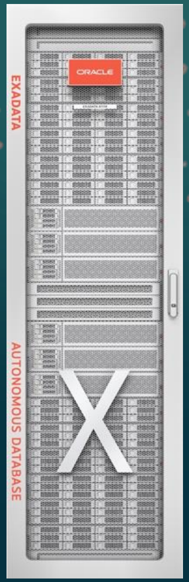
Exadata Database Machine

Storage Offload, SQL Offload, Smart Scan

Natarajan Shankar

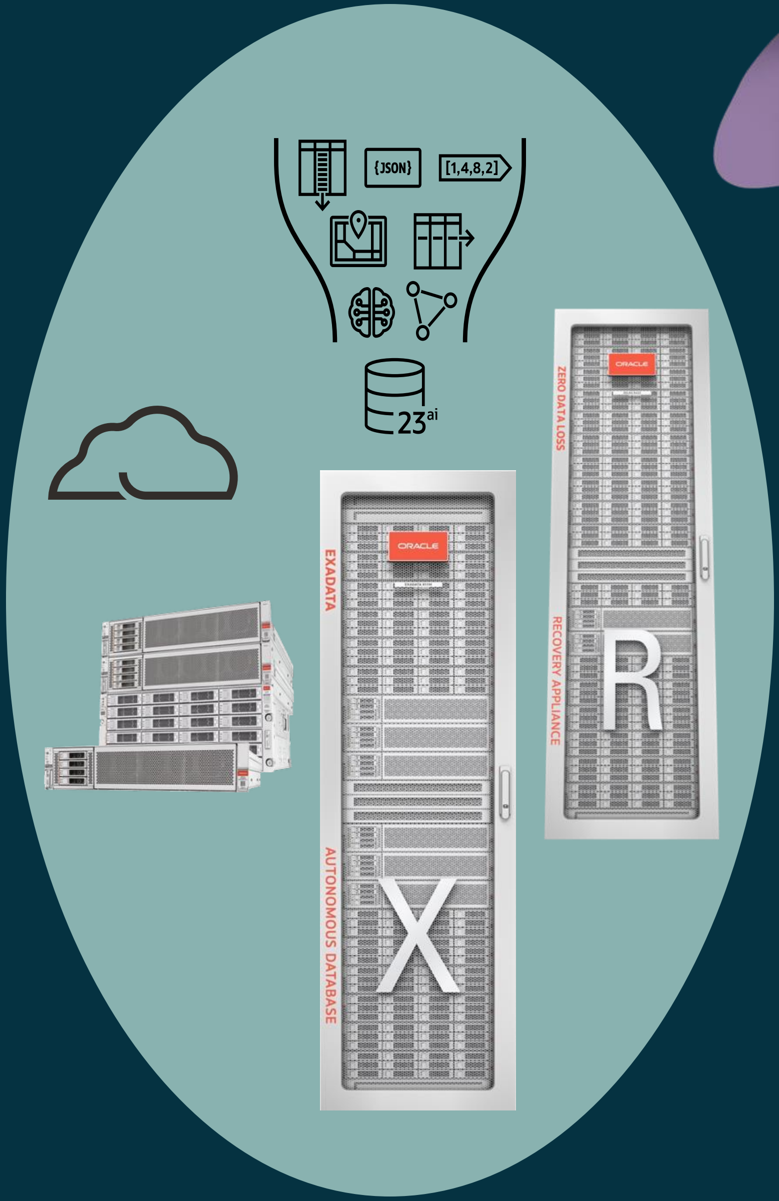
September 18, 2025



Oracle's Strategy for Database Systems

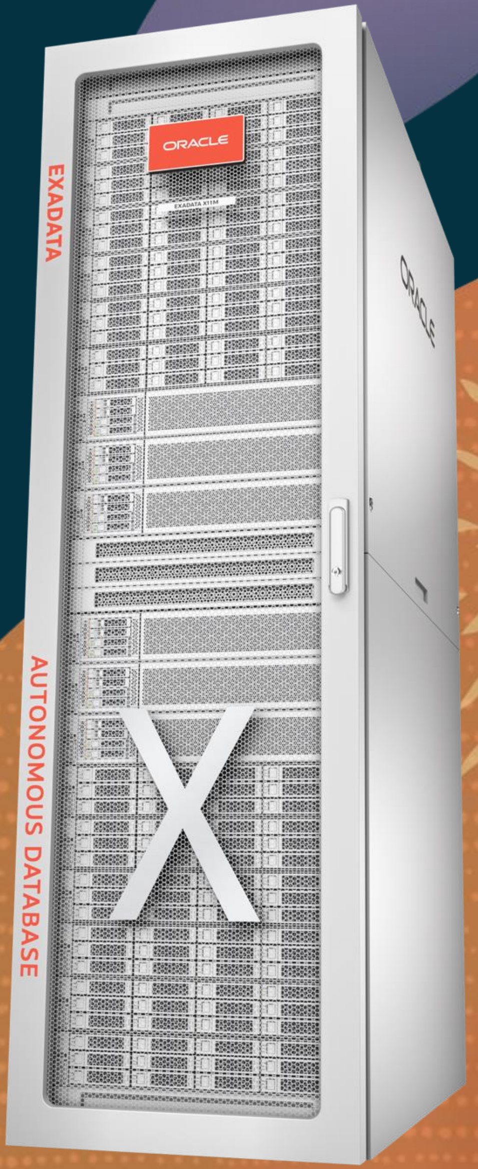
Build best-of-breed data and AI optimized hardware and software,
that are engineered to work together

Everywhere!



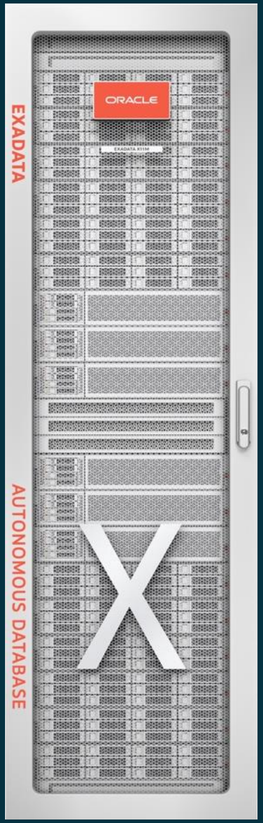
Exadata

Next Generation Intelligent Data Architecture



Exadata vision

Extreme performance, scalability, and availability for data and AI workloads, at a low cost



Data optimized hardware

Scale-out, data optimized compute, networking, and storage

Data intelligent software

Unique algorithms deliver extreme performance and efficiency for AI, analytics, and mission-critical apps, at any scale

Engineered to run everywhere

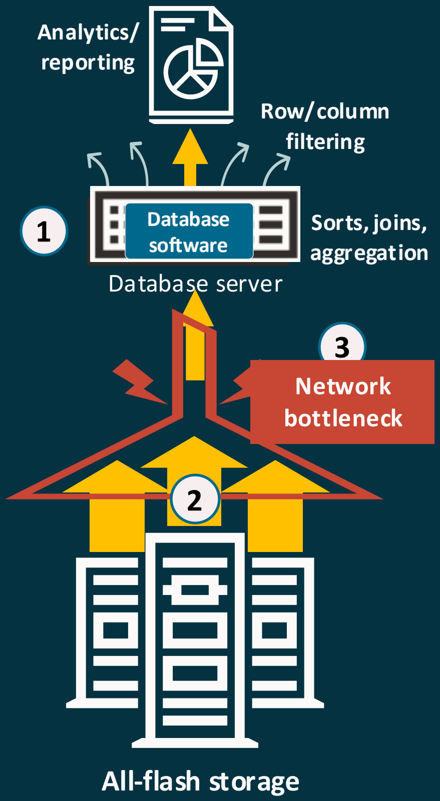
On-premises, Cloud@Customer, Oracle Cloud, and Multicloud

Breakthrough technology

Storage offloading

Non-Exadata database platforms

1. **ALL database work runs on database servers**: row/column filtering, sorts, joins, aggregation
2. Must send large, unfiltered data sets from storage
3. Modern flash storage overwhelms storage networks, **exposing the network bottleneck**



Effective Scan Performance
175 GB/s @ 192 flash drives[1]

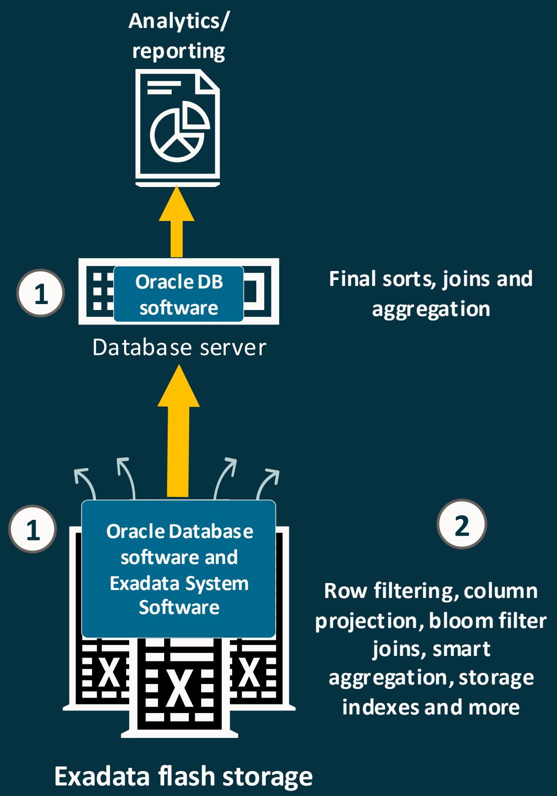
Exadata

1. Runs Oracle intelligent database software on **BOTH storage and database servers**
2. Processes data in storage, <u>before</u> sending results across the network, **eliminating the network bottleneck**
3. Exadata storage performance is fully realized

3

Effective Scan Performance
1,700 GB/s @ 68 flash drives[2]

2


Analytics/ reporting
1
Oracle DB software
Final sorts, joins and aggregation
Database server
1
Oracle Database software and Exadata System Software
2
Row filtering, column projection, bloom filter joins, smart aggregation, storage indexes and more
Exadata flash storage

Querying in a non-Exadata database environment

Inefficient data movement and analytics query performance

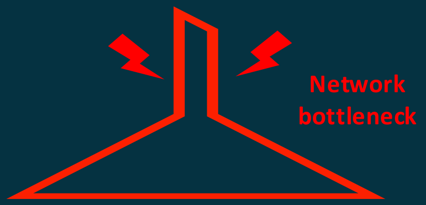
Data is retrieved from storage servers via the interconnect, filtered and processed at the database server

- The entire table is read from storage and sent to the database server, where query predicates (filters) are applied. Example query: `select count(*) from census where Age < 35;`
- This means that large, unfiltered data sets are transferred over the network, creating a performance bottleneck, and processed at the database server, consuming bandwidth, IOPS, and CPU

Every row is returned, filtering is done at the database server

ID	First Name	Last Name	Age	Gender
1	John	Doe	28	M
2	Jane	Smith	34	F
3	Alice	Johnson	23	F
4	Bob	Brown	45	M
5	Eve	Davis	36	F

Database servers are overwhelmed

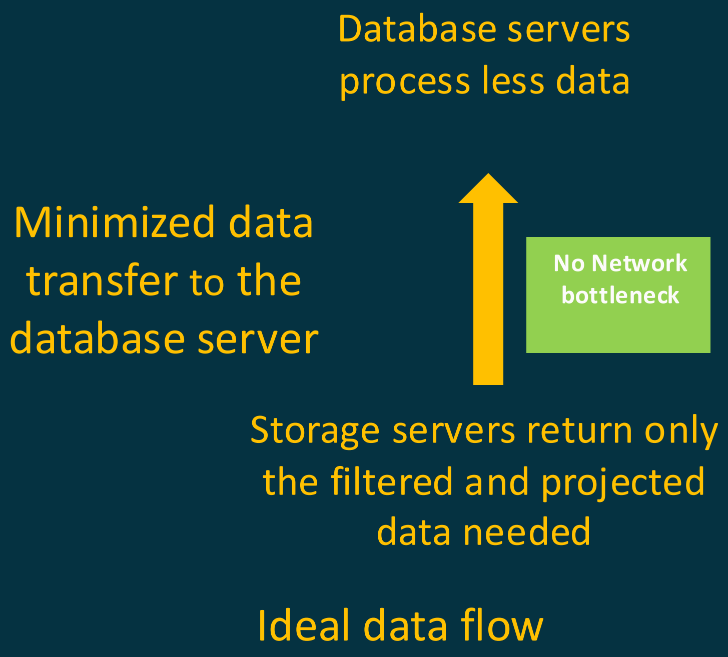


Storage networks are overwhelmed

So, what should ideally happen with queries?

Need an efficient pipeline that is data intelligent

- When only a small amount of data from a large table is needed for a query:
  - Understand the data layout in storage:
    - Know where not to search
    - Select which data to use
  - During data retrieval:
    - Only bring back the data rows relevant to the predicate
    - Only bring back the data columns that have been asked for
    - Transport data over a high-speed fabric
-
  - Process the data where it is:
    - Perform aggregations in storage
    - Minimize network I/O
    - Leverage unique Exadata efficiencies to make tasks like Joins efficient


Database servers process less data
Minimized data transfer to the database server
No Network bottleneck
Storage servers return only the filtered and projected data needed
Ideal data flow

A database platform should work smarter, not harder!

Eight efficiency features in Exadata

Unlock unmatched offload performance and throughput

Efficiency Features (Not an ordered list)	How It Works in Exadata Storage Servers	Key Benefit
Transparent Parallelism	Automatically distributed processing of queries	Reduces query times, scales dynamically
High-speed RoCE Fabric	Database and storage servers can directly access each other's memory	Enables fast, low-latency data transfers, no CPU or OS involvement
Storage Indexes	Maintains data range in memory	reduces I/O by ignoring irrelevant extents
Predicate Filtering	Evaluates clauses directly on storage servers before sending data to the database server	Minimizes data rows
Column Projection	Returns only the column values requested	Minimizes data columns
Bloom Filter based Joins	Unique filtering during Joins by applying Bloom filters	Uniquely boosts performance by filtering out unlikely matches at Storage Server
Smart Aggregation	SUM and GROUP BY SQL operations are smart scan enabled with in-memory columnar format	can be processed directly on the Exadata storage servers
Columnar Cache, Compression	Data is in a pure columnar format in flash or memory, only relevant data needs decompression	Faster queries and more efficient use of system resources

Decision process for Storage Offloading

Query Optimizer:

- Chooses an access path with full table scans, fast full index scans, or full bitmap index scans

Optimizer directs predicate and column analysis optimization to the storage server

- Filterable predicates (e.g., WHERE gender = ‘Female’)
- Required columns (e.g., SELECT First_Name, Last_Name)
    - These predicates are sent to storage servers

Execution plan has visual indicators of optimization: The query has been offloaded when the execution plan shows:

- TABLE ACCESS STORAGE FULL operation
- STORAGE-prefixed predicates in EXPLAIN PLAN output

Join optimization: For joins between large and small tables, bloom filters may be offloaded

Storage index utilization: The optimizer leverages storage indexes to skip I/O for irrelevant data regions

Parallelism – Analytics queries processed on multiple storage servers

Accelerate insights through in-parallel executed queries

Transparent Parallelism:

- On by default, no manual intervention or configuration required
- Concurrent processing across instances, final data aggregation at the database
- The user can provide a “PARALLEL” hint to the database
- Data transfer leverages high-speed RoCE networking
- Uses intelligent underlying storage-aware protocols

Intelligent Optimizer in Exadata (decides Parallel vs. Serial execution):

- Uses an understanding of data distribution to process queries

Performance Advantage:

- Linear performance gain with added storage cells
- Faster scans by eliminating unnecessary data transfer
- Automatically adjusts parallelism based on system load

Storage indexes

Accelerate analytics queries by eliminating irrelevant data region reads

Automatically built storage indexes track column min/max values

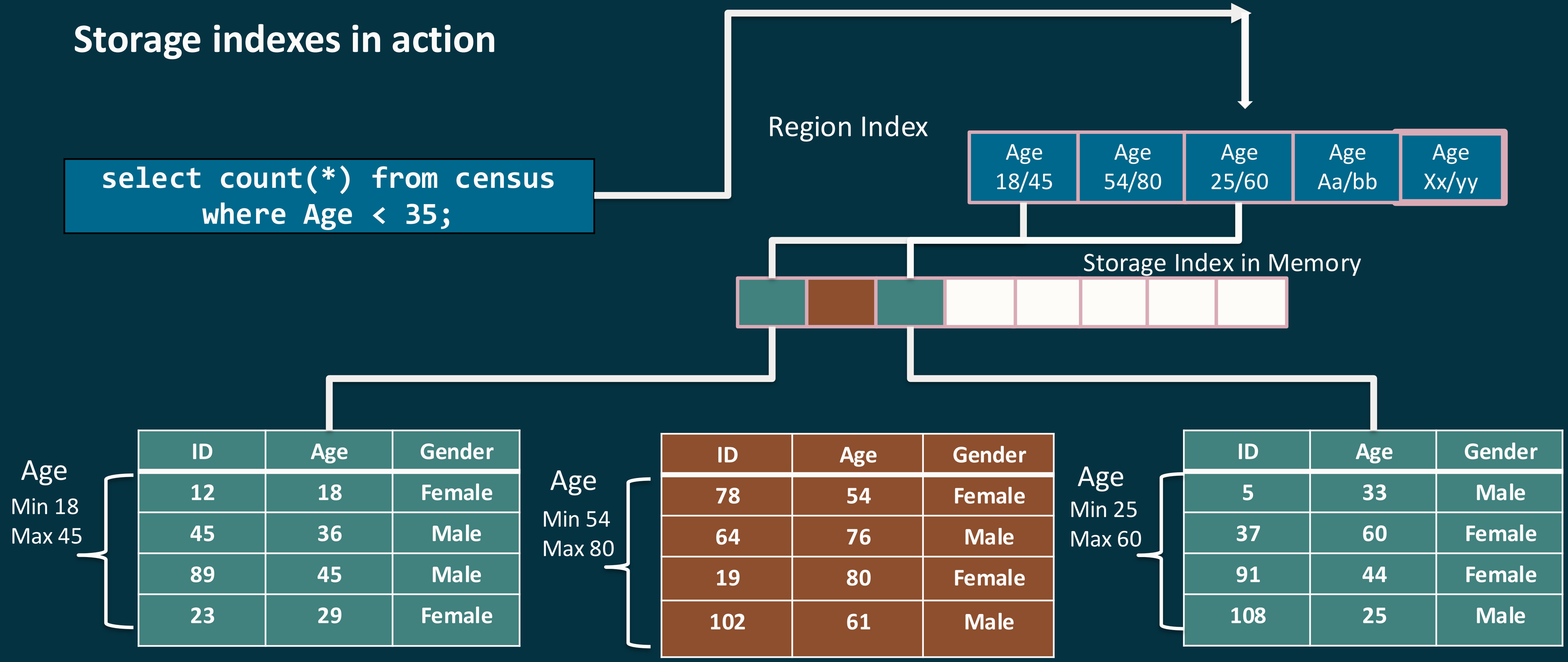
Storage Indexes hold information about a region of the disk on the Storage Cells

- an index for each 1 MB of disk space

- For the Census table example: Queries filtering on age or gender (predicate is WHERE age > 60) skip disk regions where the stored min/max values don't match the predicate
  - A traditional database machine would bring back the entire table to the buffer cache
  - Exadata will only bring back those extents where the data applies to the query, reducing IO
    - Storage indexes eliminate the need for physical IO on the disks that do not contain the data

Storage indexes in action

```
select count(*) from census
where Age < 35;
```



ID	Age	Gender
12	18	Female
45	36	Male
89	45	Male
23	29	Female

ID	Age	Gender
78	54	Female
64	76	Male
19	80	Female
102	61	Male

ID	Age	Gender
5	33	Male
37	60	Female
91	44	Female
108	25	Male

Predicate filtering – Only requested rows returned

Push query predicates to storage for smarter scans

Predicate filtering enables row filtering on storage servers

- Exadata Smart Scan offloads clause evaluation (eg. WHERE)
- Only rows matching the query condition are returned

Supported for a wide range of SQL operators:

- Operators such as =, !=, <, >, <=, >=, IS [NOT] NULL, EXISTS
- Logical operators like AND and OR

Predicate Filter is used in conjunction with Storage Indexes and Parallelism

Predicate filtering is automatically applied

- No manual intervention or configuration is required

Predicate filtering: #SmartRowDataReduction

```
SELECT First Name, Last Name FROM census WHERE Age BETWEEN 25 AND 35;
```

ID	First Name	Last Name	Age	Gender
1	John	Doe	28	Male
2	Jane	Smith	34	Female
3	Alice	Johnson	23	Female
4	Bob	Brown	45	Male
5	Eve	Davis	36	Female

Only two rows are returned

Filtration of rows happens at Storage Server

Column projection: Only requested columns returned

Reduce network load with intelligent column selection

Column Projection is the process of selecting and returning specific columns from a table in a query

- Effectively, a vertical subset of the data

```
SELECT Gender FROM census WHERE ID < 3;
```

Using the data example, return the intersection of ID and Gender

Imagine a table with 4K columns

- A traditional database would return all columns for all queries
- Exadata only returns the required columns in the SELECT statement, with the WHERE predicate applied at the storage server

Column projection: #SmartColumnDataReduction

Selected column

```
SELECT Gender FROM census WHERE ID < 3;
```

Selected Rows

ID	First Name	Last Name	Age	Gender
1	John	Doe	28	M
2	Jane	Smith	34	F
3	Alice	Johnson	23	F
4	Bob	Brown	45	M
5	Eve	Davis	36	F

Smart aggregation at storage server

Reduced data movement with in-storage aggregation

Offloads SQL operations like SUM and GROUP BY to Exadata storage servers
- allowing aggregations to be processed directly where the data resides

This reduces the amount of data sent to the database servers
- improves CPU utilization on those servers, often resulting in queries running up to 2x faster

The optimization is automatic for eligible queries
- leverages in-memory columnar formats
- enables high-performance analytics without manual intervention

Another Example:
SELECT dept, SUM(salary) FROM emp WHERE country = 'USA' GROUP BY dept;
Exadata storage servers compute the SUM(salary) for each department directly, sending only the summarized results to the database server

Bloom filter Joins between tables

Optimized to identify and reduce potential matching rows

Exadata generates a Bloom filter from the Join keys of the smaller table and sends it to the storage servers

Storage servers apply the Bloom filter while scanning the large table, quickly discarding rows that cannot possibly be part of the Join

Only rows that match (including some false positives) are sent to the database server for final Join processing.

This dramatically reduces I/O and network traffic by eliminating most non-matching rows at the storage layer

Columnar Cache – Leveraged by smart scan

Faster scans and aggregations

Accelerated Data Access:

- Check whether the required data is present in the columnar cache
- If available, data is read directly from the columnar cache (or XRMEM), bypassing slower disk

Efficient Analytic Processing:

- Process data in compressed and column-oriented format, ideal for queries that access only a subset of columns.

Happens in Parallel and provides In-Memory performance:

- Multiple storage servers perform Smart Scan operations in parallel, each accessing their local columnar cache
- Reduces CPU and I/O overhead
- With XRMEM, the most active columnar data is cached in memory, accessible via RDMA

Summary

Extreme performance through storage offload and intelligent data processing

- Storage Offload, SQL Offload, and Smart Scan collectively push query processing to Exadata storage servers, minimizing data movement and reducing load on database servers.
    - Query performance is maximized by offloading processing—including filtering, joins, and aggregation—to distributed storage nodes.
- High-performance by filtering, projecting, and aggregating data at the storage layer, resulting in faster query response times and greater scalability
- RDMA and RoCE enable low-latency, high-throughput communication between database and storage servers for efficient parallel processing
- Storage indexes, predicate filtering, and Bloom filters reduce unnecessary data transfer by ensuring only relevant rows and columns are returned.
- Hybrid Columnar Compression and columnar cache optimize storage utilization and speed up analytic workloads by reducing data size and I/O.

Smart Scan in action: Initial configuration

```
SQL> select count (*) from census;
   COUNT (* )
   ------------
   20000000

SQL> select count (*) from census where gender = 'Female';
   COUNT (* )
   ------------
   10000000
```

Smart Scan disabled: Baseline query performance

```
SQL> select count (*) from census where gender = 'Female' and age < 55;

  COUNT(*)
----------
        22

Elapsed: 00:00:00.56
Execution Plan
----------------------------------------------------------
Plan hash value: 562668156
-------------------------------------------------------------------------------------------
| Id  | Operation             | Name       | Rows   | Bytes      | Cost (%CPU)| Time      |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |            |      1 |     12     | 44702   (1)| 00:00:02 |
|   1 |  SORT AGGREGATE       |            |      1 |     12     |            |          |
|*  2 |   TABLE ACCESS FULL.  | CENSUS     |     27 |    324     | 44702   (1)| 00:00:02 |
-------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter("AGE"<55 AND "GENDER"='Female')
```

Smart Scan disabled: Interconnect metrics

```
SQL> SELECT a.name, b.value FROM v$sysstat a, v$mystat b WHERE a.statistic# = b.statistic#
 and (a.name in ('physical read total bytes', 'physical write total bytes', 'cell IO uncompressed bytes’)
         or a.name like 'cell physical%');

NAME                                                                   VALUE
---------------------------------------------------------------- ----------------------------
physical read total bytes                                           1349738496
physical write total bytes                                                   0
cell physical IO interconnect bytes                                 1349738496
cell IO uncompressed bytes                                                   0
cell physical write IO bytes eligible for offload                            0
cell physical write IO host network bytes written during offload             0
cell physical IO bytes saved during optimized file creation                  0
cell physical IO bytes saved during optimized RMAN file restore              0
cell physical IO bytes eligible for predicate offload                        0
cell physical IO bytes eligible for smart IOs                                0
cell physical IO bytes saved by columnar cache                               0

Elapsed: 00:00:00.01
```

Smart Scan disabled: Interconnect metrics

```
NAME                                                                    VALUE
---------------------------------------------------------------- ----------
cell physical IO bytes saved by storage index                             0
cell physical IO bytes added to storage index                             0
cell physical IO bytes sent directly to DB node to balance CPU            0
cell physical IO bytes processed for IM capacity                          0
cell physical IO bytes processed for IM query                             0
cell physical IO bytes processed for no memcompress                       0
cell physical IO interconnect bytes returned by smart scan                0
cell physical write bytes saved by smart file initialization              0
```

With Smart Scan enabled: Extreme query performance

```
SQL> select count (*) from census where gender = 'Female’ and age < 55;
  COUNT(*)
----------
  10000000
Elapsed: 00:00:00.11
Execution Plan
----------------------------------------------------------
Plan hash value: 562668156
---------------------------------------------------------------------------------------------
| Id  | Operation                 | Name      | Rows          | Bytes   | Cost (%CPU)| Time      |
---------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |           |             1 |       6 | 44702   (1)| 00:00:02 |
|   1 |  SORT AGGREGATE           |           |             1 |       6 |.           |          |
|*  2 |   TABLE ACCESS STORAGE FULL| CENSUS    |           10M|      57M| 44702   (1)| 00:00:02 |
---------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - storage("AGE"<55 AND "GENDER"='Female')
       filter("AGE"<55 AND "GENDER"='Female')
SQL>
```

Smart Scan enabled: Interconnect metrics

```
NAME                                                                    VALUE
---------------------------------------------------------------- ------------------
physical read total bytes                                          1349722112
physical write total bytes                                                  0
cell physical IO interconnect bytes                                     92144
cell IO uncompressed bytes                                             188416
cell physical write IO bytes eligible for offload                           0
cell physical write IO host network bytes written during offload            0
cell physical IO bytes saved during optimized file creation                 0
cell physical IO bytes saved during optimized RMAN file restore             0
cell physical IO bytes eligible for predicate offload              1349632000
cell physical IO bytes eligible for smart IOs                      1349632000
cell physical IO bytes saved by columnar cache                     0
```

Smart Scan enabled: Interconnect metrics

```
NAME                                                                  VALUE
---------------------------------------------------------------- ----------
cell physical IO bytes saved by storage index                    1349443584
cell physical IO bytes added to storage index                        131072
cell physical IO bytes sent directly to DB node to balance CPU            0
cell physical IO bytes processed for IM capacity                          0
cell physical IO bytes processed for IM query                             0
cell physical IO bytes processed for no memcompress                       0
cell physical IO interconnect bytes returned by smart scan             2032
cell physical write bytes saved by smart file initialization              0

19 rows selected.

Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------
Plan hash value: 4169710972
```

Smart Scan enabled: Check query efficiency

```
NAME                                                               VALUE
------------------------------------------------------------- ----------
cell physical IO bytes eligible for predicate offload          1349632000
cell physical IO interconnect bytes returned by smart scan           2032

Elapsed: 00:00:00.00
```

```
Efficiency of Smart Scan Interconnect
       = 1 -  (2032 / 1349632000) * 100

  = 99.99%
```