Database Design

Presented for Girl Develop It

Girl Develop It is here to provide affordable and accessible programs to learn software through mentorship and hands-on instruction.

Intros

Sondra Willhite
software developer at scrubjay technology

- over ten years working on applications with SQL Server or Microsoft Access backend
- also have experience working as a BI analyst and a brief stint working help desk / systems admin

https://www.linkedin.com/in/sondrawillhite

Agenda

- careers in database
- overview of some database models
- database management systems
- the relational database model
  - structure
  - keys
  - referential integrity
  - normalization

Careers in Database

Database Careers

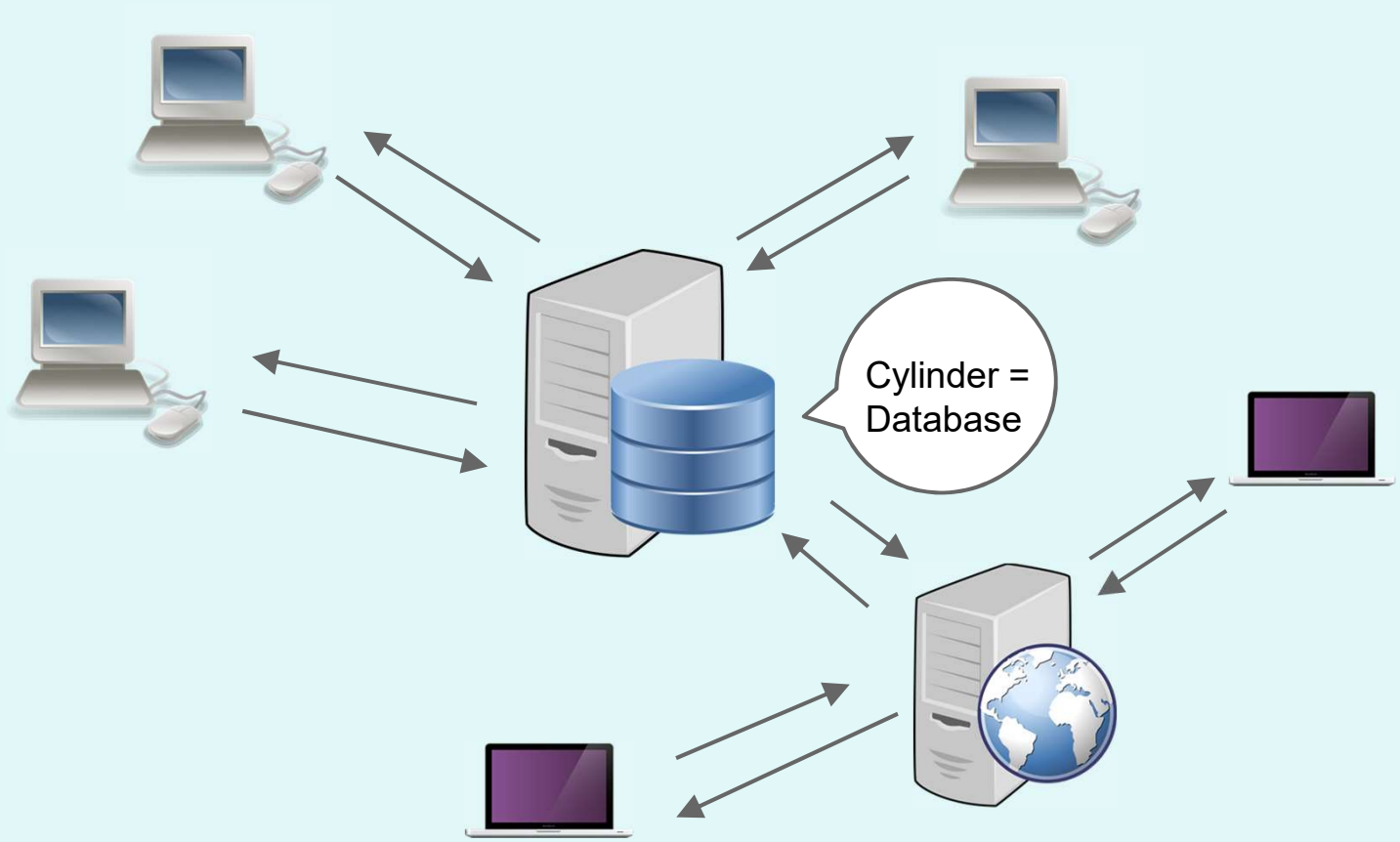
- database administrator (DBA)
- database developer
- business intelligence analyst
- database consultant
- data scientist

Database Administrator

a systems administrator for database

- backups and restores
- database availability
- partitioning
- security
- performance

Database Administrator



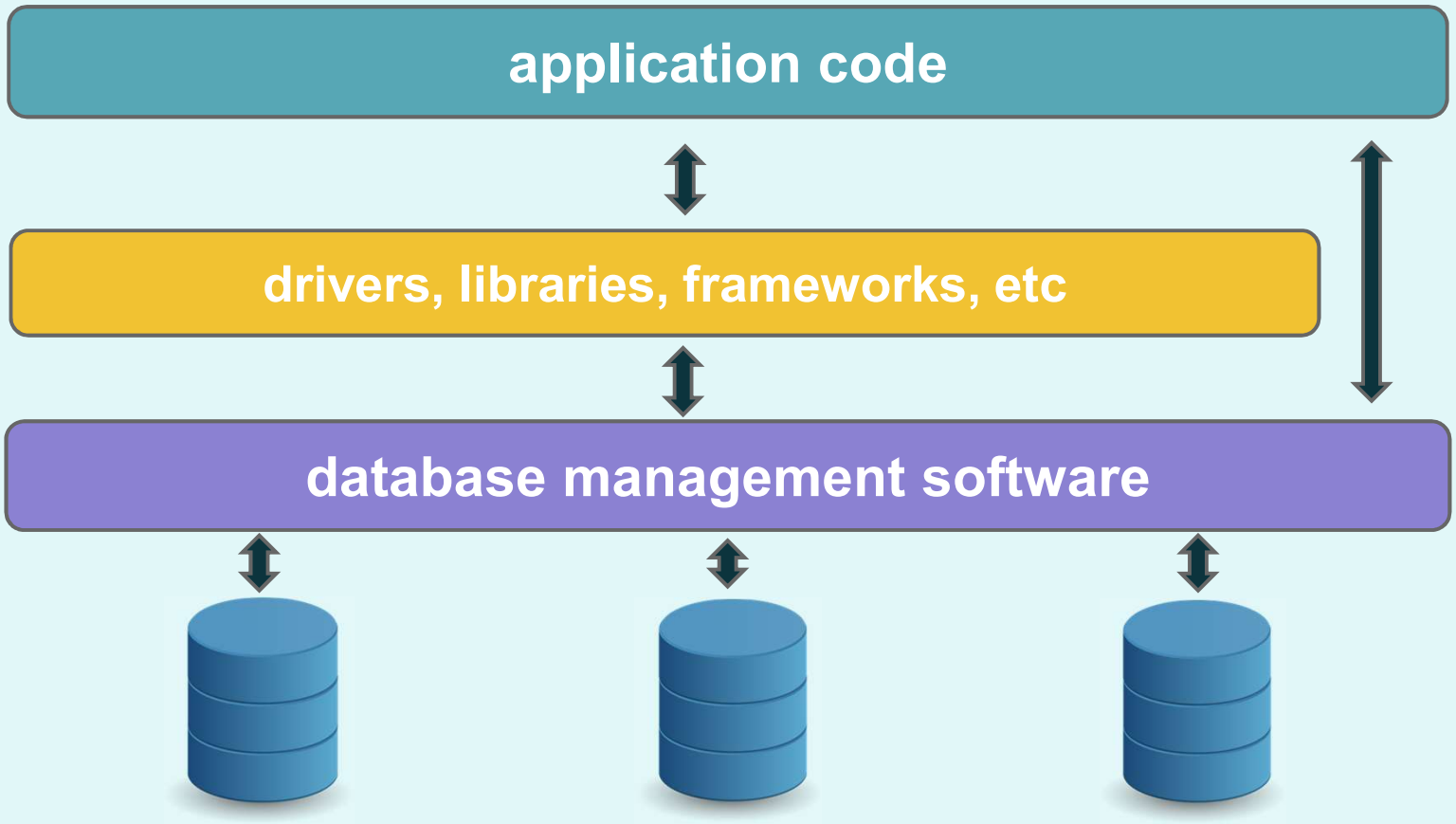
Network Diagram

Database Developer

software developers who specialize in database

- query database
- script schema changes
- stored procedures
- triggers and constraints
- database design
- performance and security

Database Developer



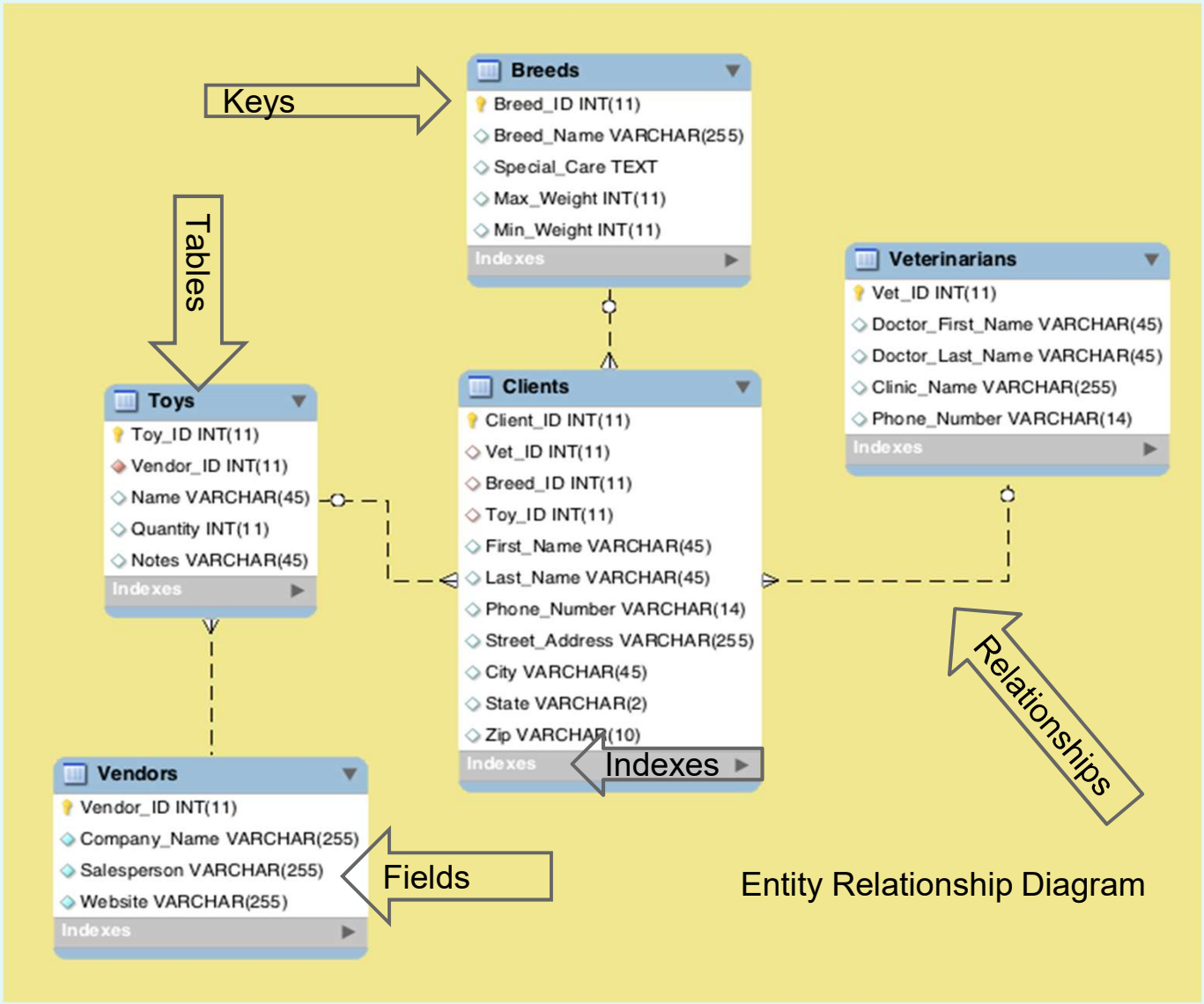
Application Stack Diagram

BI Analyst

writes custom queries and reports, including data visualizations

- aka report writer/business analyst
- SQL language experts
- platforms
  - SSRS (SQL Server Reporting Services)
  - Crystal Reports
  - Tableau / Power BI
  - Microsoft Access
- performance and security

BI Analyst


Keys
Tables
Fields
Indexes
Relationships
Breeds
Breed_ID INT(11)
Breed_Name VARCHAR(255)
Special_Care TEXT
Max_Weight INT(11)
Min_Weight INT(11)
Indexes
Veterinarians
Vet_ID INT(11)
Doctor_First_Name VARCHAR(45)
Doctor_Last_Name VARCHAR(45)
Clinic_Name VARCHAR(255)
Phone_Number VARCHAR(14)
Indexes
Toys
Toy_ID INT(11)
Vendor_ID INT(11)
Name VARCHAR(45)
Quantity INT(11)
Notes VARCHAR(45)
Indexes
Clients
Client_ID INT(11)
Vet_ID INT(11)
Breed_ID INT(11)
Toy_ID INT(11)
First_Name VARCHAR(45)
Last_Name VARCHAR(45)
Phone_Number VARCHAR(14)
Street_Address VARCHAR(255)
City VARCHAR(45)
State VARCHAR(2)
Zip VARCHAR(10)
Indexes
Vendors
Vendor_ID INT(11)
Company_Name VARCHAR(255)
Salesperson VARCHAR(255)
Website VARCHAR(255)
Indexes
Entity Relationship Diagram

Database Consultant

specialist DBAs/developers

- performance tuning
  - finding the bottlenecks
- security controls
  - groups and user controls
  - encryption (database, columns, rows)
- availability
  - mirroring and fail-over clusters
  - cloud systems (Azure, AWS)

Data Scientist

specialize in creating data models

- especially for predictive modeling
- mathematical background in statistics
- look beyond relational database models:
  - **big data**
  - **graph data**
  - **warehouse data**

Some Database Models

Database Definition

an unordered structured set of data

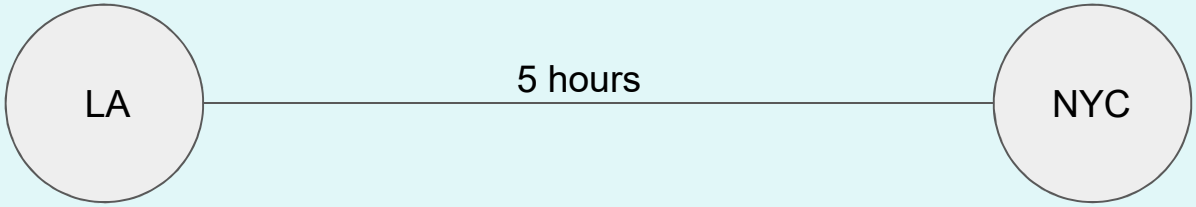
Big Data

key : value pairs

{ vendor : abc manufacturing }

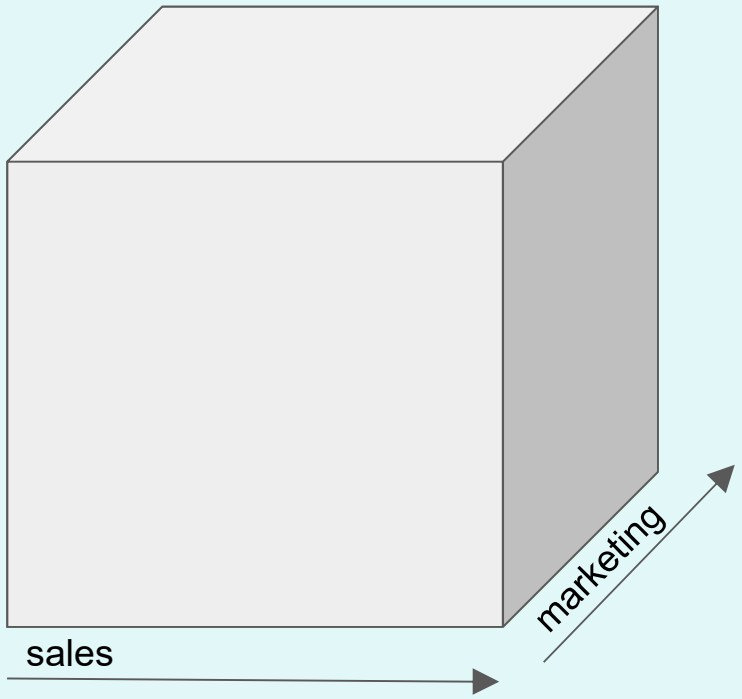
Graph Data

nodes and edges



Warehouse Data

data structured along dimensions



Relational Data

tables and fields

Customer ID	First	Last
100011	Jane	Doe

Customer ID	Item	Cost
100011	Intro Database	$25
100011	Data Modeling	$30

Database Management Systems

Database Management Systems

software that provides ways to
store, modify and retrieve data

- Microsoft SQL Server
- Oracle
- IBM's DB2
- PostgreSQL
- MySQL
- Microsoft Access

Database Management Systems

DBMS responsibilities

- data integrity
- data consistency
- multi-user access
- performance tools
- security and auditing
- backup and recovery
- extraction, transformation and load (ETL)
- business intelligence tools

Data Integrity

ensuring "valid" data

- **data types** ensure that, say, date fields only store date values
- **constraints** and **triggers** allow for complex rules to be applied (eg, you cannot delete a client who has an upcoming appointment)

Data Consistency

ensuring that database is in a "valid" state

- **record locks** prevent "dirty reads/writes"
- **commit** and **rollback** mechanisms ensure transactions are either fully completed or fully rolled back

Performance

- fine-grained record locking prevents queries from blocking others
- **indexes** speed up lookups and joins by magnitudes
- **query optimizers** find the fastest way to execute your query

Backup and Recovery

- tools to **backup** and **restore** to a point in time
  - **log files** make this possible
- database **mirroring** and **fail-over clustering**

Security and Auditing

- **multi-tiered security** (server level, database level, column level; role-level and user-level)
- **logs** can be queried for auditing (not directly)
- tied in to data integrity

ACID

another acronym to describe data integrity and data consistency concepts in relational databases

Atomicity
Consistency
Isolation
Durability

Atomicity

a transaction must be all or nothing

Consistency

invalid data causes transaction to roll back

Isolation

transactions are processed independently of other transactions

Durability

once committed a transaction is permanent, even in the event of a system failure (eg, power outage)

ACID

example: consider the transaction of moving $100 from your checking to your savings account.

steps:

1. confirm accounts valid
2. confirm checking account has available funds
3. debit checking account by $100
4. credit savings by $100

ACID

Case 1: building loses power between step 3 and 4

because the transaction was not fully completed,
atomicity ensures that the transaction is rolled back.

ACID

Case 2: memory corruption causes step 4 to credit checking by $100,000,000.

because the transaction leaves the system is an inconsistent state, **consistency** will ensure that the transaction is rolled back (sorry!)

ACID

Case 3: at the same time that you transfer funds, your utility cashes a check that brings your checking balance to $50

isolation ensures that either the utility clears before your transaction (meaning your transaction will be rejected) or your transaction finishes first (meaning the utility check will bounce)

ACID

Case 4: building loses power right after your transaction completes

durability guarantees that the transaction is permanent

Relational Database Model

Relational Data Model

data model invented by Edgar Codd in 1970

- data is stored in tables and fields
- a set of rules, the **normal forms**, ensures that the **universe of data** will be preserved
- data can be read and updated using SQL
  - Structured Query Language
  - all DBMS's recognize SQL, but some small differences exist between them

Relational Data Model

still has the lion's share of the database market

- ACID makes it reliable for critical transactional systems
- is an all-purpose database
- older models - hierarchical, network - did not always preserve the universe of data
- newer models are speciality models - they typically solve one problem but come at a high cost in other areas

Relational Data Model

two meanings of the "relation" in relational model

- data points “related” to each other are stored in a table
- tables are “related” to each other by special fields (keys)

Relational Data Model

storing the data as relations

- eliminates redundancy
  - saves space
  - reduces mistakes (ties in to consistency)

Redundancy in Excel

duplicating data not only wastes space but is error prone

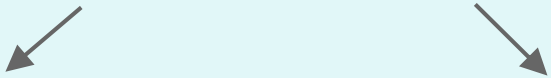
appointments.xlsx

Client	Phone	Addr	Service	Appt Date
Anna	215-123-4567	123 City Lane	Nails	5/1/2013
Nathan	267-333-4444	999 Oak Blvd	Hair	7/5/2013
Anna	215-123-4576	123 Mock Ln.	Hair	9/1/2013

Redundancy in Excel

appointments.xlsx

Client	Phone	Addr	Service	Appt Date
Anna	215-123-4567	123 City Lane	Nails	5/1/2013
Nathan	267-333-4444	999 Oak Blvd	Hair	7/5/2013
Anna	215-123-4576	123 Mock Ln.	Hair	9/1/2013



clients (table)

Name	Phone	Address
Anna	215-123-4567	123 City Lane
Nathan	267-333-4444	999 Oak Blvd

appointments (table)

Client	Service	Date
Anna	Nails	5/1/2013
Nathan	Hair	7/5/2013
Anna	Hair	9/1/2013

Relational Data Model

storing the data as relations

- eliminates redundancy
  - saves space
  - reduces mistakes (ties in to consistency)
- guarantees data completeness (the **universe of data** is preserved)

The Universe of Data

just means the set of data that we're modeling

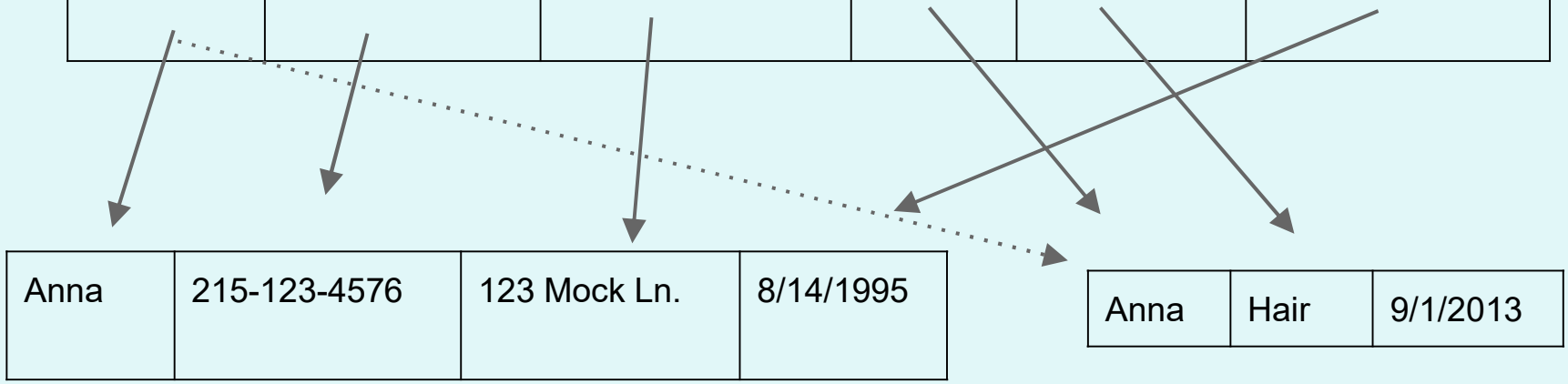
Client	Phone	Addr	Service	Appt Date	DOB
Anna	215-123-4567	123 City Lane	Nails	5/1/2013	8/14/1995
Nathan	267-333-4444	999 Oak Blvd	Hair	7/5/2013	6/1/1998
Anna	215-123-4576	123 Mock Ln.	Hair	9/1/2013	8/14/1995

in our example, the data points above is our "universe of data"

Decomposition

the process of splitting data into tables

Client	Phone	Addr	Service	Appt Date	DOB
Anna	215-123-4567	123 City Lane	Nails	5/1/2013	8/14/1995
Nathan	267-333-4444	999 Oak Blvd	Hair	7/5/2013	6/1/1998



Anna	215-123-4576	123 Mock Ln.	8/14/1995

Anna	Hair	9/1/2013

Decomposition

if we follow the rules set forth by Edgar Codd in the **normal forms** when decomposing our data into tables, then we are guaranteed that we'll be able to reconstruct our universe of data using SQL

Decomposition

our spreadsheet decomposed into two tables

Name	Phone	Address	DOB
Anna	215-123-4567	123 City Lane	8/14/1995
Nathan	267-333-4444	999 Oak Blvd	6/1/1998

Name	Service	Date
Anna	Nails	5/1/2013
Nathan	Hair	7/5/2013
Anna	Hair	9/1/2013

Decomposition

decomposing data means we need a mechanism to put it back together again

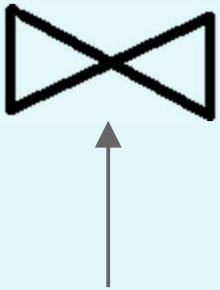
Name	Service	Date
Anna	Nails	5/1/2013
Nathan	Hair	7/5/2013
Anna	Hair	9/1/2013

Name	Phone	Address	DOB
Anna	215-123-4567	123 City Lane	8/14/1995
Nathan	267-333-4444	999 Oak Blvd	6/1/1998

Joins

in database we put back our universe by **joining** tables

Name	Phone	Address	DOB
Anna	215-123-4567	123 City Lane	8/14/1995
Nathan	267-333-4444	999 Oak Blvd	6/1/1998



Bowtie = Join

Service	Date
Nails	5/1/2013
Hair	7/5/2013
Hair	9/1/2013

Joins

every row in table C is matched to a row in table A on some field - this special field is called a **key**

Clients (C)

Name	Phone	Address	DOB
Anna	215-123-4567	123 City Lane	8/14/1995
Nathan	267-333-4444	999 Oak Blvd	6/1/1998

Appointments (A)

Service	Date
Nails	5/1/2013
Hair	7/5/2013
Hair	9/1/2013

Joins

What is C ⋈ A on DOB = Appt Date?

Clients

Name	Phone	Address	DOB
Anna	215-123-4567	123 City Lane	8/14/1995
Nathan	267-333-4444	999 Oak Blvd	6/1/1998

Appointments

Service	Appt Date
Nails	5/1/2013
Hair	7/5/2013
Hair	9/1/2013

Joins

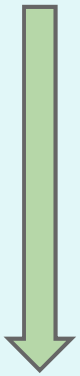
What is C ⋈ A on DOB = Appt Date ?

Clients

Name	Phone	Address	DOB
Anna	215-123-4567	123 City Lane	8/14/1995
Nathan	267-333-4444	999 Oak Blvd	6/1/1998

Appointments

Service	Appt Date
Nails	5/1/2013
Hair	7/5/2013
Hair	9/1/2013



first iteration: find every appointment with an Appt Date of 8/14/1995

Joins

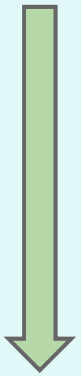
what is C ⋈ A on DOB = Appt Date ?

Clients

Name	Phone	Address	DOB
Anna	215-123-4567	123 City Lane	8/14/1995
Nathan	267-333-4444	999 Oak Blvd	6/1/1998

Appointments

Service	Appt Date
Nails	5/1/2013
Hair	7/5/2013
Hair	9/1/2013



second iteration: find every appointment with an Appt Date of 6/1/1998

Joins

Clients

Name	Phone	Address	DOB
Anna	215-123-4567	123 City Lane	8/14/1995
Nathan	267-333-4444	999 Oak Blvd	6/1/1998

Appointments

Service	Appt Date
Nails	5/1/2013
Hair	7/5/2013
Hair	9/1/2013

what is C ⋈ A on DOB = Appt Date? ∅

Service	Appt Date	Name	Phone	Addr	DOB

Joins

keys are not arbitrary - during decomposition, we always copy a special field to each table to serve as the key

Clients

Name	Phone	Address	DOB
Anna	215-123-4567	123 City Lane	8/14/1995
Nathan	267-333-4444	999 Oak Blvd	6/1/1998

Appointments

Name	Service	Appt Date
Anna	Nails	5/1/2013
Nathan	Hair	7/5/2013
Anna	Hair	9/1/2013

Joins

the universe of data is reassembled by **joining** tables.

Name	Phone	Addr	DOB	Name	Service	Appt Date
Anna	215-123-4567	123 City Lane	8/14/1995	Anna	Nails	5/1/2013
Nathan	267-333-4444	999 Oak Blvd	6/1/1998	Nathan	Hair	7/5/2013
Anna	215-123-4567	123 City Lane	8/14/1995	Anna	Hair	9/1/2013

C ⋈ A on Name = Name

Primary and Foreign Keys

Keys

tables are joined on special fields called keys

- these fields are known as the **Primary Key** and the **Foreign Key**
- picking the Name as the field that relates Clients to Appointments was intuitive
- there's rules on how to identify them

Primary Key

a field (or set of fields) that uniquely identify a row

- ➢ the minimal set of fields that the row is **functionally dependent** upon

Relational Algebra

the branch of mathematics providing the foundation for the relational database model

Functional Dependencies

let A and B be sets of fields in a table,
a **functional dependency** exists
A --> B
if for every row[A] in the table, we get back row[B]

Functional Dependencies

SSN	First Name	Last Name
111-11-1111	Anna	Jones
222-22-2222	Nathan	Smith
111-11-1111	?	?

let A = (SSN), B = (First Name, Last Name)

can we can say A --> B?

Functional Dependencies

Name	Service	Appt Date
Anna	Nails	5/1/2013
Nathan	Hair	7/5/2013
Anna	Nails	?

A = (Name, Service), B = (Appt Date)
Does (Name, Service) --> (Appt Date)?

if so, then it means our system will only let Anna make an appointment for nails on 5/1/2013 and no other date.

Functional Dependencies

functional dependencies reflect **business rules**

(SSN) → (First Name, Last Name) is a business rule set by the US government

database designers work with business folks to define functional dependencies.

the more you work databases, the more you see the same business rules over and over

Primary Key Candidate

the minimal set of fields that the row is **functionally dependent** upon

Clients

Name	Phone	Address	City
Anna Jones	215-123-4567	123 City Lane	Philadelphia
Nathan Smith	267-333-4444	999 Oak Blvd	Media

Primary Key Candidate

Clients

Name	Phone	Address	City
Anna Jones	215-123-4567	123 City Lane	Philadelphia
Nathan Smith	267-333-4444	999 Oak Blvd	Media

(Name) → (Phone, Address, City)

Primary Key Candidate

(Name) → (Phone, Address, City)

- every time I see a particular Name, I expect to get back the same address.

- does the reverse hold true?

  no!! FD's are one-way functions
  the above assertion allows more than one person to live at the same address, but prevents one person from living multiple (primary) addresses.

Primary Key Candidate

so is Name a good **primary key** candidate?

Clients

Name	Phone	Address	City
Anna Jones	215-123-4567	123 City Lane	Philadelphia
Nathan Smith	267-333-4444	999 Oak Blvd	Media

Primary Key

a field (or set of fields) that uniquely identify a row

➢ the minimal set of fields that the row is **functionally dependent** upon

(Name) → (Phone, Address, City) meets the above

Primary Key

PK additional considerations

- values in PK must be unique for each record in a table
- only one PK per table allowed
- it’s automatically indexed (for fast lookup)

Primary Key Candidate

Clients

Name	Phone	Address	City
Anna Jones	215-123-4567	123 City Lane	Philadelphia
Nathan Smith	267-333-4444	999 Oak Blvd	Media

the values stored in a **primary key** must be unique within the table - names make poor PKs

Primary Key

Clients

Client_ID	Name	Phone	Address	City
1	Anna	215-123-4567	123 City Lane	Philadelphia
2	Nathan	267-333-4444	999 Oak Blvd	Media

for this reason, you'll usually see an unique ID field used as a PK in most tables

Primary Key

primary keys are often “ID” fields in all tables

- this is done for convenience.
  - ID fields are usually **autoincrement** fields
- frameworks like CakePHP, Drupal, etc use this convention
- primary keys are automatically indexed, and numbers are faster to index
- other tables may refer back to another table’s PK, and it’s easier to bring in one field instead of multiple fields

Primary Key

Appointments

Appt_ID	Name	Service	Date
1000	Anna	Nails	5/1/2013
1001	Nathan	Hair	7/5/2013
1002	Anna	Nails	9/1/2013

(Appt_ID) → (Name, Service, Date)

Foreign Reference Keys

a field (or set of fields) that is a PK in some other table

- There can be multiple FKs in a table
- FK are how you designate that tables are related

Foreign Keys

are there any **foreign keys?**

Clients

Client_ID	Name	Phone
1	Anna	215-123-4567
2	Nathan	267-333-4444

Appointments

Appt_ID	Name	Service	Date
1000	Anna	Nails	5/1/2013
1001	Nathan	Hair	7/5/2013

Foreign Keys

a field that is a PK in some other table

Clients

Client_ID	Name	Phone
1	Anna	215-123-4567
2	Nathan	267-333-4444

Appointments

Appt_ID	Client_ID	Service	Date
1000	1	Nails	5/1/2013
1001	2	Hair	7/5/2013

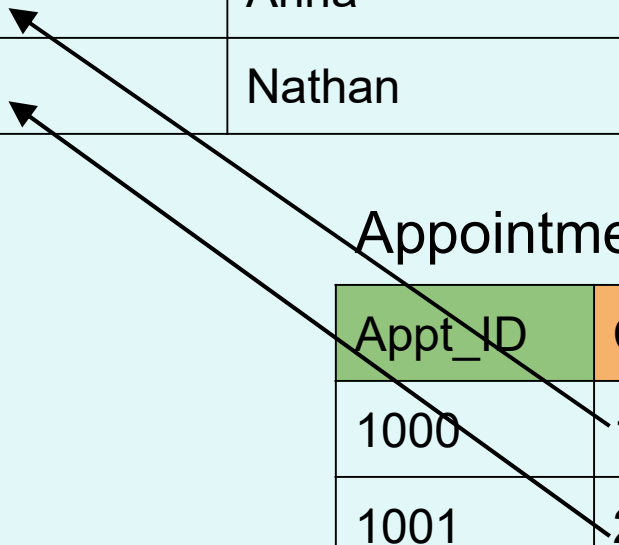
Foreign Keys

Clients

Client_ID	Name	Phone
1	Anna	215-123-4567
2	Nathan	267-333-4444

Appointments

Appt_ID	Client_ID	Service	Date
1000	1	Nails	5/1/2013
1001	2	Hair	7/5/2013



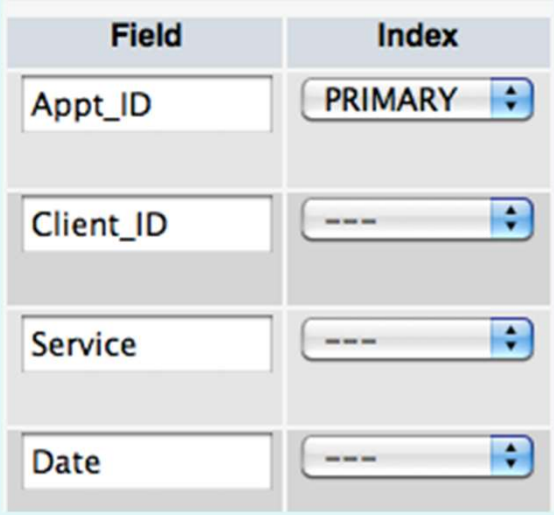
Implementing Keys

primary keys and **foreign keys** can be designated using the SQL language:

```
CREATE TABLE Appointments (
        Appt_ID INT PRIMARY KEY,
        Client_ID INT, FOREIGN KEY REFERENCES
Clients(Client_ID)
        ON UPDATE CASCADE ON DELETE RESTRICT
    Service TEXT,
        Appt_Date DATE
)
```

Implementing Keys

primary keys and **foreign keys** can also be designated via the DBMS user interface:



Snippet from PHPMyAdmin (MySQL)

Referential Integrity

Referential Integrity

a database has **referential integrity** if rules are in place that ensure that a FK can never point to a row that doesn't exist

Referential Integrity

Clients

Client_ID	Name	Phone
1	Anna	215-123-4567

?

Appointments

Appt_ID	Client_ID	Service	Date
1000	1	Nails	5/1/201
1001	2	Hair	7/5/2017

orphaned row -->

Referential Constraints

CASCADE
cascade changes in PK to all referencing FKs

RESTRICT (or NO ACTION)
don’t allow changes to PK if there is a referencing FK

Referential Constraints

what will happen if we change Anna's Client_ID from 1 to 1001?

what will happen if we delete Anna's record from the Clients table?

Break Time

Recap

- the **universe of data** is broken out into tables
- the data points in tables are related to one another
  - **functional dependencies** formalize how we recognize related data
- tables are **joined** to put the universe of data back together
- tables are joined on **primary keys** and **foreign keys**
- **primary key** values are unique within a table
- declaring foreign keys ensures that our database has **referential integrity**

Indexes

Indexes

a special data structure that speeds up data retrieval

Indexes

consider the task of trying to find all characters named Anna in *War and Peace*.

- without an index, would pretty much have to read or scan the entire novel
- a character index would require just a few page turns
- an index on a field works in a similar fashion - using a special data structure called a b-tree

Indexes

Primary Key (PK)

it's automatically indexed (for fast lookup) - why?

joins are expensive!

every row from one table is matched to every row in another table.

Joins

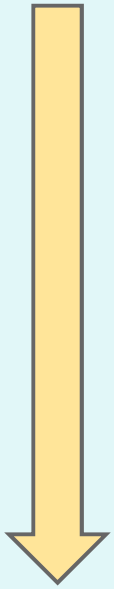
get all Clients who have had at least one Appointment

Clients

Client_ID	Name
1	Anna
2	Nathan
...	...
100000	Gia

Appointments

Appt_ID	Client_ID	Date
100001	1832	8/1/2008
100002	2432	7/5/2013
...	...	...
1000000	43901	2/1/2017



to look for any appointments for client 1, we have to scan the Appointments table up to 1 million times

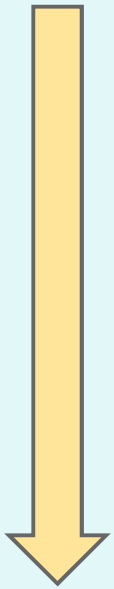
Joins

Clients

Client_ID	Name
1	Anna
2	Nathan
...	...
100000	Gia

Appointments

Appt_ID	Client_ID	Date
100001	1832	8/1/2008
100002	2432	7/5/2013
...	...	...
1000000	43901	2/1/2017



then we do the same for client 2

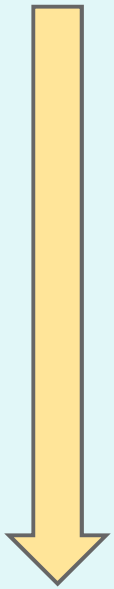
Joins

Clients

Client_ID	Name
1	Anna
2	Nathan
...	...
100000	Gia

Appointments

Appt_ID	Client_ID	Date
100001	1832	8/1/2008
100002	2432	7/5/2013
...	...	...
1000000	43901	2/1/2017



by the time we reach client 100,000, we've scanned all 1 million rows of Appointments 100000 times!

Joins

the cost of joining two tables, M and N, is M x N

in our previous example, that means
100,000 x 1,000,000 = 100,000,000,000

Joins

- computers are fast, but a M x N operation is still expensive!
- most queries will join multiple tables, not just two
- indexes reduce the time of this query to roughly M

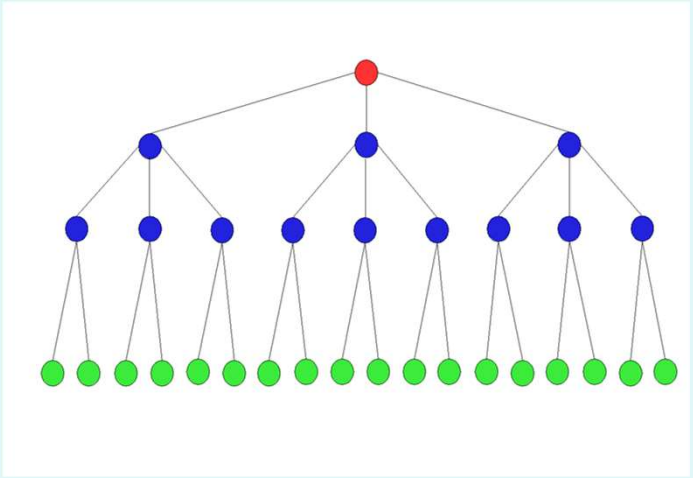
Joins using Indexes

to prevent a full table scan in table A for each row in table C, we use the index.

Clients

Client_ID	Name
1	Anna
2	Nathan
...	...

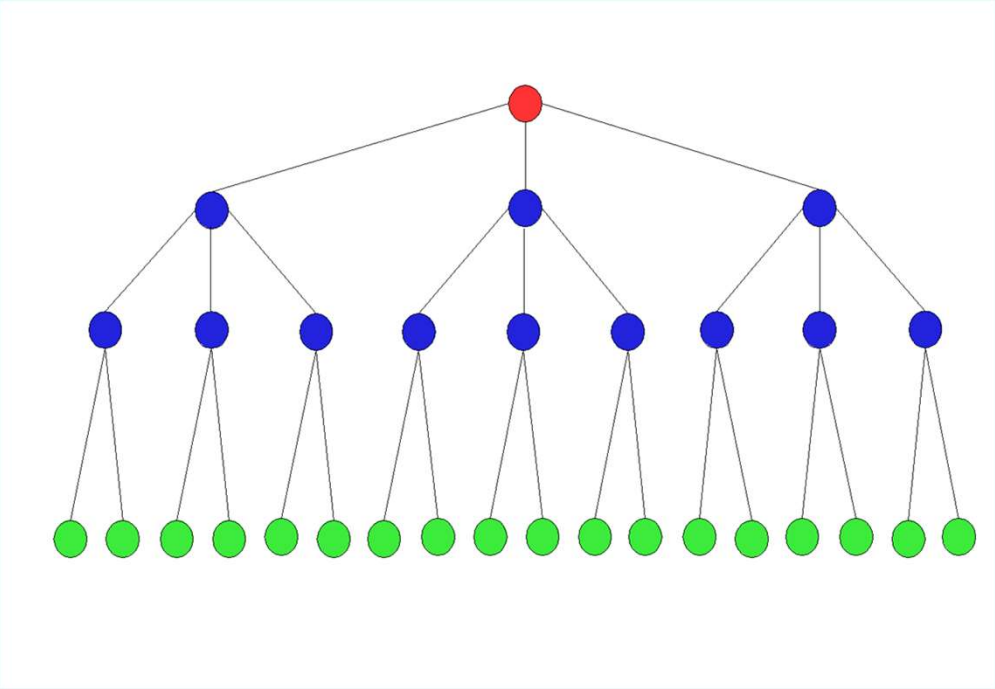
Does Client 1 have Appointment?



Yes, found Client 1 at location ….

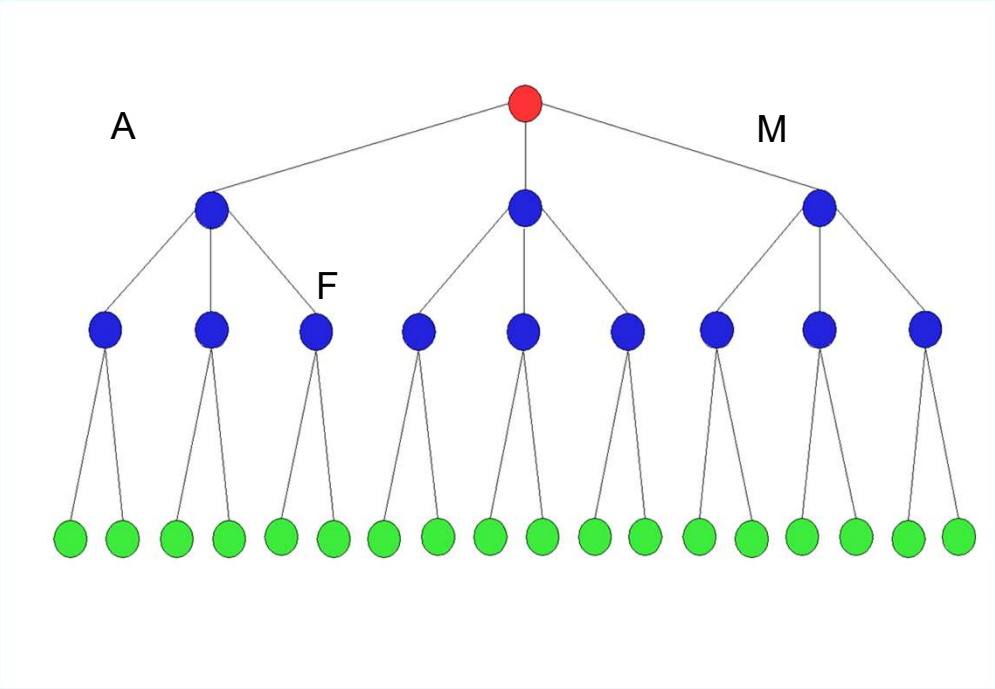
Joins using Indexes

the **index** has magical properties that allow it to find any piece of data in just a few lookups



Joins using Indexes

it's secret is that unlike a database, the values in an **index** are ordered, so it knows which branch to look in

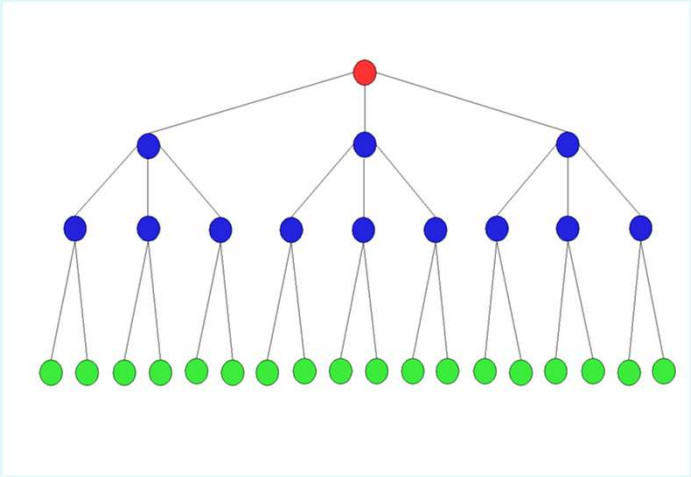


Joins using Indexes

by asking the index, it now only takes 100000 lookups (okay, more like 300000). much better than 100,000,000,000!

Clients

Client_ID	Name
1	Anna
2	Nathan
...	...



Indexes

indexes are critical for performance
so why not just index every field?

like all good things, there's a trade-off.

- while indexes speed up reads, they slow down writes.
- too many indexes can result in slower queries!
  - if the **query analyzer** can't make sense of your indexes, it won't

Indexes

what fields should be indexed?

- **primary keys** are automatically indexed by the DBMS
  - #1 reason why every table should always, always have a PK
- **foreign keys**
  - especially if you enforce referential integrity
- fields that will be queried over and over
  - name fields
  - phone number, if you lookup people by phone
  - state, if you produce mass mailings by state

Relationship Cardinality

Relationship Cardinality

there are three types of relations between tables

one-to-one
one-to-many
many-to-many

One-to-One

each row in table A has precisely one match in table B

One-to-One

Employee Public Data

Emp_ID	Dept
1	Marketing
2	Operations
...	...

Employee Secret Data

Emp_ID	SSN	Salary
1	111111111	45000
2	222222222	35000
...	...	

One-to-One

Emp_ID	Dept
1	Marketing
2	Operations
...	...
M	HR

Emp_ID	SSN	Salary
1	111111111	45000
2	222222222	35000
...	...	
?	333333333	50000

answer: M

One-to-Many

each row in A has 0 to many matches in B

One-to-Many

Clients

Client_ID	Name
1	Anna
2	Nathan
...	...

Appointments

Appt_ID	Client_ID	Service	Date
1000	1	Nails	5/1/2013
1001	2	Hair	7/5/2013
1002	1	Nails	9/1/2013
...			

One-to-Many

Clients

Client_ID	Name
1	Anna
2	Nathan
...	...
M	

Appointments

Appt_ID	Client_ID	Service	Date
1000	1	Nails	5/1/2013
1001	2	Hair	7/5/2013
1002	1	Nails	9/1/2013
...			
N			

answer: N

Inner Join

returns all rows from both tables where there is a match

Many-to-Many

each row in A has 0 to many matches in B
each row in B has 0 to many matches in A

to represent this relationship, a third table C is created (called the **cross reference table**)

Many-to-Many

Clients

Client_ID	Name
1	Anna
2	Nathan

Clients_Favorite_Things

Client_ID	Thing_ID
1	1000
2	2000
2	1000

Things

Thing_ID	Name
1000	Newspaper
2000	Baseball

Many-to-Many

Clients

Client_ID	Name
1	Anna
2	Nathan
...	...
M	

⋈

Clients_Favorite_Things

Client_ID	Thing_ID
1	1000
2	2000
2	1000
…	...
L	

⋈

Things

Thing_ID	Name
1000	Newspaper
2000	Baseball
...	...
N	

answer: L

One-to-One Implementation

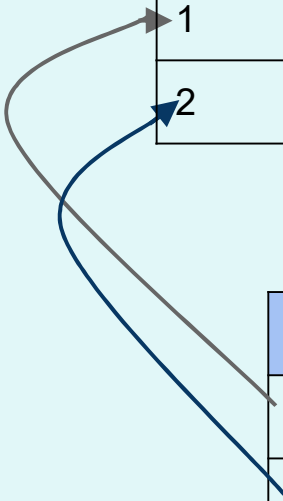
either table can store the PK or FK: the designer must choose who gets what

Employee Public Data

Emp_ID	First	Last	Title
1	April	Smith	Technician
2	Jamie	Hawkins	Marketing Manager

Employee Secret Data

Emp_ID	DOH	SSN
1	5/1/2011	111-11-1111
2	8/17/2012	222-22-2222



One-to-Many Implementation

the **many** side stores the FK pointing to the **one** side's PK

Appointments

Appt_ID	Client_ID	Service	Date
100001	1	Nails	5/1/2013
100002	2	Hair	7/5/2013
100003	1	Hair	6/1/2013

Clients

Client_ID	Name	Phone
1	Anna	215-123-4567
2	Nathan	267-333-4444

Many-to-Many Implementation

neither table can store the FK of the other - a third table represents the relationship, storing the PK of both tables

Clients

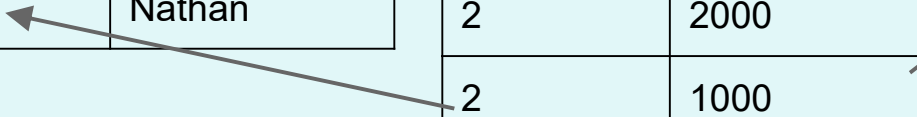
Client_ID	Name
1	Anna
2	Nathan

Clients_Favorite_Things

Client_ID	Thing_ID
1	1000
2	2000
2	1000

Things

Thing_ID	Name
1000	Newspaper
2000	Baseball



Relationship Cardinality

one of these relationships is unnecessary

one-to-one
one-to-many
many-to-many

Data Types

Data Types

in a relational database, all fields must be assigned a datatype that the values will be saved as.

most of the time this a straightforward process.

Client	Service	Date	Time	Technician	Price
Anna	Nails	5/1/2013	10:00 am	100	$30
Nathan	Hair	7/5/2013	3:30 pm	200	$25

Text
Varchar

Date

Time

Integer

Currency

Data Types

other times will require some careful thought and a judgement call

SSN	Phone	ZIP	Comments	Position
111-11-1111	(215) 111-1111	19102	Blah blah bla	1.849308339
222-22-2222	(215) 222-2222	19147	This is anoth	1.890223333

text or number?

seems like a number, but what about Canada ?

data could be truncated if text field is made too small.

storing scientific measurements? beware that "floats" may truncate your precision

The Normal Forms

Normal Forms

the **normal forms** are the specifications for how to split your fields into tables in such a way that

- eliminates redundancy
- prevents data anomalies
- **functional dependencies** are preserved, thereby enabling
- **loss-less joins** (the tables can be put back together to yield the precise universe of data)

Normal Forms

1st Normal Form (1NF)
2nd Normal Form (2NF)
3rd Normal Form
4th Normal Form (4NF)
Boyce-Codd Normal Form
5th Normal Form (5NF)
Higher forms (academic)

Normal Forms

your database is considered to be **normalized** if it is in least **3NF**

- the goal of normalizing a database is to prevent data anomalies from occurring during insert, update or delete operations, and most 3NF tables are free of these anomalies.

First Normal Form (1NF)

data in each field is atomic
(cannot be decomposed into additional fields)

First Normal Form (1NF)

each field is atomic

- basically says a field cannot contain a table (or multiple values)

Not in 1NF

EmpID	Favorite Things
1	Mittens, Raindrops
2	Raindrops, Schnitzel
3	Doorbells, Mittens

First Normal Form (1NF)

EmpID	Favorite Things
1	Mittens, Raindrops
2	Raindrops, Schnitzel
3	Doorbells, Mittens

how do we get this in 1NF?

First Normal Form (1NF)

decomposition into 1NF depends on relationship type

- ➔ **one-to-one**: make new fields
- ➔ **one-to-many**: make new table
- ➔ **many-to-many**: make two new tables

First Normal Form (1NF)

EmpID	Favorite Things
1	Mittens, Raindrops
2	Raindrops, Schnitzel
3	Doorbells, Mittens

what is the relationship type between employee and favorite things?

First Normal Form (1NF)

➔ **many-to-many**: make two new tables

EmpID
1
2

EmpID	ThingID
1	100
1	200
2	200
2	300

ThingID	Favorite Things
100	Mittens
200	Raindrops
300	Schnitzel
400	Doorbells

First Normal Form (1NF)

In 1NF?

EmpID	Name
1	Mary Smith
2	Todd T Burke

Definitely 1NF

EmpID	First Name	Last Name
1	Mary	Smith
2	Todd	Burke

depends...

- do you want to search/sort by Last Name?
- do want to be compliant with industry standard?
- business rules determine what is considered “normal”

Second Normal Form (2NF)

in 1NF, and every field in a table is functionally dependent on a subset of the PK

Third Normal Form (3NF)

in 2NF, and every field in a table is functionally dependent on all fields in the PK

Third Normal Form (3NF)

the database is already in 1NF

- every field in a table is functionally dependent on all fields in the PK

Appt_ID	Appt Date	Client_ID	Service	Service Price
1001	2/23/13	1	Nails	20
1002	2/24/13	2	Hair	30

is this in 3NF?

Third Normal Form (3NF)

to determine if table is in 3NF

- identify the primary key
- determine if every field is dependent on PK

Appt_ID	Appt Date	Client_ID	Service	Service Price
1001	2/23/13	1	Nails	20
1002	2/24/13	2	Hair	30

what's the primary key?

Third Normal Form (3NF)

is every field dependent on Appt_ID?

Appt_ID	Appt Date	Client_ID	Service	Service Price
1001	2/23/13	1	Nails	20
1002	2/24/13	2	Hair	30

(Appt_ID) -> (Date, Client, Service, Service Price)?

Do the fields contain all the information, and only the information, needed to define an “appointment”?

Third Normal Form (3NF)

if we assert (Service) -> (Service Price), then not 3NF.

Appt_ID	Appt Date	Client_ID	Service	Service Price
1001	2/23/13	1	Nails	20
1002	2/24/13	2	Hair	30

Third Normal Form (3NF)

decomposition into 3NF: independent functional dependencies become new relations (tables)

Third Normal Form (3NF)

decomposition into 3NF depends on relationship type

➔ **one-to-one**: make new table

➔ **one-to-many**: make new table

➔ **many-to-many**: make two new tables

Third Normal Form (3NF)

treat the independent
(Service) -> (Service Price)
functional dependency as a new relation (table)

Appointments

Appt_ID	Appt Date	Client	Service
1	2/23/13	Anna	Nails
2	2/24/13	Nathan	Hair

Services

Service	Service Price
Nails	20
Hair	30

Practice Time

Pet Salon

Client	Name	Person	Address	Favorite Toys	Appt Date	Service	Price
	Anna	Nicole Jones	123 A St Phila PA 19146	Shoes, Frisbee	2/23/13	Nails	30
	Nathan	Amelia Smith	999 Oak Blvd Phila PA 19102	Rubber Ball	6/15/13	Daycare	20
	Meep	Mark Doe	27 T St Phila PA 19127	Paper Balls	6/10/13	Board	35
	Boo	Nicole Jones	123 A St Phila PA 19146	crinkle ball	7/1/13	Nails	30
	Anna	Nicole Jones	123 A St Phila PA 19146	Frisbee, Newspaper	5/1/13	Daycare	20

Let’s Normalize This!

Pet Salon

Client	Name	Person	Address	Favorite Toys	Appt Date	Service	Price
	Anna	Nicole Jones	123 A St Phila PA 19146	Shoes, Frisbee	2/23/13	Nails	30
	Anna	Nicole Jones	123 A St Phila PA 19146	Frisbee, Newspaper	5/1/13	Daycare	20
	Boo	Nicole Jones	123 A St Phila PA 19146	crinkle ball	7/1/13	Nails	30

1NF violations

Favorite Toys
Address
Person

Pet Salon

Client	Name	Person	Address	Favorite Toys	Appt	Service	Price
	Anna	Nicole Jones	123 A St Phila PA 19146	Shoes, Frisbee	2/23/13	Nails	30
	Anna	Nicole Jones	123 A St Phila PA 19146	Frisbee, Newspaper	5/1/13	Daycare	20
	Boo	Nicole Jones	12 M St Phila PA 19147	crinkle ball	7/1/13	Nails	30

Favorite Toys

how do we decompose in 1NF?

1. identify relationship type
   - ➢ many-to-many
     - ➢ create two new tables

Pet Salon

Client	Name	Person	Address	Appt Date	Service	Price
	Anna	Nicole Jones	123 A St Phila PA 19146	2/23/13	Nails	30
	Anna	Nicole Jones	123 A St Phila PA 19146	5/1/13	Daycare	20
	Boo	Nicole Jones	12 M St Phila PA 19147	7/1/13	Nails	30

Toys

Toy_ID	Toy
10	Rubber Newspaper
20	Shoes
30	Frisbee
40	Crinkle ball

Clients Favorite Toys

Name	Toy_ID
Anna	20
Anna	30
Boo	40
Anna	10

Pet Salon

Client	Name	Person	Address	Appt Date	Service	Price
	Anna	Nicole Jones	123 A St Phila PA 19146	2/23/13	Nails	30
	Anna	Nicole Jones	123 A St Phila PA 19146	5/1/13	Daycare	20
	Boo	Nicole Jones	12 M St Phila PA 19147	7/1/13	Nails	30

1NF violations

Address

Person

identify relationship type

- one-to-one
  - create new fields

Pet Salon

Client	Client Name	Person First	Person Last	Street	City	State	Zip	Appt	Service	Price
	Anna	Nicole	Jones	123 A St	Phila	PA	19146	2/23/13	Nails	30
	Anna	Nicole	Jones	123 A St	Phila	PA	19146	5/1/13	Daycare	20
	Boo	Nicole	Jones	123 A St	Phila	PA	19146	7/1/13	Nails	30

Person and Address are split into separate fields

Pet Salon

Client	Client Name	Person First	Person Last	Street	City	State	Zip	Appt	Service	Price
	Anna	Nicole	Jones	123 A St	Phila	PA	19146	2/23/13	Nails	30
	Anna	Nicole	Jones	123 A St	Phila	PA	19146	5/1/13	Daycare	20
	Boo	Nicole	Jones	123 A St	Phila	PA	19146	7/1/13	Nails	30

now look for 3NF violations

- first we need identify to the primary key

  there's a chicken-egg problem here: if a table is not in 3NF, then by definition no PK candidate exists

  so to start, let's list all functional dependencies

Pet Salon

Client	Client Name	Person First	Person Last	Street	City	State	Zip	Appt	Service	Price
	Anna	Nicole	Jones	123 A St	Phila	PA	19146	2/23/13	Nails	30
	Anna	Nicole	Jones	123 A St	Phila	PA	19146	5/1/13	Daycare	20
	Boo	Nicole	Jones	123 A St	Phila	PA	19146	7/1/13	Nails	30

functional dependencies:

(Client Name) -> (Client Photo)
(Person First, Person Last) -> (Street, City, State, Zip)
(Appt Date, Service) -> (Price)

Pet Salon

Client	Client Name	Person First	Person Last	Street	City	State	Zip	Appt	Service	Price
	Anna	Nicole	Jones	123 A St	Phila	PA	19146	2/23/13	Nails	30
	Anna	Nicole	Jones	123 A St	Phila	PA	19146	5/1/13	Daycare	20
	Boo	Nicole	Jones	123 A St	Phila	PA	19146	7/1/13	Nails	30

in 3NF, each **functional dependency** is represented in a relation (table), so let's name what relations our FD's represent:

(Client Name) -> (Client Photo): **Clients**

(Person First, Person Last) -> (Street, City, State, Zip): **People**

(Appt Date, Service) -> (Price): **Appointments**

Pet Salon

Clients

Client	Client ID	Client Name	Person First	Person Last	Street	City	State	Zip	Appt	Service	Price
	1	Anna	Nicole	Jones	123 A St	Phila	PA	19146	2/23/13	Nails	30
	1	Anna	Nicole	Jones	123 A St	Phila	PA	19146	5/1/13	Daycare	20
	2	Boo	Nicole	Jones	123 A St	Phila	PA	19146	7/1/13	Nails	30

assert that table above is the Clients table

now know these two FD's are in violation of 3NF:
(Person First, Person Last) -> (Street, City, State, Zip)
(Appt Date, Service) -> (Price)

Pet Salon

Clients

Client	Client ID	Client Name	Person First	Person Last	Street	City	State	Zip	Appt	Service	Price
	1	Anna	Nicole	Jones	123 A St	Phila	PA	19146	2/23/13	Nails	30
	1	Anna	Nicole	Jones	123 A St	Phila	PA	19146	5/1/13	Daycare	20
	2	Boo	Nicole	Jones	123 A St	Phila	PA	19146	7/1/13	Nails	30

(Person First, Person Last) -> (Street, City, State, Zip)

- identify the relationship type
  - **one-to-many**
    - create a new table

Pet Salon

Clients

Client	Client ID	Client Name	Person_ID	Appt	Service	Price
	1	Anna	23	2/23/13	Nails	30
	1	Anna	23	5/1/13	Daycare	20
	2	Boo	23	7/1/13	Nails	30

People

Person_ID	Person First	Person Last	Street	City	State	Zip
23	Nicole	Jones	123 A St	Philadelphia	PA	19146

Pet Salon

Clients

Client	Client ID	Client Name	Person_ID	Appt	Service	Price
	1	Anna	23	2/23/13	Nails	30
	1	Anna	23	5/1/13	Daycare	20
	2	Boo	23	7/1/13	Nails	30

(Appt Date, Service) -> (Price)

- identify the relationship type
  - **one-to-many**
    - create a new table

Pet Salon

Clients

Client	Client_ID	Name	Person_ ID	Appt_ID
	1	Anna	23	49032
	1	Anna	23	98907
	2	Boo	23	76785

Appointments

Appt_ID	Date	Service	Price
49032	2/23/13	Nails	30
98907	5/1/13	Daycare	20
76785	7/1/13	Nails	30

Pet Salon

Clients

Client	Client_ID	Name	Person_ ID	Appt_ID
	1	Anna	23	49032
	1	Anna	23	98907
	2	Boo	23	76785

Appointments

Appt_ID	Date	Service	Price
49032	2/23/13	Nails	30
98907	5/1/13	Daycare	20
76785	7/1/13	Nails	30

In a 1:N relationship, the FK goes in the Many table. Here, one Client have many Appointments, so Appointments is our “Many” table.

Pet Salon

Clients

Client	Client_ID	Name	Person_ ID
	1	Anna	23
	2	Boo	23

Note that we removed redundancy!

Appointments

Appt_ID	Client_ID	Date	Service	Price
49032	1	2/23/13	Nails	30
98907	1	5/1/13	Daycare	20
76785	2	7/1/13	Nails	30

Pet Salon

Last but not least...

Appointments

Appt_ID	Client_ID	Date	Service	Price
49032	1	2/23/13	Nails	30
98907	1	5/1/13	Daycare	20

We saw earlier that (Service) -> (Price) is an independent FD, so we can break this out too

Pet Salon

last but not least...

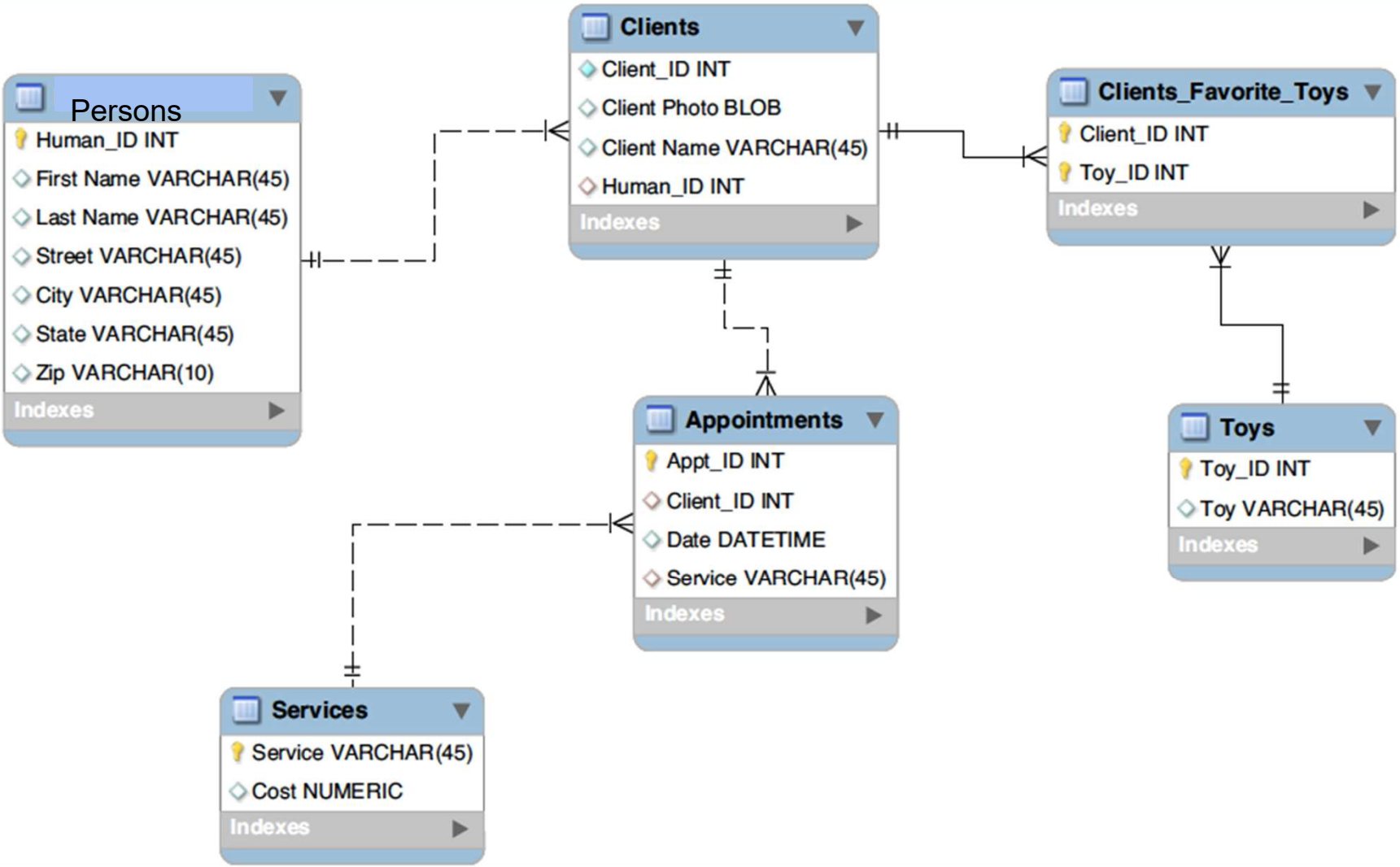
Appointments

Appt_ID	Client_ID	Date	Service
49032	1	2/23/13	Nails
98907	1	5/1/13	Daycare

Services

Service	Price
Nails	30
Daycare	20

Database Schema



Recap

- data is stored in **tables**
- **normal forms** guide us in how to determine which fields belong in which tables
- every table should be assigned a **primary key**
- tables are related to each other on **primary keys** and **foreign keys**
- three **types of relationships** - **1:1, 1:N, N:N**
- a N:N relationship is represented by a **cross-reference table**
- declaring foreign keys ensures database has **referential integrity**
- two types of referential integrity constraints: **cascade, restrict**
- all fields that will be frequently qualified in a query should be **indexed**

Resources

Free DMBS

All big players in DB world offer a free version of their DBMS:
SQL Server (Express/Developer Edition), Oracle Express

Others are open-sourced: MySQL, PostgreSQL

Free Report Writing tools

SSRS comes with SQL Server Express w/ Adv Options
Report Builder is a free download
Power BI desktop is free

Resources

Free Data

Microsoft makes sample databases available (Northwinds, AdventureWorks, WideWorldImports)

Kaggle.com – hosts data science competitions, posts free datasets to play with

Philly Open Data (www.opendataphilly.org)

Resources

Free Online Class
db-class.org (original MOOC from Stanford)

Microsoft (SQL Svr/Power BI)
http://www.sqlsaturday.com/
http://www.sqlservercentral.com/
https://mva.microsoft.com/

Relational Database Terminology

Table === Relation
Row === Tuple
Values === Domain
Field === Attribute