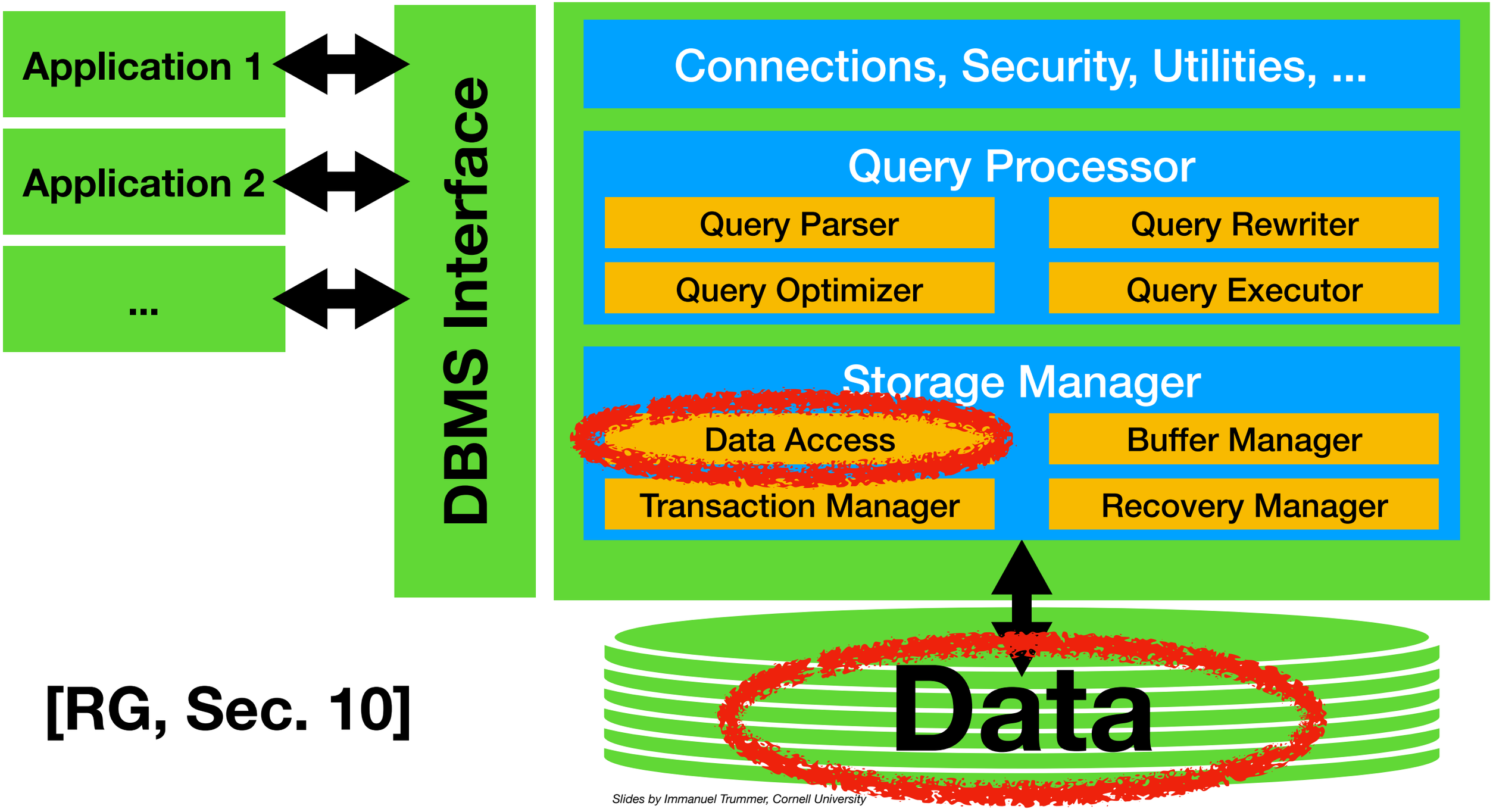
Hash Indexes

Immanuel Trummer
itrummer@cornell.edu
www.itrummer.org

Database Management Systems (DBMS)



[RG, Sec. 10]

Slides by Immanuel Trummer, Cornell University

Hash vs. Tree Indexes

- **Tree index**: traverse search tree to find interesting leafs

- **Hash index**: evaluate hash function to find buckets

Supported Predicates: Tree Indexes

- Tree indexes are based on a **sort order** between keys
  - Can handle **equality and inequality** conditions
    - **Consecutive** keys stored close together
  - Composite keys: useful for conditions on key **prefix**
    - Keys with **same prefix** value stored close together

Supported Predicates: Hash Indexes

- Hash indexes are based on key **hash values**
  - Only useful for **equality** conditions
    - **Consecutive** keys may be stored far apart
      - Similar hash value ⇏ similar key value
- Condition must constrain **all** components
  - Keys with **same prefix** may be stored far apart

Hash Index Variants

- **Static** hashing
  - Bad for dynamic data
- **Extendible** hashing
  - Expands with few high-overhead operations
- **Linear** hashing
  - Expands more "smoothly"

Static Hashing

- Hash **bucket** pages contain references to data
  - Alternatively, may contain **data** directly
- Hash buckets are associated with **hash value ranges**
- Can use hash index to find **entries with key V**
  - Calculate **hash value h** for V as h(V)
  - Look up **bucket page** associated with h

Static Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	1
Bob	0
Chan	2
Dora	5
David	1
Ester	7
Felix	4
Gert	2
Holy	7
Ida	1
Jana	0
Kyle	6
Lana	6
Levi	5
Olivia	3
Philip	7
Rosa	3
Tia	6
Victor	5
Zemin	4

PageID = Hash % NrBuckets

P0	
Bob	P23,1
Jana	P42,1
Zemin	P56,3

P1	
Alan	P24,2
David	P36,1
Ida	P62,3

P2	
Chan	P21,3
Gert	P91,1
Lana	P74,2

P3	
Olivia	P47,2
Rosa	P62,1

Updates and Static Hashing

- **Deletions** are easy - just remove associated entries
- **Insertions** are more difficult - what if bucket is full?
  - Can add "**overflow**" pages (like ISAM index!)
  - **Initial bucket** page stores pointer to first overflow page
  - Overflow pages form **linked list** if more than one
- Can **rehash** if number of overflow pages increases

Overflow Pages Example

Hash Function (Not Stored)

Key	Hash
Alan	1
Bob	0
Chan	2
Dora	5
David	1
Ester	7
Felix	4
Gert	2
Holy	7
Ida	1
Jana	0
Kyle	6
Lana	6
Levi	5
Olivia	3
Philip	7
Rosa	3
Tia	6
Victor	5
Zemin	4

PageID = Hash % NrBuckets

P0	
Bob	P23,1
Jana	P42,1
Zemin	P56,3

P1	
Alan	P24,2
David	P36,1
Ida	P62,3

P2	
Chan	P21,3
Gert	P91,1
Lana	P74,2

P3	
Olivia	P47,2
Rosa	P62,1

P101	
Felix	P76,1

P105	
Levi	P54,2
Victor	P38,2

P107	
Kyle	P44,1
Tia	P35,2

Overflow Pages

Static Hashing: Pros/Cons

- Can get data with **one read**
- May need multiple reads in case of **overflow** pages
- Will waste space if too many **deletions** (empty pages)
- Can use **rehashing** but creates significant overheads

Extendible Hashing

- Idea: use **directory** to map hash buckets to pages
  - More **flexible** than using page IDs directly
- Redistribute **overflowing** buckets to multiple pages
  - More **efficient** than having to rehash all data
- Need to **increase directory size** if too many splits

Extendible Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	1
Bob	0
Chan	2
Dora	5
David	1
Ester	7
Felix	4
Gert	2
Holy	7
Ida	1
Jana	0
Kyle	6
Lana	6
Levi	5
Olivia	3
Philip	7
Rosa	3
Tia	6
Victor	5
Zemin	4

Directory Pages

P116	
***	P131

Bucket Pages

P131	
Alan	P41,2

Insert Student "Alan"

Extendible Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	001
Bob	000
Chan	010
Dora	101
David	001
Ester	111
Felix	100
Gert	010
Holy	111
Ida	001
Jana	000
Kyle	110
Lana	110
Levi	101
Olivia	011
Philip	111
Rosa	011
Tia	110
Victor	101
Zemin	100

Directory Pages

P116	
***	P131

Bucket Pages

P131	
Alan	P41,2

Insert Student "Alan"

Slides by Immanuel Trummer, Cornell University

Extendible Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	001
Bob	000
Chan	010
Dora	101
David	001
Ester	111
Felix	100
Gert	010
Holy	111
Ida	001
Jana	000
Kyle	110
Lana	110
Levi	101
Olivia	011
Philip	111
Rosa	011
Tia	110
Victor	101
Zemin	100

Directory Pages

P116	
***	P131

Bucket Pages

P131	
Alan	P41,2
Bob	P24,1

Insert Student "Bob"

Extendible Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	001
Bob	000
Chan	010
Dora	101
David	001
Ester	111
Felix	100
Gert	010
Holy	111
Ida	001
Jana	000
Kyle	110
Lana	110
Levi	101
Olivia	011
Philip	111
Rosa	011
Tia	110
Victor	101
Zemin	100

Directory Pages

P116	
***	P131

Bucket Pages

P131	
Alan	P41,2
Bob	P24,1
Chan	P43,3

Insert Student "Chan"

Extendible Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	001
Bob	000
Chan	010
Dora	101
David	001
Ester	111
Felix	100
Gert	010
Holy	111
Ida	001
Jana	000
Kyle	110
Lana	110
Levi	101
Olivia	011
Philip	111
Rosa	011
Tia	110
Victor	101
Zemin	100

Directory Pages

P116	
**0	P131
**1	P133

Bucket Pages

P131	
Bob	P24,1
Chan	P43,3

P133	
Alan	P41,2
Dora	P49,3

Insert Student "Dora"

Extendible Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	001
Bob	000
Chan	010
Dora	101
David	001
Ester	111
Felix	100
Gert	010
Holy	111
Ida	001
Jana	000
Kyle	110
Lana	110
Levi	101
Olivia	011
Philip	111
Rosa	011
Tia	110
Victor	101
Zemin	100

Directory Pages

P116	
**0	P131
**1	P133

Bucket Pages

P131	
Bob	P24,1
Chan	P43,3

P133	
Alan	P41,2
Dora	P49,3
David	P44,2

Insert Student "David"

Extendible Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	001
Bob	000
Chan	010
Dora	101
David	001
Ester	111
Felix	100
Gert	010
Holy	111
Ida	001
Jana	000
Kyle	110
Lana	110
Levi	101
Olivia	011
Philip	111
Rosa	011
Tia	110
Victor	101
Zemin	100

Directory Pages

P116	
*00	P131
*01	P133

P117	
*10	P131
*11	P139

Bucket Pages

P131	
Bob	P24,1
Chan	P43,3

P133	
Alan	P41,2
Dora	P49,3
David	P44,2

P139	
Ester	P52,2

Insert Student "Ester"

Extendible Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	001
Bob	000
Chan	010
Dora	101
David	001
Ester	111
Felix	100
Gert	010
Holy	111
Ida	001
Jana	000
Kyle	110
Lana	110
Levi	101
Olivia	011
Philip	111
Rosa	011
Tia	110
Victor	101
Zemin	100

Directory Pages

P116	
*00	P131
*01	P133

P117	
*10	P131
*11	P139

Bucket Pages

P131	
Bob	P24,1
Chan	P43,3
Felix	P68,1

P133	
Alan	P41,2
Dora	P49,3
David	P44,2

P139	
Ester	P52,2

Insert Student "Felix"

Extendible Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	001
Bob	000
Chan	010
Dora	101
David	001
Ester	111
Felix	100
Gert	010
Holy	111
Ida	001
Jana	000
Kyle	110
Lana	110
Levi	101
Olivia	011
Philip	111
Rosa	011
Tia	110
Victor	101
Zemin	100

Directory Pages

P116	
*00	P131
*01	P133

P117	
*10	P126
*11	P139

Bucket Pages

P131	
Bob	P24,1
Felix	P68,1

P133	
Alan	P41,2
Dora	P49,3
David	P44,2

P139	
Ester	P52,2

P126	
Gert	P33,1
Chan	P43,3

Insert Student "Gert"

Extendible Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	001
Bob	000
Chan	010
Dora	101
David	001
Ester	111
Felix	100
Gert	010
Holy	111
Ida	001
Jana	000
Kyle	110
Lana	110
Levi	101
Olivia	011
Philip	111
Rosa	011
Tia	110
Victor	101
Zemin	100

Directory Pages

P116	
*00	P131
*01	P133

P117	
*10	P126
*11	P139

Bucket Pages

P131	
Bob	P24,1
Felix	P68,1

P133	
Alan	P41,2
Dora	P49,3
David	P44,2

P139	
Ester	P52,2
Holy	P71,1

P126	
Gert	P33,1
Chan	P43,3

Insert Student "Holy"

Slides by Immanuel Trummer, Cornell University

Extendible Hashing Example

Hash Function (Not Stored)

Key	Hash
Alan	001
Bob	000
Chan	010
Dora	101
David	001
Ester	111
Felix	100
Gert	010
Holy	111
Ida	001
Jana	000
Kyle	110
Lana	110
Levi	101
Olivia	011
Philip	111
Rosa	011
Tia	110
Victor	101
Zemin	100

Directory Pages

P116	
000	P131
001	P133

P117	
010	P126
011	P139

P118	
100	P131
101	P128

P119	
110	P126
111	P139

Bucket Pages

P131	
Bob	P24,1
Felix	P68,1

P133	
Alan	P41,2
David	P44,2
Ida	P42,2

P139	
Ester	P52,2
Holy	P71,1

P126	
Gert	P33,1
Chan	P43,3

P128	
Dora	P49,3

Insert Student "Ida"

Insertions Summary

- Calculate **hash value** for key of new entry
- Consult **directory** to identify current bucket
- Current bucket has **space**? Simply insert.
- Current bucket is **overflowing**?
  - Add new bucket page, **rehash** existing and new entry
    - For rehashing: consider **one more bit** of hash value
    - **Expand directory** if it does not consider enough bits

Terminology

- **Global depth**: how many hash bits directory considers

- **Local depth**: how many hash bits for specific bucket

Local vs. Global Depth

Hash Function (Not Stored)

Key	Hash
Alan	001
Bob	000
Chan	010
Dora	101
David	001
Ester	111
Felix	100
Gert	010
Holy	111
Ida	001
Jana	000
Kyle	110
Lana	110
Levi	101
Olivia	011
Philip	111
Rosa	011
Tia	110
Victor	101
Zemin	100

Directory Pages **Global Depth: 3**

P116	
000	P131
001	P133

P117	
010	P126
011	P139

P118	
100	P131
101	P128

P119	
110	P126
111	P139

Bucket Pages **Local Depth**

P131	
Bob	24,1
Felix	68,1

2

P133	
Alan	41,2
David	44,2
Ida	P42,2

3

P139	
Ester	52,2
Holy	71,1

2

P126	
Gert	33,1
Chan	P43,3

2

P128	
Dora	49,3

3

Deletions

- Can **merge bucket** pages if they become empty
- Can **half directory** size if number of buckets shrinks
- Often **no compaction** in practice
  - Assumption: **inserts** are more common than deletes

Extendible Hashing: Pros/Cons

- Avoids **overflow pages**
- No need for **expensive rehashing**
  - Only rehash **one bucket** at a time
- Need additional **directory access**
- Need to **double directory** occasionally
  - This may take up some **time**

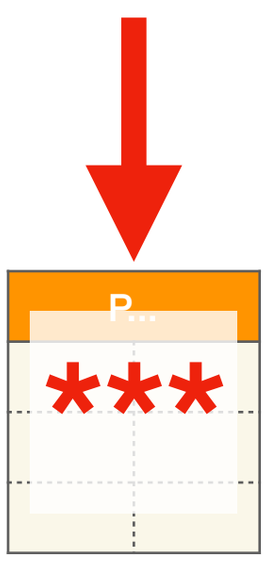
Can We Always Avoid Overflow Pages?

Linear Hashing

- Idea: avoid directory by **fixing next bucket** to split
  - Means we do not always split **overflowing** bucket!
- I.e., we may have **temporary** overflow pages
- Buckets to split are selected in **round robin** fashion
- Means overflowing bucket will be split **eventually**

Linear Hashing Example

Split Next

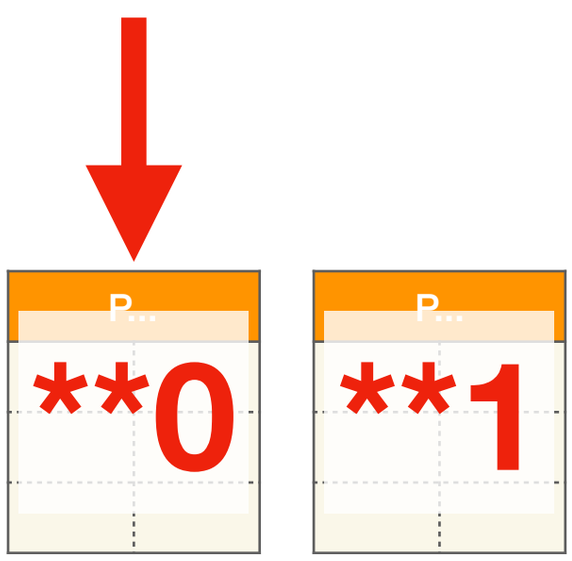


Round 1

Slides by Immanuel Trummer, Cornell University

Linear Hashing Example

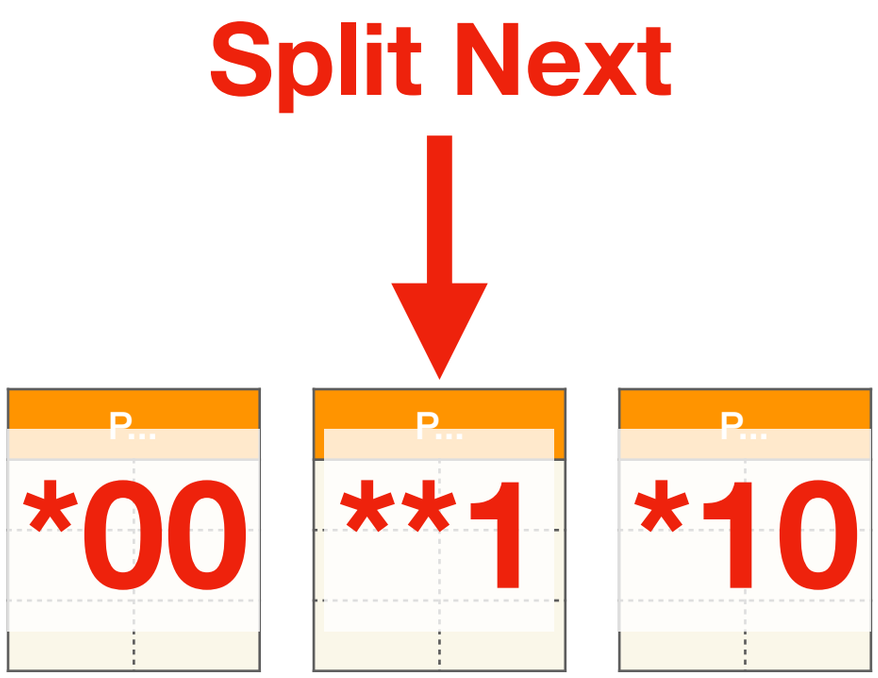
Split Next



Round 2

Slides by Immanuel Trummer, Cornell University

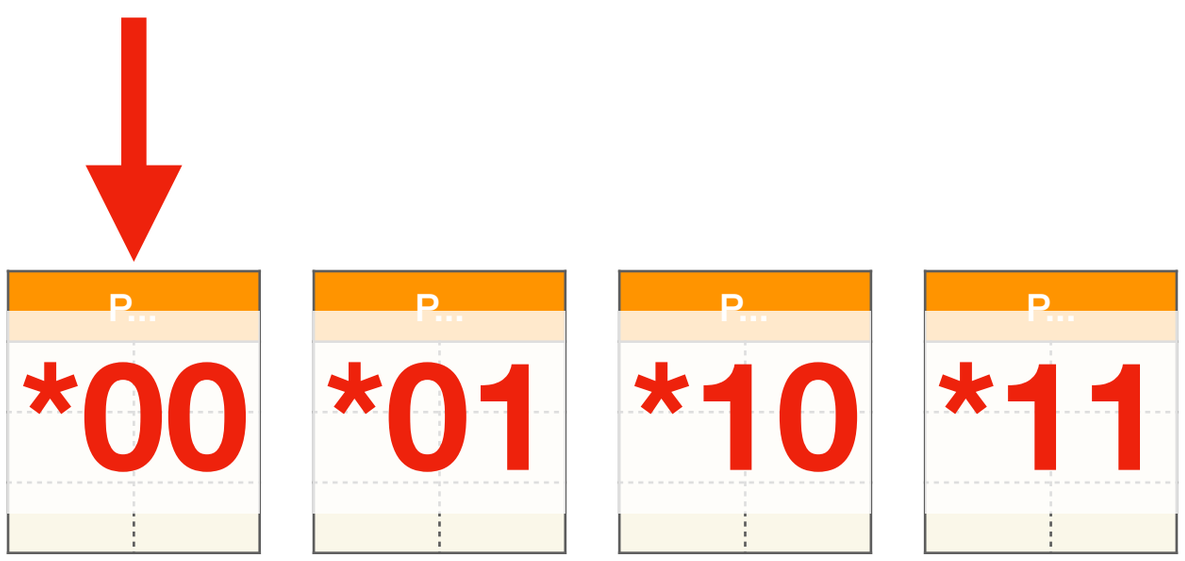
Linear Hashing Example



Round 2

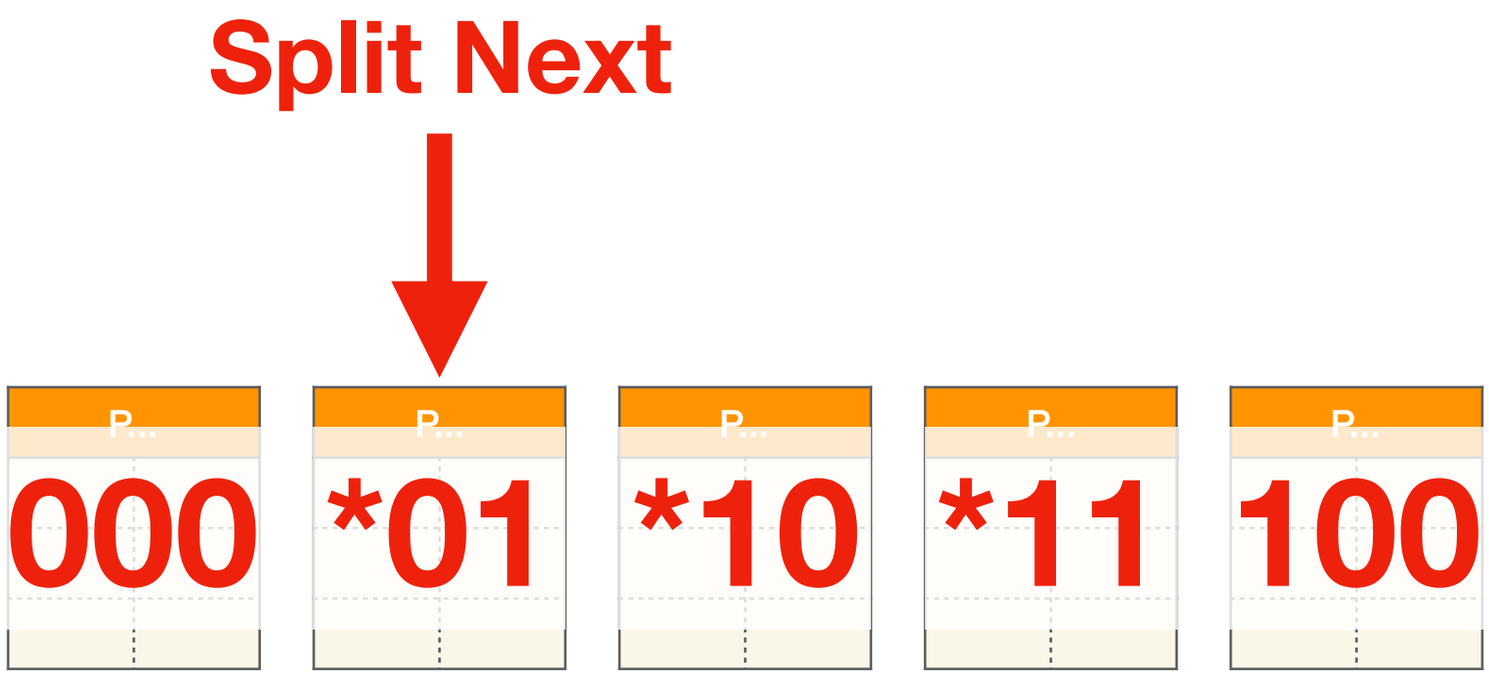
Linear Hashing Example

Split Next



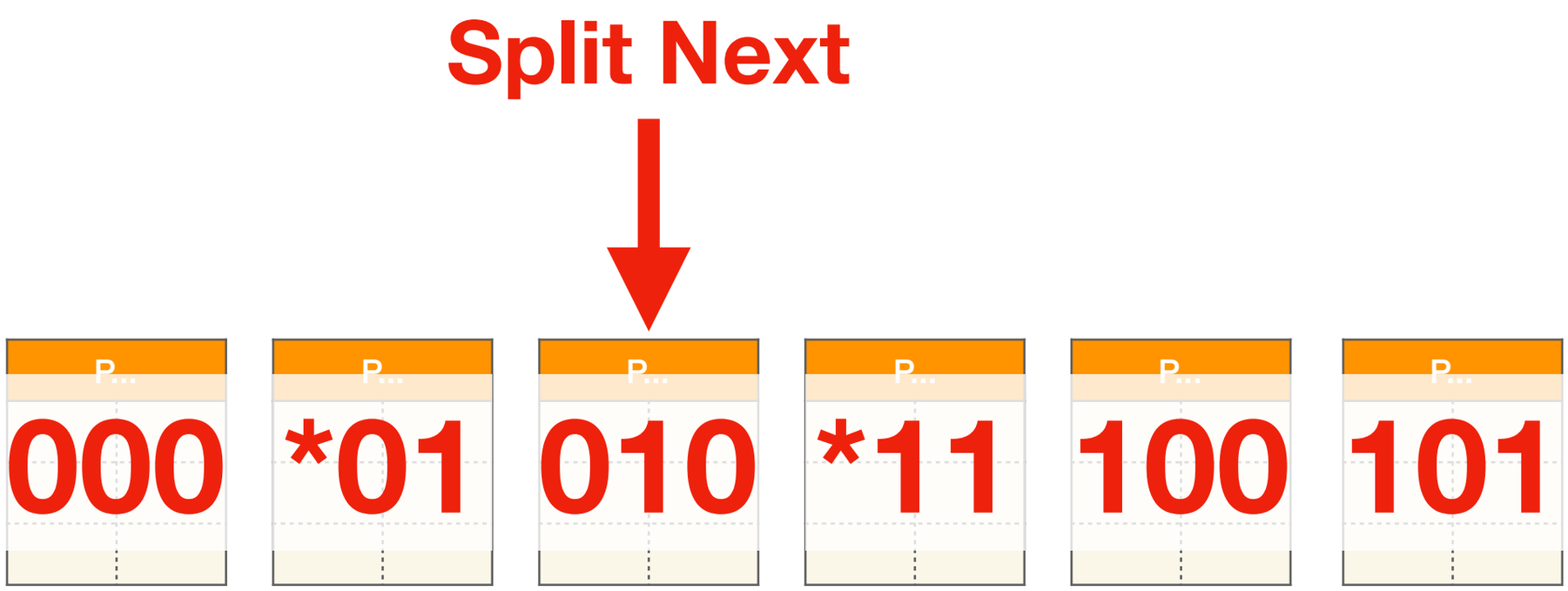
Round 3

Linear Hashing Example



Round 3

Linear Hashing Example



Round 3

Insertions Summary

- Calculate **hash value** for new entry to insert
- **Add entry** on page or - if necessary - on overflow page
- **Split next bucket** if trigger condition is satisfied
  - May **eliminate** previously generated overflow pages
  - Some flexibility in choice of **trigger condition**

Splitting Summary

- Splitting proceeds in **rounds**
  - All buckets present at round start split → round **ends**
    - "Next Split" pointer is **reset** to first page at round end
- We always split the bucket pointed to by **"Next Split"**
  - Add one new page, **redistribute** split bucket entries
    - Consider **one more bit** when redistributing

Linear Hashing: Pros/Cons

- **Avoids a directory** - no expensive directory **doubling**
- May temporarily admit **overflow** pages
- May split empty pages - inefficient **space** utilization

Optimizations

- Can apply **same optimizations** as for tree indexes
- Have **many entries** for same search key value?
  - Store key value, followed by **list** of references
- Want to get rid of one level of **indirection**?
  - Can **store data** directly instead of references
    - Leads to "**clustered index**", only one per table!
    - "Clustered index" in general: **data sorted** by index key

Choose Index Type in Postgres

- CREATE INDEX **<index-name>** ON **<table>** USING **<method>** (**<column-list>**)
- Can choose **btree** or **hash** for method
  - Other choices as well, not covered here