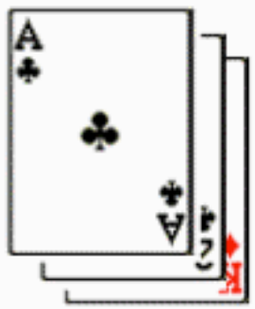
Indexing

Professor Larry Heimann
Carnegie Mellon University
Information Systems Program

Note: the plural of ‘index’ is traditionally ‘indices’, however today ‘indexes’ is considered appropriate and is the norm among database professionals.

Why index?

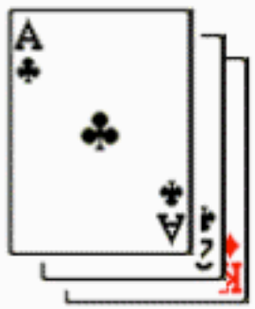
Random Pile of Cards



To Find 8 of Hearts...
Scan 52 Cards (26)
Average Flips = 26

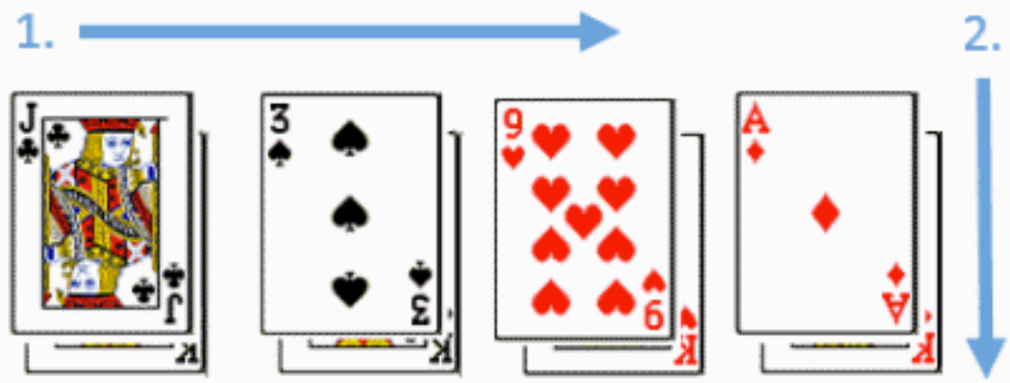
Why index?

Random Pile of Cards



To Find 8 of Hearts...
Scan 52 Cards (26)
Average Flips = 26

Random Pile of Cards By Suit



To Find 8 of Hearts...
1. Pick Suit (2)
2. Scan 13 Cards (7)
Average Flips = 9

Creating Piles reduces flip from 26 to 9!

Types of indexes

- *B-Tree* -- the default that you get in postgres. The B stands for Balanced, and the idea is that the amount of data on both sides of the tree is roughly the same so the number of levels that must be traversed to find rows is always in the same ballpark. B-Tree indexes can be used for equality and range queries efficiently.

- *Hash Indexes* -- only useful for equality comparisons, but are not transaction safe, so rarely used.

- *Generalized Inverted Indexes (GIN)* -- useful when an index must map many values to one row, whereas B-Tree indexes are optimized for when a row has a single key value. GINs are good for indexing array values as well as for implementing full-text search.

- *Generalized Search Tree (GiST) Indexes* -- allow you to build general balanced tree structures, and can be used for operations beyond equality and range comparisons. They are used to index the geometric data types, as well as full-text search. Ideal for smaller sets of records.

The trouble with indexes

- No indexing at all

  “Indexes add overhead and slow down inserts and updates”

- Index shotgun

  “Index every field to make the database superfast to query”

- Finding the middle ground

Creating indexes

- Example:

```
CREATE INDEX defects_sources_idx ON defects(source_id);

CREATE INDEX users_names_idx ON users(last_name, first_name);
```

- Creating index can take time depending on the number of records
- Creating indexes adds time to insert and update commands, but saves time on selects
- Query optimizer will determine if index exists that will help speed up query

Partial indexes

- Covers just a subset of the table's data
- Essentially a index with a where clause
- Example:

```
CREATE INDEX defects_faculty_idx
ON defects(reporter_id)
WHERE reporter_id = 1 or reporter_id = 2
```

Expression indexes

- Can create an index which uses a function
- Common example

```
CREATE INDEX users_lastname_idx ON users(lower(last_name));
```

- Another example with dates

```
CREATE INDEX articles_day ON articles(date(published_at));
```

can be used by a query containing `WHERE date(articles.created_at) = date('2011-03-07')`

What to index

- Unique fields
- Foreign keys
- Queries commonly run
- Fields commonly combined

Full-text indexing and searching