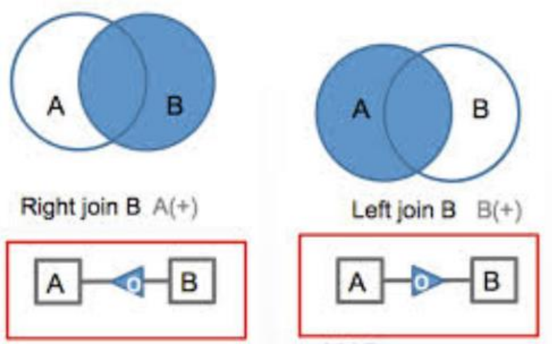
The Relational Model

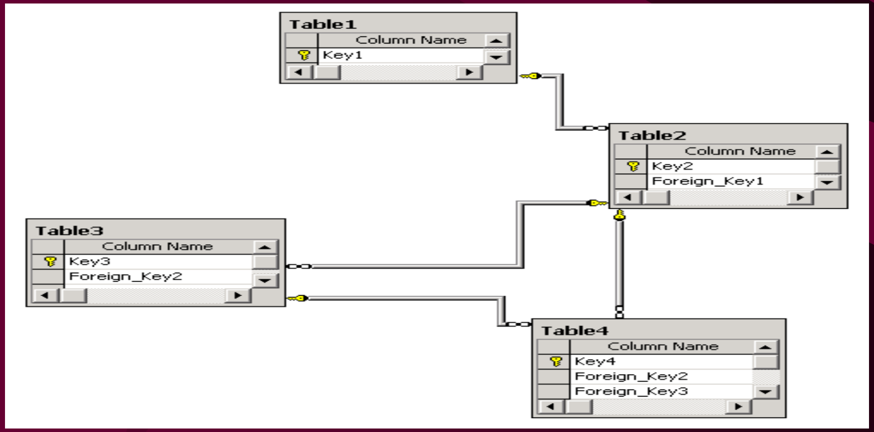


Relational Model

- Model of the <u>logical database</u> structure, not physical structure
- The relational model is a <u>table level</u> model, as opposed to the E-R model which is an <u>entity level</u> model
- Relational Model has advantages over other data models in terms of:
  - Flexibility of queries
  - Non procedural programming (queries without programming)
  - Consistency of queries and algebra

Original Reference

- E. F. Codd, “A Relational Model of Data for Large Shared Databanks,” Communications of the ACM, 1970



1951: The Univac uses **magnetic tape** as well as punched cards for data storage.

1956: IBM introduces first **magnetic hard disk drive** in its Model 305 RAMAC.

1961: Charles Bachman at GE develops the first **database management system**, IDS.

1968: IBM offers the **IMS** hierarchical database for System/360 mainframes.

1969: Edgar F. "Ted" Codd inverts the relational database.

1973: Cullinane, led by **John J. Cullinane**, ships IDMS, a network-model database for IBM mainframes.

1976: Honeywell ships **Multics Relational Data Store**, the first commercial relational database.

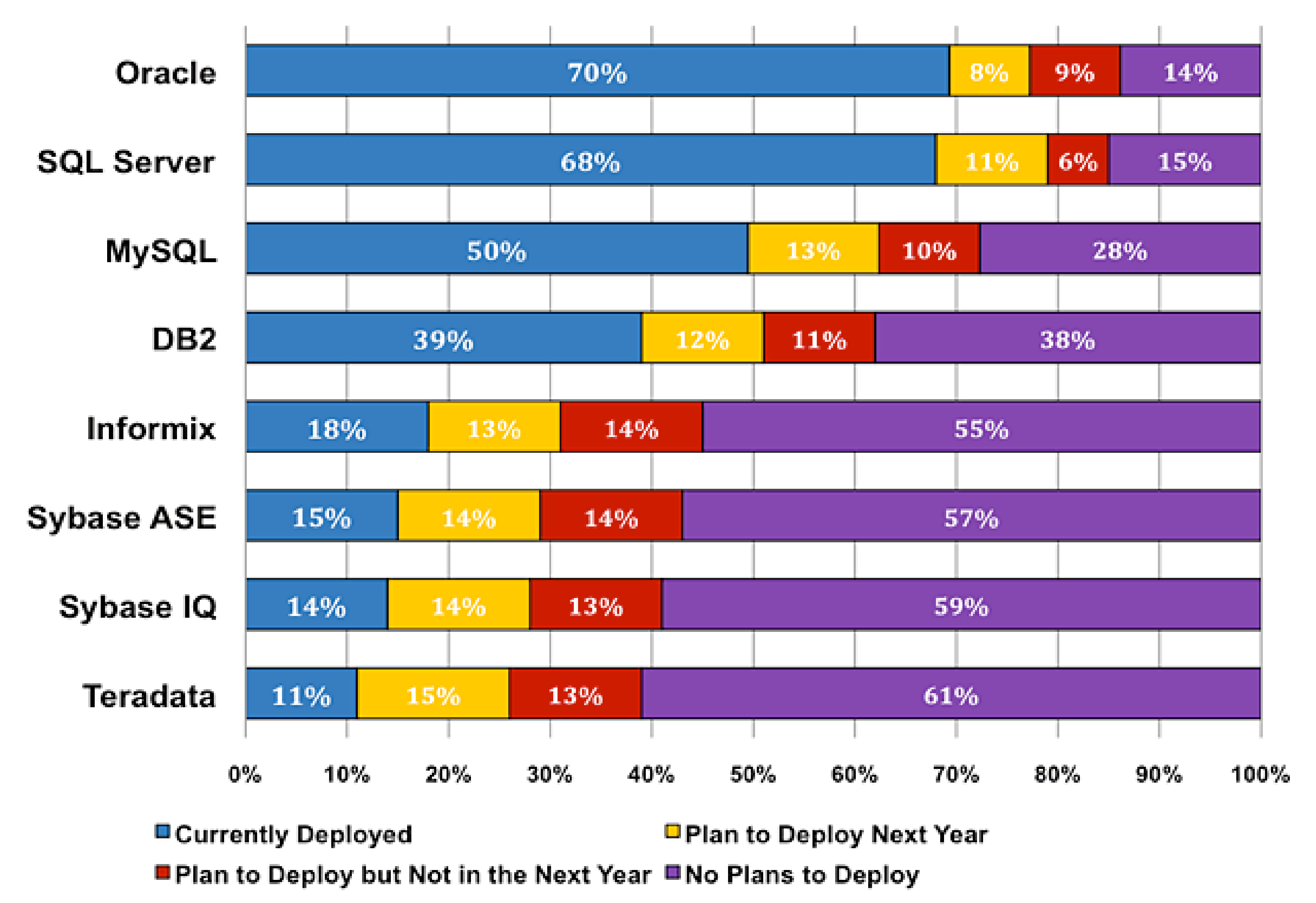
1979: Oracle introduces the first commercial **SQL relational database** management system.

1983: IBM introduces **DB2**.

1950 1960 1970 1980 1990


ORACLE
foundation
firebird
PostgreSQL
IBM. DB2.
Microsoft
SQL Server 2008
SQLite
1·2·3
MySQL
TERADATA
CSV

Database Usage



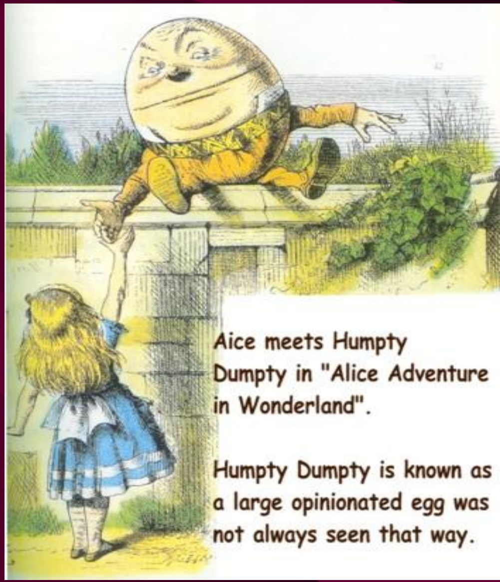
Peak Database Throughput
[JSP Net Application, PC Magazine]

- Oracle 9i – 629 pages/second
- MySQL 4.0 – 608 pages/second
- IBM DB2 Universal Database 7.2 – 494 pages/second
- Sybase Adaptive Server Enterprise 12.5 – 476 pages/second
- Microsoft SQL Server 2000/SP2 – 209 pages/second

Highly problem dependent !

Terminology

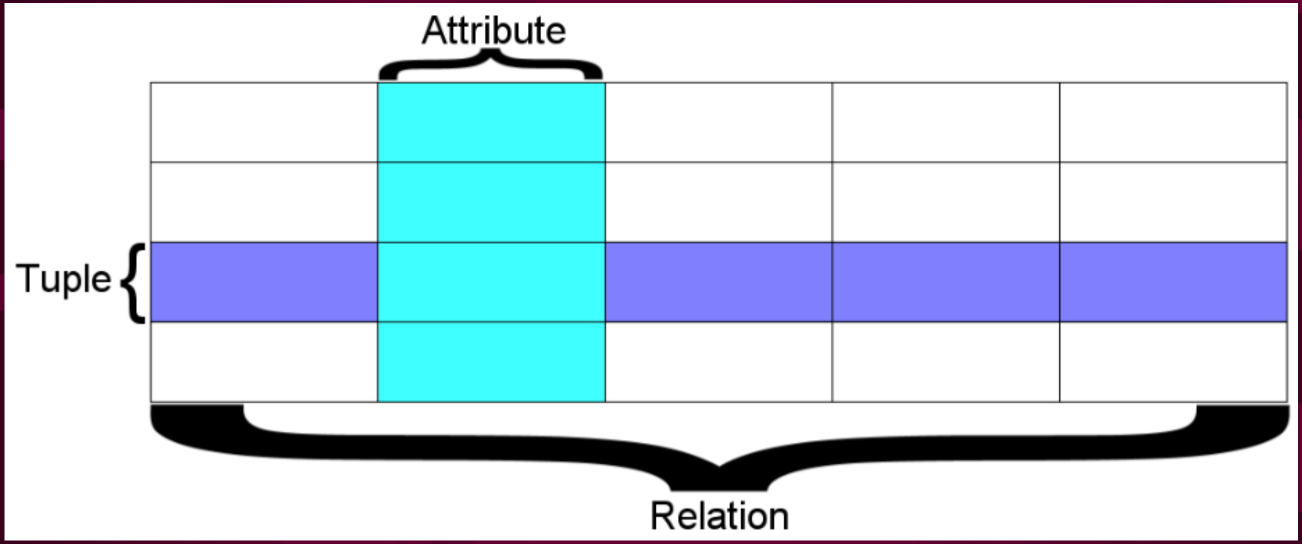
- "When I use a word, " Humpty Dumpty said, in a rather scornful tone, "It means just what I choose it to mean - neither more nor less."
- "The question is," said Alice, "Whether you can make words mean so many different things."
- *"The question is," said Humpty Dumpty, "Which is to be master - That's all."*



Terminology (con’t)

Relational	**Object Oriented**	**Programming**	**User**
Relation	Object Type	File	Table
Tuple	Object	Record	Row
Attribute	Mapping to an instance of another type	Field	Column
Degree or arity	# of properties	# of fields	# of columns
cardinality	# of mappings	# of records	# of rows

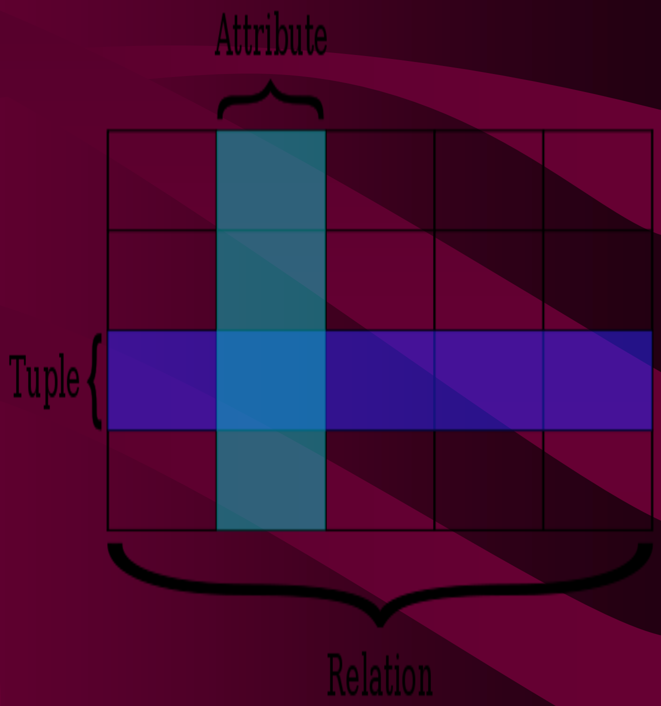
The “relation” is a Mathematical Table



Copyright Dan Brandon, PhD,PMP   Christian Brothers University

Relational Model

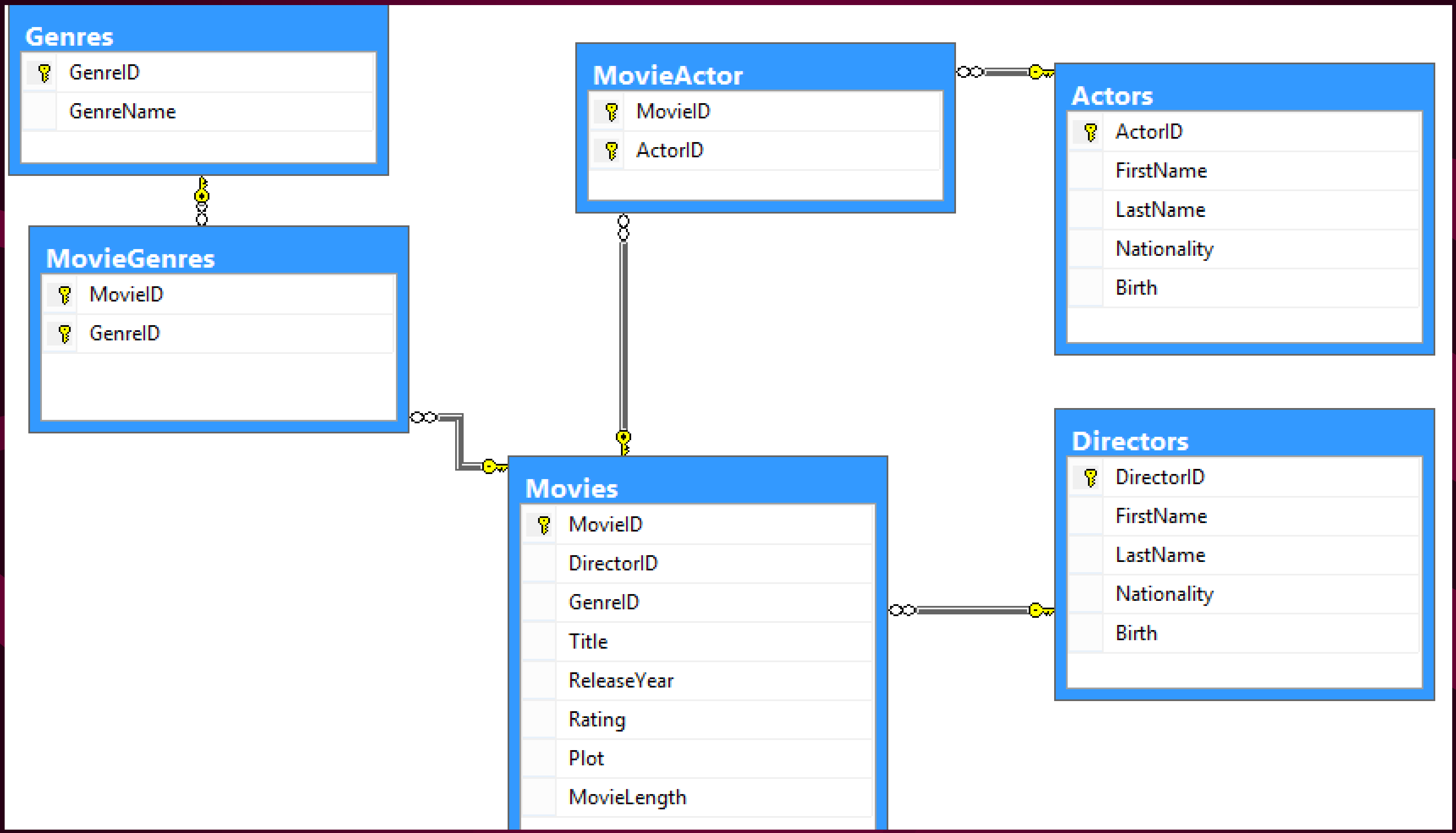
- Model of the logical database structure, not physical structure
- ER models (discussed later in course) can be converted to relational models
- Relational Model has advantages over other data models in terms of flexibility of queries, non procedural programming, and consistency of queries and algebra



Advantages of Relational Model

- Represents relationships by values not pointers
- **Allows non-procedural access to information** (indicate what information you want, not how to get it)
- Additional relationships added relatively easily
- Sound mathematical foundation to avoid inconsistencies and irregularities
- Uses “indexes” for speed of access, and indexes can be dynamically created (and cached) from data tables

Relations and Relationships
(foreign keys used for relationships)



Copyright Dan Brandon, PhD,PMP Christian Brothers University

Relation

- A relation is a two dimensional table that holds information about something (entity or object)
- The "relational" in relational database comes from the word relation, not relationships
- But not just any kind of table:
  - The cells of the table must be single valued; no repeating groups. The single value need not be of fixed length
  - All of the entries in a column must be of the same, and satisfy the ***domain*** of that column. A column has a unique name
  - No two rows may be the same

Characteristics of Tables

1	A table is perceived as a two-dimensional structure composed of rows and columns.
2	Each table row (tuple) represents a single entity occurrence within the entity set.
3	Each table column represents an attribute, and each column has a distinct name.
4	Each intersection of a row and column represents a single data value.
5	All values in a column must conform to the same data format.
6	Each column has a specific range of values known as the attribute domain.
7	The order of the rows and columns is immaterial to the DBMS.
8	Each table must have an attribute or combination of attributes that uniquely identifies each row.

Copyright Dan Brandon, PhD,PMP   Christian Brothers University

TITLE Table [ISBN is PK, Publisher is FK]

Isbn	Title	Publisher
0-03-123456-6	T1	*P1*
0-02-654321-9	T2	*P1*
0-01-234567-8	T3	*P2*

COPY Table [ISBN/Copy # is PK, Student & ISBN are FK's]

Title	CopyNumber	Student	Due Date
Isbn-T1	**1**	**null**	**null**
Isbn-T1	**2**	***SId-S1***	**4/1/96**
Isbn-T1	**3**	***SId-S2***	**5/2/96**
Isbn-T2	**1**	**null**	**null**
Isbn-T2	**2**	**null**	**null**
Isbn-T3	**1**	***SId-S1***	**3/27/96**
Isbn-T3	**2**	**null**	**null**

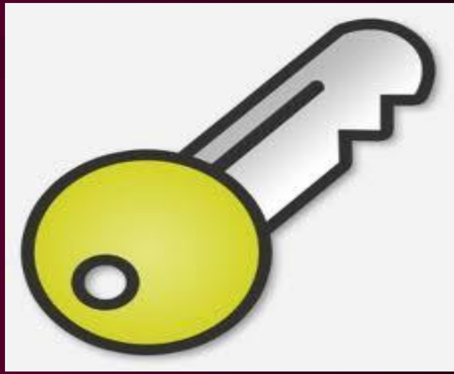
Relation (Table) of Students:
STUDENT (Name, StudentId, Major, Sex)

Name	StudentId	Major	Sex
Jones, Sam	**11111111**	**CS**	**M**
Smith, Jane	**55555555**	**Biology**	**F**
Brown, Ed	**777777777**	**ChemE**	**M**
Moore, Sally	**222222222**	**IT**	**F**

Key (mathematically)

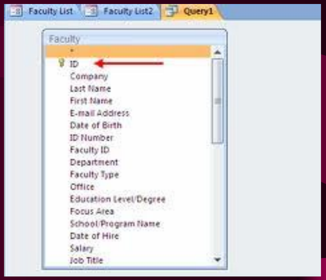
- A group of one or more attributes that <u>uniquely</u> identifies a row
- In the relation:
  - STUDENT (StudentId, Major, Name, Address,...)
  - StudentId is a key
- In the relation:
  - GRADE (StudentId, Course, Grade)
  - (StudentId, Course) is the key [assuming a student only completes the same course once, or only the last taking of the course is retained in the database]

Unique Key



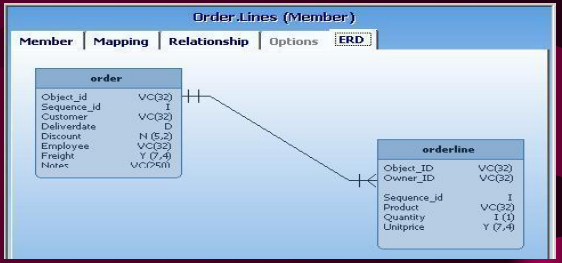
- The notion of key described on the previous slide is the mathematical term
- In practice this is usually called a *unique key*
- There may be other keys that are not unique (and these could be used as indexes)
- There may be more than one unique key (*candidate keys*), but only one is the main table key (or <u>*primary key*</u>)

Primary Key



- In an employee table, the primary key would probably be employeeNumber; another unique key would be social security number (a unique index would probably be set up on social security number to avoid duplicates)
- A table can only have one "primary key"
- A non-unique key may be "name"
- In some terminology the word primary has to do with the primary <u>physical organization</u> of the records
- Usually the table's primary organization (hashed or b-tree data structure) is consistent with the main unique key

Foreign Key



- <u>A Foreign Key in one table is a value that is the primary (main) key of another table</u>
- <u>A table can have none, one, or many foreign keys</u>
- The value of Isbn in the COPY table is a foreign key, since the value of Isbn is the primary key in the TITLE table
- The value of StudentId in the COPY table is a foreign key, since the value of StudentId is the primary key of the STUDENT table

COPY Table [ISBN/Copy # is PK, Student & ISBN are FK's]

Title	CopyNumber	Student	Due Date
Isbn-T1	**1**	**null**	**null**
Isbn-T1	**2**	***SId-S1***	**4/1/96**
Isbn-T1	**3**	***SId-S2***	**5/2/96**
Isbn-T2	**1**	**null**	**null**
Isbn-T2	**2**	**null**	**null**
Isbn-T3	**1**	***SId-S1***	**3/27/96**
Isbn-T3	**2**	**null**	**null**

VEND_CODE is FK in the PRODUCT Table

Table name: PRODUCT
Primary key: PROD_CODE
Foreign key: VEND_CODE

Database name: Ch03_SaleCo

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	18.63	6	231
SRE-657UG	Rat-tail file	2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	6.79	8	235

link

Table name: VENDOR
Primary key: VEND_CODE
Foreign key: none

VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425

Copyright Dan Brandon, PhD,PMP   Christian Brothers University

Types of Keys

- Several different types of keys are used in the relational model
- Some authors and RDBMS's may have slightly differing definitions
  - Composite key: key that is composed of more than one attribute
  - Key attribute: attribute that is a part of a key
  - Superkey: key that can uniquely identify any row in the table
  - Candidate key: minimal superkey
  - Primary key: A candidate key selected to uniquely identify all other attribute values in any given row; cannot contain null entries
  - Entity integrity: condition in which each row in the table has its own unique identity
    - All of the values in the primary key column must be unique
    - No key attribute in the primary key can contain a null
  - Null: absence of any data value
    - Unknown attribute value, known but missing attribute value, or inapplicable condition
  - Referential integrity: every reference to an entity instance by another entity instance is valid
  - Foreign key: primary key of one table that has been placed into another table to create a common attribute
  - Secondary key (index): key used strictly for data retrieval purposes

Indexes

- Used to quickly and logically access rows in a table
  - Index key: index's reference point that leads to data location identified by the key
  - Unique index: index key can have only one pointer value (row) associated with it
  - Non-unique index: index key can have multiple rows associated with it
- Each index is associated with only one table
  - An index can have multiple attributes

RDBMS Integrity

Entity Integrity	Description
Requirement	All primary key entries are unique, and no part of a primary key may be null.
Purpose	Each row will have a unique identity, and foreign key values can properly reference primary key values.
Example	No invoice can have a duplicate number, nor can it be null; in short, all invoices are uniquely identified by their invoice number.
Referential Integrity	**Description**
Requirement	A foreign key may have either a null entry, as long as it is not a part of its table's primary key, or an entry that matches the primary key value in a table to which it is related (every non-null foreign key value must reference an existing primary key value).
Purpose	It is possible for an attribute not to have a corresponding value, but it will be impossible to have an invalid entry; the enforcement of the referential integrity rule makes it impossible to delete a row in one table whose primary key has mandatory matching foreign key values in another table.
Example	A customer might not yet have an assigned sales representative (number), but it will be impossible to have an invalid sales representative (number).

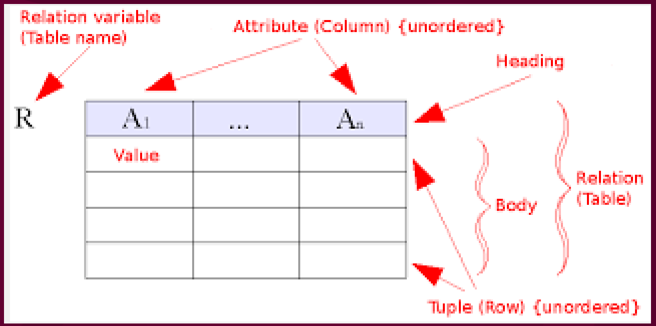
Codd's Relational Rules

Rule	Rule Name	Description
1	Information	All information in a relational database must be logically represented as column values in rows within tables.
2	Guaranteed access	Every value in a table is guaranteed to be accessible through a combination of table name, primary key value, and column name.
3	Systematic treatment of nulls	Nulls must be represented and treated in a systematic way, independent of data type.
4	Dynamic online catalog based on the relational model	The metadata must be stored and managed as ordinary data—that is, in tables within the database; such data must be available to authorized users using the standard database relational language.
5	Comprehensive data sublanguage	The relational database may support many languages; however, it must support one well-defined, declarative language as well as data definition, view definition, data manipulation (interactive and by program), integrity constraints, authorization, and transaction management (begin, commit, and rollback).
6	View updating	Any view that is theoretically updatable must be updatable through the system.
7	High-level insert, update, and delete	The database must support set-level inserts, updates, and deletes.

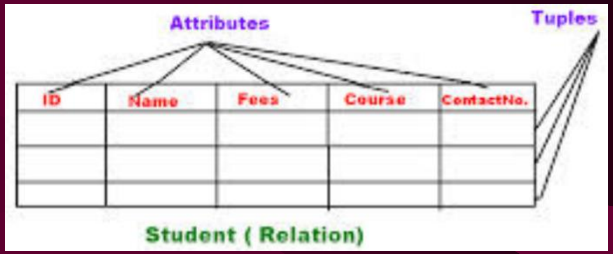
Codd’s Relational Rules (con’t)

Rule	Rule Name	Description
8	Physical data independence	Application programs and ad hoc facilities are logically unaffected when physical access methods or storage structures are changed.
9	Logical data independence	Application programs and ad hoc facilities are logically unaffected when changes are made to the table structures that preserve the original table values (changing order of columns or inserting columns).
10	Integrity independence	All relational integrity constraints must be definable in the relational language and stored in the system catalog, not at the application level.
11	Distribution independence	The end users and application programs are unaware of and unaffected by the data location (distributed vs. local databases).
12	Nonsubversion	If the system supports low-level access to the data, users must not be allowed to bypass the integrity rules of the database.
13	Rule zero	All preceding rules are based on the notion that to be considered relational, a database must use its relational facilities exclusively for management.

Relational Algebra



Relational Database Data Manipulation



- Procedural
  - Relational Algebra
- Non-procedural
  - Query by Example (QBE) - [ie Access]
  - Relational Calculus (Predicate Calculus)
  - Transform Languages (ie SQL)
    - Command Line
    - Within programming language (embedded or call level)

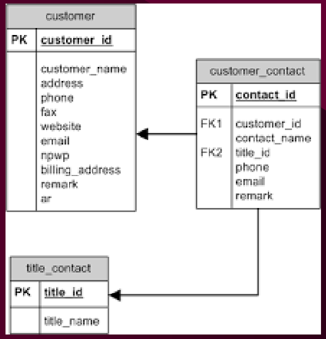
Relational Calculus

- <u>Non – Procedural</u> (indicates properties of data to be retrieved):
  - Tuple Relational Calculus – integrates over rows [basic SQL]
  - Domain Relational Calculus – integrates over domain (columns) [SQL subqueries]
- Implemented in SQL

Relational Model

- S (<u>SID</u>, SName, City)
- P (<u>PID</u>, PName, Size, Price)
- SP (<u>*SID, PID*</u>, Qty) Intersection Table

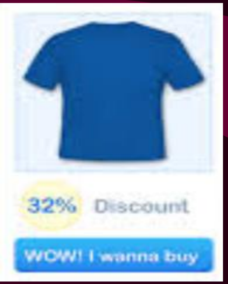
Foreign Keys ?



Salesperson Table (S)

SID	Sname	City
S1	Peterson	Aarhus
S2	Olsen	Copenhagen
S4	Hansen	Odense
S5	Jensen	Copenhagen

Product Table (P)



PID	PName	Size	Price
P1	Shirt	6	50
P3	Trousers	5	90
P4	Socks	7	20
P5	Blouse	6	50
P8	Blouse	8	60

SP Table (Intersection Table)

SID	PID	Qty
S2	P1	200
S2	P3	100
S4	P5	200
S4	P8	100
S5	P1	50
S5	P3	500
S5	P4	800
S5	P5	500
S5	P8	100

Relational Algebra

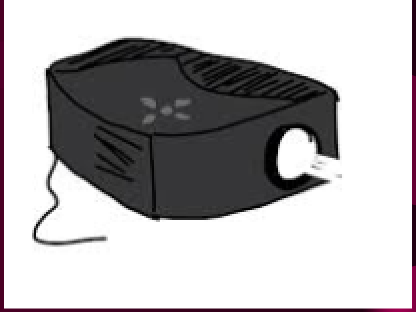
- Relational algebra combines relations by means of operators to form new relations which, combined with other relations by means of a further operator, can form new relations (in other words relational operators can be nested)
- Relational Algebra is a <u>procedural</u> approach to data manipulation
- Understanding Relational Algebra makes it easier to see what how SQL performs its work

Eight Operators

- Projection
- Selection
- Union
- Intersection
- Difference
- Product
- Division
- Join

The result of an operation is always a relation, even if it is a “singleton” or “null relation” !

Projection



- A vertical section of a table; the projection statement selects some columns from a table and creates a new table whose rows only contain the desired columns
- T = S [City];
- OR “Which cities have salespersons ?”

Table: T (note that T contains one less row than S because there are two salespersons in Copenhagen)

City
Aarhus
Copenhagen
Odense

Projection Examples

Original table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

PROJECT PRICE yields

New table

PRICE
5.26
25.15
10.99
1.92
1.47
34.99

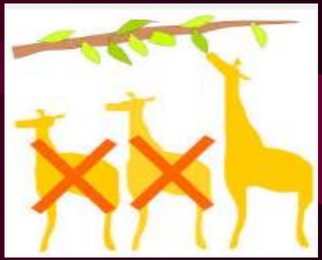
PROJECT P_DESCRIPT and PRICE yields

P_DESCRIPT	PRICE
Flashlight	5.26
Lamp	25.15
Box Fan	10.99
9v battery	1.92
100W bulb	1.47
Powerdrill	34.99

PROJECT P_CODE and PRICE yields

P_CODE	PRICE
123456	5.26
123457	25.15
123458	10.99
213345	1.92
254467	1.47
311452	34.99

Selection



- A selection is a subset of the set of the rows in a table - sometimes called “restriction”
- The selection is made on the basis of the contents of one or more columns
- T = S WHERE City = “Copenhagen”;
- OR

Which salespersons are based in Copenhagen?

Which salespersons are based in Copenhagen?

SID	Sname	City
S2	Olsen	Copenhagen
S5	Jensen	Copenhagen

Selection Examples

Original table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

SELECT ALL yields

New table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

SELECT only PRICE less than $2.00 yields

P_CODE	P_DESCRIPT	PRICE
213345	9v battery	1.92
254467	100W bulb	1.47

SELECT only P_CODE = 311452 yields

P_CODE	P_DESCRIPT	PRICE
311452	Powerdrill	34.99

Combine Selection and Projection

- What quantity of product P1 has been sold by salesperson S5 ?
  - T = SP[Qty] WHERE PID = P1 and SID = S5;
- The answer is the single column, single row table of 50.

Union

- The union of two tables is a new table containing the rows of both tables; duplicate rows are eliminated
- The two tables in the union have to have the <u>same columns</u> (same number of attributes, and attributes in corresponding columns have same domain)
- Similar to the logical "OR"

Which of the salespersons are either in Copenhagen or have sold some P8 ?

- T1 = S [SID] WHERE City = Copenhagen;
- T2 = SP [SID] WHERE PID = P8;
- T = T1 UNION T2; (or T1+T2)
- or in one statement:
- T= (S[SID] WHERE City = “Copenhagen”) UNION (SP[SID] WHERE PID = P8);
- Note that T1 and T2 both have just one column of salespersons

T1

SID

S2

S5

T2

SID

S4

S5

T (Union of T1 and T2)

SID
S2
S4
S5

Union Example

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

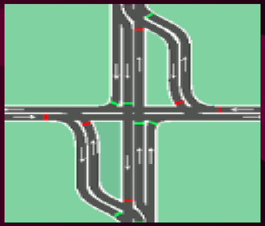
UNION

P_CODE	P_DESCRIPT	PRICE
345678	Microwave	160.00
345679	Dishwasher	500.00
123458	Box Fan	10.99

yields

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99
345678	Microwave	160
345679	Dishwasher	500

Intersection

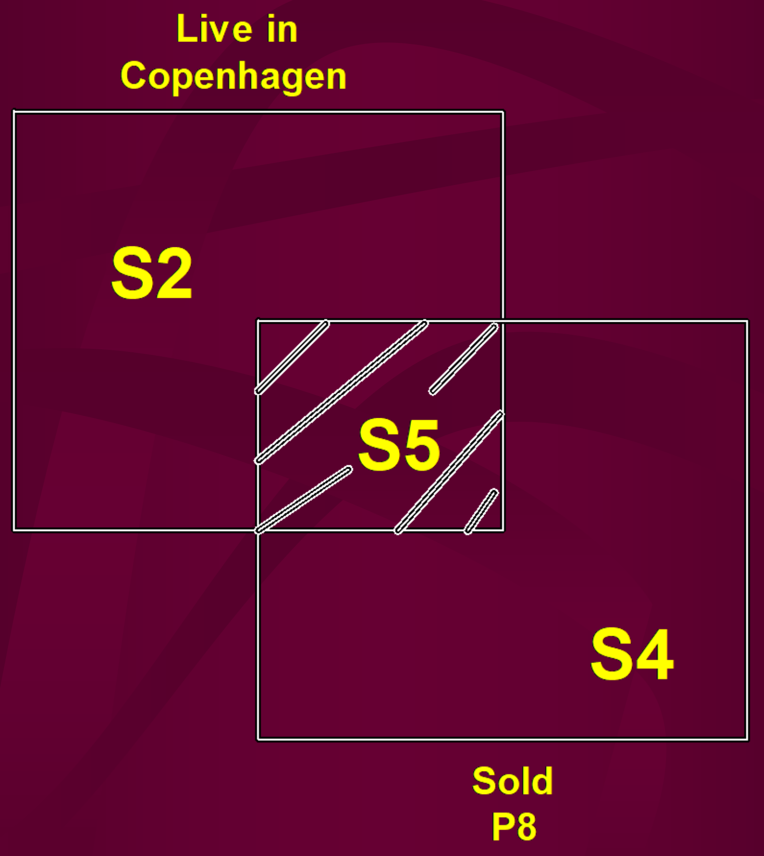


- The intersection of two tables is a new table containing only those entries that are members of <u>both</u> tables
- The tables must have the same columns
- Similar to the logical “AND”

Which salespersons are both in Copenhagen and have sold P8 ?

- Using T1 and T2 from the last example
- T = T1 INTERSECT T2;
- The answer is the table with only S5 in it

Intersection



Intersect Example

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

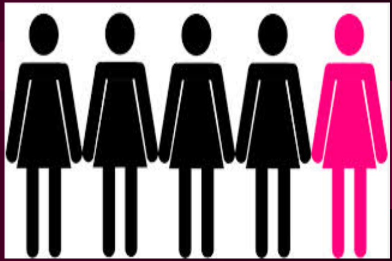
INTERSECT

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

STU_FNAME	STU_LNAME
Franklin	Johnson

Difference



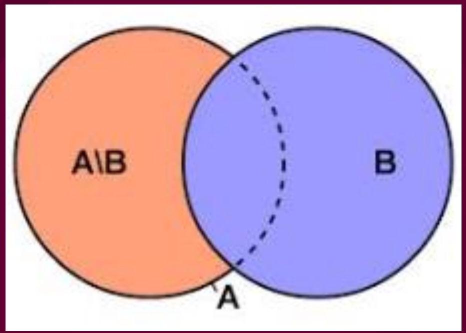
- The difference of two tables is the set of entities that are members of the first table <u>but not of the second</u>
- Both tables have the same columns

Which salespersons are in Copenhagen but have not sold any P8 ?

- Using T1 and T2 from the last example
- T = T1 - T2;
- The answer is the table with only S2 in it

T = T1 - T2 = [S2]

- T1 (in Copenhagen)
  - S2
  - S5
- T2 (sold P8)
  - S4
  - S5



The difference between knowing your shit and knowing you're shit.

Difference Example

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

DIFFERENCE

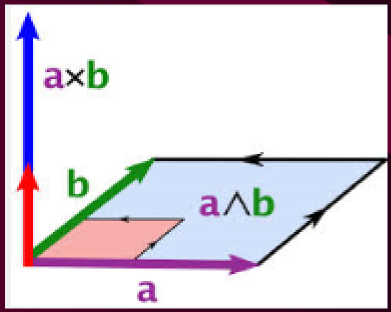
EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

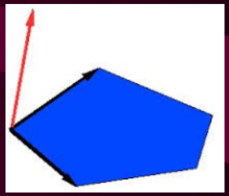
yields

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Martin	Lopez

Product (Cartesian)

- The product of two tables is a new table containing rows corresponding to all combinatorial possibilities of the rows of the two tables
- Similar to matrix multiplication, except concatenate rows instead of arithmetic multiplication



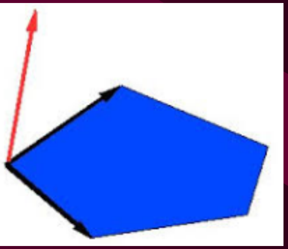


Product Example:

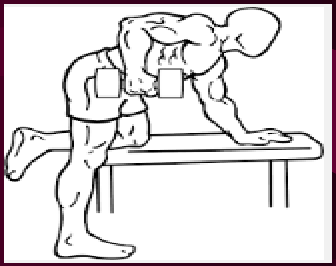
- Table 1 has two rows the first of which contains a1 and b1, the second of which contains a2 and b2
  - a1 b1
  - a2 b2
- Table 2 has two rows the first of which contains c1 and d1, the second of which contains c2 and d2
  - c1 d1
  - c2 d2

a1	b1	X	c1	d1
a2	b2		c2	d2

- The product of the two tables contains 4 rows:
  - a1, b1, c1, d1
  - a1, b1, c2, d2
  - a2, b2, c1, d1
  - a2, b2, c2, d2



Resulting table size…



- Multiply Rows: If the first table in a product has N rows and the second table in a product has M rows, the product will have N times M rows
- Add Columns: If the first table in a product has X columns and the second table in a product has Y columns, the product will have X plus Y columns

What are all possible combinations of Salespersons and Products ?

- T = MULTIPLY S[SID] WITH P [PID]; (or S[SID] x P[PID])
- The result is a table with two columns and twenty rows (for each combination of SID[4] and PID [5])

Product Example

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

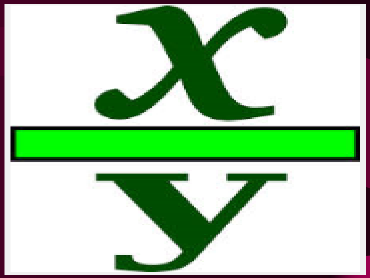
PRODUCT

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

yields

P_CODE	P_DESCRIPT	PRICE	STORE	AISLE	SHELF
123456	Flashlight	5.26	23	W	5
123456	Flashlight	5.26	24	K	9
123456	Flashlight	5.26	25	Z	6
123457	Lamp	25.15	23	W	5
123457	Lamp	25.15	24	K	9
123457	Lamp	25.15	25	Z	6
123458	Box Fan	10.99	23	W	5
123458	Box Fan	10.99	24	K	9
123458	Box Fan	10.99	25	Z	6
213345	9v battery	1.92	23	W	5
213345	9v battery	1.92	24	K	9
213345	9v battery	1.92	25	Z	6
311452	Powerdrill	34.99	23	W	5
311452	Powerdrill	34.99	24	K	9
311452	Powerdrill	34.99	25	Z	6
254467	100W bulb	1.47	23	W	5
254467	100W bulb	1.47	24	K	9
254467	100W bulb	1.47	25	Z	6

Division



- Division involves a dividend, divisor, remainder, and quotient:
  - DIVIDEND / DIVISOR = QUOTIENT
- The dividend table will have M + N columns
- The divisor table will have N columns with the same columns as N columns of the dividend
- The quotient will have M columns
- The remainder will have M + N columns
- Most RDMS still do not support division

Division (con't)

- Division is the <u>inverse</u> of multiplication
- The table formed by the union of
  - the quotient multiplied by the divisor
  - and the remainder
- is the original table

Subject

Name	Course
Systems	BCS
Database	BCS
Database	MCS
Algebra	MCS

÷

Course

Course
BCS
MCS

→

Result

Name
Database

Which salesperson has sold some of all products?

- T = DIVIDE SP BY P [PID] ;
- The result (quotient) is only S5
- Another interpretation of division is: Which sets of salesman rows (M=1) of the dividend (SP) have the attributes of the divisor (ie which S's in SP are related to all rows in the P table)
- Used for complex pattern matching searches: "Who has blond hair, medium build, owns a red corvette, ..."

Divisor

P(PID)

P1

P3

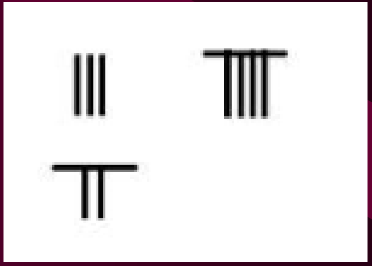
P4

P5

P8

Remainder - SP rows not containing quotient (S5)

SID	PID
S2	P1
S2	P3
S4	P5
S4	P8



Divide Example

P_CODE	CUS_CODE
123456	10400
123456	11501
123456	10030
123456	12550
234567	12350
234567	10040
234567	10600
234567	10030
234567	12550
345678	10400
345678	11630
345678	12550
456789	11630
567890	10900
567890	10030
567890	12550
678901	11500
678901	10400
678901	11630

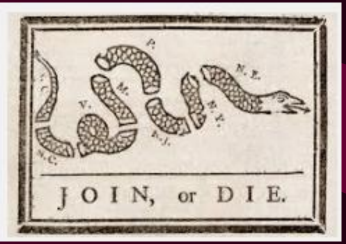
DIVIDE

P_CODE
123456
234567
567890

yields →

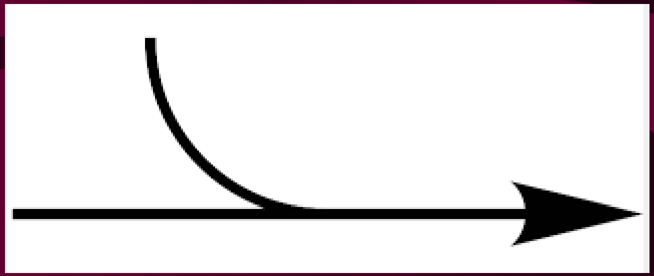
CUS_CODE
10030
12550

Join



- A join usually* requires two tables to have one or more columns in common (normally PK in one table and FK in other !)
- The columns in common between the two tables are related by a "join condition"
- **The two tables are multiplied together, then all rows not matching the join condition are eliminated**

* The "cross join" is the same as the Cartesian product



- If the join condition is the equality between the columns in common, the join is called an <u>equijoin</u>
- If one of the two common columns in an equijoin is eliminated, then it is called a <u>natural join</u> (the most common kind of join which removes the redundancy)

Join example:

- T1:
  - a1, b1 *typically the b column here is a foreign key*
  - a2, b2
- T2:
  - b1, c1 *typically the b column here is a primary key*
  - b2, c2
- Join T1 with T2 where T1.col2 = T2.col1:
  - a1, b1, b1, c1
  - *a1, b1, b2, c2* *eliminate since T1.column 2*
  - *a2, b2, b1, c1* *not equal T2.column1*
  - a2, b2, b2, c2

- Resulting table from *equi join*:
  - a1, b1, b1, c1
  - a2, b2, b2, c2
- For a *natural join*, the identical columns are removed also:
  - a1, b1, c1
  - a2, b2, c2
- What we have done here is to supplement T1 with information from T2, corresponding to matching b (foreign key to primary key)

Class Exercise: Find natural join of these two tables on DID:

- EMPLOYEE (EID, EName, DID)

– 123	Doe	ABC
– 456	Ray	ABC
– 789	Mei	XYZ

- DEPARTMENT (DID, DName)

– ABC	Sales
– DEF	Mfg
– XYZ	Acct

- DID would be a FK in EMPLOYEE and the PK in DEPARTMENT

Don’t look ahead !

Copyright Dan Brandon, PhD, PMP

Step 1 - Multiply the Tables:

- 123 Doe ABC ABC Sales
- 123 Doe ABC DEF Mfg
- 123 Doe ABC XYX Acct
- 456 Ray ABC ABC Sales
- 456 Ray ABC DEF Mfg
- 456 Ray ABC XYZ Acct
- 789 Mei XYZ ABC Sales
- 789 Mei XYZ DEF Mfg
- 789 Mei XYZ XYZ Acct

Step 2 - Eliminate rows where EMPLOYEE.DID not equal DEPARTMENT.DID

123	Doe	ABC	Sales
456	Ray	ABC	Sales
789	Mei	XYZ	Acct

- We have supplemented the employee data with the the department names form the department table !

Class Exercise

Find the names of the salespersons who sold 500 or more of any product

Write the relational algebra expression

Salesperson Table (S)

SID	Sname	City
S1	Peterson	Aarhus
S2	Olsen	Copenhagen
S4	Hansen	Odense
S5	Jensen	Copenhagen

SP Table (Intersection Table)

SID	PID	Qty
S2	P1	200
S2	P3	100
S4	P5	200
S4	P8	100
S5	P1	50
S5	P3	500
S5	P4	800
S5	P5	500
S5	P8	100

Don’t look ahead !

Copyright Dan Brandon, PhD, PMP

Find the Names of Products Sold

- Product names are in P, and products sold are in SP !!!
- T1 = JOIN SP WITH P where SP [PID] = P [PID];
- T2 = T1 [PName];
- or in one statement:
- T = (JOIN SP WITH P where SP [PID] = P [PID]) [PName];
- Other syntax:
  - T = SP JOIN P (SP [PID] = P [PID]) [PName];
  - T = SP JOIN (SP [PID] = P [PID]) P [PName] ;

Find the products sold and quantities sold for salespersons in Copenhagen.

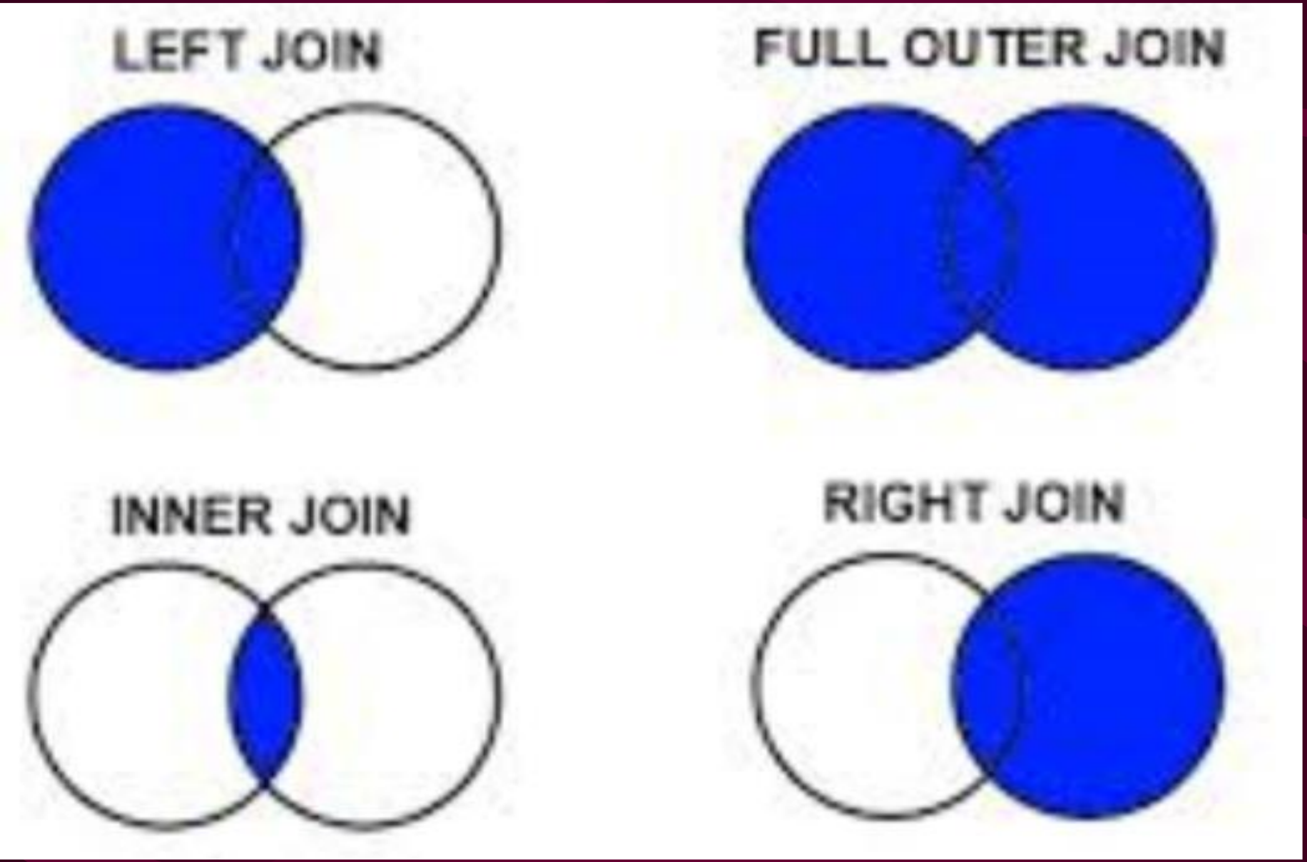
- T1 = S [SID] WHERE City = Copenhagen;
- T2 = JOIN T1 WITH SP WHERE T1 [SID] = SP [SID] ;
- or
- T1 = JOIN SP with S [SID, City] where SP [SID] = S [SID];
- T2 = T1 WHERE City = “Copenhagen”
- <u>Do selections before joins !</u>

Outer Joins

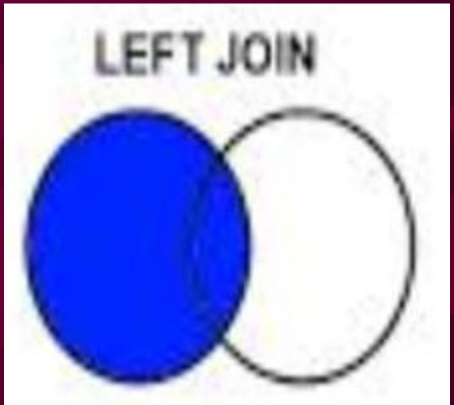
- Previous joins were *inner joins*
- If a join condition condition preserves all rows in the first table, but places nulls in the columns for the second table where no matching row exists, then it is called a left outer join [left is default]

- If a join condition preserves all rows in the second table, but places nulls in the columns for the first table where no matching row exists, then it is called a right outer join
- If a join condition preserves all rows in both tables, it is called a full outer join
- Outer joins are very important in accounting and other business applications since they give a full picture of the business situation and can be used for audit reports

Joins



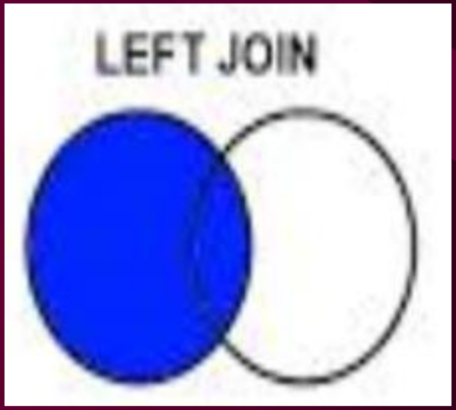
Outer join example:



- T1:
  - a1, b1
  - a2, b3
- T2:
  - b1, c1
  - b2, c2
- Join T1 with T2 where T1.col2 = T2.col1:
  - a1, b1, b1, c1
  - *a1, b1, b2, c2*
  - *a2, b3, b1, c1*
  - *a2, b3, b2, c2*

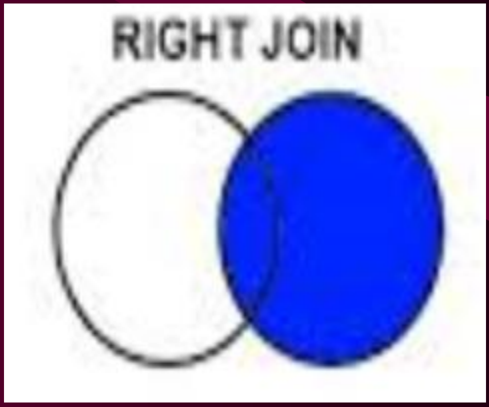
eliminate since T1.column 2 not equal T2.column1

- Resulting table from *equi join*:
  - a1, b1, b1, c1
- For a *natural join*, the identical columns are removed also:
  - a1, b1, c1
- For *left outer join* must <u>preserve all rows in first table</u>:
  - a1, b1, c1
  - a2, b3, null



Class exercise: find the right outer natural join of EMPLOYEE with DEPARTMENT {show resulting table data}

- EMPLOYEE (EID, EName, DID)
  - 123 Doe ABC
  - 456 Ray ABC
  - 789 Mei XYZ
- DEPARTMENT (DID, DName)
  - ABC Sales
  - DEF Mfg
  - XYZ Acct



Don’t look ahead !

Copyright Dan Brandon, PhD, PMP

- 123 Doe ABC Sales
- 456 Ray ABC Sales
- 789 Mei XYZ Acct
- NULL NULL DEF Mfg

References

- E. F. Codd, “A Relational Model of Data for Large Shared Databanks,” Communications of the ACM, 1970
- Birth of the Relational Model
  - Intelligent Enterprise, October 1998, p 61, ff
- Fundamentals of Relational Data Organization
  - Byte, November 1981, p 48, ff
- Inside Relational Databases with Examples in Access by Mark Whitehorn and Bill Marklyn
- Beginning Relational Data Modeling, Second Edition by Sharon Allen and Evan Terry
- Understanding Relational Database Query Languages by Dietrich; ISBN: 0-13-028652-4
- Relational Database Design and Implementation, Third Edition: Clearly Explained 3e (Morgan Kaufmann Series in Data Management Systems) by Jan L. Harrington

Homework

- Textbook Chapter Three
- Review Questions 1 thru 6
- Textbook Problems 1 thru 4
- Project overview due:
  - Topic
  - Problem Statement
  - Major Entities