CS186 Discussion Section Week 4

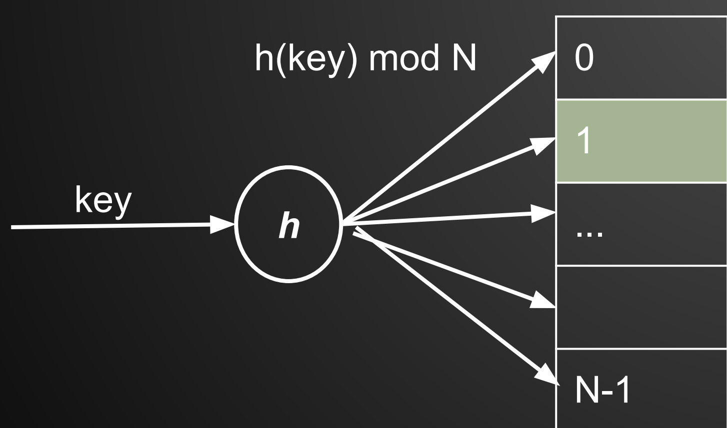
Hash-based Indexing
and
The Relational Algebra

Today

1. Hash vs. Tree Indexes
2. Hash-based Indexing
3. The Relational Algebra
4. Exercises!

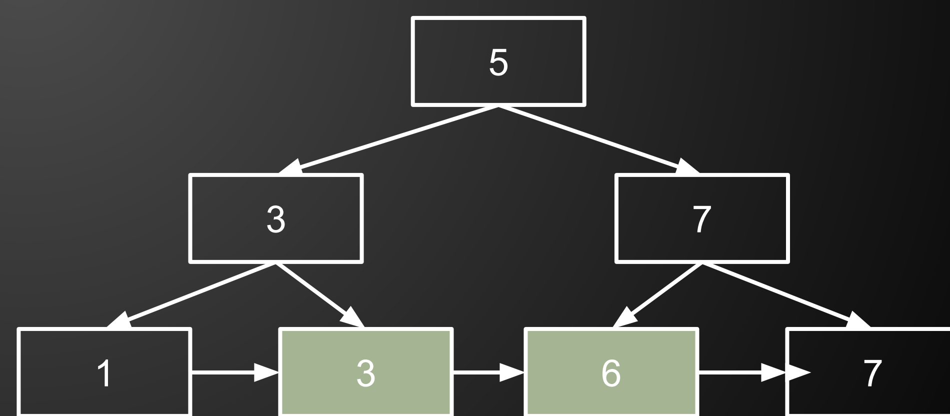
Review: Trees vs. Hashing

- Hashes good at:
  - *Equality*
  - *k == 1*
- Because:



Ideally one lookup!

- Trees good at:
  - *Range Queries*
  - *3 <= k <= 6*
- Because:



Data laid out to scan!

Today: Hash-based Indexes

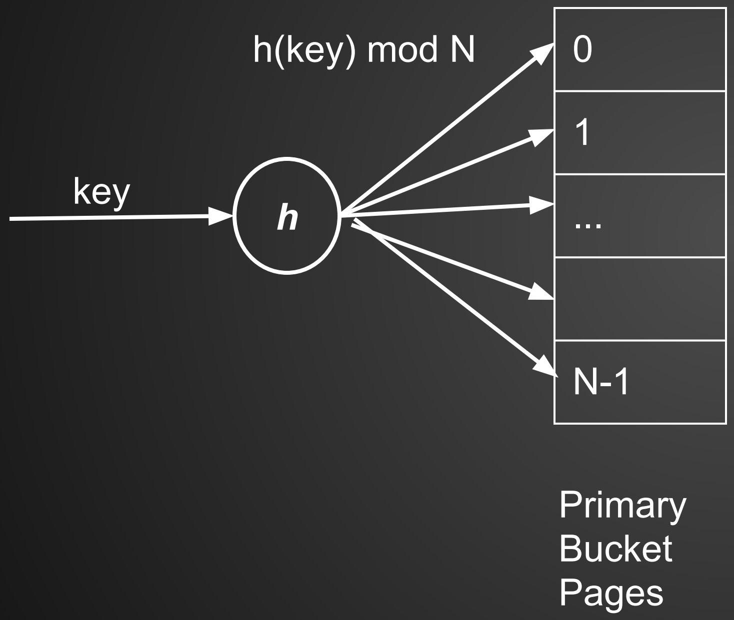
We are concerned with 3 kinds of hash-based indexes.

1. Static
   a. Like the ISAM of the hash world.
   b. Static structure - overflow pages.
2. Extendible
   a. Directory grows as data is added to it.
   b. Possibly too much!
3. Linear
   a. Lazy, dynamic variant of Extendible Hashing.

Static Hashing

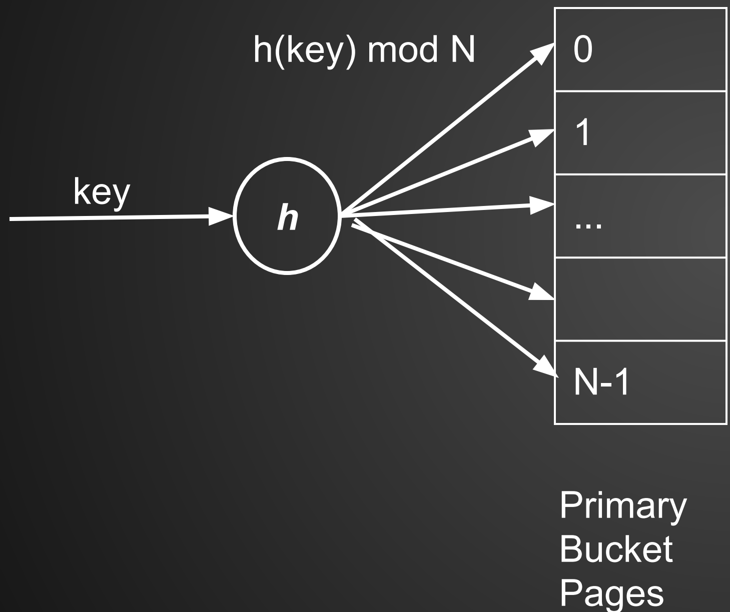
- Assume we have a file of N buckets.
- One *primary page* per bucket.
- As many *overflow pages* as needed.

Static Hashing



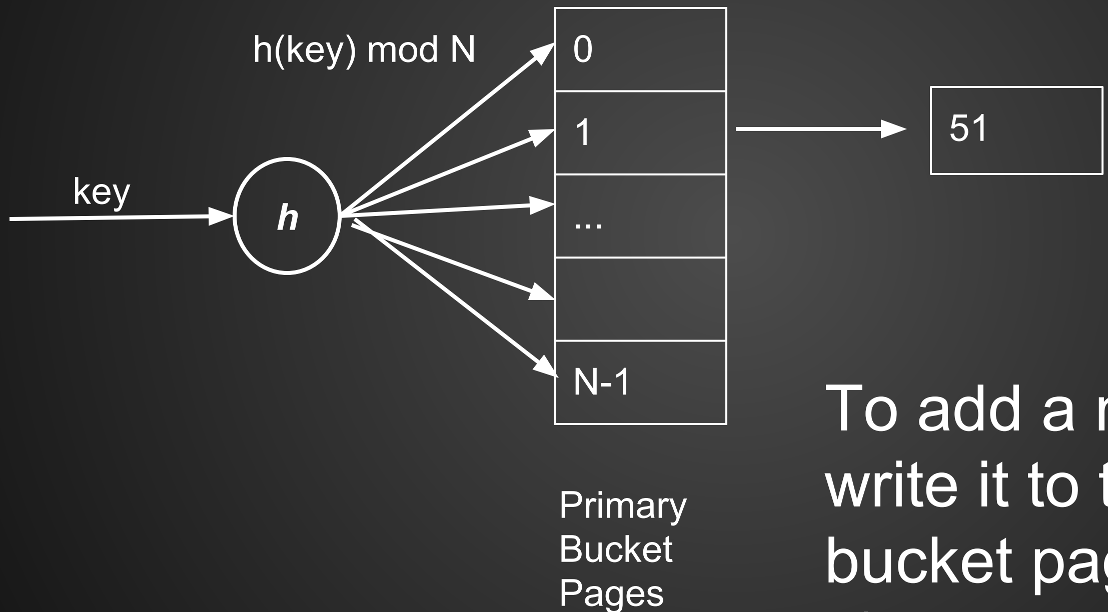
Static Hashing - Insert

Assume N=50 and that we can fit 1 record per page - also assume all pages are full!



Static Hashing - Insert

Assume N=50 and that we can fit 1 record per page - also assume all pages are full!



To add a new value, just write it to the corresponding bucket page. (51 mod 50 = 1)

Allocate new overflow page if necessary.

Extendible Hashing

- We don’t want overflow chains.
- What if we just rehash everything when our buckets get full?
  - Instead of mod N use mod N+1
- Problem!
  - Have to rehash everything!
  - Have to rewrite pages.
- Solution! Indirection!
  - Use a Directory.

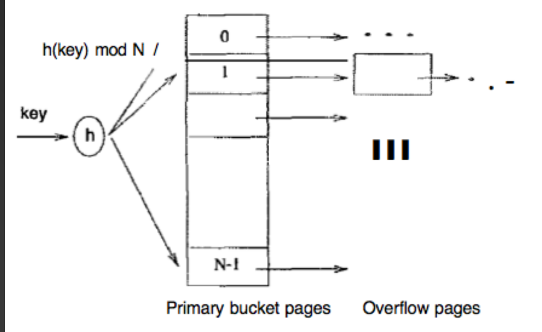


Figure 11.1 Static Hashing

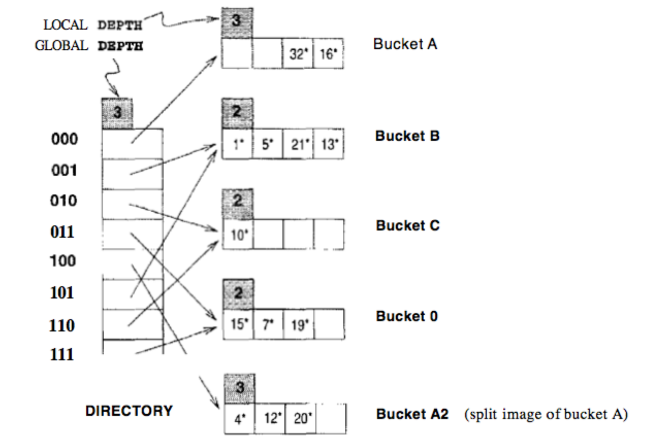


Figure 11.5 After Inserting Entry r with $h(r)$ = 20

Extendible Hashing

- Normal insert - simple!
  - Look up the right page and write to it.
- Insert to a full page - clever!
  - Redistribute the contents of the full page onto two new pages, increment local depth pointer.
  - If local depth <= global depth, update directory to point to two new pages.
  - Otherwise, double the size of the directory.
    - Update pointers to go to new page.
- Note we're doubling the size of the *directory*, which is probably small relative to our data pages.

Extendible Hashing - Example

Want to insert 20

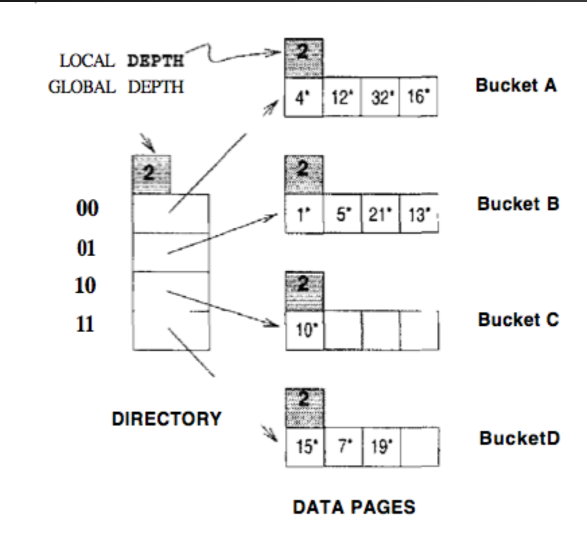


Figure 11.3 After Inserting Entry r with $h(r) = 13$

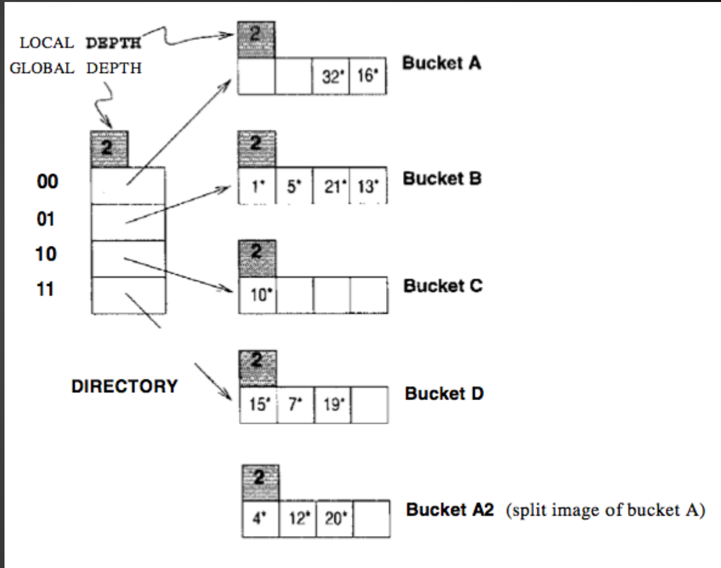


Figure 11.4 While Inserting Entry r with $h(r)=20$

Before

Reallocate the data and add 20

Extendible Hashing - Example

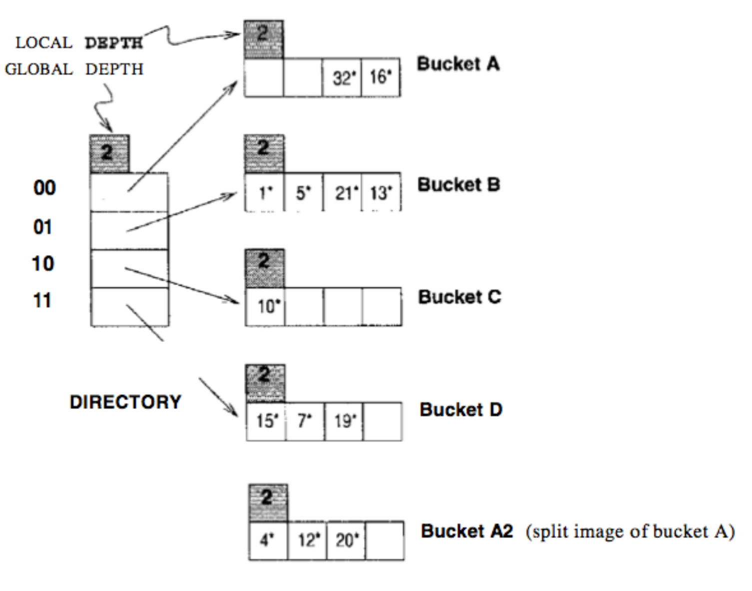


Figure 11.4 While Inserting Entry r with $h(r)=20$

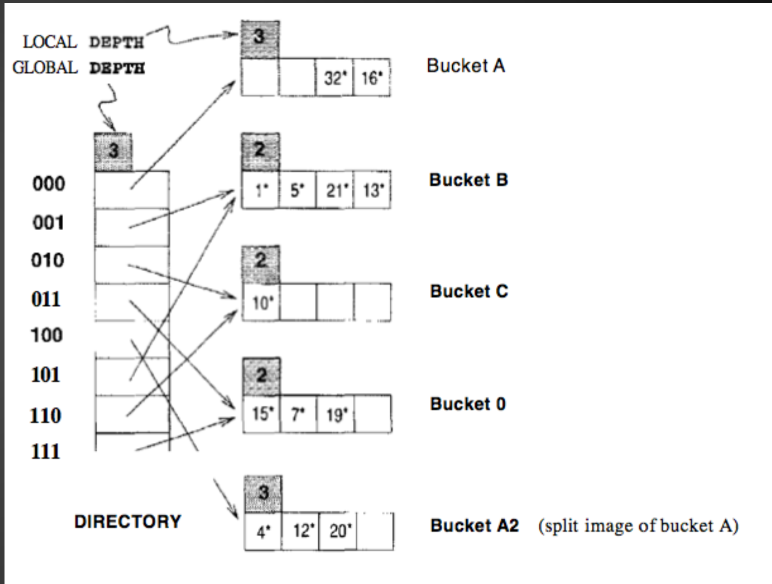


Figure 11.5 After Inserting Entry r with $h(r)$ = 20

Reallocate the data and add 20

Increase local depth, double directory, increase global depth.

Linear Hashing

- Dynamic, like extendible hashing
- No directory! (yay!)
- Overflow pages (boo!)
  - But not too many! (huzzah!)
- Slightly more complicated..

Linear Hashing - Setup

- Have a *family* of hash functions *h1, h2, h3…*
- With the property that each function has twice the RANGE of the previous one.
  - Usually $h_i(val) = h(val) \bmod (2^i N)$
  - Choose N to be a power of 2 (lets us look at last d_i bits of h(val) to evaluate $h_i(val)$.
  - d_i is related to our initial N ($\log_2(N)+i$)
- Example: take N=32 and h_i as above.

i	0	1	2	3
d	5	6	7	8

Linear Hashing - Rounds

- During a round L, we only use h_L and h_{L+1}
- Buckets present at the beginning of the round are split, one-by-one, until we've doubled the number of buckets.
- Keep track of *Next* bucket to be split (at beginning of round, *Next*=0).
- Round ends when *Next*==L.

Linear Hashing - Search

1. Apply $h_L(k)$
2. If $h_L(k)$ >= *Next*
   a. Key is in bucket $h_L(k)$
3. Else
   a. Key could be in $h_L(k)$ or $h_L(k)+2^L*N$ - have to check $h_{L+1}(k)$ be sure.

Linear Hashing - Insert

1. Apply $h_L(k)$ - if record fits, add to bucket.
2. Else
   a. Add overflow page to bucket $h_L(k)$ and insert record.
   b. Split *Next* bucket and increment *Next*.
      i. To split:
         1. Add new bucket to the end.
         2. Rehash all elements in *Next* according to h_{L+1} send them to the right page

- Note: *Next* is likely not same as $h_L(k)$
- We avoid long overflow chains by incrementally adding space
  - This breaks in the case of extreme *skew.*

Linear Hashing - Insert 43

Level=0

h_1	h_0
000	00
001	01
010	10
011	11

32	44	36	
9	25	5	
14	18	10	30
31	35	7	11

Next = 0

Primary Pages

Overflow Pages

Linear Hashing - Insert 43

Level=0

h_1	h_0
000	00
001	01
010	10
011	11
100	00

32			
9	25	5	
14	18	10	30
31	35	7	11
44	36		

Next = 1

Primary Pages

43			

Overflow Pages

***Once** Next > 3 we reset and increment the level.*

Today: Relational Algebra

- Way to ask queries of relations.
- Operators

Operation	Symbol	Explanation
Selection	σ	Selects rows
Projection	π	Selects columns
Union	$\cup$	
Intersection	$\cap$	
Cross-product	$\times$	
Join	$\bowtie$	Join tables on some condition
Division	/	Later..

Selection

- Select rows
- Example: $\sigma_{\text{birth\_year} < 1950}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Selection

- Select rows
- Example: $\sigma_{birth\_year < 1950}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Projection

- Select columns
- Example: $\pi_{\text{username, email}}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Projection

- Select columns
- Example: $\pi_{\text{username, email}}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Selection & Projection

- What's the difference between the following:
  - $\sigma_{\text{birth\_year} < 1950}(\pi_{\text{username, email}}(R))$
  - $\pi_{\text{username, email}}(\sigma_{\text{birth\_year} < 1950}(R))$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Selection & Projection

- What's the difference between the following:
  - $\sigma_{\text{birth\_year} < 1950}(\pi_{\text{username, email}}(R))$
  - $\pi_{\text{username, email}}(\sigma_{\text{birth\_year} < 1950}(R))$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Selection & Projection

- What’s the difference between the following:
  - $\sigma_{\text{birth\_year} < 1950}(\pi_{\text{username, email}}(R))$
  - $\pi_{\text{username, email}}(\sigma_{\text{birth\_year} < 1950}(R))$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Selection & Projection

- What's the difference between the following:
  - $\sigma_{\text{birth\_year} < 1950}(\pi_{\text{username, email}}(R))$
  - $\pi_{\text{username, email}}(\sigma_{\text{birth\_year} < 1950}(R))$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Selection & Projection

- What's the difference between the following:
  - $\sigma_{birth\_year < 1950}(\pi_{username,\ email}(R))$
  - $\pi_{username,\ email}(\sigma_{birth\_year < 1950}(R))$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Union

- Set union between two relations with same fields
- Example: $\sigma_{birth\_year < 1950}(R) \cup \sigma_{birth\_year\%2==0}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Union

- Set union between two relations with same fields
- Example: $\sigma_{birth\_year < 1950}(R) \cup \sigma_{birth\_year\%2==0}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920

U

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980

Union

- Set union between two relations with same fields
- Example: $\sigma_{\text{birth\_year} < 1950}(R) \cup \sigma_{\text{birth\_year\%2==0}}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980

Intersection

- Set intersection between two relations with same fields
- Example: $\sigma_{\text{birth\_year} < 1950}(R) \cap \sigma_{\text{birth\_year\%2==0}}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Intersection

- Set intersection between two relations with same fields
- Example: $\sigma_{\text{birth\_year} < 1950}(R) \cap \sigma_{\text{birth\_year\%2==0}}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920

$\cap$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980

Intersection

- Set intersection between two relations with same fields
- Example: $\sigma_{\text{birth\_year} < 1950}(R) \cap \sigma_{\text{birth\_year\%2==0}}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920

Set Difference

- A – B takes out all rows in B from A.
- Example: $\sigma_{birth\_year\%2==0}(R) - \sigma_{birth\_year < 1950}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Set Difference

- A – B takes out all rows in B from A.
- Example: $\sigma_{birth\_year\%2==0}(R) - \sigma_{birth\_year < 1950}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980

-

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920

Set Difference

- A – B takes out all rows in B from A.
- Example: $\sigma_{birth\_year\%2==0}(R) - \sigma_{birth\_year < 1950}(R)$

Username	Email	Birth Year
beep	beep@gmail.com	1980

Cross product

- A x B takes all rows from A and combines them with all rows from B.
- Example: $\pi_{username}(R) \times \pi_{birth\_year}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Cross product

- A x B takes all rows from A and combines them with all rows from B.
- Example: $\pi_{username}(R) \times \pi_{birth\_year}(R)$

Username
meep
notmeep
beep
beepboop

X

Birth Year
1920
1980
1985

=

Cross product

- A x B takes all rows from A and combines them with all rows from B.
- Example: $\pi_{username}(R) \times \pi$

Username
meep
notmeep
beep
beepboop

X

Birth Year
1920
1980
1985

=

Username	Birth Year
meep	1920
meep	1980
meep	1985
notmeep	1920
notmeep	1980
notmeep	1985
beep	1920
beep	1980
beep	1985
beepboop	1920
beepboop	1980
beepboop	1985

Join

- A $\bowtie$ B joins A and B based on some column.
  - Natural join: Default, joins on matching columns.
  - Can also specify a particular join predicate.
- Example: $\pi_{\text{username,email}}(R) \bowtie \pi_{\text{username,birth\_year}}(R)$

Username	Email
meep	meep@gmail.com
notmeep	notMeep@gmail.com
beep	beep@gmail.com
beepboop	blitzcrank@gmail.com

$\bowtie$

Username	Birth Year
meep	1920
notmeep	1920
beep	1980
beepboop	1985

Join

- A $\bowtie$ B joins A and B based on some column.
  - Natural join: Default, joins on matching columns.
  - Can also specify a particular join predicate.
- Example: $\pi_{username,email}(R) \bowtie \pi_{username,birth\_year}(R)$

Username	Email	Birth Year
meep	meep@gmail.com	1920
notmeep	notMeep@gmail.com	1920
beep	beep@gmail.com	1980
beepboop	blitzcrank@gmail.com	1985

Division

- A / B projects the columns from A that are not in B, while selecting elements of A that fully intersect with B.
- Think of it as opposite of cross-product!
- Example:

Person	Class
meep	cs186
meep	cs162
moop	cs186
moop	cs61
beep	cs186
beep	cs162

/

Class
cs186
cs162

=

Person
meep
beep

Division

- Can be expressed with the following equation:

$$\pi_x(A) - \pi_x((\pi_x(A) \times B) - A)$$

- Things to eliminate: Things that are in the cross product of $\pi_x(A)$ and B that are NOT in A.
- Subtract the left hand side (projection) of those candidates from $\pi_x(A)$.