Chapter 9
Cardinality Estimation
How Many Rows Does a Query Yield?

Architecture and Implementation of Database Systems
Winter 2010/11
Cardinality Estimation

Estimating Operator Cardinality
- Selection $\sigma$
- Projection $\pi$
- Set Operations $\cup$, $\setminus$, $\times$
- Join $\bowtie$

Histograms
- Equi-Width
- Equi-Depth

Statistical Views

DBMS
- data files, indices, ...

Database

Operator Evaluator
- Optimizer

SQL Commands
- Executor
- Parser

Files and Access Methods
- Files and Access Methods

Buffer Manager

Transaction Manager

Lock Manager

Parser

Web Forms
- Applications
- SQL Interface

Parser

Web Forms
- Applications
- SQL Interface

DBMS
- data files, indices, ...

Database

92
A relational query optimizer performs a phase of **cost-based plan search** to identify the—presumably—“cheapest” alternative among 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 ⇒ limited accuracy.
   - In order to improve accuracy, the system can record histograms to more closely model the actual value distributions in relations.
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 Database Profiles.
Database Profiles

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

<table>
<thead>
<tr>
<th>Typical database profile for relation $R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
</tr>
<tr>
<td>$N_R$</td>
</tr>
<tr>
<td>$s(R)$</td>
</tr>
<tr>
<td>$V(A, R)$</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
### Database Profiles: IBM DB2

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

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

<table>
<thead>
<tr>
<th>TABNAME</th>
<th>CARD</th>
<th>NPAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDERS</td>
<td>1500000</td>
<td>44331</td>
</tr>
<tr>
<td>CUSTOMER</td>
<td>150000</td>
<td>6747</td>
</tr>
<tr>
<td>NATION</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>REGION</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>PART</td>
<td>200000</td>
<td>7578</td>
</tr>
<tr>
<td>SUPPLIER</td>
<td>10000</td>
<td>406</td>
</tr>
<tr>
<td>PARTSUPP</td>
<td>800000</td>
<td>31679</td>
</tr>
<tr>
<td>LINEITEM</td>
<td>6001215</td>
<td>207888</td>
</tr>
</tbody>
</table>

8 record(s) selected.
```

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

---

Cardinality Estimation

- Estimating Operator Cardinality
  - Selection $\sigma$
  - Projection $\pi$
  - Set Operations $\cup, \setminus, \times$
  - Join $\bowtie$

- Histograms
  - Equi-Width
  - Equi-Depth

- Statistical Views

---

9.7

Cardinality Estimation

- Torsten Grust
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. 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. Values of different attributes are independent of each other.

**Worst case (unrealistic)**

No knowledge about relation contents at all. In case of a selection $\sigma_p$, 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. Values of different attributes are independent of each other.

**Worst case (unrealistic)**

No knowledge about relation contents at all. In case of a selection $\sigma_p$, 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 $\sigma$ (Equality Predicate)

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

**Selectivity** $sel(A = c)$ \hspace{1cm} 1/$V(A, R)$ \hspace{1cm} **Uniformity**

**Cardinality** $|Q|$ \hspace{1cm} $sel(A = c) \cdot |R|$

**Record size** $s(Q)$ \hspace{2cm} $s(R)$

**Value Distribution** $V(A', Q)$

\[
V(A', Q) = \begin{cases} 
1, & \text{for } A' = A, \\
\frac{c(V(A', R), |Q|)}{c(V(A', R))}, & \text{otherwise.}
\end{cases}
\]

with (\# of distinct colors obtained by drawing $r$ balls from a bag of balls of $m$ colors)$^{1}$:

\[
c(m, r) = \begin{cases} 
    r, & \text{for } r < m/2, \\
    \frac{r + m}{3}, & \text{for } m/2 \leq r < 2m, \\
    m, & \text{for } r \geq 2m
\end{cases}
\]

---

$^{1}$“Selection without replacement”: $c(m, r) = m \cdot (1 - (1 - 1/m)^r)$. 

Cardinality Estimation
Torsten Grust

Cardinality Estimation
Database Profiles
Assumptions
Estimating Operator Cardinality
Selection $\sigma$
Projection $\pi$
Set Operations $\cup, \setminus, \times$
Join $\Join$
Histograms
Equi-Width
Equi-Depth
Statistical Views
Selectivity Estimation for $\sigma$ (Other Predicates)

- **Equality between attributes** ($Q \equiv \sigma_{A=B}(R)$):
  Approximate selectivity by
  \[
  sel(A = B) = \frac{1}{\max(V(A, R), V(B, R))}.
  \]
  ( Assumes that each value of the attribute with fewer distinct values has a corresponding match in the other attribute.)

- **Range selections** ($Q = \sigma_{A>c}(R)$):
  In the database profile, maintain the minimum and maximum value of attribute $A$ in relation $R$, $Low(A, R)$ and $High(A, R)$.
  Approximate selectivity by
  \[
  sel(A > c) =
  \begin{cases}
  \frac{High(A, R) - c}{High(A, R) - Low(A, R)}, & Low(A, R) \leq c \leq High(A, R) \\
  0, & \text{otherwise}
  \end{cases}
  \]
Cardinality Estimation for \( \pi \)

- For \( Q \equiv \pi_L(R) \), estimating the number of result rows is difficult (\( L = \langle A_1, A_2, \ldots, A_n \rangle \): list of projection attributes):

\[
|Q| = \begin{cases} 
V(A, R), & \text{if } L = \langle A \rangle \\
|R|, & \text{if keys of } R \in L \\
|R|, & \text{no dup. elim.} \\
\min(|R|, \prod_{A_i \in L} V(A_i, R)), & \text{otherwise}
\end{cases}
\]

Independence

Record size \( s(Q) = \sum_{A_i \in L} s(A_i) \)

Val. Dist. \( V(A_i, Q) = V(A_i, R) \) for \( A_i \in L \)
Cardinality Estimation for \( \cup, \setminus, \times \)

### Case 1: \( Q \equiv R \cup S \)

\[
|Q| \leq |R| + |S| \\
s(Q) = s(R) = s(S) \quad \text{schemas of } R, S \text{ identical} \\
V(A, Q) \leq V(A, R) + V(A, S)
\]

### Case 2: \( Q \equiv R \setminus S \)

\[
\max(0, |R| - |S|) \leq |Q| \leq |R| \\
s(Q) = s(R) = s(S) \\
V(A, Q) \leq V(A, R)
\]

### Case 3: \( Q \equiv R \times S \)

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

**Database Profiles**

- Assumptions
- Estimating Operator
- Cardinality
- Selection \( \sigma \)
- Projection \( \pi \)
- Set Operations \( \cup, \setminus, \times \)
- Join \( \Join \)

**Histograms**

- Equi-Width
- Equi-Depth

**Statistical Views**
Cardinality Estimation for

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

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

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

The foreign key constraint guarantees $\pi_A(S) \subseteq \pi_A(R)$. Thus:

$$|Q| = |S|.$$
Cardinality Estimation for \( Q = R \bowtie_{R.A=S.B} S \)

\[
|Q| = \begin{cases} 
\frac{|R| \cdot |S|}{V(A, R)}, & \pi_B(S) \subseteq \pi_A(R) \\
\frac{|R| \cdot |S|}{V(B, S)}, & \pi_A(R) \subseteq \pi_B(S) 
\end{cases}
\]

\[
s(Q) = s(R) + s(S)
\]

\[
V(A', Q) \leq \begin{cases} 
V(A', R), & \text{if } A' \text{ attribute in } R \\
V(A', S), & \text{otherwise}
\end{cases}
\]
**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 \( b_{i-1} < r \cdot A \leq b_i \), 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*)
Histogarms in IBM DB2

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

<table>
<thead>
<tr>
<th>SEQNO</th>
<th>COLVALUE</th>
<th>VALCOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+00000000000996.01</td>
<td>3001</td>
</tr>
<tr>
<td>2</td>
<td>+0000000004513.26</td>
<td>315064</td>
</tr>
<tr>
<td>3</td>
<td>+0000000007367.60</td>
<td>633128</td>
</tr>
<tr>
<td>4</td>
<td>+0000000011861.82</td>
<td>948192</td>
</tr>
<tr>
<td>5</td>
<td>+0000000015921.28</td>
<td>1263256</td>
</tr>
<tr>
<td>6</td>
<td>+0000000019922.76</td>
<td>1578320</td>
</tr>
<tr>
<td>7</td>
<td>+0000000024103.20</td>
<td>1896384</td>
</tr>
<tr>
<td>8</td>
<td>+0000000027733.58</td>
<td>2211448</td>
</tr>
<tr>
<td>9</td>
<td>+0000000031961.80</td>
<td>2526512</td>
</tr>
<tr>
<td>10</td>
<td>+0000000035584.72</td>
<td>2841576</td>
</tr>
<tr>
<td>11</td>
<td>+0000000039772.92</td>
<td>3159640</td>
</tr>
<tr>
<td>12</td>
<td>+0000000043395.75</td>
<td>3474704</td>
</tr>
<tr>
<td>13</td>
<td>+0000000047013.98</td>
<td>3789768</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• 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 \textbf{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 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.
Example (Actual value distribution)

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

![Histogram](image)
Equi-Width Histograms

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

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

**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 \((Q \equiv \sigma_{A=5}(R))\)

- 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 \( Q \equiv \sigma_{A > 7 \text{ AND } A \leq 16}(R) \)

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

\[
|Q| = 27 + 13 + \frac{19}{4} \approx 45 .
\]

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 $High(A, R)$ and $Low(A, R)$.
  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 \((Q \equiv \sigma_{A=5}(R))\)

- Value 5 is in first bucket \([1, 7]\) (with \(d = 16\) tuples)
- Assume **uniform distribution within the bucket**:
  \[|Q| = \frac{d}{7} = \frac{16}{7} \approx 2\]

(Actual: \(|Q| = 1\))
Example \((Q \equiv \sigma_{A \geq 5 \text{ AND } A \leq 16}(R))\)

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

\[
|Q| = 16 + 16 + 16 + \frac{2}{7} \cdot 16 \approx 53.
\]

(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. $b_0 = \text{Low}(A, R)$, 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 $R.A$:
   $$\langle 1,2,3,3,5,6,6,6,6,6,6,7,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9,9,9,10,10,\ldots \rangle$$
3. Boundaries of $d$-sized chunks in sorted $R$:
   $$\langle 1,2,2,3,3,5,6,6,6,6,6,6,7,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9,9,9,9,9,9,10,10,\ldots \rangle$$
   $b_1 = 7$
   $b_2 = 9$
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; \( K \) is key of \( S \), \( \pi_A(R) \subseteq \pi_K(S) \))

\[
R \bowtie_{R.A=S.K} (\sigma_{B=10}(S))
\]

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

• Whenever the value distribution of \( A \) in \( R \) does not match the distribution of \( B \) in \( S \), the cardinality estimate may be severely off.
A Cardinality (Mis-)Estimation Scenario

Example (Excerpt of a data warehouse)

Dimension table STORE:

<table>
<thead>
<tr>
<th>STOREKEY</th>
<th>STORE_NUMBER</th>
<th>CITY</th>
<th>STATE</th>
<th>DISTRICT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

63 rows

Dimension table PROMOTION:

<table>
<thead>
<tr>
<th>PROMOKEY</th>
<th>PROMOTYPE</th>
<th>PROMODESC</th>
<th>PROMOVALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

35 rows

Fact table DAILY_SALES:

<table>
<thead>
<tr>
<th>STOREKEY</th>
<th>CUSTKEY</th>
<th>PROMOKEY</th>
<th>SALES_PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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/continuously update on each sale.

⇒ Histograms are maintained for the dimension tables.
A Cardinality (Mis-)Estimation Scenario

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*”):

```sql
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:

\[
\text{sel}(\text{STORE_NUMBER} = '01') = \frac{18}{63} \quad (28.57\%)
\]

\[
\text{sel}(\text{PROMOTYPE} = 'XMAS') = \frac{1}{35} \quad (2.86\%)
\]
A Cardinality (Mis-)Estimation Scenario

**Estimated cardinalities and selected plan**

```sql
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):

```
<table>
<thead>
<tr>
<th>1</th>
<th>2.15448e+07</th>
<th>2.15448e+08</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6.15567e+06</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

### Cardinality Estimation
Torsten Grust

### Database Profiles
Assumptions

### Estimating Operator Cardinality
Selection \( \sigma \)
Projection \( \pi \)
Set Operations \( \cup, \setminus, \times \)
Join \( \bowtie \)

### Histograms
Equi-Width
Equi-Depth

### Statistical Views
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.

```sql
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

Estimated cardinalities and selected plan after reoptimization

```
+-------------------+------------------+
IXAND              |------------------|
+-------------------+------------------|
6.99152e+07        |1.12845e+08      |
NLJOIN             |                 |
+-------------------+------------------|
6.99152e+07        |1.12845e+08      |
NLJOIN             |                 |
+-------------------+------------------|
1.04627e+07        |                 |
IXAND              |------------------|
+-------------------+------------------|
3.88418e+06        |1.0845e+08       |
IXSCAN             |                 |
+-------------------+------------------|
18                 |1.0845e+08       |
FETCH              |IXSCAN            |
+-------------------+------------------|
63                 |7.54069e+08      |
IXSCAN             |STORE             |
+-------------------+------------------|
63                 |7.54069e+08      |
IXSCAN             |INDEX: DB2DBA    |
+-------------------+------------------|
7.54069e+08        |35                |
INDEX: DB2DBA      |PROMOTION         |
+-------------------+------------------|
35                 |7.54069e+08      |
IXSCAN             |INDEX: DB2DBA    |
+-------------------+------------------|
35                 |7.54069e+08      |
IXSCAN             |STOREX1           |
+-------------------+------------------|
35                 |7.54069e+08      |
IXSCAN             |INDEX: DB2DBA    |
+-------------------+------------------|
7.54069e+08        |35                |
INDEX: DB2DBA      |PROMOTION_PK_IDX |
+-------------------+------------------|
35                 |7.54069e+08      |
IXSCAN             |INDEX: DB2DBA    |
+-------------------+------------------|
7.54069e+08        |35                |
INDEX: DB2DBA      |PROMOTION_PK_IDX |
+-------------------+------------------|
35                 |7.54069e+08      |
IXSCAN             |INDEX: DB2DBA    |
+-------------------+------------------|
7.54069e+08        |35                |
INDEX: DB2DBA      |PROMOTION_PK_IDX |
+-------------------+------------------|
35                 |7.54069e+08      |
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 %)
```