INF 102
CONCEPTS OF PROG. LANGS
SQL AND SPREADSHEETS

Instructors: James Jones

Data-Centric Programming

- Focus on data

- Interactive data: SQL
- Dataflow: Spreadsheets
- Dataflow: Iterators, generators, coroutines

SQL

Standard Query Language

History

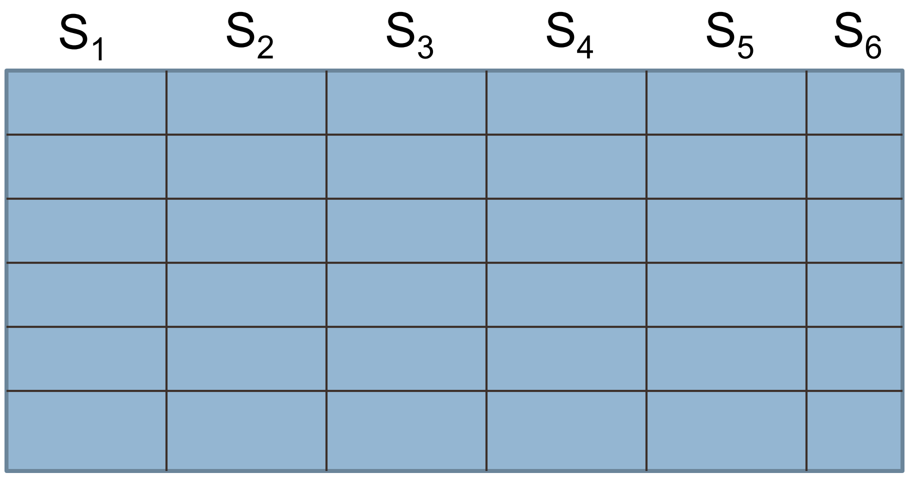
- Data banks since 1950s
- Disks (direct access storage) in 1960s
- How to store and retrieve data from disk
  - Efficiently, cleanly
- Before 1970:
  - Hierarchical models (trees)
  - Network models (graphs)
- E. Codd, 1970:
  - Relational model

Relational model

- Logic deductive system
  - Data independence – isolate applications from data representations
  - Data inconsistency
- "Relation" as in Mathematics:
  - Given sets $S_1, S_2, \ldots S_n$: R is a relation on these sets iff $R = \{\{e_1, e_2, \ldots, e_n\}, \ldots\}$ where $e_i \in S_i$
  - $R \subseteq S_1 x\ S_2 x \ldots xS_n$
    (R is a subset of the Cartesian product)

Relations

- Each column represents a domain
  - Ordering of columns is important – order of the domains of R
- Each row represents an n-tuple of R
  - Ordering of rows is immaterial
  - All rows are distinct



Relations in data bases

- Relations define subsets of the domain:
  $R \subseteq S_1 x\ S_2 x \ldots xS_n$

supply (supplier part project quantity)

supplier	part	project	quantity
1	2	5	17
1	3	5	23
2	3	7	9
2	7	5	4
4	1	1	12

Supply is a relation (subset) from
supplier x part x project x quantity → supplier x part x project x quantity
And
supplier, part, project, quantity all subsets of Int

Relations in data bases

- Relations may include repeated domains

component (part part quantity)

part	part	quantity
1	5	9
2	5	7
3	5	2
2	6	12
3	6	3

"Part 1 is a subpart of part 5, and there needs to be 9 part 1s to make a part 5"

Component is a relation (subset) from
part x part x quantity → part x part x quantity
And
part, part, project, quantity all subsets of Int

Relationships

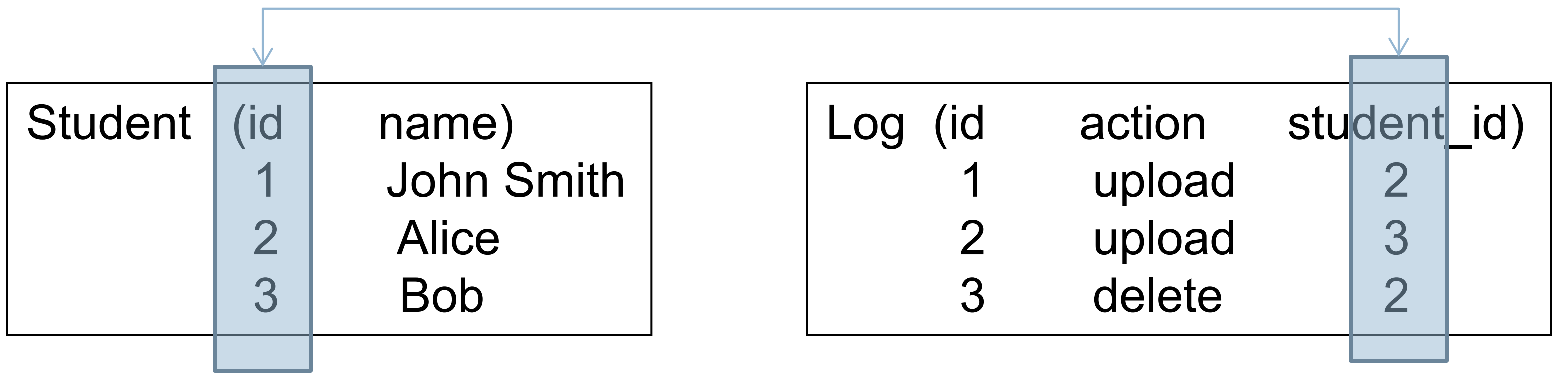
- Domain order not important
- Same domains are distinguished by role names: attributes
- User-facing model

component (**subpart** **part** quantity)

subpart	part	quantity
1	5	9
2	5	7
3	5	2
2	6	12
3	6	3

Cross-references

- Elements of a relation can cross-reference elements of the same or another relation
- Done via Keys



Student	(id	name)
	1	John Smith
	2	Alice
	3	Bob

Log	(id	action	student_id)
	1	upload	2
	2	upload	3
	3	delete	2

Operations on relations

- Permutation
  - Interchanging columns yields converse relations
- Subsetting
  - Selecting only a subset of tuples
- Projection
  - Selection of only a subset of columns
- Join
  - Merging two or more relations without loss of information

Relational Model → SQL

- Data Definition Language (DDL)
  - Create/alter/delete tables and their attributes
- Data Manipulation Language (DML)
  - Query one or more tables
  - Insert/delete/modify tuples in tables

Subsetting

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   *
FROM     Product
WHERE    category='Gadgets'
```

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

"selection"

Projection+Subsetting

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100
```

PName	Price	Manufacturer
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

“selection” and “projection”

Joins

Product (<u>pname</u>, price, category, manufacturer)
Company (<u>cname</u>, stockPrice, country)

Find all products under $200 manufactured in Japan;
return their names and prices.

Join
between Product
and Company

```
SELECT   PName, Price
FROM     Product, Company
WHERE    Manufacturer=CName AND Country='Japan'
         AND Price <= 200
```

Joins

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

Cname	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

```
SELECT   PName, Price
FROM     Product, Company
WHERE    Manufacturer=CName AND Country=‘Japan’
         AND Price <= 200
```

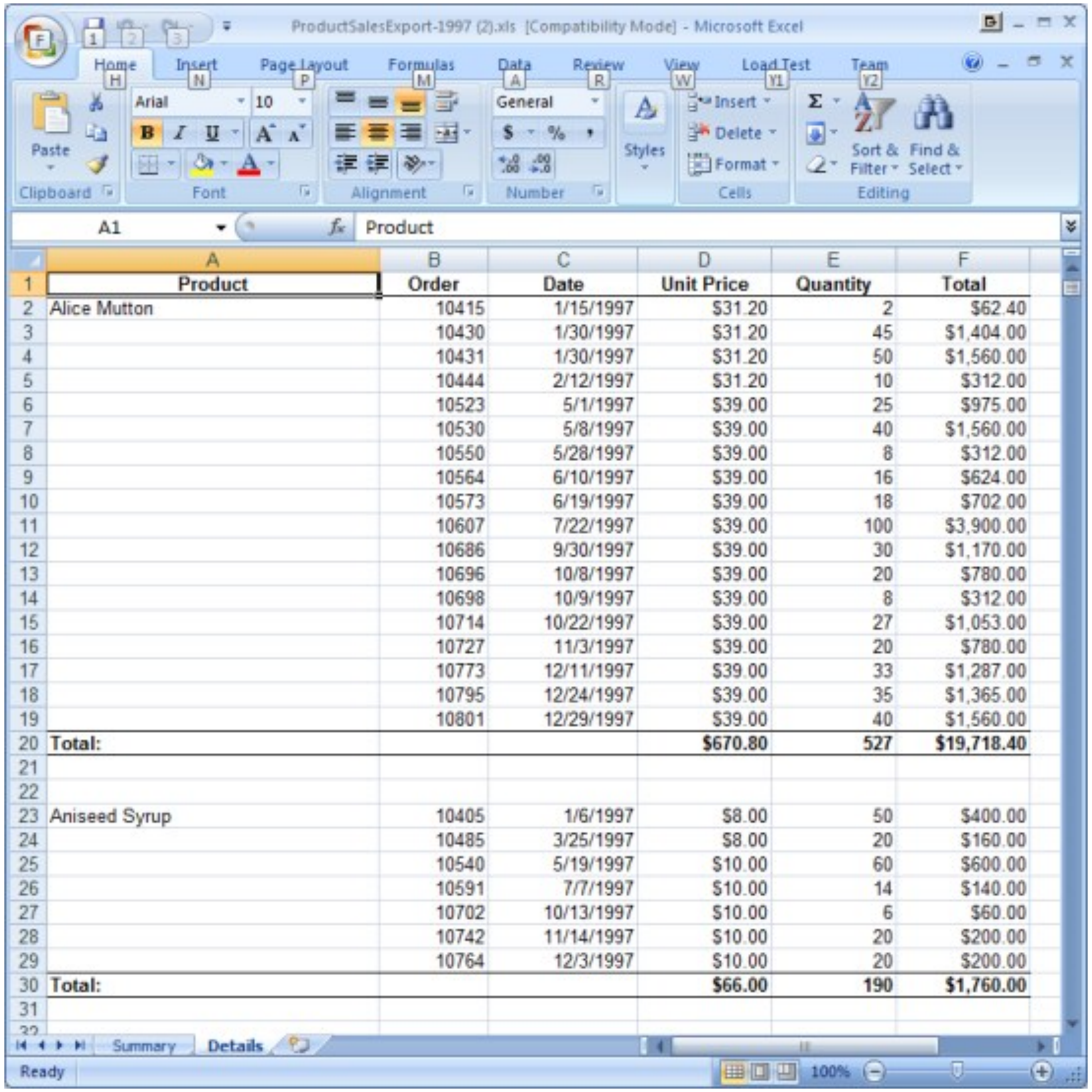
PName	Price
SingleTouch	$149.99

Full SQL

- Very powerful query language
  - Ordering, Grouping, aggregation, rich type system, …
- Declarative
  - Say *what* you want, not *how* you want it to happen
  - Nothing related to query processing or internal data representations

Spreadsheets

Spreadsheets

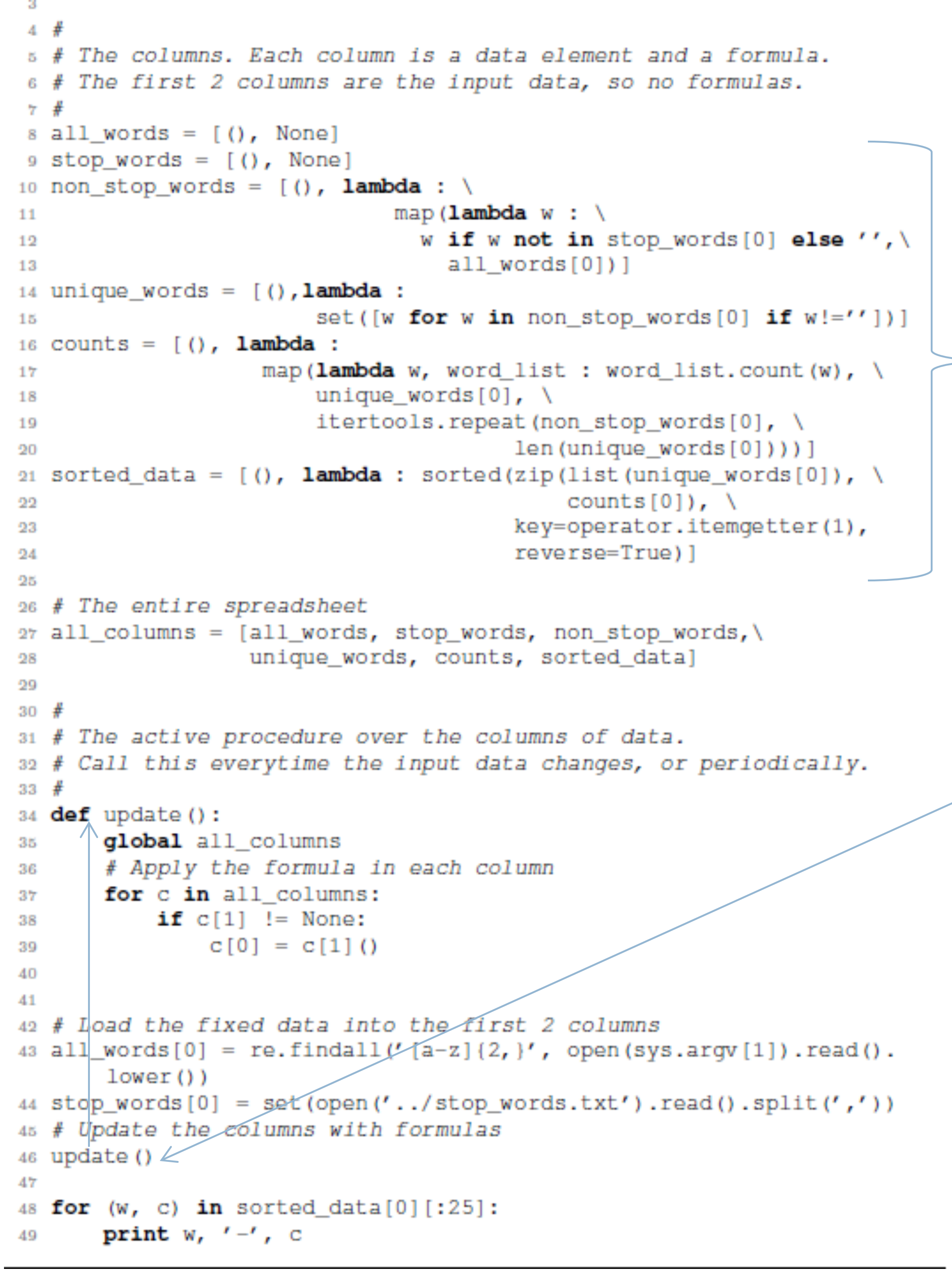


	A	B	C	D	E	F
1	Product	Order	Date	Unit Price	Quantity	Total
2	Alice Mutton	10415	1/15/1997	$31.20	2	$62.40
3		10430	1/30/1997	$31.20	45	$1,404.00
4		10431	1/30/1997	$31.20	50	$1,560.00
5		10444	2/12/1997	$31.20	10	$312.00
6		10523	5/1/1997	$39.00	25	$975.00
7		10530	5/8/1997	$39.00	40	$1,560.00
8		10550	5/28/1997	$39.00	8	$312.00
9		10564	6/10/1997	$39.00	16	$624.00
10		10573	6/19/1997	$39.00	18	$702.00
11		10607	7/22/1997	$39.00	100	$3,900.00
12		10686	9/30/1997	$39.00	30	$1,170.00
13		10696	10/8/1997	$39.00	20	$780.00
14		10698	10/9/1997	$39.00	8	$312.00
15		10714	10/22/1997	$39.00	27	$1,053.00
16		10727	11/3/1997	$39.00	20	$780.00
17		10773	12/11/1997	$39.00	33	$1,287.00
18		10795	12/24/1997	$39.00	35	$1,365.00
19		10801	12/29/1997	$39.00	40	$1,560.00
20	**Total:**			**$670.80**	**527**	**$19,718.40**
21						
22						
23	Aniseed Syrup	10405	1/6/1997	$8.00	50	$400.00
24		10485	3/25/1997	$8.00	20	$160.00
25		10540	5/19/1997	$10.00	60	$600.00
26		10591	7/7/1997	$10.00	14	$140.00
27		10702	10/13/1997	$10.00	6	$60.00
28		10742	11/14/1997	$10.00	20	$200.00
29		10764	12/3/1997	$10.00	20	$200.00
30	**Total:**			**$66.00**	**190**	**$1,760.00**
31						

Spreadsheets

- One of the most successful software genres
- Centuries-old accounting practices…
  - Some cells contain primitive values
  - Some cells contain values derived from formulas
- …with computers
  - Automatic update of derived values when primitive values change

→ Dataflow programming



```
#
# The columns. Each column is a data element and a formula.
# The first 2 columns are the input data, so no formulas.
#
all_words = [(), None]
stop_words = [(), None]
non_stop_words = [(), lambda : \
                            map(lambda w : \
                              w if w not in stop_words[0] else '',\
                                all_words[0])]
unique_words = [(),lambda :
                    set([w for w in non_stop_words[0] if w!=''])]
counts = [(), lambda :
                map(lambda w, word_list : word_list.count(w), \
                    unique_words[0], \
                    itertools.repeat(non_stop_words[0], \
                                   len(unique_words[0])))]
sorted_data = [(), lambda : sorted(zip(list(unique_words[0]), \
                                       counts[0]), \
                                   key=operator.itemgetter(1),
                                   reverse=True)]

# The entire spreadsheet
all_columns = [all_words, stop_words, non_stop_words,\
               unique_words, counts, sorted_data]

#
# The active procedure over the columns of data.
# Call this everytime the input data changes, or periodically.
#
def update():
    global all_columns
    # Apply the formula in each column
    for c in all_columns:
        if c[1] != None:
            c[0] = c[1]()


# Load the fixed data into the first 2 columns
all_words[0] = re.findall('[a-z]{2,}', open(sys.argv[1]).read().
    lower())
stop_words[0] = set(open('../stop_words.txt').read().split(','))
# Update the columns with formulas
update()

for (w, c) in sorted_data[0][:25]:
    print w, '-', c
```

In Python:

Columns = 2-part lists:
- data
- formula

All formulas run on updates

In OOP

- Columns = Objects with 2 parts, data and formula
- Formulas = Objects with method “execute”
- “Map” function: applies a given function to one or more list of values
  - Check for equivalents in C++ (Boost maybe?), C# (Select)
  - Not hard to do by hand: iterate

Reactive Style

- Dataflow programming in the spreadsheet style
- Observables: Product values
- Operators: Take 1 or more Observables and produce new Observable
- Observers consume (or listen to) values from Observables
  - (Observers “subscribe” to Observables)