Unit 4: Databases

Dave Abel

February 24th, 2016

Transition

- We’ve got our tools:
  - Binary (represent information)
  - Logic (manipulate information)
  - Programming (communicate with computers)
  - Algorithms (how to solve problems)

Transition

- What can we do with them?
  - Unit 4: Databases
  - Unit 5: Artificial Intelligence & Machine Learning
  - Unit 6: Computer Vision & Natural Language Processing
  - Unit 7: Theory of Computation
  - Unit 8: Compression and Error Correction
  - Unit 9: Recursion
  - Unit 10: Cryptography

Transition

- What can we do with them?
  - **Unit 4: Databases**
  - Unit 5: Artificial Intelligence & Machine Learning
  - Unit 6: Computer Vision & Natural Language Processing
  - Unit 7: Theory of Computation
  - Unit 8: Compression and Error Correction
  - Unit 9: Recursion
  - Unit 10: Cryptography

Takeaway

- A database is a collection of data!
- Critical to:
  - Banking
  - Information Retrieval
  - Commerce
  - Government
  - Artificial Intelligence

Outline

- Database example, Tables
  - Terminology
  - Transactions
    - ACID
- SQL
- Replicated Databases
- Two Way Databases
- Dictionaries

Database

- Most databases store information in a “table”
  - A *row* describes a single object
  - A *column* corresponds to a single attribute

Database: Example

- Attributes:
  - picture
  - prime?
  - sale price
  - ratings
  - name
  - etc.

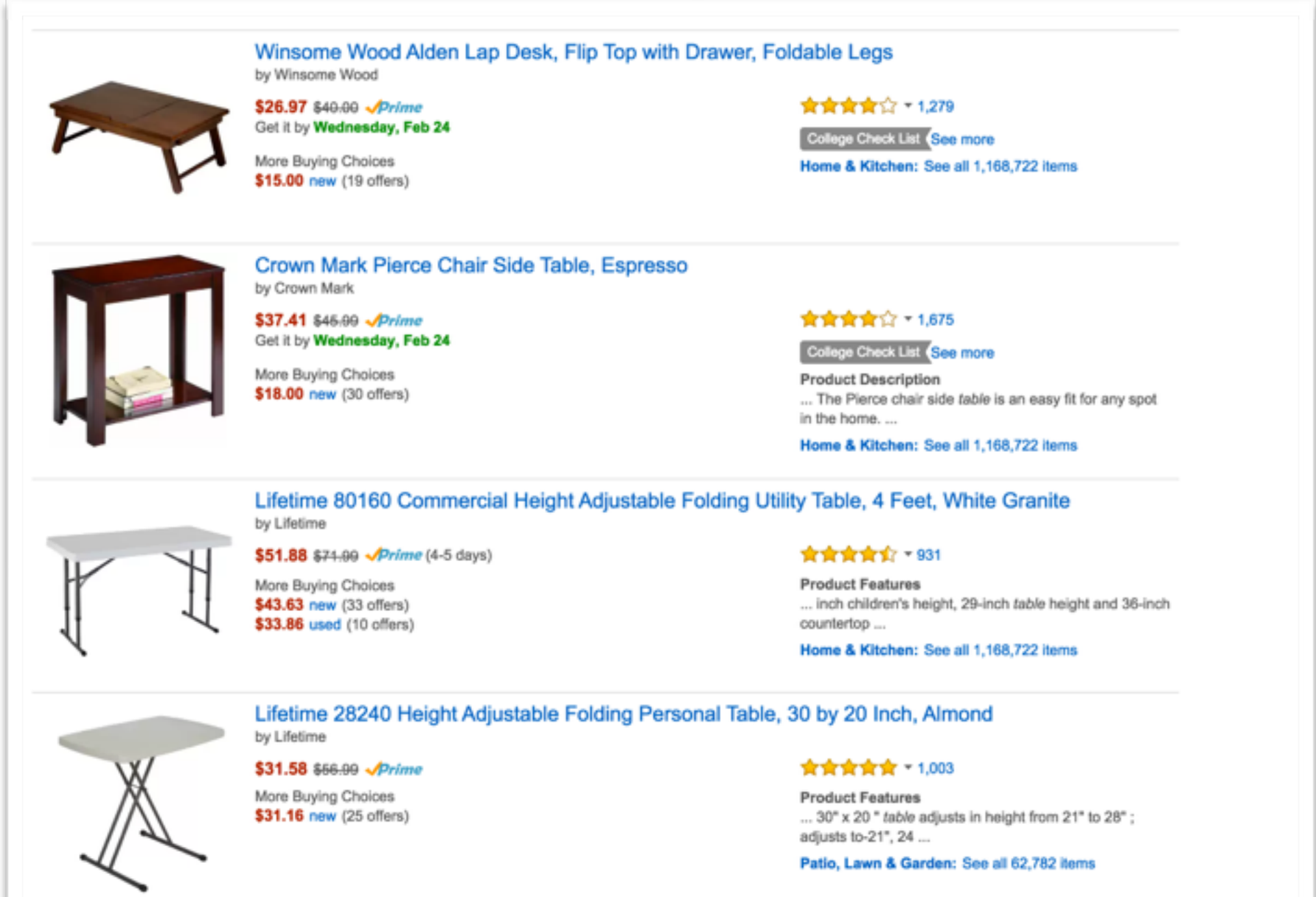


Table Format

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Table: Primary Keys

- Each entry in the database needs a *primary key*.
- **Definition:** a *primary key* is an attribute that is guaranteed to be unique for every item in the database.
- Lets us look each individual item up, if we want to.

Table: Primary Key

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Table: Primary Key

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Changing Tables

- How do we interact with our database?
- Modifications made through *transactions*, like:
  - Add a row (e.g. new item on Alabamazon)
  - Delete a row (e.g. item no longer available)
  - Change something about a row (e.g. new price)

Uh Oh!

- Suppose we make the following three transactions:

  1. All items over $50 added to Prime Shipping

  2. Change the price of item with Product ID 4273 to $47

  3. Make Product ID 4273 not have Prime

4273	Barkley's	FALSE	$53	463	4.5

Uh Oh!

- Suppose we make the following three transactions:

  1. All items over $50 added to Prime Shipping

  2. Change the price of item with Product ID 4273 to $47

  3. Make Product ID 4273 not have Prime

4273	Barkley's	FALSE	$53	463	4.5

Uh Oh!

- Suppose we make the following three transactions:

  1. All items over $50 added to Prime Shipping

  2. Change the price of item with Product ID 4273 to $47

  3. Make Product ID 4273 not have Prime

4273	Barkley's	**TRUE**	$53	463	4.5

Uh Oh!

- Suppose we make the following three transactions:

1. All items over $50 added to Prime Shipping

2. Change the price of item with Product ID 4273 to $47

3. Make Product ID 4273 not have Prime

4273	Barkley's	TRUE	$53	463	4.5

Uh Oh!

- Suppose we make the following three transactions:

1. All items over $50 added to Prime Shipping

2. Change the price of item with Product ID 4273 to $47

3. Make Product ID 4273 not have Prime

4273	Barkley's	TRUE	**$47**	463	4.5

Uh Oh!

- Suppose we make the following three transactions:

  1. All items over $50 added to Prime Shipping

  2. Change the price of item with Product ID 4273 to $47

  3. Make Product ID 4273 not have Prime

4273	Barkley's	TRUE	$47	463	4.5

Uh Oh!

- Suppose we make the following three transactions:

  1. All items over $50 added to Prime Shipping

  2. Change the price of item with Product ID 4273 to $47

  3. Make Product ID 4273 not have Prime

4273	Barkley's	**FALSE**	$47	463	4.5

Uh Oh!

- Suppose we make the following three transactions:

  1. All items over $50 added to Prime Shipping

  2. Change the price of item with Product ID 4273 to $47

3. Make Product ID 4273 not have Prime

4273	Barkley's	**FALSE**	$47	463	4.5

Q: Is this really what we wanted?

Clicker Question!

- Suppose we did the transactions in this order instead:

1. Make Product ID 4273 not have Prime

2. All items over $50 added to Prime Shipping

3. Change the price of item with Product ID 4273 to $47

4273	Barkley's	FALSE	$53	463	4.5

Q: Would Product ID 4723 have Prime shipping?

Clicker Question!

- Suppose we did the transactions in this order instead:

  1. Make Product ID 4273 not have Prime

  2. All items over $50 added to Prime Shipping

  3. Change the price of item with Product ID 4273 to $47

4273	Barkley's	FALSE	$53	463	4.5

Q: Would Product ID 4723 have Prime shipping?

[A] Yes! [B] No!

Clicker Answer!

- Suppose we did the transactions in this order instead:

  1. Make Product ID 4273 not have Prime

  2. All items over $50 added to Prime Shipping

  3. Change the price of item with Product ID 4273 to $47

4273	Barkley's	FALSE	$53	463	4.5

Q: Would Product ID 4723 have Prime shipping?

[A] **Yes!** [B] No!

Clicker Answer!

- Suppose we did the transactions in this order instead:

1. Make Product ID 4273 not have Prime
2. All items **over $50 added to Prime Shipping**
3. Change the price of item with Product ID 4273 to $47

4273	Barkley's	FALSE	$53	463	4.5

Q: Would Product ID 4723 have Prime shipping?

[A] **Yes!** [B] No!

Problem!

- The *order* in which transactions are received actually affects the outcome, a lot!
  - But databases crash, internet connection crashes…
  - And, it's really hard to guarantee that things happen in a certain order, without some extra effort.

Problem!

- The *order* in which transactions are received actually affects the outcome, a lot!
  - But databases crash, internet connection crashes…
  - And, it's really hard to guarantee that things happen in a certain order, without some extra effort.
- **Fix:** *write-ahead logging.* Before you do anything to do the database, write down the list of things you want to do in a safe place.
  - Critically, each transaction needs to be "idempotent", which just means that if you run it twice, it's going to do the same thing as if you ran it once.

Clicker Question!

Q: Which of these is not idempotent?

[A] All Prime items get a $5 discount

[B] Make Product ID 8123 cost $15

[C] Make all items below $15 eligible for Prime

[D] Any item made by “Barkley’s” is prime eligible

Definition: an idempotent transaction is one such that if you run it twice, it’s going to do the same thing as if you ran it once.

Clicker Answer!

Q: Which of these is not idempotent?

[A] All Prime items get a $5 discount

[B] Make Product ID 8123 cost $15

[C] Make all items below $15 eligible for Prime

[D] Any item made by “Barkley’s” is prime eligible

Definition: an idempotent transaction is one such that if you run it twice, it’s going to do the same thing as if you ran it once.

Clicker Answer!

Q: Which of these is not idempotent?

[A] All Prime items get a $5 discount

[B] Make Product ID 8123 cost $15

[C]

4273	Barkley's	TRUE	$47	463	4.5

Definition: an idempotent transaction is one such that if you run it twice, it's going to do the same thing as if you ran it once.

Clicker Answer!

Q: Which of these is not idempotent?

[A] All Prime items get a $5 discount

[B] Make Product ID 8123 cost $15

[C]

4273	Barkley's	TRUE	**$42**	463	4.5

Definition: an idempotent transaction is one such that if you run it twice, it's going to do the same thing as if you ran it once.

Clicker Answer!

Q: Which of these is not idempotent?

[A] All Prime items get a $5 discount

[B] Make Product ID 8123 cost $15

[C]

4273	Barkley's	TRUE	**$37**	463	4.5

Definition: an idempotent transaction is one such that if you run it twice, it's going to do the same thing as if you ran it once.

Problem!

- Databases Crash!
- **One Fix:** *write-ahead logging.* Before you do anything to do the database, write down the list of things you want to do in a safe place.
- Another idea: restrict the type of transactions that can be made so that crashes (and the issues) don't cause problems.

ACID

- A set of properties that ensure our transactions are processed reliably

  - **Atomicity**: Transactions are “all or nothing”. If one part of the transaction fails, the whole thing fails.

  - **Consistency**: you always leave the database in an okay state!

  - **Isolation**: If you do things in parallel, must result in a state that could have been achieved via non-parallel transactions.

  - **Durability**: Once the transaction is in, nothing can stop it! (even a crash, etc.)

Querying a Database

- Logic returns!
  - Come up with a question we want answered
  - Express it in computer-readable form
- Scratch: *imperative*, we tell it the strategy for answering the question
- Database queries: *declarative*, we tell it the question, it figures out the answer!

Database Queries: *SQL*

- *SQL:* the declarative language typically used for interacting with databases.

- Pronounced “sequel”.

- Flexibility level: broken crayon (it’s *really good* at coloring, but not much else).

Database Queries: *SQL*

- Examples:

```
SELECT company FROM alabamazon;
```

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Database Queries: *SQL*

- Examples:

```
SELECT company FROM alabamazon;
```

Company
Winsome Wood
Crown Mark
Lifetime
Lifetime

Database Queries: *SQL*

- Examples:

```
SELECT company FROM alabamazon where rating=“4”;
```

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Database Queries: *SQL*

- Examples:

```
SELECT company FROM alabamazon where rating=“4”;
```

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	**Winsome Wood**	**TRUE**	**26.97**	**1,279**	**4**
3249814	**Crown Mark**	**TRUE**	**37.41**	**1,675**	**4**
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Database Queries: *SQL*

- Examples:

```
SELECT company FROM alabamazon;
```

Company
Winsome Wood
Crown Mark

Database Queries: *SQL*

- Examples:

```
SELECT * FROM alabamazon;
```

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Database Queries: *SQL*

- Examples:

*** means “all”**

```
SELECT * FROM alabamazon;
```

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Database Queries: *SQL*

- Examples:

*** means “all”**

```
SELECT * FROM alabamazon where rating = 4;
```

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Database Queries: *SQL*

- Examples:

*** means “all”**

```
SELECT * FROM alabamazon where rating = 4;
```

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	**4**
3249814	Crown Mark	TRUE	37.41	1,675	**4**
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Database Queries: *SQL*

- Examples:

*** means “all”**

```
SELECT * FROM alabamazon where rating > 4;
```

4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4

Database Queries: *SQL*

- Examples: Logic!

```
SELECT * FROM alabamazon where rating > 4 AND
Prime = True;
```

```
SELECT * FROM alabamazon where
company="Barkley's" OR Prime = True;

SELECT * FROM alabamazon where
company="Barkley's" AND NOT(Prime = True);
```

Database Queries: *SQL*

- Examples: *aggregating functions*

```
SELECT count(*) FROM alabamazon where rating > 4;
```

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Database Queries: *SQL*

- Examples: *aggregating functions*

```
SELECT count(*) FROM alabamazon where rating > 4;
```

Reports: 2

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	**Lifetime**	**TRUE**	**51.88**	**931**	**4.5**
4224	**Lifetime**	**TRUE**	**31.58**	**1,003**	**5**

Clicker Question!

Suppose we have a data set of all Jeopardy questions, where the available attributes are the dollar amount of the question ($), the episode # it appeared on, and the answer to the question.

Q: Which of the following SQL queries will report the total amount of money given from questions since 1990?

[A]
```
SELECT count($) FROM jeopardy where NOT(year = 1990);
```

[B]
```
SELECT sum($) FROM jeopardy where NOT(year < 1990);
```

[C]
```
SELECT sum($) FROM jeopardy where NOT(year = 1990);
```

[D]
```
SELECT count($) FROM jeopardy where NOT(year < 1990);
```

Clicker Answer!

Suppose we have a data set of all Jeopardy questions, where the available attributes are the dollar amount of the question ($), the episode # it appeared on, and the answer to the question.

Q: Which of the following SQL queries will report the total amount of money given from questions since 1990?

[A] `SELECT count($) FROM jeopardy where NOT(year = 1990);`

[B] `SELECT sum($) FROM jeopardy where NOT(year < 1990);`

[C] `SELECT sum($) FROM jeopardy where NOT(year = 1990);`

[D] `SELECT count($) FROM jeopardy where NOT(year < 1990);`

Database Crash!

- Databases *do* crash…
- Q: What can we do to prevent bad things from happening when they do?
- A: Enter the “replicated database”

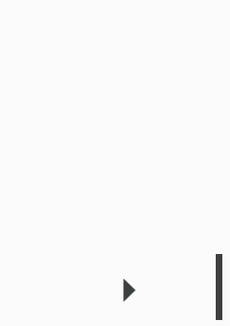
Replicated Database

- Idea: store the database more than once in other secure locations!

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Replicated Database

- Idea: store the database more than once in other secure locations!



Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Replicated Database

- Idea: store the database more than once in other secure locations!

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Replicated Database

- Idea: store the database more than once in other secure locations!

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

Replicated Database

- One Idea: only change both.

Add: “82374, Dave’s Chairs, FALSE, 450, 1, 5”

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

In Providence, RI

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5

In Portland, OR

Replicated Database

- One Idea: only change both.

Add: “82374, Dave’s Chairs, FALSE, 450, 1, 5”

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5
82734	Dave’s Chairs	FALSE	450	1	5

In Providence, RI

Product ID	Company	Prime	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	TRUE	26.97	1,279	4
3249814	Crown Mark	TRUE	37.41	1,675	4
1235561	Lifetime	TRUE	51.88	931	4.5
4224	Lifetime	TRUE	31.58	1,003	5
82734	Dave’s Chairs	FALSE	450	1	5

In Portland, OR

Replicated Database: The Good!

- A replicated database allows you to have a backup in the event that one of the databases is unavailable.

- If a replica restarts while it is being updated, it may be out of sync with the master database until it has time to repair itself.

- A replicated database can improve how quickly users querying the system far away from the master database receive responses to their queries.

- One of the reasons for storing replica databases geographically far from each other is to provide security against natural disasters like earthquakes destroying one of the databases.

Two Way Database

- Databases can often carry *redundant* information.
- A Two Way database is a way to consolidate.

Two Way Database

Product ID	Company	Company Ships to Canada	Price	# Ratings	Avg. Rating	Company Verified
4615253	Winsome Wood	TRUE	26.97	1,279	4	TRUE
3249814	Crown Mark	TRUE	37.41	1,675	4	TRUE
1235561	Lifetime	TRUE	51.88	931	4.5	TRUE
4224	Lifetime	TRUE	31.58	1,003	5	TRUE
129384	Dave’s Chairs	FALSE	450	1	5	FALSE
33332341	Dave’s Chairs	FALSE	27	948	5	FALSE
9723	Dave’s Chairs	FALSE	12	44	5	FALSE
761	Dave’s Chairs	FALSE	900	2	5	FALSE

Two Way Database

Product ID	Company	Company Ships to Canada	Price	# Ratings	Avg. Rating	Company Verified
4615253	Winsome Wood	TRUE	26.97	1,279	4	TRUE
3249814	Crown Mark	TRUE	37.41	1,675	4	TRUE
1235561	**Lifetime**	**TRUE**	51.88	931	4.5	**TRUE**
4224	**Lifetime**	**TRUE**	31.58	1,003	5	**TRUE**
129384	**Dave's Chairs**	**FALSE**	450	1	5	**FALSE**
33332341	**Dave's Chairs**	**FALSE**	27	948	5	**FALSE**
9723	**Dave's Chairs**	**FALSE**	12	44	5	**FALSE**
761	**Dave's Chairs**	**FALSE**	900	2	5	**FALSE**

Two Way Database

Product ID	Company	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	26.97	1,279	4
3249814	Crown Mark	37.41	1,675	4
1235561	Lifetime	51.88	931	4.5
4224	Lifetime	31.58	1,003	5
129384	Dave's Chairs	450	1	5
33332341	Dave's Chairs	27	948	5
9723	Dave's Chairs	12	44	5
761	Dave's Chairs	900	2	5

Company	Ships to Canada	Verified
Winsome Wood	TRUE	TRUE
Crown Mark	TRUE	TRUE
Lifetime	TRUE	TRUE
Dave's Chairs	FALSE	FALSE

Two Way Database

Product ID	Company	Price	# Ratings	Avg. Rating
4615253	Winsome Wood	26.97	1,279	4
3249814	Crown Mark	37.41	1,675	4
1235561	Lifetime	51.88	931	4.5
4224	Lifetime	31.58	1,003	5
129384	Dave's Chairs	450	1	5
33332341	Dave's Chairs	27	948	5
9723	Dave's Chairs	12	44	5
761	Dave's Chairs	900	2	5

Company	Ships to Canada	Verified
Winsome Wood	TRUE	TRUE
Crown Mark	TRUE	TRUE
Lifetime	TRUE	TRUE
Dave's Chairs	FALSE	FALSE

Same info, less storage!

Clicker Question!

JEOPARDY!

Episode #	Year	Month	Category	Answer
1	1981	January	CEREAL	Frosted Flakes
1	1981	January	PINK STARFISH	Patrick
1	1981	January	CEREAL	Raisin Brain
2	1981	February	KINDS OF WOOD	Oak
2	1981	February	KINDS OF WOOD	Birch
2	1981	February	PLACES IN RHODE ISLAND	Providence
2	1981	February	PLACES IN RHODE ISLAND	Pawtucket
3	2016	July	DAVE'S OFFICE NUMBER	321

Q: The above table has 40 cells. How many cells *combined* would our smallest two way table have?

Clicker Question!

Episode #	Year	Month	Category	Answer
1	1981	January	CEREAL	Frosted Flakes
1	1981	January	PINK STARFISH	Patrick
1	1981	January	CEREAL	Raisin Brain
2	1981	February	KINDS OF WOOD	Oak
2	1981	February	KINDS OF WOOD	Birch
2	1981	February	PLACES IN RHODE ISLAND	Providence
2	1981	February	PLACES IN RHODE ISLAND	Pawtucket
3	2016	July	DAVE'S OFFICE NUMBER	321

[A] 20

[B] 26

[C] 33

[D] 40

Q: The above table has 40 cells. How many cells *combined* would our smallest two way table have?

Clicker Answer!

JEOPARDY!

Episode #	Year	Month	Category	Answer
1	1981	January	CEREAL	Frosted Flakes
1	1981	January	PINK STARFISH	Patrick
1	1981	January	CEREAL	Raisin Brain
2	1981	February	KINDS OF WOOD	Oak
2	1981	February	KINDS OF WOOD	Birch
2	1981	February	PLACES IN RHODE ISLAND	Providence
2	1981	February	PLACES IN RHODE ISLAND	Pawtucket
3	2016	July	DAVE'S OFFICE NUMBER	321

[A] 20

[B] 26

[C] 33

[D] 40

Q: The above table has 40 cells. How many cells *combined* would our smallest two way table have?

Clicker Answer!

Episode #	Year	Month	Category	Answer
1	**1981**	**January**	CEREAL	Frosted Flakes
1	**1981**	**January**	PINK STARFISH	Patrick
1	**1981**	**January**	CEREAL	Raisin Brain
2	**1981**	**February**	KINDS OF WOOD	Oak
2	**1981**	**February**	KINDS OF WOOD	Birch
2	**1981**	**February**	PLACES IN RHODE ISLAND	Providence
2	**1981**	**February**	PLACES IN RHODE ISLAND	Pawtucket
3	**2016**	**July**	DAVE'S OFFICE NUMBER	321

[A] 20

[B] 26

[C] 33

[D] 40

Q: The above table has 40 cells. How many cells *combined* would our smallest two way table have?

Clicker Answer!

Episode #	Category	Answer
1	CEREAL	Frosted Flakes
1	PINK STARFISH	Patrick
1	CEREAL	Raisin Brain
2	KINDS OF WOOD	Oak
2	KINDS OF WOOD	Birch
2	PLACES IN RHODE ISLAND	Providence
2	PLACES IN RHODE ISLAND	Pawtucket
3	DAVE'S OFFICE NUMBER	321

3x8=24

Episode #	Year	Month
1	**1981**	**January**
2	**1981**	**February**
3	**2016**	**July**

3x3=9

[A] 20

[B] 26

[C] 33

[D] 40

Q: The above table has 40 cells. How many cells *combined* would our smallest two way table have?

Clicker Answer!

JEOPARDY!

Episode #	Category	Answer
1	CEREAL	Frosted Flakes
1	PINK STARFISH	Patrick
1	CEREAL	Raisin Brain
2	KINDS OF WOOD	Oak
2	KINDS OF WOOD	Birch
2	PLACES IN RHODE ISLAND	Providence
2	PLACES IN RHODE ISLAND	Pawtucket
3	DAVE'S OFFICE NUMBER	321

3x8=24

Episode #	Year	Month
1	**1981**	**January**
2	**1981**	**February**
3	**2016**	**July**

3x3=9

9 + 24 = 33

[A] 20

[B] 26

[C] 33

[D] 40

Q: The above table has 40 cells. How many cells *combined* would our smallest two way table have?

Dictionaries

Word	Definition
Cottage	A small house, especially in a countryside
Hummus	A delicious sauce made from Garbanzo Beans
Brackish	(of water) A mixture of river water and sea water

Dictionaries

Word	Definition
Cottage	A small house, especially in a countryside
Hummus	A delicious sauce made from Garbanzo Beans
Brackish	(of water) A mixture of river water and sea water

Definition: a *dictionary* is a database with one primary key, and only one other column.

Dictionaries

Word	Definition
Cottage	A small house, especially in a countryside
Hummus	A delicious sauce made from Garbanzo Beans
Brackish	(of water) A mixture of river water and sea water

Definition: a *dictionary* is a database with one primary key, and only one other column.

Definition: a *dictionary* is a list where you index into the list with an arbitrary object.

Dictionaries: List!

Index	Definition
1	Apple
2	Coconut
3	Banana

Definition: a *dictionary* is a database with one primary key, and only one other column.

Definition: a *dictionary* is a list where you index into the list with an arbitrary object.

Dictionaries

Word	Definition
Cottage	A small house, especially in a countryside
Hummus	A delicious sauce made from Garbanzo Beans
Brackish	(of water) A mixture of river water and sea water

Useful any time we want to associate one piece of information with one other piece of information

Dictionaries: Counting Words!

Word	Number of Occurrences
Cottage	1
Hummus	0
Brackish	0

Could count the number of times words appear in Charlie and the Chocolate Factory…

Dictionaries: Jeopardy Questions Correct

Word	Number of Correct Questions
Sam	1506
Anushka	2234
Tori	1187

Dictionaries

KEY	VALUE
Sam	1506
Anushka	2234
Tori	1187

In general: the “word” is called the *key*, the “definition” is called the *value*

Dictionaries

KEY	VALUE
Sam	1506
Anushka	2234
Tori	1187

"Key" Idea: we need to turn the *key* into something numeric so that it corresponds uniquely to one row in the table.

Dictionaries

KEY	VALUE
Sam	1506
Anushka	2234
Tori	1187

"Key" Idea: we need to turn the *key* into something numeric so that it corresponds **uniquely** to one row in the table.

Any thoughts?

Dictionaries

One idea:

0	1	2	3
a	b	c	d

"Key" Idea: we need to turn the *key* into something numeric so that it corresponds **uniquely** to one row in the table.

Any thoughts?

Dictionaries

One idea:

0	1	2	3
a	b	c	d

We can basically do this for any data type, as we saw in unit 1!

So we can get a “dictionary” where any data type is the key!

Dictionaries

KEY (sound)	***VALUE***
“Siri, what is the weather like?”	<Query the weather>
“Siri, what is the date?”	<Query the date>
“Siri, what is your name?”	<Query response about my name>

With a big enough table…

We can basically do this for any data type, as we saw in unit 1!

So we can get a “dictionary” where any data type is the key!

Dictionaries: Python!

Dictionaries: Scratch!

Let's take a look!