Advanced Database Design
SQL Basics

Gregory S. DeLozier, Ph.D.

gdelozie@kent.edu

Topics

- SQL
- SQL Statements
- Search Optimization
- Object Relational Mapping

SQL STATEMENTS

Select

- For a single table
  - Returns a subset of columns
  - Returns a subset of rows (based on where)
  - Orders rows
  - Groups rows

Select

- For a multiple tables
  - Creates a larger view onto both tables
  - “where” defines connection between tables
  - Returns a subset of columns and rows as before
  - Orders rows, groups rows

Select Example

1	Suzy	Dog	12
2	Sandy	Cat	4
3	Whiskers	Hamster	2

- This is a “flat file”

Select Example

1	Suzy	Dog	12
2	Sandy	Cat	4
3	Whiskers	Hamster	2

select name,kind from animals

Suzy	Dog
Sandy	Cat
Whiskers	Hamster

Select Example

1	Suzy	Dog	12
2	Sandy	Cat	4
3	Whiskers	Hamster	2

select name,kind from animals where age > 10

Suzy	Dog

Select Example

1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
45	Hamster

- This is relational data

Select Example

1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
45	Hamster

- This is a relation

Select Example

1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
45	Hamster

- This is a many-to-one relation

Select Example

1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
45	Hamster

select name,kind

Select Example

1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
45	Hamster

```
select name,kind
    from animals, kinds
```

Select Example

1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
45	Hamster

```
select name,kind
    from animals, kinds
```

Select Example

1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
45	Hamster

```
select name,kind
    from animals, kinds
    where animals.kindid = species.id
```

Select Example

1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
45	Hamster

```
select name,kind
    from animals, kinds
    where animals.kindid = species.id
```

Suzy	Dog
Sandy	Cat
Whiskers	Hamster

Select Example

1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
45	Hamster

```
select name,kind
    from animals, kinds
    where animals.kindid = species.id
    and animals.age > 10
```

Suzy	Dog
Heidi	Dog

DATA JOINS

Join

- For a multiple tables
- Creates views on multiple tables
- Rows can be “connected” as with select

Join Example

1	Suzy	22	12
7	Chipper	25	6
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
37	Fish
45	Hamster

Add some rows that do not relate…

Join Example

1	Suzy	22	12
7	Chipper	25	6
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
37	Fish
45	Hamster

Related rows…

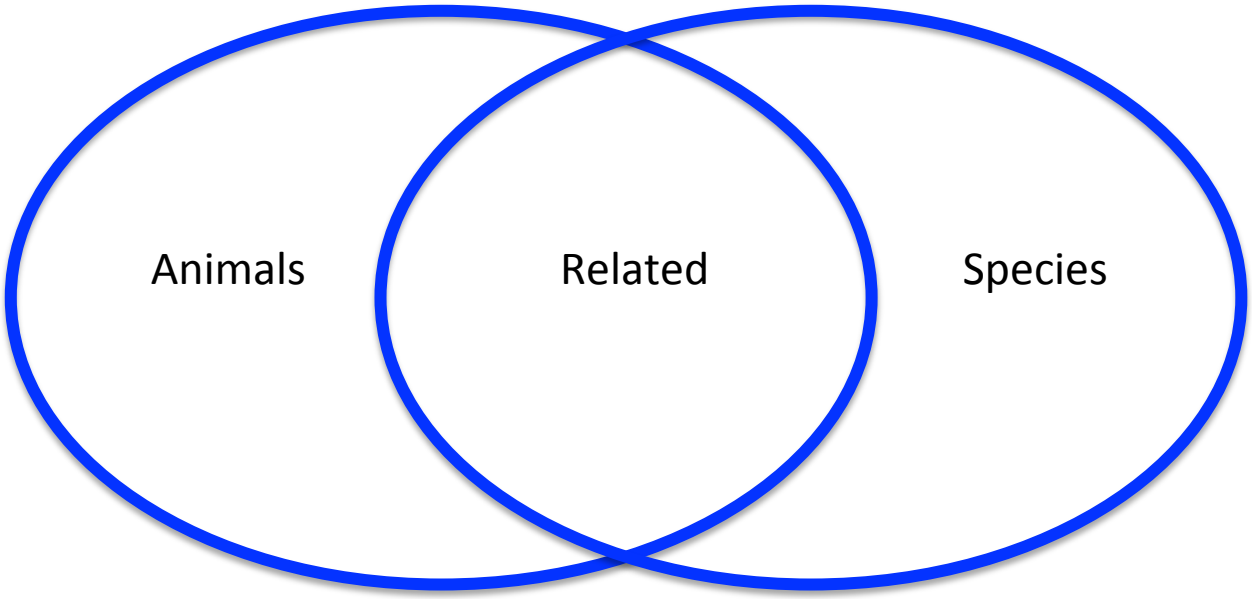
Join Example

1	Suzy	22	12
7	**Chipper**	**25**	**6**
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
37	**Fish**
45	Hamster

Unrelated rows… how do we handle these?

Join Types



Cross Join

LETTERS	NUMBERS
A	1
B	2
C	3
D	

Cross Join

LETTERS	NUMBERS
A	1
B	2
C	3
D	

Select * from letters, numbers

Cross Join

LETTERS	NUMBERS
A	1
B	2
C	3
D	

Select * from letters, numbers

A 1
A 2
A 3
B 1
B 2
B 3
C 1
C 2
C 3
D 1
D 2
D 3

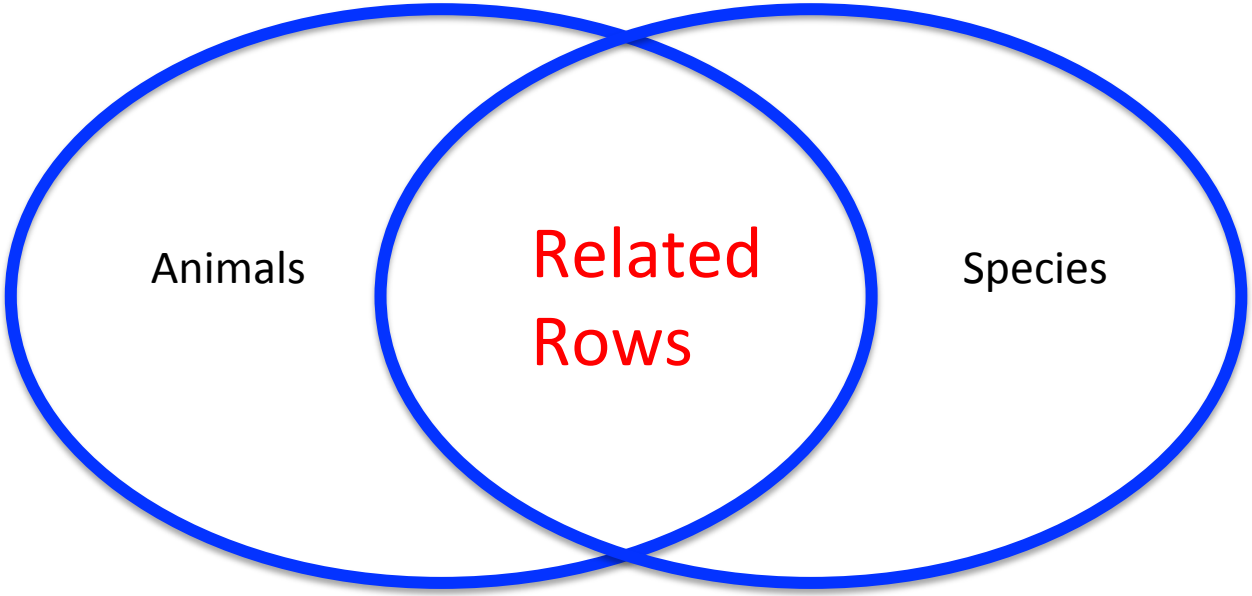
Cross Join

LETTERS	NUMBERS
A	1
B	2
C	3
D	

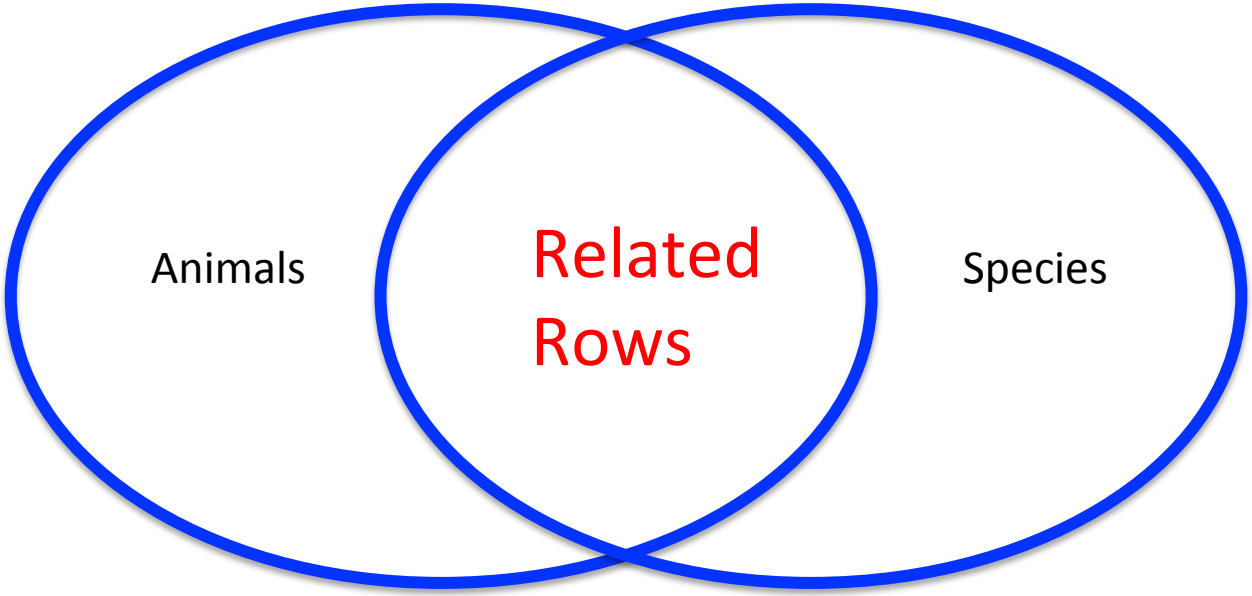
Select * from letters, numbers where n <= 2

A 1
A 2
B 1
B 2
C 1
C 2
D 1
D 2

Inner Join



Inner Join



Suzy	Dog
Sandy	Cat
Whiskers	Hamster
Heidi	Dog

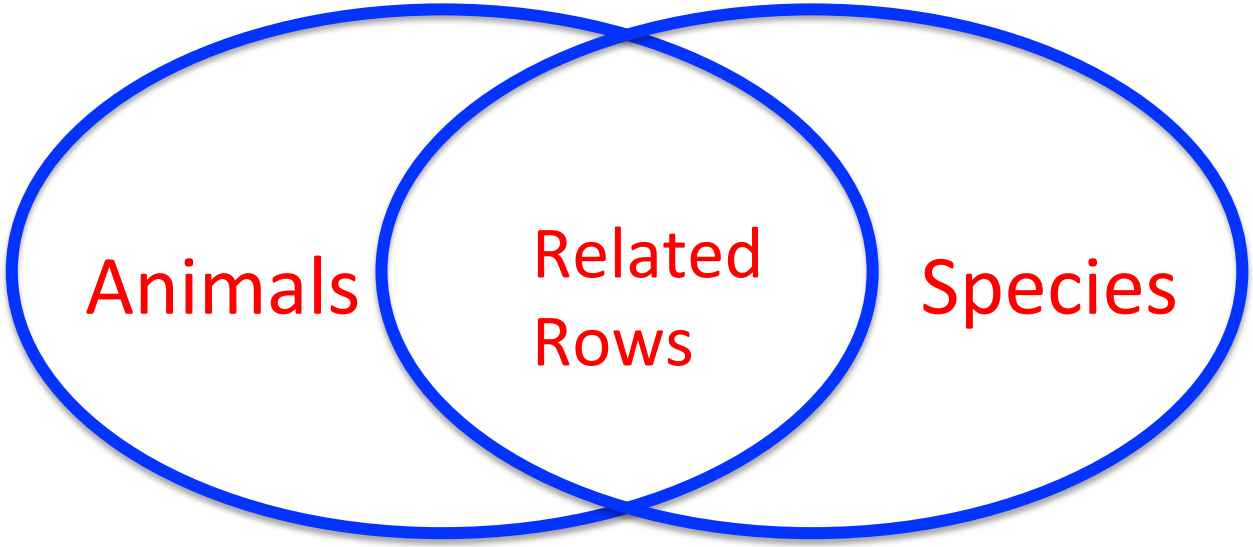
Join Example

1	Suzy	22	12
7	**Chipper**	**25**	**6**
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

22	Dog
31	Cat
37	**Fish**
45	Hamster

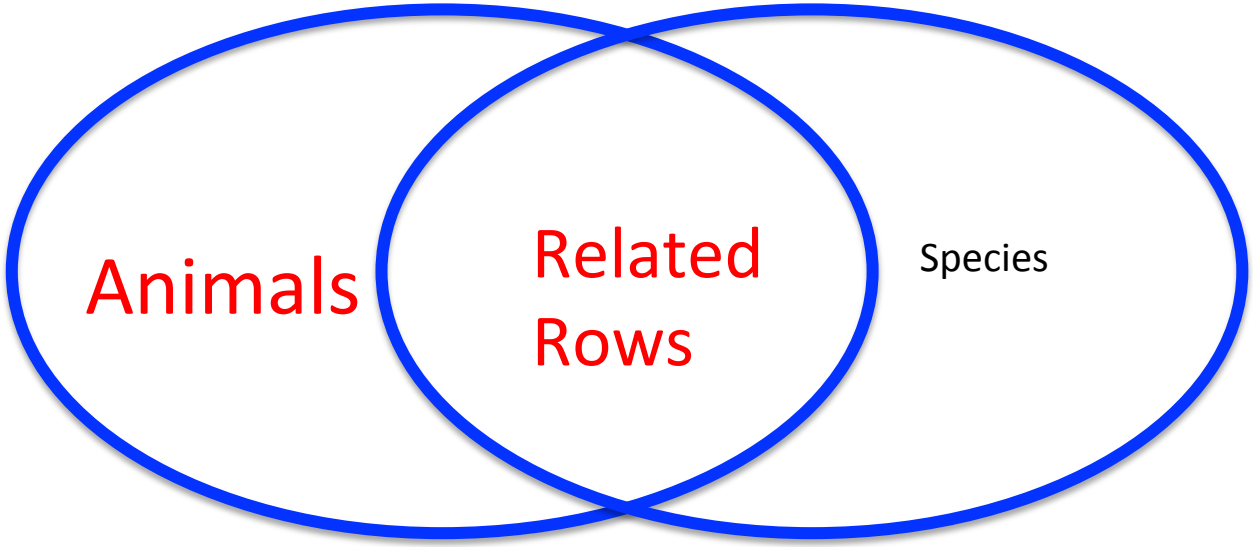
Unrelated rows…

Outer Join



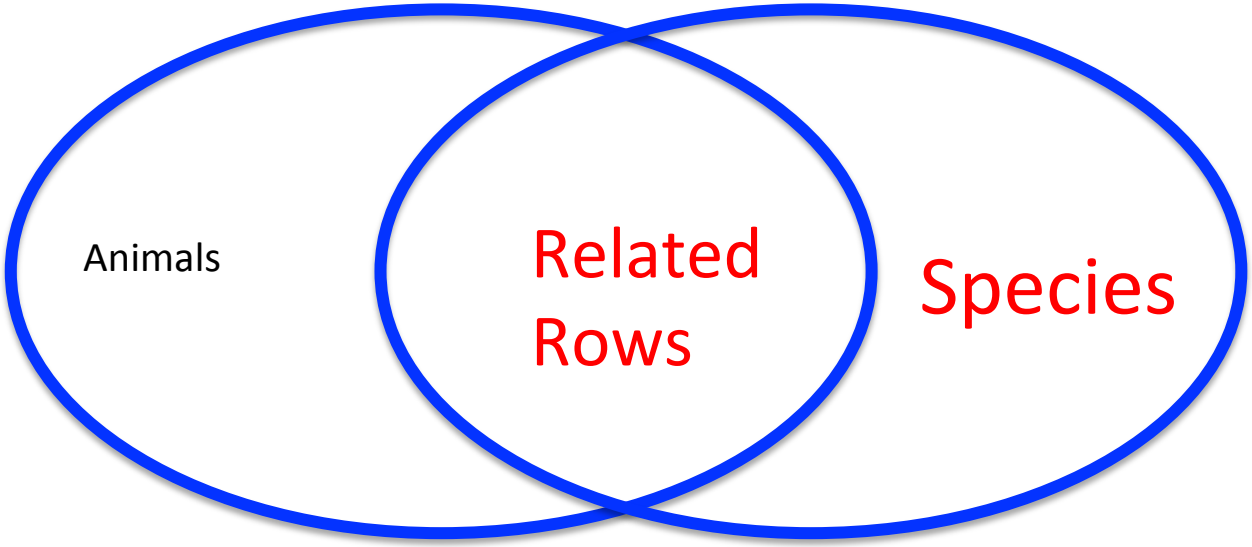
Suzy	Dog
Sandy	Cat
Whiskers	Hamster
<null>	Fish
Chipper	<null>
Heidi	Dog

Left Outer Join



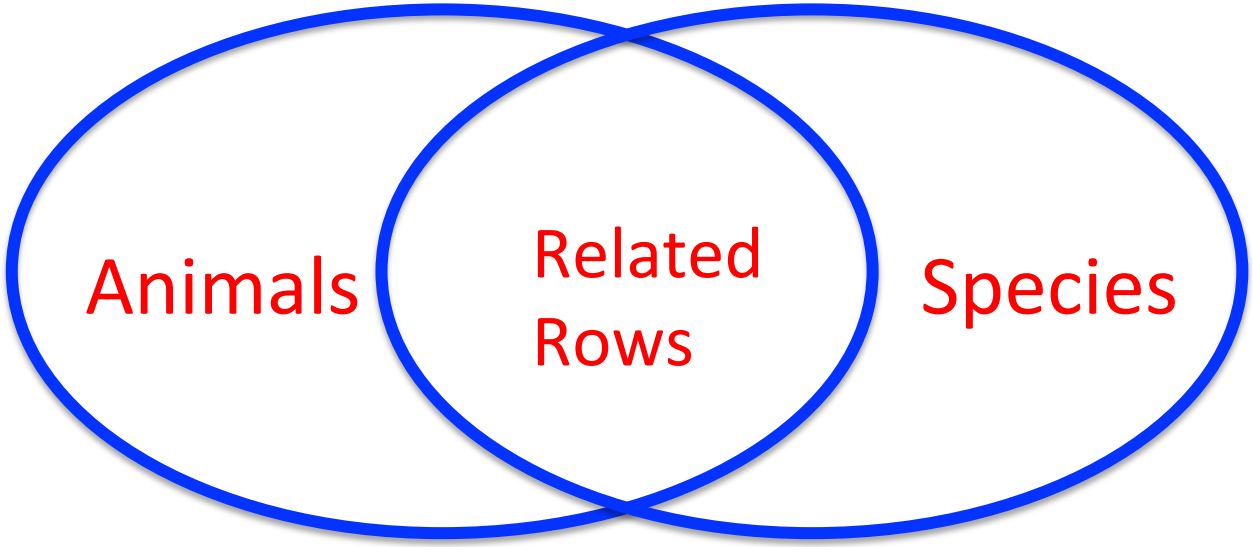
Suzy	Dog
Sandy	Cat
Whiskers	Hamster
Chipper	<null>
Heidi	Dog

Right Outer Join



Suzy	Dog
Sandy	Cat
Whiskers	Hamster
<null>	Fish
Heidi	Dog

(Full) Outer Join



Suzy	Dog
Sandy	Cat
Whiskers	Hamster
<null>	Fish
Chipper	<null>
Heidi	Dog

SEARCH OPTIMIZATION

Looking at Query Optimization

- Tables vs Indices
- A table is a set of records
- An index is a tree (usually) that can be searched
  - For a single field
  - For multiple fields
- The tree result is an index (i.e. row number)
- It matters a great deal if indices are created
- Good examples online for SQLIte

Optimizer Strategies

rowid	fruit	state	price
1	Orange	FL	0.85
2	Apple	NC	0.45
4	Peach	SC	0.60
5	Grape	CA	0.80
18	Lemon	FL	1.25
19	Strawberry	NC	2.45
23	Orange	CA	1.05

Figure 1: Logical Layout Of Table "FruitsForSale"

Full Table Scan

```
SELECT price FROM fruitsforsale WHERE fruit='Peach';
```

rowid	fruit	state	price
1	Orange	FL	0.85
2	Apple	NC	0.45
4	Peach	SC	0.60
5	Grape	CA	0.80
18	Lemon	FL	1.25
19	Strawberry	NC	2.45
23	Orange	CA	1.05

Figure 2: Full Table Scan

Row Number Fetch

```
SELECT price FROM fruitsforsale WHERE rowid=4;
```

rowid	fruit	state	price
1	Orange	FL	0.85
2	Apple	NC	0.45
4	Peach	SC	0.60
5	Grape	CA	0.80
18	Lemon	FL	1.25
19	Strawberry	NC	2.45
23	Orange	CA	1.05

Figure 3: Lookup By Rowid

Create an Index

```
CREATE INDEX idx1 ON fruitsforsale(fruit);
```

fruit	rowid
Apple	2
Grape	5
Lemon	18
Orange	1
Orange	23
Peach	4
Strawberry	19

Figure 4: An Index On The Fruit Column

Using an Index

```
SELECT price FROM fruitsforsale WHERE fruit='Peach';
```

fruit	rowid
Apple	2
Grape	5
Lemon	18
Orange	1
Orange	23
Peach	4
Strawberry	19

rowid	fruit	state	price
1	Orange	FL	0.85
2	Apple	NC	0.45
4	Peach	SC	0.60
5	Grape	CA	0.80
18	Lemon	FL	1.25
19	Strawberry	NC	2.45
23	Orange	CA	1.05

Figure 5: Indexed Lookup For The Price Of Peaches

Using an Index

```
SELECT price FROM fruitsforsale WHERE fruit='Orange'
```

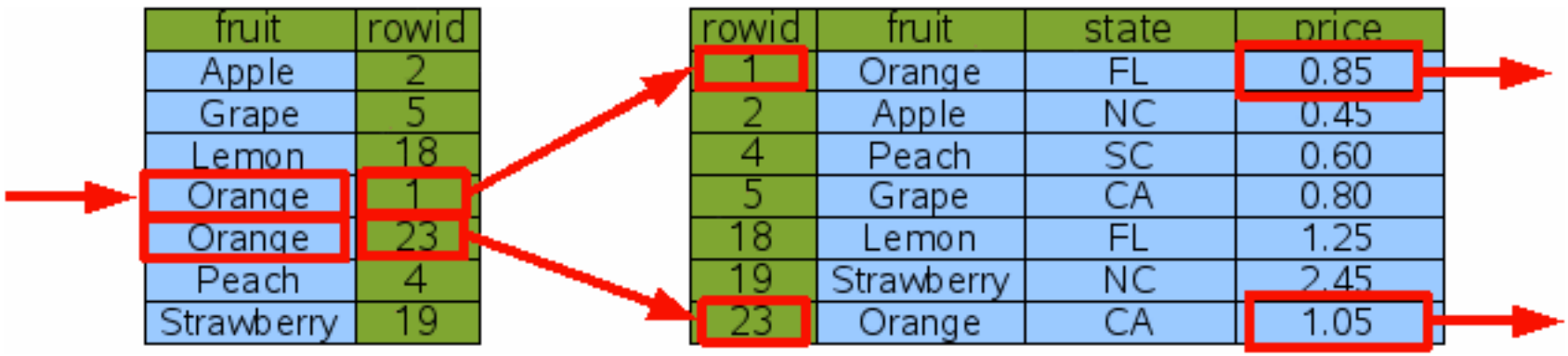


Figure 6: Indexed Lookup For The Price Of Oranges

Using an Index

```
SELECT price FROM fruitsforsale WHERE fruit='Orange' AND state='CA'
```

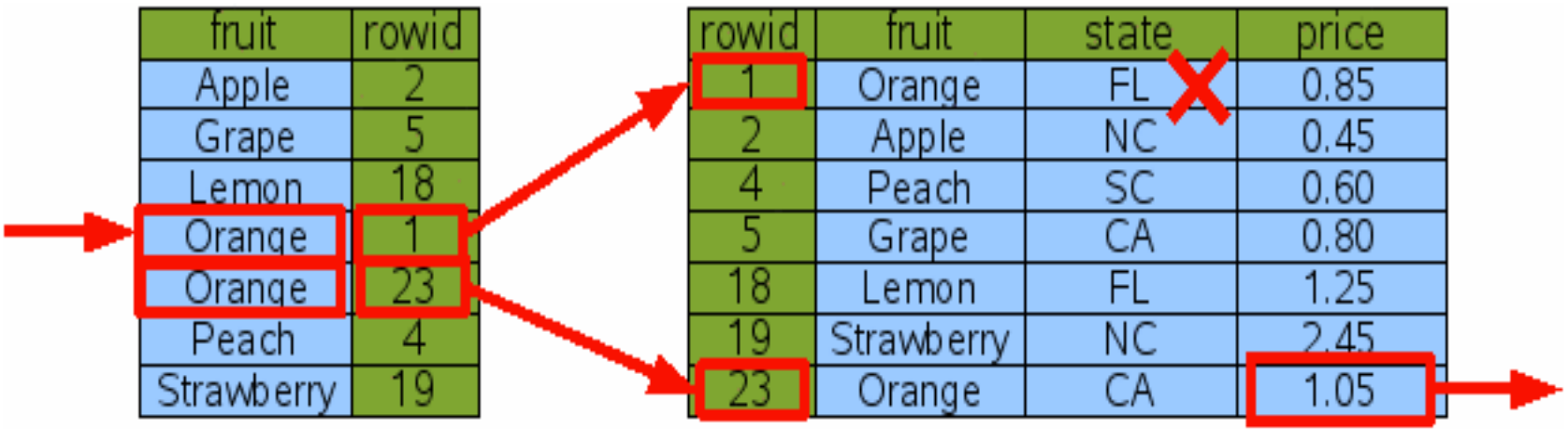


Figure 7: Indexed Lookup Of California Oranges

Creating an Alternate Index

```
CREATE INDEX Idx2 ON fruitsforsale(state);
```

state	rowid
CA	5
CA	23
FL	1
FL	18
NC	2
NC	19
SC	4

Figure 8: Index On The State Column

Using an Alternate Index

state	rowid
CA	5
CA	23
FL	1
FL	18
NC	2
NC	19
SC	4

rowid	fruit	state	price
1	Orange	FL	0.85
2	Apple	NC	0.45
4	Peach	SC	0.60
5	Grape	CA	0.80
18	Lemon	FL	1.25
19	Strawberry	NC	2.45
23	Orange	CA	1.05

Figure 9: Indexed Lookup Of California Oranges

Creating a Multiple Column Index

```
CREATE INDEX Idx3 ON FruitsForSale(fruit, state);
```

fruit	state	rowid
Apple	NC	2
Grape	CA	5
Lemon	FL	18
Orange	CA	23
Orange	FL	1
Peach	SC	4
Strawberry	NC	19

Figure 1: A Two-Column Index

Using a Multiple Column Index

```
SELECT price FROM fruitsforsale WHERE fruit='Orange' AND state='CA'
```

fruit	state	rowid
Apple	NC	2
Grape	CA	5
Lemon	FL	18
Orange	CA	23
Orange	FL	1
Peach	SC	4
Strawberry	NC	19

rowid	fruit	state	price
1	Orange	FL	0.85
2	Apple	NC	0.45
4	Peach	SC	0.60
5	Grape	CA	0.80
18	Lemon	FL	1.25
19	Strawberry	NC	2.45
23	Orange	CA	1.05

Figure 11: Lookup Using A Two-Column Index

Using a Multiple Column Index

```
SELECT price FROM fruitsforsale WHERE fruit='Peach'
```

fruit	state	rowid
Apple	NC	2
Grape	CA	5
Lemon	FL	18
Orange	CA	23
Orange	FL	1
Peach	SC	4
Strawberry	NC	19

rowid	fruit	state	price
1	Orange	FL	0.85
2	Apple	NC	0.45
4	Peach	SC	0.60
5	Grape	CA	0.80
18	Lemon	FL	1.25
19	Strawberry	NC	2.45
23	Orange	CA	1.05

Figure 12: Single-Column Lookup On A Multi-Column Index

Using Index Values

- Where clause can search using:

```
column = expression
column > expression
column >= expression
column < expression
column <= expression
expression = column
expression > column
expression >= column
expression < column
expression <= column
column IN (expression-list)
column IN (subquery)
column IS NULL
```

- > create index *myindex* on *mytable*(a,b…)

Creating a Covering Index

```
CREATE INDEX Idx4 ON FruitsForSale(fruit, state, price);
```

fruit	state	price	rowid
Apple	NC	0.45	2
Grape	CA	0.80	5
Lemon	FL	1.25	18
Orange	CA	1.05	23
Orange	FL	0.85	1
Peach	SC	0.60	4
Strawberry	NC	2.45	19

Figure 13: A Covering Index

Using a Covering Index

```
SELECT price FROM fruitsforsale WHERE fruit='Orange' AND state='CA';
```

fruit	state	price	rowid
Apple	NC	0.45	2
Grape	CA	0.80	5
Lemon	FL	1.25	18
Orange	CA	1.05	23
Orange	FL	0.85	1
Peach	SC	0.60	4
Strawberry	NC	2.45	19

Figure 14: Query Using A Covering Index

Using an OR search

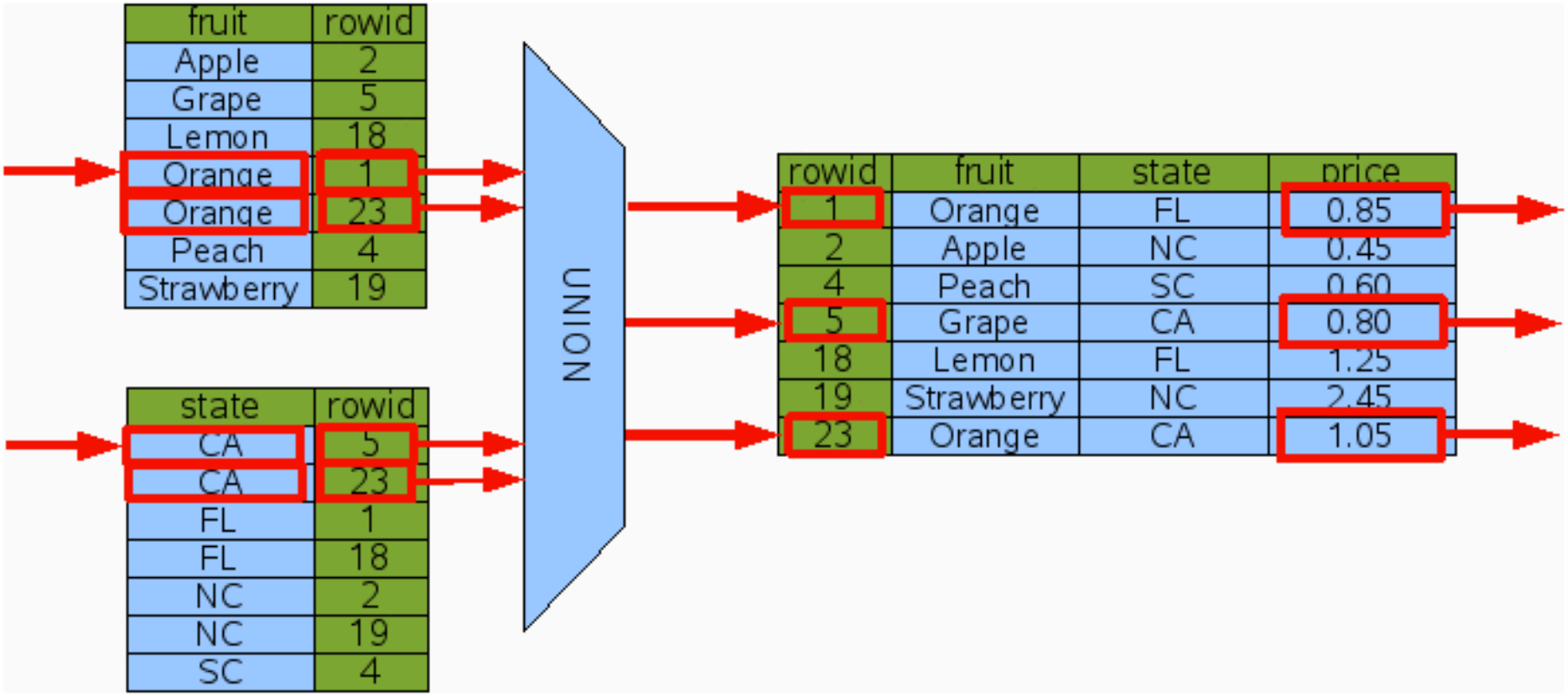


Figure 15: Query With OR Constraints

Sorting

```
SELECT * FROM fruitsforsale ORDER BY fruit;
```

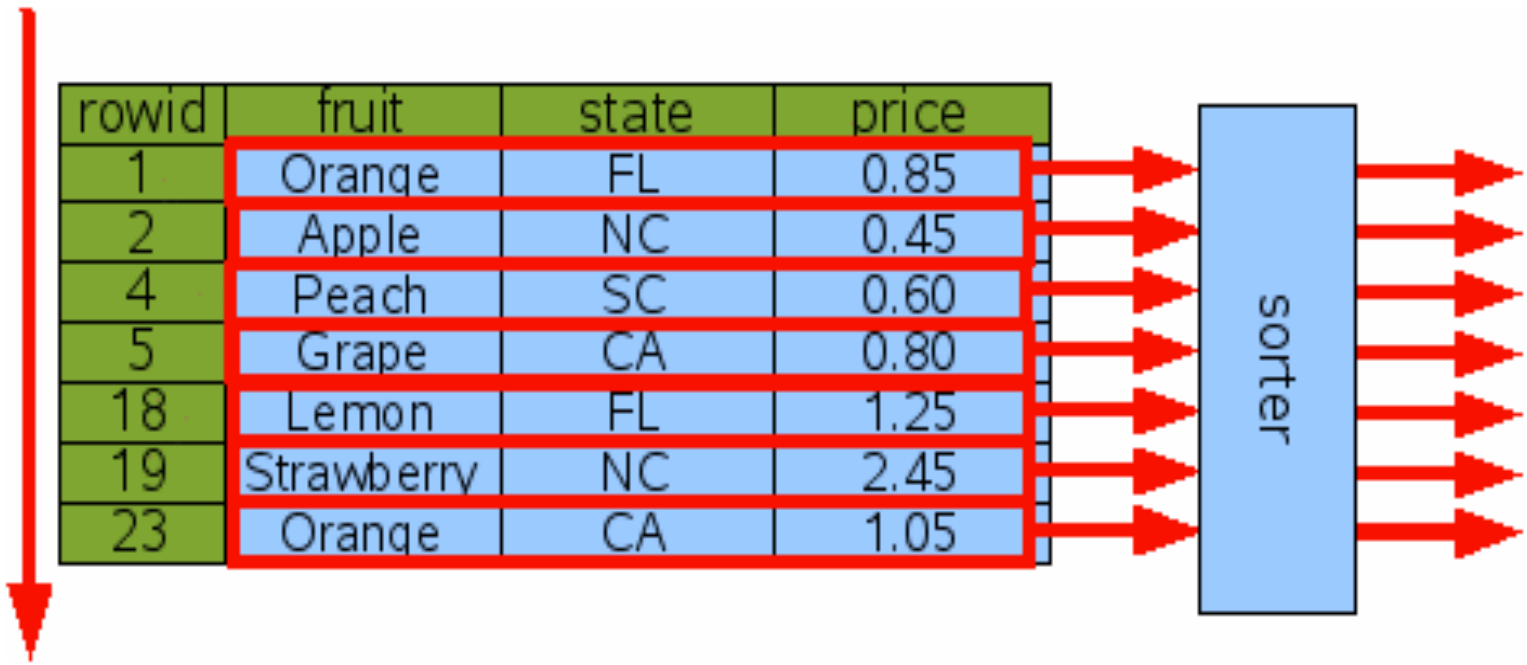
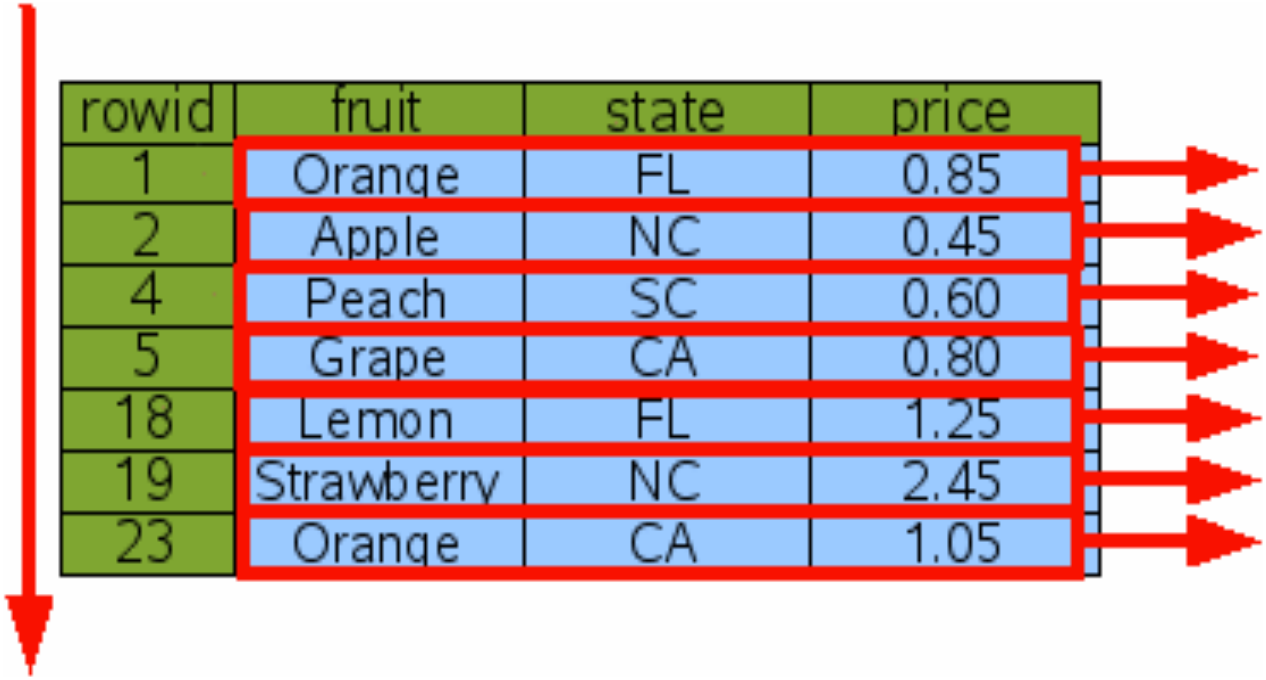


Figure 16: Sorting Without An Index

Sorting

```
SELECT * FROM fruitsforsale ORDER BY rowid;
```



rowid	fruit	state	price
1	Orange	FL	0.85
2	Apple	NC	0.45
4	Peach	SC	0.60
5	Grape	CA	0.80
18	Lemon	FL	1.25
19	Strawberry	NC	2.45
23	Orange	CA	1.05

Figure 17: Sorting By Rowid

Sorting

```
SELECT * FROM fruitsforsale ORDER BY fruit;
```

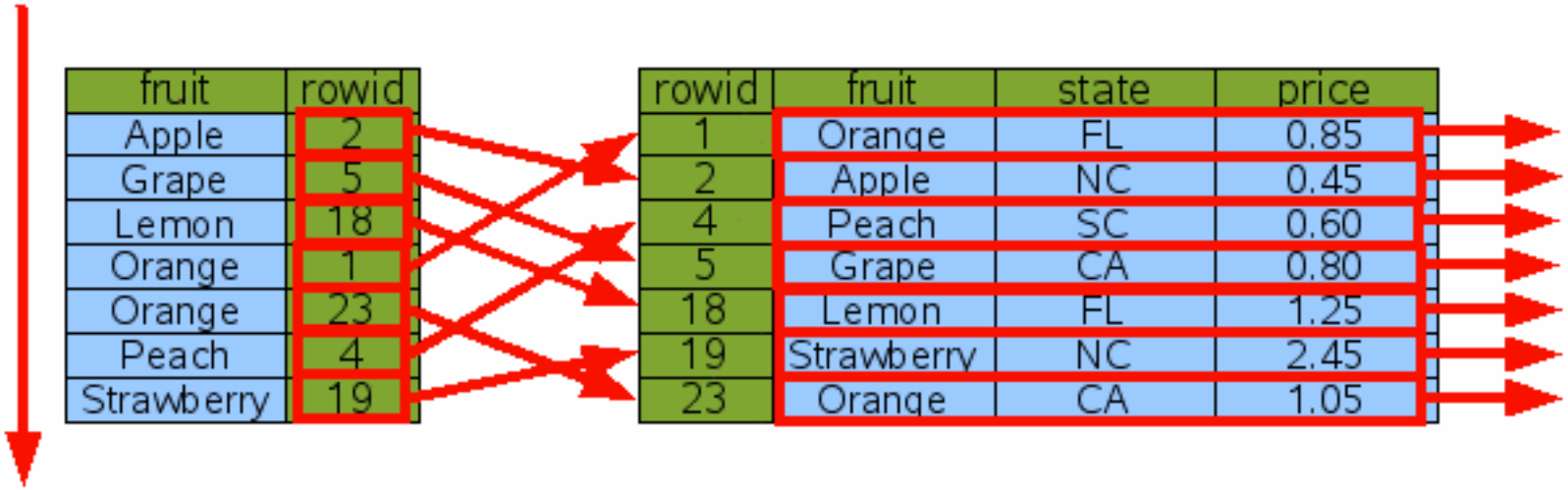


Figure 18: Sorting With An Index

Sorting

fruit	state	price	rowid
Apple	NC	0.45	2
Grape	CA	0.80	5
Lemon	FL	1.25	18
Orange	CA	1.05	23
Orange	FL	0.85	1
Peach	SC	0.60	4
Strawberry	NC	2.45	19

Figure 19: Sorting With A Covering Index

Partial Sorting

```
SELECT * FROM fruitforsale ORDER BY fruit, price
```

Figure 22: Partial Sort By Index

Optimizer Strategies

- SQLite uses query strategies
  - http://www.sqlite.org/queryplanner.html
- SQLite planner does optimization
  - Uses strategies to decide how to search
  - ttps://www.sqlite.org/optoverview.html
- Query plans come with explanations
  - http://www.sqlite.org/eqp.html

OBJECT RELATIONAL MAPPING

Object Relational Mapper - Review

Creates a map between data and objects
Program in terms of your application objects

Data:

<name> <address> <phone>

Object:

```
Class Person:
    name = ‘ ‘
    address = ‘ ‘
    phone = ‘ ‘
```

Object Relational Mapper

Creates a map between operations:

- Create table -> Object class initialization
- Create record -> Object initialization
- Update, Delete record -> Object methods
- Searches -> Class methods using object descriptions
- Key enforcement -> Relationship management

ACID CRITERIA

ACID CRITERIA

Common properties of relational databases

- Atomic
- Consistent
- Isolated
- Durable

https://en.wikipedia.org/wiki/ACID

Atomic

- All or nothing
- No partial transactions

Consistent

- All data meets constraints
- No violation states

Isolated

- Concurrent transactions cant see each other's stats
- There must be a sequential equivalent

Durable

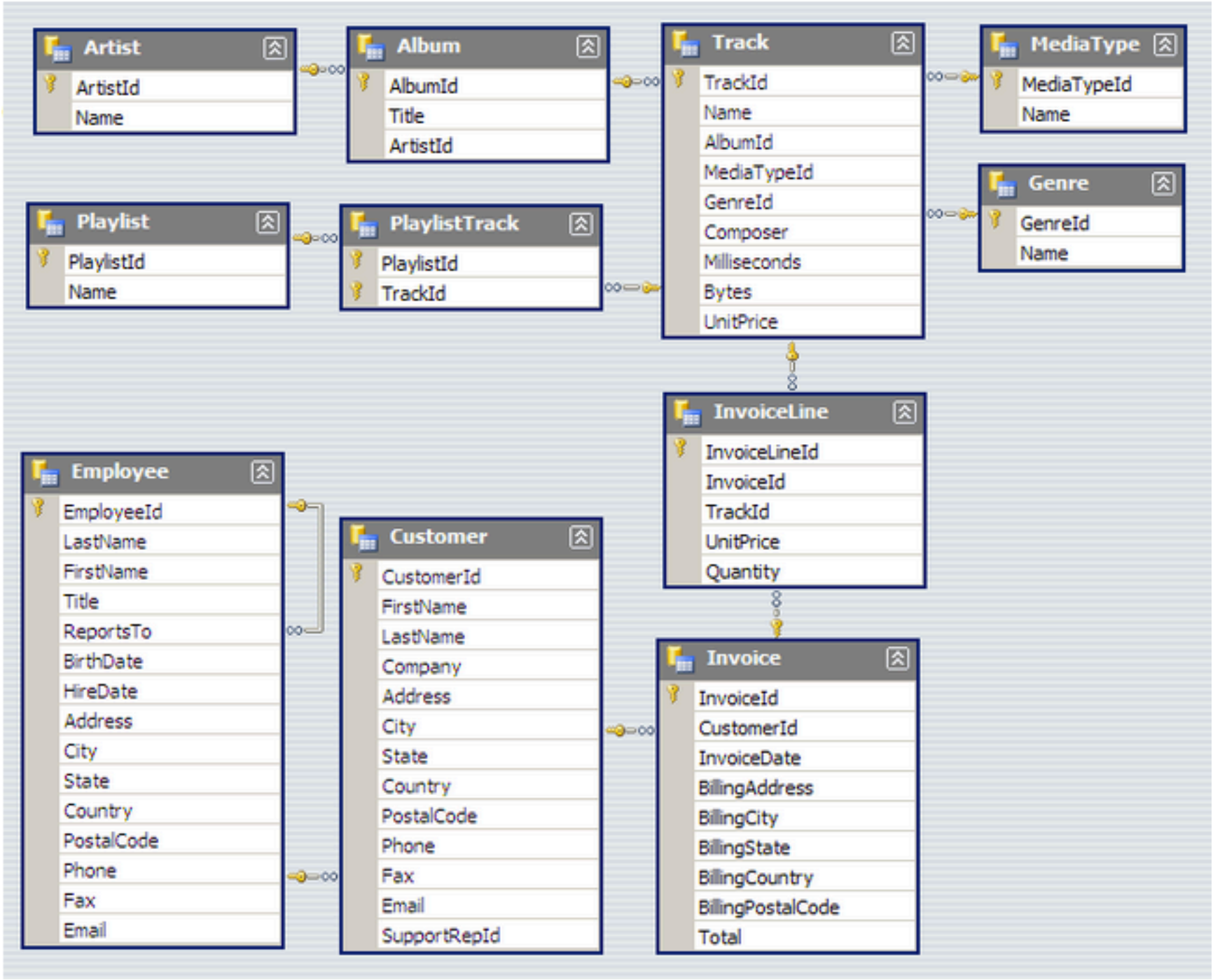
- Any committed transaction must be permanent
- No excuses
- Not even power outages or crashes

CHINOOK EXAMPLE DATABASE

An Example Database

- Chinook – Music Store database
  - Reasonable example of a relational databases
  - Tunes, tracks, albums, artists, genres, customers
  - Available in lots of formats

- https://chinookdatabase.codeplex.com/


Artist
ArtistId
Name
Album
AlbumId
Title
ArtistId
Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice
MediaType
MediaTypeId
Name
Genre
GenreId
Name
Playlist
PlaylistId
Name
PlaylistTrack
PlaylistId
TrackId
InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity
Employee
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email
Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId
Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total