accenture

HOW NOT TO BUILD A(N AUTONOMOUS) DATA WAREHOUSE

DAVID KURTZ
UKOUG TECH2018

WHO AM I

Accenture Enkitec Group

Performance tuning

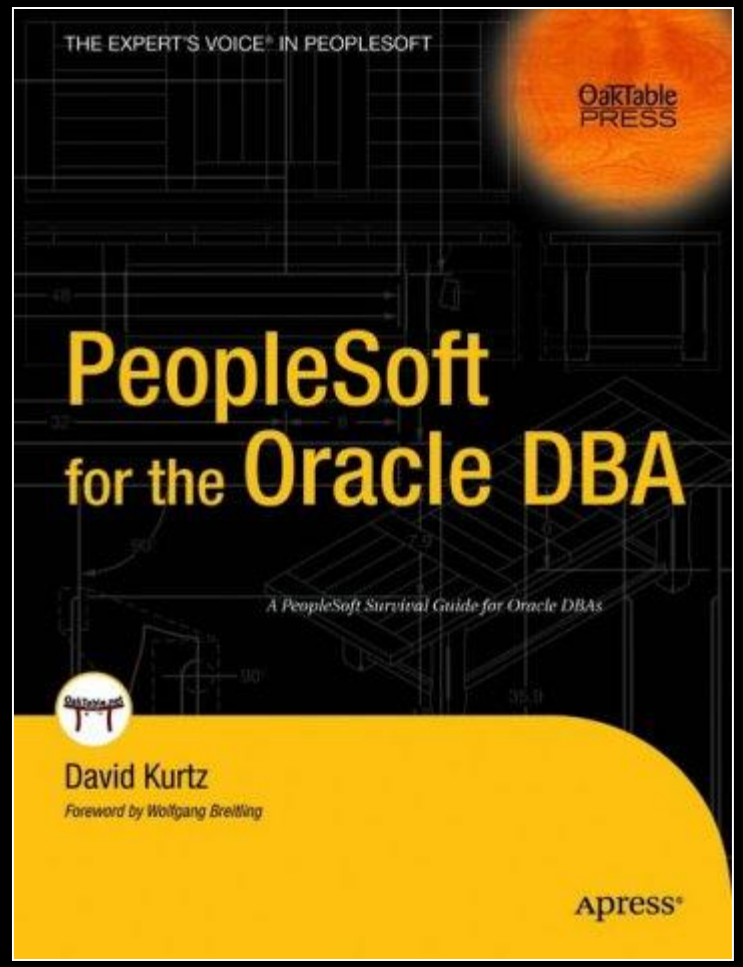
- PeopleSoft ERP
- Oracle RDBMS

Book

- www.go-faster.co.uk
- blog.psftdba.com

Oak Table

ORACLE ACE Director



SESSION OBJECTIVES

- Advice to help you make better decisions, earlier.

- Tell the Optimizer the Truth
  - (if you lie to it, don't expect it to work properly)

- The Fastest way to do anything is not to do it at all.
  - Save work by not doing unnecessary things.

AGENDA

Mistakes

- Lack of Foreign Keys
- Effective Dated Dimension/Facts
- Dates that are not dates

Other Considerations

- Indexing: Star Transformation –v- Full Scan/Bloom Filter
- Snowflaking & Lost Skew

Engineered Systems

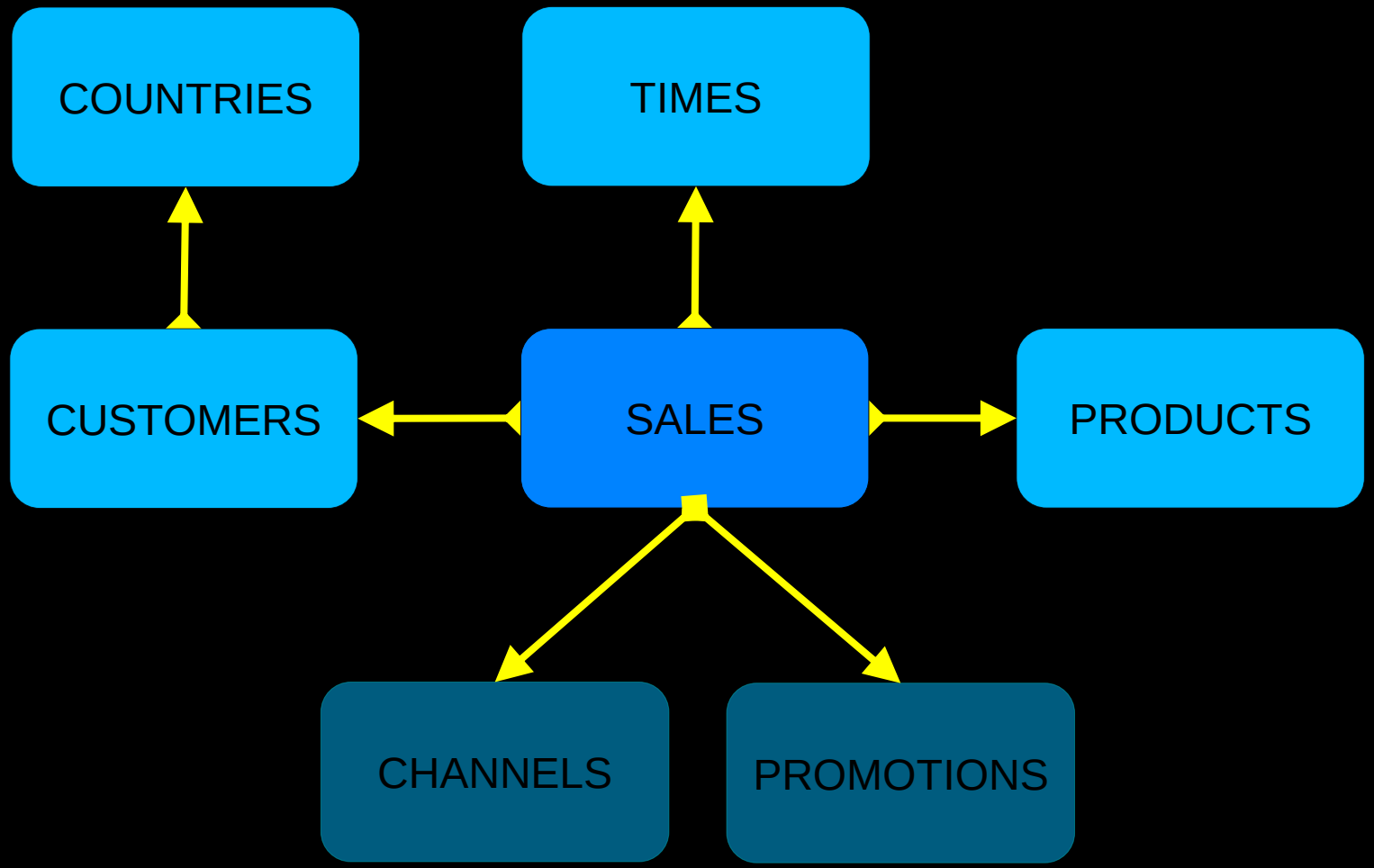
Autonomous Data Warehouse Cloud

SAMPLE DATA MODEL

ORACLE DEMO SALES HISTORY SCHEMA (SH)

- SH schema described in the Oracle documentation
- You can download the installation scripts from Github.
- It is available by default in ADWC
  - but in a slightly different form.

- I will take examples from real life and recreate them on this data model.
  - At times I will deliberate break parts of this model.



HOW NOT TO BUILD A(N AUTONOMOUS) DATA WAREHOUSE:

MISTAKE 1: LACK OF FOREIGN KEYS

WHAT IS A FOREIGN KEY?

- A column (or group of columns) on one table that uniquely identifies a single row on a parent table.
- If enforced
  - prevents child (fact) rows being inserted for which there is no parent (dimension)
  - Prevents parent (dimension) rows being delete for which there are child (fact) rows
  - Database guarantees this referential integrity.
- Permits Oracle optimizer to perform foreign key join elimination
  - If you are querying child and parent (fact and dimension) tables
    - without referencing any parent (dimension) columns (other than join column)
    - Oracle can omit query and join to dimension table because it does not alter result
- Helps develops design applications and write sensible SQL code

"9 REASONS WHY THERE ARE NO FOREIGN KEYS IN YOUR DATABASE" – PIOTR KONONOW

1. Performance: degrades DML performance as foreign keys are validated
2. Legacy data is not referentially integral in the first place.
3. Full Table Reload. Should disable, reload, and then reenable and revalidate constraints.
4. High Level Framework doesn't create foreign keys.
5. Cross-Database relations
6. Database platform agnosticism (eg. PeopleSoft)
7. Open for Change
8. Lazy Architect
9. Table relationships are not clear/revealed.

I'll add one more unacceptable excuse of my own this list:

- Extracting referentially integral OLTP data to data warehouse, so don't need more foreign keys to revalidate it there also.

WITHOUT FOREIGN KEY CONSTRAINTS

```
alter table sales modify constraint sales_channel_fk disable novalidate;
alter table sales modify constraint sales_customer_fk disable novalidate;
alter table sales modify constraint sales_product_fk disable novalidate;
alter table sales modify constraint sales_promo_fk disable novalidate;
alter table sales modify constraint sales_time_fk disable novalidate;
```

QUERY FOR CERTAIN PRODUCTS (ELECTRONICS) IN A SINGLE YEAR (1999).

```
select          p.prod_category
,               t.fiscal_year
,               count(*)
from            sales s
,               products p
,               times t
,               customers c
WHERE           s.time_id = t.time_id
AND             s.prod_id = p.prod_id
AND             t.fiscal_year = 1999
and             c.cust_id = s.cust_id
AND             p.prod_category = 'Electronics'
AND             p.prod_category = 'Software/Other'
group by p.prod_category
,               t.fiscal_year
order by 1;
```

QUERY FOR CERTAIN PRODUCTS (ELECTRONICS) IN A SINGLE YEAR (1999).

```
select * from table(dbms_xplan.display_cursor(null,null,'ADVANCED +ADAPTIVE
PROJECTION +ALLSTATS LAST, IOSTATS'));
```

Id	Operation	Name	Starts	E-Rows	E-Bytes	Cost (%CPU)	E-Time	Pstart	Pstop	A-Rows	A-Time	Buffers	Reads	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1			1167 (100)				1	00:00:00.21	3386	472			
1	SORT GROUP BY NOSORT		1	1	44	1167 (3)	00:00:01			1	00:00:00.21	3386	472			
2	NESTED LOOPS		1	2840	122K	1167 (3)	00:00:01			2840	00:00:00.21	3386	472			
3	VIEW	VW_GBF_35	1	2840	108K	1167 (3)	00:00:01			2840	00:00:00.19	544	472			
4	HASH GROUP BY		1	2840	138K	1167 (3)	00:00:01			2840	00:00:00.19	544	472	1137K	1137K	1403K (0)
* 5	HASH JOIN		1	41362	2019K	1163 (3)	00:00:01			23678	00:00:00.17	544	472	1695K	1695K	1571K (0)
6	PART JOIN FILTER CREATE	:BF0000	1	364	4368	16 (0)	00:00:01			364	00:00:00.01	55	0			
* 7	TABLE ACCESS FULL	TIMES	1	364	4368	16 (0)	00:00:01			364	00:00:00.01	55	0			
* 8	HASH JOIN		1	165K	6156K	1146 (3)	00:00:01			26637	00:00:00.17	488	472	1572K	1572K	1390K (0)
* 9	VIEW	index$_join$_002	1	13	273	2 (0)	00:00:01			13	00:00:00.01	5	0			
* 10	HASH JOIN		1							13	00:00:00.01	5	0	1355K	1355K	1376K (0)
* 11	INDEX RANGE SCAN	PRODUCTS_PROD_CAT_IX	1	13	273	1 (0)	00:00:01			13	00:00:00.01	1	0			
12	INDEX FAST FULL SCAN	PRODUCTS_PK	1	13	273	1 (0)	00:00:01			72	00:00:00.01	4	0			
13	PARTITION RANGE JOIN-FILTER		1	918K	14M	1136 (2)	00:00:01	:BF0000	:BF0000	296K	00:00:00.13	482	472			
14	TABLE ACCESS FULL	SALES	5	918K	14M	1136 (2)	00:00:01	:BF0000	:BF0000	296K	00:00:00.13	482	472			
* 15	INDEX UNIQUE SCAN	CUSTOMERS_PK	2840	1	5	0 (0)				2840	00:00:00.01	2842	0			

WITH FOREIGN KEY CONSTRAINTS

```
alter table sales modify constraint sales_channel_fk enable validate;
alter table sales modify constraint sales_customer_fk enable validate;
alter table sales modify constraint sales_product_fk enable validate;
alter table sales modify constraint sales_promo_fk enable validate;
alter table sales modify constraint sales_time_fk enable validate;
```

FOREIGN KEY JOIN ELIMINATION

```
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name                 | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers | Reads  |  OMem |  1Mem | Used-Mem |
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                      |      1 |        |       |  1163 (100)|          |       |       |      1 |00:00:00.87 |     544 |    472 |       |       |          |
|   1 |  SORT GROUP BY NOSORT         |                      |      1 |      1 |    45 |  1163   (3)| 00:00:01 |       |       |      1 |00:00:00.87 |     544 |    472 |       |       |          |
|*  2 |   HASH JOIN                   |                      |      1 |  41362 |  1817K|  1163   (3)| 00:00:01 |       |       |  23678 |00:00:00.85 |     544 |    472 |  1695K|  1695K| 1669K (0)|
|   3 |    PART JOIN FILTER CREATE    | :BF0000              |      1 |    364 |  4368 |    16   (0)| 00:00:01 |       |       |    364 |00:00:00.01 |      55 |      0 |       |       |          |
|*  4 |     TABLE ACCESS FULL         | TIMES                |      1 |    364 |  4368 |    16   (0)| 00:00:01 |       |       |    364 |00:00:00.01 |      55 |      0 |       |       |          |
|*  5 |    HASH JOIN                  |                      |      1 |    165K|  5346K|  1146   (3)| 00:00:01 |       |       |  26637 |00:00:00.79 |     488 |    472 |  1572K|  1572K| 1331K (0)|
|*  6 |     VIEW                      | index$_join$_002     |      1 |     13 |   273 |     2   (0)| 00:00:01 |       |       |     13 |00:00:00.01 |       5 |      0 |       |       |          |
|*  7 |      HASH JOIN                |                      |      1 |        |       |            |          |       |       |     13 |00:00:00.01 |       5 |      0 |  1355K|  1355K| 1377K (0)|
|*  8 |       INDEX RANGE SCAN        | PRODUCTS_PROD_CAT_IX |      1 |     13 |   273 |     1   (0)| 00:00:01 |       |       |     13 |00:00:00.01 |       1 |      0 |       |       |          |
|   9 |       INDEX FAST FULL SCAN    | PRODUCTS_PK          |      1 |     13 |   273 |     1   (0)| 00:00:01 |       |       |     72 |00:00:00.01 |       4 |      0 |       |       |          |
|  10 |     PARTITION RANGE JOIN-FILTER|                     |      1 |    918K|    10M|  1136   (2)| 00:00:01 |:BF0000|:BF0000|    296K|00:00:00.43 |     482 |    472 |       |       |          |
|  11 |      TABLE ACCESS FULL        | SALES                |      5 |    918K|    10M|  1136   (2)| 00:00:01 |:BF0000|:BF0000|    296K|00:00:00.25 |     482 |    472 |       |       |          |
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
…
Note
-----
 - rely constraint used for this statement
```

COMPARE THE PLANS

Without Foreign Keys

Id	Operation	Name	Starts	E-Rows	E-Bytes	Cost (%CPU)	E-Time	Pstart	Pstop	A-Rows	A-Time	Buffers	Reads	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1			1167 (100)				1	00:00:00.21	3386	472			
1	SORT GROUP BY NOSORT		1	1	44	1167 (3)	00:00:01			1	00:00:00.21	3386	472			
2	NESTED LOOPS		1	2840	122K	1167 (3)	00:00:01			2840	00:00:00.21	3386	472			
3	VIEW	VW_GBF_35	1	2840	108K	1167 (3)	00:00:01			2840	00:00:00.19	544	472			
4	HASH GROUP BY		1	2840	138K	1167 (3)	00:00:01			2840	00:00:00.19	544	472	1137K	1137K	1403K (0)
* 5	HASH JOIN		1	41362	2019K	1163 (3)	00:00:01			23678	00:00:00.17	544	472	1695K	1695K	1571K (0)
6	PART JOIN FILTER CREATE	:BF0000	1	364	4368	16 (0)	00:00:01			364	00:00:00.01	55	0			
* 7	TABLE ACCESS FULL	TIMES	1	364	4368	16 (0)	00:00:01			364	00:00:00.01	55	0			
* 8	HASH JOIN		1	165K	6156K	1146 (3)	00:00:01			26637	00:00:00.17	488	472	1572K	1572K	1390K (0)
* 9	VIEW	index$_join$_002	1	13	273	2 (0)	00:00:01			13	00:00:00.01	5	0			
* 10	HASH JOIN		1							13	00:00:00.01	5	0	1355K	1355K	1376K (0)
* 11	INDEX RANGE SCAN	PRODUCTS_PROD_CAT_IX	1	13	273	1 (0)	00:00:01			13	00:00:00.01	1	0			
12	INDEX FAST FULL SCAN	PRODUCTS_PK	1	13	273	1 (0)	00:00:01			72	00:00:00.01	4	0			
13	PARTITION RANGE JOIN-FILTER		1	918K	14M	1136 (2)	00:00:01	:BF0000	:BF0000	296K	00:00:00.13	482	472			
14	TABLE ACCESS FULL	SALES	5	918K	14M	1136 (2)	00:00:01	:BF0000	:BF0000	296K	00:00:00.13	482	472			
* 15	INDEX UNIQUE SCAN	CUSTOMERS_PK	2840	1	5	0 (0)				2840	00:00:00.01	2842	0			

With Foreign Keys

Id	Operation	Name	Starts	E-Rows	E-Bytes	Cost (%CPU)	E-Time	Pstart	Pstop	A-Rows	A-Time	Buffers	Reads	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1			1163 (100)				1	00:00:00.87	544	472			
1	SORT GROUP BY NOSORT		1	1	45	1163 (3)	00:00:01			1	00:00:00.87	544	472			
* 2	HASH JOIN		1	41362	1817K	1163 (3)	00:00:01			23678	00:00:00.85	544	472	1695K	1695K	1669K (0)
3	PART JOIN FILTER CREATE	:BF0000	1	364	4368	16 (0)	00:00:01			364	00:00:00.01	55	0			
* 4	TABLE ACCESS FULL	TIMES	1	364	4368	16 (0)	00:00:01			364	00:00:00.01	55	0			
* 5	HASH JOIN		1	165K	5346K	1146 (3)	00:00:01			26637	00:00:00.79	488	472	1572K	1572K	1331K (0)
* 6	VIEW	index$_join$_002	1	13	273	2 (0)	00:00:01			13	00:00:00.01	5	0			
* 7	HASH JOIN		1							13	00:00:00.01	5	0	1355K	1355K	1377K (0)
* 8	INDEX RANGE SCAN	PRODUCTS_PROD_CAT_IX	1	13	273	1 (0)	00:00:01			13	00:00:00.01	1	0			
9	INDEX FAST FULL SCAN	PRODUCTS_PK	1	13	273	1 (0)	00:00:01			72	00:00:00.01	4	0			
10	PARTITION RANGE JOIN-FILTER		1	918K	10M	1136 (2)	00:00:01	:BF0000	:BF0000	296K	00:00:00.43	482	472			
11	TABLE ACCESS FULL	SALES	5	918K	10M	1136 (2)	00:00:01	:BF0000	:BF0000	296K	00:00:00.25	482	472			

WITH DISABLED RELIABLE CONSTRAINTS

```
alter table sales modify constraint sales_customer_fk rely disable novalidate;
alter session set query_rewrite_integrity = TRUSTED;
```

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name                | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers | Reads  |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                     |      1 |        |       |  1164 (100)|          |       |       |      1 |00:00:01.55 |     544 |    472 |       |       |          |
|   1 |  SORT GROUP BY NOSORT          |                     |      1 |      1 |    45 |  1164   (3)| 00:00:01 |       |       |      1 |00:00:01.55 |     544 |    472 |       |       |          |
|*  2 |   HASH JOIN                    |                     |      1 |  81133 |  3565K|  1164   (3)| 00:00:01 |       |       |    110K|00:00:01.46 |     544 |    472 |  1476K|  1476K| 1528K (0)|
|*  3 |    VIEW                        | index$_join$_002    |      1 |     26 |   546 |     2   (0)| 00:00:01 |       |       |     26 |00:00:00.01 |       5 |      0 |       |       |          |
|*  4 |     HASH JOIN                  |                     |      1 |        |       |            |          |       |       |     26 |00:00:00.01 |       5 |      0 |  1298K|  1298K| 1612K (0)|
|*  5 |      INDEX RANGE SCAN          | PRODUCTS_PROD_CAT_IX|      1 |     26 |   546 |     1   (0)| 00:00:01 |       |       |     26 |00:00:00.01 |       1 |      0 |       |       |          |
|   6 |      INDEX FAST FULL SCAN      | PRODUCTS_PK         |      1 |     26 |   546 |     1   (0)| 00:00:01 |       |       |     72 |00:00:00.01 |       4 |      0 |       |       |          |
|*  7 |    HASH JOIN                   |                     |      1 |    229K|  5369K|  1160   (3)| 00:00:01 |       |       |    246K|00:00:01.00 |     538 |    472 |  1695K|  1695K| 1683K (0)|
|   8 |     PART JOIN FILTER CREATE    | :BF0000             |      1 |    364 |  4368 |    16   (0)| 00:00:01 |       |       |    364 |00:00:00.01 |      55 |      0 |       |       |          |
|*  9 |      TABLE ACCESS FULL         | TIMES               |      1 |    364 |  4368 |    16   (0)| 00:00:01 |       |       |    364 |00:00:00.01 |      55 |      0 |       |       |          |
|  10 |     PARTITION RANGE JOIN-FILTER|                     |      1 |    918K|    10M|  1136   (2)| 00:00:01 |:BF0000|:BF0000|    296K|00:00:00.47 |     482 |    472 |       |       |          |
|  11 |      TABLE ACCESS FULL         | SALES               |      5 |    918K|    10M|  1136   (2)| 00:00:01 |:BF0000|:BF0000|    296K|00:00:00.28 |     482 |    472 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
…
Note
-----
 - rely constraint used for this statement
```

MULTI-COLUMN FOREIGN KEY JOIN ELIMINATION

You have always been able to create foreign keys on multiple columns

- From Oracle 12.2 you can get join-elimination on such foreign keys

```
alter table child
add constraint child_fk_parent foreign key (id_g, id_p)
references parent (id_g, id);
```

Jonathan Lewis' blog demo

- Bug: join elimination depends on order of tables in the from clause
- Workaround: List parents before children in from clause

MISTAKE 1: LACK OF FOREIGN KEYS GOOD PRACTICE

- Single Column Primary Keys on all tables
  - Enforced primary key constraints with indexes
- Foreign Keys on all dimension columns
  - Referencing primary keys though equality joins
  - Foreign Key join elimination on multi-column keys from 12.2
    - but better avoided due to from clause order sensitivity
- If enforcing Foreign Key constraints
  - Index foreign key columns to avoid TM locking
- If not enforcing Foreign Key constraints
  - `ALTER TABLE … MODIFY CONSTRAINT … RELY NOVALIDATE DISABLE`
  - From Oracle 12c, `QUERY_REWRITE_INTEGRITY=TRUSTED`

MISTAKE 2: EFFECTIVE DATED DIMENSIONS

MISTAKE 2: EFFECTIVE DATED DIMENSIONS

- Multiple rows for the same dimension ID
  - Different Effective from dates (and sometimes also effective to dates)

- SH demo Schema: PRODUCTS tables
  - Usually just PROD_ID is the primary key
  - Validity dates on dimension, but just unindexed attributes
    - PROD_EFF_FROM
    - PROD_EFF_TO

  - I am going to build a BAD_PRODUCTS table
    - PROD_EFF_FROM is going to be part of the Primary Key

THE WRONG WAY: BAD_PRODUCTS TABLE

```
CREATE TABLE bad_products (
    prod_id                    NUMBER(6)        NOT NULL,
    prod_name                  VARCHAR2(50)     NOT NULL,
    prod_desc                  VARCHAR2(4000)   NOT NULL,
    prod_subcategory           VARCHAR2(50)     NOT NULL,
    prod_subcategory_id        NUMBER           NOT NULL,
    prod_subcategory_desc      VARCHAR2(2000)   NOT NULL,
    prod_category              VARCHAR2(50)     NOT NULL,
    prod_category_id           NUMBER           NOT NULL,
    prod_category_desc         VARCHAR2(2000)   NOT NULL,
    prod_weight_class          NUMBER(3)        NOT NULL,
    prod_unit_of_measure       VARCHAR2(20)     ,
    prod_pack_size             VARCHAR2(30)     NOT NULL,
    supplier_id                NUMBER(6)        NOT NULL,
    prod_status                VARCHAR2(20)     NOT NULL,
    prod_list_price            NUMBER(8,2)      NOT NULL,
    prod_min_price             NUMBER(8,2)      NOT NULL,
    prod_total                 VARCHAR2(13)     NOT NULL,
    prod_total_id              NUMBER           NOT NULL,
    prod_src_id                NUMBER           ,
    prod_eff_from              DATE             ,
    prod_eff_to                DATE             ,
    prod_valid                 VARCHAR2(1)      )
/
ALTER TABLE bad_products
ADD CONSTRAINT bad_products_pk PRIMARY KEY (prod_id, prod_eff_from)
/
```


THE WRONG WAY: BAD_PRODUCTS TABLE

```
INSERT INTO bad_products
WITH s AS
(
SELECT s.prod_id
, TRUNC(s.time_id,'YYYY') time_id
FROM sales s
GROUP BY s.prod_id, trunc(s.time_id,'YYYY')
)
SELECT p.prod_id
,p.prod_name
,p.prod_desc||' '||TO_CHAR(s.time_id,'YYYY')
,p.prod_subcategory, p.prod_subcategory_id
,p.prod_subcategory_desc||' '||TO_CHAR(s.time_id,'YYYY')
,p.prod_category, p.prod_category_id
,p.prod_category_desc||' '||TO_CHAR(s.time_id,'YYYY')
,p.prod_weight_class, p.prod_unit_of_measure, p.prod_pack_size, p.supplier_id, p.prod_status, p.prod_list_price,
p.prod_min_price, p.prod_total, p.prod_total_id, p.prod_src_id
,s.time_id PROD_EFF_FROM
,ADD_MONTHS(s.time_id,12)-1 PROD_EFF_TO
,p.prod_valid
FROM products p
,    s
WHERE s.prod_id = p.prod_id
/
CREATE INDEX bad_products_prod_cat_ix ON bad_products (prod_category);
CREATE BITMAP INDEX bad_products_prod_status_bix ON bad_products (prod_status);
CREATE INDEX bad_products_prod_subcat_ix ON bad_products (prod_subcategory);
```

CAN'T CREATE FOREIGN KEYS

I can't use PROD_ID only for a foreign key because that does not match the primary key.

```
ALTER TABLE sales
  ADD CONSTRAINT sales_bad_product_fk
  FOREIGN KEY (prod_id) REFERENCES bad_products (prod_id)
;
ERROR at line 3:
ORA-02270: no matching unique or primary key for this column-list
```

CAN'T CREATE FOREIGN KEYS

- I can't build the foreign key on PROD_ID and TIME_ID
  - TIME_IDs are merely inside the effective range but are not on the PRODUCTS table

```
ALTER TABLE sales
  ADD CONSTRAINT sales_bad_product_fk
  FOREIGN KEY (prod_id, time_id) REFERENCES bad_products (prod_id, prod_eff_from)
;
ERROR at line 2:
ORA-02298: cannot validate (SH.SALES_BAD_PRODUCT_FK) - parent keys not found
```

CAN'T CREATE FOREIGN KEYS

- I have no choice but to code two inequality conditions on the product table
  - with one of them on a column that is not part of the foreign key.

```
…
from          sales s
,             bad_products p
,             times t
WHERE         s.time_id = t.time_id
AND           s.prod_id = p.prod_id
AND           t.time_id >= p.prod_eff_from
AND           (t.time_id <= p.prod_eff_to OR p.prod_eff_to IS NULL)
…
```

- Even on Oracle 12.2,
- Even if I create the above constraint NOVALIDATE RELY and set QUERY_REWRITE_INTEGRITY to TRUSTED
- I still cannot get foreign key join elimination
  - Because the query references PROD_EFF_TO, that is not in the foreign key

THE RIGHT WAY:
NEW YEAR, NEW PRODUCT, NEW PROD_ID

- Duplicate products table
  - Prefix PROD_ID with year, so new product ID every year

```
INSERT INTO products2
WITH s AS
(
SELECT s.prod_id
, TRUNC(s.time_id,'YYYY') time_id
FROM sales s
GROUP BY s.prod_id, trunc(s.time_id,'YYYY')
)
SELECT p.PROD_ID+(1000*TO_NUMBER(TO_CHAR(s.time_id,'YYYY')))
,p.prod_name ,p.prod_desc||' '||TO_CHAR(s.time_id,'YYYY')
,p.prod_subcategory ,p.prod_subcategory_id
,p.prod_subcategory_desc||' '||TO_CHAR(s.time_id,'YYYY')
,p.prod_category ,p.prod_category_id ,p.prod_category_desc||' '||TO_CHAR(s.time_id,'YYYY')
,p.prod_weight_class, p.prod_unit_of_measure ,p.prod_pack_size ,p.supplier_id ,p.prod_status
,p.prod_list_price ,p.prod_min_price ,p.prod_total ,p.prod_total_id ,p.prod_src_id
,s.time_id PROD_EFF_FROM
,ADD_MONTHS(s.time_id,12)-1 PROD_EFF_TO
,p.prod_valid
FROM products p
,    s
WHERE s.prod_id = p.prod_id
/
```


THE RIGHT WAY:

- Duplicate sales table
  - Prefix PROD_ID with year, so new product ID every year

```
CREATE TABLE sales2…

INSERT /*+APPEND*/ INTO sales2
SELECT
 prod_id+1000*TO_NUMBER(TO_CHAR(time_id,'YYYY')) prod_id
,cust_id
,time_id
,channel_id
,promo_id
,quantity_sold
,amount_sold
FROM sales;
```

THE RIGHT WAY:
QUERY U.S. SOFTWARE SALES BY YEAR

```
SELECT c.country_name
,      p.prod_category_id
,      p.prod_category
,      t.fiscal_year
,      COUNT(*) num_sales
,      SUM(s.amount_sold) total_amount_sold
FROM   sales2 s
,      customers u
,      products2 p
,      times t
,      countries c
WHERE  s.time_id = t.time_id
AND    s.prod_id = p.prod_id
AND    u.cust_id = s.cust_id
AND    u.country_id = c.country_id
AND    c.country_iso_code = 'US'
AND    p.prod_category_id = 205
AND    t.time_id >= p.prod_eff_from
AND    (t.time_id <= p.prod_eff_to OR p.prod_eff_to IS NULL)
AND    s.time_id >= p.prod_eff_from
AND    (s.time_id <= p.prod_eff_to OR p.prod_eff_to IS NULL)
GROUP BY c.country_name
,        p.prod_category_id
,        p.prod_category
,        t.fiscal_year
ORDER BY 1,2
```

THE WRONG WAY:
WITH STAR QUERY TRANSFORMATION

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                          | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                               |      1 |        |       |  2000 (100)|          |       |       |      5 |00:00:06.60 |     305K|       |       |          |
|   1 |  TEMP TABLE TRANSFORMATION               |                               |      1 |        |       |            |          |       |       |      5 |00:00:06.60 |     305K|       |       |          |
|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D7230_A4BC21     |      1 |        |       |            |          |       |       |      0 |00:00:00.08 |    1525 |  1024 |  1024 |          |
|*  3 |    HASH JOIN                             |                               |      1 |   2921 | 81788 |   418   (1)| 00:00:01 |       |       |  18520 |00:00:00.05 |    1524 |  1185K|  1185K|  721K (0)|
|*  4 |     TABLE ACCESS FULL                    | COUNTRIES                     |      1 |      1 |    18 |     2   (0)| 00:00:01 |       |       |      1 |00:00:00.01 |       2 |       |       |          |
|   5 |     TABLE ACCESS FULL                    | CUSTOMERS                     |      1 |  55500 |  541K|   416   (1)| 00:00:01 |       |       |  55500 |00:00:00.02 |    1521 |       |       |          |
|   6 |   SORT GROUP BY                          |                               |      1 |      5 |   495 |  1582   (1)| 00:00:01 |       |       |      5 |00:00:06.52 |     303K|  2048 |  2048 | 2048  (0)|
|*  7 |    HASH JOIN                             |                               |      1 |  11568 | 1118K|  1580   (1)| 00:00:01 |       |       |   237K|00:00:06.29 |     303K|  2290K|  1666K| 2158K (0)|
|   8 |     TABLE ACCESS FULL                    | SYS_TEMP_0FD9D7230_A4BC21     |      1 |   2921 | 43815 |     4   (0)| 00:00:01 |       |       |  18520 |00:00:00.01 |       0 |       |       |          |
|*  9 |     HASH JOIN                            |                               |      1 |  11568 |  948K|  1576   (1)| 00:00:01 |       |       |   237K|00:00:06.18 |     303K|  1744K|  1744K| 1704K (0)|
|  10 |      TABLE ACCESS FULL                   | TIMES                         |      1 |   1826 | 21912 |    16   (0)| 00:00:01 |       |       |   1826 |00:00:00.01 |      55 |       |       |          |
|* 11 |     HASH JOIN                            |                               |      1 |  30764 | 2163K|  1559   (1)| 00:00:01 |       |       |   237K|00:00:06.09 |     303K|  1115K|  1115K| 1352K (0)|
|* 12 |      TABLE ACCESS FULL                   | BAD_PRODUCTS                  |      1 |    102 |  4284 |     5   (0)| 00:00:01 |       |       |    102 |00:00:00.01 |      14 |       |       |          |
|  13 |      VIEW                                | VW_ST_F7251F01                |      1 |  48359 | 1416K|  1554   (1)| 00:00:01 |       |       |   237K|00:00:05.02 |     303K|       |       |          |
|  14 |       NESTED LOOPS                       |                               |      1 |  48359 | 2361K|  1526   (1)| 00:00:01 |       |       |   237K|00:00:04.82 |     303K|       |       |          |
|  15 |        PARTITION RANGE ALL               |                               |      1 |  48359 |  991K|   397   (2)| 00:00:01 |     1 |    28 |   237K|00:00:03.36 |     302K|       |       |          |
|  16 |         BITMAP CONVERSION TO ROWIDS      |                               |     28 |  48359 |  991K|   397   (2)| 00:00:01 |       |       |   237K|00:00:03.15 |     302K|       |       |          |
|  17 |          BITMAP AND                      |                               |     28 |        |       |            |          |       |       |     16 |00:00:03.01 |     302K|       |       |          |
|  18 |           BITMAP MERGE                   |                               |     28 |        |       |            |          |       |       |     16 |00:00:00.04 |    1819 |  1024K|   512K|36864  (0)|
|  19 |            BITMAP KEY ITERATION          |                               |     28 |        |       |            |          |       |       |   1572 |00:00:00.03 |    1819 |       |       |          |
|  20 |             BUFFER SORT                  |                               |     28 |        |       |            |          |       |       |   2856 |00:00:00.01 |       9 | 73728 | 73728 |          |
|* 21 |              TABLE ACCESS BY INDEX ROWID BATCHED| BAD_PRODUCTS           |      1 |    102 |   816 |    24   (0)| 00:00:01 |       |       |    102 |00:00:00.01 |       9 |       |       |          |
|  22 |               BITMAP CONVERSION TO ROWIDS|                               |      1 |        |       |            |          |       |       |    375 |00:00:00.01 |       1 |       |       |          |
|  23 |                BITMAP INDEX FULL SCAN    | BAD_PRODUCTS_PROD_STATUS_BIX  |      1 |        |       |            |          |       |       |      1 |00:00:00.01 |       1 |       |       |          |
|* 24 |               BITMAP INDEX RANGE SCAN    | SALES_PROD_BIX                |   2856 |        |       |            |          |     1 |    28 |   1572 |00:00:00.02 |    1810 |       |       |          |
|  25 |           BITMAP MERGE                   |                               |     28 |        |       |            |          |       |       |     16 |00:00:02.97 |     301K|  8316K|  1039K|  317K (0)|
|  26 |            BITMAP KEY ITERATION          |                               |     28 |        |       |            |          |       |       |  19186 |00:00:02.85 |     301K|       |       |          |
|  27 |             BUFFER SORT                  |                               |     28 |        |       |            |          |       |       |    296K|00:00:00.42 |       0 |    26M|  1871K|  865K (0)|
|  28 |              TABLE ACCESS FULL           | SYS_TEMP_0FD9D7230_A4BC21     |      5 |   2921 | 14605 |     4   (0)| 00:00:01 |       |       |  18520 |00:00:00.01 |       0 |       |       |          |
|* 29 |              BITMAP INDEX RANGE SCAN     | SALES_CUST_BIX                |   296K|        |       |            |          |     1 |    28 |  19186 |00:00:01.97 |     301K|       |       |          |
|  30 |         TABLE ACCESS BY USER ROWID       | SALES                         |   237K|      1 |    29 |  1157   (1)| 00:00:01 | ROWID | ROWID |   237K|00:00:00.92 |     829 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
…
Predicate Information (identified by operation id):
---------------------------------------------------

 11 - access("ITEM_3"="P"."PROD_ID")
      filter(("ITEM_1">="P"."PROD_EFF_FROM" AND ("ITEM_1"<="P"."PROD_EFF_TO" OR "P"."PROD_EFF_TO" IS NULL)))
```


THE RIGHT WAY:
WITH STAR QUERY TRANSFORMATION

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                        | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                             |      1 |        |       |  1207 (100)|          |       |       |      5 |00:00:05.08 |     304K|       |       |          |
|   1 |  TEMP TABLE TRANSFORMATION               |                             |      1 |        |       |            |          |       |       |      5 |00:00:05.08 |     304K|       |       |          |
|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D7231_A4BC21   |      1 |        |       |            |          |       |       |      0 |00:00:00.07 |    1525 |  1024 |  1024 |          |
|*  3 |    HASH JOIN                             |                             |      1 |   2921 | 81788 |   418   (1)| 00:00:01 |       |       |  18520 |00:00:00.05 |    1524 |  1185K|  1185K|  689K (0)|
|*  4 |     TABLE ACCESS FULL                    | COUNTRIES                   |      1 |      1 |    18 |     2   (0)| 00:00:01 |       |       |      1 |00:00:00.01 |       2 |       |       |          |
|   5 |     TABLE ACCESS FULL                    | CUSTOMERS                   |      1 |  55500 |   541K|   416   (1)| 00:00:01 |       |       |  55500 |00:00:00.02 |    1521 |       |       |          |
|   6 |   SORT GROUP BY                          |                             |      1 |      5 |   435 |   789   (1)| 00:00:01 |       |       |      5 |00:00:05.01 |     302K|  2048 |  2048 | 2048  (0)|
|*  7 |    HASH JOIN                             |                             |      1 |  17849 |  1516K|   787   (1)| 00:00:01 |       |       |   237K|00:00:04.83 |     302K|  2290K|  1666K| 2123K (0)|
|   8 |     TABLE ACCESS FULL                    | SYS_TEMP_0FD9D7231_A4BC21   |      1 |   2921 | 43815 |     4   (0)| 00:00:01 |       |       |  18520 |00:00:00.01 |       0 |       |       |          |
|*  9 |     HASH JOIN                            |                             |      1 |  17849 |  1255K|   782   (1)| 00:00:01 |       |       |   237K|00:00:04.74 |     302K|  1744K|  1744K| 1704K (0)|
|  10 |      TABLE ACCESS FULL                   | TIMES                       |      1 |   1826 | 21912 |    16   (0)| 00:00:01 |       |       |   1826 |00:00:00.01 |      55 |       |       |          |
|* 11 |     HASH JOIN                            |                             |      1 |  17849 |  1045K|   766   (1)| 00:00:01 |       |       |   237K|00:00:04.68 |     302K|  1298K|  1298K| 1658K (0)|
|* 12 |      TABLE ACCESS FULL                   | PRODUCTS2                   |      1 |    102 |  2856 |     5   (0)| 00:00:01 |       |       |    102 |00:00:00.01 |      14 |       |       |          |
|  13 |      VIEW                                | VW_ST_F7251F01              |      1 |  17937 |   560K|   761   (1)| 00:00:01 |       |       |   237K|00:00:04.18 |     302K|       |       |          |
|  14 |       NESTED LOOPS                       |                             |      1 |  17937 |   963K|   733   (1)| 00:00:01 |       |       |   237K|00:00:03.99 |     302K|       |       |          |
|  15 |        PARTITION RANGE ALL               |                             |      1 |  17936 |   402K|   395   (1)| 00:00:01 |     1 |    28 |   237K|00:00:02.76 |     302K|       |       |          |
|  16 |         BITMAP CONVERSION TO ROWIDS      |                             |     28 |  17936 |   402K|   395   (1)| 00:00:01 |       |       |   237K|00:00:02.60 |     302K|       |       |          |
|  17 |          BITMAP AND                      |                             |     28 |        |       |            |          |       |       |    16 |00:00:02.49 |     302K|       |       |          |
|  18 |           BITMAP MERGE                   |                             |     28 |        |       |            |          |       |       |    16 |00:00:00.03 |    1700 |  1024K|   512K| 9216  (0)|
|  19 |            BITMAP KEY ITERATION          |                             |     28 |        |       |            |          |       |       |   399 |00:00:00.02 |    1700 |       |       |          |
|  20 |             BUFFER SORT                  |                             |     28 |        |       |            |          |       |       |  2856 |00:00:00.01 |       9 | 73728 | 73728 |          |
|* 21 |              TABLE ACCESS BY INDEX ROWID BATCHED| PRODUCTS2           |      1 |    102 |  1020 |    24   (0)| 00:00:01 |       |       |    102 |00:00:00.01 |       9 |       |       |          |
|  22 |               BITMAP CONVERSION TO ROWIDS|                             |      1 |        |       |            |          |       |       |    275 |00:00:00.01 |       1 |       |       |          |
|  23 |                BITMAP INDEX FULL SCAN    | PRODUCTS2_PROD_STATUS_BIX   |      1 |        |       |            |          |       |       |      1 |00:00:00.01 |       1 |       |       |          |
|* 24 |               BITMAP INDEX RANGE SCAN    | SALES2_PROD_BIX             |   2856 |        |       |            |          |     1 |    28 |    399 |00:00:00.01 |    1691 |       |       |          |
|  25 |           BITMAP MERGE                   |                             |     28 |        |       |            |          |       |       |     16 |00:00:02.46 |     300K|  8400K|  1050K|  312K (0)|
|  26 |            BITMAP KEY ITERATION          |                             |     28 |        |       |            |          |       |       |  19192 |00:00:02.36 |     300K|       |       |          |
|  27 |             BUFFER SORT                  |                             |     28 |        |       |            |          |       |       |   296K|00:00:00.33 |       0 |   26M|  1871K|  865K (0)|
|  28 |              TABLE ACCESS FULL           | SYS_TEMP_0FD9D7231_A4BC21   |      5 |   2921 | 14605 |     4   (0)| 00:00:01 |       |       |  18520 |00:00:00.01 |       0 |       |       |          |
|* 29 |              BITMAP INDEX RANGE SCAN     | SALES2_CUST_BIX             |   296K |        |       |            |          |     1 |    28 |  19192 |00:00:01.63 |     300K|       |       |          |
|  30 |        TABLE ACCESS BY USER ROWID        | SALES2                      |   237K |      1 |    32 |   366   (1)| 00:00:01 | ROWID | ROWID |   237K|00:00:00.76 |     830 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
…
Predicate Information (identified by operation id):
---------------------------------------------------

  9 - access("ITEM_3"="T"."TIME_ID")
 11 - access("ITEM_2"="P"."PROD_ID")
```


THE WRONG WAY: FULL SCAN/BLOOM FILTER

```
Plan hash value: 2324880021

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name         | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers | Reads  |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |              |      1 |        |       |  1592 (100)|          |       |       |      5 |00:00:01.94 |    5805 |   4128 |       |       |          |
|   1 |  SORT GROUP BY                     |              |      1 |     11 |  1144 |  1592   (3)| 00:00:01 |       |       |      5 |00:00:01.94 |    5805 |   4128 |  2048 |  2048 | 2048  (0)|
|*  2 |   HASH JOIN                        |              |      1 |   9845 |   999K|  1590   (3)| 00:00:01 |       |       |    237K|00:00:01.74 |    5805 |   4128 |  1744K|  1744K| 1620K (0)|
|   3 |    PART JOIN FILTER CREATE         | :BF0000      |      1 |   1826 | 21912 |    16   (0)| 00:00:01 |       |       |   1826 |00:00:00.01 |      55 |      0 |       |       |          |
|   4 |     TABLE ACCESS FULL              | TIMES        |      1 |   1826 | 21912 |    16   (0)| 00:00:01 |       |       |   1826 |00:00:00.01 |      55 |      0 |       |       |          |
|*  5 |    HASH JOIN                       |              |      1 |  26182 |  2352K|  1574   (3)| 00:00:01 |       |       |    237K|00:00:01.65 |    5700 |   4128 |  1115K|  1115K| 1390K (0)|
|*  6 |     TABLE ACCESS FULL              | BAD_PRODUCTS |      1 |    102 |  4284 |     5   (0)| 00:00:01 |       |       |    102 |00:00:00.01 |      14 |      0 |       |       |          |
|*  7 |     HASH JOIN                      |              |      1 |  48360 |  2361K|  1568   (3)| 00:00:01 |       |       |    526K|00:00:01.07 |    5685 |   4128 |  2290K|  1666K| 2216K (0)|
|*  8 |      HASH JOIN                     |              |      1 |   2921 |  81788 |   416   (1)| 00:00:01 |       |       |  18520 |00:00:00.04 |    1524 |      0 |  1236K|  1236K|  771K (0)|
|*  9 |       TABLE ACCESS FULL            | COUNTRIES    |      1 |      1 |    18 |     2   (0)| 00:00:01 |       |       |      1 |00:00:00.01 |       2 |      0 |       |       |          |
|  10 |       TABLE ACCESS FULL            | CUSTOMERS    |      1 |  55500 |   541K|   416   (1)| 00:00:01 |       |       |  55500 |00:00:00.02 |    1521 |      0 |       |       |          |
|  11 |      PARTITION RANGE JOIN-FILTER   |              |      1 |    918K|    19M|  1142   (3)| 00:00:01 |:BF0000|:BF0000|    918K|00:00:00.77 |    4160 |   4128 |       |       |          |
|  12 |       TABLE ACCESS FULL            | SALES        |     20 |    918K|    19M|  1142   (3)| 00:00:01 |:BF0000|:BF0000|    918K|00:00:00.76 |    4160 |   4128 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("S"."TIME_ID"="T"."TIME_ID")
       filter((("T"."TIME_ID"<="P"."PROD_EFF_TO" OR "P"."PROD_EFF_TO" IS NULL) AND "T"."TIME_ID">="P"."PROD_EFF_FROM"))
   5 - access("S"."PROD_ID"="P"."PROD_ID")
       filter(("S"."TIME_ID">="P"."PROD_EFF_FROM" AND ("S"."TIME_ID"<="P"."PROD_EFF_TO" OR "P"."PROD_EFF_TO" IS NULL)))
```

THE RIGHT WAY: FULL SCAN/BLOOM FILTER

```
Plan hash value: 1297664275

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name      | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |           |      1 |        |       |   875 (100)|          |       |       |      5 |00:00:00.92 |    3126 |       |       |          |
|   1 |  SORT GROUP BY                 |           |      1 |     11 |  1012 |   875   (4)| 00:00:01 |       |       |      5 |00:00:00.92 |    3126 |  2048 |  2048 | 2048  (0)|
|*  2 |   HASH JOIN                    |           |      1 |  17849 |  1603K|   873   (4)| 00:00:01 |       |       |    237K|00:00:00.75 |    3126 |  1744K|  1744K| 1621K (0)|
|   3 |    PART JOIN FILTER CREATE     | :BF0000   |      1 |   1826 | 21912 |    16   (0)| 00:00:01 |       |       |   1826 |00:00:00.01 |      55 |       |       |          |
|   4 |     TABLE ACCESS FULL          | TIMES     |      1 |   1826 | 21912 |    16   (0)| 00:00:01 |       |       |   1826 |00:00:00.01 |      55 |       |       |          |
|*  5 |    HASH JOIN                   |           |      1 |  17849 |  1394K|   857   (4)| 00:00:01 |       |       |    237K|00:00:00.70 |    3070 |  1298K|  1298K| 1703K (0)|
|*  6 |     TABLE ACCESS FULL          | PRODUCTS2 |      1 |    102 |  2856 |     5   (0)| 00:00:01 |       |       |    102 |00:00:00.01 |      14 |       |       |          |
|*  7 |     HASH JOIN                  |           |      1 |  48360 |  2455K|   852   (4)| 00:00:01 |       |       |    526K|00:00:00.59 |    3055 |  2290K|  1666K| 2216K (0)|
|*  8 |      HASH JOIN                 |           |      1 |   2921 | 81788 |   418   (1)| 00:00:01 |       |       |  18520 |00:00:00.04 |    1524 |  1236K|  1236K|  774K (0)|
|*  9 |       TABLE ACCESS FULL        | COUNTRIES |      1 |      1 |    18 |     2   (0)| 00:00:01 |       |       |      1 |00:00:00.01 |       2 |       |       |          |
|  10 |       TABLE ACCESS FULL        | CUSTOMERS |      1 |  55500 |   541K|   416   (1)| 00:00:01 |       |       |  55500 |00:00:00.03 |    1521 |       |       |          |
|  11 |       PARTITION RANGE JOIN-FILTER|         |      1 |    918K|    21M|   426   (6)| 00:00:01 |:BF0000|:BF0000|    918K|00:00:00.35 |    1530 |       |       |          |
|  12 |        TABLE ACCESS FULL       | SALES2    |     20 |    918K|    21M|   426   (6)| 00:00:01 |:BF0000|:BF0000|    918K|00:00:00.35 |    1530 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
```


MISTAKE 2: EFFECTIVE DATED DIMENSIONS

- Leads to inequality conditions between fact and dimension tables.
  - Can't express inequalities in foreign key – must be equality join to primary key/unique key
  - Hence, impossible to achieve foreign key join elimination
  - More work evaluating the inequality conditions when you eventually hash join these dimensions into a result set.
  - Bloom Filters only work with equality predicates.
    - The Bloom cannot reduce the data on the inequality conditions, so they return more data to the hash operation that then takes longer to execute.

- Conclusion
  - Rigorously follow the star schema principles
    - so that you can always define a foreign key between facts and dimensions.

MISTAKE 3: DATES THAT ARE NOT DATES

MISTAKE 3: DATES THAT ARE NOT DATES

- Good Practice
  - Not uncommon to see a time dimension
    - with one row per day
    - Primary key is a date
  - Various periodic attributes for each grouping
    - Avoids need for complex grouping functions

- SH.TIMES dimension →

```
Name                                      Null?    Type
----------------------------------------- -------- ------------
TIME_ID                                   NOT NULL DATE
DAY_NAME                                  NOT NULL VARCHAR2(9)
DAY_NUMBER_IN_WEEK                        NOT NULL NUMBER(1)
DAY_NUMBER_IN_MONTH                       NOT NULL NUMBER(2)
CALENDAR_WEEK_NUMBER                      NOT NULL NUMBER(2)
FISCAL_WEEK_NUMBER                        NOT NULL NUMBER(2)
WEEK_ENDING_DAY                           NOT NULL DATE
WEEK_ENDING_DAY_ID                        NOT NULL NUMBER
CALENDAR_MONTH_NUMBER                     NOT NULL NUMBER(2)
FISCAL_MONTH_NUMBER                       NOT NULL NUMBER(2)
CALENDAR_MONTH_DESC                       NOT NULL VARCHAR2(8)
CALENDAR_MONTH_ID                         NOT NULL NUMBER
FISCAL_MONTH_DESC                         NOT NULL VARCHAR2(8)
FISCAL_MONTH_ID                           NOT NULL NUMBER
DAYS_IN_CAL_MONTH                         NOT NULL NUMBER
DAYS_IN_FIS_MONTH                         NOT NULL NUMBER
END_OF_CAL_MONTH                          NOT NULL DATE
END_OF_FIS_MONTH                          NOT NULL DATE
CALENDAR_MONTH_NAME                       NOT NULL VARCHAR2(9)
FISCAL_MONTH_NAME                         NOT NULL VARCHAR2(9)
CALENDAR_QUARTER_DESC                     NOT NULL CHAR(7)
CALENDAR_QUARTER_ID                       NOT NULL NUMBER
FISCAL_QUARTER_DESC                       NOT NULL CHAR(7)
FISCAL_QUARTER_ID                         NOT NULL NUMBER
DAYS_IN_CAL_QUARTER                       NOT NULL NUMBER
DAYS_IN_FIS_QUARTER                       NOT NULL NUMBER
END_OF_CAL_QUARTER                        NOT NULL DATE
END_OF_FIS_QUARTER                        NOT NULL DATE
CALENDAR_QUARTER_NUMBER                   NOT NULL NUMBER(1)
FISCAL_QUARTER_NUMBER                     NOT NULL NUMBER(1)
CALENDAR_YEAR                             NOT NULL NUMBER(4)
CALENDAR_YEAR_ID                          NOT NULL NUMBER
FISCAL_YEAR                               NOT NULL NUMBER(4)
FISCAL_YEAR_ID                            NOT NULL NUMBER
DAYS_IN_CAL_YEAR                          NOT NULL NUMBER
DAYS_IN_FIS_YEAR                          NOT NULL NUMBER
END_OF_CAL_YEAR                           NOT NULL DATE
END_OF_FIS_YEAR                           NOT NULL DATE
```

MISTAKE 3: DATES THAT ARE NOT DATES

- **Not Good Practice**
  - Date dimensions with 200 years (1900-2100).
    - Not a problem if queries always specify a date range on a dimension.
    - Can be an issue if use an unbounded inequality.

```
AND t.time_id < TO_DATE('31122018','DDMMYYYY')
```

  - Keep size of date dimension reasonable

MISTAKE 3: DATES THAT ARE NOT DATES

- Bad Practice
  - You should represent a date as a date and not as a string or a number.
    - 31st December 2018
    - Not 20,181,231
    - Not '20181231’
- If you represent a date as a string you can get miscalculations in the optimizer.
- For example
  - the difference between 31st December 2018 and 1st January 2019 should be 1 day.
  - However, if you use the string representation of a day and then make it a number
    - **20190101-20181231=8870**
  - This can lead to bad cardinality estimates, and in more complex queries it can cause subsequent bad optimizer decisions later in the query.

BAD_TIMES: NUMERIC TIME_ID PRIMARY KEY

```
CREATE TABLE SH.BAD_TIMES (
       "TIME_ID" NUMBER NOT NULL,
       "CALENDAR_DATE" DATE NOT NULL,
       "DAY_NAME" VARCHAR2(9) NOT NULL,
…
 CONSTRAINT "BAD_TIMES_PK" PRIMARY KEY ("TIME_ID")
)
/
```

```
INSERT /*+APPEND*/ INTO bad_times
( TIME_ID, CALENDAR_DATE, DAY_NAME
…
) select
  TO_NUMBER(TO_CHAR(TIME_ID,'YYYYMMDD')) time_id, TIME_ID calendar_date, DAY_NAME
…
FROM times
/
```

BAD_TIMES: NUMERIC TIME_ID PRIMARY KEY

- TIME_ID is now a number that contains the date string
- CALENDAR_DATE is the original date value

```
 TIME_ID CALENDAR_
---------- ---------
  19980101 01-JAN-98
  19980102 02-JAN-98
  19980103 03-JAN-98
  19980104 04-JAN-98
  19980105 05-JAN-98
  19980106 06-JAN-98
  19980107 07-JAN-98
  19980108 08-JAN-98
  19980109 09-JAN-98
  19980110 10-JAN-98
...
```

BAD_SALES: NUMERIC TIME_ID ATTRIBUTE

```
INSERT /*+APPEND*/ INTO bad_sales
(prod_id, cust_id, time_id, channel_id, promo_id, quantity_sold, amount_sold)
SELECT prod_id, cust_id
, TO_NUMBER(TO_CHAR(time_id,'yyyymmdd'))
, channel_id, promo_id, quantity_sold, amount_sold
from sales
/
```

THE WORST OF TIMES: MONTHLY SALES ANALYSIS FOR FY 1999

```
SELECT        t.fiscal_year, t.fiscal_month_id, t.fiscal_month_desc
,             COUNT(*) num_sales
,             SUM(s.amount_sold) total_amount_sold
from          bad_sales s
,             customers u
,             products p
,             bad_times t
WHERE         s.time_id = t.time_id
AND           s.prod_id = p.prod_id
AND           u.cust_id = s.cust_id
AND           t.calendar_date >= TO_DATE('27121998','DDMMYYYY')
AND           t.calendar_date <  TO_DATE('27121999','DDMMYYYY')
GROUP BY t.fiscal_year, t.fiscal_month_id, t.fiscal_month_desc
ORDER BY 1
/
```

THE WORST OF TIMES: MONTHLY SALES ANALYSIS FOR FY 1999

```
Plan hash value: 4232725394

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name      | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |           |      1 |        |       |   509 (100)|          |       |       |     13 |00:00:00.15 |     540 |       |       |          |
|   1 |  SORT GROUP BY                 |           |      1 |    366 | 22692 |   509  (19)| 00:00:01 |       |       |     13 |00:00:00.15 |     540 |  2048 |  2048 | 2048  (0)|
|*  2 |   HASH JOIN                    |           |      1 |    366 | 22692 |   508  (18)| 00:00:01 |       |       |    365 |00:00:00.15 |     540 |  1335K|  1335K| 1207K (0)|
|   3 |    JOIN FILTER CREATE          | :BF0001   |      1 |    366 | 10980 |    16   (0)| 00:00:01 |       |       |    365 |00:00:00.01 |      56 |       |       |          |
|   4 |     PART JOIN FILTER CREATE    | :BF0000   |      1 |    366 | 10980 |    16   (0)| 00:00:01 |       |       |    365 |00:00:00.01 |      56 |       |       |          |
|*  5 |      TABLE ACCESS FULL         | BAD_TIMES |      1 |    366 | 10980 |    16   (0)| 00:00:01 |       |       |    365 |00:00:00.01 |      56 |       |       |          |
|   6 |    VIEW                        | VW_GBC_5  |      1 |   1460 | 46720 |   492  (19)| 00:00:01 |       |       |    370 |00:00:00.15 |     483 |       |       |          |
|   7 |     HASH GROUP BY              |           |      1 |   1460 | 16060 |   492  (19)| 00:00:01 |       |       |    370 |00:00:00.15 |     483 |  1079K|  1079K| 2561K (0)|
|   8 |      JOIN FILTER USE           | :BF0001   |      1 |    918K|  9870K|   423   (6)| 00:00:01 |       |       |    250K|00:00:00.10 |     483 |       |       |          |
|   9 |       PARTITION RANGE JOIN-FILTER|         |      1 |    918K|  9870K|   423   (6)| 00:00:01 |:BF0000|:BF0000|    250K|00:00:00.09 |     483 |       |       |          |
|* 10 |        TABLE ACCESS FULL       | BAD_SALES |      5 |    918K|  9870K|   423   (6)| 00:00:01 |:BF0000|:BF0000|    250K|00:00:00.09 |     483 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
```


THE BEST OF TIMES: MONTHLY SALES ANALYSIS FOR FY 1999

```
SELECT      t.fiscal_year, t.fiscal_month_id, t.fiscal_month_desc
,           COUNT(*) num_sales
,           SUM(s.amount_sold) total_amount_sold
FROM        sales s
,           customers u
,           products p
,           times t
WHERE       s.time_id = t.time_id
AND         s.prod_id = p.prod_id
AND         u.cust_id = s.cust_id
AND         t.time_id >= TO_DATE('27121998','DDMMYYYY')
AND         t.time_id <  TO_DATE('27121999','DDMMYYYY')
GROUP BY t.fiscal_year, t.fiscal_month_id, t.fiscal_month_desc
ORDER BY 1
/
```

THE BEST OF TIMES: MONTHLY SALES ANALYSIS FOR FY 1999

```
Plan hash value: 3667272686

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                              | Name      | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time    | Buffers | Reads  |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |           |      1 |        |       |   165 (100)|          |       |       |     13 |00:00:00.28 |     500 |    303 |       |       |          |
|   1 |  SORT GROUP BY                         |           |      1 |    365 | 21170 |   165  (14)| 00:00:01 |       |       |     13 |00:00:00.28 |     500 |    303 |  2048 |  2048 | 2048  (0)|
|*  2 |   HASH JOIN                            |           |      1 |    365 | 21170 |   164  (13)| 00:00:01 |       |       |    365 |00:00:00.28 |     500 |    303 |  1298K|  1298K| 1527K (0)|
|   3 |    PART JOIN FILTER CREATE             | :BF0000   |      1 |    366 |  8784 |    14   (0)| 00:00:01 |       |       |    365 |00:00:00.01 |      14 |      0 |       |       |          |
|   4 |     TABLE ACCESS BY INDEX ROWID BATCHED| TIMES     |      1 |    366 |  8784 |    14   (0)| 00:00:01 |       |       |    365 |00:00:00.01 |      14 |      0 |       |       |          |
|*  5 |      INDEX RANGE SCAN                  | TIMES_PK  |      1 |    366 |       |     3   (0)| 00:00:01 |       |       |    365 |00:00:00.01 |       3 |      0 |       |       |          |
|   6 |    VIEW                                | VW_GBC_5  |      1 |    366 | 12444 |   150  (14)| 00:00:01 |       |       |    365 |00:00:00.28 |     486 |    303 |       |       |          |
|   7 |     HASH GROUP BY                      |           |      1 |    366 |  4758 |   150  (14)| 00:00:01 |       |       |    365 |00:00:00.28 |     486 |    303 |  1063K|  1063K| 2552K (0)|
|   8 |      PARTITION RANGE AND               |           |      1 |    230K|  2924K|   133   (4)| 00:00:01 |KEY(AP)|KEY(AP)|    247K|00:00:00.21 |     486 |    303 |       |       |          |
|*  9 |       TABLE ACCESS FULL                | SALES     |      5 |    230K|  2924K|   133   (4)| 00:00:01 |KEY(AP)|KEY(AP)|    247K|00:00:00.21 |     486 |    303 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
…
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("ITEM_1"."T"."TIME_ID")
   5 - access("T"."TIME_ID">=TO_DATE(' 1998-12-27 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND "T"."TIME_ID"<TO_DATE(' 1999-12-27 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
   9 - filter(("S"."TIME_ID"<TO_DATE(' 1999-12-27 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND "S"."TIME_ID">=TO_DATE(' 1998-12-27 00:00:00', 'syyyy-mm-dd hh24:mi:ss')))
```

BAD TIMES:
BAR CARDINALITY ACROSS YEAR END

```
select          t.fiscal_year, t.fiscal_month_id, t.fiscal_month_desc
,               count(*) num_sales
,               SUM(s.amount_sold) total_amount_sold
from            bad_sales s
,               customers u
,               products p
,               bad_times t
WHERE           s.time_id = t.time_id
AND             s.prod_id = p.prod_id
AND             u.cust_id = s.cust_id
and             t.time_id >= 19982712
AND             t.time_id <  19992712
group by t.fiscal_year, t.fiscal_month_id, t.fiscal_month_desc
order by 1
/
```

BAD TIMES:
BAD CARDINALITY ACROSS YEAR END

```
Plan hash value: 1098183223

-----------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name      | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |           |      1 |        |       |   176 (100)|          |       |       |     13 |00:00:00.16 |     459 |       |       |          |
|   1 |  SORT GROUP BY           |           |      1 |    445 | 24030 |   176  (18)| 00:00:01 |       |       |     13 |00:00:00.16 |     459 |  2048 |  2048 | 2048  (0)|
|*  2 |   HASH JOIN              |           |      1 |    445 | 24030 |   175  (18)| 00:00:01 |       |       |    365 |00:00:00.16 |     459 |  1355K|  1355K| 1540K (0)|
|   3 |    PART JOIN FILTER CREATE| :BF0000  |      1 |    445 |  9790 |    16   (0)| 00:00:01 |       |       |    365 |00:00:00.01 |      56 |       |       |          |
|*  4 |     TABLE ACCESS FULL    | BAD_TIMES |      1 |    445 |  9790 |    16   (0)| 00:00:01 |       |       |    365 |00:00:00.01 |      56 |       |       |          |
|   5 |    VIEW                  | VW_GBC_5  |      1 |    470 | 15040 |   159  (19)| 00:00:01 |       |       |    365 |00:00:00.16 |     402 |       |       |          |
|   6 |     HASH GROUP BY        |           |      1 |    470 |  5170 |   159  (19)| 00:00:01 |       |       |    365 |00:00:00.16 |     402 |  1079K|  1079K| 2576K (0)|
|   7 |      PARTITION RANGE AND |           |      1 |    295K|  3177K|   138   (7)| 00:00:01 |KEY(AP)|KEY(AP)|    247K|00:00:00.10 |     402 |       |       |          |
|*  8 |       TABLE ACCESS FULL  | BAD_SALES |      4 |    295K|  3177K|   138   (7)| 00:00:01 |KEY(AP)|KEY(AP)|    247K|00:00:00.09 |     402 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
```

JULIAN NUMERIC DATE

- If you already have a numeric primary key on your time dimension
  - And you really can't change it
  - Populate it with Julian date values – then the arithmetic will be correct.

	From	To	Difference	Function
Date	31st December 2018	1st January 2019	1	
Number	20181231	20190101	8070	TO_CHAR(…,'YYYMMDD')
Julian	2458484	2458485	1	TO_CHAR(….'J')

STAR TRANSFORMATION –V- FULL SCAN/BLOOM FILTER

TO INDEX, OR NOT TO INDEX, THAT IS THE QUESTION

- Star Transformation
  - Introduced in Oracle 8i
- Bitmap Index on each foreign key column(s) on the FACT table
  - Though you don't actually need the foreign key to be defined
  - Index on primary key on DIMENSIONs
  - Bitmap merge multiple dimensions
- Requires
  - STAR_TRANSFORMATION_ENABLED=TRUE
  - Or /*+STAR_TRANSFORMATION*/
  - And it also has to be cheaper*

- Full Scan, Bloom Filter, Hash Join
  - Originally introduced in 10g
    - Hash dimension
    - each hash value is a bit in a map
  - Hash fact table
    - Filter against the bit map
  - Exact match with hash function
    - But it is now a smaller hash

STAR TRANSFORMATION

- **Cost Based Transformation**
  - **(it is supposed to be, but I have a case where it transforms to a more expensive plan)**

- **Oracle documentation says:**
  - ***"The optimizer performs the star transformation by identifying the fact and constraint dimension tables automatically. The optimizer performs the star transformation only if the cost of the transformed plan is lower than the alternatives. Also, the optimizer attempts temporary table transformation automatically whenever materialization improves performance“***
  - ***"The STAR_TRANSFORMATION hint instructs the optimizer to use the best plan in which the transformation has been used. Without the hint, the optimizer could make a query optimization decision to use the best plan generated without the transformation, instead of the best plan for the transformed query"***

STAR TRANSFORMATION: US SALES IN 1999 BY STATE

```
SELECT          c.country_name
,               u.cust_state_province
,               COUNT(*) num_sales
,               SUM(s.amount_sold) total_amount_sold
from            sales s
,               customers u
,               products p
,               times t
,               countries c
WHERE           s.time_id = t.time_id
AND             s.prod_id = p.prod_id
AND             u.cust_id = s.cust_id
AND             u.country_id = c.country_id
AND             c.country_iso_code = 'US'
AND             t.fiscal_year = 1999
GROUP BY        c.country_name, u.cust_state_province
ORDER BY 1,2
/
```

STAR TRANSFORMATION: WITH TEMP TABLE OPTIMISATION

```
Plan hash value: 3988189767

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                       | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers | Reads  |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                            |      1 |        |       |  2133 (100)|          |       |       |     45 |00:00:02.74 |   98078 |    195 |       |       |          |
|   1 |  TEMP TABLE TRANSFORMATION               |                            |      1 |        |       |            |          |       |       |     45 |00:00:02.74 |   98078 |    195 |       |       |          |
|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D77CE_A4BC21  |      1 |        |       |            |          |       |       |      0 |00:00:00.08 |    1525 |      0 |  1024 |  1024 |          |
|   3 |   HASH JOIN                              |                            |      1 |   2921 |   111K|   418   (1)| 00:00:01 |       |       |  18520 |00:00:00.05 |    1524 |      0 |  1185K|  1185K|  599K (0)|
|   4 |    TABLE ACCESS FULL                     | COUNTRIES                  |      1 |      1 |    18 |     2   (0)| 00:00:01 |       |       |      1 |00:00:00.01 |       2 |      0 |       |       |          |
|   5 |    TABLE ACCESS FULL                     | CUSTOMERS                  |      1 |  55500 |  1138K|   416   (1)| 00:00:01 |       |       |  55500 |00:00:00.02 |    1521 |      0 |       |       |          |
|   6 |   HASH JOIN                              |                            |      1 |   2359 |   101K|  1715   (1)| 00:00:01 |       |       |     45 |00:00:02.66 |   96552 |    195 |  6144 |  6144 | 6144  (0)|
|   7 |    HASH JOIN                             |                            |      1 |  12057 |   518K|  1713   (1)| 00:00:01 |       |       |    141K|00:00:02.49 |   96552 |    195 |  2391K|  1595K| 2015K (0)|
|   8 |     TABLE ACCESS FULL                    | SYS_TEMP_0FD9D77CE_A4BC21  |      1 |   2921 | 75946 |     6   (0)| 00:00:01 |       |       |  18520 |00:00:00.01 |       0 |      0 |       |       |          |
|   9 |     VIEW                                 | VW_ST_B6F3B15C             |      1 |  12057 |   211K|  1707   (1)| 00:00:01 |       |       |    141K|00:00:02.18 |   96552 |    195 |       |       |          |
|  10 |      NESTED LOOPS                        |                            |      1 |  12057 |   553K|  1684   (1)| 00:00:01 |       |       |    141K|00:00:02.08 |   96552 |    195 |       |       |          |
|  11 |       PARTITION RANGE SUBQUERY           |                            |      1 |  12056 |   294K|   322   (1)| 00:00:01 |KEY(SQ)|KEY(SQ)|    141K|00:00:01.95 |   96103 |      0 |       |       |          |
|  12 |        BITMAP CONVERSION TO ROWIDS       |                            |      5 |  12056 |   294K|   322   (1)| 00:00:01 |       |       |    141K|00:00:00.94 |   96047 |      0 |       |       |          |
|  13 |         BITMAP AND                       |                            |      5 |        |       |            |          |       |       |      5 |00:00:00.87 |   96047 |      0 |       |       |          |
|  14 |          BITMAP MERGE                    |                            |      5 |        |       |            |          |       |       |      5 |00:00:00.02 |    1914 |      0 |  1024K|   512K|40960  (0)|
|  15 |           BITMAP KEY ITERATION           |                            |      5 |        |       |            |          |       |       |    364 |00:00:00.02 |    1914 |      0 |       |       |          |
|  16 |            BUFFER SORT                   |                            |      5 |        |       |            |          |       |       |   1820 |00:00:00.01 |      55 |      0 | 73728 | 73728 |          |
|  17 |             TABLE ACCESS FULL            | TIMES                      |      1 |    364 |  4368 |    16   (0)| 00:00:01 |       |       |    364 |00:00:00.01 |      55 |      0 |       |       |          |
|  18 |             BITMAP INDEX RANGE SCAN      | SALES_TIME_BIX             |   1820 |        |       |            |          |KEY(SQ)|KEY(SQ)|    364 |00:00:00.01 |    1859 |      0 |       |       |          |
|  19 |          BITMAP MERGE                    |                            |      5 |        |       |            |          |       |       |      5 |00:00:00.84 |   94133 |      0 |  8624K|  1078K|  319K (0)|
|  20 |           BITMAP KEY ITERATION           |                            |      5 |        |       |            |          |       |       |   6566 |00:00:00.81 |   94133 |      0 |       |       |          |
|  21 |            BUFFER SORT                   |                            |      5 |        |       |            |          |       |       |  92600 |00:00:00.14 |       0 |      0 |    28M|  1930K|  928K (0)|
|  22 |             TABLE ACCESS FULL            | SYS_TEMP_0FD9D77CE_A4BC21  |      1 |   2921 | 14605 |     6   (0)| 00:00:01 |       |       |  18520 |00:00:00.01 |       0 |      0 |       |       |          |
|  23 |             BITMAP INDEX RANGE SCAN      | SALES_CUST_BIX             |  92600 |        |       |            |          |KEY(SQ)|KEY(SQ)|   6566 |00:00:00.54 |   94133 |      0 |       |       |          |
|  24 |        TABLE ACCESS BY USER ROWID        | SALES                      |   141K |      1 |    22 |  1385   (1)| 00:00:01 | ROWID | ROWID |    141K|00:00:00.76 |     450 |    195 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Note
-----
   - cbqt star transformation used for this statement
   - rely constraint used for this statement
```

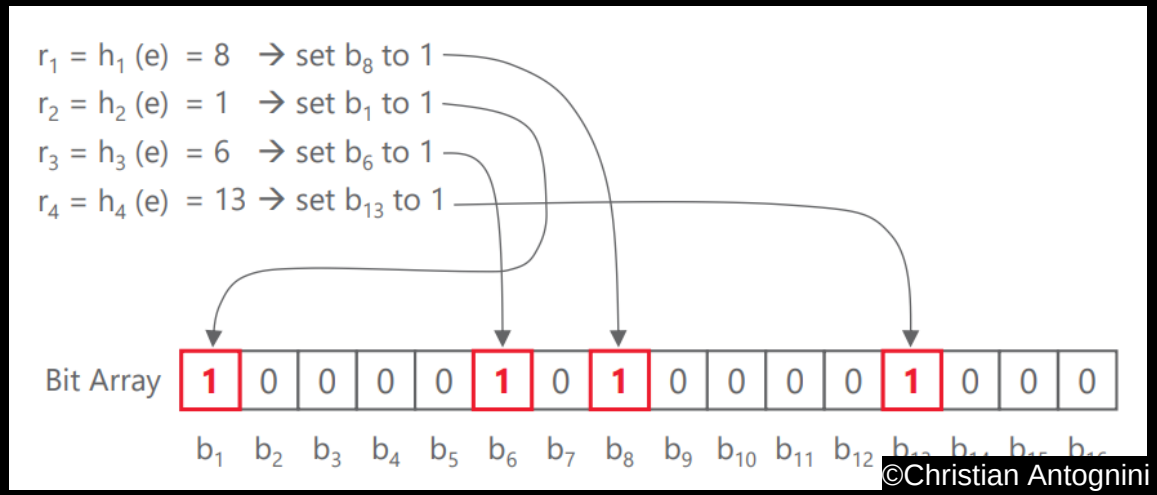

STAR TRANSFORMATION: DISABLE TEMP TABLE OPTIMISATION

```
Plan hash value: 2782741738

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name           | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                |      1 |        |       |  2950 (100)|          |       |       |     45 |00:00:02.91 |   99599 |       |       |          |
|   1 |  SORT GROUP BY                      |                |      1 |     45 |  2565 |  2950   (1)| 00:00:01 |       |       |     45 |00:00:02.91 |   99599 |  6144 |  6144 | 6144  (0)|
|*  2 |   HASH JOIN                         |                |      1 |    635 | 36195 |  2949   (1)| 00:00:01 |       |       |    141K|00:00:02.77 |   99599 |   12M|  2994K|   16M (0)|
|   3 |    MERGE JOIN CARTESIAN             |                |      1 |  12057 |   423K|  2533   (1)| 00:00:01 |       |       |    141K|00:00:02.39 |   98078 |       |       |          |
|*  4 |     TABLE ACCESS FULL               | COUNTRIES      |      1 |      1 |    18 |     2   (0)| 00:00:01 |       |       |      1 |00:00:00.01 |       2 |       |       |          |
|   5 |     BUFFER SORT                     |                |      1 |  12057 |   211K|  2531   (1)| 00:00:01 |       |       |    141K|00:00:02.29 |   98076 |  6219K|  1010K| 5527K (0)|
|   6 |      VIEW                           | VW_ST_958ED0D5 |      1 |  12057 |   211K|  2531   (1)| 00:00:01 |       |       |    141K|00:00:02.03 |   98076 |       |       |          |
|   7 |       NESTED LOOPS                  |                |      1 |  12057 |   553K|  2097   (1)| 00:00:01 |       |       |    141K|00:00:01.92 |   98076 |       |       |          |
|   8 |        PARTITION RANGE SUBQUERY     |                |      1 |  12056 |   294K|   734   (1)| 00:00:01 |KEY(SQ)|KEY(SQ)|    141K|00:00:01.16 |   97626 |       |       |          |
|   9 |         BITMAP CONVERSION TO ROWIDS |                |      5 |  12056 |   294K|   734   (1)| 00:00:01 |       |       |    141K|00:00:01.05 |   97571 |       |       |          |
|  10 |          BITMAP AND                 |                |      5 |        |       |            |          |       |       |      5 |00:00:00.98 |   97571 |       |       |          |
|  11 |           BITMAP MERGE              |                |      5 |        |       |            |          |       |       |      5 |00:00:00.02 |    1914 |  1024K|   512K|40960  (0)|
|  12 |            BITMAP KEY ITERATION     |                |      5 |        |       |            |          |       |       |    364 |00:00:00.02 |    1914 |       |       |          |
|  13 |             BUFFER SORT             |                |      5 |        |       |            |          |       |       |   1820 |00:00:00.01 |      55 | 73728 | 73728 |          |
|* 14 |              TABLE ACCESS FULL      | TIMES          |      1 |    364 |  4368 |    16   (0)| 00:00:01 |       |       |    364 |00:00:00.01 |      55 |       |       |          |
|* 15 |             BITMAP INDEX RANGE SCAN | SALES_TIME_BIX |   1820 |        |       |            |          |KEY(SQ)|KEY(SQ)|    364 |00:00:00.01 |    1859 |       |       |          |
|  16 |           BITMAP MERGE              |                |      5 |        |       |            |          |       |       |      5 |00:00:00.96 |   95657 |  8624K|  1078K|  319K (0)|
|  17 |            BITMAP KEY ITERATION     |                |      5 |        |       |            |          |       |       |   6566 |00:00:00.92 |   95657 |       |       |          |
|  18 |             BUFFER SORT             |                |      5 |        |       |            |          |       |       |  92600 |00:00:00.21 |    1524 |   36M|  2148K| 1180K (0)|
|* 19 |              HASH JOIN              |                |      1 |   2921 | 52578 |   418   (1)| 00:00:01 |       |       |  18520 |00:00:00.04 |    1524 |  1922K|  1922K|  565K (0)|
|* 20 |               TABLE ACCESS FULL     | COUNTRIES      |      1 |      1 |     8 |     2   (0)| 00:00:01 |       |       |      1 |00:00:00.01 |       2 |       |       |          |
|  21 |               TABLE ACCESS FULL     | CUSTOMERS      |      1 |  55500 |   541K|   416   (1)| 00:00:01 |       |       |  55500 |00:00:00.02 |    1521 |       |       |          |
|* 22 |             BITMAP INDEX RANGE SCAN | SALES_CUST_BIX |  92600 |        |       |            |          |KEY(SQ)|KEY(SQ)|   6566 |00:00:00.58 |   94133 |       |       |          |
|  23 |         TABLE ACCESS BY USER ROWID  | SALES          |   141K|      1 |    22 |  1797   (1)| 00:00:01 | ROWID | ROWID |    141K|00:00:00.48 |     450 |       |       |          |
|  24 |  TABLE ACCESS FULL                  | CUSTOMERS      |      1 |  55500 |  1138K|   416   (1)| 00:00:01 |       |       |  55500 |00:00:00.03 |    1521 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

WHAT IS A BLOOM FILTER?

- Bloom Filters by Christian Antognini, 2008
  - See also Bloom Filters @ DOAG Webinar 2016
  - Space/Time trade-offs in hash coding with allowable errors by Burton H. Bloom, 1970



©Christian Antognini

Hash value calculated for each dimension join column value

Each hash value sets a bit in a map

Same hash function used to calculate hash value of fact join column

Rows filtered against bitmap

False positives possible because multiple input values can have same hash value

Then do exact match with hash join to remove false positives, but now on a smaller data set

FULL SCAN/BLOOM FILTER

```
Plan hash value: 2588754131

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name       | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time    | Buffers | Reads  |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |            |      1 |        |       |  1586 (100)|          |       |       |     45 |00:00:00.46 |    2063 |    472 |       |       |          |
|   1 |  SORT GROUP BY                 |            |      1 |     45 |  3105 |  1586   (3)| 00:00:01 |       |       |     45 |00:00:00.46 |    2063 |    472 |  6144 |  6144 | 6144  (0)|
|*  2 |   HASH JOIN                    |            |      1 |  12057 |   812K|  1585   (3)| 00:00:01 |       |       |    141K|00:00:00.31 |    2063 |    472 |  1995K|  1995K| 1655K (0)|
|   3 |    PART JOIN FILTER CREATE     | :BF0000    |      1 |    364 |  4368 |    16   (0)| 00:00:01 |       |       |    364 |00:00:00.01 |      55 |      0 |       |       |          |
|*  4 |     TABLE ACCESS FULL          | TIMES      |      1 |    364 |  4368 |    16   (0)| 00:00:01 |       |       |    364 |00:00:00.01 |      55 |      0 |       |       |          |
|*  5 |    HASH JOIN                   |            |      1 |  48360 |  2691K|  1568   (3)| 00:00:01 |       |       |    169K|00:00:00.27 |    2007 |    472 |  2391K|  1595K| 1990K (0)|
|*  6 |     HASH JOIN                  |            |      1 |   2921 |   111K|   418   (1)| 00:00:01 |       |       |  18520 |00:00:00.03 |    1524 |      0 |  1236K|  1236K|  728K (0)|
|*  7 |      TABLE ACCESS FULL         | COUNTRIES  |      1 |      1 |    18 |     2   (0)| 00:00:01 |       |       |      1 |00:00:00.01 |       2 |      0 |       |       |          |
|   8 |      TABLE ACCESS FULL         | CUSTOMERS  |      1 |  55500 |  1138K|   416   (1)| 00:00:01 |       |       |  55500 |00:00:00.02 |    1521 |      0 |       |       |          |
|   9 |     PARTITION RANGE JOIN-FILTER|            |      1 |    918K|    15M|  1142   (3)| 00:00:01 |:BF0000|:BF0000|    296K|00:00:00.15 |     482 |    472 |       |       |          |
|  10 |      TABLE ACCESS FULL         | SALES      |      5 |    918K|    15M|  1142   (3)| 00:00:01 |:BF0000|:BF0000|    296K|00:00:00.15 |     482 |    472 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```


STAR TRANSFORMATION –V- FULL SCAN/BLOOM FILTER?

Which is better?

It depends

Star Transformation

- **Frequently performs better for queries that return a small result set compared to the total data set to be searched.**
- **But there is a catch…**

Full Scan/Bloom Filter/Hash Join

- **Frequently performs better for queries that return a larger result set compared to the data set to be searched.**
- **Engineered system optimisations**

THE TROUBLE WITH BITMAP INDEXES: THEY WORK BEST ON STATIC DATA!

- "There is a locking implication by design in bitmap indexes, the keys point to hundreds of rows and if I lock a single key entry in a bitmap index, I've locked the key entry for hundreds of other rows.
- That is WHY bitmaps are not recommended - if you update a bitmap indexed column OR you insert/delete from the table (which will modify the bitmap index), you'll find serialization at best and deadlocks all over the place at worst."

Tom Kyte, Ask Tom, 2007

- You may also see the size of the bitmap index grow significantly and quickly.

Jonathan Lewis's Blog:

- Bitmap Indexes index page
  - Bitmap Loading (deferred index maintenance)
  - Bitmap Nulls

TEST QUERY: 1999 SALES FOR A COUNTRY BY STATE/COUNTY/PROVINCE/DEPARTMENT/LAND

```
SELECT          c.country_name
,               u.cust_state_province
,               COUNT(*) num_sales
,               SUM(s.amount_sold) total_amount_sold
from            sales s
,               customers u
,               products p
,               times t
,               countries c
WHERE           s.time_id = t.time_id
AND             s.prod_id = p.prod_id
AND             u.cust_id = s.cust_id
AND             u.country_id = c.country_id
AND             c.country_iso_code = '&&iso_country_code'
AND             t.fiscal_year = 1999
GROUP BY        c.country_name, u.cust_state_province
ORDER BY 1,2
/
```

STAR TRANSFORMATION –V- FULL SCAN/BLOOM FILTER

ISO Country Code	Country Name	Number of Sales in 1999	Number of Sales	Star Transformation		Full Scan-Bloom Filter	
				A-Time	Buffers	A-Time	Buffers
US	United States of America	2662	526212	1.76	97999	0.55	2068
DE	Germany	561	81978	0.51	45572	0.29	2068
JP	Japan	281	60183	0.16	7118	0.23	2068
GB	United Kingdom	391	58638	0.38	42339	0.25	2068
IT	Italy	257	42570	0.53	43526	0.38	2068
AU	Australia	228	33685	0.14	8113	0.26	2068
FR	France	161	33078	0.3	23401	0.27	2068
SG	Singapore	80	25253	0.16	6928	0.4	2068
CA	Canada	90	22858	0.27	14118	0.32	2068
ES	Spain	85	17136	0.16	14282	0.22	2068
DK	Denmark	89	16651	0.08	5844	0.24	2068
AR	Argentina	3	202	0.07	5724	0.21	2068
BR	Brazil	9	180	0.09	7912	0.26	2068
TR	Turkey	1	168	0.05	4127	0.19	2068
CN	China	4	19	0.1	7265	0.18	2068
PL	Poland	2	18	0.11	7267	0.23	2068
SA	Saudi Arabia	0	7	0.07	4037	0.26	2068

SNOWFLAKING & LOST SKEW

SNOWFLAKING & LOST SKEW

In the previous example, the cost of the execution plans didn't change as I changed the size of the country changed.

This is sometimes called ‘lost skew’.

SNOWFLAKING & LOST SKEW: QUERY BY COUNTRY_ISO_CODE

```
Plan hash value: 3095970037

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   Id  | Operation                                | Name                       | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT                         |                            |      1 |        |       |  1473 (100)|          |       |       |     45 |00:00:01.77 |     101K|       |       |          |
|     1 |  TEMP TABLE TRANSFORMATION               |                            |      1 |        |       |            |          |       |       |     45 |00:00:01.77 |     101K|       |       |          |
|     2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D7C68_A4BC21  |      1 |        |       |            |          |       |       |      0 |00:00:00.13 |    1889 |  1024 |  1024 |          |
|  *  3 |    HASH JOIN                             |                            |      1 |   2413 | 94107 |   418   (1)| 00:00:01 |       |       |  18520 |00:00:00.10 |    1888 |  1185K|  1185K|  639K (0)|
|  *  4 |     TABLE ACCESS FULL                    | COUNTRIES                  |      1 |      1 |    18 |     2   (0)| 00:00:01 |       |       |      1 |00:00:00.01 |       2 |       |       |          |
|     5 |     TABLE ACCESS FULL                    | CUSTOMERS                  |      1 |  55500 |  1138K|   416   (1)| 00:00:01 |       |       |  55500 |00:00:00.02 |    1521 |       |       |          |
|     6 |   SORT GROUP BY                          |                            |      1 |   2359 |   101K|  1053   (1)| 00:00:01 |       |       |     45 |00:00:01.65 |   99111 |  6144 |  6144 |  6144  (0)|
|  *  7 |    HASH JOIN                             |                            |      1 |   3597 |   154K|  1054   (1)| 00:00:01 |       |       |  64818 |00:00:01.58 |   99111 |  2391K|  1595K|  2025K (0)|
|     8 |     TABLE ACCESS FULL                    | SYS_TEMP_0FD9D7C68_A4BC21  |      1 |   2413 | 62738 |     5   (0)| 00:00:01 |       |       |  18520 |00:00:00.01 |       0 |       |       |          |
|     9 |     VIEW                                 | VW_ST_C525CEF3             |      1 |   3597 | 64746 |  1048   (1)| 00:00:01 |       |       |  64818 |00:00:01.44 |   99111 |       |       |          |
...
```

- There are 55500 rows on CUSTOMERS
- There are 23 rows on COUNTRIES
- Oracle expects 2413 rows on joining those tables
  - 55500÷23= 2413.04
  - Actually got 18520
  - Oracle assumes the data is evenly distributed between countries
  - Although there are histograms on COUNTRY_ISO_CODE and COUNTRY_ID.
    - Jonathan Lewis' blog: Bitmap John Indexes

DIFFERENT TEST QUERY: BY COUNTRY_ID

```
SELECT          c.country_name
,               u.cust_state_province
,               COUNT(*) num_sales
,               SUM(s.amount_sold) total_amount_sold
from            sales s
,               customers u
,               products p
,               times t
,               countries c
WHERE           s.time_id = t.time_id
AND             s.prod_id = p.prod_id
AND             u.cust_id = s.cust_id
AND             u.country_id = c.country_id
AND             c.country_id = '&&country_id'
AND             t.fiscal_year = 1999
GROUP BY        c.country_name, u.cust_state_province
ORDER BY 1,2
/
```

TEST QUERY: BY COUNTRY_ID STAR_TRANSFORMATION

- Replace the predicate on COUNTRY_ISO_CODE with a predicate on COUNTRY_ID

```
Plan hash value: 1339390240

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                              | Name                     | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |                          |      1 |        |       |  6922 (100)|          |       |       |     45 |00:00:01.50 |   97998 |       |       |          |
|   1 |  TEMP TABLE TRANSFORMATION             |                          |      1 |        |       |            |          |       |       |     45 |00:00:01.50 |   97998 |       |       |          |
|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D7C6A_A4BC21|      1 |        |       |            |          |       |       |      0 |00:00:00.06 |    1524 |  1024 |  1024 |          |
|   3 |    NESTED LOOPS                        |                          |      1 |  18520 |   651K|   417   (1)| 00:00:01 |       |       |  18520 |00:00:00.04 |    1523 |       |       |          |
|   4 |     TABLE ACCESS BY INDEX ROWID        | COUNTRIES                |      1 |      1 |    15 |     1   (0)| 00:00:01 |       |       |      1 |00:00:00.01 |       2 |       |       |          |
|   5 |      INDEX UNIQUE SCAN                 | COUNTRIES_PK             |      1 |      1 |       |     0   (0)|          |       |       |      1 |00:00:00.01 |       1 |       |       |          |
|   6 |     TABLE ACCESS FULL                  | CUSTOMERS                |      1 |  18520 |   379K|   416   (1)| 00:00:01 |       |       |  18520 |00:00:00.03 |    1521 |       |       |          |
|   7 |   SORT GROUP BY                        |                          |      1 |   2359 |   101K|  6505   (1)| 00:00:01 |       |       |     45 |00:00:01.43 |   96473 |  6144 |  6144 | 6144  (0)|
|   8 |    HASH JOIN                           |                          |      1 |  82724 |  3554K|  6499   (1)| 00:00:01 |       |       |  64818 |00:00:01.37 |   96473 |  2391K|  1595K| 2002K (0)|
|   9 |     TABLE ACCESS FULL                  | SYS_TEMP_0FD9D7C6A_A4BC21|      1 |  18520 |   470K|    25   (0)| 00:00:01 |       |       |  18520 |00:00:00.01 |       0 |       |       |          |
…
```

- The estimate of the number of rows from customers is correctly 18520 rows.
- The cost of the star transformation has gone up from 1473 to 6922.
- In fact, I only get this star transformation if I force it with a hint

TEST QUERY: BY COUNTRY_ID

- Replace the predicate on COUNTRY_ISO_CODE with a predicate on COUNTRY_ID

```
Plan hash value: 3784979335

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name         | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop | A-Rows |   A-Time   | Buffers | Reads  |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |              |      1 |        |       |  1595 (100)|          |       |       |     45 |00:00:00.54 |    2065 |    472 |       |       |          |
|   1 |  SORT GROUP BY                 |              |      1 |     45 |  3510 |  1595   (3)| 00:00:01 |       |       |     45 |00:00:00.54 |    2065 |    472 |  6144 |  6144 | 6144  (0)|
|   2 |   HASH JOIN                    |              |      1 |        |       |            |          |       |       |        |00:00:00.43 |    2065 |    472 |  2337K|  2200K| 2221K (0)|
|   3 |    TABLE ACCESS FULL           | CUSTOMERS    |      1 |  18520 |  379K|   416   (1)| 00:00:01 |       |       |  18520 |00:00:00.02 |    1521 |      0 |       |       |          |
|   4 |    HASH JOIN                   |              |      1 |  81133 |  4516K|  1172   (3)| 00:00:01 |       |       |    110K|00:00:00.35 |     544 |    472 |  2546K|  2546K| 1610K (0)|
|   5 |     TABLE ACCESS FULL          | PRODUCTS     |      1 |     26 |   208 |     3   (0)| 00:00:01 |       |       |     26 |00:00:00.01 |       4 |      0 |       |       |          |
|   6 |     HASH JOIN                  |              |      1 |    229K|    10M|  1167   (3)| 00:00:01 |       |       |    246K|00:00:00.30 |     539 |    472 |  1133K|  1133K| 1698K (0)|
|   7 |      PART JOIN FILTER CREATE   | :BF0000      |      1 |    364 |  9828 |    17   (0)| 00:00:01 |       |       |    364 |00:00:00.01 |      57 |      0 |       |       |          |
|   8 |       NESTED LOOPS             |              |      1 |    364 |  9828 |    17   (0)| 00:00:01 |       |       |    364 |00:00:00.01 |      57 |      0 |       |       |          |
|   9 |        TABLE ACCESS BY INDEX ROWID| COUNTRIES |      1 |      1 |    15 |     1   (0)| 00:00:01 |       |       |      1 |00:00:00.01 |       2 |      0 |       |       |          |
|  10 |         INDEX UNIQUE SCAN      | COUNTRIES_PK |      1 |      1 |       |     0   (0)|          |       |       |      1 |00:00:00.01 |       1 |      0 |       |       |          |
|  11 |        TABLE ACCESS FULL       | TIMES        |      1 |    364 |  4368 |    16   (0)| 00:00:01 |       |       |    364 |00:00:00.01 |      55 |      0 |       |       |          |
|  12 |      PARTITION RANGE JOIN-FILTER|             |      1 |    918K|    19M|  1142   (3)| 00:00:01 |:BF0000|:BF0000|    296K|00:00:00.21 |     482 |    472 |       |       |          |
|  13 |       TABLE ACCESS FULL        | SALES        |      5 |    918K|    19M|  1142   (3)| 00:00:01 |:BF0000|:BF0000|    296K|00:00:00.20 |     482 |    472 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

ENGINEERED SYSTEMS

ENGINEERED SYSTEMS OPTIMISATIONS: BLOOM FILTERS

- Bloom filters are pushed to storage cells during smart scan
  - Full scan of the fact tables are much more efficient.

  Both Star Transformation and Full Scan/Bloom Filter much faster on Exadata
  - In my tests, balance point between two about the same on Exadata
    - (and that was without Hybrid Columnar Compression)
  - Now more difficult to justify overhead of bitmap indexes
  - Shifts balance away from CBQT Star Transformation
  - You should be less likely to want to add bitmap indexes.

EXADATA: STAR TRANSFORMATION –V- FULL SCAN/BLOOM FILTER

ISO Country Code	Country Name	Number of Sales in 1999	Number of Sales	Star Transformation		Full Scan-Bloom Filter	
				A-Time	Buffers	A-Time	Buffers
US	United States of America	2662	526212	.45	98056	.13	2173
DE	Germany	561	81978	.14	45538	.07	2173
JP	Japan	281	60183	.19	7118	.29	2068
GB	United Kingdom	391	58638	.41	4238	.25	2068
IT	Italy	257	42570	.12	43473	.07	2173
AU	Australia	228	33685	.04	8186	.06	2173
FR	France	161	33078	.07	23434	.07	2173
SG	Singapore	80	25253	.04	6984	.06	2173
CA	Canada	90	22858	.05	14156	.06	2173
ES	Spain	85	17136	.05	14319	.06	2173
DK	Denmark	89	16651	.03	5195	.06	2173
AR	Argentina	3	202	.02	5975	.06	2173
BR	Brazil	9	180	.03	7957	.05	2173
TR	Turkey	1	168	.02	4200	.05	2173
CN	China	4	19	.03	7342	.06	2173
PL	Poland	2	18	.03	7342	.06	2173
SA	Saudi Arabia	0	7	.02	4093	.05	2173

AUTONOMOUS DATA WAREHOUSE

AUTONOMOUS DATA WAREHOUSE (ADWC)

- ***'Additionally, Autonomous Data Warehouse does not require any tuning. Autonomous Data Warehouse is designed as a "load and go" service: you start the service, define tables, load data, and then run queries. When you use Autonomous Data Warehouse, no tuning is necessary. You do not need to consider any details about parallelism, partitioning, indexing, or compression. The service automatically configures the database for high-performance queries.'***
  - from Getting Started with Autonomous Data Warehouse

Oracle Cloud Infrastructure Documentation

- Connecting to an Autonomous Data Warehouse
- Autonomous Data Warehouse
- Loading Data from Files in the Cloud
  - JSON, Data Pump

AUTONOMOUS DATA WAREHOUSE: IMPORT WITH DATA PUMP

```
impdp admin/password@ADWC1_high \
     directory=data_pump_dir \
     credential=def_cred_name \
     dumpfile= https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/adwc/adwc_user/export%u.dmp \
     parallel=16 \
     partition_options=merge \
     transform=segment_attributes:n \
     transform=dwcs_cvt_iots:y \
     transform=constraint_use_default_index:y \
     exclude=index, cluster, indextype, materialized_view, materialized_view_log
                  , materialized_zonemap, db_link
```

WHAT IS AUTONOMOUS DATA WAREHOUSE?

It is Exadata (an engineered system)

- 12.2.0.1.0 Single Instance RAC
- SGA: 3.3Gb, PGA: 5Gb, RAM: 708Gb
- Intel(R) Xeon(R) CPU E7-8867 v4 @ 2.40GHz 84 CPUs/Core/Threads
- 12 storage servers

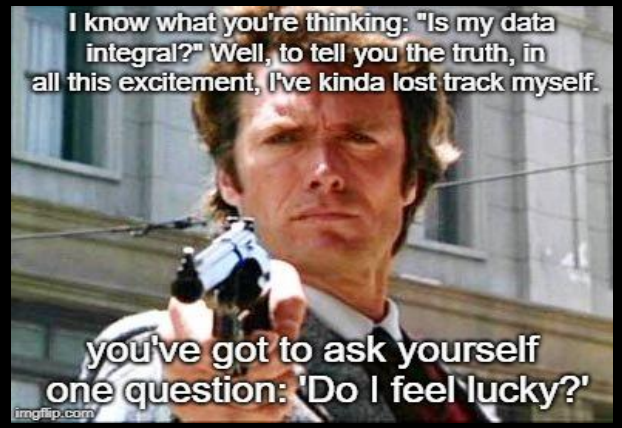
Delivered Sample Schemas

- SH Sales History (but it is a bit different)
- SSB – Star Schema Benchmark

FIRST ADVENTURES IN ADWC: THE DELIVERED SALES HISTORY DEMO

- SH Sales History in ADWC
  - Primary Key Constraints
    - Disabled, Not Validated, Rely
    - So also no Unique Indexes!
  - Foreign Keys
    - Disabled, Not Validated, Rely
    - Foreign Key columns are not indexed
    - *QUERY_REWRITE_INTEGRITY=TRUSTED*
  - Physical Attributes
    - No Partitioning!
    - PCTFREE defaults to 0
    - Hybrid Columnar Compressed (Query High)

- This is set up as static data
- New data loaded in
  - discrete batches, not continuously.
  - Direct path mode
- What if you accidentally run a load batch twice?



FIRST ADVENTURES IN ADWC: BUILDING MY OWN SALES HISTORY DEMO

I also built my own sales history demo per the standard scripts.

- I can build indexes
  - Primary and Foreign keys
  - Bitmap indexes (but why would I)
- I can specify
  - Tablespaces (but why would I, I can't create them)
  - Storage (PCTFREE, Compression, partitioning)
- I can do some alter session commands but not others
  - Can't *alter session set star_transformation_enabled = TRUE/FALSE*;
  - Can *ALTER SESSION SET optimizer_ignore_hints =FALSE;*
  - Only then can I get star transformation (but why would I)

AUTONOMOUS DATA WAREHOUSE: NON-DEFAULT PARAMETERS

QUERY_REWRITE_INTEGRITY=TRUSTED

- Constraints not enforced, but are RELYable
- Because we still want Foreign Key Join elimination

AUTONOMOUS DATA WAREHOUSE: NON-DEFAULT PARAMETERS

RESULT_CACHE_MODE=FORCE

- Always used by default
- perhaps it would worth setting on non-autonomous DW systems too?

```
Plan hash value: 2719715383

-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                          | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | A-Rows |   A-Time   |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                               |      1 |        |       |    38 (100)|          |     45 |00:00:00.01 |
|   1 |  RESULT CACHE                      | 8992dgrw00p4p9zu2vmq8p3nwg    |      1 |        |       |            |          |     45 |00:00:00.01 |
|   2 |   SORT GROUP BY                    |                               |      0 |    102 |  8262 |    38  (48)| 00:00:01 |      0 |00:00:00.01 |
|   3 |    HASH JOIN                       |                               |      0 |   3478 |   275K|    37  (46)| 00:00:01 |      0 |00:00:00.01 |
|   4 |     TABLE ACCESS STORAGE FULL      | PRODUCTS                      |      0 |     26 |   208 |     2   (0)| 00:00:01 |      0 |00:00:00.01 |
|   5 |     HASH JOIN                      |                               |      0 |   9819 |   699K|    35  (49)| 00:00:01 |      0 |00:00:00.01 |
|   6 |      JOIN FILTER CREATE            | :BF0000                       |      0 |    364 |  4368 |     2   (0)| 00:00:01 |      0 |00:00:00.01 |
|   7 |       TABLE ACCESS STORAGE FULL    | TIMES                         |      0 |    364 |  4368 |     2   (0)| 00:00:01 |      0 |00:00:00.01 |
|   8 |      HASH JOIN                     |                               |      0 |  39950 |  2379K|    33  (52)| 00:00:01 |      0 |00:00:00.01 |
|   9 |       JOIN FILTER CREATE           | :BF0001                       |      0 |   2413 | 94107 |     9  (12)| 00:00:01 |      0 |00:00:00.01 |
|  10 |        HASH JOIN                   |                               |      0 |   2413 | 94107 |     9  (12)| 00:00:01 |      0 |00:00:00.01 |
|  11 |         JOIN FILTER CREATE         | :BF0002                       |      0 |      1 |    18 |     2   (0)| 00:00:01 |      0 |00:00:00.01 |
|  12 |          TABLE ACCESS STORAGE FULL | COUNTRIES                     |      0 |      1 |    18 |     2   (0)| 00:00:01 |      0 |00:00:00.01 |
|  13 |         JOIN FILTER USE            | :BF0002                       |      0 |  55500 |  1138K|     7  (15)| 00:00:01 |      0 |00:00:00.01 |
|  14 |          TABLE ACCESS STORAGE FULL | CUSTOMERS                     |      0 |  55500 |  1138K|     7  (15)| 00:00:01 |      0 |00:00:00.01 |
|  15 |        JOIN FILTER USE             | :BF0000                       |      0 |    918K|    19M|    19  (58)| 00:00:01 |      0 |00:00:00.01 |
|  16 |         JOIN FILTER USE            | :BF0001                       |      0 |    918K|    19M|    19  (58)| 00:00:01 |      0 |00:00:00.01 |
|  17 |          TABLE ACCESS STORAGE FULL | SALES                         |      0 |    918K|    19M|    19  (58)| 00:00:01 |      0 |00:00:00.01 |
-----------------------------------------------------------------------------------------------------------------------------------
```

AUTONOMOUS DATA WAREHOUSE: NON-DEFAULT PARAMETERS

Parameter	Value	Comment
_default_pct_free	1	Default value for PCT_FREE, usually 10%, in order to pack data
_optimizer_gather_stats_on_load_all	TRUE	Statistics gathered on-line on direct path insert into table that is not empty
_optimizer_gather_stats_on_load_hist	TRUE	Histograms also gathered on-line
optimizer_ignore_hints	TRUE	ignore hints embedded in SQL
optimizer_ignore_parallel_hints	TRUE	ignore embedded parallel hints
result_cache_max_size	100M	Maximum size of result cache
result_cache_max_result	1	% of result cache that one result can use
inmemory_size	1G	Size of Inmemory Column Store
_cell_offload_vector_groupby	FALSE	In-Memory Aggregation optimisation is disabled

AUTONOMOUS DATA WAREHOUSE: SUMMARY

- It is still an Oracle database, everything that was true, is still true
  - Though built and configured in a very particular way.
- Build data model with primary and foreign keys throughout
  - I think working without enforced primary keys is brave. I don't think I would do that.
  - Single column equality joins between dimension and fact tables, no inequalities
  - Make foreign keys RELY if not ENABLEd to get Join Elimination
    - (works on multi-column FKs from 12.2, but there are bugs)
- Expect to load data by periodic incremental bulk load, rather than continuously
- Design to access data with Full Scan/Bloom Filter
  - This is approach is optimised on Exadata
  - Compress your data

WITH ANYTHING RUN ON AUTONOMOUS DATA WAREHOUSE?

You can alter ADWC settings to get all the traditional Star Transformation behaviour.

That is what I would describe as setting a field for bad bowling.

If you data warehouse data model/application does not fit with the generally accepted principle of good design, including those on the previous slide…

…then don't put it on the Autonomous Data Warehouse.

ADWC is another example of Oracle database software, that is optimised for applications that are built as Oracle would advise you to build them.

QUESTIONS