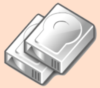
Chapter 9

Cardinality Estimation

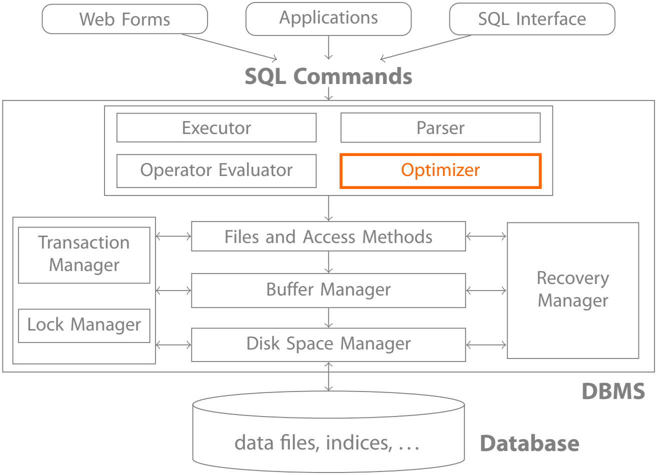
How Many Rows Does a Query Yield?

Architecture and Implementation of Database Systems
Summer 2014

Torsten Grust
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

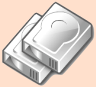


Cardinality Estimation



Cardinality Estimation

- A relational query optimizer performs a phase of **cost-based plan search** to identify the—presumably—“cheapest” alternative among a a set of equivalent execution plans (↗ Chapter on Query Optimization).
- Since page I/O cost dominates, the **estimated cardinality of a (sub-)query result** is crucial input to this search.
  - Cardinality typically measured in pages or rows.
- **Cardinality estimates** are also valuable when it comes to buffer “right-sizing” **before query evaluation starts** (*e.g.*, allocate B buffer pages and determine blocking factor b for external sort).



Estimating Query Result Cardinality

There are two principal approaches to query cardinality estimation:

1. **Database Profile.**
   Maintain statistical information about *numbers and sizes of tuples*, *distribution of attribute values* for **base relations**, as part of the **database catalog** (meta information) **during database updates**.

   - *Calculate* these parameters for **intermediate query results** based upon a (simple) statistical model during query optimization.

   - Typically, the statistical model is based upon the **uniformity** and **independence assumptions**.
   - Both are typically **not valid**, but they allow for simple calculations $\Rightarrow$ **limited accuracy**.
   - In order to improve accuracy, the system can record **histograms** to more closely model the actual value distributions in relations.

Estimating Query Result Cardinality

2 Sampling Techniques.
Gather the necessary characteristics of a query plan (base relations and intermediate results) at **query execution time**:

- **Run query on a small sample** of the input.
- **Extrapolate** to the full input size.

- It is crucial to find the right balance between sample size and the resulting accuracy.

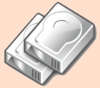
These slides focus on **1 Database Profiles.**

Database Profiles

Keep profile information in the **database catalog**. Update whenever SQL DML commands are issued (database updates):

Typical database profile for relation R

$\lvert R\rvert$	**number of records** in relation R
N_R	**number of disk pages** allocated for these records
$s(R)$	average **record size** (width)
$V(\mathtt{A}, R)$	**number of distinct values** of attribute `A`
$MCV(\mathtt{A}, R)$	**most common values** of attribute `A`
$MCF(\mathtt{A}, R)$	**frequency** of most common values of attribute `A`
$\vdots$	*possibly many more*



Database Profiles: IBM DB2

DB2 Excerpt of IBM DB2 catalog information for a TPC-H database

```
db2 => SELECT TABNAME, CARD, NPAGES
db2 (cont.) => FROM SYSCAT.TABLES
db2 (cont.) => WHERE TABSCHEMA = 'TPCH';

TABNAME        CARD                 NPAGES
-------------- -------------------- --------------------
ORDERS                      1500000                44331
CUSTOMER                     150000                 6747
NATION                           25                    2
REGION                            5                    1
PART                         200000                 7578
SUPPLIER                      10000                  406
PARTSUPP                     800000                31679
LINEITEM                    6001215               207888

  8 record(s) selected.
```

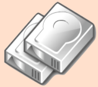
- **Note**: Column CARD $\equiv |R|$, column NPAGES $\equiv N_R$.

Database Profile: Assumptions

In order to obtain tractable cardinality estimation formulae, assume *one of the following*:

Uniformity & independence (simple, yet rarely realistic)

All values of an attribute uniformly appear with the same probability (even distribution). Values of different attributes are independent of each other.



Database Profile: Assumptions

In order to obtain tractable cardinality estimation formulae, assume *one of the following*:

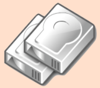
Uniformity & independence (simple, yet rarely realistic)

All values of an attribute uniformly appear with the same probability (even distribution). Values of different attributes are independent of each other.

Worst case (unrealistic)

No knowledge about relation contents at all. In case of a selection σ_p, assume all records will satisfy predicate p.

(May only be used to compute upper bounds of expected cardinality.)



Database Profile: Assumptions

In order to obtain tractable cardinality estimation formulae, assume *one of the following*:

Uniformity & independence (simple, yet rarely realistic)

All values of an attribute uniformly appear with the same probability (even distribution). Values of different attributes are independent of each other.

Worst case (unrealistic)

No knowledge about relation contents at all. In case of a selection σ_p, assume all records will satisfy predicate p.

(May only be used to compute upper bounds of expected cardinality.)

Perfect knowledge (unrealistic)

Details about the exact distribution of values are known. Requires huge catalog or prior knowledge of incoming queries.

(May only be used to compute lower bounds of expected cardinality.)

Cardinality Estimation for σ (Equality Predicate)

- Database systems typically operate under the **uniformity assumption**. We will come across this assumption multiple times below.

Query: $Q \equiv \sigma_{\mathtt{A}=c}(R)$

Selectivity $sel(\mathtt{A} = c)$	$MCF(\mathtt{A}, R)[c]$	if $c \in MCV(\mathtt{A}, R)$
Selectivity $sel(\mathtt{A} = c)$	$1/V(\mathtt{A}, R)$	**Uniformity**
Cardinality $\lvert Q \rvert$	$sel(\mathtt{A} = c) \cdot \lvert R \rvert$	
Record size $s(Q)$	$s(R)$	

Selectivity Estimation for σ (Other Predicates)

- **Equality between attributes** ($Q \equiv \sigma_{\mathtt{A}=\mathtt{B}}(R)$):
  Approximate selectivity by

$$sel(\mathtt{A} = \mathtt{B}) \quad = \quad 1/\max(V(\mathtt{A}, R), V(\mathtt{B}, R)) \ .$$

  (Assumes that each value of the attribute with fewer distinct values has a corresponding match in the other attribute.) **Independence**

- **Range selections** ($Q = \sigma_{\mathtt{A}>c}(R)$):
  In the database profile, **maintain the minimum and maximum value** of attribute $\mathtt{A}$ in relation R, $Low(\mathtt{A}, R)$ and $High(\mathtt{A}, R)$.

  Approximate selectivity by **Uniformity**

$$sel(\mathtt{A} > c) =$$

$$\begin{cases} \dfrac{High(\mathtt{A}, R) - c}{High(\mathtt{A}, R) - Low(\mathtt{A}, R)}, & Low(\mathtt{A}, R) \leqslant c \leqslant High(\mathtt{A}, R) \\ 0, & \text{otherwise} \end{cases}$$

Cardinality Estimation for π

- For $Q \equiv \pi_L(R)$, estimating the number of result rows is difficult ($L = \langle \mathtt{A}_1, \mathtt{A}_2, \dots, \mathtt{A}_n \rangle$: list of projection attributes):

$Q \equiv \pi_L(R)$ (duplicate elimination)

Cardinality $|Q|$

$$
|Q| = \begin{cases} V(\mathtt{A}, R), & \text{if } L = \langle \mathtt{A} \rangle \\ |R|, & \text{if keys of } R \in L \\ |R|, & \text{no dup. elim.} \\ \min\left(|R|, \prod_{\mathtt{A}_i \in L} V(\mathtt{A}_i, R)\right), & \text{otherwise} \end{cases}
$$

Independence

Record size $s(Q)$ $\sum_{\mathtt{A}_i \in L} s(\mathtt{A}_i)$

Cardinality Estimation for $\cup, \setminus, \times$

$Q \equiv R \cup S$

$$\begin{aligned} |Q| &\leqslant |R| + |S| \\ s(Q) &= s(R) = s(S) \qquad \text{schemas of } R,S \text{ identical} \end{aligned}$$

$Q \equiv R \setminus S$

$$\max(0, |R| - |S|) \leqslant |Q| \leqslant |R|$$
$$s(Q) = s(R) = s(S)$$

$Q \equiv R \times S$

$$\begin{aligned} |Q| &= |R| \cdot |S| \\ s(Q) &= s(R) + s(S) \\ V(\mathtt{A}, Q) &= \begin{cases} V(\mathtt{A}, R), & \text{for } \mathtt{A} \in R \\ V(\mathtt{A}, S), & \text{for } \mathtt{A} \in S \end{cases} \end{aligned}$$

Cardinality Estimation for $\bowtie$

- Cardinality estimation for the general join case is challenging.
- A special, yet very common case: **foreign-key relationship** between input relations R and S:

Establish a foreign key relationship (SQL)

```sql
CREATE TABLE R (A INTEGER NOT NULL,
                ...
                PRIMARY KEY (A));
CREATE TABLE S (...,
                A INTEGER NOT NULL,
                ...
                FOREIGN KEY (A) REFERENCES R);
```

$Q \equiv R \bowtie_{R.\mathtt{A}=S.\mathtt{A}} S$

The foreign key constraint guarantees $\pi_{\mathtt{A}}(S) \subseteq \pi_{\mathtt{A}}(R)$. Thus:

$$|Q| = |S| \; .$$

Cardinality Estimation for $\bowtie$

$Q \equiv R \bowtie_{R.\mathtt{A}=S.\mathtt{B}} S$

$$|Q| = \begin{cases} \dfrac{|R| \cdot |S|}{V(\mathtt{A}, R)}, & \pi_{\mathtt{B}}(S) \subseteq \pi_{\mathtt{A}}(R) \\[2ex] \dfrac{|R| \cdot |S|}{V(\mathtt{B}, S)}, & \pi_{\mathtt{A}}(R) \subseteq \pi_{\mathtt{B}}(S) \end{cases}$$

$$s(Q) = s(R) + s(S)$$

Histograms

- In realistic database instances, values are **not uniformly distributed** in an attribute's **active domain** (actual values found in a column).
- To keep track of this non-uniformity for an attribute `A`, maintain a **histogram** to **approximate the actual distribution**:

  1. Divide the active domain of `A` into adjacent intervals by selecting **boundary values** b_i.
  2. Collect statistical parameters for each interval between boundaries, *e.g.*,
     - # of rows r with $b_{i-1} < r.\mathtt{A} \leqslant b_i$, or
     - # of distinct `A` values in interval $(b_{i-1}, b_i]$.

- The histogram intervals are also referred to as **buckets.**

(↗ Y. Ioannidis: The History of Histograms (Abridged), *Proc. VLDB 2003*)

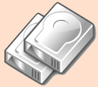
Histograms in IBM DB2

DB2® Histogram maintained for a column in a TPC-H database

```
SELECT SEQNO, COLVALUE, VALCOUNT
  FROM SYSCAT.COLDIST
 WHERE TABNAME = 'LINEITEM'
   AND COLNAME = 'L_EXTENDEDPRICE'
   AND TYPE = 'Q';

SEQNO COLVALUE          VALCOUNT
----- ----------------- --------
    1 +0000000000996.01     3001
    2 +0000000004513.26   315064
    3 +0000000007367.60   633128
    4 +0000000011861.82   948192
    5 +0000000015921.28  1263256
    6 +0000000019922.76  1578320
    7 +0000000024103.20  1896384
    8 +0000000027733.58  2211448
    9 +0000000031961.80  2526512
   10 +0000000035584.72  2841576
   11 +0000000039772.92  3159640
   12 +0000000043395.75  3474704
   13 +0000000047013.98  3789768
                  ⋮
```

- Catalog table `SYSCAT.COLDIST` also contains information like
  - the *n* most frequent values (and their frequency),
  - the number of distinct values in each bucket.
- Histograms may even be manipulated **manually** to tweak optimizer decisions.



Histograms

- Two types of histograms are widely used:

  1. **Equi-Width Histograms.**
     All buckets have the **same width**, *i.e.*, boundary $b_i = b_{i-1} + w$, for some fixed w.

  2. **Equi-Depth Histograms.**
     All buckets contain the **same number of rows** (*i.e.*, their width is varying).

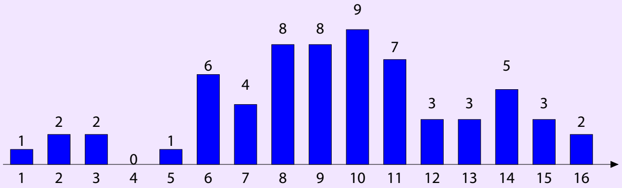
- Equi-depth histograms (2) are able to adapt to data skew (high non-uniformity).

- The number of buckets is the *tuning knob* that defines the **tradeoff** between estimation quality (**histogram resolution**) and **histogram size**: catalog space is limited.

Equi-Width Histograms

Example (Actual value distribution)

Column A of SQL type INTEGER (domain $\{\ldots,-2,-1,0,1,2,\ldots\}$).
Actual non-uniform distribution in relation *R*:

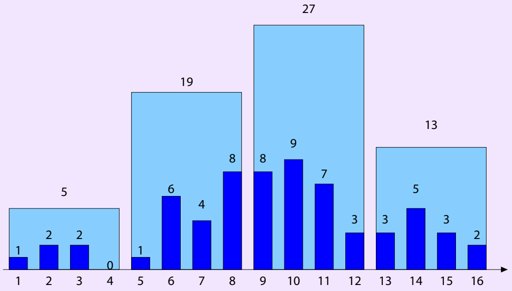


Equi-Width Histograms

- Divide **active domain** of attribute A into B buckets of equal width. The **bucket width** w will be

$$w = \frac{High(\mathtt{A}, R) - Low(\mathtt{A}, R) + 1}{B}$$

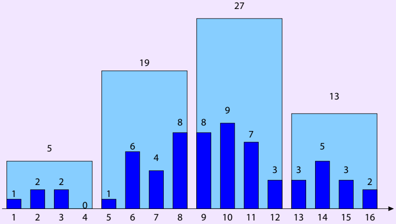
Example (Equi-width histogram ($B = 4$))



- Maintain **sum of value frequencies** in each bucket (in addition to bucket boundaries b_i).

Equi-Width Histograms: Equality Selections

Example ($Q \equiv \sigma_{\mathtt{A}=5}(R)$)



- Value 5 is in bucket $[5, 8]$ (with 19 tuples)
- Assume **uniform distribution within the bucket**:

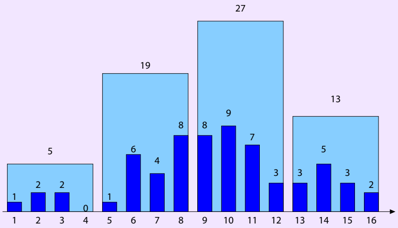
$$|Q| = 19/w = 19/4 \approx 5 \ .$$

Actual: $|Q| = 1$

What would be the cardinality under the uniformity assumption (no histogram)?

Equi-Width Histograms: Range Selections

Example ($Q \equiv \sigma_{\texttt{A}>7 \texttt{ AND } \texttt{A}\leqslant 16}(R)$)



- Query interval $(7, 16]$ covers buckets $[9, 12]$ and $[13, 16]$. Query interval touches $[5, 8]$.

$$|Q| = 27 + 13 + {}^{19}\!/_{4} \approx 45 \quad .$$

Actual: $|Q| = 48$

What would be the cardinality under the uniformity assumption (no histogram)?

Equi-Width Histogram: Construction

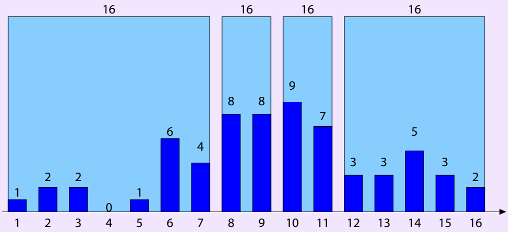
- To **construct** an equi-width histogram for relation R, attribute `A`:
  1. Compute boundaries b_i from $High(\mathtt{A}, R)$ and $Low(\mathtt{A}, R)$.
  2. Scan R once sequentially.
  3. While scanning, maintain B running tuple frequency counters, one for each bucket.
- If scanning R in step 2 is prohibitive, scan small sample $R_{sample} \subset R$, then scale frequency counters by $|R|/|R_{sample}|$.
- To **maintain** the histogram under insertions (deletions):
  1. Simply increment (decrement) frequency counter in affected bucket.

Equi-Depth Histograms

- Divide **active domain** of attribute `A` into B buckets of roughly the same number of tuples in each bucket, **depth** d of each bucket will be

$$d = \frac{|R|}{B} \ .$$

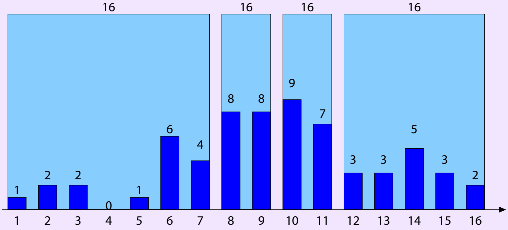
Example (Equi-depth histogram ($B = 4, d = 16$))



- Maintain **depth** (and bucket boundaries b_i).

Equi-Depth Histograms

Example (Equi-depth histogram ($B = 4, d = 16$))

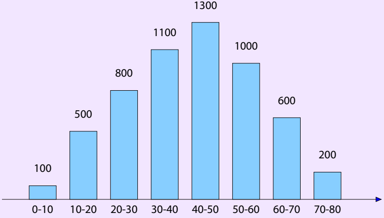


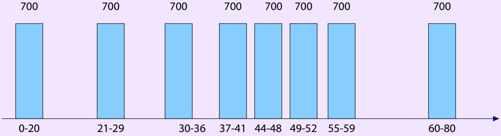
Intuition:

- High value frequencies are more important than low value frequencies.
- Resolution of histogram adapts to skewed value distributions.

Equi-Width vs. Equi-Depth Histograms

Example (Histogram on *customer age* attribute ($B = 8$, $|R| = $ 5,600))

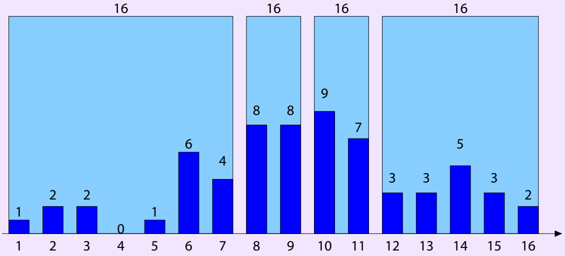




- Equi-depth histogram “invests” bytes in the densely populated customer age region between 30 and 59.

Equi-Depth Histograms: Equality Selections

Example ($Q \equiv \sigma_{\mathtt{A}=5}(R)$)



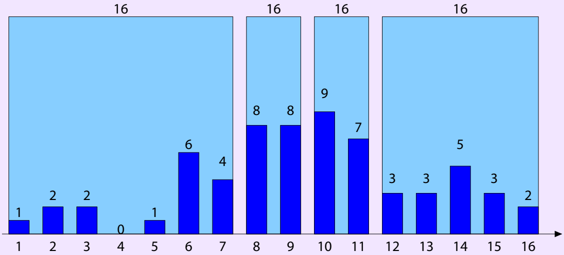
- Value 5 is in first bucket $[1, 7]$ (with $d = 16$ tuples)
- Assume **uniform distribution within the bucket**:

$$|Q| = d/7 = 16/7 \approx 2 \ .$$

(Actual: $|Q| = 1$)

Equi-Depth Histograms: Range Selections

Example ($Q \equiv \sigma_{\texttt{A}>5\ \texttt{AND}\ \texttt{A}\leqslant 16}(R)$)



- Query interval $(5, 16]$ covers buckets $[8, 9]$, $[10, 11]$ and $[12, 16]$ (all with $d = 16$ tuples). Query interval touches $[1, 7]$.

$$|Q| = 16 + 16 + 16 + {}^{2}\!/_{7} \cdot 16 \approx 53 \quad .$$

(Actual: $|Q| = 59$)

Equi-Depth Histograms: Construction

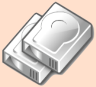
- To **construct** an equi-depth histogram for relation R, attribute `A`:
  1. Compute depth $d = {}^{|R|}/_{B}$.
  2. Sort R by sort criterion `A`.
  3. $b_0 = Low(\mathtt{A}, R)$, then determine the b_i by dividing the sorted R into chunks of size d.

Example ($B = 4$, $|R| = 64$)

1. $d = {}^{64}/_{4} = 16$.
2. Sorted $R.\mathtt{A}$:

   $\langle 1,2,2,3,3,5,6,6,6,6,6,6,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9,10,10,\ldots\rangle$
3. Boundaries of d-sized chunks in sorted R:

   $\langle \underbrace{1,2,2,3,3,5,6,6,6,6,6,6,7,7,7,7}_{b_1=7},\underbrace{8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9}_{b_2=9},10,10,\ldots\rangle$



A Cardinality (Mis-)Estimation Scenario

- Because exact cardinalities and estimated selectivity information is provided for base tables only, the DBMS relies on **projected cardinalities** for derived tables.
- In the case of foreign key joins, IBM DB2 promotes selectivity factors for one join input to the join result, for example.

Example (Selectivity promotion; $\mathtt{K}$ is key of S, $\pi_{\mathtt{A}}(R) \subseteq \pi_{\mathtt{K}}(S)$)

$$R \bowtie_{R.\mathtt{A}=S.\mathtt{K}} \left(\sigma_{\mathtt{B}=10}(S)\right)$$

If $sel(\mathtt{B} = 10) = x$, then assume that the join will yield $x \cdot |R|$ rows.

- Whenever the value distribution of $\mathtt{A}$ in R does not match the distribution of $\mathtt{B}$ in S, the cardinality estimate may be severly off.

A Cardinality (Mis-)Estimation Scenario

Example (Excerpt of a data warehouse)

Dimension table `STORE`:

STOREKEY	STORE_NUMBER	CITY	STATE	DISTRICT
...	...	...	...	...

} 63 rows

Dimension table `PROMOTION`:

PROMOKEY	PROMOTYPE	PROMODESC	PROMOVALUE
...	...	...	...

} 35 rows

Fact table `DAILY_SALES`:

STOREKEY	CUSTKEY	PROMOKEY	SALES_PRICE
...	...	...	...

} 754 069 426 rows

Let the tables be arranged in a **star schema**:

- The **fact table** references the **dimension tables**,
- the dimension tables are small/stable, the fact table is large/continously update on each sale.

$\Rightarrow$ Histograms are maintained for the dimension tables.

A Cardinality (Mis-)Estimation Scenario

DB2 Query against the data warehouse

Find the number of those sales in store `'01'` (18 of the overall 63 locations) that were the result of the sales promotion of type `'XMAS'` (**"star join"**):

```
SELECT COUNT(*)
FROM   STORE d1, PROMOTION d2, DAILY_SALES f
WHERE  d1.STOREKEY = f.STOREKEY
AND    d2.PROMOKEY = f.PROMOKEY
AND    d1.STORE_NUMBER = '01'
AND    d2.PROMOTYPE = 'XMAS'
```

The query yields 12,889,514 rows. The histograms lead to the following selectivity estimates:

$$
\begin{aligned}
sel(\texttt{STORE\_NUMBER = '01'}) &= {}^{18}/_{63} \quad (28.57\,\%) \\
sel(\texttt{PROMOTYPE = 'XMAS'}) &= {}^{1}/_{35} \quad (2.86\,\%)
\end{aligned}
$$

A Cardinality (Mis-)Estimation Scenario

DB2 Estimated cardinalities and selected plan

```
SELECT COUNT(*)
FROM   STORE d1, PROMOTION d2, DAILY_SALES f
WHERE  d1.STOREKEY = f.STOREKEY
AND    d2.PROMOKEY = f.PROMOKEY
AND    d1.STORE_NUMBER = '01'
AND    d2.PROMOTYPE = 'XMAS'
```

Plan fragment (top numbers indicates estimated cardinality):

```
                                     .
                                     .
                                     .
                                 6.15567e+06
                                   IXAND
                                   (   8)
                        /------------------+------------------\
                  2.15448e+07                             2.15448e+08
                    NLJOIN                                  NLJOIN
                    (   9)                                  (  13)
              /---------+--------\                    /---------+--------\
             1                2.15448e+07            18                1.19694e+07
           FETCH                IXSCAN              FETCH                IXSCAN
           (  10)               (  12)              (  14)               (  16)
         /---+---\                |               /---+---\                |
       35        35          7.54069e+08        18        63          7.54069e+08
     IXSCAN  TABLE: DB2DBA  INDEX: DB2DBA     IXSCAN  TABLE: DB2DBA   INDEX: DB2DBA
     (  11)      PROMOTION    PROMO_FK_IDX    (  15)      STORE           STORE_FK_IDX
       |                                        |
       35                                       63
  INDEX: DB2DBA                            INDEX: DB2DBA
 PROMOTION_PK_IDX                          STOREX1
```

IBM DB2: Statistical Views

- To provide database profile information (estimate cardinalities, value distributions, . . .) for **derived tables**:
  1. Define a **view** that precomputes the derived table (or possibly a small sample of it, IBM DB2: 10 %),
  2. use the view result to gather and keep **statistics**, then delete the result.

DB2® Statistical views

```
CREATE VIEW sv_store_dailysales AS
   (SELECT s.*
    FROM   STORE s, DAILY_SALES ds
    WHERE  s.STOREKEY = ds.STOREKEY);

CREATE VIEW sv_promotion_dailysales AS
   (SELECT p.*
    FROM   PROMOTION p, DAILY_SALES ds
    WHERE  p.PROMOKEY = ds.PROMOKEY);

ALTER VIEW sv_store_dailysales ENABLE QUERY OPTIMIZATION;
ALTER VIEW sv_promotion_dailysales ENABLE QUERY OPTIMIZATION;

RUNSTATS ON TABLE sv_store_dailysales WITH DISTRIBUTION;
RUNSTATS ON TABLE sv_promotion_dailysales WITH DISTRIBUTION;
```

Cardinality Estimation with Statistical Views

DB2 Estimated cardinalities and selected plan after reoptimization

```
                              1.04627e+07
                                IXAND
                               (   8)
                    /------------------+------------------\
           6.99152e+07                                  1.12845e+08
             NLJOIN                                       NLJOIN
             (   9)                                       (  13)
        /---------+--------\                         /---------+--------\
       18              3.88418e+06                  1               1.12845e+08
     FETCH               IXSCAN                   FETCH                IXSCAN
     (  10)              (  12)                   (  14)               (  16)
    /---+---\              |                     /---+---\               |
  18         63        7.54069e+08          35           35         7.54069e+08
 IXSCAN  TABLE:DB2DBA  INDEX: DB2DBA      IXSCAN  TABLE: DB2DBA   INDEX: DB2DBA DB2DBA
 (  11)    STORE         STORE_FK_IDX     (  15)     PROMOTION        PROMO_FK_IDX
   |                                        |
  63                                       35
INDEX: DB2DBA                            INDEX: DB2DBA
STOREX1                                  PROMOTION_PK_IDX
```

Note new estimated selectivities *after* join:

- Selectivity of `PROMOTYPE = 'XMAS'` now only 14.96 % (was: 2.86 %)
- Selectivity of `STORE_NUMBER = '01'` now 9.27 % (was: 28.57 %)