DATABASE MANAGEMENT SYSTEM – DSE
IV Semester
JAN 2022

Relational Model

Introduction To Relational Model

- A data model is a collection of conceptual tools.
- **Relational Database Model** is the most common model in industry today.
- A relational database is based on the relational model developed by Edgar. F. Codd (12 rules).
- The relational model- collection of tables and the relationships among those data.
- A relational database consists of a collection of **tables,** each table is assigned with unique name**.**
- Correspondence between the concept of **table** and the mathematical concept of **relation**

Properties of a relation

- Each relation contains only one record type.
- Each relation has a fixed number of columns that are explicitly named. Each attribute name within a relation is unique.
- No two rows(tuples) in a relation are the same.
- Each item or element in the relation is atomic.
- Rows have no ordering associated with them.
- Columns have no ordering associated with them.

Relational Terminology

Terms	Definition
Relation	Set of rows(tuples), each row therefore has the same columns(attributes).
Tuple	It is a row in the relation.
Attribute	It is a column in the relation.
Degree of a relation	Number of columns in the relation
Cardinality of a relation	Number of rows in the relation
N-ary relation	Relation with degree N.
Domain	Set of allowed values for each attribute.

These relations represent University Database.

Example of a Relation

attributes
(or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples
(or rows)

The instructor relation

Example of a Relation

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

The course relation.

course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

The *prereq* relation

Some Terms

Relation Instance: A specific instance of a relation, i.e. set of rows in a relation at an instance.

In general, a relation schema consists of a list of Attributes and their corresponding domains.

Database schema, which is the logical design of the database.

Database Instance, which is a snapshot of the data in the Database at a given instant in time.

Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute.
  - A valid range value for a Marks attribute may be 0-100
  - Marks Domain is {0,1,2,..50,51,..100}
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain.
- The **null** value **causes** complications in the definition of many operations.

Relation Schema and Instance

- *If* $A_1, A_2, \ldots, A_n$ are **attributes***, then*
- $R = (A_1, A_2, \ldots, A_n)$ is a **relation schema**

  Example: *instructor* = (*ID, name, dept_name, salary*)

  Let $D_1, D_2, \ldots. D_n$ *be the Domains of* $A_1, A_2, \ldots, A_n$ *respectively.*

- Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation *r*** is a **subset of** $D_1 \times D_2 \times \ldots \times D_n$ ***(cartesian product)***

  Thus, a relation is a set of *n*-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$

Relation :

- A row in a table represents a relationship among a set of values.
- A table is a collection of such relationships, there is a close correspondence between the concept of table and the mathematical concept of relation.

Ex:

Cours_IDs set A={ BIO_301,BIO_399,CS_190,...} **(set of all valid Course_ID)**

Prereq_IDs set B={ BIO_101,CS_101,..} **(set of all valid Prrereq_ID)**

A x B ={ (BIO_301, BIO_101),(BIO_301,CS_101),..
(BIO_399,BIO_101), (BIO_399,CS_101),..
(CS_190, BIO_101),(CS_190,CS_101),..
}

A set (table in previous slide) *Prereq* is a subset of A x B

***Prereq* ={(BIO_301, BIO_101), (BIO_399,BIO_101), (CS_190, CS_101),...} is a Relation.**

Compare it with ***Prereq*** relation

A x B ={ (BIO_301, BIO_101),(BIO_301,CS_101),..
…,(BIO_399,BIO_101), (BIO_399,CS_101),..
….,(CS_190, BIO_101),(CS_190,CS_101),..
}

A x B gives all possible combinations of domain values, which involve real world facts- such as (BIO_301, BIO_101) i.e. for Course BIO-301, BIO-101 is Prerequisite course. Also A x B is having information such as (BIO_301,CS_101) which is not a real world fact.

A relation such as **Prereq** is the **subset of A x B** which represents real world fact.

course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

***Prereq* =**{**(BIO_301, BIO_101), (BIO_399,BIO_101), (CS_190, CS_101),…**}

Keys

- A superkey is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation.
- Let **K** $\subseteq$ **R**, **K** is a **superkey** of ***R*** if values for **K** are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID , name} are both super keys of *instructor.*
  - ***Instructor(ID, Name, Dept_Name, Salary)***
  - *i.e* ***R={ID, Name, Dept_Name, Salary} & assume K= {ID, name}***
- A superkey may contain **extreneous attributes**.
  - *Example: In {ID, Name} ,* ***name*** *is a* ***extreneous attribute****, which not really required to identify a row uniquely, in other words, only ID is enough to identify rows uniquely.*

Keys

- **Minimal super key** is called candidate key.
- Super key **K** is a **candidate key** if **K** is minimal.
- Minimal Super key means-Minimum number of attribute of K required to identify every row uniquely.

  Example: K= {*ID, Name*} is a Super key, but *Name* attribute is not necessary to identify each row uniquely. Name is extraneous

  Hence ID is minimum required attribute to identify every row uniquely

Therefore ID is candidate key for *Instructor*

- **Primary key** is a term used by the database designer to denote a candidate key.

Keys…

- Answer the following by understanding the requirements given below.
- **CUSTOMER(Custid, Name,Mid_Name,LastName, City, phone, email)**

  ACCOUNT(AccNo, CustId, Intr_CustId, AccType, Branch)
- Is **(Phone, Email)** is a Super Key for CUSTOMER? If yes, is it a minimal Super key ?

In Bank every customer will have Unique **CustomerID**. **Intr_CustId** is the Customer Id of customer who is introducing a new customer to the Bank.

A customer can have multiple accounts such as SB, Current, Loan etc. Every **Accno** is unique. **Name , Mid_name and Last_Name** information about a customer must be distinguishable from other customers. **Phone** - phone number of the customer. **Email-** Email Id of the customer. Every Customer has a unique phone number and email id.

Keys

- Is **(Phone, Email)** is a Super Key , if yes is it a minimal Super key ?

- **K= (Phone, Email), Super key –YES**
- **IS K minimal Super Key?**
  - **K - Phone ={Email} Email alone can be used to identify every tuple uniquely, hence K is not minimal.**
  - **Email is minimal Super key & hence it is a Candidate key.**
  - Another possibility is
  - **K - Email ={ Phone}**
  - **Phone is minimal Super key & hence it is a Candidate key.**
  - **In this case Phone, Email & CustId , (Name, Mid_Name, Last_Name) is also Candidate Key.**

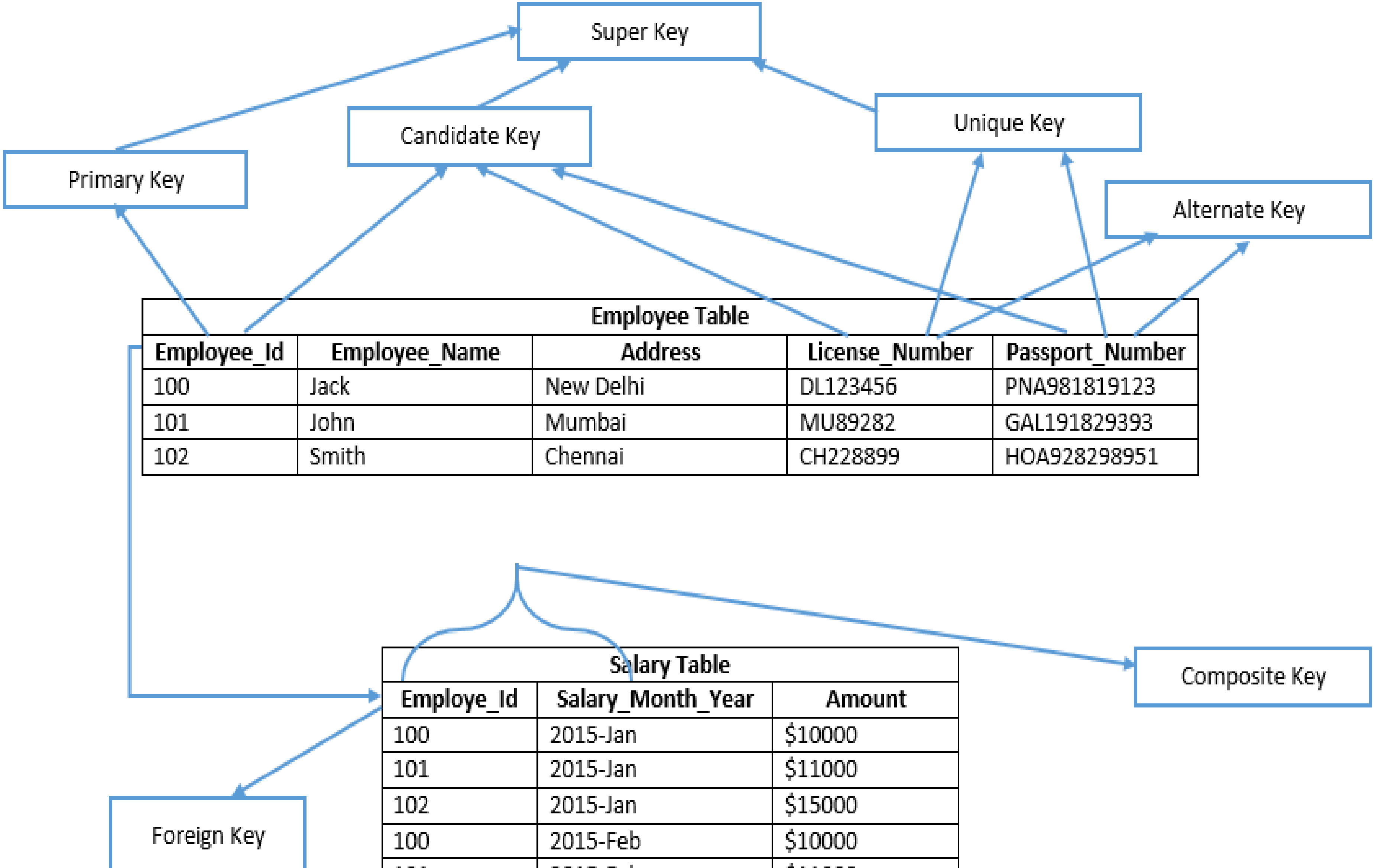
- Is **(AccNo, CustId)** a Super Key in **ACCOUNT(AccNo, CustId, Intr_CustId, AccType, Branch) ,** if yes, is it a minimal Super key ?
- IS (Custid, Name, Mid_Name, LastName) a Super Key in CUSTOMER Relation?
  - **I**f Yes, is it minimal Super key ?
- List Possible minimal Super keys (Candidate Keys) in

ACCOUNT(AccNo, CustId, Intr_CustId, AccType, Branch)

CUSTOMER(Custid, Name, Mid_Name, LastName, City, phone, email)

A Relation may have multiple minimal Super Keys (Candidate Key)

Types of Keys



Employee Table				
Employee_Id	**Employee_Name**	**Address**	**License_Number**	**Passport_Number**
100	Jack	New Delhi	DL123456	PNA981819123
101	John	Mumbai	MU89282	GAL191829393
102	Smith	Chennai	CH228899	HOA928298951

Salary Table		
Employe_Id	**Salary_Month_Year**	**Amount**
100	2015-Jan	$10000
101	2015-Jan	$11000
102	2015-Jan	$15000
100	2015-Feb	$10000
101	2015-Feb	$11000
102	2015-Feb	$15000

Keys

- A Relation may have **multiple minimal Super Keys** (Candidate Key)

- One of them may be considered as Primary Key
  - Ex: **Cust_Id** in Customer may be Primary Key
- Remaining all Candidate Keys are called as **Alternate Keys.**
- There can be only **ONE Primary Key** for a relation- but it may be Simple or Composite primary key.
- Ex: Stud(RegNo, Course_Id, Grade) -simple

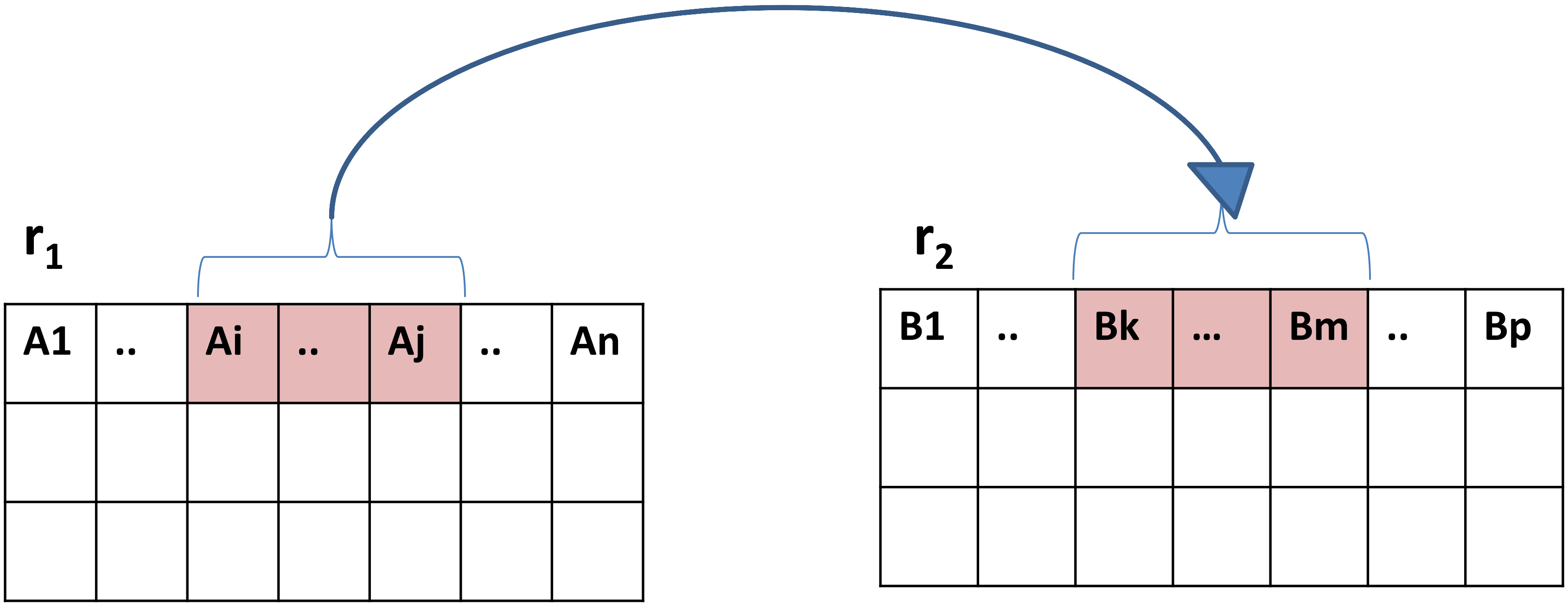
  Person(Name, Mname, Lname, Age) -composite

Foreign Keys-Referential Constraint

- Some attributes of a relation **r1**$(A_1, A_2, ..A_n)$ shares domains and derives values from primary key attributes of another(or same also possible) relation **r2**$(B_1, B_2, ..B_p)$.
- Such attributes of **r1** is called a **foreign key** referencing **r2**.
- The relation **r1** is called **referencing(Child) relation for** the foreign key dependency.
- The relation **r2** is called **referenced(Parent) relation** for the foreign key.

Foreign key can also be – Simple or Composite

Foreign Keys-Referential Constraint



If **Ai,..,Aj** attributes of **r1** derive values from primary key of **r2** say, **Bk,..Bm** then

(Ai,..,Aj) forms Foreign key (child columns), **r1** (child table) is **referencing relation**.

(Bk,...,Bm) forms parent columns, **r2** (Parent table) is **referenced relation** for the foreign key.

(Ai,..,Aj) derives values from **(Bk,...,Bm) .**

Example: Foreign Keys-Referential Constraint

SID	Name	Age
S101	Ram	
S102	Akshay	
S103	Santosh	

Students (parent table, Students (SID) is parent column)

SID	CNo	Year	Grade
S101	C10	2012	
S101	C11	2013	
S103	C11	2013	
S103	C10	2012	
~~S120~~	~~C13~~	2012	

Enrollment (child)

CNo (Child Column) –Foreign Key referencing CID in Courses Table (parent column)

SID (Child Column) –Foreign Key referencing SID (Parent Column) of **Students Table**

CID	C_Name	Credits	Duration
C10	E.Maths	4	
C11	CSc	4	
C12	Electronics	4	

Courses (parent table, Courses(CID) is Parent column)

Properties:

A Foreign key can contain-

- Only values present in the corresponding Parent Column/s.
- **NULL** values (unless additional **NOT NULL** constraint imposed)

Example: Primary key and Foreign key relationship (recursive) in same table

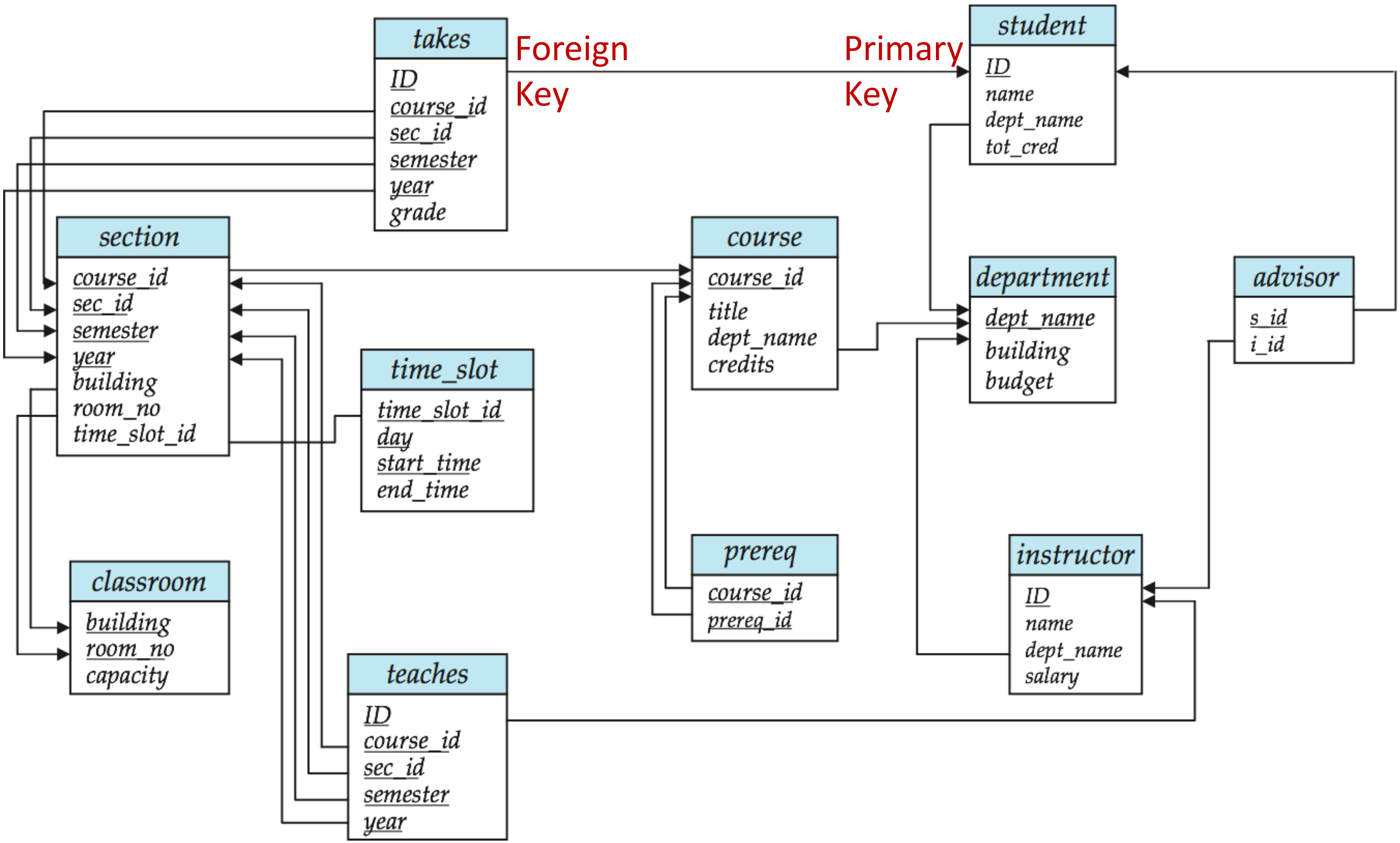
EMP table

EMPNO	ENAME	MGRNO
100		103
101		100
103		104
104		104
105		

MGRNO is the Employee number of Manger. Employee with EMpno 103 is the Manger for Employee with Empno 100. Therefore MGRNO is Foreign Key Referencing EMPNO

Insert / update / Delete should not violate primary key & foreign key relationship constraints

Schema Diagram for University Database



Visit: Later in ER model chapter

Relational Query Languages

- ***Query languages:*** Allow manipulation and retrieval of data from a database.

- Query Languages **!=** programming languages
  - QLs not intended to be used for complex calculations.
  - QLs support easy, efficient access to large data sets.

Formal Relational Query Languages

- Two mathematical Query Languages form the basis for “real” languages (e.g. SQL), and for implementation:
  - *Relational Algebra*: More operational(procedural), very useful for representing execution plans.
  - *Relational Calculus*: Lets users to describe what they want, rather than how to compute it. (Non-operational, *declarative*.)

Relational Algebra

- Query Language-
  Procedural & Non-Procedural
- There are a number of "pure" query languages:
  - The relational algebra is procedural,
  - The tuple relational calculus and domain relational calculus are nonprocedural.

- The relational algebra consists of a **set of operations** that take one or two relations as input and produce a new relation as their result.

- They **illustrate the fundamental techniques for extracting data** from the database.

-

Selection of tuples (σ)

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

comparisons operators **=, ≠, <, ≤, >,** and **≥**
connectives *and* (∧), *or* (∨), and *not* (¬).

- Select tuples with A=B and D > 5

- $\sigma_{A=B \text{ and } D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Quiz Q1:

$\sigma_{A<>B \text{ OR } D < 7}$ (r) has (1) 1 tuple (2) 2 tuples (3) 3 tuples (4) 4 tuples

Instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Result

ID	*name*	*dept_name*	*salary*
12121	Wu	Finance	90000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
83821	Brandt	Comp. Sci.	92000

Result of Instructors having salary more than $85000

Instructor

$$\sigma_{Salary > 85000}(\text{Instructor})$$

Selection of Columns (Attributes)- π

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- Select A and C attributes
- Projection
- $\pi_{A,\, C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Quiz Q2:

The projection operation (1) removes duplicates (2) does not remove duplicates

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Instructor

ID	*name*
22222	Einstein
12121	Wu
32343	El Said
45565	Katz
98345	Kim
76766	Crick
10101	Srinivasan
58583	Califieri
83821	Brandt
15151	Mozart
33456	Gold
76543	Singh

Result of Projection on ID and Name columns of Instructors relation.

$$\pi_{\text{ID, Name}}(\text{Instructor})$$

Discards duplicates, retains only one copy.

Joining two relations – Cartesian Product X

- Relations *r, s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Union, Intersection & Set Difference

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

Two conditions

*r and s must be of the **same arity**

***Ith** attribute in r and s must be from **same domain**

- Union:

r ∪ s

A	B
α	1
α	2
β	1
β	3

- Intersection

r ∩ s:

A	B
α	2

- Set Difference

r - s:

A	B
α	1
β	1

Is this true ?

$$r \cap s = r - (r - s)$$

Joining two relations – Natural Join

- Let *r* and *s* be relations on schemas *R* and *S* respectively. Then, the "natural join" of relations *R* and *S* is a relation on schema ***R* ∪ *S*** obtained as follows:
  - Consider each pair of tuples t_r from *r* and t_s from *s*.
  - If t_r and t_s have the **same value** on each of the attributes in $R \cap S$, add a tuple *t* to the result, where
    - ***t*** has the same value as t_r on ***r***
    - ***t*** has the same value as t_s on ***s***

A	B	C	D
α	1	α	a
β	2	γ	a

r

B	D	E
1	a	α
3	a	β

s

A	B	C	D	E
α	1	α	a	α

Equating attributes of the same name, and Projecting out one copy of each pair of equated attributes

Natural Join Example

- **Relations r, s:**

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

Cartesian product followed by SELECT(σ) operation. Selection is based on equality on common Attributes in both relations. Finally removes duplicate attributes

- **Natural Join**
  - r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Consider two relations *r*(*R*) and *s*(*S*).

***R* ∩ *S* = {*A1, A2, . . .,An*}.**

$$r \bowtie s = \Pi_{R \cup S}\left(\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots \wedge r.A_n = s.A_n}\, r \times s\right)$$

Quiz Q3: The natural join operation matches tuples (rows) whose values for common attributes are (1) not equal (2) equal (3) weird Greek letters (4) null

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Instructor

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Department

Instructor $\bowtie$ Department

ID	*name*	*salary*	*dept_name*	*building*	*budget*
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
12121	Wu	90000	Finance	Painter	120000
15151	Mozart	40000	Music	Packard	80000
22222	Einstein	95000	Physics	Watson	70000
32343	El Said	60000	History	Painter	50000
33456	Gold	87000	Physics	Watson	70000
45565	Katz	75000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
76543	Singh	80000	Finance	Painter	120000
76766	Crick	72000	Biology	Watson	90000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000

Figure 2.12 Result of natural join of the *instructor* and *department* relations.

theta join

The *theta join* operation is a variant of the natural-join operation that allows us to combine a selection and a Cartesian product into a single operation.

Consider relations ***r*** (***R***) and ***s***(***S***), and let **ɵ** be a predicate(condition) on attributes in the schema *R* **∪** *S*.

The **theta join** operation ***r*** ***s*** is defined as follows:

$$r \bowtie_\theta s = \sigma_\theta(r \times s)$$

It is equivalent to-

- Take the product **r X s**.
- Then apply $\boldsymbol{\sigma}_{\boldsymbol{\theta}}$ to the result.

As for **σ, ɵ** can be any Boolean-valued condition. Historic versions of this operator allowed only A θ B, **where θ is =, <, etc**.; hence the name “theta-join.”

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- Example:

$$\rho_x(E)$$

returns the expression *E* under the name *X*

- If a relational-algebra expression *E* has arity *n*, then

$$\rho_{x(A_1, A_2, \ldots, A_n)}(E)$$

returns the result of expression *E* under the name *X*, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

Banking Example

branch (*<u>branch_name</u>*, *branch_city*, *assets*)

customer (*<u>customer_name</u>*, *customer_street*, *customer_city*)

account (*<u>account_number</u>*, *branch_name*, *balance*)

loan (*<u>loan_number</u>*, *branch_name*, *amount*)

depositor (*<u>customer_name, account_number</u>*)

borrower (*<u>customer_name, loan_number</u>*)

Example Queries

- Find all loans of over $1200

$\sigma_{amount\ >\ 1200}\ (loan)$

Loan

loan_number	*branch_name*	*amount*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

- Find the loan number for each loan of an amount greater than $1200 and less than $2000

$$\Pi_{loan\_number}\ (\sigma_{amount\ >\ 1200\ \wedge\ amount\ <2000}\ (loan))$$

- Find the names of all customers who have a loan, an account, or both, from the bank

$$\Pi_{customer\_name}\ (borrower) \cup \Pi_{customer\_name}\ (depositor)$$

Example Queries

- Find the names of all customers who have a loan at the Perryridge branch.

$$\Pi_{customer\_name}\ (\sigma_{branch\_name=\text{“Perryridge”}}$$

$$(\sigma_{borrower.loan\_number\ =\ loan.loan\_number}(borrower\ x\ loan)))$$

Borrower

customer_name	*loan_number*
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Loan

loan_number	*branch_name*	*amount*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

(*borrower x loan*)

customer_name	borrower.loan_number	loan.loan_number	branch_name	amount
Adams	L-16	L-11	Round Hill	900
Adams	L-16	L-14	Downtown	1500
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Adams	L-16	L-17	Downtown	1000
Adams	L-16	L-23	Redwood	2000
Adams	L-16	L-93	Mianus	500
Curry	L-93	L-11	Round Hill	900
Curry	L-93	L-14	Downtown	1500
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Curry	L-93	L-17	Downtown	1000
Curry	L-93	L-23	Redwood	2000
Curry	L-93	L-93	Mianus	500
Hayes	L-15	L-11		900
Hayes	L-15	L-14		1500
Hayes	L-15	L-15		1500
Hayes	L-15	L-16		1300
Hayes	L-15	L-17		1000
Hayes	L-15	L-23		2000
Hayes	L-15	L-93		500
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
Smith	L-23	L-11	Round Hill	900
Smith	L-23	L-14	Downtown	1500
Smith	L-23	L-15	Perryridge	1500
Smith	L-23	L-16	Perryridge	1300
Smith	L-23	L-17	Downtown	1000
Smith	L-23	L-23	Redwood	2000
Smith	L-23	L-93	Mianus	500
Williams	L-17	L-11	Round Hill	900
Williams	L-17	L-14	Downtown	1500
Williams	L-17	L-15	Perryridge	1500
Williams	L-17	L-16	Perryridge	1300
Williams	L-17	L-17	Downtown	1000
Williams	L-17	L-23	Redwood	2000
Williams	L-17	L-93	Mianus	500

customer_name	*borrower. loan_number*	*loan. loan_number*	*branch_name*	*amount*
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Hayes	L-15	L-15	Perryridge	1500
Hayes	L-15	L-16	Perryridge	1300
Jackson	L-14	L-15	Perryridge	1500
Jackson	L-14	L-16	Perryridge	1300
Jones	L-17	L-15	Perryridge	1500
Jones	L-17	L-16	Perryridge	1300
Smith	L-11	L-15	Perryridge	1500
Smith	L-11	L-16	Perryridge	1300
Smith	L-23	L-15	Perryridge	1500
Smith	L-23	L-16	Perryridge	1300
Williams	L-17	L-15	Perryridge	1500
Williams	L-17	L-16	Perryridge	1300

customer_name
Adams
Hayes

Example Queries

Find the names of all customers who have a loan at the **Perryridge** branch but do not have an account at any branch of the bank.

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Perryridge”}}(\sigma_{borrower.loan\_number = loan.loan\_number}(\text{borrower x loan}))) - \Pi_{customer\_name}(\text{depositor})$$

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Borrower

loan_number	branch_name	amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

Loan

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Depositor

Example Queries

Find the names of all customers who have a loan at the **Perryridge** branch.

- Query 1

$\Pi_{customer\_name}$ ($\sigma_{branch\_name = \text{“Perryridge”}}$ ($\sigma_{borrower.loan\_number = loan.loan\_number}$ (borrower x loan)))

- Query 2

$\Pi_{customer\_name}$($\sigma_{loan.loan\_number = borrower.loan\_number}$ (($\sigma_{branch\_name = \text{“Perryridge”}}$ (loan))x borrower))

Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P is a predicate on attributes in E_1
  - $\prod_s(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_x (E_1)$, x is the new name for the result of E_1

Assignment Operation

- The assignment operation ($\leftarrow$) provides a convenient way to express complex queries.
  - Write query as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query.
  - Assignment must always be made to a temporary relation variable.

Assignment Operation

- Example: $temp1 \leftarrow \prod_{R-S}(r)$
  $temp2 \leftarrow \prod_{R-S}((temp1 \times s) - \prod_{R-S,S}(r))$
  $result = temp1 - temp2$

Bank Example Queries

Find the names of all customers who have a loan and an account at bank.

$$\Pi_{customer\_name}\ (borrower) \cap \Pi_{customer\_name}\ (depositor)$$

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Borrower

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Depositor

Bank Example Queries

Find the name of all customers who have a loan at the bank and the loan amount

$$\Pi_{customer\_name,\ loan\_number,\ amount}\ (borrower \bowtie loan)$$

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

borrower

loan_number	branch_name	amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

loan

Bank Example Queries

Find the names of all customers who have a loan at the **Downtown** branch.

$\Pi_{customer\_name} (\sigma_{branch\_name = \text{"Downtown"}} ((borrower \bowtie loan)))$

customer_name	*loan_number*
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

borrower

loan_number	*branch_name*	*amount*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

loan

Find the names of all customers who have a balance amount >500 $

$$\Pi_{customer\_name}(\sigma_{balance > 500}(Account) \bowtie Depositor)$$

account_number	*branch_name*	*balance*
A-101	Downtown	500
A-215	Mianus	700
A-102	Perryridge	400
A-305	Round Hill	350
A-201	Brighton	900
A-222	Redwood	700
A-217	Brighton	750

Account

customer_name	*account_number*
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Depositor

Aggregate Functions and Operations

- **Aggregation function** takes a collection of values and returns a single value as a result.

  avg: average value
  min: minimum value
  max: maximum value
  sum: sum of values
  count: number of values

- **Aggregate operation** in relational algebra

$$\boxed{{}_{G_1, G_2, \ldots, G_n} \mathcal{G}_{F_1(A_1), F_2(A_2), \ldots, F_n(A_n)}(E)}$$

  E is any relational-algebra expression/ a relation

  - ***G_1, G_2 …, G_n*** is a list of attribute/s on which to group (can be empty)
  - Each ***F_i*** is an aggregate function
  - Each ***A_i*** is an attribute name on which aggregate function applied.

- Note: Some books/articles use γ (gamma) instead of $\mathcal{G}$ (Calligraphic G)

Aggregate Operation – Example

- Relation *r*:

A	***B***	***C***
α	α	7
α	β	7
β	β	3
β	β	10

- $\mathcal{G}_{\textbf{sum(c)}}(\mathrm{r})$

sum(*c*)
27

- $_{\textbf{A}}\mathcal{G}_{\textbf{sum(c)}}(\mathrm{r})$

A	**sum**(*c*)
α	14
β	13

- **What is the result for the following expression ?**

$_{\textbf{A,B}}\mathcal{G}_{\textbf{sum(c)}}(\mathrm{r})$

Aggregate Operation – Example

- Find the average salary in each department

$$_{dept\_name}\mathcal{G}_{\text{avg}(salary)}(instructor)$$

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join.
- Uses *null* values:
  - *null* signifies that the value is unknown or does not exist
  - All comparisons involving *null* are (roughly speaking) **false** by definition.

Outer Join – Example

- Relation *loan*

loan_number	*branch_name*	*amount*
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relation *borrower*

customer_name	*loan_number*
Jones	L-170
Smith	L-230
Hayes	L-155

Outer Join – Example

- **Join**

  loan* ⋈ *borrower

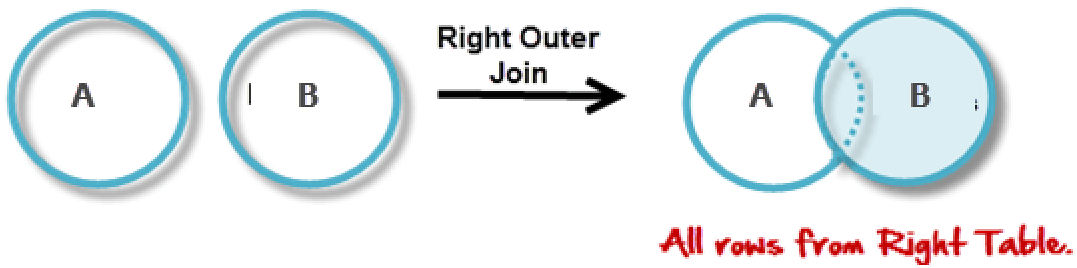
loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

- **Left Outer Join**

  loan* ⟕ *borrower

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

Outer Join – Example



- Right Outer Join

  loan ⟖ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	***null***	***null***	Hayes

- **Full Outer Join**

  loan* ⟗ *borrower

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	***null***	***null***	Hayes

Composition of Relational Operations

“Find the names of all instructors in the Physics department.”

$$\Pi_{name}\,(\sigma_{dept\ name\ =\text{“Physics”}}\,(instructor))$$

Exercise

- Consider the following Student database:
- Student(Id,Stdname,City,tutor)
- Enrolment(Id,Code,Marks)
- Course(Code,title,Department)

Write the following queries in relational algebra ,assume that the attribute tutor is the Id of the tutor.

a) List all courses offered by Computer department

b) List the names of the students along with the names of the course opted.

c) List the names of students who have scored >80 marks.

d) List the details of students who have taken up the course offered by Physics department

e) Display the Student Id along with the total marks

a) $\pi_{code,\ title}(\sigma_{Department = \text{"computer"}}(course))$

b) $\pi_{stdname,\ title}((student \bowtie enrolment) \bowtie course)$

c) $\pi_{stdname}(\sigma_{marks > 80}(student \bowtie enrolment))$

d) $\pi_{ID,\ stdname}(\sigma_{Department = \text{"physics"}}(course) \bowtie enrolment \bowtie student)$

e) ${}_{ID}\mathcal{G}_{sum(marks)}(enrolment)$

Exercise

Consider a relations **A(ID, Name, Age) , B(EID, Phone, City) ,**

C(ID, Salary, DeptName)

Note: EID and ID derived from Same domain

Write a relational Algebraic expression –

- To find ID, Name, Phone of Employees.
- To find Name of Employees having **Salary>90000**
- To find DeptName and total salary of each department.
- To find Name of Employees who from city - **Manipal**

Solution: A(ID, Name, Age) , B(EID, Phone, City) , C(ID, Salary, DeptName)

Note: EID and ID derived from Same domain

- To find ID, Name, Phone of Employees.
  - **PROJECT$_{\text{ID, Name, Phone}}$ (A Theat$_{\text{ID=EID}}$ B)**
- To find Name of Employees having **Salary>90000**
  - **PROJECT $_{\text{Name}}$ (SELECT $_{\text{Salary>90000}}$ (A N.JOIN C))**

    other way is
  - **PROJECT $_{\text{Name}}$ ((SELECT $_{\text{Salary>90000}}$ (C)) N.JOIN A)**
- To find DeptName and total salary of each department.
  - **$_{\text{DeptName}}$ G $_{\text{SUM(Salary)}}$ (C)**
- To find Name of Employees who from city - **Manipal**
  - **PROJECT** $_{\text{Name}}$ (A **Theta** $_{\text{(ID=EID AND City='Manipal')}}$ B)

END