The Case for a Wide-Table Approach to Manage Sparse Relational Data Sets

Eric Chu

Jennifer Beckmann

Jeffrey Naughton

[CPS399.28] Presented by Vamsidhar Thummala

Slides by Eric Chu

“Sparse” Data Sets

- Large number of attributes (100’s – 1000’s)
- Large portion of null values (>50% – 99%)
- Evolving schema

The Problem with Sparse Data Sets

- What is the appropriate storage model?
- How to effectively build ad hoc queries over thousands of attributes?
- How to evaluate these queries efficiently?

Contributions of the paper

- What is the appropriate storage model?
  - Wide table approach (their previous work)
- How to effectively build ad hoc queries over thousands of attributes?
  - Keyword search
- How to evaluate these queries efficiently?
  - Sparse B-Tree indexes, Hidden Schema, View Materialization

RDBMS Support for Sparse Data

- One-table VS Multi-table
- Building queries over sparse data
- Evaluating queries over a wide table

One-table vs Multi-table

- Horizontal/Positional Storage
- Vertical/Column Storage
- Challenging schema design
  - One table
    - Null values (Horizontal)
    - Joins (Vertical)
  - Multi-table
    - Minimal null values
    - Reasonable number of tables => large joins

Horizontal

Oid	A1	A2	A3	A4
1			v4	v2
2		v1		
3	v3		v1	
4	v1	v4		
5			v5	

Vertical

Oid	Attr	Value
3	A1	v3
4	A1	v1
2	A2	v1
4	A2	v4
1	A3	v4
3	A3	v1
5	A3	v5
1	A4	v2

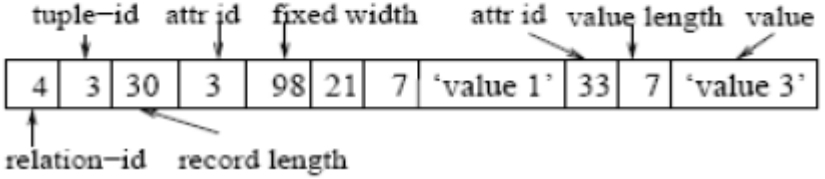
Wide table Storage

- Interpreted storage
  - **No null values**
  - **No need to join tables**

Interpreted Catalog

name	id	type	size
A1	3	INT	4
A2	21	VARCHAR(16)	16
A3	45	VARCHAR(16)	16
A4	33	VARCHAR(16)	16

Interpreted Record



RDBMS Support for Sparse Data

- One-table VS Multi-table
- Building queries over sparse data
- Evaluating queries over a wide table

Querying Sparse Data

```
SELECT    *
FROM      WideTable
WHERE     ? = '%apple%'
```

So many attributes...

supported cd audio, amplifier response bandwidth, built-in decoders, equalizer bands, thx certified, furniture features, media capacity, capacity, cables type, connections qty, built-in devices, width, diagonal size (inches), display type, multi-language select, multi-subtitle select, additional features, body material, combined with, compatible game consoles, device type:type, display screen size compatibility, image aspect ratio, image stabilizer, media format, networking type, package type, product type:additional handsets qty, shielding material, eight (shipping):shipping weight, wireless interface, sensitivity, pressure levels, still image format, archival life, dialed calls memory, received calls memory, 3g services / included services, mobile email, supported sms functions, consumables included, included accessories:included video adapter, accessories, modem connector qty, interface gender, interface provided, miscellaneous compliant standards, slot(s) provided type, technology / form factor:type, tv tuner channel coverage, instruction set, ff/rew speeds, on-screen program guide, ram installed, license validation period, min supported color depth, other compatible software, cd / dvd write speed, type, modem / comm., electronic program guide, tuner type (qty), favorite channel list, video signal-to-noise ratio, analog video signal, video output interface type, field coverage, response / service time.....

Keyword Search

```
SELECT   *
FROM     WideTable
WHERE    ? = '%apple%'
```

"apple"

- Find all rows that contain the keyword "apple"

Potential Problem of Keyword Search

SQL: SELECT *
FROM WideTable
WHERE **Brand** = '%apple%'

Keyword: "apple"

Oid	Brand	Drink	Dessert	Fruit
1	Apple			
73		apple juice		apple
201			apple strudel	apple

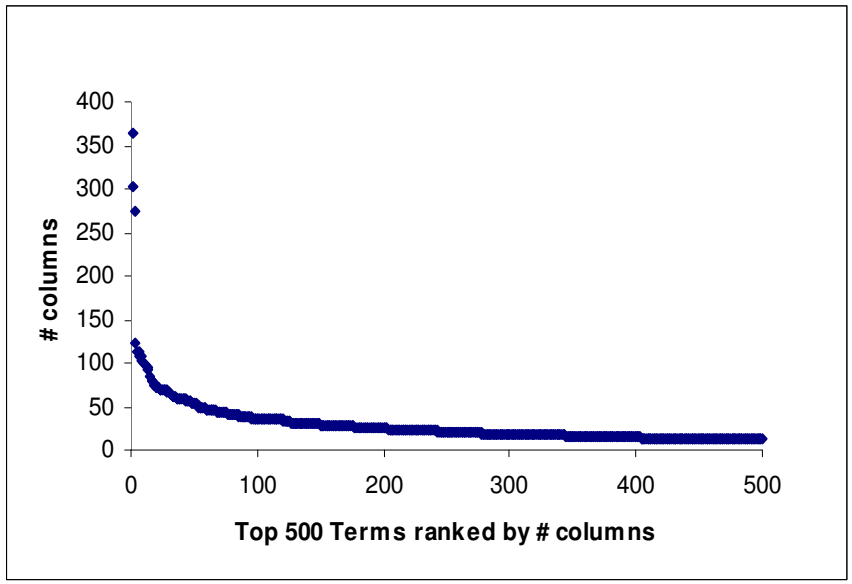
Is this potential problem real?

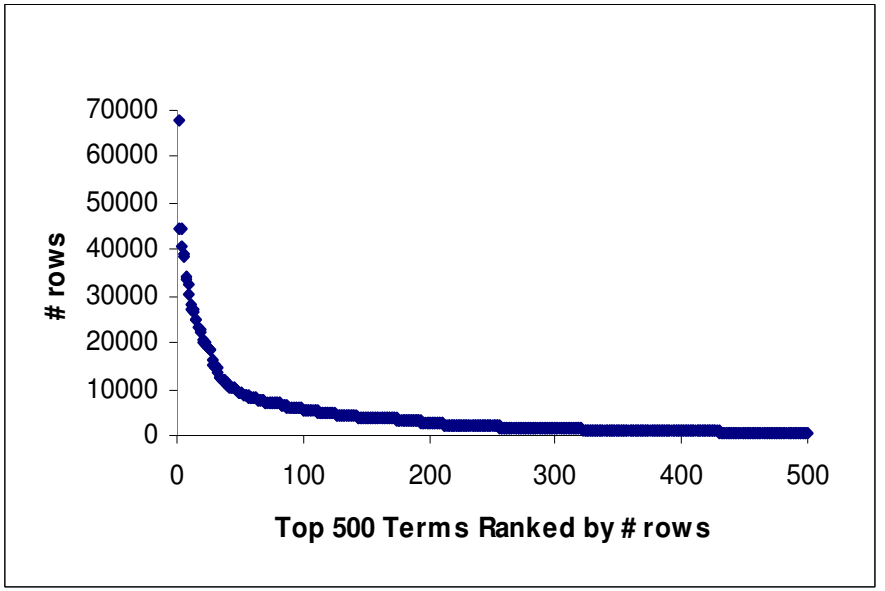
- To gain insight: examine real-world sparse data sets
- CNET: 2,984 columns, 233,304 rows, 11 non-null values per row
- Tokenize terms in sparse columns
  - 3 terms: apple, juice, strudel

Oid	Brand	Beverage	Dessert	Fruit
1	Apple			
73		apple juice		apple
201			apple strudel	apple

CNET Term Distribution (1)

- Number of rows & columns that contain the term





- Zipf's distribution

CNET Term Distribution (2)

# Columns \ # Rows	1 only	2-25	26-50	51-150	>150
>15	0%	0%	0%	0%	5%
11-15	0%	0%	0%	1%	2%
6-10	0%	1%	1%	2%	3%
2-5	1%	18%	4%	4%	2%
1 only	21%	31%	2%	1%	1%

- Keyword query with 1 term
- Result of keyword search surprisingly focused
  - 71% of terms appear in <26 rows and <6 columns.
- Even more focused w/ multiple keywords

Keyword Search over Sparse Data

- In general, keyword search is effective IF terms follow a Zipf-like distribution
  - No need to specify attributes
- Many data sets follow Zipf-like term distributions

What if you need attribute names?

```
SELECT  * FROM WideTable
WHERE     "laptop price" < 1200
          AND "screen size" > 14
```

- Idea: fuzzy attributes

  "laptop price": price, cost, laptop_price, ...

  "screen size": ScreenSize, dimension, ...

Queries with "Fuzzy" Attribute Names

- Use name-based schema-matching techniques to find matching attributes
- Multiple matches for a fuzzy attribute
  - Most likely match – may miss right tuples
  - Multiple matches – low precision again

RDBMS Support for Sparse Data

- One-table VS Multi-table
- Building queries over sparse data
- Evaluating queries over a wide table

Motivation for B-tree Indexes

```
SELECT  * FROM   WideTable
WHERE     price < 1200 AND screen_size > 14
```

- Can't use inverted index
  - B-tree indexes
  - But we have thousands of attributes
  - Folk wisdom: building and maintaining thousands of B-tree indexes on a dense table considered infeasible

Solution: Sparse B-tree Indexes

- Map non-null values to oids
- Similar to *partial indexes*

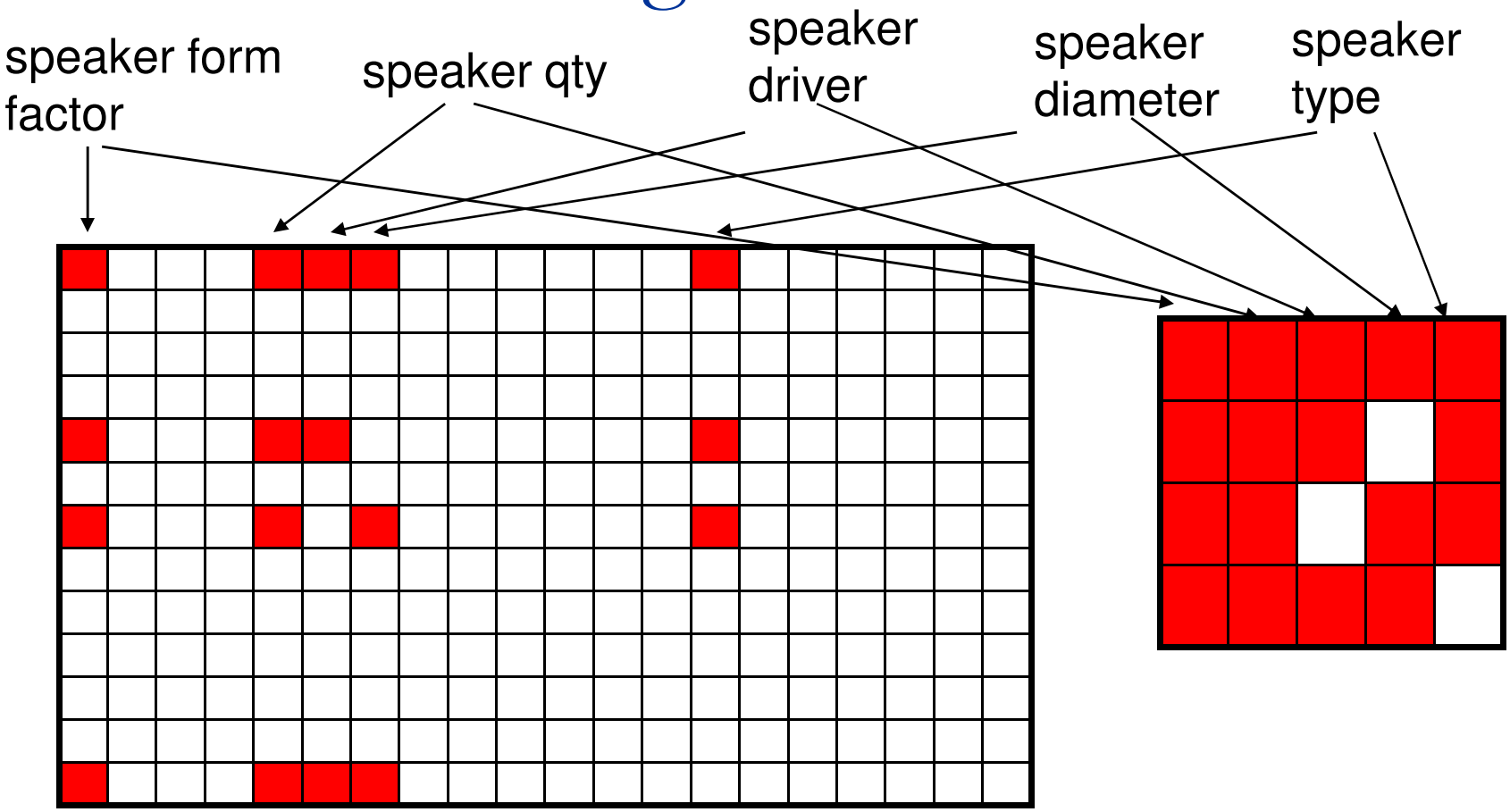
```
CREATE INDEX sparseInd ON table(a1)
WHERE a1 is not NULL
```

- More suitable for sparse data than partial indexes
  - More efficient index maintenance
  - Lower overhead for lookups

Advantages over Full Indexs

- Experiment
  - 250k rows, 7 non-null values per row out of 640 varchar(16) attributes
- 1-column sparse index 50 times smaller than full counterpart
  - 640 sparse indexes take less space than 13 full indexes
- Tuple insertion/deletion
  - 7 updates for sparse VS 640 for full
- More efficient bulkloading

Data Partitioning



- Useful for creating materialized projection views and covering indexes

Hidden Schema

- Our approach
  - Group together attributes that are either both non-null or both null in a row
- Infer hidden schema automatically
  - Jaccard(A_X, A_Y) = $|X \cap Y| / ||X \cup Y|$

    X = set of rows with non-null values in attribute A_X
  - Use k-mean clustering
- No constraints on # partitions or # joins to get each object

CNET

Row Count	Average Jaccard	Attributes in Cluster
1423	0.944	printer output type, printer type, media feeder(s), media type, printer output...
346	0.932	audio output type, input device type, projector image brightness
3116	0.949	configuration device type, device type, hard drive size, storage controller type...
442	0.984	camera flash type, connections type, lens systems type, still simage format...
125	0.86	speaker form factor, speaker qty, speaker driver diameter, speaker type...

- Top five attribute groupings from k-mean clustering
- **233,304** rows total

Browsing Directory

- When partitions make semantic sense
  - Can build a browsing directory based on hidden schema
- In addition to keyword search and SQL queries with fuzzy attributes

Storage and Maintenance of Materialized Projection Views

- Costs could be high when data is dense
  - But surprisingly low when data is sparse
- Extra Storage – about the same as wide table using interpreted storage
- Maintenance – 2 updates for tuples belonging to one partition
  - Base table and the view

Conclusion

- "Single table" actually good approach for sparse data
  - Interpreted storage for space efficiency (previous)
  - Sparse index for scalable indexability
  - Automatically discovered hidden schema for defining views and covering indexes
- Querying remains a challenge
  - Combination of keyword search, SQL with "fuzzy" attributes, and directory based on hidden schema

Discussion

- Defining Good Partitions is still challenging
  - The metrics (Jaccard coefficient, NullRatio) need more justification
  - Clustering algorithm has no constraint on number of partitions
- Algebric approaches to store the data?
- Road to future work
  - Keyword search and hidden schema's on SQL queries