

B16

# Speeding up DB2 UDB Using Sampling

Peter J. Haas



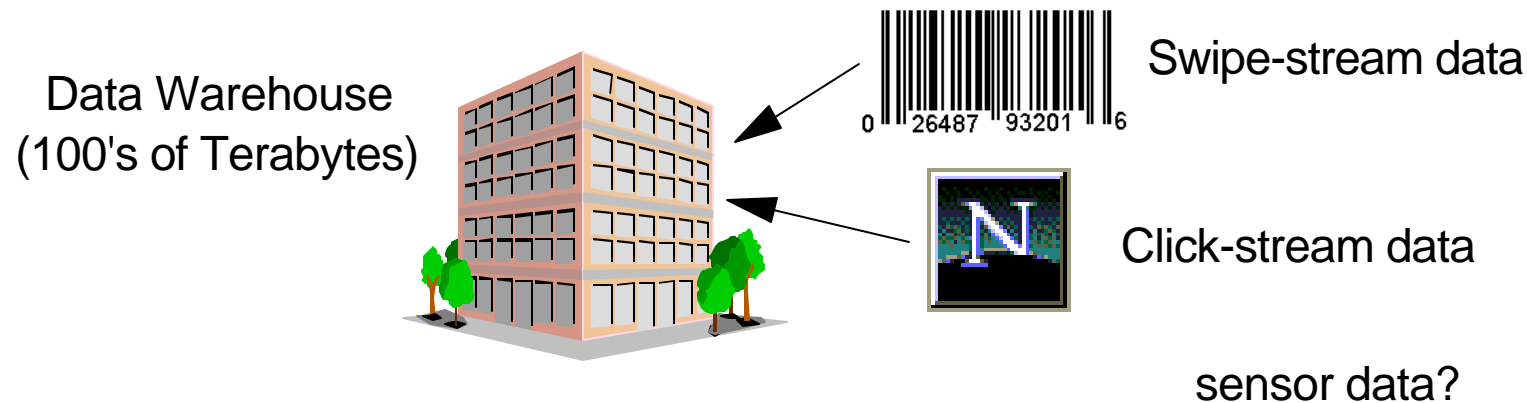
IBM Data Management Technical Conference

Anaheim, CA

Sept 9 - 13, 2002

# The Challenge: Massive Data Volumes

- Online data volume is growing faster than Moore's law

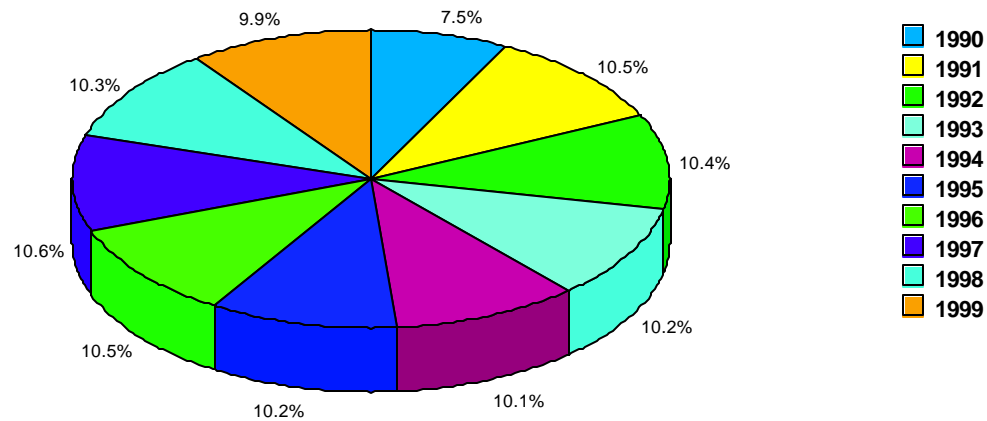


- Users want to run:
  - BI queries / OLAP queries
  - Advanced analytical/mining/statistical algorithms
  - Exploratory/interactive analyses ("fishing expeditions")
- A big problem:
  - Many algorithms don't scale with increasing data volume
  - Fishing expeditions infeasible

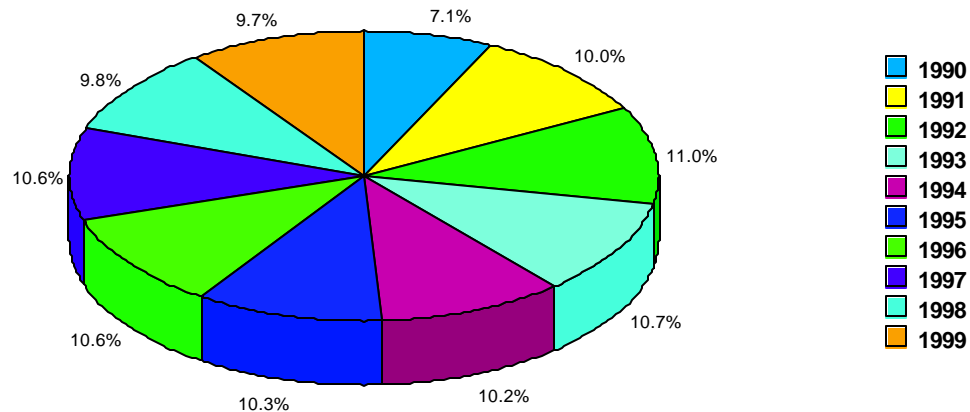
# Solution: Sampling

- Processing less data => huge performance improvement
- Approximate answers often suffice

## No Sampling (4 min.)



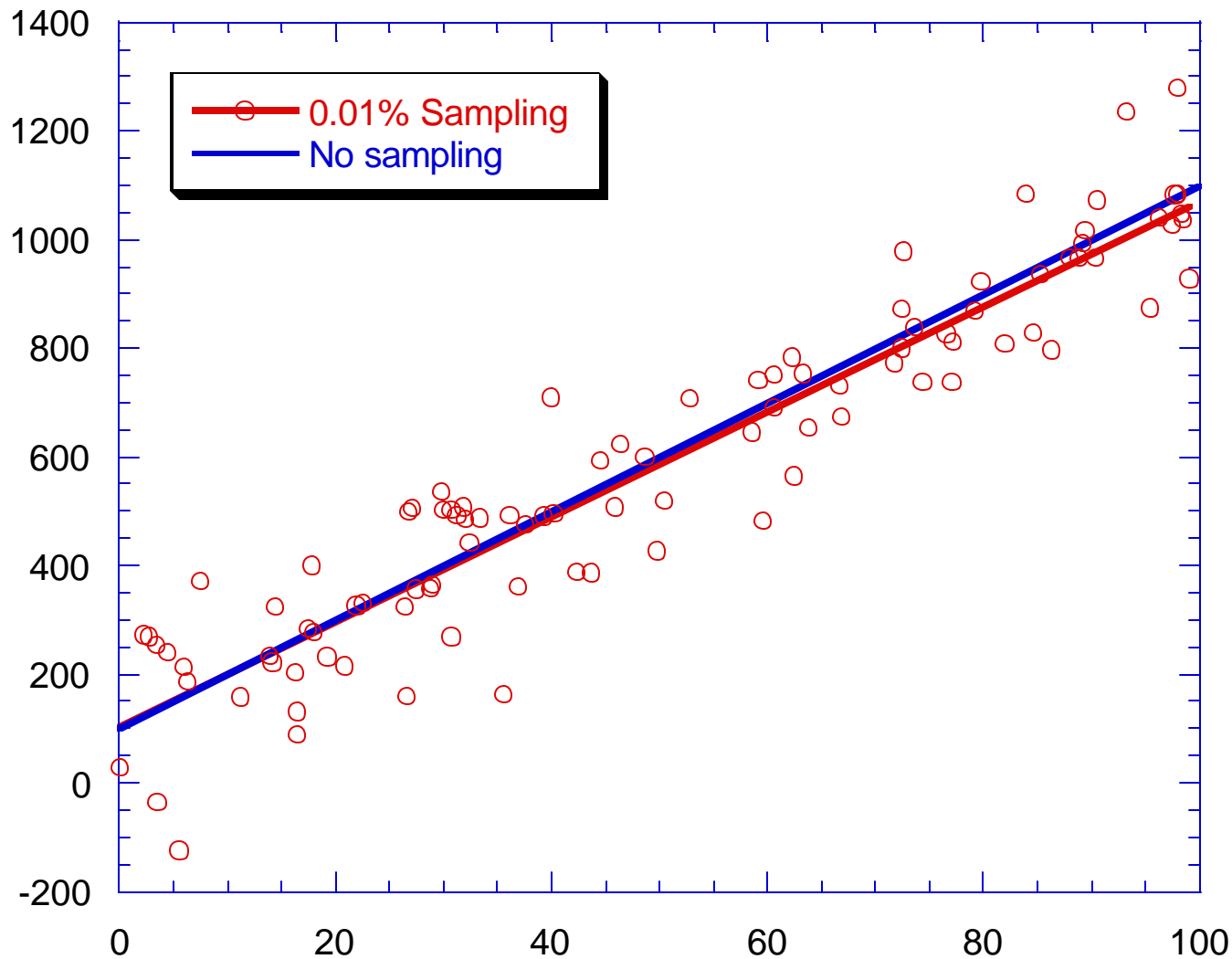
## 1% sampling (2 sec)



# Sampling for Scalable Analytics

- Example: Linear regression on a million-row table

## Line Fit



# Main Uses of Database Sampling

- Quick approximate answers to aggregation queries

```
SELECT AVG(sales) FROM salesdata  
GROUP BY year
```

- Quick approximate answers to analytical queries

```
SELECT REGR_SLOPE(sales, ad_budget),  
       REGR_ICPT(sales, ad_budget)  
FROM ad_campaign_data
```

- Compressed representation of dataset
  - For auditing
  - For "fuzzy exploration"
  - For input into mining/analysis applications

# Road Map

- Basic database sampling schemes
- Sampling in DB2 V7 and earlier
- The proposed ISO sampling standard
- Sampling support in DB2 V8
- Combining sampling and SQL
- Some research directions

# Database Sampling Schemes

- Focus on sampling from a single table
- Many kinds of samples
  - Often want "representative" sample
  - Often formed using probabilistic selection
  - Sometimes want more of some types of rows than others
- Close connection between sampling and **estimation** methods
  - Estimation formulas need to take sampling scheme into account
  - Often want measure of **precision** of estimate
    - **Standard error**: measures variability of an estimate

Large-table results:

-----  
within +/- 1.96 SE's: 95%  
within +/- 2.58 SE's: 99%

# Row-Level Bernoulli Sampling

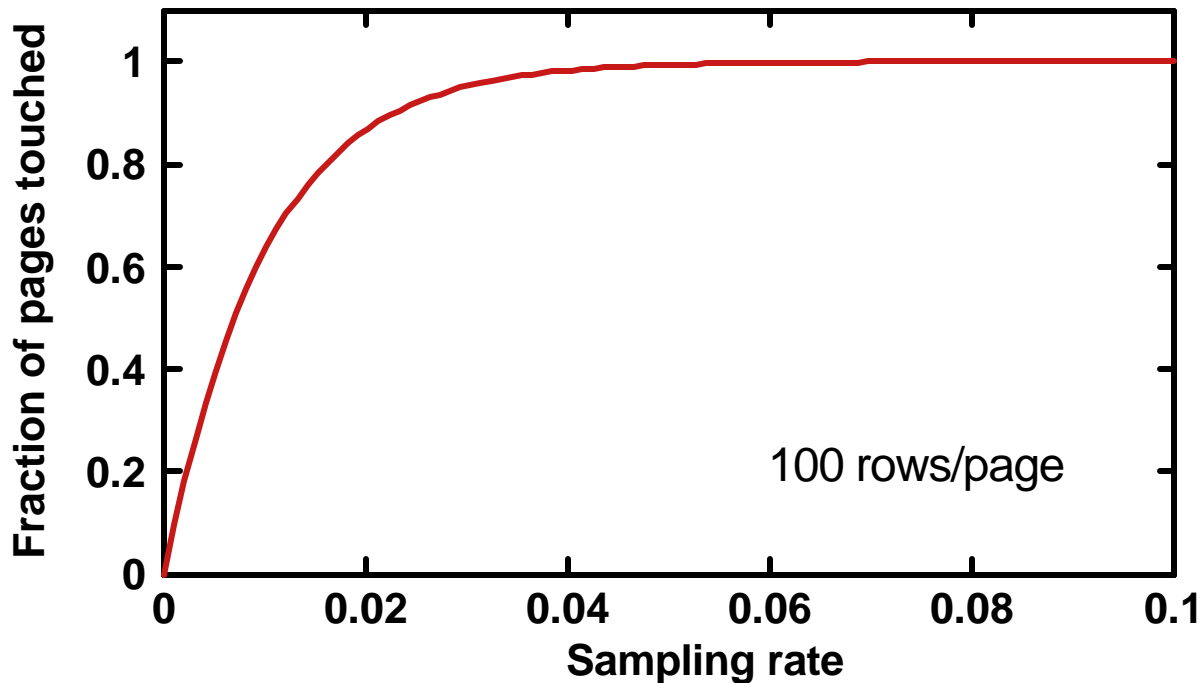
- "Coin flip" sampling
  - Flip a weighted coin for each row (independently)
  - Include row into sample if coin is "heads"
  - Probability of "heads" is called the "inclusion probability"
  - Each row in the table is equally likely to be included
- Sample size is **random**
  - Expected size = (inclusion probability) \* (# rows in table)
  - With high probability, actual size within  $\pm q$  of expected size
    - $q = 2 (Np(1-p))^{1/2}$   
( $p$  = inclusion probability and  $N$  = population size)
    - ex:  $N = 1,000,000$ ,  $p = 1/2$

actual size within 0.2% of expected size  
with high probability



# Row-Level Bernoulli Sampling (Cont'd)

- Advantages:
  - Integrates well with SQL (treat sampling clause as a predicate)
  - Easy to parallelize
- Disadvantage: **High I/O cost**
  - Naive implementation: touch every page
  - Clever implementation: touch (almost) every page



# Page-Level Bernoulli Sampling

- Sample **pages** first
  - Flip a coin for each page
  - Use all rows on a page
- Unlike row-level, inclusion of rows is not independent
- Advantage: **Dramatic I/O savings**
  - More efficient use of rows on pages touched
  - Quick algorithms for generating page lists
  - Can exploit prefetching

# The Fly in the Ointment: Clustering

- What if data values are clustered on pages?



page 1	1, 0, 1, 0, ...
page 2	0, 0, 1, 1, ...
...	...
page 10,000	1, 0, 1, 0, ...

Table 1 (Unclustered)

VS

1, 1, 1, 1, ...
0, 0, 0, 0, ...
...
1, 1, 1, 1, ...

Table 2 (Clustered)

$10^6$  rows  
100 rows/pg

- Example: Take 1% sample of Table 2 to estimate the sum
  - Estimator:  $100 \times$  (sum of numbers in sample)
  - **Row-level sampling**: average value = 500,000 and SE = 7,036
  - **Page-level sampling**: average value = 500,000 and SE = 70,356
  - Estimated SE if page-level treated as row-level:  $\approx 7,036$  (on avg)
- Moral of the story:
  - Both methods yield correct answer on average (**unbiased**)
  - Page-level sampling has **10x** the error for clustered data
  - **Must use correct formula** to estimate the SE (false optimism)

# Clustering: What's Going On?

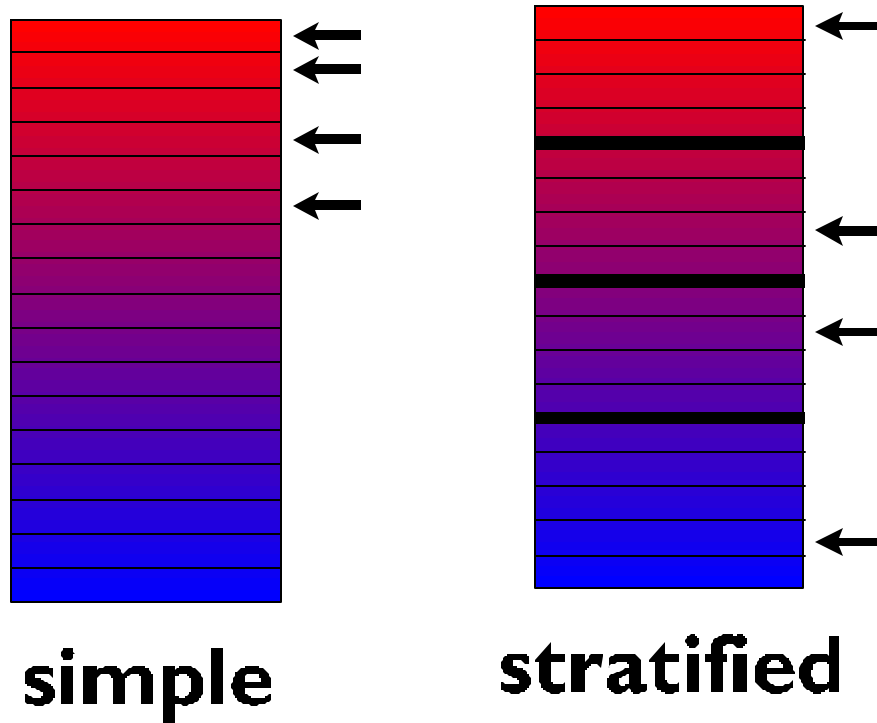
- **Q:** Why is page-level sampling less precise (though much cheaper) than row-level sampling when the data is clustered?
- **A1:** What you see is more dependent on luck-of-the-draw
  - E.g., sum = 0 versus sum = 100
- **A2:** Not much information from a sampled page
  - First row contains most of the information
  - Rest of rows "wasted"
  - Want to see a lot of pages (i.e., row level sampling)
- Issue: Detecting clustering (will discuss later)

# Simple Random Sampling (SRS)

- Sample size  $n$  is **deterministic** and prespecified
- Each sample of size  $n$  is equally likely
- Conceptually, to create a SRS:
  - Choose first row  $r_1$  randomly + uniformly from table  $T$
  - Choose  $r_2$  randomly + uniformly from  $T - \{r_1\}$
  - Choose  $r_3$  randomly + uniformly from  $T - \{r_1, r_2\}$
  - etc.
- Can also choose with replacement (SRSWR)

# Stratified Sampling

- Reduces "bunching up" due to luck-of-the-draw



- Data values within strata should be similar
  - "If you've seen one, you've seen 'em all"

# Cluster Sampling

- Population members grouped into disjoint clusters
- Sample clusters at random (Bernoulli or SRS or ...)
- Include every member of a sampled cluster into final sample
- Ex: Page-level Bernoulli sampling (cluster = page)
- Ex: Sample of customer transactions
  - Sample customers at random
  - Include all transactions for each sampled customer

# Road Map

- Basic database sampling schemes
- Sampling in DB2 V7.1 and earlier
- The proposed ISO sampling standard
- Sampling support in DB2 V8
- Combining sampling and SQL
- Some research directions
  - Integrating sampling and analytics
  - Interactive querying

# Row-Level Bernoulli Sampling

- Use **RAND()** function dynamically
  - RAND returns a uniform random number between 0 and 1
  - Ex: let **:samp\_rate** be a host variable (e.g., samp\_rate = 0.01)

```
SELECT loc.country AS country, year(t.pdate) AS year,  
       sum(ti.sales) / :samp_rate AS est_sales,  
FROM trans t, transitem ti, loc loc  
WHERE t.transid = ti.transid AND loc.locid = t.locid  
      AND rand() < :samp_rate  
GROUP BY loc.country, year(t.pdate)
```

country	year	est_sales
-----	-----	-----
USA	1998	127505
USA	1999	236744
GERMANY	1998	86278
GERMANY	1999	126737

...

Need to scale up  
query result

- Problem: High I/O cost (as always)

# Simple Random Sampling

- Append column of random numbers to table

```
ALTER TABLE mytable ADD COLUMN randnum DOUBLE;  
UPDATE mytable SET randnum = RAND();
```

- Can obtain multiple, disjoint SRS's
  - Ex: An SRS of 100 rows

```
WITH dt AS (SELECT c1,...,cN,  
                ROWNUMBER() OVER (ORDER BY randnum) AS num  
            FROM mytable)  
SELECT c1,...,cN FROM dt  
WHERE num >= 500 AND num < 600
```

- Can cluster by randnum on disk to amortize cost
  - Sample = scan
  - Degrades performance of other queries (database design)
  - Refresh is expensive

# Road Map

- Basic database sampling schemes
- Sampling in DB2 V7.1 and earlier
- The proposed ISO sampling standard
- Sampling support in DB2 V8
- Combining sampling and SQL
- Some research directions
  - Integrating sampling and analytics
  - Interactive querying

# The Proposed ISO Sampling Standard

- Can place sampling clause after any SQL table reference

```
SELECT *  
FROM t1 TABLESAMPLE BERNOULLI(10.0)  
WHERE ...
```

- General form of sampling clause

`TABLESAMPLE <sampling_method>(<p>) REPEATABLE(<n>)`

- `<sampling_method>` is
  - **BERNOULLI**: row-level Bernoulli sampling
  - **SYSTEM**: vendor-defined (efficient) sampling method
- `<p>` = inclusion probability for each row (%)  
= (expected) sampling fraction

# ISO Sampling Standard (Cont'd)

- **REPEATABLE(<n>)** clause
  - Same result at each query execution provided <n> is fixed
  - Example (row-level sampling):
    - **REPEATABLE(3)**: r56, r107, r3, ... every time
    - **REPEATABLE(48)**: r2002, r5, r86, ... every time
  - Used for debugging and testing
  - Need only be provided on "best effort" basis
    - Parallelism is an issue

# ISO Sampling Standard (Cont'd)

- Argument to <sampling method> can be an **expression**
  - Ex: I want about 1000 rows from t1 but don't know size of t1

```
SELECT * FROM t1
  TABLESAMPLE BERNOULLI(100e0*(1000 / (SELECT COUNT(*) FROM
t1)))
```

- Argument to REPEATABLE also can be an expression

# ISO Sampling Semantics

- Sample each table, then execute rest of query
- Ex: Select employees with salary > (estimated dept. average)

```
SELECT name, salary FROM emp e
WHERE salary > (SELECT AVG(salary)
                FROM emp e1 tablesample bernoulli(1.0)
                WHERE e1.deptno = e.deptno
                )
```

- Estimated salary is **fixed** for each department
  - Within a single query execution
  - Because e1 is sampled only once
  - Values vary for different executions unless REPEATABLE

# Some Unresolved Standards Issues

- Scaleup issues

- Ex: query to estimate total sales in \$

```
SELECT SUM(ti.qty * ti.price) / .01 AS amount  
FROM stars.transitem1 as ti TABLESAMPLE BERNOULLI(1)
```

- Values of SUM, COUNT, etc. are for rows retrieved
- Need scale-up functions to extrapolate these to entire database
- Can be nontrivial: COUNT(DISTINCT)
- No consensus on syntax for expressing scaleup

- Error estimates

- User must provide own formulas or UDFs for std error etc.
- Separate formula/UDF needed for each aggregate in SELECT
- No consensus yet on syntax (a hard language-design problem)

# Road Map

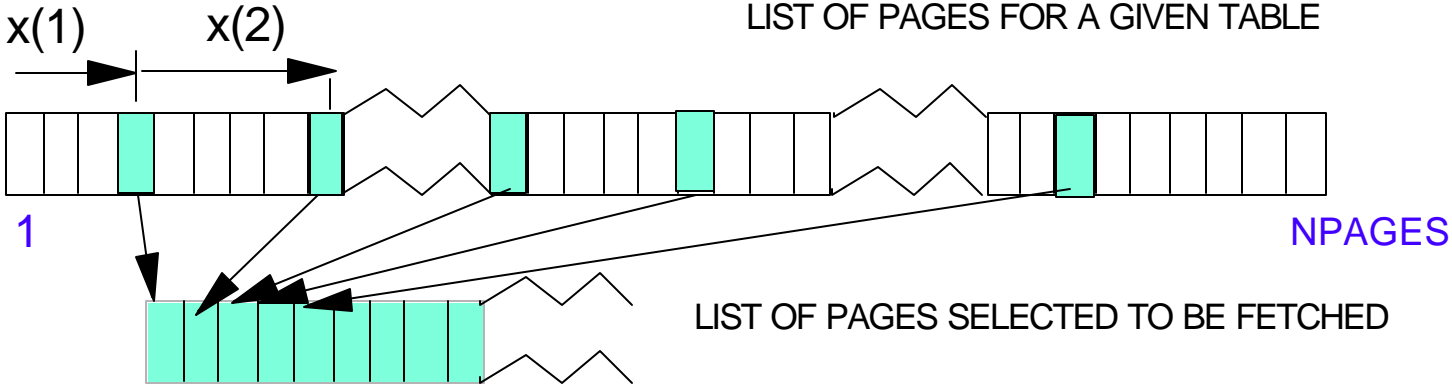
- Basic database sampling schemes
- Sampling in DB2 V7.1 and earlier
- The proposed ISO sampling standard
- Sampling support in DB2 V8
- Combining sampling and SQL
- Some research directions
  - Integrating sampling and analytics
  - Interactive querying

# DB2 V8 Support of ISO Standard

- Currently scheduled for a V8 Fixpack
- BERNOULLI: Fully implemented (MPP parallizable)
- SYSTEM: Implemented as **page-level Bernoulli sampling**
- Current limitations
  - Can only sample from **materialized tables**
  - Only **simple expressions** as args to <sampling\_method>
  - Only **simple integer expressions** as args to REPEATABLE
  - SMP parallelism not exploited
  - REPEATABLE is still best effort
    - OK in non-parallel setting (If data, partitioning, etc. unchanged)

# Page-Level Sampling: What's Going On?

1. Use random-number generator to determine  $X(i)$

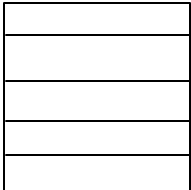


2. Map from Object-Relative to Buffer-Pool-Relative

3. Sort the Page Numbers



4. Call Prefetcher on Each Block of Pages



5. Scan ALL the Rows on Each Page Fetched

# Road Map

- Basic database sampling schemes
- Sampling in DB2 V7.1 and earlier
- The proposed ISO sampling standard
- Sampling support in DB2 V8
- Combining sampling and SQL
- Some research directions
  - Integrating sampling and analytics
  - Interactive querying

# Combining Sampling and SQL

- Greatly enhances power of sampling
- Applications
  - Quick approximate analytics
  - Quick approximate answers to aggregation queries
    - Precision measures (standard error)
  - Implementing other sampling schemes
  - Drill-down in "fuzzy" OLAP cubes

# Quick Approximate Analytics

- Ex: Linear regression

```
SELECT REGR_SLOPE(sales, ad_budget),  
       REGR_ICPT(sales, ad_budget)  
FROM ad_campaign_data TABLESAMPLE SYSTEM(10.0)
```

- For other examples see Session B15
- Of course, can also feed sample into SAS, IntelligentMiner, etc.

# Standard Error: Row-level Sampling

- Ex: Sample from single table to estimate total sales

```
SELECT sum(sales) / :samp_rate AS est_sales,  
       SQRT((1e0/:samp_rate)*((1e0/:samp_rate)-1e0)*sum(sales*sales)) AS std_err  
FROM transactions TABLESAMPLE BERNOULLI(100 * :samp_rate)
```

```
est_sales  std_err  
-----  -  
146821    1225.61
```

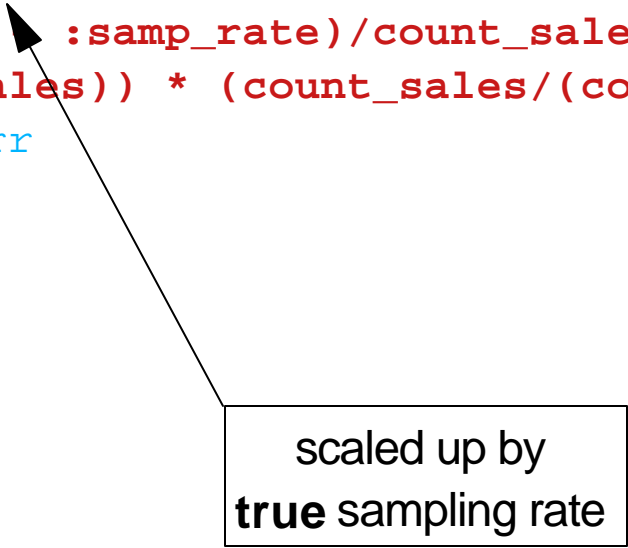
- Ex: Same as above, but for average sales (can add predicate)

```
WITH dt AS  
  (SELECT COUNT(*) / :samp_rate as est_count,  
        SUM(sales) / COUNT(*) AS est_avg,  
        (1e0/:samp_rate)*((1e0/:samp_rate)-1e0)*SUM(sales*sales) AS v_sum,  
        (1e0/:samp_rate)*((1e0/:samp_rate)-1e0)*COUNT(*) AS v_count,  
        (1e0/:samp_rate)*((1e0/:samp_rate)-1e0)*SUM(sales) AS cov_cs  
  FROM transactions TABLESAMPLE BERNOULLI(100 * :samp_rate))  
SELECT est_avg,  
       SQRT(v_sum-2e0*est_avg*cov_cs+est_avg*est_avg*v_count)/est_count  
       AS std_err  
FROM dt;
```

# A More Precise Estimator

- Row-level Bernoulli sampling as before (but scaleup is different)

```
WITH
  dt1 AS (SELECT COUNT(*) AS big_count FROM transactions),
  dt2 AS (SELECT AVG(sales) AS avg_sales,
           count(*) as count_sales,
           var(sales) AS var_sales
           FROM transactions TABLESAMPLE BERNOULLI(100 * :samp_rate))
SELECT avg_sales * big_count AS est_sales,
  big_count * sqrt( ((1e0 * :samp_rate)/count_sales)
    * (1e0 - (1e0/count_sales)) * (count_sales/(count_sales-1e0))
    * var_sales) AS std_err
FROM dt1, dt2
```



scaled up by  
**true** sampling rate

# SE Formulas for Row-Level Sampling in Complicated Queries

- Can sometimes use single table formulas

```
SELECT loc.country AS country, year(t.pdate) AS year,
       sum(ti.sales) / :samp_rate AS est_sales,
       sqrt((1e0/:samp_rate)*((1e0/:samp_rate)-1e0)*sum(sales*sales)) AS std_err
FROM trans t, transitem ti TABLESAMPLE BERNOULLI(100 * :samp_rate), loc loc
WHERE t.transid = ti.transid AND loc.locid = t.locid
GROUP BY loc.country, year(t.pdate)
```

country	year	est_sales	std_err
-----	-----	-----	-----
USA	1998	127505	1326.09
USA	1999	236744	2133.17
GERMANY	1998	86278	961.45
GERMANY	1999	126737	1488.66

...

- But be careful!! Alternative SE formulas needed for
  - Sampling from `trans` table (cluster sampling)
  - Sampling from both tables
  - etc.

# Standard Error: Page-level Sampling

- Ex: Sample from single table to estimate total sales
  - Use **SAMPLE\_UNIT\_ID( )** to get page number of sampled row
  - Argument is any column in sampled table
  - (Not in standard yet, but proposed for DB2 release)

```
WITH
  dt1 AS (SELECT sales FROM transactions TABLESAMPLE SYSTEM(100*:samp_rate))
  dt2 AS (SELECT SUM(sales) AS s_sales
          FROM dt1
          GROUP BY SAMPLE_UNIT_ID(sales)),
  dt3 AS (SELECT SUM(s_sales*s_sales) AS ss_sales,
          SUM(s_sales) AS tot_samp_sales
          FROM dt2)
SELECT tot_samp_sales / :samp_rate AS est_sales,
       SQRT((1e0/:samp_rate)*((1e0/:samp_rate)-1e0)*ss_sales) AS std_err
FROM dt3;
```

# SE for Average Sales (Page-Level Version)

```
WITH
  dt1 AS (SELECT SUM(sales) AS s_sales, COUNT(*) AS c_sales
          FROM transactions TABLESAMPLE SYSTEM(100* :samp_rate)
          GROUP BY SAMPLE_UNIT_ID(sales)),
  dt2(est_count, est_avg, v_sum, v_count, cov_cs) AS
  (SELECT SUM(c_sales) / :samp_rate,
         SUM(s_sales) / SUM(c_sales),
         (1e0/:samp_rate)*((1e0/:samp_rate)-1e0)*SUM(s_sales*s_sales),
         (1e0/:samp_rate)*((1e0/:samp_rate)-1e0)*SUM(c_sales*c_sales),
         (1e0/:samp_rate)*((1e0/:samp_rate)-1e0)*SUM(s_sales*c_sales)
   FROM dt1)
SELECT est_avg,
       SQRT(v_sum-2e0*est_avg*cov_cs+est_avg*est_avg*v_count)/est_count
       AS std_err
FROM dt2
```

# Summary

- If unclustered data: use page-level sampling
  - use row-level formulas for error estimation
  - enjoy the speedup!
- If clustered data:
  - if quick approximate answers
    - use page-level sampling with appropriate error formulas
    - may need to increase sampling rate
  - if more complicated analysis ("pure" sample needed)
    - use row-level sampling
    - try to amortize cost over multiple queries  
(at least CPU is reduced)
- So it is important to know whether data is clustered

# Detecting Clustering

- Kiefer-Komogorov-Smirnov Test (on sample of 5-10 pages)
  - Compute

$$kks = \max_{j \in \text{column}} \sqrt{\sum_i n_i (F_i(x_j) - F(x_j))^2}$$

where

- $F_i(x)$  = fraction of values on page  $i$   $\leq x$
  - $F(x)$  = fraction of values in column  $\leq x$
  - $n_i$  = number of values on page  $i$
- Basic idea
    - kks is "small" if no clustering, "large" if clustering
    - Compute  $\text{Prob}\{\text{observing value } \geq \text{kks when no clustering}\}$
    - If probability is close to 0, then  $c1$  is clustered
    - Can compute probability using a UDF

# Detecting Clustering in DB2

- Assume for simplicity that table has a key column

WITH

```
dt1 AS (SELECT t.c1, t.key as rid, SAMPLE_UNIT_ID(c1) AS pg
        FROM t TABLESAMPLE SYSTEM(1)),
dt2 AS (SELECT pg AS i, COUNT(*) AS ni FROM dt1 GROUP BY pg),
dt3 AS (SELECT COUNT(*) AS dv_cnt FROM dt2),
dt4 AS (SELECT c1, pg, rid, i FROM dt1,dt2),
dt5 AS (SELECT rid, c1, pg, i,
        (SELECT count(*) FROM dt4 AS x WHERE x.pg = dt4.i
         AND x.c1 <= dt4.c1)/dv_cnt AS Fi FROM dt4,dt3),
dt6 AS (SELECT rid,
        (SELECT count(*) FROM dt1 AS x WHERE x.c1 <= dt1.c1)
        AS F FROM dt1),
dt7 AS (SELECT COUNT(*) AS row_cnt FROM dt1),
dt8 AS (SELECT dt5.rid as rid, SUM(dt2.ni*POWER((REAL(dt5.Fi)/dt2.ni)
        - (REAL(dt6.F)/dt7.row_cnt),2e0)) AS term
        FROM dt2,dt5,dt6,dt7
        WHERE dt5.rid = dt6.rid AND dt5.i = dt2.i GROUP BY dt5.rid),
dt9 AS (SELECT MAX(term) AS maxterm FROM dt8)
SELECT SQRT(maxterm) AS kks, SMALLINT(dv_cnt) AS num_pages,
       KKSProb(SQRT(maxterm),SMALLINT(dv_cnt)) AS prob
FROM dt9,dt3
```

UDF



# Detecting Clustering: Examples

- Parameters
  - 50,000 rows with 50 rows per page
  - Real values between 0 and 5000
- **Table A:** Values randomly assigned to pages

KS	NUM_PAGES	PROB
1.3522	7	0.8180

- **Table B:** Values perfectly clustered

KS	NUM_PAGES	PROB
9.2582	7	0.0000

# Simple Random Sampling

- Row-level oversampling
  - Ex: 75 sample rows from 10,000 row table

```
WITH dt AS (SELECT * FROM mytable TABLESAMPLE BERNOULLI(1))
SELECT * FROM dt ORDER BY RAND()
FETCH FIRST 75 ROWS ONLY
```

- Page-level oversampling
  - Ex: 75 sample pages from 10,000 page table

```
WITH dt1 AS (SELECT c1,..., cN,SAMPLE_UNIT_ID(c1) AS page_num
             FROM mytable TABLESAMPLE SYSTEM(1)),
     dt2 AS (SELECT DISTINCT page_num FROM dt1),
     dt3 AS (SELECT page_num FROM dt2
             ORDER BY RAND()
             FETCH FIRST 75 ROWS ONLY)
SELECT c1,...,cN
FROM dt1,dt3
WHERE dt1.page_num = dt3.page_num
```

# Cluster Sampling

- Ex 1: Transactions for a customer
  - Select random set of customers
  - Include all transactions for each selected customer

```
WITH sampcust AS
  (SELECT cust_id
   FROM customers TABLESAMPLE SYSTEM(5))
SELECT transactions.amount,
       transactions.date
FROM transactions, sampcust
WHERE transactions.custid = sampcust.custid
```

- Ex 2: Previous example with TRANS and TRANSITEM

# Stratified Sampling

```
(SELECT name, address, salary, gender
  FROM census TABLESAMPLE BERNOULLI(10)
 WHERE gender = 'M'
 ORDER BY RAND() SELECT FIRST 75 ROWS ONLY)
UNION
(SELECT name, address, salary, gender
  FROM census TABLESAMPLE BERNOULLI(10)
 WHERE gender = 'F'
 ORDER BY RAND() SELECT FIRST 75 ROWS ONLY)
```

# Systematic Sampling

- Fix sampling parameter k
- Select row at random from first k rows
- Select every kth row thereafter

```
WITH
```

```
  dt1(startval) AS (VALUES(1 + INTEGER(rand() * :k))),  
  dt2 AS (SELECT ROWNUMBER() OVER () AS rownum, t.*  
          FROM t)
```

```
SELECT dt2.*
```

```
FROM dt1, dt2
```

```
WHERE MOD(dt2.rownum-dt1.startval, :k) = 0
```

# Split Samples

- Mark each row in a 2-column table
  - As training (1) or testing (2) or not in sample (0)
- 30% sample, with 1/3 training and 2/3 testing

```
WITH dt AS
  (SELECT c1, c2, RAND(1) AS rno FROM t)
SELECT c1, c2,
  CASE
    WHEN rno <= 0.1 THEN 1
    WHEN (rno > 0.1) AND (rno <= 0.3)
THEN 2
    ELSE 0
  END AS sampleid
FROM dt
```

# Approximate Drill-Down

```
SELECT country, state, city, year, month,  
       avg(value) as avg_sales,  
FROM trans tablesample system(:samp_rate), loc  
WHERE trans.locid = loc.locid  
GROUP BY ROLLUP(country, state, city),  
         ROLLUP(year, month)  
HAVING count(*) > 100
```

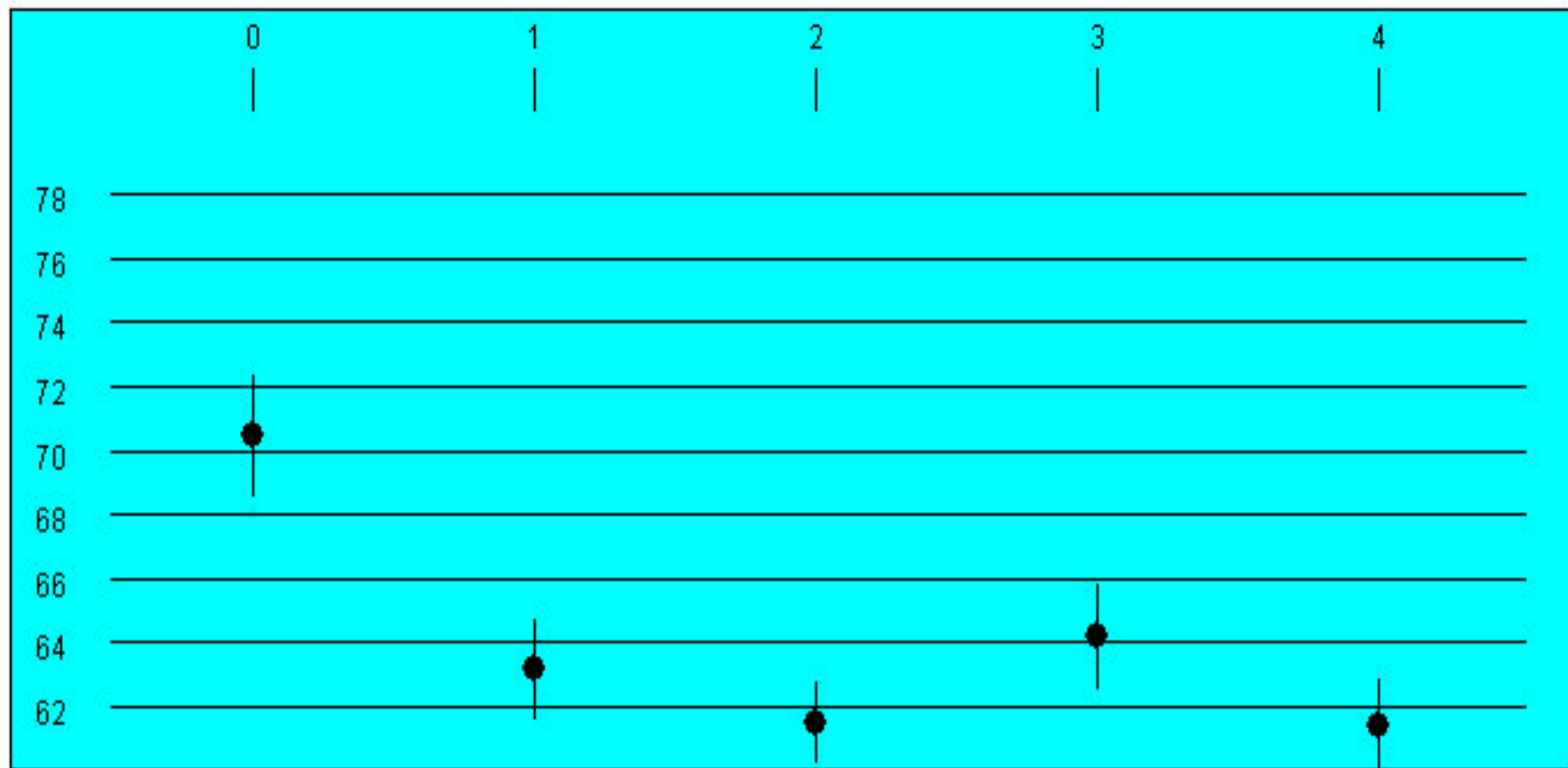
- Drill down by increasing sampling rate
  - low sampling rates => (year), (country), (year, country)
  - high sampling rates => (year, month, country, state, city) etc.
- More natural than usual (non-sampling) OLAP computation
  - Greater drill-down => more work

# Continuous Refinement Via Sampling

- A prototype exploratory interface (74 rows processed)

```
SELECT AVG(temp) FROM t GROUP BY site;
```

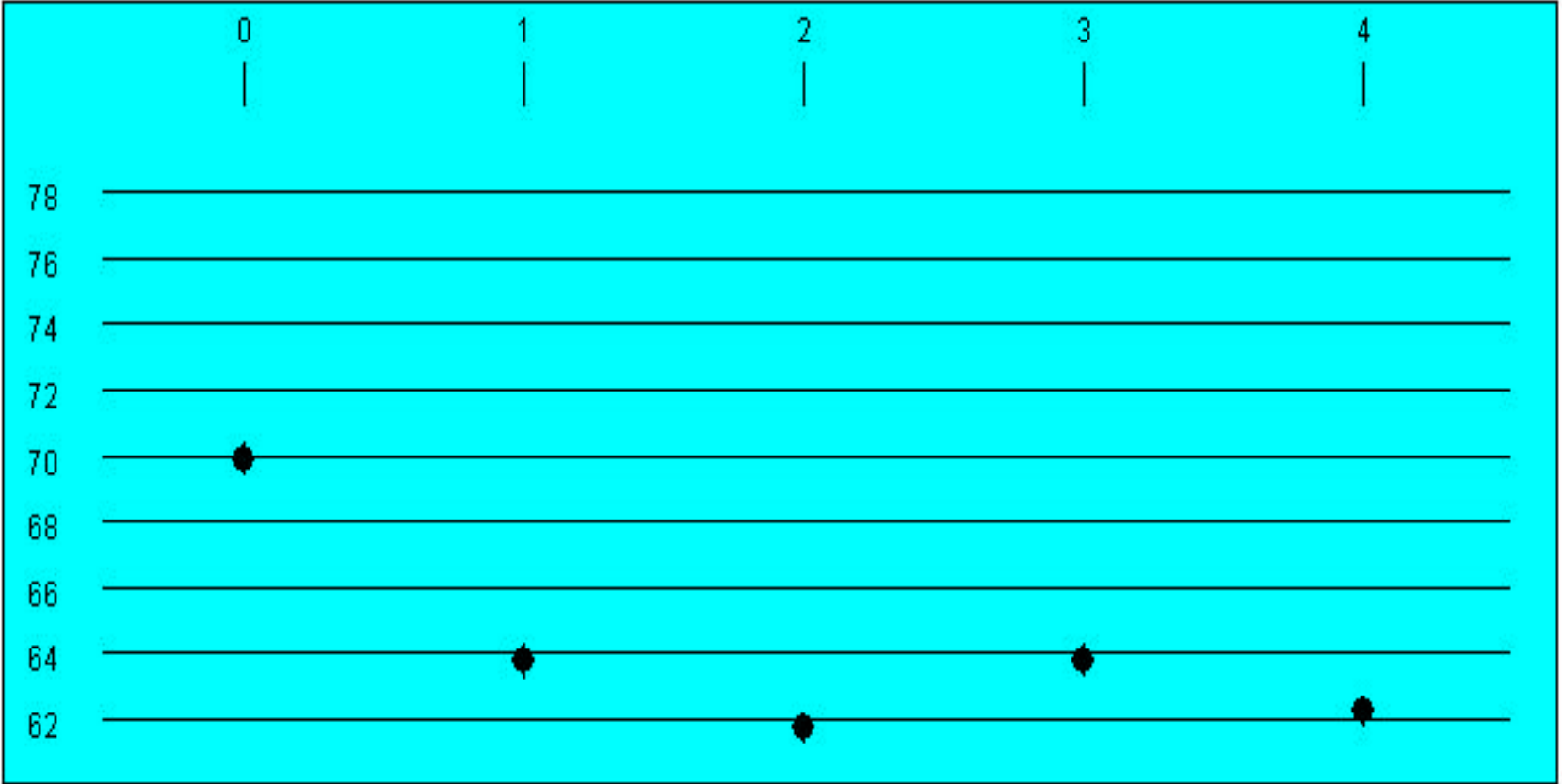
Conf. Level:  % Output Interval:     Rows read: 74 Total rows: 327296



# Continuous Refinement, Continued

- After 834 rows processed:

Conf. Level:  % Output Interval:     Rows read: 834 Total rows: 327296



# Online Technology

- Additional features of online-aggregation interface
  - slowdown/speedup/abort for individual groups
    - i.e., query modification on the fly
    - more resources left for processing other groups
- Lots of new technology
  - new "ripple join" algorithms, online reordering, etc
- Interesting applications
  - online visualization, online spreadsheets
- Collaboration with UC Berkeley (CONTROL project)
  - See references for overviews

# Selected References

## • Introductory Statistics

- Larry Gonick, et al : *The Cartoon Guide to Statistics*, HarperCollins, 1994
- Jeffrey Clark and Douglas A. Downing: *Forgotten Statistics : A Self-Teaching Refresher Course*, Barrons, 1996
- Lloyd R. Jaisingh: *Statistics for the Utterly Confused*, McGraw-Hill, 2000

## • Advanced Statistics/Sampling

- Cochran: *Sampling Techniques*, 3rd Ed., Wiley, 1977
- B. W. Lindgren: *Statistical Theory*, 3rd Ed., MacMillan, 1976
- R. G. Miller: *Beyond ANOVA, Basics of Applied Statistics*, Wiley, 1986
- C.-E. Sarndal, B. Swenson, and J. Wretman: *Model Assisted Survey Sampling*, Springer-Verlag, 1992
- M. E. Thompson, *Theory of Sample Surveys*, Chapman & Hall, 1997

## • Online Processing

- SIGMOD '01 tutorial: <http://control.cs.berkeley.edu/sigmod01>
- *IEEE Computer*, August, 1999
- *Proc. 11th Conf. Sci. and Statist. Database Management (SSDBM)*, 1999

# Many Thanks To...

- Joe Hellerstein
- Michelle Jou
- George Lapis
- Guy Lohman
- Eric Louie
- Gregor Meyer
- Carlene Nakagawa
- Hamid Pirahesh
- Ashutosh Singh
- Mike Winer
- Markos Zaharioudakis
- ...

# For More Information

- Go to: [www.almaden.ibm.com/cs/people/peterh](http://www.almaden.ibm.com/cs/people/peterh)
  - Latest version of this talk
  - Screencam demo of online aggregation interface