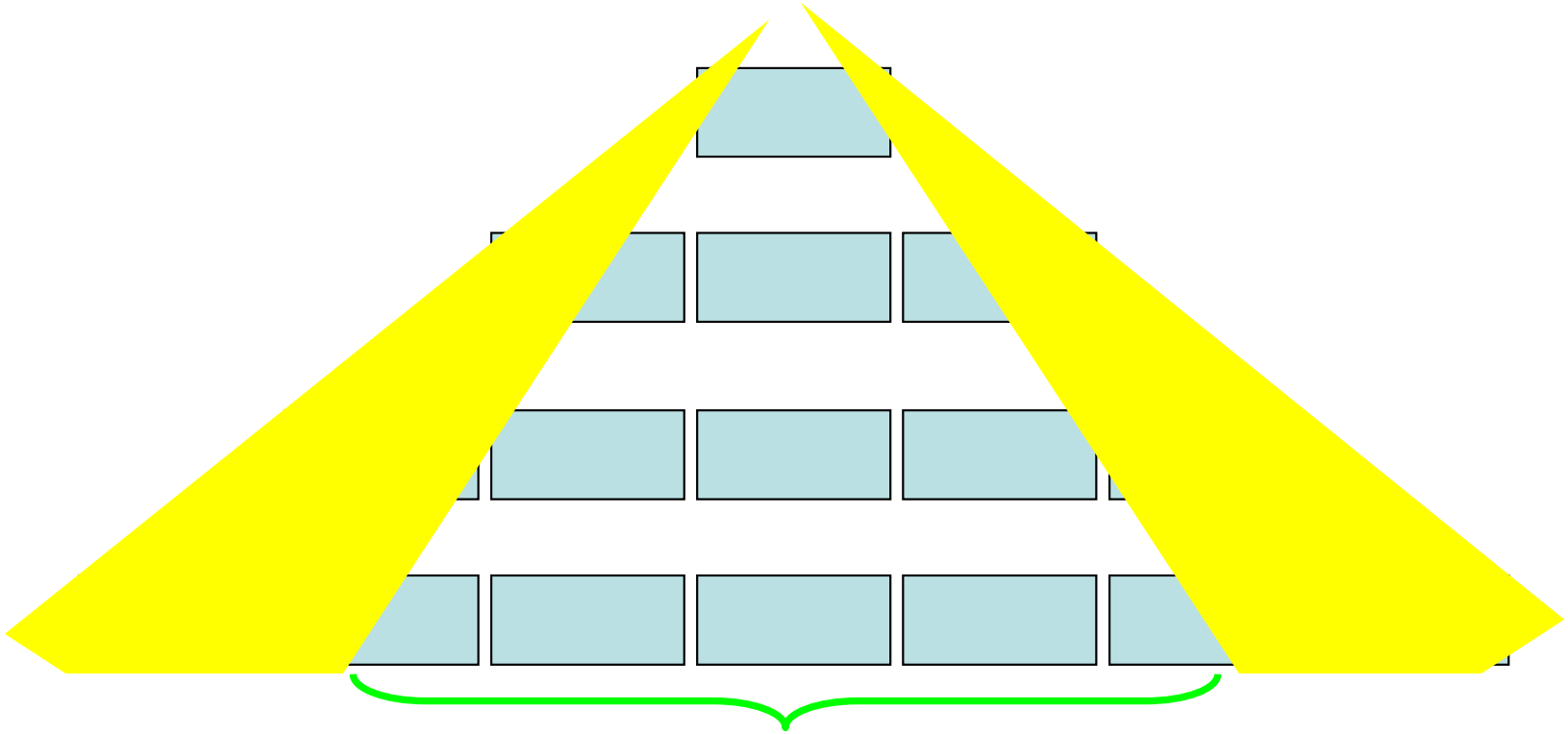
- Composite index defined on columns (DayOfWeek, FamilySize, part, partdesc).
- (DayOfWeek, FamilySize) have lots of duplicates
- Consider using self joins on columns with **distributions**

Query:

```
SELECT *  FROM tab
WHERE  (DayOfWeek >= 1 and DayOfWeek <= 5)
AND  (FamilySize >= 2 and FamilySize <= 4)
AND (part >= 100 and part <= 102)
```

Prior to Version 11

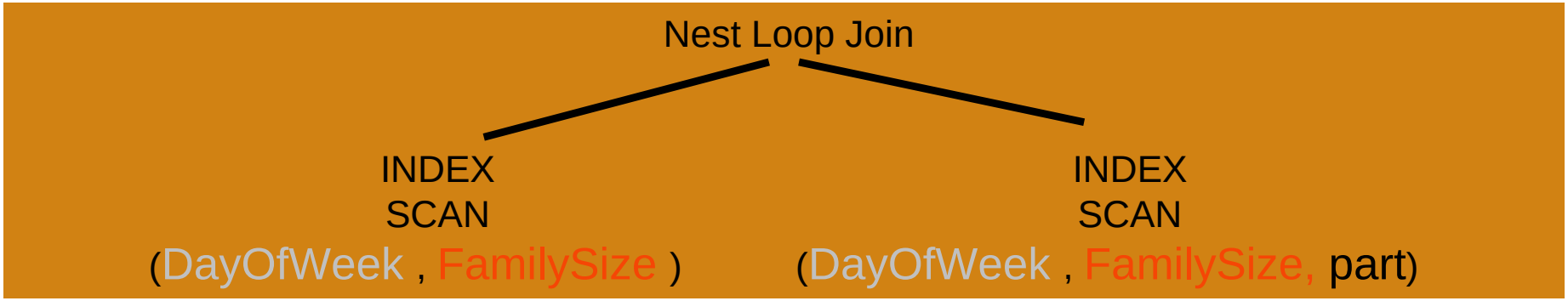
Lower Filter DayOfWeek **>= 1**
Upper Filter DayOfWeek **<= 5**



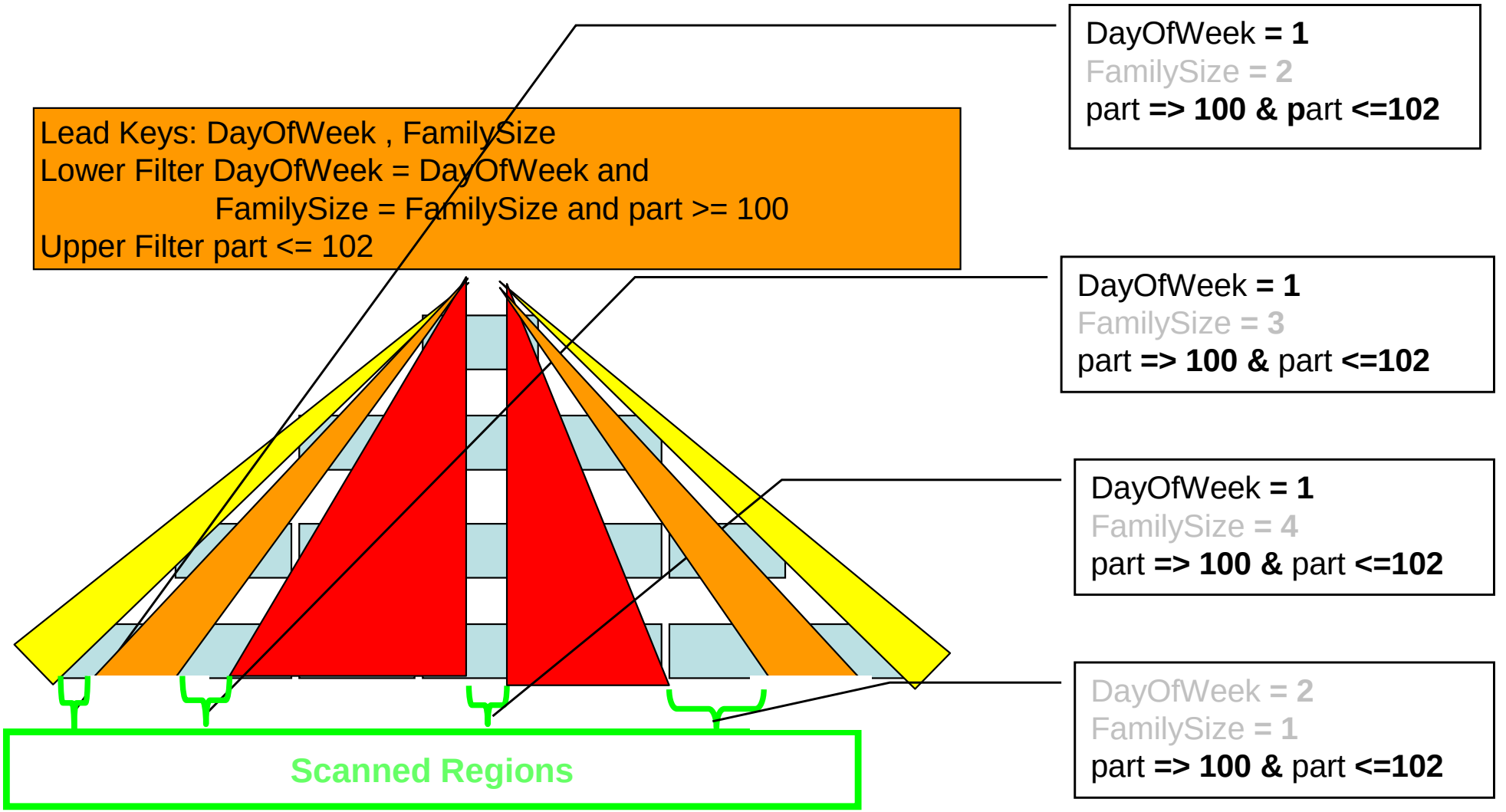
The following filters are applied after reading the data row
(FamilySize >= 2 and FamilySize <= 4) AND (part >= 100 and part <= 102)

Index Self-join functionality (with 11)

- Original query filters:
  (DayOfWeek >= 1 and DayOfWeek <= 5) and
  (FamilySize >= 2 and FamilySize <= 4) and
  (part >= 100 and part <= 102)
- Logically transformed query filters:
  DayOfWeek = 1 and FamilySize = 2 and part >= 100 and part <= 102
  UNION
  DayOfWeek = 1 and FamilySize = 3 and part >= 100 and part <= 102
  UNION
  DayOfWeek = 1 and FamilySize = 4 and part >= 100 and part <= 102
  UNION
  DayOfWeek = 2 and FamilySize = 2 and part >= 100 and part <= 102
  UNION
  DayOfWeek = 2 and FamilySize = 3 and part >= 100 and part <= 102



Using Index Self Join



Setup of Index Self Join Example

```
create raw table customer
    (
    dayofweek           integer,
    familysize          integer,
    part                integer,
    partdesc            char(60)
    ) lock mode row;

INSERT INTO customer  SELECT
   mod(A.tabid,7), mod(A.tabid,4),
    value+A.tabid, tabname
FROM sysmaster:systables A,
    sysmaster:sysshmhdr B,
    sysmaster:syscolumns
WHERE B.number=4;
```

```
create index ix1 on customer
(dayofweek,familysize,part,partdesc)

insert into customer values (1,3,77,“JUICE");
insert into customer values (2,3,77," JUICE ");
insert into customer values (3,3,78,“MILK");
insert into customer values (4,3,78," MILK ");
insert into customer values (5,3,78," MILK ");
insert into customer values (1,4,77," JUICE ");
insert into customer values (2,4,77," JUICE ");
insert into customer values (3,4,78," MILK ");
insert into customer values (4,4,78," MILK ");
insert into customer values (5,4,78," MILK ");
```

Using the directive to not use index self join

```
select {+ AVOID_INDEX_SJ(customer ix1) }
    count(*) from customer
where dayofweek >=1 and dayofweek <=5
      AND familysize>=3 and familysize<=4
      and partnumber  >76 and partnumber < 80;
```

Vs.

```
select count(*) from customer
where dayofweek >=1 and dayofweek <=5
      AND familysize>=3 and familysize<=4
      and partnumber  >76 and partnumber < 80;
```

Retrieving Query Results

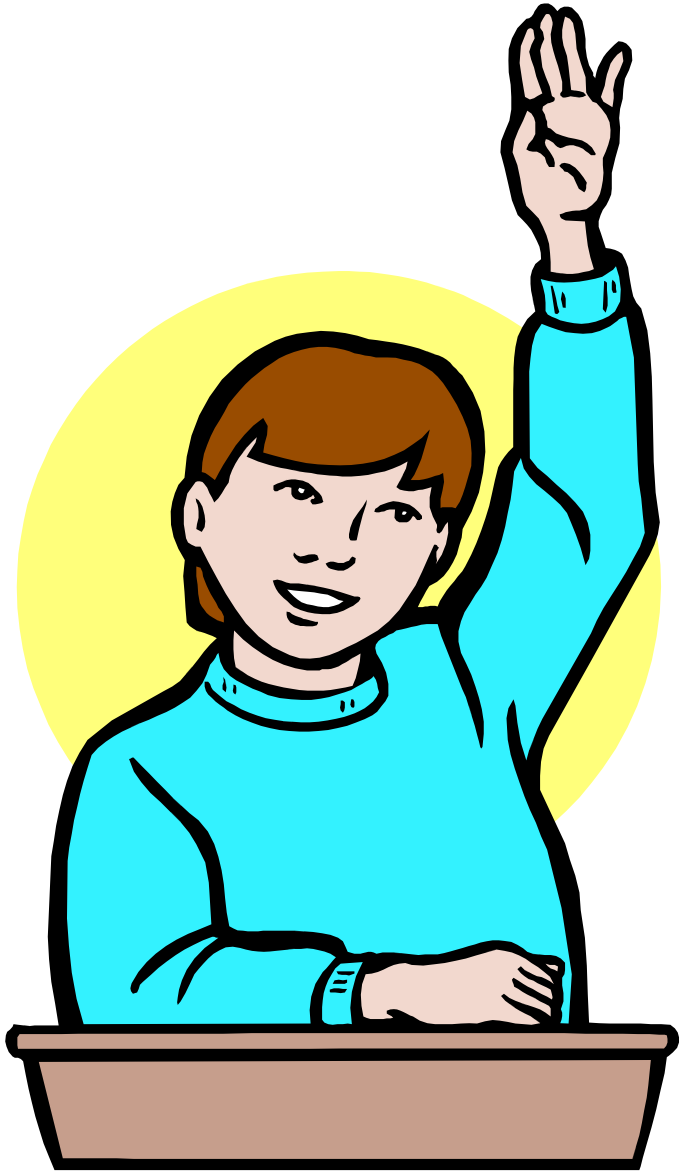
```
SELECT sql_runtime,sql_bfreads,sql_pgreads,sql_statement[1,30]
FROM sysmaster:syssqltrace
WHERE sql_sid = DBINFO('sessionid')
AND lower(sql_tablelist) matches "*customer*"
```

```
sql_runtime  sql_bfreads sql_statement

1.0920813      303236     select {+ AVOID
0.3031994       92435     select count(*)
```

- **Add distributions to the second column in the index**

```
update statistics high for table
customer(familysize) distributions only;
```

```
sql_runtime  sql_bfreads sql_statement

1.0920813     303236      select {+ AVOID
0.3031994      92435      select count(*)
0.0006977        142      select count(*)
```



Information Management

IBM

Data Management

Informix in a DSS environment

Alfatec Conference 2010
Joe ‘Celo’ Baric – jbaric@us.ibm.com