Database Design

Processing Speeds

Nanoseconds – within CPU, cache memory

Microseconds – SSD

Milliseconds – spinning disk, HDD

Per Jim Pappas of Intel, a fellow member of SNIA (the Storage Networking Industry Association)

- The difference between the speed of system memory and that of a hard disk drive (HDD) is roughly 6 orders of magnitude, or 1 million times
- SSDs split the gap. An SSD is about 1,000 times faster than an HDD, and is about 1,000 times slower than system memory. Memory access times are measured in nanoseconds (ns), SSDs in microseconds (μs) and HDDs in milliseconds (ms)

Database Design

System Event	Actual Latency	Scaled Latency
One CPU cycle	0.4 ns	**1 s**
Level 1 cache access	0.9 ns	**2 s**
Level 2 cache access	2.8 ns	**7 s**
Level 3 cache access	28 ns	**1 min**
Main memory access (DDR DIMM)	~100 ns	**4 min**
Intel® Optane™ DC persistent memory access	~350 ns	**15 min**
Intel® Optane™ DC SSD I/O	<10 μs	**7 hrs**
NVMe SSD I/O	~25 μs	**17 hrs**
SSD I/O	50–150 μs	**1.5–4 days**
Rotational disk I/O	1–10 ms	**1–9 months**
Internet call: San Francisco to New York City	65 ms[3]	**5 years**
Internet call: San Francisco to Hong Kong	141 ms[3]	**11 years**

https://www.prowesscorp.com/computer-latency-at-a-human-scale/

The problem: To most of us these measurements are all so tiny that they appear to be about the same.

To put things in a human perspective let's slow everything down by about 1 billion times:

- System memory access = a human heartbeat… Every heartbeat you get another chunk of data (block, page)
- An SSD access = the time it takes you to walk a mile
- An HDD access = the time that it would take you to bicycle from San Francisco to Miami

"No wonder that everyone's excited about SSDs. Now you can walk to the neighborhood store for a carton of milk rather than bicycle across an entire continent. …. The result is that you can get a lot more work done significantly faster." [Jim Pappas]

Recap of stuff so far…

- Why use databases?
- Why RELATIONAL?
- Terms.

Let me illustrate.

1. The notebook. ←
2. The bag.

1. MARGARET MARY PARADISE
2. 9728 GRANTVIEW FOREST DR.
3. ST. LOUIS, MO, 63123
4. 314-842-2556 (H)
5. 314-477-5294 (C)
6. 08-27-1990
7. FINE ART
8. COMMUNICATIONS
9. 5195 RALEIGH ST.
10. DENVER, CO, 80212
11. LAURA BIRA
12. 3.6
13. 3.7
14. 801243577
15. 586-42-3896

Sequential File

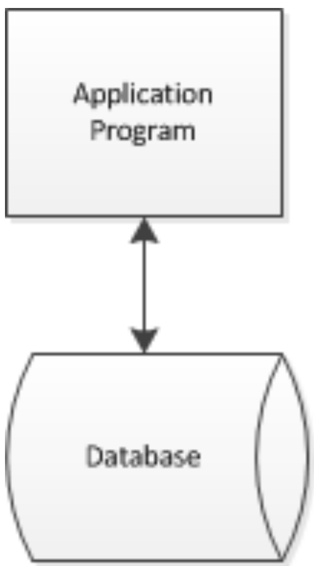
1. We can store up to 500 characters per page
2. You can only read from front to back, one page at a time
3. Pages are in ascending Lastname, Firstname order

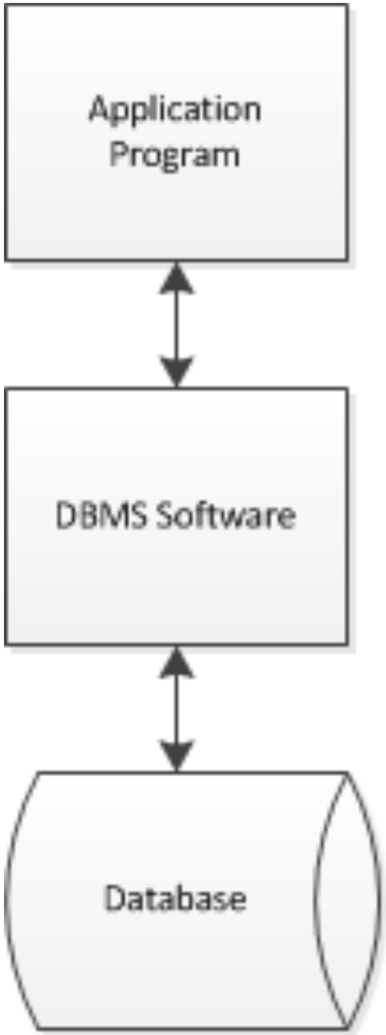
Indexed File

1. Pages are indexed by first letter of the last name
2. Index tabs are in alphabetical order
3. Within names beginning with one letter, you can flip half-way (binary search)

With either sequential or indexed files

1. Your application program MUST contain a field-by-field record definition
2. If the record structure changes, you MUST change your application code





Adds a layer of separation between the application code and the files, records, fields, pointer, chains.

Revolutionized the software industry

Shifting Gears:

From ***Introductory Concepts*** to ***Design Steps***

Or,

“Now that we’ve learned a bit about relational databases, their history, and conceptual principles, let’s look into how one goes about designing a database for an organization.”

Design

Design Steps:

Collect all the information you can about the organization's data (i.e. "requirements")

- What do they call their entities and attributes?
- Data types and lengths
- How and when do they use each piece of data?
- Where does it come from? Where does it go to?
- Who uses it for what purpose?
- How many occurrences/instances of each entity do they deal with?

How?

1. Conduct interviews
2. Review documentation
3. Review current systems and processes
   - User data entry screens
   - Paper forms
   - Paper reports
   - Computer reports
   - Process flows

Design Steps:

Once data requirements are collected and documented, organize the data:

- Identify all **Functional Dependencies**
- Put the data into **3rd Normal Form**
- Create a logical **data model**

Functional Dependency

Definition: (Ullman text, page 68)

"If the values of one or more attributes (A1, A2, A3, etc.) of a relation functionally determine the value of another attribute (B) of that relation, then we can say that (B) is functionally dependent on (A1, A2, A3, etc.)"

In other words,

If I know the value of an attribute (or set of attributes),
I can determine the value of another attribute.

Let me illustrate.

1. The notebook.

2. The bag. 🡐

Examples: (from the text)

The "Movies1" relation

title	*year*	*length*	*genre*	*studioName*	*starName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

Functional Dependency

Follows Codd's relational rules, but it is bad design.

Why?

Let's look at functional dependencies.

Functional Dependency

title	*year*	*length*	*genre*	*studioName*	*starName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

FD = title + year ➔ length + genre + studioName

If 2 tuples have the same values in title + year,
those 2 tuples will have the same values in length, genre and studioName

TRUE

Functional Dependency

title	*year*	*length*	*genre*	*studioName*	*starName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

FD = title + year ➔ starName

If 2 tuples have the same values in title + year,
those 2 tuples will have the same values in starName

FALSE

Functional Dependency

A formal definition of a Key

1. "A set of attributes (A1, A2, A3,etc.) is a KEY for a relation if that set of attributes functionally determines all the other attributes of the relation.

And

2. No subset of those attributes (A1, A2, A3,etc.) functionally determines all the other attributes of the relation."

(FD doesn't differentiate Primary Key versus Candidate Key)

Functional Dependency

Apply some FD tests to this relation.

title	*year*	*length*	*genre*	*studioName*	*starName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

title
title+year
title+starName
title+year+starName

- *The Manchurian Candidate* (1962 film), starring Frank Sinatra
- *The Manchurian Candidate* (2004 film), starring Denzel Washington
- *Blade Runner (1982 film) staring Harrison Ford*
- *Blade Runner (2017 film) staring Harrison Ford*

Let's practice

A relation of people in the U.S.

name, SSN, address, city, state, ZIP, area code, 7-digit phone

what FD's are TRUE?

Functional Dependency

More practice

Project Code	Project Title	Project Manager	Project Budget	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	Pensions System	M Phillips	24500	S10001	A Smith	L004	IT	22.00
PC010	Pensions System	M Phillips	24500	S10030	L Jones	L023	Pensions	18.50
PC010	Pensions System	M Phillips	24500	S21010	P Lewis	L004	IT	21.00
PC045	Salaries System	H Martin	17400	S10010	B Jones	L004	IT	21.75
PC045	Salaries System	H Martin	17400	S10001	A Smith	L004	IT	18.00
PC045	Salaries System	H Martin	17400	S31002	T Gilbert	L028	Database	25.50
PC045	Salaries System	H Martin	17400	S13210	W Richards	L008	Salary	17.00
PC064	HR System	K Lewis	12250	S31002	T Gilbert	L028	Database	23.25
PC064	HR System	K Lewis	12250	S21010	P Lewis	L004	IT	17.50
PC064	HR System	K Lewis	12250	S10034	B James	L009	HR	16.50

Why do we care about all this?

- If I am designing a database for an organization, I must consider all their data. I must organize that data into relations.
- I need to understand **Functional Dependencies** so I can determine **keys** for every relation.
- I need **keys** so that I can **normalize** the data.
- I must normalize the data to design the database properly.

Some FD rules

If I know a true FD for a relation, I can deduce other true FD’s for that relation.

Transitive Rule

In a relation R(A, B, C)

if A ➔ B, and B ➔ C, then A➔C

Example:

Project Code	Project Title	Project Manager	Project Budget	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	Pensions System	M Phillips	24500	S10001	A Smith	L004	IT	22.00
PC010	Pensions System	M Phillips	24500	S10030	L Jones	L023	Pensions	18.50
PC010	Pensions System	M Phillips	24500	S21010	P Lewis	L004	IT	21.00
PC045	Salaries System	H Martin	17400	S10010	B Jones	L004	IT	21.75
PC045	Salaries System	H Martin	17400	S10001	A Smith	L004	IT	18.00
PC045	Salaries System	H Martin	17400	S31002	T Gilbert	L028	Database	25.50
PC045	Salaries System	H Martin	17400	S13210	W Richards	L008	Salary	17.00
PC064	HR System	K Lewis	12250	S31002	T Gilbert	L028	Database	23.25
PC064	HR System	K Lewis	12250	S21010	P Lewis	L004	IT	17.50
PC064	HR System	K Lewis	12250	S10034	B James	L009	HR	16.50

EmployeeNo ➔ EmployeeName

EmployeeName ➔ DepartmentNo, therefore

EmployeeNo ➔ DepartmentNo

Augmentation Rule

In a relation R(A, B, C)

if A ➔ C, then A+B ➔ C+B

Example:

Project Code	Project Title	Project Manager	Project Budget	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	Pensions System	M Phillips	24500	S10001	A Smith	L004	IT	22.00
PC010	Pensions System	M Phillips	24500	S10030	L Jones	L023	Pensions	18.50
PC010	Pensions System	M Phillips	24500	S21010	P Lewis	L004	IT	21.00
PC045	Salaries System	H Martin	17400	S10010	B Jones	L004	IT	21.75
PC045	Salaries System	H Martin	17400	S10001	A Smith	L004	IT	18.00
PC045	Salaries System	H Martin	17400	S31002	T Gilbert	L028	Database	25.50
PC045	Salaries System	H Martin	17400	S13210	W Richards	L008	Salary	17.00
PC064	HR System	K Lewis	12250	S31002	T Gilbert	L028	Database	23.25
PC064	HR System	K Lewis	12250	S21010	P Lewis	L004	IT	17.50
PC064	HR System	K Lewis	12250	S10034	B James	L009	HR	16.50

EmployeeNo + DepartmentNo ➔ EmployeeName + DepartmentNo

Functional Dependency

Combining + Splitting Rule

In a relation R(A, B, C, D)
if A+B ➔ C, and A+B ➔ D, then A+B ➔ C, D

title	*year*	*length*	*genre*	*studioName*	*starName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

title+year ➔ length
title+year ➔ genre
title+year ➔ studioName

title+year ➔ length, genre, studioName (combine the right side)

Combining + Splitting Rule

title	*year*	*length*	*genre*	*studioName*	*starName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

But these are false: (can't split the left side)

title ➔ length

year ➔ length

Closure

For a relation, we seek ALL TRUE functional dependency rules

A relation: R(A, B, C, G, H, I)

FD's = $A \rightarrow B$
$A \rightarrow C$
$C+G \rightarrow H$
$C+G \rightarrow I$
$B \rightarrow H$

Closure

1. A→ H (transitive)
2. C+G→ H,I (combining)
3. A+G→ I (augmentation)
   - A→C
   - A+G→C+G
   - C+G→I

FD's = A→B
A→C
C+G→H
C+G→I
B→ H

So, the closure (full set of FD's) is all of these.