IIT School of Applied Technology

ILLINOIS INSTITUTE OF TECHNOLOGY

information technology & management

526 Data Warehousing

March 23, 2016

Week 10 Presentation

Introduction to SQL Optimization for DW

Keys to SQL Optimization

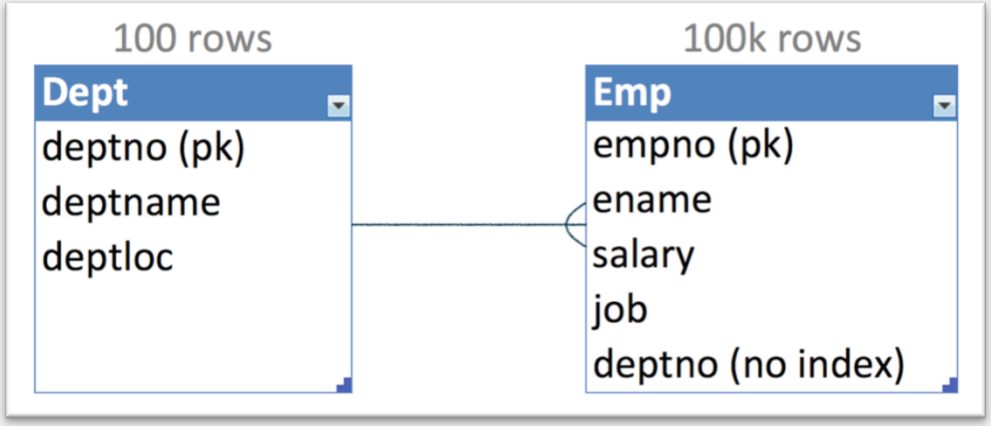
"Whoever touches the least number of blocks wins"

- Understanding Optimizer Behaviors
- Paradigm Shift: Procedure → Set
- SQL As an Application
  - Not Just As a Statement
- Clean and Simple Data Modeling

ITMD - 526

Introduction to SQL Optimization for DW

Understanding Optimizer Behaviors



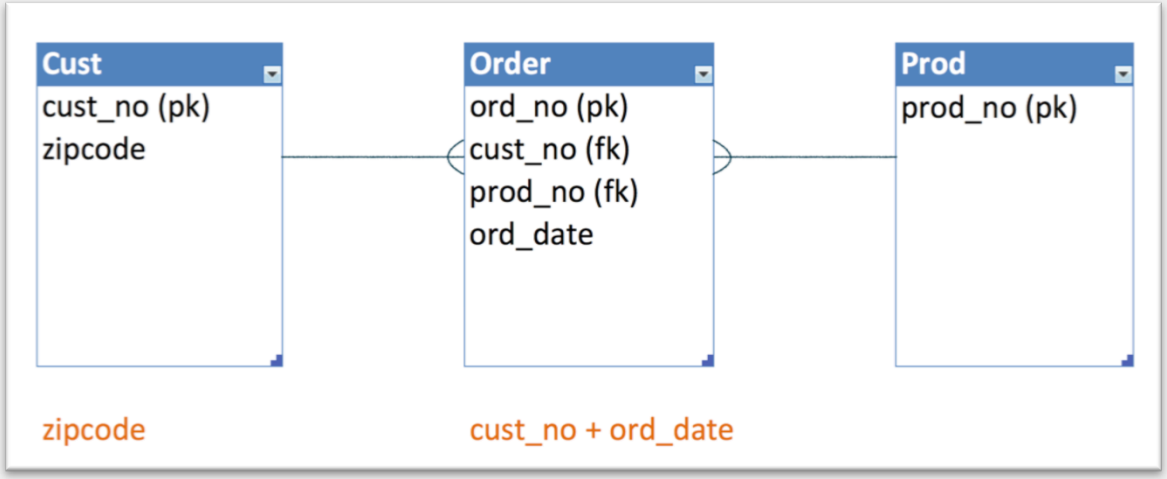

```
SELECT b.deptname, a.empno,
       a.ename, a.sal, …
  FROM emp a, dept b
 WHERE a.deptno = b.deptno
```

- Which to use as a driving table when using nested loop join?
  - emp: `100 dept rows + 100k emp rows`
  - dept: `100 dept rows X 50k emp rows`

- Optimizer "generally" makes smart decisions

ITMD - 526

Introduction to SQL Optimization for DW
Understanding Optimizer Behaviors (cont'd)




```
SELECT b.cust_name, a.ord_date,
       a.ord_amount, ...
  FROM order a
 INNER JOIN cust b
    ON a.custno = b.custno
 WHERE b.zipcode = :v1
   AND a.ord_date LIKE '201512%';
```

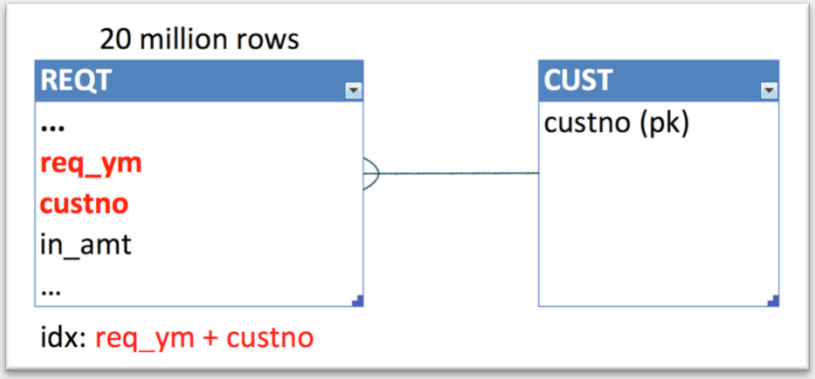
VS

```
SELECT b.cust_name, a.ord_date,
       a.ord_amount, ...
  FROM order a
 LEFT OUTER JOIN cust b
    ON a.custno = b.custno
 WHERE b.zipcode = :v1
   AND a.ord_date LIKE '201512%';
```

- Both queries return the same result
- But they may have very difference performance

ITMD - 526

Introduction to SQL Optimization for DW
Understanding Optimizer Behaviors (cont'd)




```
UPDATE reqt x
   SET in_amt = in_amt + :amt
 WHERE req_ym = '201506'
   AND custno IN
         (SELECT custno
            FROM cust y
           WHERE pay_cust = :cust
             AND x.cust= y.cust)
```

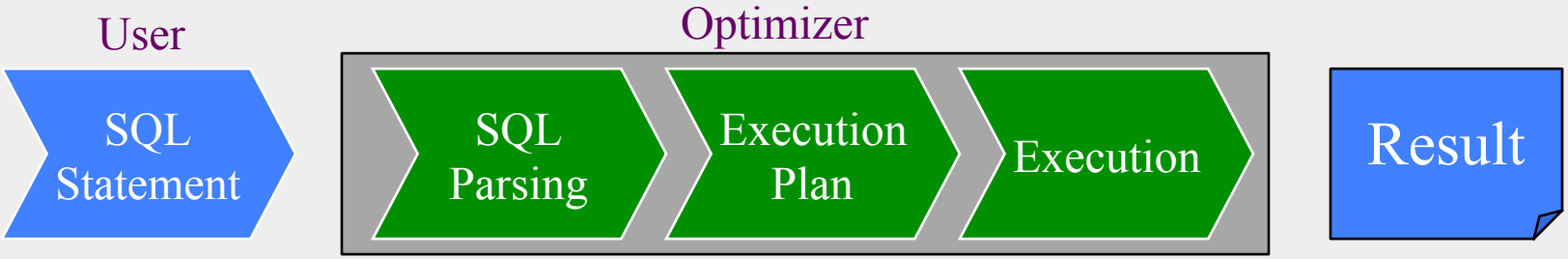
VS

```
UPDATE reqt x
   SET in_amt = in_amt + :amt
 WHERE req_ym = '201506'
   AND custno IN
         (SELECT custno
            FROM cust y
           WHERE pay_cust = :cust)
```

- Both queries return the same result
- But the first query may have severe performance problem when using nested loop

Introduction to SQL Optimization for DW

The Paradox of SQL for RDB



- User specifies conditions and results via SQL
- Optimizer generates execution plans
- The execution plans can produce hugely different performances
- Users must provide strategic factors for the optimizer to use in establishing execution plan

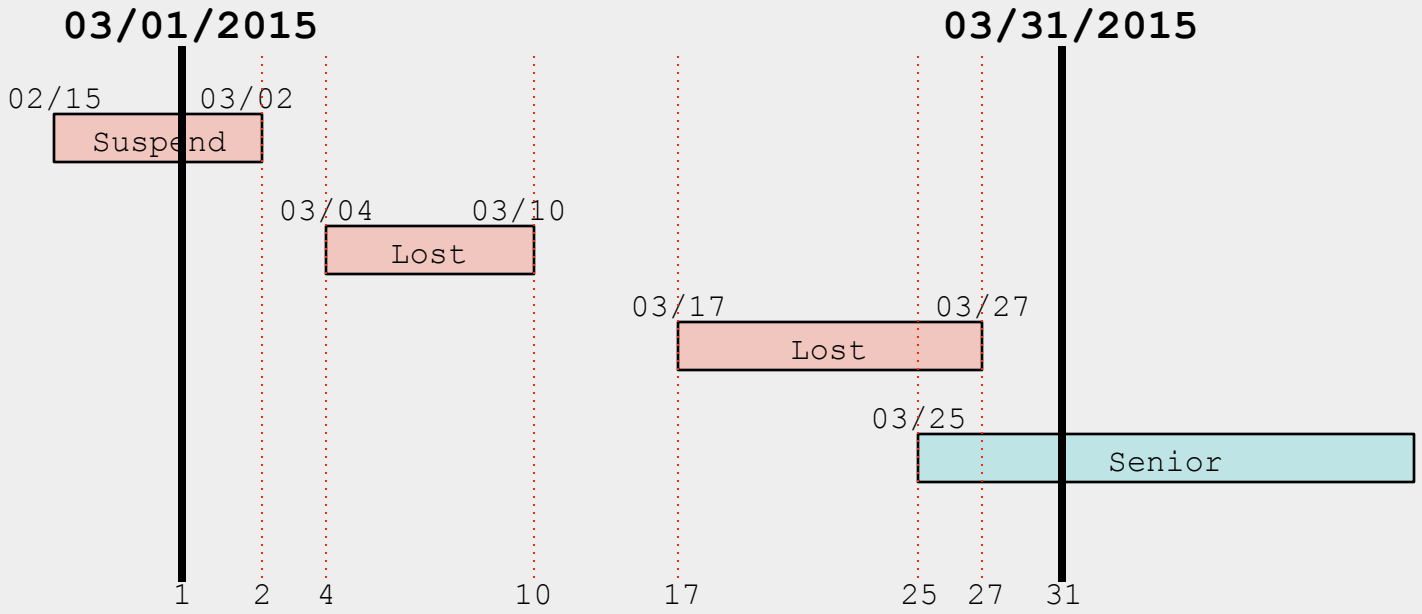
- SQL is Both Easy and Difficult
  - SQL is quite easy to learn
  - The optimizer's behavior can go out of control

ITMD - 526

Introduction to SQL Optimization for DW

Paradigm Shift: Procedure → Set

- Monthly Billing of a Telecommunication Company



- 1st ~ 2nd: 90% discount
- 4th ~ 10th: 90% discount
- 17th ~ 25th: 90% discount
- 25th ~ 27th: 90% discount * 20% discount
- 27th ~ 31st: 20% discount

ITMD - 526

Introduction to SQL Optimization for DW

Paradigm Shift: Procedure → Set (cont'd)

- Input Table

EMPLID	DEPTID	PAYGROUP	EMPL_STATUS	POSITION_NBR	JOBCODE	EFFDT	EFFSEQ
1234567	331600	BIT	A		100029	20030421	0
1234567	331600	BIR	A	00036477	102590	20031202	0
1234567	331600	BIR	A	00036477	102590	20040829	0
1234567	331600	BIR	A	00036477	102590	20040901	0
1234567	331600	BIR	A	00036477	100278	20041201	0
1234567	331600	MON	A	00036477	100278	20041201	1
1234567	331600	MON	A	00036477	100278	20050901	0
1234567	331600	MON	A	00036477	100278	20050923	0
1234567	331600	MON	A	00036477	104342	20120201	0
1234567	331600	MON	A	00036477	104342	20120901	0
1234567	331600	MON	A	00036477	104342	20130101	0
1234567	331600	MON	A	00036477	104342	20130901	0
1234567	791200	MON	A	00055668	213000	20140106	0
1234567	791200	MON	A	00055668	213000	20140601	0
1234567	791200	MON	A	00055668	213000	20140901	0
1234567	791200	MON	T	00055668	213000	20150720	0
1234567	791200	MON	A	00055668	213000	20150814	0
1234567	791200	MON	A	00055668	213000	20150908	0

Introduction to SQL Optimization for DW

Paradigm Shift: Procedure → Set (cont'd)

- Output Result

EMPLID	POSITION_NBR	PAYGROUP	POS_XFR_TO	EFFDT_FROM	EFFDT_TO
1234567		BIT	00036477	21-APR-2003	01-DEC-2003
1234567	00036477	BIR		02-DEC-2003	30-NOV-2004
1234567	00036477	MON	00055668	01-DEC-2004	05-JAN-2014
1234567	00055668	MON		06-JAN-2014	19-JUL-2015
1234567	00055668	MON		14-AUG-2015	01-JAN-2999

- Key: Reduction to Common Denominator

Introduction to SQL Optimization for DW

Paradigm Shift: Procedure → Set (cont'd)

- Procedural approach requires unnecessary DMBS calls and network overhead

```
declare c1 cursor -- 100k rows
select key, a, b, ....
  from trx
 where trx.date = sysdate;

open c1;
for (;;)
{
    fetch c1 into :v1, :v2, ....

    update master_table set field1 = :v1
     where key = :a_condition;

    if (sqlca.sqlcode == 1403)
      insert into master_table values (:v1, :v2, ...)

    commit work;
}
close c1;
```

100k oci calls

100k oci calls

10k oci calls

Introduction to SQL Optimization for DW

Paradigm Shift: Procedure → Set (cont'd)

- Identify a problem with the query on the left

```
SELECT COALESCE(p.name,c.name) name
  FROM person p,
       corporate c,
       account a
 WHERE a.smart_col = 'smart val'
   AND ( (a.type = 'P'
          AND a.id = p.id)
          OR
         (a.type = 'C'
          AND a.id = c.id)
       )
```

VS

```
SELECT p.name name
  FROM person p, account a
 WHERE a.smart_col = 'smart val'
   AND a.type = 'P'
   AND a.id = p.id
 UNION ALL
SELECT c.name name
  FROM corporate c, account a
 WHERE a.smart_col = 'smart val'
   AND a.type = 'C'
   AND a.id = c.id;
```

- Injecting procedural approach into an SQL can cause high performance/maintenance cost

Introduction to SQL Optimization for DW

SQL as An Application

"I was once asked in a question-and-answer session what I thought was the most underutilized Oracle Database feature. I responded almost immediately with the answer "SQL." The questioner then asked me, "But what do you mean—everyone uses SQL all the time." My response was that everyone uses very, very simple SQL and avoids 99 percent of its actual capabilities."

Source: http://www.oracle.com/technetwork/issue-archive/2011/11-sep/o51asktom-453438.html

- 99% of SQL developers work with 1% of its capability

Introduction to SQL Optimization for DW

SQL as An Application

	ITEM	TOTAL	JAN	FEB	MAR	APR	MAY	JUN
1	Sales Total	$45,000.00	$10,000.00	$20,000.00	$15,000.00	…	…	…
2	Production Cost Total	$38,000.00	$9,000.00	$18,000.00	$11,000.00	…	…	…
3	**Total Sales Profit**	$7,000.00	$1,000.00	$2,000.00	$4,000.00	…	…	…
4	Distribution Cost	$330.00	$100.00	$110.00	$120.00	…	…	…
5	General Management Expenses	$334.00	$100.00	$111.00	$123.00	…	…	…
6	Direct R&D Cost	$423.00	$150.00	$130.00	$143.00	…	…	…
7	Interest Cost	$481.00	$163.00	$121.00	$197.00	…	…	…
8	Non-Operating Revenue	$416.00	$110.00	$156.00	$150.00	…	…	…
9	Non-Operating Expense	$484.00	$176.00	$186.00	$122.00	…	…	…
10	Home Office Cost	$380.00	$111.00	$135.00	$134.00	…	…	…
11	**Total Direct Cost**	$2,016.00	$690.00	$637.00	$689.00	…	…	…
12	**Contribution Margin**	$4,984.00	$310.00	$1,363.00	$3,311.00	…	…	…
13	Indirect Cost	$1,060.00	$322.00	$270.00	$468.00	…	…	…
14	**Ordinary Profit**	$3,924.00	$12.00	$1,093.00	$2,843.00	…	…	…

- Line 3 = 1 – 2
- Line 11 = 4 + 5 + 6 + 7 – 8 + 9 + 10
- Line 12 = 3 – 11
- Line 14 = 12 – 13

Introduction to SQL Optimization for DW

Questions?