Relational Algebra and Calculus: Introduction to SQL

University of California, Berkeley

School of Information

IS 257: Database Management

Announcements

- Queries/Comments/Concerns?
- Everyone enrolled? Come see me at break.
- Assignment 1 and 2a questions?
- Everybody on Piazza?
- Class Structure
  - Lecture
  - Lab MySQL walkthrough and debugging local install
  - https://github.com/munners17/INFO257-Sp2019/blob/master/diveshop/Postgres%20Diveshop%20Restore.md

Readings So Far

- Past: DB Environment and Development Process, Modeling Data in the Organization, Enhanced E-R Model, OO Data Modeling
- Current: Intro to SQL, Advanced SQL

Lecture Outline

- The Relational Model Revisited
- Relational Algebra
- Relational Calculus
- Introduction to SQL

Data Models(2): History

- Relational Model (1980's)
  - Provides a conceptually simple model for data as relations (typically considered "tables") with all data visible.

Book ID	Title	pubid	Author id
1	Introductio	2	1
2	The history	4	2
3	New stuff a	3	3
4	Another tit	2	4
5	And yet m	1	5

pubid	pubname
1	Harper
2	Addison
3	Oxford
4	Que

Authorid	Author nan
1	Smith
2	Wynar
3	Jones
4	Duncan
5	Applegate

Book ID	Subid
1	2
2	1
3	3
4	2
4	3

Subid	Subject
1	cataloging
2	history
3	stuff

Relational Terminology

- **Relation** (AKA Table)
- **Tuple** (AKA Row in a Table)
- **Domains** (AKA Attributes AKA Columns in a Table)

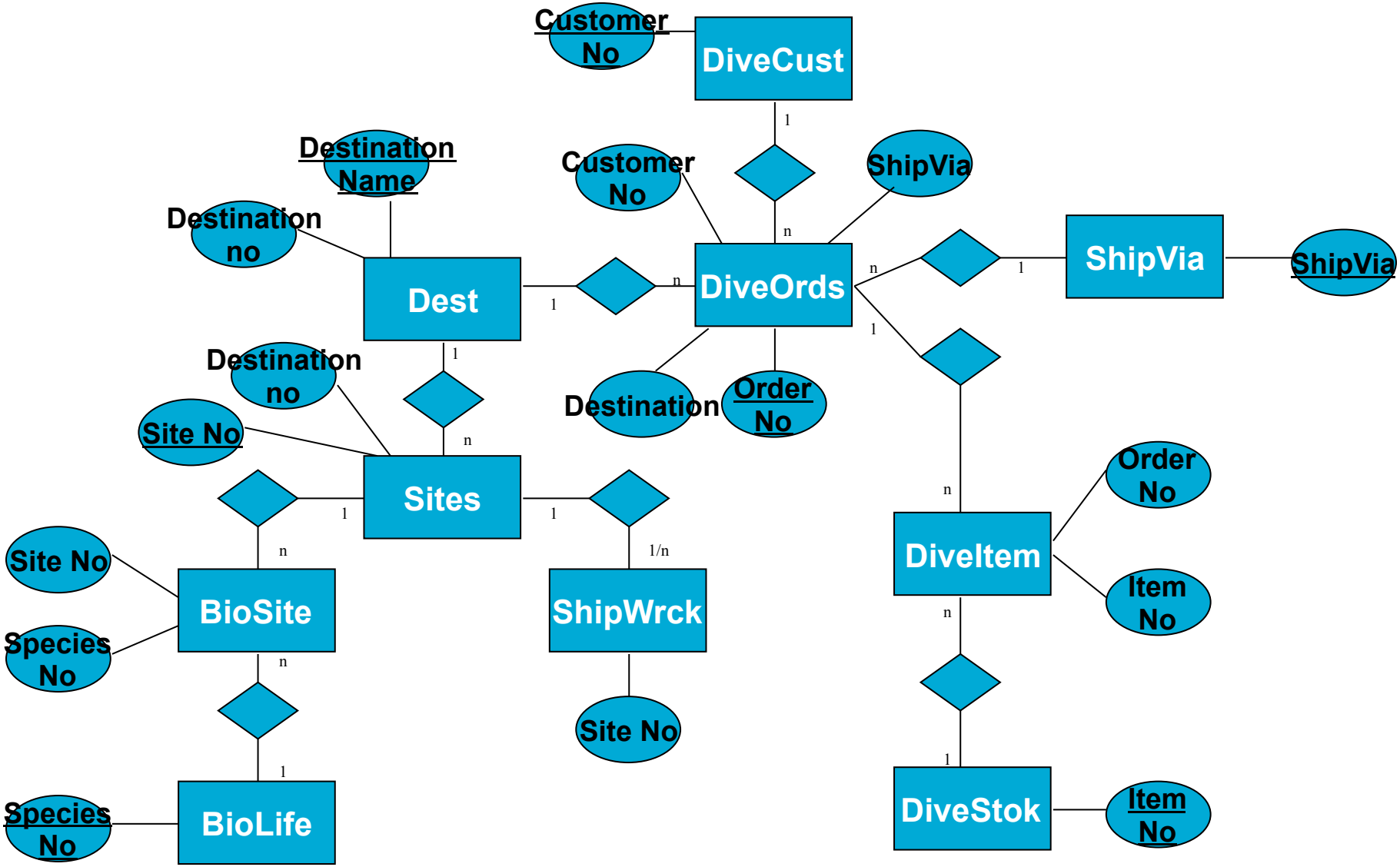
Relation

Tuple

Book ID	Title	pubid	Author id
1	Introductio	2	1
2	The history	4	2
3	New stuff a	3	3
4	Another tit	2	4
5	And yet m	1	5

Domain

DiveShop ER Diagram



Lecture Outline

- The Relational Model Revisited
- Relational Algebra
- Relational Calculus
- Introduction to SQL

Relational Algebra

- Relational Algebra is a collection of operators that take relations as their operands and return a relation as their results
- It's a procedural query language used to query the database tables to access data in different ways.
- First defined by Codd
  - Include 8 operators
    - 4 derived from traditional set operators
    - 4 new relational operations

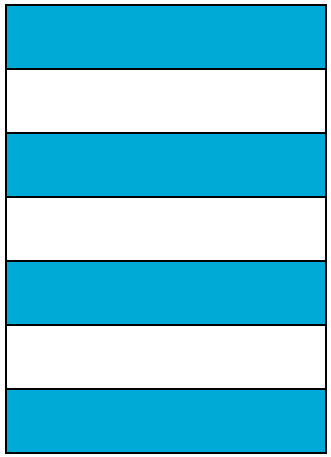
From: C.J. Date, Database Systems 8th ed.

Relational Algebra Operations

- Restrict
- Project
- Product
- Union
- Intersect
- Difference
- Join
- Divide

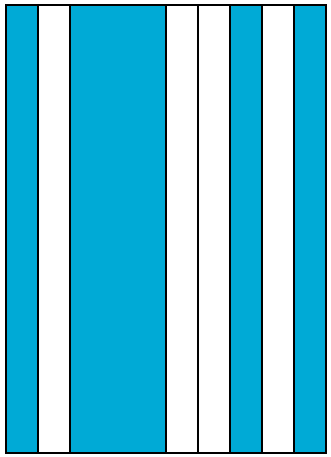
Restrict

- Extracts specified tuples (rows) from a specified relation (table)
  - Restrict is AKA “Select”



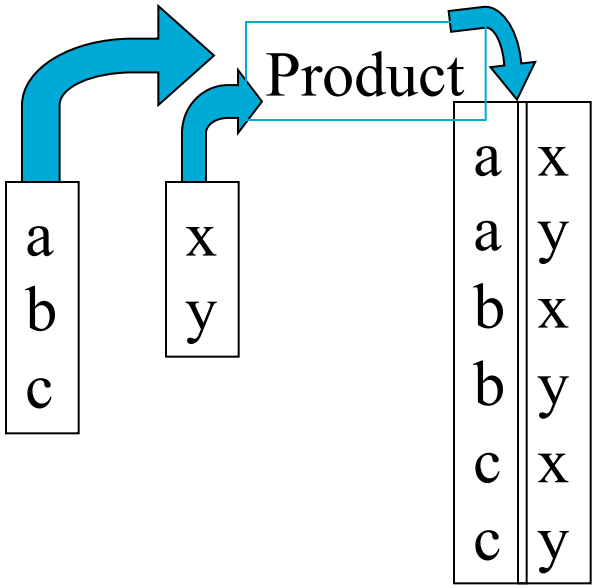
Project

- Extracts specified attributes(columns) from a specified relation.



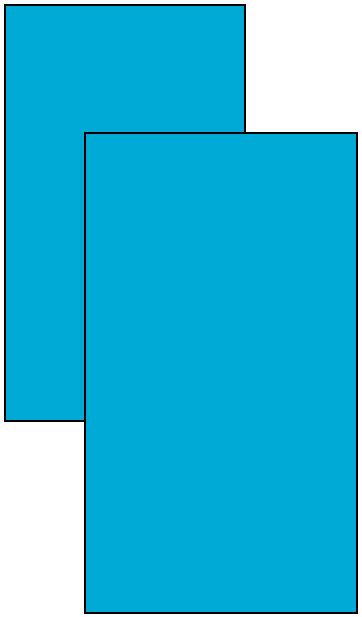
Product

- Builds a relation from two specified relations consisting of all possible concatenated pairs of tuples, one from each of the two relations. (AKA Cartesian Product)



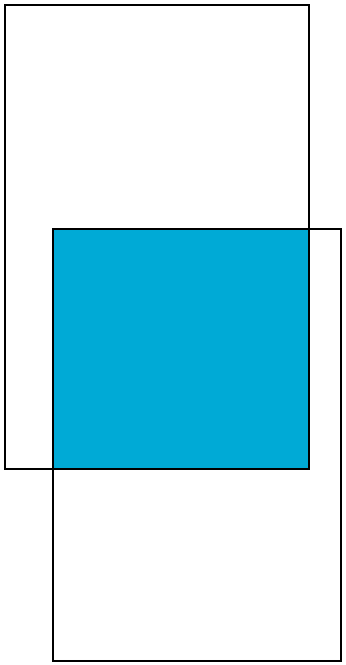
Union

- Builds a relation consisting of all tuples appearing in either or both of two specified relations.



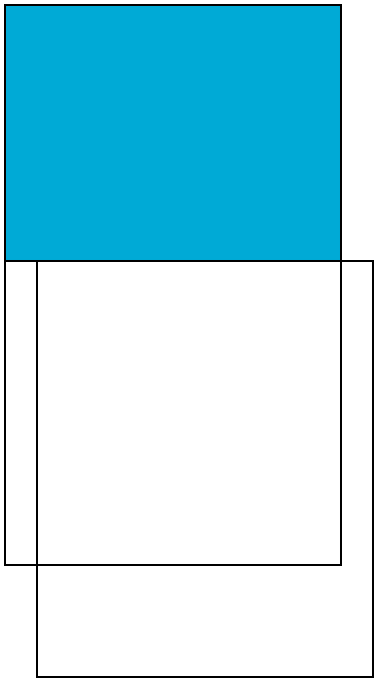
Intersect

- Builds a relation consisting of all tuples appearing in both of two specified relations



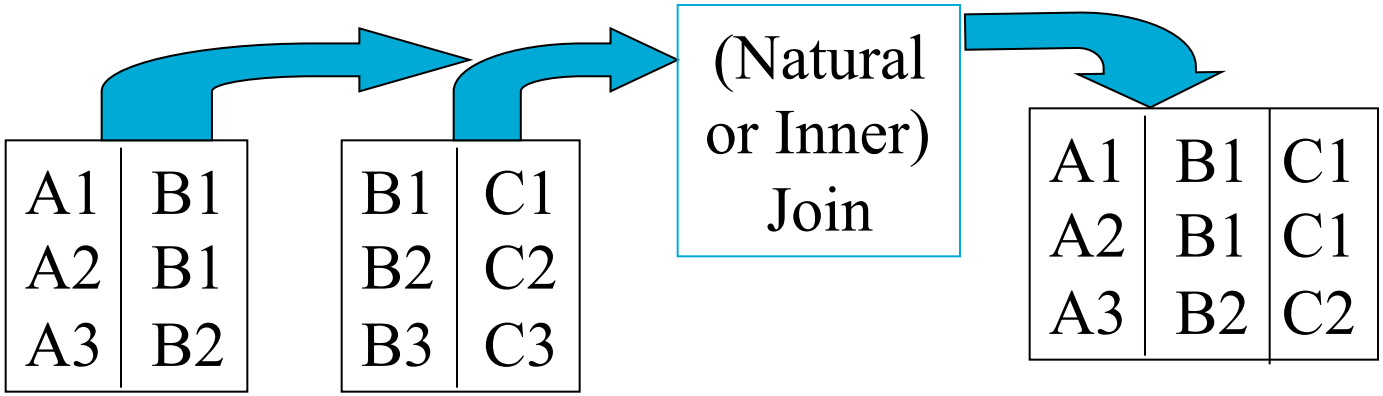
Difference

- Builds a relation consisting of all tuples appearing in first relation but not the second.



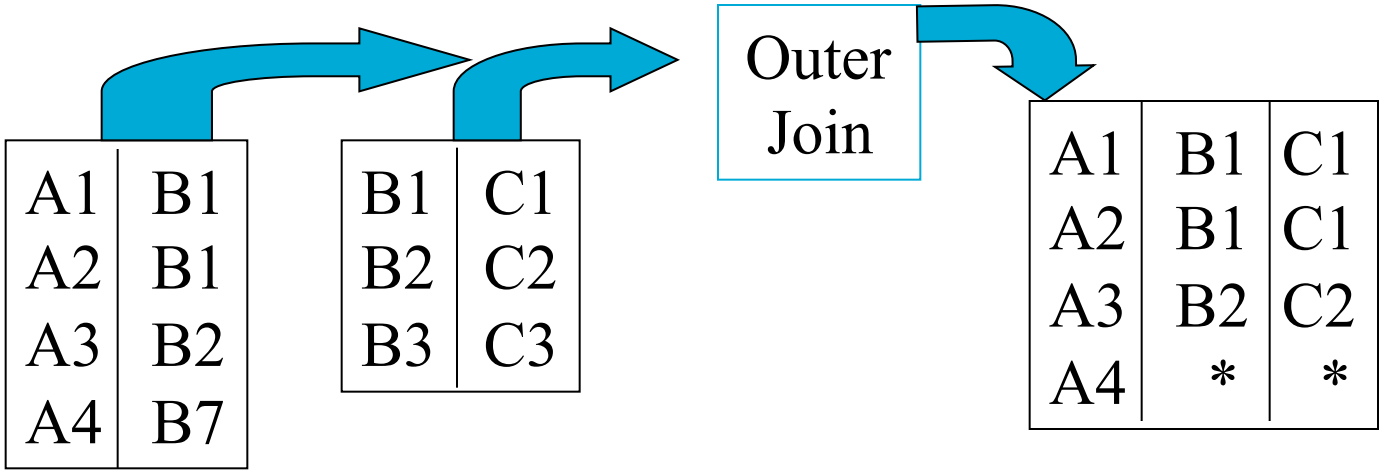
Join

- Builds a relation from two specified relations consisting of all possible concatenated pairs, one from each of the two relations, such that in each pair the two tuples satisfy some condition. (E.g., equal values in a given col.)



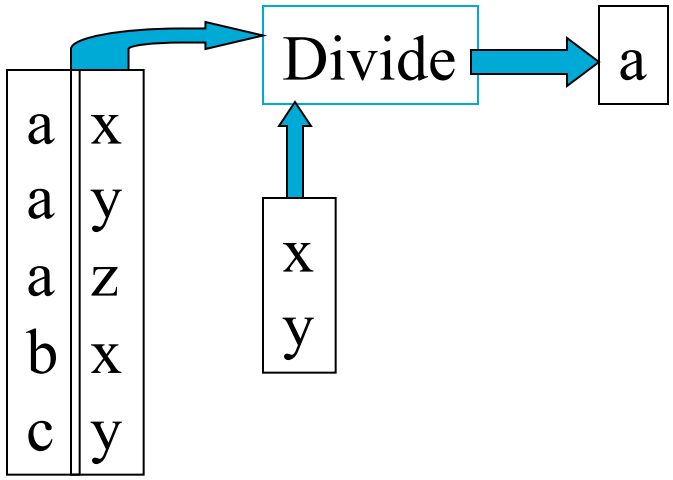
Outer Join

- Outer Joins are similar to PRODUCT -- but will leave NULLs for any row in the first table with no corresponding rows in the second.

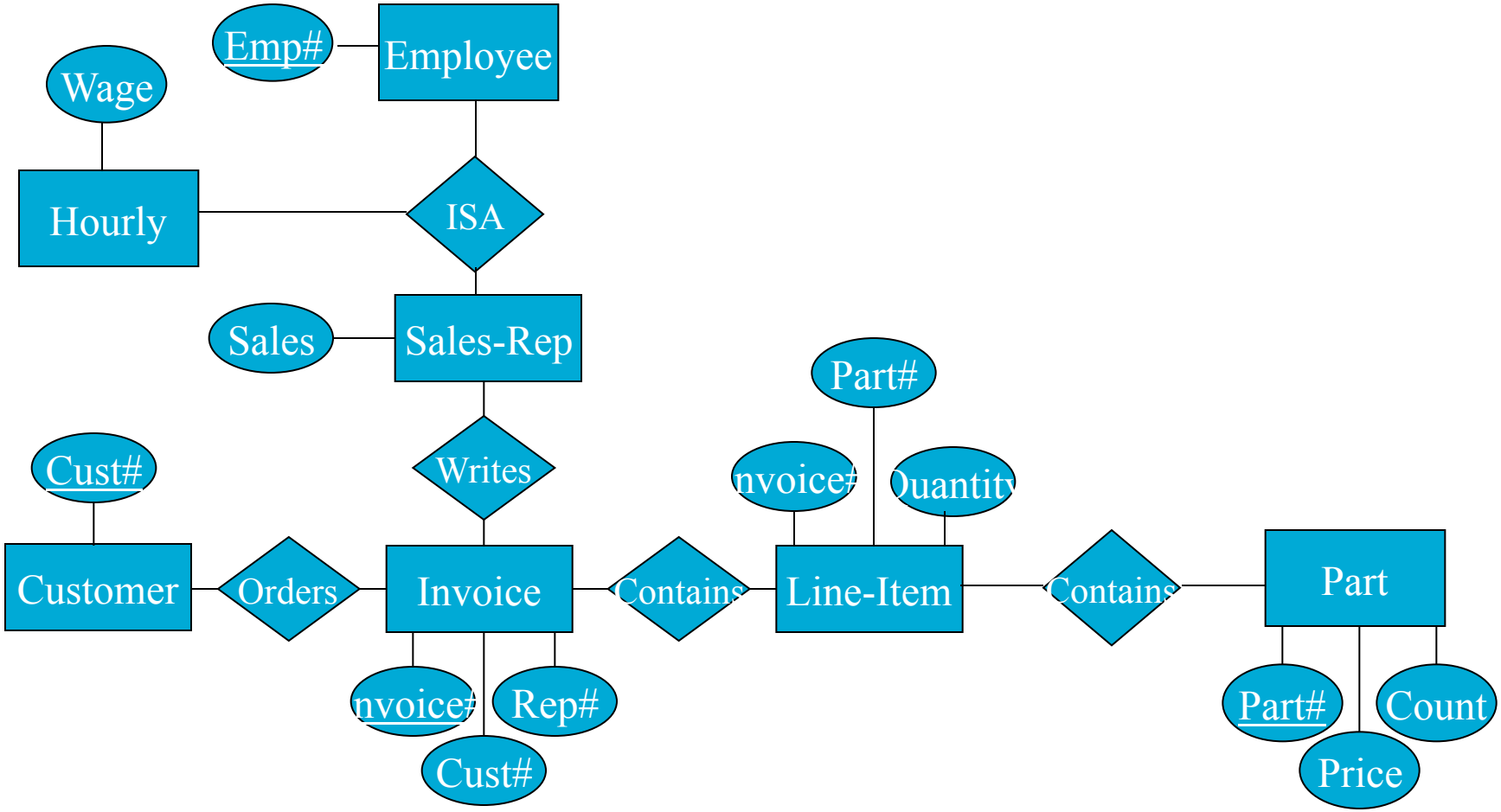


Divide

- Takes two relations, one binary and one unary, and builds a relation consisting of all values of one attribute of the binary relation that match (in the other attribute) ALL values in the unary relation.



ER Diagram: Acme Widget Co.



Employee

SSN	Lastname	Firstname	Middlename	Birthdate	Address
123-76-3423	Jones	Janet	Mary	6/25/1963	234 State
342-88-7865	Smith	Thomas	Frederick	8/4/1970	12 Lambert
486-87-6543	Hendersen	Charles	Robert	9/23/1961	44 Central
843-36-7659	Martinez	Roberto	Garcia	7/8/1958	76 Highland

Part

Part #	Name	Price	Count
1	Big blue widget	3.76	2
2	Small blue Widget	7.35	4
3	Tiny red widget	5.25	7
4	large red widget	157.23	23
5	double widget rack	10.44	12
6	Small green Widge	30.45	58
7	Big yellow widget	7.96	1
8	Tiny orange widget	81.75	42
9	Big purple widget	55.99	9

Sales-Rep

SSN	Rep #	Sales
123-76-3423	1	$12,345.45
843-36-7659	2	$231,456.75

SSN	Wage
342-88-7865	$12.75
486-87-6543	$20.50

Customer

Cust #	COMPANY	STREET1	STREET2	CITY	STATE	ZIPCODE
1	Integrated Standards Ltd.	35 Broadway	Floor 12	New York	NY	02111
2	MegaInt Inc.	34 Bureaucracy Plaza	Floors 1-172	Phildelphia	PA	03756
3	Cyber Associates	3 Control Elevation Place	Cyber Assicates Center	Cyberoid	NY	08645
4	General Consolidated	35 Libra Plaza		Nashua	NH	09242
5	Consolidated MultiCorp	1 Broadway		Middletown	IN	32467
6	Internet Behometh Ltd.	88 Oligopoly Place		Sagrado	TX	78798
7	Consolidated Brands, Inc.	3 Independence Parkway		Rivendell	CA	93456
8	Little Mighty Micro	34 Last One Drive		Orinda	CA	94563

Invoice

Invoice #	Cust #	Rep #
93774	3	1
84747	4	1
88367	5	2
88647	9	1
776879	2	2
65689	6	2

Line-Item

Invoice #	Part #	Quantity		
93774	3	10		
84747	23	1		
88367	75	2		
88647	4	3		
776879	22	5		
65689	76	12		
93774	23	10		
88367	34	2		

Relational Algebra

- What is the name of the customer who ordered Large Red Widgets?
  - Restrict "large Red Widgets" row from Part as *temp1*
  - Join *temp1* with Line-item on Part # as *temp2*
  - Join *temp2* with Invoice on Invoice # as *temp3*
  - Join *temp3* with Customer on cust # as *temp4*
  - Project Company from *temp4* as *answer*

Join Items

Invoice #	Part #	Quantity
93774	3	10
84747	23	1
88367	75	2
88647	4	3
776879	22	5
65689	76	12
93774	23	10
88367	34	2

Part #	Name	Price	Count
1	Big blue widget	3.76	2
2	Small blue Widget	7.35	4
3	Tiny red widget	5.25	7
4	large red widget	157.23	23
5	double widget rack	10.44	12
6	Small green Widge	30.45	58
7	Big yellow widget	7.96	1
8	Tiny orange widget	81.75	42
9	Big purple widget	55.99	9

Invoice #	Cust #	Rep #
93774	3	1
84747	4	1
88647	5	2
88367	9	1
776879	2	2
65689	6	2

Cust #	COMPANY	STREET1	STREET2	CITY	STATE	ZIPCODE
1	Integrated Standards Ltd.	35 Broadway	Floor 12	New York	NY	02111
2	MegaInt Inc.	34 Bureaucracy Plaza	Floors 1-172	Phildelphia	PA	03756
3	Cyber Associates	3 Control Elevation Place	Cyber Assicates Center	Cyberoid	NY	08645
4	General Consolidated	35 Libra Plaza		Nashua	NH	09242
5	Consolidated MultiCorp	1 Broadway		Middletown	IN	32467
6	Internet Behometh Ltd.	88 Oligopoly Place		Sagrado	TX	78798
7	Consolidated Brands, Inc.	3 Independence Parkway		Rivendell	CA	93456
8	Little Mighty Micro	34 Last One Drive		Orinda	CA	94563
9	SportLine Ltd.	38 Champion Place	Suite 882	Compton	CA	95328

Lecture Outline

- The Relational Model Revisited
- Relational Algebra
- Relational Calculus
- Introduction to SQL

Relational Calculus

- Relational Algebra provides a set of explicit operations (select, project, join, etc) that can be used to build some desired relation from the database
- Relational Calculus provides a notation for formulating the definition of that desired relation in terms of the relations in the database without ***explicitly*** stating the operations to be performed
- *SQL is based on the relational calculus **and** algebra*

Lecture Outline

- The Relational Model Revisited
- Relational Algebra
- Relational Calculus
- Introduction to SQL

SQL

- **S**tructured **Q**uery **L**anguage
- Used for both Database Definition, Modification and Querying
- Basic language is standardized across relational DBMS's. Each system may have proprietary extensions to standard.
- Relational Calculus combines Restrict, Project and Join operations in a single command. SELECT.

SQL - History

- QUEL (Query Language from Ingres)
- SEQUEL from IBM San Jose
- ANSI 1992 Standard is the first version used by most DBMS today (SQL92)
- Basic language is standardized across relational DBMSs. Each system may have proprietary extensions to the standard.
- Standard continues to be refined and expanded up to today (more later on)

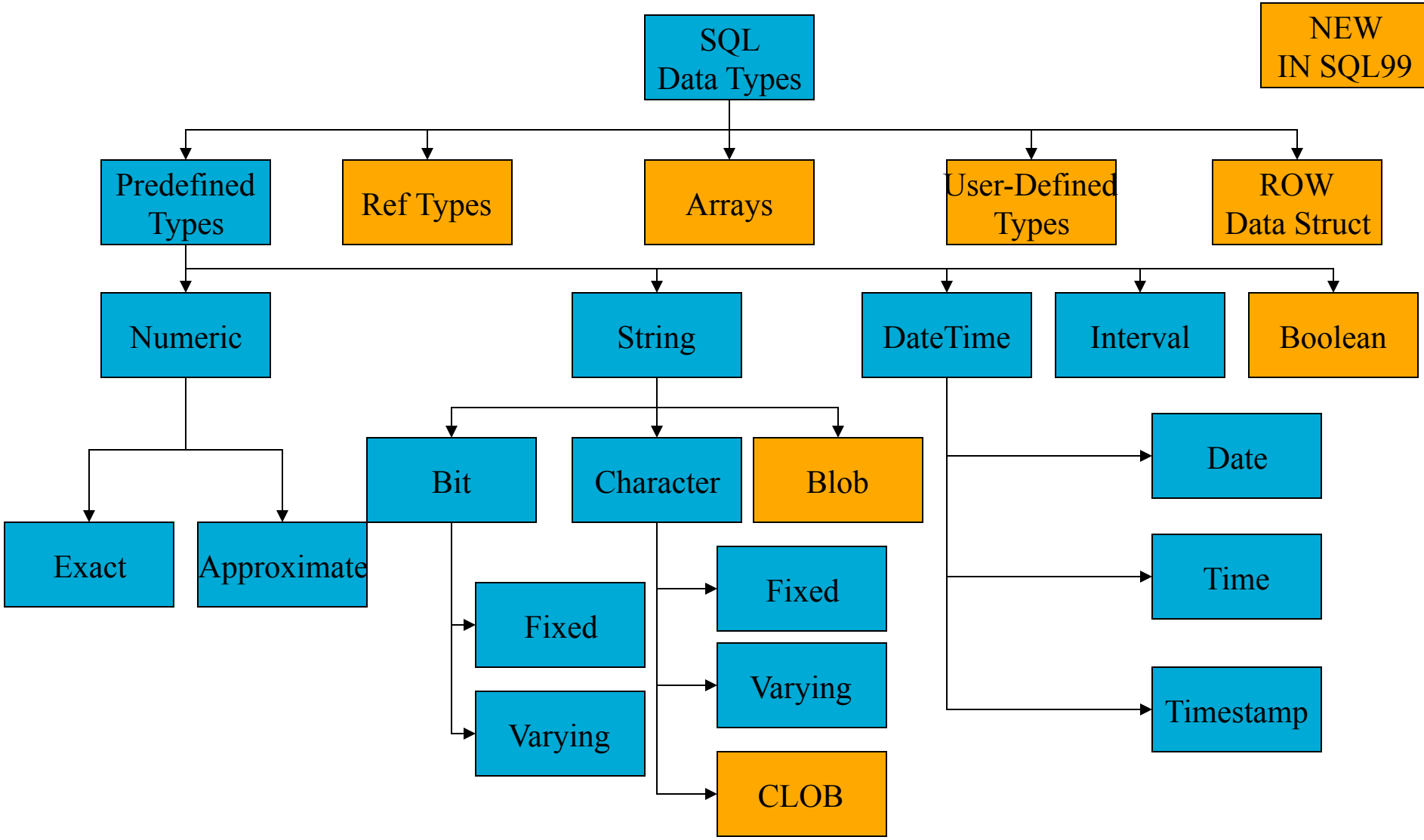
SQL Standard – Advantages

- Reduced training costs
- Productivity
- Application portability
- Application longevity
- Reduced dependence on a single vendor
- Cross-system communication

SQL Standard – Disadvantages

- Stifle creativity
- Offspring of compromises among many parties
- Difficult to change
- Added features may result in loss of portability

SQL99 (Builtin) Data Types



SQL 2016 – New Featues

- Improved JSON functionality
- Row Pattern Recognition: Matching a sequence of rows against a regex.
- Date and time formatting and parsing
- Transform values from rows into delimited string

SQL Uses

- Database Definition and Querying
  - Can be used as an interactive query language
  - Can be imbedded in programs
- Relational Calculus combines Select, Project and Join operations in a single command: SELECT

SELECT

- Syntax:

```
SELECT  [DISTINCT] attr1, attr2,…, attr3
  FROM rel1 r1, rel2 r2,… rel3 r3 WHERE
  condition1 {AND | OR} condition2  ORDER BY
  attr1 [DESC], attr3 [DESC]
```

SELECT

- Syntax:

```
SELECT a.`Author Name`, b.Title FROM
  authors a, books b WHERE a.Authorid =
  b.Authorid  ORDER BY a.`Author name` ;
```

books

Book ID	Title	pubid	Author id
1	Introductio	2	1
2	The history	4	2
3	New stuff a	3	3
4	Another tit	2	4
5	And yet m	1	5

pubid	pubname
1	Harper
2	Addison
3	Oxford
4	Que

authors

Authorid	Author nan
1	Smith
2	Wynar
3	Jones
4	Duncan
5	Applegate

Book ID	Subid
1	2
2	1
3	3
4	2
4	3

Subid	Subject
1	cataloging
2	history
3	stuff

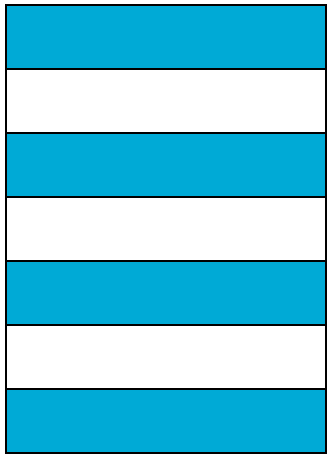
SELECT Conditions

- = equal to a particular value
- >= greater than or equal to a particular value
- > greater than a particular value
- <= less than or equal to a particular value
- <> not equal to a particular value
- LIKE “%term%” (may be other wild cards in other systems)
- IN (“opt1”, “opt2”,…,”optn”)
- BETWEEN val1 AND val2
- IS NULL

Relational Algebra Restrict using SELECT

- Syntax:

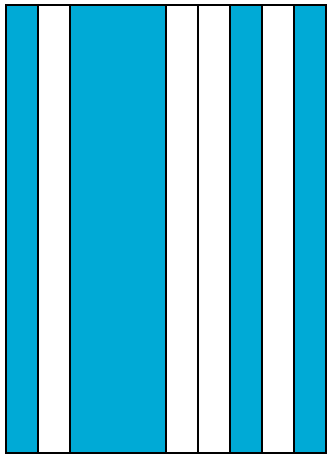
SELECT * WHERE condition1 {AND | OR}
condition2;



Relational Algebra Projection using SELECT

- Syntax:

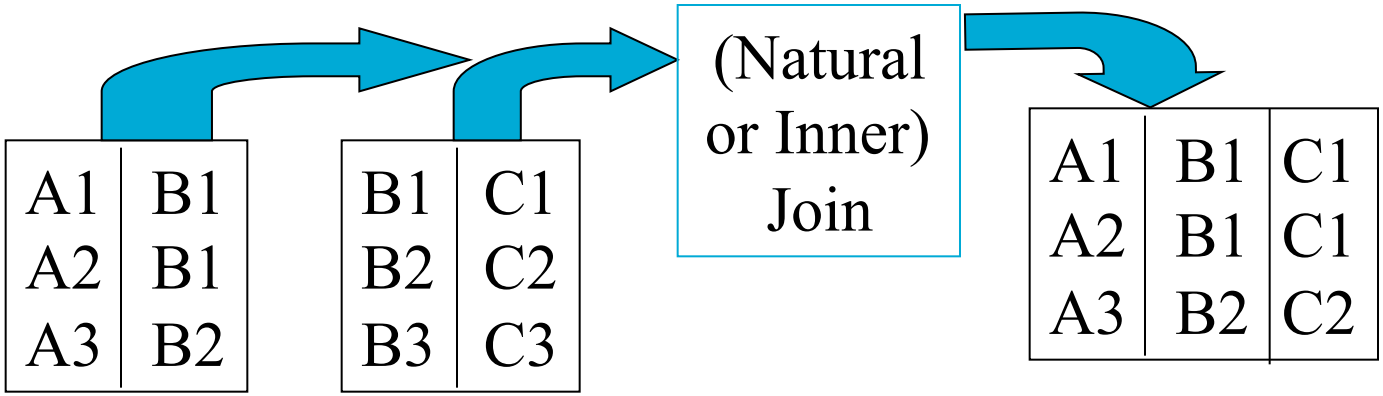
```
SELECT  [DISTINCT] attr1, attr2,…, attr3
  FROM rel1 r1;
```



Relational Algebra Join using SELECT

- Syntax:

```
SELECT  * FROM rel1 r1, rel2 r2 WHERE
  r1.linkattr = r2.linkattr ;
```



Sorting

- SELECT BIOLIFE.Common_Name, BIOLIFE.Length_cm

  FROM BIOLIFE

  ORDER BY BIOLIFE.Length_cm DESC;

Subqueries

- SELECT SITES.Site_Name, SITES.Destination_no FROM SITES

  WHERE sites.Destination_no

  IN (SELECT Destination_no from DEST where Avg_Temp_F >= 78);

- *Can be used as a form of JOIN*

Aggregate Functions

- Count
- Avg
- SUM
- MAX
- MIN
- Many others are available in different systems

Using Aggregate functions

- SELECT attr1, Sum(attr2) AS name
  FROM tab1, tab2 ...
  GROUP BY attr1, attr3 HAVING condition;

Using an Aggregate Function

Implied Joins

- SELECT DIVECUST.Name,
  Sum(Rental_Price*Qty) AS Total
  FROM DIVECUST, DIVEORDS, DIVEITEM,
  DIVESTOK WHERE DIVECUST.Customer_No =
  DIVEORDS.Customer_No AND
  DIVEORDS.Order_No = DIVEITEM.Order_No
  AND DIVEITEM.Item_No = DIVESTOK.Item_No
  GROUP BY DIVECUST.Name
  HAVING ((DIVECUST.Name) LIKE
  '%Jazdzewski%');

Using an Aggregate Function

Explicit Join statements

- SELECT DIVECUST.Name,
  Sum(Rental_Price*Qty) AS Total
  FROM (DIVECUST INNER JOIN DIVEORDS ON
  DIVECUST.Customer_No =
  DIVEORDS.Customer_No) INNER JOIN
  DIVEITEM ON DIVEORDS.Order_No =
  DIVEITEM.Order_No INNER JOIN DIVESTOK ON
  DIVEITEM.Item_No = DIVESTOK.Item_No
  GROUP BY DIVECUST.Name
  HAVING ((DIVECUST.Name) LIKE
  '%Jazdzewski%');

GROUP BY

- SELECT DEST.Destination_Name, Count(*) AS Expr1

  FROM DEST INNER JOIN DIVEORDS ON DEST.Destination_Name = DIVEORDS.Destination

  GROUP BY DEST.Destination_Name

  HAVING ((Count(*))>1);

- *Provides a list of Destinations with the number of orders going to that destination*

SQL Commands

- Data Definition Language (DDL)
  - For creation of relations/tables, views, indexes.
- Data Manipulation Language (DML)
  - For maintaining db, including those for updating, inserting, modifying and querying.
- Data Control Language (DCL)
  - For db administration, user privileges and storing/removing erroneous transactions.

Create Table

- CREATE TABLE table-name (attr1 attr-type PRIMARYKEY, attr2 attr-type,…,attrN attr-type);
  - Adds a new table with the specified attributes (and types) to the database.

- In MySQL (5.5+) and SQLite3
  - CREATE TABLE newtablename AS SELECT …
    - Creates *new table* with contents from SELECT command including data types

INSERT

- INSERT INTO table-name (col1, col2, col3, …, colN) VALUES (val1, val2, val3, …, valN);

- INSERT INTO table-name (col1, col2, col3, …, colN) SELECT…

- Column list is optional, if omitted assumes all columns in table definition and order

Access Data Types (*Not MySQL*)

- **Numeric** (1, 2, 4, 8 bytes, fixed or float)
- **Text** (255 max)
- **Memo** (64000 max)
- **Date/Time** (8 bytes)
- **Currency** (8 bytes, 15 digits + 4 digits decimal)
- **Autonumber** (4 bytes)
- **Yes/No** (1 bit)
- **OLE** (limited only by disk space)
- **Hyperlinks** (up to 64000 chars)

Access Numeric types

- **Byte**
  - Stores numbers from 0 to 255 (no fractions). 1 byte
- **Integer**
  - Stores numbers from –32,768 to 32,767 (no fractions) 2 bytes
- **Long Integer** *(Default)*
  - Stores numbers from –2,147,483,648 to 2,147,483,647 (no fractions). 4 bytes
- **Single**
  - Stores numbers from -3.402823E38 to –1.401298E–45 for negative values and from 1.401298E–45 to 3.402823E38 for positive values. 4 bytes
- **Double**
  - Stores numbers from –1.797693134862 31E308 to –4.940656458412 47E–324 for negative values and from 1.797693134862 31E308 to 4.940656458412 47E–324 for positive values. 15 8 bytes
- **Replication ID**
  - Globally unique identifier (GUID) N/A 16 bytes

MySQL Data Types

- MySQL supports all of the standard SQL numeric data types. These types include the exact numeric data types (INTEGER, SMALLINT, DECIMAL, and NUMERIC), as well as the approximate numeric data types (FLOAT, REAL, and DOUBLE PRECISION). The keyword INT is a synonym for INTEGER, and the keyword DEC is a synonym for DECIMAL
- Numeric (can also be declared as UNSIGNED)
  - TINYINT (1 byte)
  - SMALLINT (2 bytes)
  - MEDIUMINT (3 bytes)
  - INT (4 bytes)
  - BIGINT (8 bytes)
  - NUMERIC or DECIMAL
  - FLOAT
  - DOUBLE (or DOUBLE PRECISION)

MySQL Data Types

- The date and time types for representing temporal values are DATETIME, DATE, TIMESTAMP, TIME, and YEAR. Each temporal type has a range of legal values, as well as a "zero" value that is used when you specify an illegal value that MySQL cannot represent
  - DATETIME '0000-00-00 00:00:00'
  - DATE '0000-00-00'
  - TIMESTAMP (4.1 and up) '0000-00-00 00:00:00'
  - TIMESTAMP (before 4.1) 00000000000000
  - TIME '00:00:00'
  - YEAR 0000

MySQL Data Types

- The string types are CHAR, VARCHAR, BINARY, VARBINARY, BLOB, TEXT, ENUM, and SET
- Maximum length for CHAR is 255 and for VARCHAR is **65,535**

Value	CHAR(4)	Storage	VARCHAR(4)	Storage
""	" "	4	""	1
"ab"	"ab "	4	"ab"	3
"abcd"	"abcd"	4	"abcd"	5
"abcdefg"	"abcd"	4	"abcd"	5

- VARCHAR uses 1 or 2 bytes for the length
- For longer things there is BLOB and TEXT

MySQL Data Types

- A **BLOB** is a binary large object that can hold a variable amount of data.
- The four BLOB types are TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB. These differ only in the maximum length of the values they can hold
- The four TEXT types are TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT. These correspond to the four BLOB types and have the same maximum lengths and storage requirements
- TINY=1byte, BLOB and TEXT=2bytes, MEDIUM=3bytes, LONG=4bytes

MySQL Data Types

- BINARY and VARBINARY are like CHAR and VARCHAR but are intended for binary data of 255 bytes or less
- ENUM is a list of values that are stored as their addresses in the list
  - For example, a column specified as ENUM('one', 'two', 'three') can have any of the values shown here. The index of each value is also shown:
    - **Value = Index**
    - NULL = NULL
    - ‘’ = 0
    - 'one’ = 1
    - ‘two’ = 2
    - ‘three’ = 3
  - An enumeration can have a maximum of 65,535 elements.

MySQL Data Types

- The final string type (for this version) is a SET
- A SET is a string object that can have zero or more values, each of which must be chosen from a list of allowed values specified when the table is created.
- SET column values that consist of multiple set members are specified with members separated by commas (‘,’)
- For example, a column specified as SET('one', 'two') NOT NULL can have any of these values:
  - ''
  - 'one'
  - 'two'
  - 'one,two ‘
- A set can have up to 64 member values and is stored as an 8byte number

PostgreSQL Data Types

- Boolean
- Character (text, varchar, char)
- Binary
- Date/time (timestamp/time with/without timezone, date, interval)
- Money
- Enum
- HStore, an extension enabled key-value store within PostgreSQL[33]
- Arrays (variable length and can be of any data type, including text and composite types) up to 1 GB in total storage size
- **Geometric primitives**
- XML supporting XPath queries
- UUID
- JSON, and a faster binary JSONB (since version 9.4; not the same as BSON[34])

Other characteristics of attributes

- You can also declare attributes with certain properties, e.g.,
  - PRIMARY KEY
  - FOREIGN KEY
  - NOT NULL
  - UNIQUE
  - CHECK expressions
  - DEFAULT values
  - COMMENTs
  - Etc.