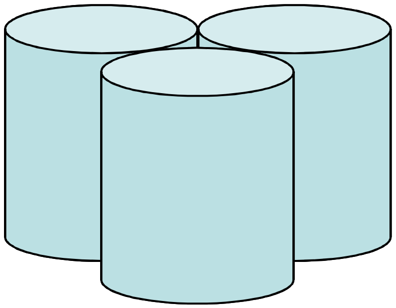
Relational Cloud: A Database-as-a-Service for the Cloud

C. Curino, E. P. C. Jones, R. A. Popa, N. Malviya, E. Wu,
S. Madden, H. Balakrishnan, N.Zeldovich

Database as a Service

- Transactional, Relational DB Service
  - hide complexity
  - exploit resource pooling
  - increase automation
  - (both for *private* and *public* cloud)



Existing Services

- Existing Commercial DB Services
  – Amazon RDS, SQL Azure *(and many others)*
- What they got right
  – simplified provisioning/deployment
  – reduced administration/tuning headaches
- What is still missing?
  – workload placement (to reduce hw cost)
  – automatic partitioning
  – Encryption (to achieve data privacy)

Relational Cloud Architecture

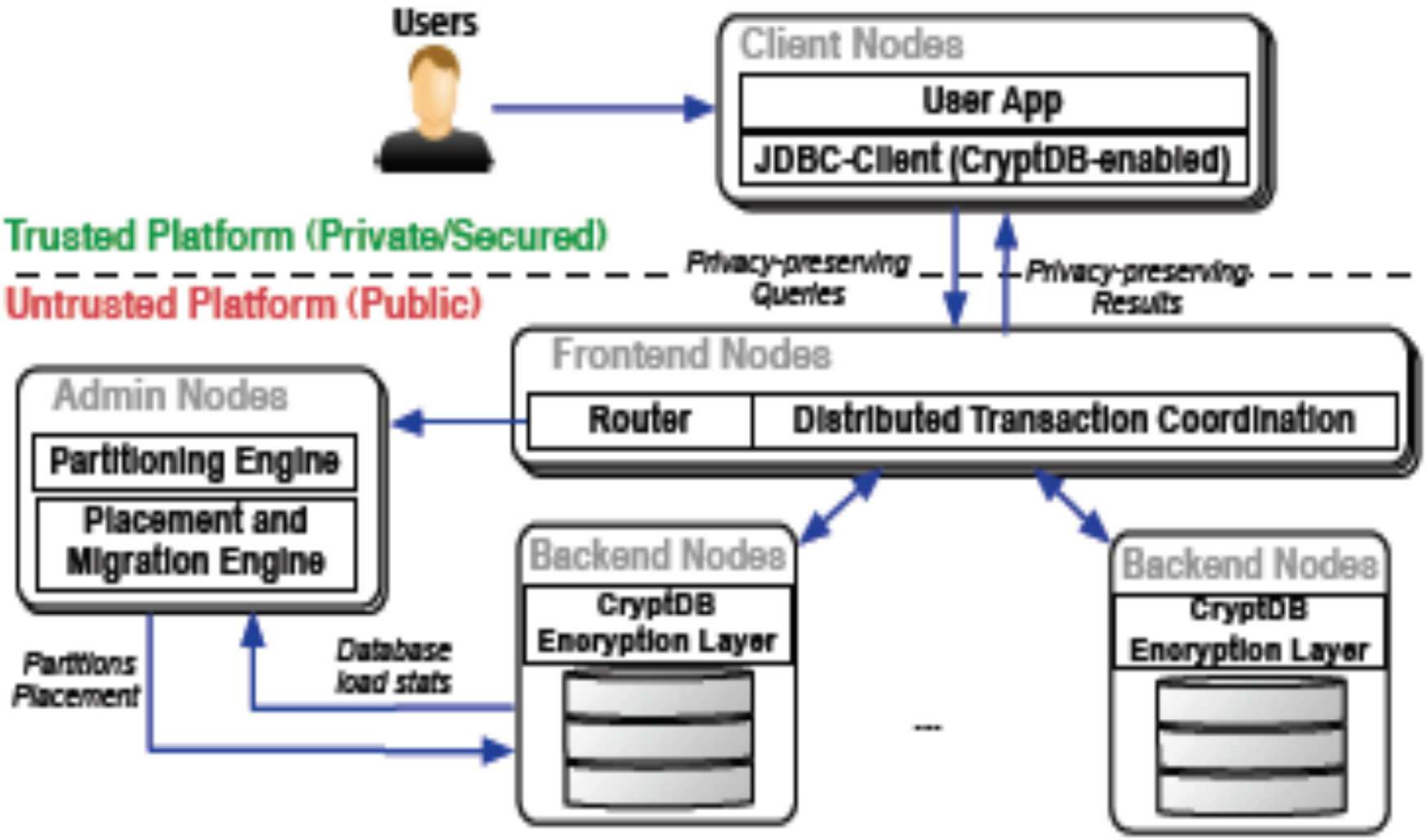
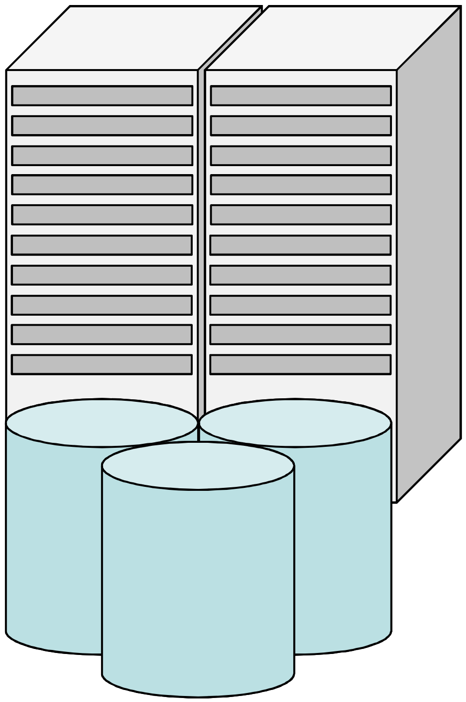


Figure 1: Relational Cloud Architecture.

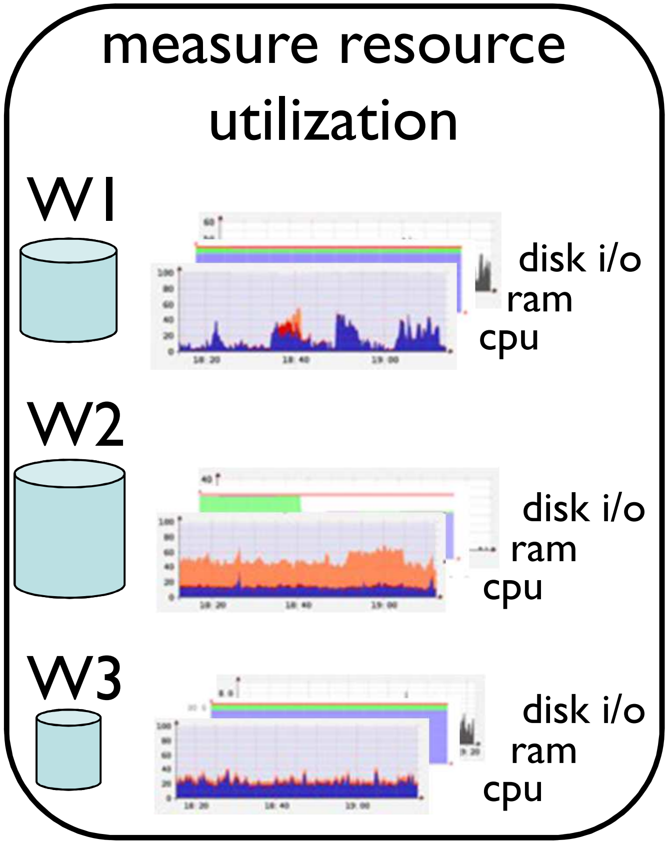
Workload Placement

- Why
  - Load balancing
  - High Performance

- Problem Definition
  - Allocate workloads to servers in a way that
    1. minimizes number of servers used
    2. balances load across servers
    3. maintains performance



Workload Placement

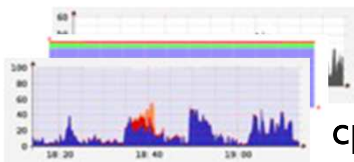


DBMS's tend to use all available resources

Workload Placement

measure resource utilization

W1



disk i/o
ram
cpu

W2

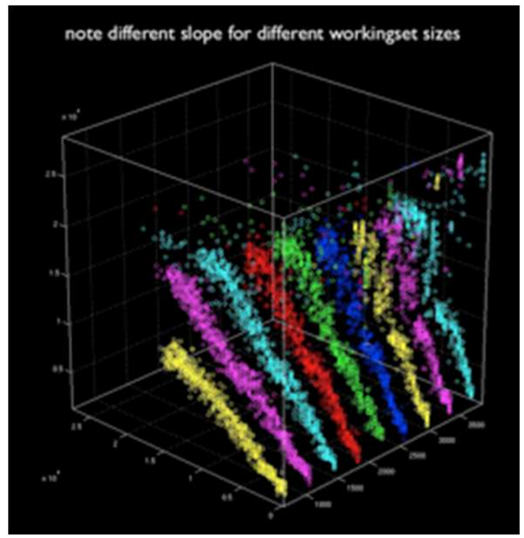
disk i/o
ram
cpu

W3

disk i/o
ram
cpu

estimate combined load

numerical models



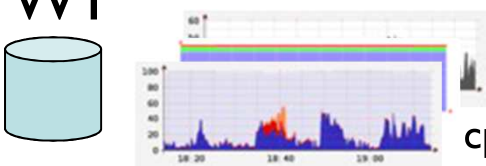
DBMS's tend to use all available resources

resource non-linearities

Workload Placement

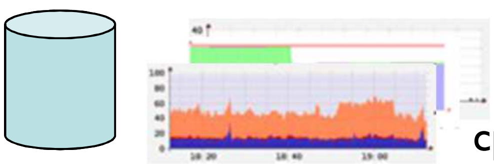
measure resource utilization

W1



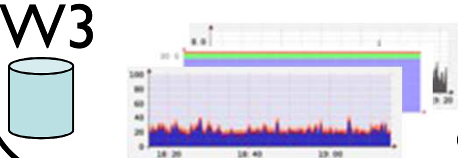
disk i/o
ram
cpu

W2



disk i/o
ram
cpu

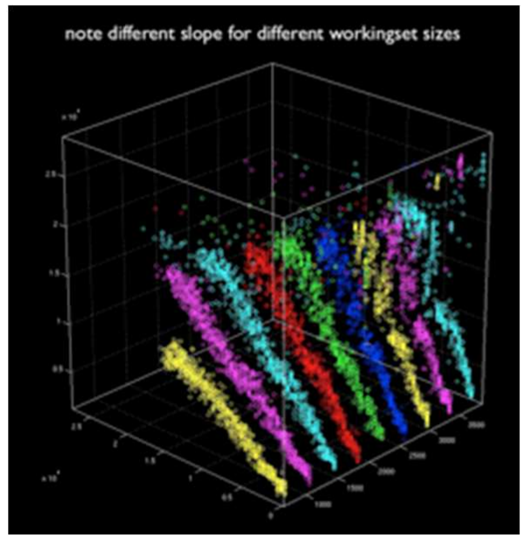
W3



disk i/o
ram
cpu

estimate combined load

numerical models



find optimal assignment

non-linear programming

minimize$_x$ $\sum_{tj}(e^{(\sum_t C_{ti} * x_{tj})} * signum(\sum_i x_{ij}));$

subject to $\forall i \sum_j x_{ij} = R_i;$

$\forall j\ max_t(\sum_i CPU_{ti} * x_{ij}) < MaxCPU_j;$

$\forall j\ max_t(\sum_i MEM_{ti} * x_{ij}) < MaxMEM_j;$

$\forall j\ diskModel(DISK_{ti}, x_{ij}) < MaxDISK_j;$

$\dots$

additional placement constraints

$\forall i, j x_{ij} \in N; 0 \leq x_{ij} \leq 1$

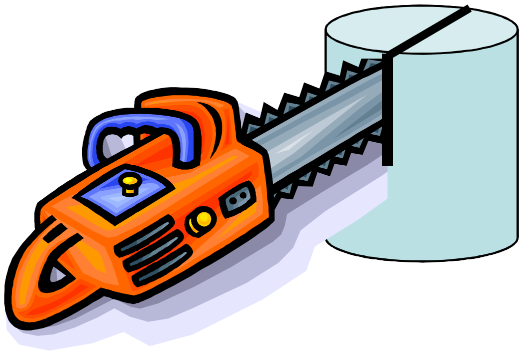
DBMS's tend to use all available resources

resource non-linearities

non-linear constraints and objective function

Partitioning

- Why
  - Scalability
  - Manageability

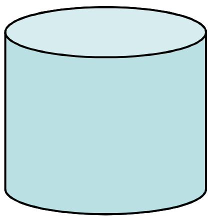


- Problem Definition
  - Partition the database into *N* chunks in a way that maximizes the workload performance

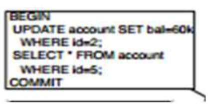
Graph-based Partitioning

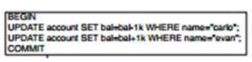
Input

Database

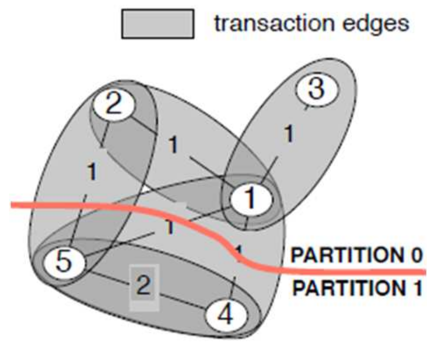


Workload Trace





Graph Representation



Explanation

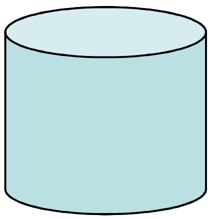
P: 0	ID < 4
P: 1	ID >= 4

Input
(logs + preprocessing)

Graph-based Partitioning

Input

Database

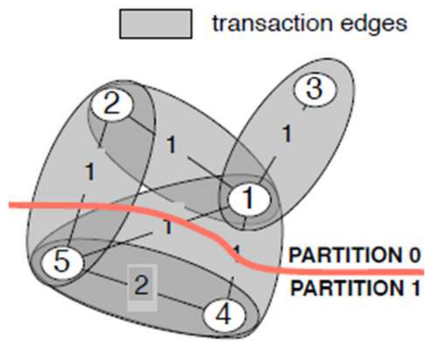


Workload Trace

```
BEGIN
 UPDATE account SET bal=60k
  WHERE id=2;
 SELECT * FROM account
  WHERE id=5;
COMMIT
```

```
BEGIN
UPDATE account SET bal=bal-1k WHERE name="carlo";
UPDATE account SET bal=bal+1k WHERE name="evan";
COMMIT
```

Graph Representation



Explanation

P: 0	ID < 4
P: 1	ID >= 4

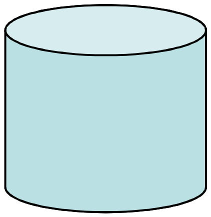
Input
(logs + preprocessing)

Graph Partitioning
(METIS)

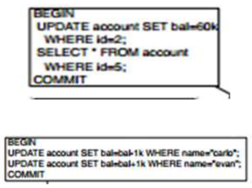
Graph-based Partitioning

Input

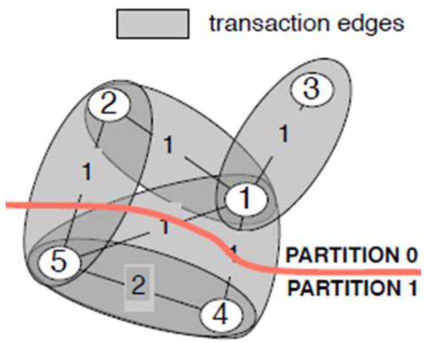
Database



Workload Trace



Graph Representation



Explanation

P: 0	ID < 4
P: 1	ID >= 4

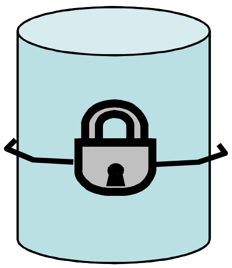
Input
(logs + preprocessing)

Graph Partitioning
(METIS)

Classification
(Decision Tree)

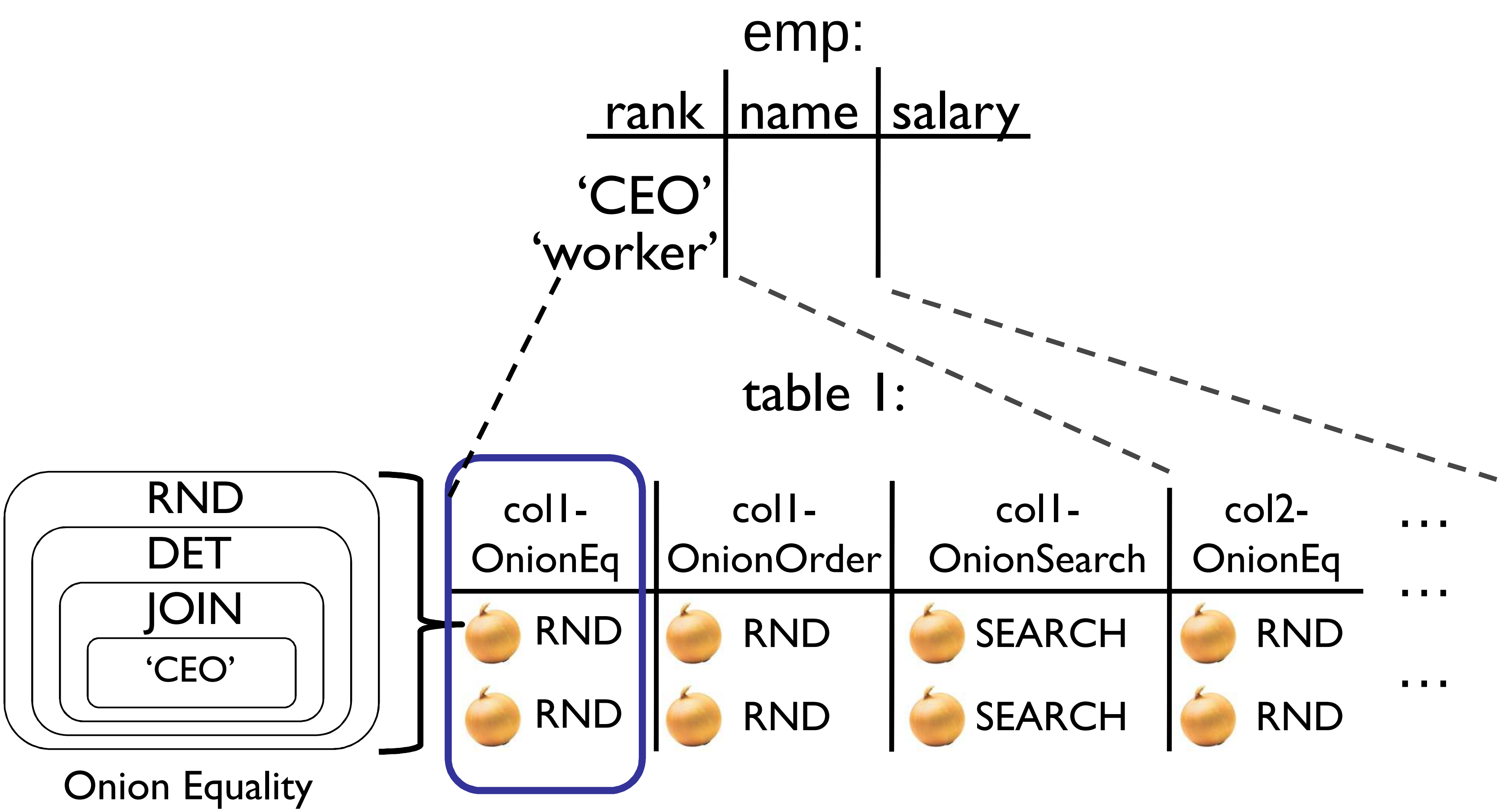
Encryption

- Why
  - Data Privacy



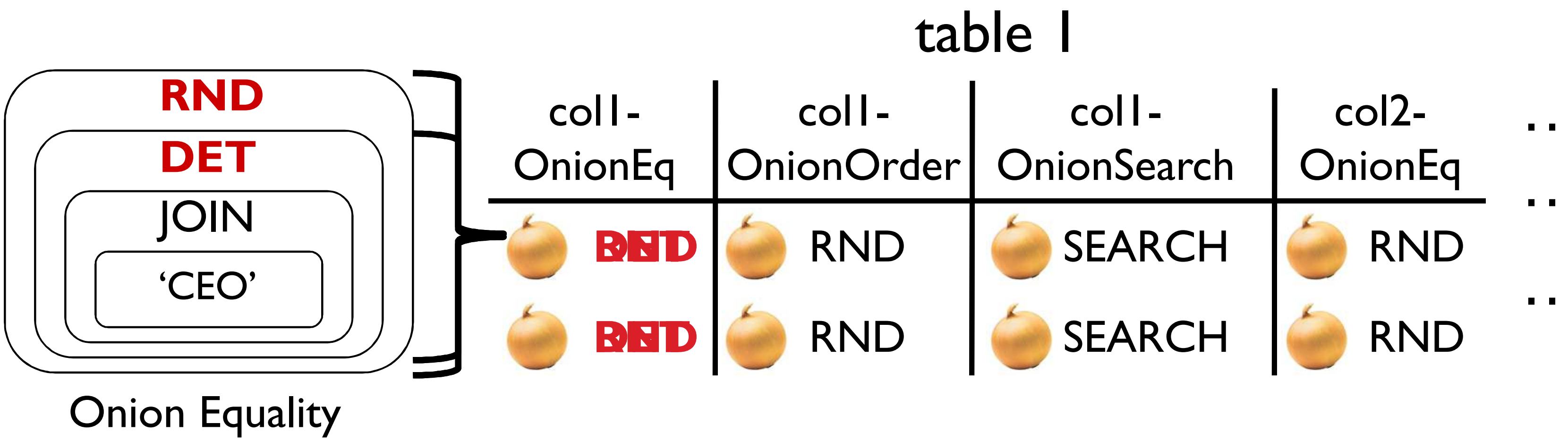
- Problem Definition
  - Minimize confidential info released to server $\Leftrightarrow$ Efficiently execute queries
  - Minimize the amt of data leaked when application server is compromised

Onion Encryption



SELECT * FROM emp WHERE rank = 'CEO';

Onion Encryption



SELECT * FROM emp WHERE rank = 'CEO';

UPDATE table1 SET col1-OnionEq =
Decrypt_RND(key, col1-OnionEq);

SELECT * FROM table1 WHERE col1-OnionEq = xda5c0407;

Conclusions

- Database as a Service has *real* potential
- Key Features to fully enable DBaaS
  - Workload Placement
  - Automatic Partitioning
  - Provable Privacy