Privacy Integrated Queries (PINQ)

An Extensible Platform for Privacy-Preserving Data Analysis

Frank McSherry
Microsoft Research, SVC
mcsherry@microsoft.com

The Setting:

Analysts need / want / have access to sensitive data. Want privacy.
Data holders want to give access to non-experts; are non-experts.

PINQ provides data analysis tools with formal privacy guarantees.
Absolutely no privacy training required for data holders or analysts.

What is PINQ?

PINQ looks a lot like **LINQ**, a data access / manipulation API.
Data are transformed using `Where`, `Select`, `GroupBy`, `Join`, ...

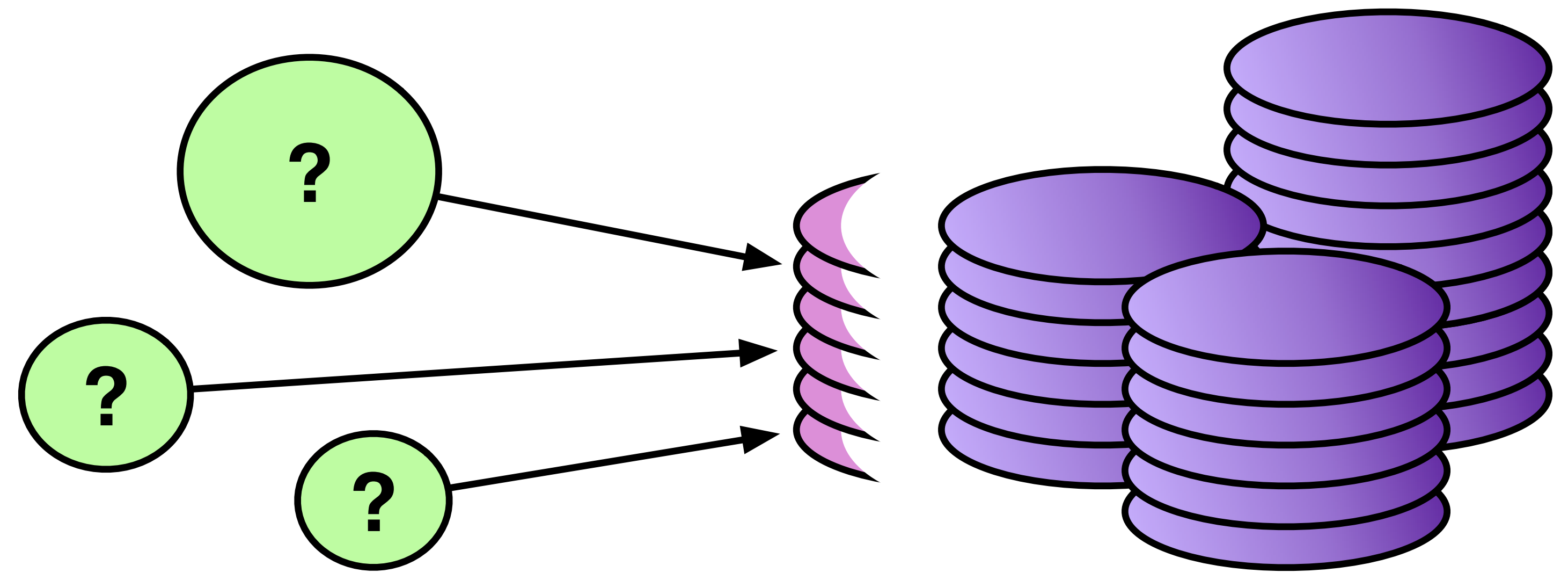
1. No direct access to the data. Standard aggregations disabled.
2. Users get access to a set of privacy-preserving aggregations.
3. **PINQ** ensures that any/all queries provide **differential privacy**.

What is **Differential Privacy**? (recent work w/Cynthia Dwork)

1. Gives unconditional guarantees about privacy, in any context.
2. Applies to arbitrary data types: numeric, text, medical images, ...

How PINQ Works

Thin layer around any **LINQ** data.
Gives **LINQ**-like interface to data.



Queries to data are intercepted:

1. Checked for naughty tricks.
2. Checked against data policy.
3. Executed, but randomized.

Raw data never leaves **PINQ**.

Where is it useful?

Medicine, Education, Finance, ...
External research / collaboration.
Best practices within groups.
Accelerating R&D into PPDM.

Privacy Integrated Queries (PINQ)

An Extensible Platform for Privacy-Preserving Data Analysis

Frank McSherry
Microsoft Research, SVC
mcsherry@microsoft.com

Differential Privacy [DM06]

Strongest (useful) privacy standard currently known:

For any data set **D**, and any person **P**, and any output **O**,

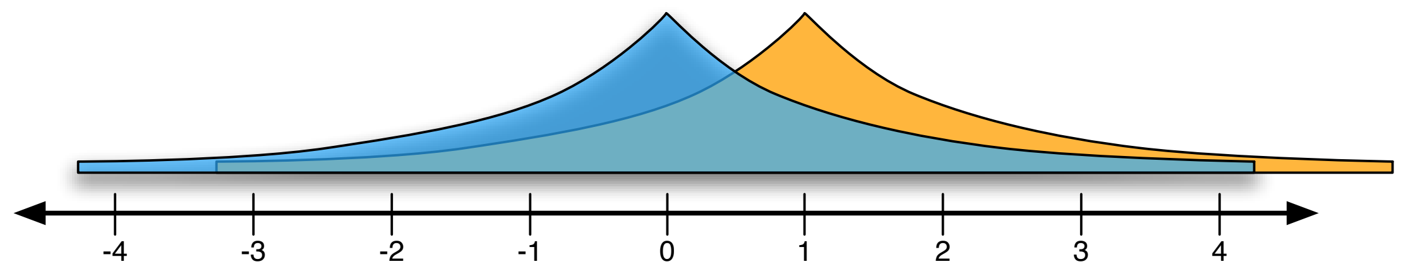
Pr[**O** with **P** in **D**] < exp(eps) **Pr**[**O** with **P** not in **D**]

Computation indistinguishable from setting where user opted out.
Any consequence of the computation as likely with or without you.

Examples [DMNS06]

Easy example of Differential Privacy:

```
// returns Count() +/- error
int NoisyCount(double eps);
{
  return Count() + Laplace(1.0/eps);
}
```



Many more interesting examples exist.
Analysts shouldn't have to write them.

Differential Privacy vs.

Properties other approaches lack:

1. Composes. Can use it a lot.
2. Independent of data type.
3. No cryptographic assumptions.
4. Easy to explain to participants.
5. Many, many other properties.
6. Actually guarantees privacy.

Privacy and Transformations

Many transformations **T** are "stable": for any data sets **A** and **B**:

|| **T**(**A**) - **T**(**B**) || is at most **C** || **A** - **B** ||

Any **C**-stable transformation is compatible with differential privacy.
Apply transformation, then analysis. eps increases by factor of **C**.

Some **C** values: `Where()` = 1, `Select()` = 1, `GroupBy()` = 2, `Join()` =

Privacy Integrated Queries (PINQ)

An Extensible Platform for Privacy-Preserving Data Analysis

Frank McSherry
Microsoft Research, SVC
mcsherry@microsoft.com

Using PINQ

Based on `PINQueryable<T>` type. Protected data set. Looks like `IQueryable<T>`, has transformations like:

```
PINQueryable Where(Expr<Func<T,bool>> pred);
PINQueryable Select(Expr<Func<T,S>> function);
...
```

Also supports a few differential privacy analyses:

```
int NoisyCount(double epsilon);  // eps-private
```

Each method carefully considered, many excluded.

Inside a PINQueryable

A `PINQueryable` contains only two member variables:

```
IQueryable source; // any LINQ data source
PINQAgent  agent;  // acts as privacy proxy
```

A `PINQAgent` provides: `bool alert(double epsilon)`

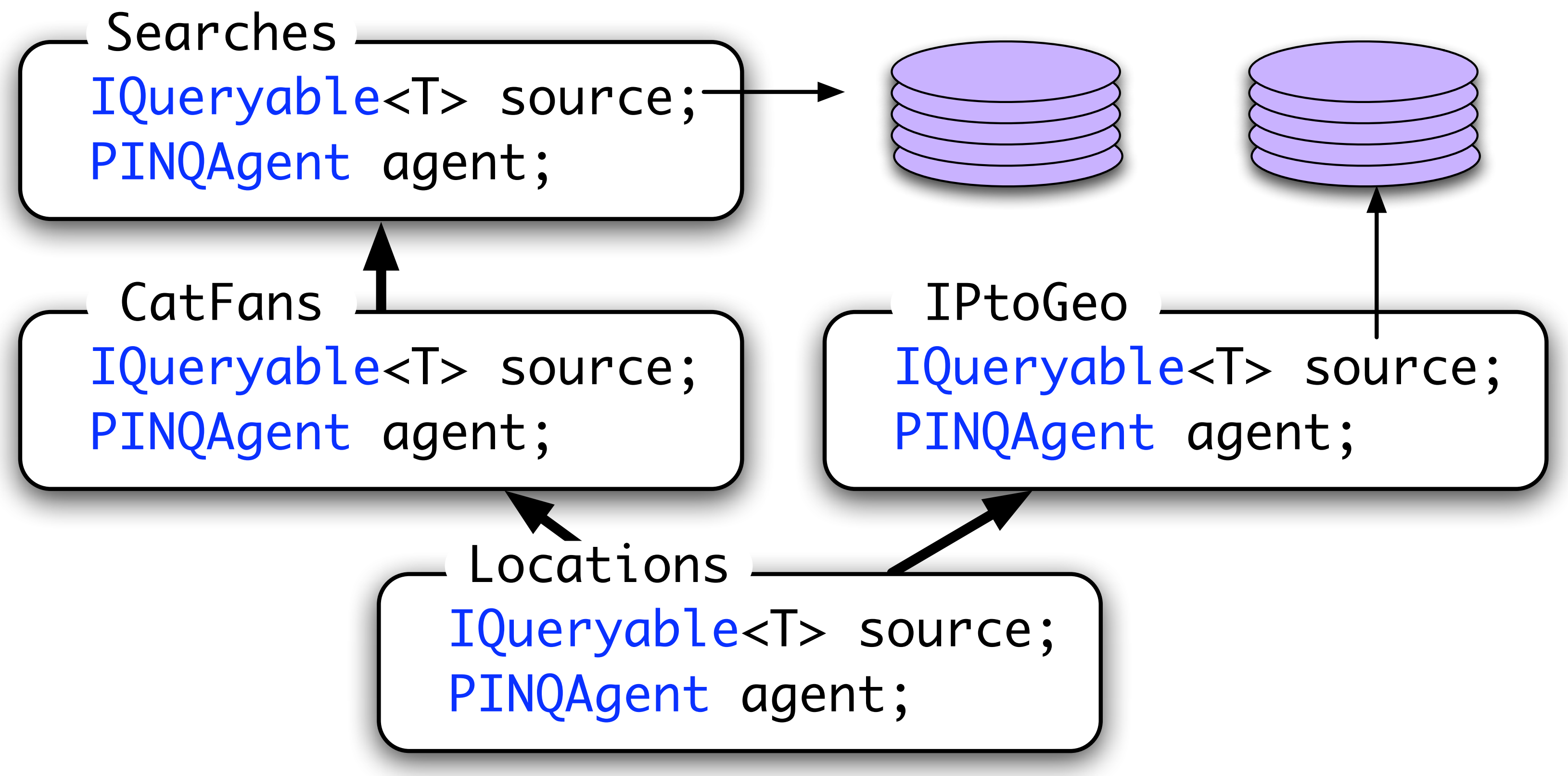
Accepts or rejects requests for additional epsilon. Data provider writes arbitrary code for this policy.

A Simple Example:

```
PINQueryable<SearchQuery> Searches = openSearches(password);

var CatFans = Searches.GroupBy(search => search.UserID).
                       Where(user => user.Where(query => query == "cats").
                                        Count() > 10);

PINQueryable<IPGeo> IPtoGeo = openIPtoGeo(password);
var Locations = CatFans.Join(IPtoGeo, x => x.IP, y => y.IP, (x,y) => y.LatLon);
```



Searches
IQueryable<T> source;
PINQAgent agent;

CatFans
IQueryable<T> source;
PINQAgent agent;

IPtoGeo
IQueryable<T> source;
PINQAgent agent;

Locations
IQueryable<T> source;
PINQAgent agent;

Arguments to methods checked for sneaky tricks. Should have no side effects, always return, etc...

```
Console.WriteLine("Count: {0}", Locations.Distinct().NosiyCount(0.1);
```

Analyses trigger `PINQAgent::alert(epsilon)` requests. If all come back affirmative, **LINQ** layer is invoked.

Privacy Integrated Queries (PINQ)

An Extensible Platform for Privacy-Preserving Data Analysis

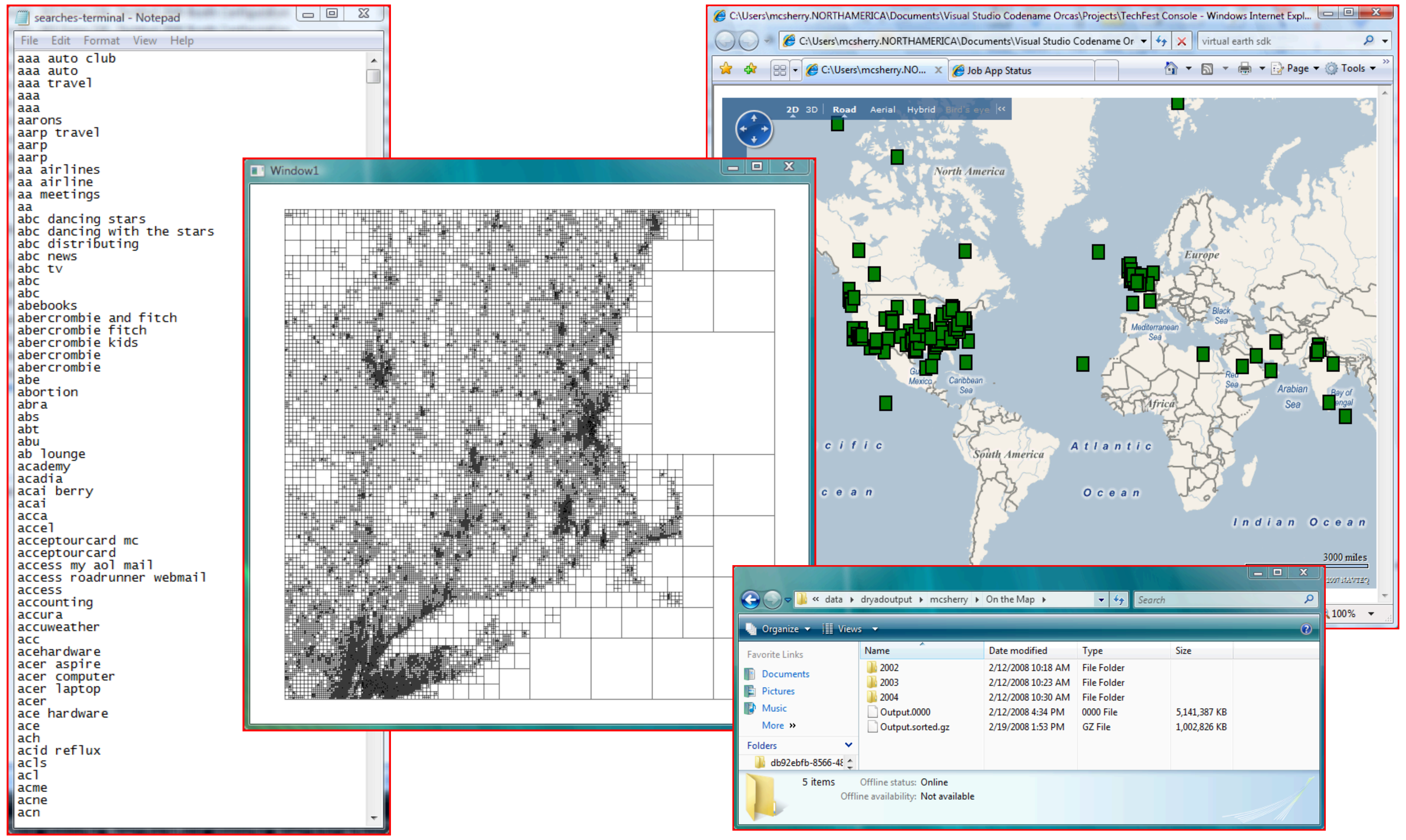
Frank McSherry
Microsoft Research, SVC
mcsherry@microsoft.com

Programming with PINQ

Most user code gets written in C# outside of **PINQ**.
No restrictions. Write to disk, network; debug; etc...

Code passed to **PINQ** (expressions) must be simple.
Programming in **PINQ** takes thought. Makes sense.

Things PINQ has Output: (ask to see the code!)



searches-terminal - Notepad

```
aaa auto club
aaa auto
aaa travel
aaa
aaa
aarons
aarp travel
aarp
aarp
aa airlines
aa airline
aa meetings
aa
abc dancing stars
abc dancing with the stars
abc distributing
abc news
abc tv
abc
abc
abebooks
abercrombie and fitch
abercrombie fitch
abercrombie kids
abercrombie
abercrombie
abe
abortion
abra
abs
abt
abu
ab lounge
academy
acadia
acai berry
acai
acca
accel
acceptourcard mc
acceptourcard
access my aol mail
access roadrunner webmail
access
accounting
accura
accuweather
acc
acehardware
acer aspire
acer computer
acer laptop
acer
ace hardware
ace
ach
acid reflux
acls
acl
acme
acne
acn
```

« data ▸ dryadoutput ▸ mcsherry ▸ On the Map ▸

Name	Date modified	Type	Size
2002	2/12/2008 10:18 AM	File Folder	
2003	2/12/2008 10:23 AM	File Folder	
2004	2/12/2008 10:30 AM	File Folder	
Output.0000	2/12/2008 4:34 PM	0000 File	5,141,387 KB
Output.sorted.gz	2/19/2008 1:53 PM	GZ File	1,002,826 KB

5 items Offline status: Online
Offline availability: Not available

A Neat PINQ Program:

A quick program to see what people search for:

```
using PINQ; // its just a library! :D

// Outputs common strings, found by expanding common prefixes.
public void TextAnalysis(PINQueryable<string> data, string prefix)
{
  // only continue as long as there is evidence of data
  if (data.NoisyCount(0.1) > 100)
  {
    // partition data by first character in the string
    var parts = data.Partition(alphabet, x => x.Substring(0,1));
    foreach(var letter in alphabet)
    {
      if (letter != "")  // continue on strings that have not ended
      {
        var suffixes = parts[letter].Select(x => x.Substring(1));
        TextAnalysis(suffixes, prefix + letter);
      }
    }

    // if many input strings equal prefix, output it
    if (parts[""].NoisyCount(1.0)  > 10)
      Console.WriteLine("{0}", prefix);
  }
}

PINQueryable<string[]> searchLogs = readSearchLog("searchLogs.txt");

TextAnalysis(searchLogs.Select(x => x[20]), "");
TextAnalysis(searchLogs.Where(x => x[13] == "WA").Select(x => x[20]), "");
TextAnalysis(searchLogs.Where(x => x[0] == "-").Select(x => x[20]), "");
```

No prior knowledge of what terms are "sensitive".