MONTANA
STATE UNIVERSITY

CSCI 440 - Database Systems
Final Review

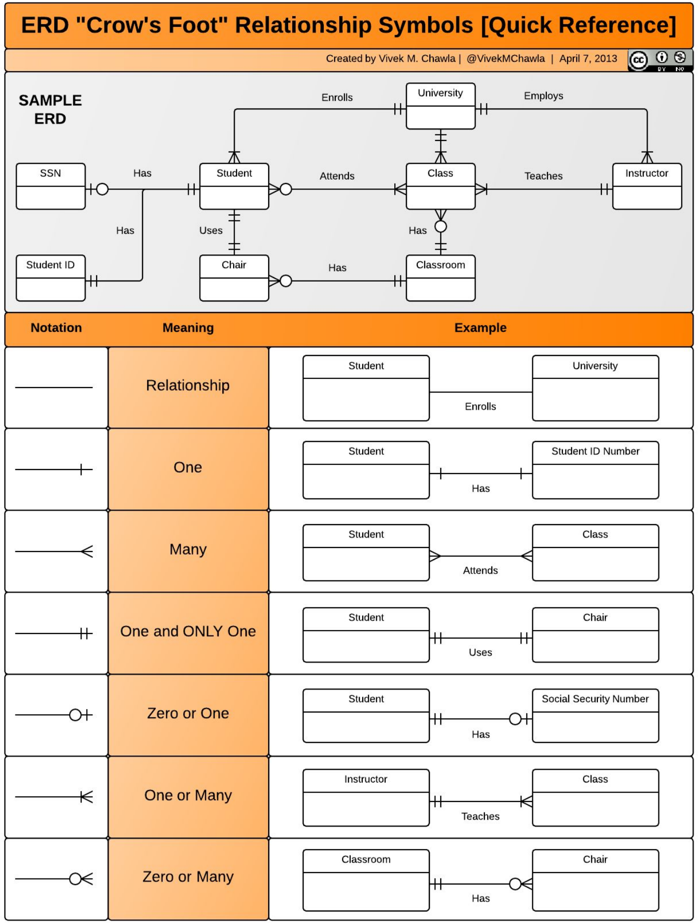
Instructor: Carson Gross
carson.gross@montana.edu

Course Goals

- To give you a broad understanding of relational databases
- To help you become proficient in SQL
- To help you be confident in schema design
- Enable you to work with databases in code (Java)
- Learn a bit of database theory and implementation
- Learn about some non-relational modern tools
  - NoSQL
  - Cloud Architectures

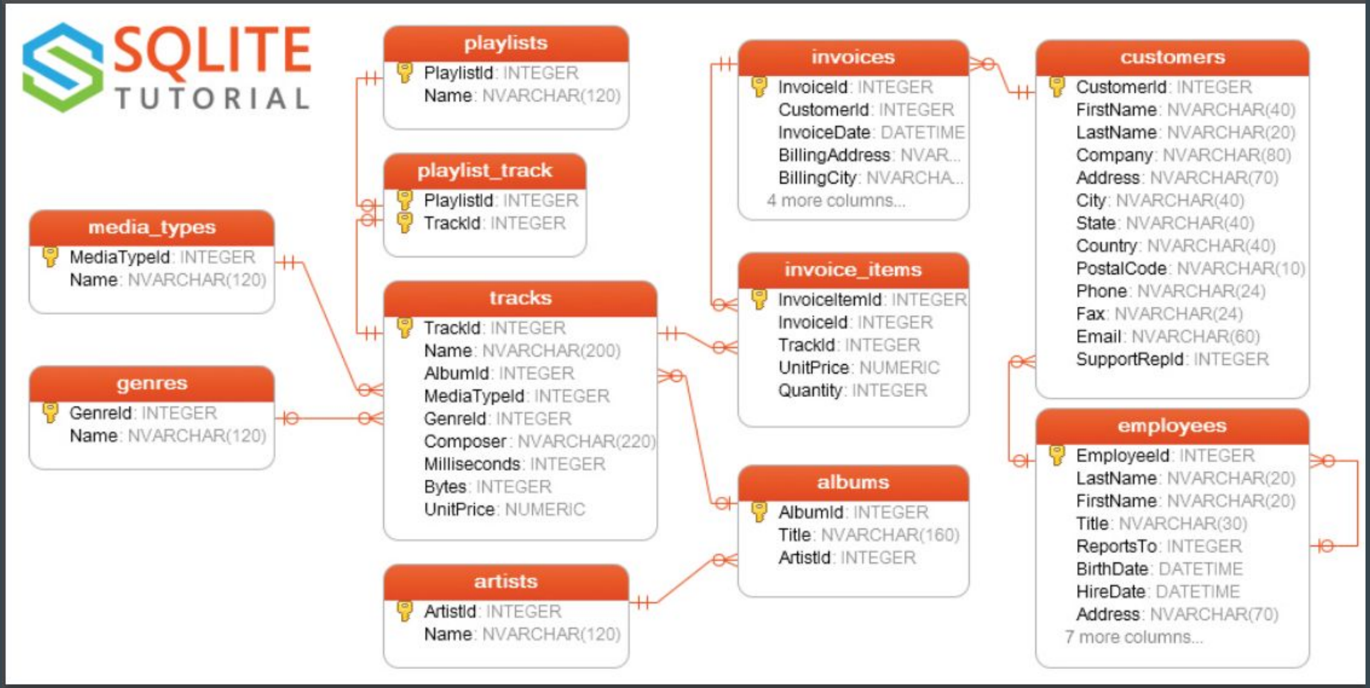
E/R Diagrams

- Most commonly used E/R format today is “Crows Feet”
- Cheat sheet available here: https://www.vivekmchawla.com/erd-crows-foot-relationship-symbols-cheat-sheet/
- Please understand all the relationships on this diagram!



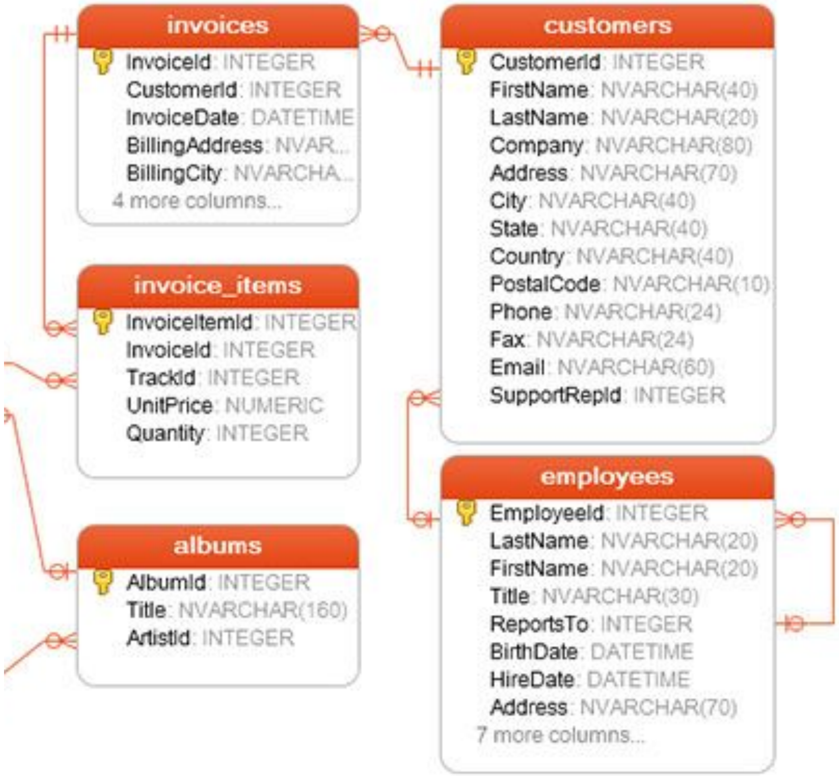
Visualizing A Relational Model

- You should be familiar with the ChinookDB schema:



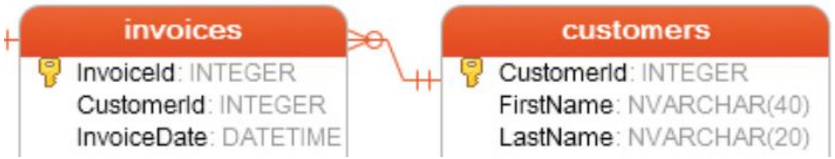
The Relational Model

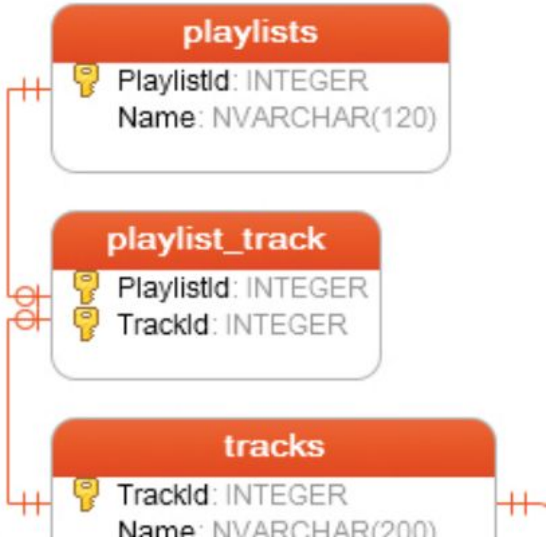
- Understand how Foreign Key references work
- `What is invoices.CustomerId encoding?`
- Why does Employees have a self-referential Foriegn Key?



The Relational Model

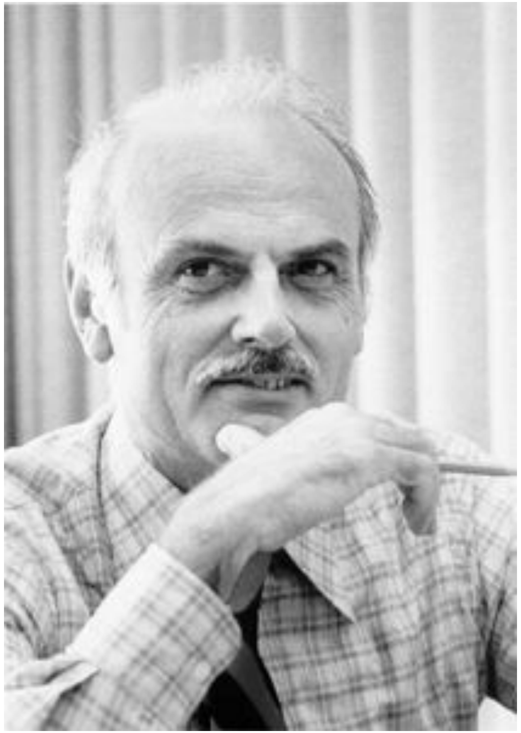
- 1-N Relationship
  - FK is in N table
- N-N Relationship
  - Done with a join table with FK of both tables
- 1-1 Relationship
  - FK can be in either table





Database Normalization

- Structuring database tables such that
  - Redundancy is minimized
  - Data integrity is maximized
- Edgar F Codd: a pioneer in databases
  - Proposed “1st Normal Form” in 1970
  - Went on to propose many more increasingly strict normalized forms



1st Normal Form

- There is a key
- Consider this table. Is there a key?
    - No: duplicate rows

Grades

Grade	Student	Class	Teacher	Satisfied?
B	Joe Smith	CSCI 366	C Gross	Yes
A	Marge Liu	CSCI 366	C Gross	Yes
A	Kelly Chen	CSCI 440	M Wittie	Yes
B	Xerces Orion	CSCI 366	C Gross	Yes
A	Marge Liu	CSCI 366	C Gross	Yes
C	Ted Jacobs	CSCI 440	M Wittie	Yes

2nd Normal Form

- To achieve 2NF, all data must depend on the entire key

Grades

Grade	Student	Class	Teacher	Satisfied?
B	Joe Smith	CSCI 366	C Gross	Yes
A	Marge Liu	CSCI 366	C Gross	Yes
A	Kelly Chen	CSCI 440	M Wittie	Yes
B	Xerces Orion	CSCI 366	C Gross	Yes
C	Ted Jacobs	CSCI 440	M Wittie	Yes

2nd Normal Form

- The key for this table is {Student, Class}
- Is there any data that depends only on part of that key?

Grades

Grade	Student	Class	Teacher	Satisfied?
B	Joe Smith	CSCI 366	C Gross	Yes
A	Marge Liu	CSCI 366	C Gross	Yes
A	Kelly Chen	CSCI 440	M Wittie	Yes
B	Xerces Orion	CSCI 366	C Gross	Yes
C	Ted Jacobs	CSCI 440	M Wittie	Yes

2nd Normal Form

- Teacher depends **only** on Class
- To fix this, we need to pull Teacher data out to a separate table

Grades

Grade	Student	Class	Teacher	Satisfied?
B	Joe Smith	CSCI 366	C Gross	Yes
A	Marge Liu	CSCI 366	C Gross	Yes
A	Kelly Chen	CSCI 440	M Wittie	Yes
B	Xerces Orion	CSCI 366	C Gross	Yes
C	Ted Jacobs	CSCI 440	M Wittie	Yes

2nd Normal Form

- We are now in 2NF
- Note that `C Gross` and `M Wittie` only appear once
  - Data redundancy has been removed
  - Easier to avoid update errors

Teaching

Class	Teacher
CSCI 366	C Gross
CSCI 440	M Wittie

Grades

Grade	Student	Class	Satisfied?
B	Joe Smith	CSCI 366	Yes
A	Marge Liu	CSCI 366	Yes
A	Kelly Chen	CSCI 440	Yes
B	Xerces Orion	CSCI 366	Yes
C	Ted Jacobs	CSCI 440	Yes

3rd Normal Form

- 3NF demands that all data depend *only* on the key
- What data here that does not depend on the key?
- The satisfied column depends on the Grade column only

Teaching

Class	Teacher
CSCI 366	C Gross
CSCI 440	M Wittie

Grades

Grade	Student	Class	Satisfied?
B	Joe Smith	CSCI 366	Yes
A	Marge Liu	CSCI 366	Yes
A	Kelly Chen	CSCI 440	Yes
B	Xerces Orion	CSCI 366	Yes
C	Ted Jacobs	CSCI 440	Yes

3rd Normal Form

- We now have a database in 3NF
- It is also in BCNF
- 3NF typically satisfies BCNF, especially with surrogate keys

Teaching

Class	Teacher
CSCI 366	C Gross
CSCI 440	M Wittie

Satisfied

Grade	Satisfied?
A	Yes
B	Yes
C	Yes
D	No
F	No

Grades

Grade	Student	Class
B	Joe Smith	CSCI 366
A	Marge Liu	CSCI 366
A	Kelly Chen	CSCI 440
B	Xerces Orion	CSCI 366
C	Ted Jacobs	CSCI 440

3rd Normal Form

- What have we accomplished?
- Data redundancy has been minimized
- Update complexity has been minimized
  - E.g. it is easy to change "Satisfied" criteria now

Teaching

Class	Teacher
CSCI 366	C Gross
CSCI 440	M Wittie

Satisfied

Grade	Satisfied?
A	Yes
B	Yes
C	Yes
D	No
F	No

Grades

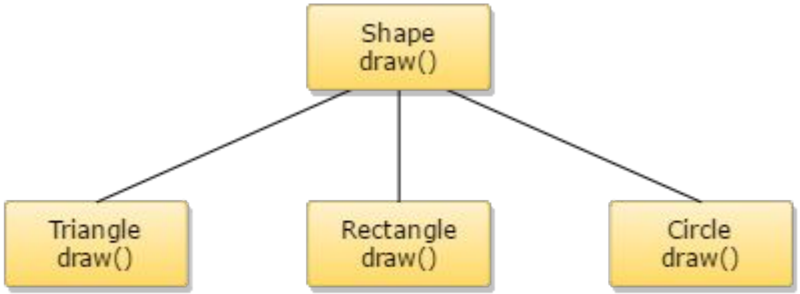
Grade	Student	Class
B	Joe Smith	CSCI 366
A	Marge Liu	CSCI 366
A	Kelly Chen	CSCI 440
B	Xerces Orion	CSCI 366
C	Ted Jacobs	CSCI 440

Normal Form Summary

- Each non-key column in a relation depends on
  - The key (1NF)
  - The whole key (2NF)
  - And nothing but the key (3NF/BCNF)
  - *So help me Cobb* ;)
- In the presence of a surrogate key, things become pretty obvious
  - In industry, there is *always* a surrogate key
- What’s The General Principle?

Polymorphism

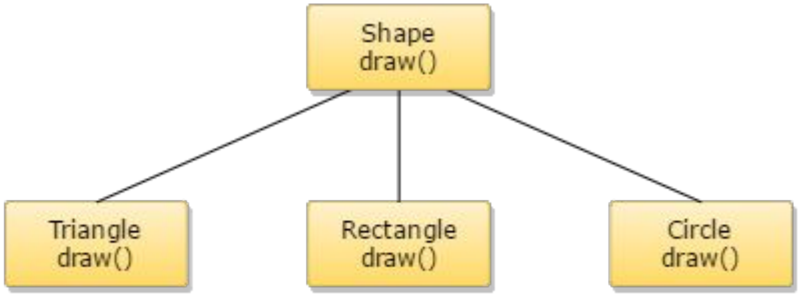
- You are probably familiar with this idea from your Object Oriented classes
  - Super-classes
  - Sub-classes
  - Sub-classes extend the super-class
    - Add methods and ***attributes***



Polymorphism

The Relational Model

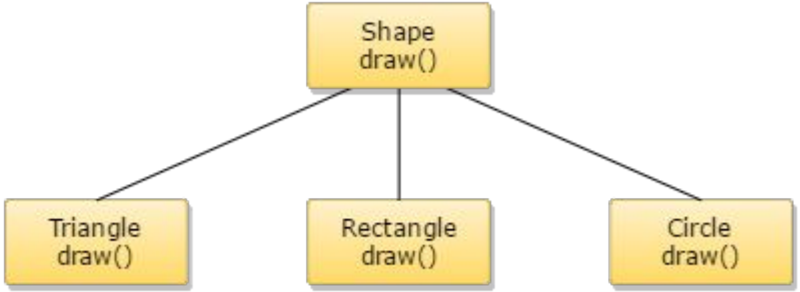
- Relations are just tables and foriegn keys
- How should we model this object hierarchy if we want to store it in a table?



Polymorphism

Three Approaches

- Single Table Inheritance
- Class Table Inheritance
- Concrete Table Inheritance



Polymorphism

Single Table Inheritance

- A single table is used for all sub-class instances
- The columns are the union of all columns of sub-classes
- Advantages?
- Disadvantages?

Single Table Inheritance

Shape Type	x	y	radius	length	width	height
Triangle	1	5			10	20
Rectangle	3	6		20	10	
Circle	8	3	6			
Triangle	4	4			6	9

Class Table Inheritance

- There is one table per class in the object hierarchy
- Sub-classes include a foreign key reference to their parent classes
- Advantages?
- Disadvantages?

Shapes

id	x	y
1	1	5
2	3	6
3	8	3
4	4	4

Circles

shape_id	radius
3	6

Rectangles

shape_id	length	width
2	20	10

Triangles

shape_id	length	width
1	10	20
4	6	9

Concrete Table Inheritance

- There is one table per **concrete** class in your object hierarchy
- Advantages?
- Disadvantages?

Circles

x	y	radius
8	3	6

Rectangles

x	y	length	width
3	6	20	10

Triangles

x	y	width	height
1	5	10	20
4	4	6	9

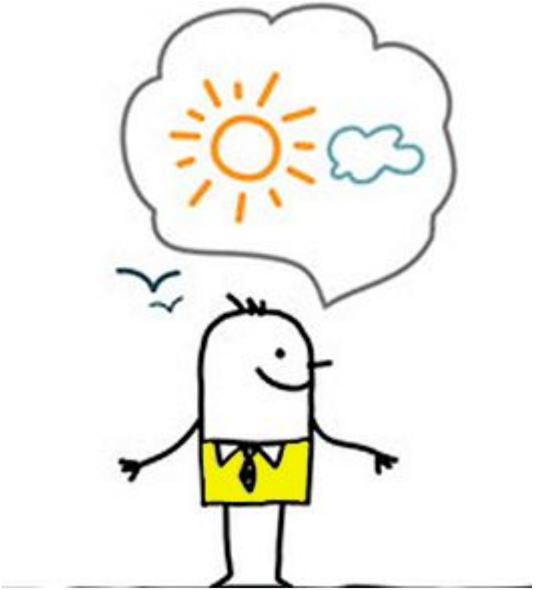
Indexes

- Indexes make queries faster by building a side data structure (a b-tree) that can be consulted when querying the database
- What query does this index make faster?
- What are the downsides of indexes?

```sql
-- email index
CREATE UNIQUE INDEX idx_employees_email
    ON employees (Email);
```

Optimistic Concurrency

- Optimistic concurrency allows concurrency issues to happen, but....
  - Reacts to them when they do
  - Assume that, for the most part, things will work out



Optimistic Concurrency

- Here is an implementation of optimistic concurrency, using the old value of the Name field to ensure an update only occurs if the Name has not been changed

```
UPDATE artists
SET Name="DC/AC"
WHERE Name="AC/DC" AND ArtistId=1;
```

Implementing O/C

- Check the number of rows updated
    - If 1 row was updated, success
    - If 0 rows were updated, failure
- On success, our optimism paid off!
- On failure, oh well, let the user know and maybe they will try again...

```
UPDATE artists
SET Name="DC/AC"
WHERE Name="AC/DC" AND ArtistId=1;
```

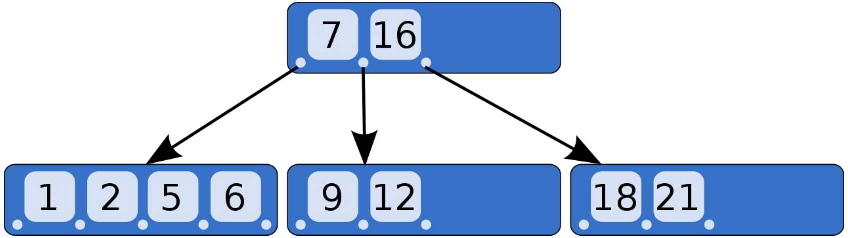
Implementing

- Note that this approach is very web friendly
- If a user is looking at tickets and wanders away, no locks are being held
- If a user accidentally picks a seat already taken (in the meantime), no worse than when using pessimistic concurrency

```
UPDATE artists
SET Name="DC/AC"
WHERE Name="AC/DC" AND ArtistId=1;
```

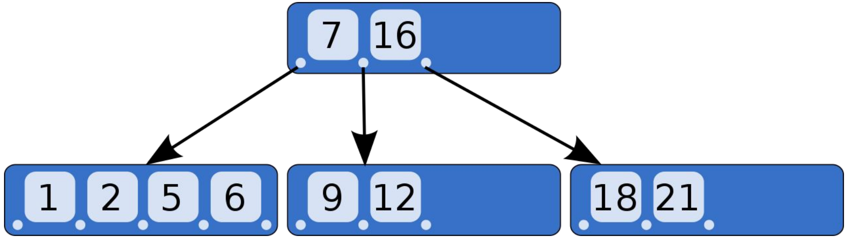
What is a B-Tree?

- Well, first off, it is obviously a *Tree*
    - A collection of hierarchically arranged data
- It is also *self balancing*
    - No branch can get too deep compared with other branches
- As a tree, it supports *logarithmic time operations*
    - search, insert, delete



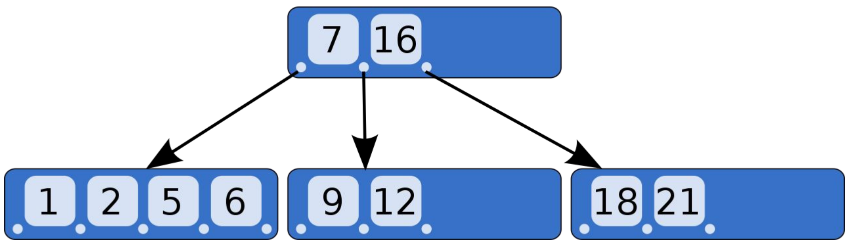
What is a B-Tree?

- B-Trees are particularly well suited for *block storage*
  - Block storage is a storage system that consists of large chunks, or blocks, of data
  - E.g. Hard drives
- Because of this advantage, B-Trees are commonly used in databases, file systems, etc.



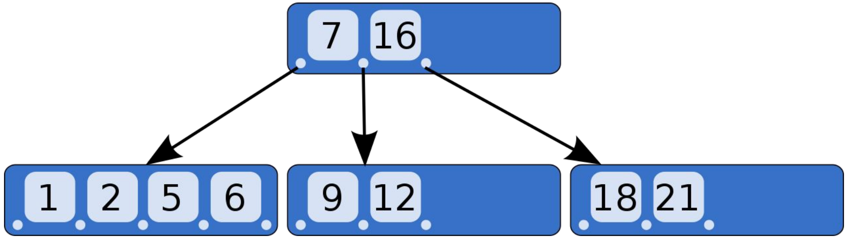
What is a B-Tree?

- Pictured at right is a simple B Tree
- The B Tree consists of *Nodes*
- Nodes hold *Keys* (values) and *Pointers*
- The top node is the *Root Node*
- The bottom nodes are all *Leaf Nodes*



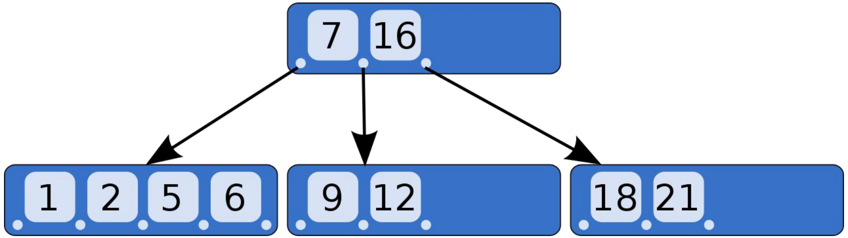
What is a B-Tree?

- Note that a node in this B tree has room for four values (keys) and five pointers
- The *Branching Factor* of this tree is 5
  - Note that the number of values available is always (*Branching Factor* - 1)
- The Branching Factor is sometimes called the *Degree* of the B Tree



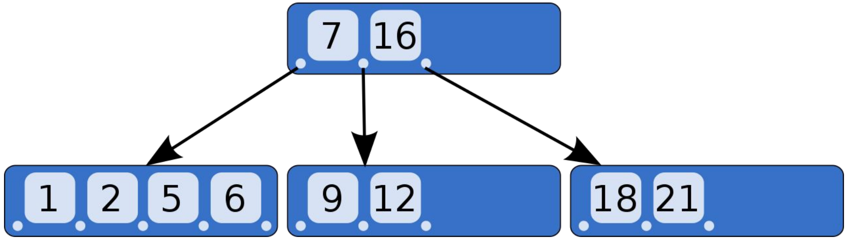
What is a B-Tree?

- Because of this relationship, you will sometimes see specific B Trees described as "*N-M B Trees*"
  - N = number of keys
  - M = number of pointers
- In our example we have a 4-5 B-Tree



B Tree Operations - Search

- Search in a B Tree is similar to other tree search operations
- Search for value 12
  - Begin in root
  - Scan numbers
  - 12 is < 16, so follow pointer
  - Scan leaf
  - Find value at second position
- O(log(n)) - n = number of keys



Redis

- Redis is a *NoSQL* data store
- *Re*mote *Di*ctionary *S*ervice
- Core concept is that of key-value pairs
  - Not unlike a giant hash table
- *Widely* used in industry
  - Almost every major website you use has Redis somewhere

MongoDB

- Like Redis, MongoDB is a *NoSQL* data store
- Unlike Redis, MongoDB is a *Document Database*
    - Stores JSON-like documents rather than offering various data types


THE TWO STATES OF
EVERY PROGRAMMER
I AM A GOD.
I HAVE NO IDEA
WHAT I'M DOING.

MONTANA
STATE UNIVERSITY