Lecture 21: Hash Tables and Query Execution

Friday, March 3, 2006

1

Outline

- Hash-tables (13.4)
- Query execution: 15.1 – 15.5

2

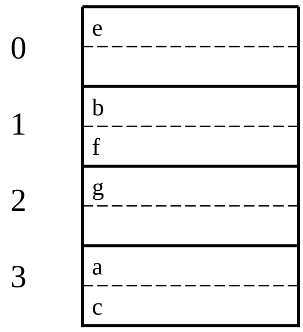
Hash Tables

- Secondary storage hash tables are much like main memory ones
- Recall basics:
  - There are n <u>*buckets*</u>
  - A hash function f(k) maps a key k to {0, 1, …, n-1}
  - Store in bucket f(k) a pointer to record with key k
- Secondary storage: bucket = block, use overflow blocks when needed

3

Hash Table Example

- Assume 1 bucket (block) stores 2 keys + pointers
- h(e)=0
- h(b)=h(f)=1
- h(g)=2
- h(a)=h(c)=3

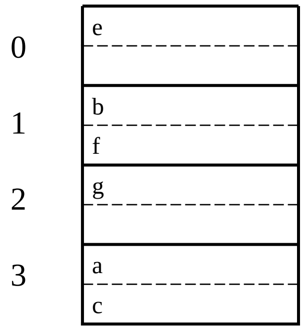


Here: h(x) = x mod 4

4

Searching in a Hash Table

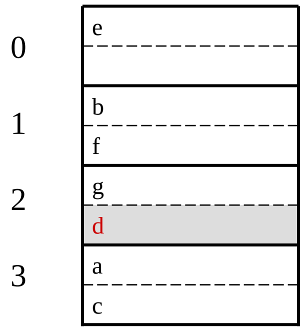
- Search for a:
- Compute h(a)=3
- Read bucket 3
- 1 disk access



5

Insertion in Hash Table

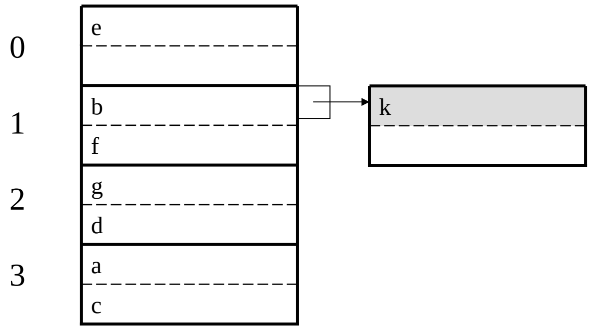
- Place in right bucket, if space
- E.g. h(d)=2



6

Insertion in Hash Table

- Create overflow block, if no space
- E.g. h(k)=1



- More over-flow blocks may be needed

7

Hash Table Performance

- Excellent, if no overflow blocks
- Degrades considerably when number of keys exceeds the number of buckets (I.e. many overflow blocks).

8

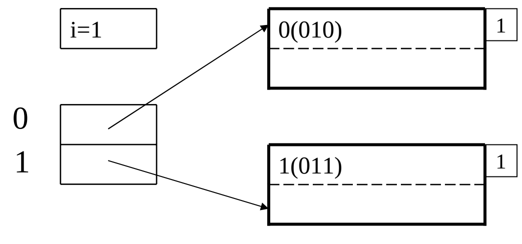
Extensible Hash Table

- Allows has table to grow, to avoid performance degradation
- Assume a hash function h that returns numbers in $\{0, \ldots, 2^k - 1\}$
- Start with $n = 2^i << 2^k$, only look at first i most significant bits

9

Extensible Hash Table

- E.g. $i=1$, $n=2^i=2$, $k=4$

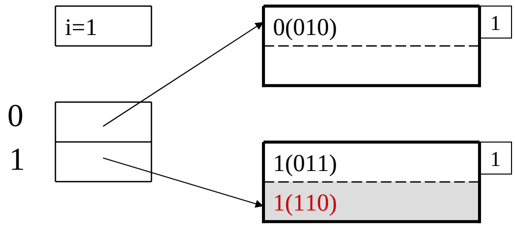


- Note: we only look at the first bit (0 or 1)

10

Insertion in Extensible Hash Table

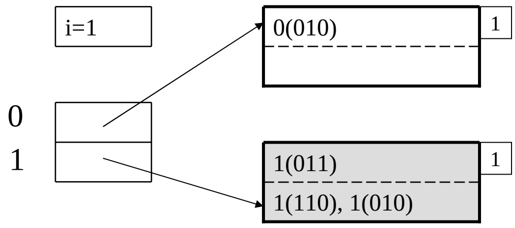
- Insert 1110



11

Insertion in Extensible Hash Table

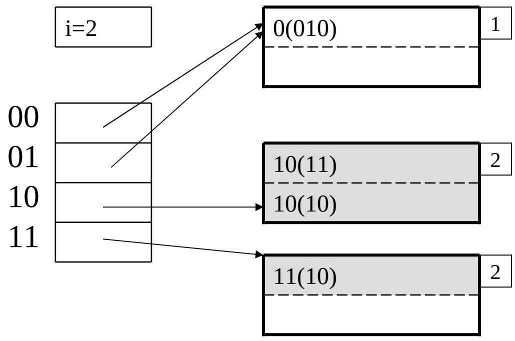
- Now insert 1010



- Need to extend table, split blocks
- i becomes 2

12

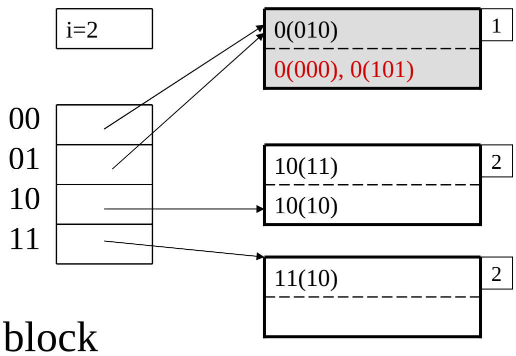
Insertion in Extensible Hash Table



13

Insertion in Extensible Hash Table

- Now insert 0000, then 0101

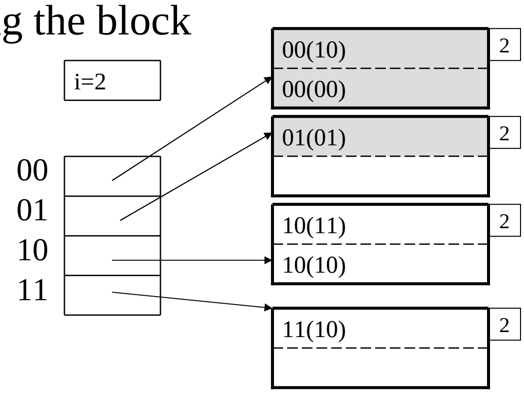


- Need to split block

14

Insertion in Extensible Hash Table

- After splitting the block



15

Extensible Hash Table

- How many buckets (blocks) do we need to touch after an insertion ?

- How many entries in the hash table do we need to touch after an insertion ?

16

Performance Extensible Hash Table

- No overflow blocks: access always one read
- BUT:
  - Extensions can be costly and disruptive
  - After an extension table may no longer fit in memory

17

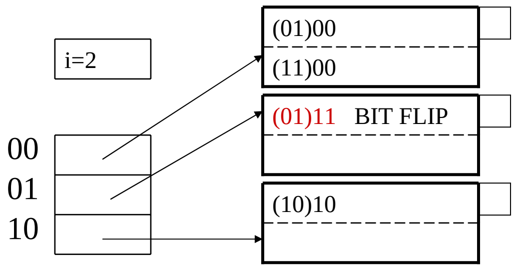
Linear Hash Table

- Idea: extend only one entry at a time
- Problem: n= no longer a power of 2
- Let i be such that $2^i <= n < 2^{i+1}$
- After computing h(k), use last i bits:
  - If last i bits represent a number > n, change msb from 1 to 0 (get a number <= n)

18

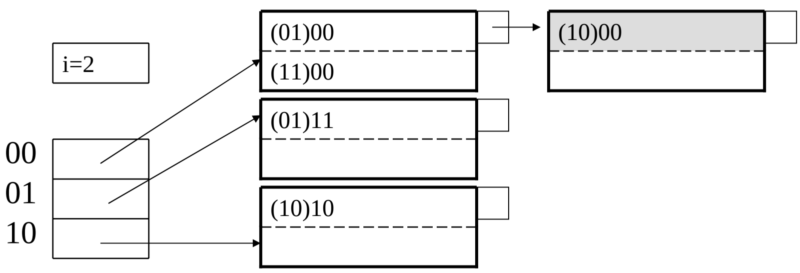
Linear Hash Table Example

- n=3



Linear Hash Table Example

- Insert 1000: overflow blocks…

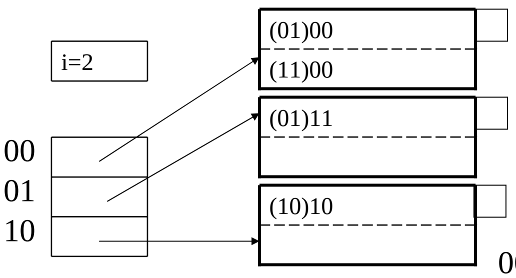


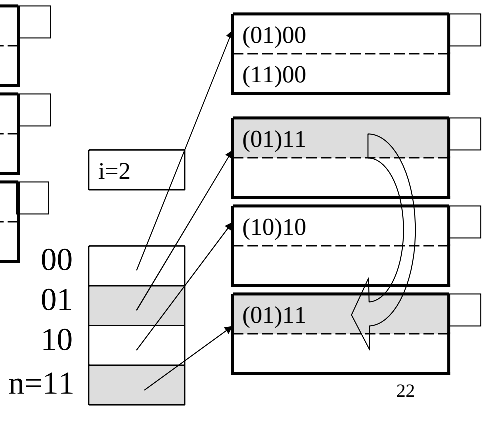
Linear Hash Tables

- Extension: independent on overflow blocks
- Extend n:=n+1 when average number of records per block exceeds (say) 80%

Linear Hash Table Extension

- From n=3 to n=4

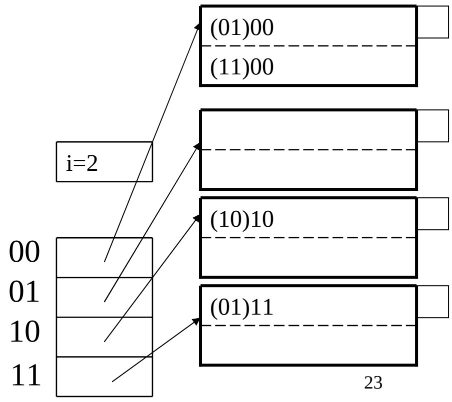




- Only need to touch one block (which one ?)

Linear Hash Table Extension

- From n=3 to n=4 finished
- Extension from n=4 to n=5 (new bit)
- Need to touch every single block (why ?)



Summary on Hash Tables

- Alternative index structures:
  - Simpler than B+ trees
  - Faster then B+ trees (when not full)
  - Degrade rapidly (when full)
- Used intensively during query processing

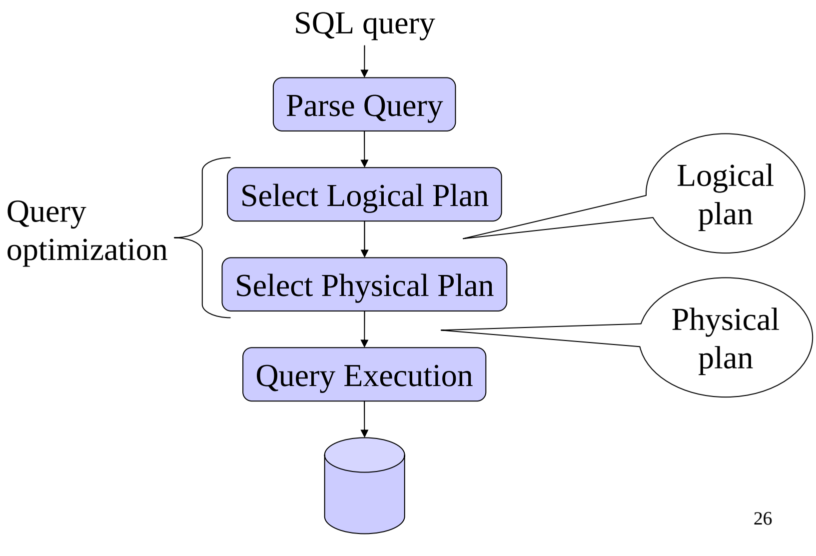
DBMS Architecture

How does a SQL engine work ?

- SQL query → relational algebra plan
- Relational algebra plan → Optimized plan
- Execute each operator of the plan

25

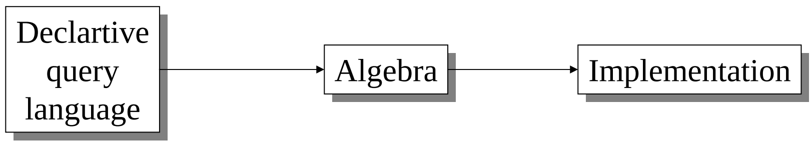
Architecture of a Database Engine



26

Relational Algebra

- Formalism for creating new relations from existing ones
- Its place in the big picture:



SQL,
relational calculus

Relational algebra
Relational bag algebra

27

Relational Algebra

- Five operators:
  - Union: $\cup$
  - Difference: -
  - Selection: σ
  - Projection: Π
  - Cartesian Product: $\times$
- Derived or auxiliary operators:
  - Intersection, complement
  - Joins (natural,equi-join, theta join, semi-join)
  - Renaming: ρ

28

1. Union and 2. Difference

- R1 $\cup$ R2
- Example:
  - ActiveEmployees $\cup$ RetiredEmployees
- R1 – R2
- Example:
  - AllEmployees -- RetiredEmployees

29

What about intersection ?

- It is a derived operator
- R1 $\cap$ R2 = R1 – (R1 – R2)
- Also expressed as a join (will see later)
- Example
  - UnionizedEmployees $\cap$ RetiredEmployees

30

3. Selection

- Returns all tuples which satisfy a condition
- Notation: $\sigma_c(R)$
- Examples
  - $\sigma_{Salary > 40000}$ (Employee)
  - $\sigma_{name = \text{"Smith"}}$ (Employee)
- The condition c can be =, <, ≤, >, ≥, <>

31

SSN	Name	Salary
1234545	John	200000
5423341	Smith	600000
4352342	Fred	500000

$\sigma_{Salary > 40000}$ (Employee)

SSN	Name	Salary
5423341	Smith	600000
4352342	Fred	500000

32

4. Projection

- Eliminates columns, then removes duplicates
- Notation: $\Pi_{A1,\ldots,An}(R)$
- Example: project social-security number and names:
  - $\Pi_{SSN,\ Name}$ (Employee)
  - Output schema: Answer(SSN, Name)

33

SSN	Name	Salary
1234545	John	200000
5423341	John	600000
4352342	John	200000

$\Pi_{Name,Salary}$ (Employee)

Name	Salary
John	20000
John	60000

34

5. Cartesian Product

- Each tuple in R1 with each tuple in R2
- Notation: R1 × R2
- Example:
  - Employee × Dependents
- Very rare in practice; mainly used to express joins

35

Name	SSN
John	9999999999
Tony	7777777777

EmpSSN	Dname
9999999999	Emily
7777777777	Joe

Name	SSN	EmpSSN	Dname
John	9999999999	9999999999	Emily
Tony	7777777777	7777777777	Joe
John	9999999999	9999999999	Emily
Tony	7777777777	7777777777	Joe

36

Relational Algebra

- Five operators:
  - Union: $\cup$
  - Difference: -
  - Selection: σ
  - Projection: Π
  - Cartesian Product: $\times$
- Derived or auxiliary operators:
  - Intersection, complement
  - Joins (natural,equi-join, theta join, semi-join)
  - Renaming: ρ

37

Renaming

- Changes the schema, not the instance
- Notation: $\rho_{B1,\dots,Bn}$ (R)
- Example:
  - $\rho_{LastName,\ SocSocNo}$ (Employee)
  - Output schema: Answer(LastName, SocSocNo)

38

Renaming Example

Employee

Name	SSN
John	999999999
Tony	777777777

$\rho_{LastName,\ SocSocNo}$ **(Employee)**

LastName	SocSocNo
John	999999999
Tony	777777777

39