A 11 110 Duppender Billy duppender all sets a fill a first	HRU Algorithm		
Roll-UP: Symmatize Data, dimension reduction, less details!	Assuming 'a' is already materialized, we should materialize?	what are the best 3 other views tha	Relational trigebra L'xamples
Som (Selds) PROM AUSALES F, Shore S GILDUW BY country, newID, EUSKIZ	100	table of values. "Gain" is the top	Find name of actors in Javs and Spongebob "Joining with
From Ansales F, Shoe S, Custon Henzin united St county, Henzin , Age	a (a) su	btracted with the current one.	Jows ← Trame (( σ title= "Jous" (Movie)) ⋈ StorsIn) ⋈ MovieStor
· so when GROUP BY ottr. replaced by swith more generic, or less attr	b solution h s	rst Choice Second Choice Third Choi	Storigenes - The Contract Storigenese Character Stories
Drill-Down: higher level summary to love level, more dimen	sions C	<b>D*6=300 -</b> - 5*6=150 25*2=50 25*1=25	Res & Jaue N Spongebob
Gina actails FROM AUSALES F. Store S. Curton Som Gales) (Gilovi BY country, iter/LD, age	20 / 50 / 40	0*3=240 30*3=90 30*3=90	Find nome of actors in <u>all</u> movies "Has to be connected in some voy"
S Span Allsales F, Shee S, Custo		0*4=280 20*4=80 20*4=80	Thank (TI star ID, movieID (Stars In) / T movieID (Movie)) & Moviestar)
HemID County ThemID is GROUP BY City, HemID, ruge		0*3=180 60*3=120 -	Find all profs in exactly oll the committees prof piper is in
when GROUP BY attr. replaced by smith more specific, or more Slicing pides a specific value for one of the dimensions	3 y y y y	9*2=198 49*2=98 <b>49*2=98</b> 0*2=180 40*2=80 30*2=60	R2 - Th commname ( of profinames 'piper' ( committee )) committee piper is in
SUNCALES (GROUPBY COUNTY, ItenID,		99 49 39	R3 - TI commone (committee) - R2 committee piper is not in
S WHERE S. country - Santa C	Remember that node includes itself		Committee / R2 - Th prefrome (committee ▷ R3)
. when GROUP BY loses an attr. 2. WHERE has specified cond.	Data warehousing: Finding number o		piper's comitee committees from R3
Dicing: sabcube by picting specific values for multiple dimen	Data warehouse with dimensions D1, 99, 24, 4, and 19 values, . Sparsity is 1		Find all profs who have offices in at least all the buildings that Piper has offices in
· ( GROUP BY city, ItenID, age	The size of the full cube is (999+1)(99+1)(24+1 sparsity factor of 10%, the size of the sparse c	)(4+1)(19+1) = 250,000,000 tuples. With a	"all"
HEATT (1) Optomore C. A MATTER ONLY : red ANO STATE = Y	A <sup>1</sup> 25,000,000 tuples.		For each peson, find all pizzes the peson ets that aren't surved by any pizzeria the Person frequents: Eats - TTNAME, pizza (Frequents X Serves)
where shorified in WHERE For it dimension, group by differ elements?	Apriori Algorithm minimum suppor T1 tree, cup, paper		ATE for each Bats(name, pizza) Pizzarias the person frequents
Pivoting state cube to poulde alternative pres. of data	T2 book, tomato, pen		has to do with domain some domain Serves (pizza, pizzania, price)
GROUP BY stare ID, cu	T4 tomato, pen, cup		Find names of people who frequent only pizzarias serving at least one pizza
A GROUP BY STORE ID, TH			they eat all the pizzarios someone frequents
ThenID St. when GB dalf shuff I gress.	1. Find possible itemsets, and supp C_1 = {tree}: 3, {cup}: 2, {paper}: 2, {bo		TT nome (Person) - Π nome (Frequent's - Π nome, pizzaria (Evis M Sarvas)) all the pizzorias
Decomposing into BNF: #1 lossless join	2. Find frequent itemsets of size k, d F_1 = {{tree},{cup}, {paper}, {book}, {		withint sorting pizza someone eats Find names of people who frequent all the pizzarios serving at least one pizza
I Given FDS F, compute F: minMal cover for F	It's these values because they occur	more then 33% (2/6 times).	they eat.
2. Decompose using F'if violating 3NF similar to BCNF- all the way to BCNF	2. Go up one k, and ONLY use possib C_2 = {{tree, cup}: 1 F_2 = {{tree, paper},		∏ nome (Person) - ∏nome (∏nome, pizzaria (Eorts ⋈ Serves)-Frequents)
3. After each decomp, Identify set of dependences N in F' that are not preserved by the decomp.	{tree, paper}: 2 {tree, book}: 2 {book, pen}, {tom	ato, pen}}	he must expensive pass cost
A grand For each X-> b in N create a relation	{tree, pen}: 1 finally return all the		∏ cost (Pass) - ∏cost (Pcost -> newcost (Poss) ≥ newcost cost (Poss))
Ry (XUb) & add it to the decomposition	{cup, paper}: 1 {cup, book}: 0 {cup, tomato}: 1	•	The min price book TT ison(Book) - TT bl.ison(σbl.price > b2.price(ρ(bl.Book) χρ(b2,Book))) TT ison(Book) - TT bl.ison(σbl.price > b2.price(ρ(bl.Book) χρ(b2,Book)))
Valid: The patterns hold in general Data mining is the exploration and Novel: We did not know the pattern before hard.	{cup, pen}: 1 Apriori speeds up ca	lculating association rules ation that each subset of a	TT ison (Book) - II billison (Oblephe > Disprese ( Constraint in the state one other P tooks with a grander price then at least one other Book w/ smallest price tooks
(huciness intelligence)	ovel, {paper, tomato}: 0 frequent itemset mu	st also be a frequent itemset Ex.	Book w/ smallest proc book Find Pizzania serving cheapest pepperoni pizza. For thes, neturn all of chapest pepperoni
Under-	{book, tomato}: 1 rice only appears on terns {book, pen}: 3 more times with any	e time, it can't appear two or thing else.	pizzaria (O pizza = pepperoni (Serves)) - Tisipizzaria (ONote (P(Serves, SI) × P(Serves, SZ))
standable: We can interpret and comprehend the patterns SQL Examples A	gradate Operations	ess-less join, when we repoin the lecomposed tables, no loss of inform	
Find students who've taken <b>all</b> classes (division example)	G, MIN, MAX, SUM, COUNT	(no <u>there</u> or <u>less</u> tuples) ow to <u>direct lossless</u> : common attribut ave to be a superbey in X1 or X2	
CELEOT	ELECT country, city, count(*)	check common atts. of X1 & X2, ex. BC BC <sup>+</sup> = FBC3 which is not superkey to	
FROM student s	HERE country LIKE '%a%' this is for che		WHERE commname IN (SELECT *
FROM class c) student	ROUP BY country, city	filtering sums/maxes/mins etc,	FROM Committee WHERE profname = 'Piper')
(SELECT e.cname courses student enrolled	RDER BY count(*) DESC is like a where	condition for group by	RA Statement "I a potnome (So. potname = potri A ci. patriane = cz. patriane (Poi (committee)) × Pos (committee))
FROM enrolled e is enrolled in in N	Il Values tuples can have null values. unkr can use IS NULL (IS NOT NULL) in WHERE		
			Find all professors who are in at least all those commitees that professor Piper is i
	Il and 3-valued logic		SELECT DISTINCT c.profname
SELECT sname FROM student s	- Unknown or the = the false = unknown unknown = unknown	-Anula	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commame
SELECT sname	<ul> <li>Walknown or the = the and a which own unbrown unbrown = unbrown</li> <li>which own and the = index unbrown Mich = index unbrown = unbrown = unbrown = 5</li> </ul>	nown Anull	SELECT DISTINCT c.profname FROM Committee c
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum	Malicaewan Or TARE = TARE     Failse + Michaewan     Failse + Michaewan     unichewan = Wachaewan     worker and the failse + Michaewan     worker = Michaewan     swatchewan = Wachaewan     Sa     work workers work = Wachaewan	LIKE is used for string	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commame FROM Committee a WHERE profname = 'Piper') EXCEPT
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname	Micrown Or Trill = Trill False - Micrown     false - Micrown     false - Micrown     wichown a willown     willown     wichown - Willown     wichown	LIKE is used for string	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commane FROM Commitee a WHERE profname = 'Piper') EXCEPT (SELECT commname FROM committee a
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all	Markannyn Or Thre = Thre False = Michannyn     Markannyn = Wichannyn     Wichannyn Wic	LIKE is used for string matching •stands for any one character	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commane FROM Commitee a WHERE profname = 'Piper') EXCEPT (SELECT commname FROM committee a WHERE, profname=.profname) Find all enclosures that were never visited by the visitors born before 2000-01-01
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all CREATE VIEW Temp(major, average) AS SELECT major, average	Withown Or The s The faile + whown the set of the	LIKE is used for string matching • _ stands for any one character • % is 0 or more arbitrar characters	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commane FROM Committee a WHERE profname = 'Piper') EXCEPT (SELECT commname FROM committee a WHERE a, profname=. profname) Find all enclosures that were never visited by the visitors born before 2000-01-01 Y SELECT E.id Visitor(visitorID, firstName, lastName, dateofBirth)
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all CREATE VIEW Temp(major, average) AS SELECT S.major, AVG(s.age) AS average FROM Temp FROM Student S WHERE average = (SELECT r	Waterwin Or The s Thile Faile + winchown winchown = winchown	LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters or and but didn't B	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commane FROM Commitee a WHERE profname = 'Piper') EXCEPT (SELECT commname FROM committee a WHERE profname=c.profname) Find all enclosures that were never visited by the visitors born before 2000-01-01 Y SELECT E.id FROM Employee E Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E EXCEPT ExcEPT ExcEPT
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all CREATE VIEW Temp(major, average) AS SELECT S.major, AVG(s.age) AS average FROM Temp	Waterwin Or The s Thile False + Init Annual False + Init Annual False + Init Annual Water Annua	LIKE is used for string matching stands for any one character - % is 0 or more arbitrar characters or and but didn't B see not been in' R _s	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT command FROM Committee a WHERE profname = 'Piper') EXCEPT (SELECT commname FROM committee a WHERE a.profname=c.profname) Find all enclosures that were never visited by the visitors born before 2000-01-01 V SELECT E.id FROM Employee E Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E Visits(passID, enclosureID) EXCEPT Enclosure(id, latitude, longitude) (SELECT E.id FROM Visitor v, Visits vs, Enclosure EZ
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all CREATE VIEW Temp(major, average) AS SELECT S.major, AVG(s.age) AS average FROM Temp FROM Student S GROUP BY major, getting average age selecting minimum over all Get all students using natural join	wataswin         Ori         102. st.         Thick         Thick           wataswin         Ori         Thick         Thick         Thick           wataswin         Thick         Thick         Thick         Thick           majors         Set operations         Thick         Thick         Thick           najors         UNION usually used for or         Thick         Thick         Thick           najors         Thick         EXCEPT usually used for Ar         Thick           najors         Thick         Thick         Thick	LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters or and but didn't B ave not been in'	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commane FROM Committee a WHERE profname = 'Piper') EXCEPT (SELECT commname FROM committee a WHERE a profname=c.profname) Find all enclosures that were never visited by the visitors born before 2000-01-01 Y SELECT E.id Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E Visitor(visitorID, firstName, lastName, dateofBirth) FROM Visitor v, Visits vs, Enclosure EZ WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id AND v.DateOfBirth < Date('2001-01-01')
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all <b>CREATE VIEW Temp(major, average) AS</b> SELECT S.major, AVG(s.age) AS average FROM Temp FROM Student S GROUP BY major, getting average age selecting minimum over all <b>Get all students using natural join</b> SELECT * SELECT *	witchewin         Or         Thile         S         Thile         Th	LIKE is used for string matching stands for any one character stands for any one character stands for any one character stands for any one characters stands for any one characters 	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commane FROM Committee a WHERE profname = 'Piper') EXCEPT (SELECT commane FROM committee a WHERE a.profname=c.profname) Find all enclosures that were never visited by the visitors born before 2000-01-01 Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E Visitor(visitorID, firstName, lastName, dateofBirth) FROM Visitor v, Visits vs, Enclosure EZ WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id AND v.DateofBirth < Date('2001-01-01') Nature Find all the animals of species cat that are taken care of by 'Sam Smith'
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all <b>CREATE VIEW Temp(major, average) AS</b> SELECT S.major, AVG(s.age) AS average FROM Temp FROM Student S GROUP BY major, getting average age Select s.maior, average age Get all students using natural join SELECT * SELECT * SELEC	Makawan         Or         The s         The s <tht< td=""><td>LIKE is used for string matching stands for any one character - % is 0 or more arbitrar characters or and but didn't B ave not been in' - Standard Reference - Standar</td><td>SELECT DISTINCT c.profname       Committee ( Committee ( Committee )         FROM Committee c       WHERE NOT EXISTS (SELECT commane FROM committee a WHERE profname = 'Piper')         EXCEPT       (SELECT commname FROM committee a WHERE a.profname=c.profname)         Find all enclosures that were never visited by the visitors born before 2000-01-01         Y       SELECT E.id         Visitor(visitorID, firstName, lastName, dateofBirth)         FROM Employee E       Visitor(sistorID, firstName, lastName, dateofBirth)         EXCEPT       Enclosure(id, latitude, longitude)         (SELECT E.id       Visits(passID, enclosureID)         EXCEPT       Enclosure(id, latitude, longitude)         (SELECT E.id       Visits(passID AND vs.enclosureID = ez.id         AND v.DateofBirth &lt; Date('2001-01-01')</td>       Nuture find all the animals of species cat that are taken care of by 'Sam Smith'         Vistor       Find all the animals of species cat that are taken care of by 'Sam Smith'         C       ELECT employeeID         FROM TakesCareOf NATURAL INNER JOIN Employee       Enclosure</tht<>	LIKE is used for string matching stands for any one character - % is 0 or more arbitrar characters or and but didn't B ave not been in' - Standard Reference - Standar	SELECT DISTINCT c.profname       Committee ( Committee ( Committee )         FROM Committee c       WHERE NOT EXISTS (SELECT commane FROM committee a WHERE profname = 'Piper')         EXCEPT       (SELECT commname FROM committee a WHERE a.profname=c.profname)         Find all enclosures that were never visited by the visitors born before 2000-01-01         Y       SELECT E.id         Visitor(visitorID, firstName, lastName, dateofBirth)         FROM Employee E       Visitor(sistorID, firstName, lastName, dateofBirth)         EXCEPT       Enclosure(id, latitude, longitude)         (SELECT E.id       Visits(passID, enclosureID)         EXCEPT       Enclosure(id, latitude, longitude)         (SELECT E.id       Visits(passID AND vs.enclosureID = ez.id         AND v.DateofBirth < Date('2001-01-01')
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all <b>CREATE VIEW Temp(major, average) AS</b> SELECT S.major, AVG(s.age) AS average FROM <b>Temp</b> FROM Student S Get all students using natural join SELECT * SELECT *	witchewin         Or         Thile         S         Thile         Th	LIKE is used for string matching stands for any one character - % is 0 or more arbitrar characters - % is 0 or more arbitrar - % is 0 or more arbitrar	SELECT DISTINCT c.profname FROM Committee c       committee / Teammant. ( dpreame - foor ( committee))         WHERE NOT EXISTS (SELECT commaname FROM committee a WHERE profname = 'Piper') EXCEPT (SELECT commaname FROM committee a WHERE a.profname=c.profname)         Find all enclosures that were never visited by the visitors born before 2000-01-01         Y         SELECT E.id         Visitor(VisitorID, firstName, lastName, dateofBirth) FROM Employee E         Visitor(visitorID, firstName, lastName, dateofBirth)         FROM Visitor v, Visits vs, Enclosure EZ WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id AND v.DateofBirth < Date('2001-01-01')
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS(SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all CREATE VIEW Temp(major, average) AS SELECT S.major, AVG(s.age) AS average FROM Temp FROM Student S SELECT S.major, average age Selecting minimum over all Get all students using natural join SELECT * SELECT * SELECT * FROM student s, enrolled e WHERE s.snum = e.snum Enrolled E Find the names of all movie stars who've been in a movie SELECT DISTINCT Name Need distinct or else can Hide of the set of the	Makeswon         Ort         State           Indication         Indication         Indication           Indication         Indication         Indication <td>LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters or and but didn't B ave not been in' • § 5 • § 5 •</td> <td>SELECT DISTINCT c.profname       committee / Transmet ( Option - option</td>	LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters or and but didn't B ave not been in' • § 5 •	SELECT DISTINCT c.profname       committee / Transmet ( Option - option
SELECT sname FROM student s WHERE NOT EXISTS (SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all <b>CREATE VIEW Temp(major, average) AS</b> SELECT S.major, AVG(s.age) AS average FROM <b>Temp</b> FROM Student S GROUP BY major, getting average age selecting minimum over all <b>Get all students</b> , enrolled e FROM Student S, enrolled e FROM Student S, enrolled E FROM Student S, enrolled E Find the names of all movie stars who've been in a movie SELECT <u>DISTINCT</u> Name Need distinct or else can FROM Starsin s, Moviestar m get movie star name	witcown         Or         ThiL         *         ThiL         *           witcown         witcown         witcown         witcown         witcown         witcown           najors         Set operations          UNION usually used for or         intrace of a point           najors          UNION usually used for or         intrace of a point         intrace of a point           najors          UNION usually used for A         intrace of a point         intrestown           noin	LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters or and but didn't B ave not been in' • § 5 •	SELECT DISTINCT c.profname       committee / Transmet. ( dprende + pper ( committee))         WHERE NOT EXISTS (SELECT commane FROM Committee a WHERE profname = 'Piper')       EXCEPT (SELECT commane FROM committee a WHERE a.profname=.c.profname)         Find all enclosures that were never visited by the visitors born before 2000-01-01         Y       SELECT E.id Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E Visits(passID, enclosureID) EXCEPT EXCEPT Enclosure(id, latitude, longitude) (SELECT EZ.id FROM Visitor v, Visits vs, Enclosure EZ WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id AND v.DateOfBirth < Date('2001-01-01') SELECT employeeID         Nuture I or SELECT end Committee of Species cat that are taken care of by 'Sam Smith' SELECT employeeID         FROM TakesCareOf NATURAL INNER JOIN Employee WHERE animalID IN (SELECT id from Animal WHERE Species='Cat') AND firstName = 'Same' AND lastName = 'Smith' Find the names of saliors who have reserved at least 2 boats SELECT s.sname FROM Saliors s, Reserves r1, Reserves r2
SELECT sname FROM student s WHERE NOT EXISTS (SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all <b>CREATE VIEW Temp(major, average) AS</b> SELECT major, average) AS SELECT Simajor, AVG(s.age) AS average FROM Temp FROM Student S Get all students using natural join SELECT * FROM student s, enrolled e FINOL BY major, getting average age SELECT * FROM Student s, enrolled e Find the names of all movie stars who've been in a movie SELECT DISTINCT Name FROM Stars m get movie star name WHERE s.starsid = m.starid MUICAL AND	witchewin         Ort         5         Thick         5           witchewin         witchewin         witchewin         witchewin         Witchewin           witchewin         witchewin         witchewin         Witchewin         Witchewin         Witchewin           witchewin         witchewin         witchewin         Witchewin	LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters or and but didn't B we not been in' • * * * * * * * * * * * * * * * * * * *	SELECT DISTINCT c.profname       committee / Transmet. ( dprende + fper ( committee))         WHERE NOT EXISTS (SELECT commaname       FROM Committee a         WHERE profname = 'Piper')       EXCEPT         (SELECT commaname       FROM committee a         WHERE a.profname= c.profname       FROM committee a         WHERE a.profname=c.profname)       WHERE a.profname=c.profname)         Find all enclosures that were never visited by the visitors born before 2000-01-01         Y       SELECT E.id         Visitor(visitorID, firstName, lastName, dateofBirth)         FROM Employee E       Visits(passID, enclosureID)         EXCEPT       Enclosure(id, latitude, longitude)         (SELECT EZ.id       Visitor(Visitor)D, enclosureID)         EXCEPT       Enclosure(id, latitude, longitude)         (SELECT EZ.id       FROM Visitor v, Visits vs, Enclosure EZ         WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id       AND v.DateofBirth < Date('2001-01-01')
SELECT sname FROM student s WHERE NOT EXISTS (SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all <b>CREATE VIEW Temp(major, average) AS</b> SELECT S.major, AVG(s.age) AS average FROM <b>Temp</b> FROM Student S GROUP BY major, getting average age selecting minimum over all <b>Get all students</b> , enrolled e FROM Student S, enrolled e WHERE s.snum = e.snum SELECT * SELECT	witcown         Or         Thile         Thile         Thile           witcown         witcown         witcown         witcown         witcown           witcown	LIKE is used for string matching stands for any one character s is 0 or more arbitrar stands for any one characters s is 0 or more arbitrar stands for any one characters s is 0 or more arbitrar stands for any one characters stands for any one characters stands stan	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commname FROM Committee a WHERE profname = 'Piper') EXCEPT (SELECT commname FROM committee a WHERE a profname = c.profname) Find all enclosures that were never visited by the visitors born before 2000-01-01 SELECT E.id Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E Visits(passID, enclosureID) EXCEPT Enclosure(id, latitude, longitude) (SELECT EZ.id FROM Visitor v, Visits vs, Enclosure EZ WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id AND v.DateOfBirth < Date('2001-01-01') SELECT employeeID SELECT employeeID SELECT employeeID SELECT enclosure(id, latitude, longitude) SELECT employeeID FROM TakesCareOf NATURAL INNER JOIN Employee WHERE animalID IN (SELECT id from Animal WHERE Species='Cat') AND firstName = 'Same' AND lastName = 'Smith' Find the names of sailors who have reserved at least 2 boats SELECT s.sname FROM Sailors s, Reserves r1, Reserves r2 WHERE r1.sid=s.sid AND r2.sid=s.sid AND r1.bid <> r2.bid Find the sailor id of the sailors with the highest rating SELECT s.sid
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS(SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all <b>CREATE VIEW Temp(major, average) AS</b> SELECT 5.major, AVG(s.age) AS average FROM <b>Temp</b> FROM Student S GROUP BY major GROUP BY major, getting average age selecting minimum over all Get all students using natural join SELECT * SELECT * FROM student s, enrolled e WHERE s.snum = e.snum FROM Student S, enrolled e FROM Student s, enrolled e WHERE s.snum = e.snum Enrolled E Find the names of all movie stars who've been in a movie SELECT DISTINCT Name Need distinct or else can FROM Starsin s, Moviestarm get movie star name WHERE s.starsid = m.starid duplicates! Find the departments that have more then one faculty member SELECT DISTINCT f1.deptid seeing if different faculties have same FROM Faculty f1, Faculty f2 department. WHERE f1.fid <> f2.fid AND f1.deptid = f2.deptid Find IDs of MovieStars who've been in a movie in 1944 or 1974	Mitsey         Or         THL         THL </td <td>LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters or and but didn't B ave not been in' • * * * * * * * * * * * * * * * * * * *</td> <td>SELECT DISTINCT c.profname       connittee / Transact ( optimite / provide a fROM Committee c         WHERE NOT EXISTS (SELECT commname       FROM Commitee a         WHERE profname = 'Piper')       EXCEPT         (SELECT commname       FROM committee a         WHERE not failed / Transact ( optimite commande)       WHERE profname = 'Piper')         EXCEPT       (SELECT commname         FROM committee a       WHERE a profname: profname)         Y       SELECT E.id       Visitor(visitorID, firstName, lastName, dateofBirth)         FROM Employee E       Visits(passID, enclosureID)         EXCEPT       Enclosure(id, latitude, longitude)         (SELECT E.id       Visits(passID, enclosureID)         EXCEPT       Enclosure(id, latitude, longitude)         (SELECT E.Z.id       FROM Visitor v, Visits vs., Enclosure EZ         WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id       AND v.DateOfBirth &lt; Date('2001-01-01')</td> SELECT employeeID       FROM TakesCareOf NATURAL INNER JOIN Employee         FROM TakesCareOf NATURAL INNER JOIN Employee       WHERE animalID IN (SELECT id from Animal WHERE Species='Cat') AND firstName = 'Same' AND lastName = 'Smith'         Find the names of sailors who have reserved at least 2 boats       SELECT s.sname         FROM Sailors s, Reserves r1, Reserves r2       WHERE r1.sid=s.sid AND r2.sid=s.sid AND r1.bid <> r2.bid	LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters or and but didn't B ave not been in' • * * * * * * * * * * * * * * * * * * *	SELECT DISTINCT c.profname       connittee / Transact ( optimite / provide a fROM Committee c         WHERE NOT EXISTS (SELECT commname       FROM Commitee a         WHERE profname = 'Piper')       EXCEPT         (SELECT commname       FROM committee a         WHERE not failed / Transact ( optimite commande)       WHERE profname = 'Piper')         EXCEPT       (SELECT commname         FROM committee a       WHERE a profname: profname)         Y       SELECT E.id       Visitor(visitorID, firstName, lastName, dateofBirth)         FROM Employee E       Visits(passID, enclosureID)         EXCEPT       Enclosure(id, latitude, longitude)         (SELECT E.id       Visits(passID, enclosureID)         EXCEPT       Enclosure(id, latitude, longitude)         (SELECT E.Z.id       FROM Visitor v, Visits vs., Enclosure EZ         WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id       AND v.DateOfBirth < Date('2001-01-01')
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all CREATE VIEW Temp(major, average) AS SELECT 5.major, AVG(s.age) AS average FROM Temp FROM Student S GROUP BY major GROUP BY major GROUP BY major, getting average age SelECT * SELECT * SELECT * FROM student s, enrolled e FROM Student s, enrolled e WHERE s.snum = e.snum FROM Student s, enrolled e FIND the names of all movie stars who've been in a movie SELECT DISTINCT PLAGENT duplicates! Find the departments that have more then one faculty member SELECT DISTINCT F1.deptid seeing if different faculties have same FROM Faculty f1, Faculty f2 department. WHERE f1.fid <> f2.fid AND f1.deptid = f2.deptid Find IDs of MovieStars who've been in a movie in 1944 or 1974 SELECT Starld EPDM Mayia M Starel s. For 1944 and 1974 use INTERSECT	Indices       Thick = Thick         Indices       Indices         Indinde       Indices	LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters • % is 0 or more arbitrar • % * % * % * * * * * * * * * * * * * *	SELECT DISTINCT c.profname       townittee ( dpinone - loger ( contitue))         FROM Committee c       WHERE NOT EXISTS (SELECT commane FROM Committee a WHERE profname = 'Piper') EXCEPT (SELECT commane FROM committee a WHERE a.profname=c.orofname)         Find all enclosures that were never visited by the visitors born before 2000-01-01         Y         SELECT E.id         Visitor(VisitorID, firstName, lastName, dateofBirth)         FROM Employee E         Visits(passID, enclosureID)         EXCEPT         EXCEPT         ENCOSUME (J, latitude, longitude)         (SELECT E.id         Visitor(visitorID, firstName, lastName, dateofBirth)         FROM Visitor v, Visits vs, Enclosure EZ         WHERE vvisitorID = vs.passID AND vs.enclosureID = ez.id         AND v.DateOfBirth < Date('2001-01-01')
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS(SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all <b>CREATE VIEW Temp(major, average) AS</b> SELECT S.major, AVG(s.age) AS average FROM <b>Temp</b> FROM Student S GROUP BY major, getting average age selecting minimum over all Get all students using natural join SELECT * SELECT * FROM student s, enrolled e FROM Student S, enrolled e FROM Student s, enrolled e FROM Student s, enrolled e FROM Student s, moviestars who've been in a movie SELECT <u>ISITINCT</u> Name Need distinct or else can FROM Starsin s, Moviestar m get movie star name WHERE s.starsid = m.starid duplicates! Find the departments that have more then one faculty member SELECT DISTINCT f1.deptid seeing if different faculties have same FROM Faculty f1, Faculty f2 department. WHERE f1.fid <> f2.fid AND f1.deptid = f2.deptid Find IDs of MovieStars who've been in a movie in 1944 or 1974 SELECT Starld FROM Movie M, StarsIn S For 1944 and 1974 use INTERSECT WHERE M.MovielD = S.MovieID AND Year = 1944	Mixeswin         OT         TNL         S         TNL         TNL </td <td>LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters • % is 0 or more arbitrar • % is 0 or more arbitrar • % * % * % * % * % * % * % * % * % * %</td> <td>SELECT DISTINCT c.profname       Committee (         FROM Committee c       WHERE NOT EXISTS (SELECT commname         FROM Committee a       WHERE profname = 'Piper')         EXCEPT       (SELECT commname         FROM committee a       WHERE a profname = corofname)         Find all enclosures that were never visited by the visitors born before 2000-01-01         Y       SELECT E.id         Visitor(visitorID, firstName, lastName, dateofBirth)         FROM Employee E       Visitor(sassID, enclosureID)         EXCEPT       Enclosure(d, latitude, longitude)         (SELECT E.id       Visitor(2001-01-01)         Nature       FROM Visitor v, Visits vs, Enclosure EZ         WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id         AND v.DateOfBirth &lt; Date('2001-01-01')</td> Nature       Find all the animals of species cat that are taken care of by 'Sam Smith'         SELECT employeeID       FROM TakesCareOf NATURAL INNER JOIN Employee         FROM TakesCareOf NATURAL INNER JOIN Employee       WHERE naimalID IN (SELECT id from Animal WHERE Species='Cat') AND firstName = 'Same' AND lastName = 'Smith'         Find the names of sailors who have reserved at least 2 boats       SELECT s.sname         FROM Sailors s, Reserves r1, Reserves r2       WHERE r1.sid=s.sid AND r2.sid=s.sid AND r1.bid <> r2.bid         Find the sailor id of the sailors with the highest ratin	LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters • % is 0 or more arbitrar • % is 0 or more arbitrar • % * % * % * % * % * % * % * % * % * %	SELECT DISTINCT c.profname       Committee (         FROM Committee c       WHERE NOT EXISTS (SELECT commname         FROM Committee a       WHERE profname = 'Piper')         EXCEPT       (SELECT commname         FROM committee a       WHERE a profname = corofname)         Find all enclosures that were never visited by the visitors born before 2000-01-01         Y       SELECT E.id         Visitor(visitorID, firstName, lastName, dateofBirth)         FROM Employee E       Visitor(sassID, enclosureID)         EXCEPT       Enclosure(d, latitude, longitude)         (SELECT E.id       Visitor(2001-01-01)         Nature       FROM Visitor v, Visits vs, Enclosure EZ         WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id         AND v.DateOfBirth < Date('2001-01-01')
SELECT sname FROM student s WHERE NOT EXISTS (SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all <b>CREATE VIEW Temp(major, average) AS</b> SELECT S.major, AVG(s.age) AS average FROM <b>Temp</b> FROM Student S WHERE average = (SELECT r GROUP BY major, getting average age selecting minimum over all <b>Get all students using natural join</b> SELECT * SELECT * SELECT * SELECT * SELECT * SELECT * SELECT mains , Moviestar m WHERE s.snum = e.snum WHERE s.starsid = m.starid duplicates! Find the departments that have more then one faculty member SELECT DISTINCT f1.deptid seeing if different faculties have same FROM Faculty f1, Faculty f2 department. WHERE f1.fid <> 2.fid AND f1.deptid = f2.deptid Find Ds of MovieStars who've been in a movie in 1944 or 1974 SELECT Starid FROM Movie M, Starsin S For 1944 and 1974 use INTERSECT WHERE M.MovielD = S.MovieID AND Year = 1944 UNION ALL keeps duplicates SELECT Starid Publicates SELECT Starid Publicates Publicates SELECT Starid Publicates Publicates Publicates Publicates Publicates Publica	Indication       Ort - Mile + Michaever Helic + Michaever Heli	LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters or and but didn't B ave not been in' • * * * * * * * * * * * * * * * * * * *	SELECT DISTINCT c.profname       Committee ( diplicance - inper ( comittee)         WHERE NOT EXISTS (SELECT commname       FROM Committee a         WHERE NOT EXISTS (SELECT commname       FROM Committee a         WHERE profname = 'Piper')       EXCEPT         (SELECT commname       FROM committee a         WHERE a profname=c.profname;       WHERE a profname=c.profname;         Y       SELECT E.id       Visitor(visitorID, firstName, lastName, dateofBirth)         FROM Employee E       Visits(passID, enclosureID)         EXCEPT       Enclosure(id, latitude, longitude)         (SELECT E.id       Fandouse (context), enclosureID = ez.id         AND v.DateofBirth < Date('2001-01-01')
SELECT sname FROM student s WHERE NOT EXISTS (SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all <b>CREATE VIEW Temp(major, average)</b> AS SELECT S.major, AVG(s.age) AS average FROM Temp FROM Student S GROUP BY major, getting average age selecting minimum over all Get all students using natural join SELECT * SELECT * SELECT * FROM student s, enrolled e WHERE s.snum = e.snum Enrolled E Find the names of all movie stars who've been in a moviel SELECT DISTINCT Name Need distinct or else can FROM Starsin s, Moviestar m get movie star name WHERE s.starsid = m.starid duplicates! Find the departments that have more then one faculty member SELECT DISTINCT f1.deptid SELECT Starid FROM Movie M, Starsin S For 1944 and 1974 use INTERSECT WHERE f1.fid <> f2.fid AND f1.deptid = f2.deptid Find IDs of MovieStars who've been in a movie in 1944 or 1974 SELECT Starid FROM Movie M, Starsin S For 1944 and 1974 use INTERSECT WHERE M.MovielD = S.movielD AND Year = 1944 UNION ALL keeps duplicates SELECT Starid FROM Movie M, Starsin S Vear=1944 OR year=1974 SELECT Starid FROM Moviel M, Starsin S Select Starid Select Starid FROM Moviel M, Starsin S Select Starid Select Starid S	Mikeswin         Or         File         File           Halle         Halle         Halle         Halle           Halle         Hall	LIKE is used for string matching •stands for any one character • % is 0 or more arbitrar characters or and but didn't B ave not been in' • * * * * * * * * * * * * * * * * * * *	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commname FROM Committee a WHERE profname = 'Piper') EXCEPT (SELECT commname FROM committee a WHERE a profname = c.profname) Find all enclosures that were never visited by the visitors born before 2000-01-01 SELECT E.id Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E Visitor(sistorID, firstName, lastName, dateofBirth) FROM Employee E Visitor(sistorID, firstName, lastName, dateofBirth) FROM Visitor v, Visits vs, Enclosure(10) EXCEPT Enclosure(id, latitude, longitude) (SELECT E2.id FROM Visitor v, Visits vs, Enclosure EZ WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id AND v.DateOfBirth < Date('2001-01-'01') SELECT employeeID SELECT employeeID SELECT employeeID FROM TakesCareOf NATURAL INNER JOIN Employee WHERE a inmallD IN (SELECT id from Animal WHERE Species='Cat') AND firstName = 'Same' AND lastName = 'Smith' Find the names of sailors who have reserved at least 2 boats SELECT s.sname FROM Sailors s, Reserves r1, Reserves r2 WHERE firstName = 'Same' AND lastName = 'Smith' Find the sailor id of the sailors with the highest rating SELECT s.sid FROM sailors s WHERE firstName > (SELECT MAX(s2.rating) FROM Sailors s2) Find names of sailors who have reserved all boats whose name starts w/ "typhoon' SELECT s.sname FROM Sailors s "no boat w/ typhoon that we have not reserved"
SELECT sname FROM student s WHERE NOT EXISTS (SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all CREATE VIEW Temp(major, average) AS SELECT S.major, AVG(s.age) AS average FROM Temp FROM Student S WHERE average = (SELECT r GROUP BY major, getting average age selecting minimum over all Get all students using natural join SELECT * SELECT * SELECT * SELECT * SELECT * SELECT * SELECT * SELECT Name = e.snum Enrolled E Find the names of all movie stars who've been in a movie SELECT DISTINCT Name WHERE s.starsid = m.starid duplicates! Find the departments that have more then one faculty member SELECT DISTINCT f1.deptid seeing if different faculties have same FROM Starsin S, Moviestars who've been in a movie in 1944 or 1974 SELECT Starid FROM Movie M, Starsin S Elect Starid FROM Movie M, Starsin S SELECT Starid FROM Movie M, St	Mikeswin         Or         File         File           Halle         Halle         Halle         Halle           Halle         Hall	LIKE is used for string matching stands for any one character - % is 0 or more arbitrar characters - % is 0 or more arbitrar - % is 0 or mor	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commname FROM Committee a WHERE profname = 'Piper') EXCEPT (SELECT commname FROM committee a WHERE a profname=c profname) Find all enclosures that were never visited by the visitors born before 2000-01-03 SELECT E.id Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E Visits(passID, enclosureID) EXCEPT Enclosure(d, latitude, longitude) (SELECT E.id FROM Visitor V, visits vs, Enclosure EZ WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id AND V.bateOfBirth < Date('2001-01-01') New WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id AND v.bateOfBirth < Date('2001-01-01') SELECT EZ.id FROM TakesCareOf NATURAL INNER JOIN Employee FROM TakesCareOf NATURAL INNER JOIN Employee FROM TakesCareOf NATURAL INNER JOIN Employee FROM Sailors s, Reserves r1, Reserves r2 WHERE r1.sid=s.sid AND r2.sid=s.sid AND r1.bid <> r2.bid Find the names of sailors who have reserved at least 2 boats SELECT s.sname FROM Sailors s WHERE stating >> (SELECT MAX(s2.rating) FROM Sailors s2) WHERE stating >> (SELECT b.bid FROM Boats b WHERE name LIKE "%typhoon%' EXCEPT (SELECT r.bid FROM Reserves r WHERE r.sid = s.sid)) Find the name & age of the oldest sailor
SELECT sname FROM student s WHERE NOT EXISTS(SELECT c.name FROM class c WHERE NOT EXISTS(SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all CREATE VIEW Temp(major, average) AS SELECT S.major, AVG(s.age) AS SELECT S.major, AVG(s.age) AS SELECT S.major, AVG(s.age) AS WHERE average = (SELECT r GROUP BY major grouping by major, getting average age selecting minimum over all Get all students using natural join SELECT * SELECT * SELECT * SELECT * SELECT * SELECT * SELECT S.maine, AVG(s.age) AS SELECT * FROM student s, enrolled = FROM Student S NATURAL LEFT OUTER WHERE s.snum = e.snum FROM Starsin s, Moviestar m get movie star name WHERE s.starsid = m.starid duplicates! Find the departments that have more then one faculty member SELECT DISTINCT R.aeuty f2 department. WHERE f1.fid <> f2.fid AND f1.deptid = f2.deptid Find IDs of MovieStars who've been in a movie in 1944 or 1974 SELECT Starld SELECT Starld Movie M, Starsin S FOT 1944 and 1974 use INTERSECT WHERE M.MovieID = S.MovieID AND Year = 1944 UNION ALL keeps duplicates FIROM Movie M, Starsin S SUECT Starld MUERE M.MovieID = s.movieID AND Year = 1974 SELECT Starld FIROM Movie M, Starsin S SUECT Starld FIROM Movie M, Starsin S SUECT Starld FIROM MovieID = s.movieID AND Year = 1974 SELECT Starld FIROM MovieID = s.movieID AND Year = 1974 SELECT Starld FIROM MovieID = s.movieID AND Year = 1974 SELECT Starld FIROM Student S2 SUECT Starld FIROM Stud	Mikeword         Orthold is         Thick is         Thick is           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Hand Count         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Micha	LIKE is used for string matching • _stands for any one character • % is 0 or more arbitrar characters or and but didn't B ave not been in' • * * * * * * * * * * * * * * * * * * *	SELECT DISTINCT c.profname       connittee ( definition of the second of t
SELECT sname FROM student s WHERE NOT EXISTS (SELECT c.name FROM class c WHERE NOT EXISTS (SELECT e.snum FROM enrolled e WHERE c.name = e.cname and e.snum = s.snum)) Find those majors for which their average age is the minimum over all <b>CREATE VIEW Temp(major, average)</b> AS SELECT S.major, AVG(s.age) AS average FROM Temp FROM Student S GROUP BY major, getting average age selecting minimum over all <b>Get all students using natural join</b> SELECT * SELECT * SELECT * SELECT * SELECT * SELECT bissing average age SELECT bissing average age SELECT bissing average age SELECT * FROM student s, enrolled e FROM student s, enrolled e FROM student s, enrolled e FROM student s, enrolled e SELECT DISTINCT Name WHERE s.starsid = m.starid duplicates! Find the departments that have more then one faculty member SELECT DISTINCT f1.deptid seeing if different faculties have same FROM Average the stars who've been in a movie in 1944 or 1974 SELECT Starid FROM MovieM, Starsin S SELECT starid FROM MovieM, Starsin S SELECT Starid FROM MovieM, Starsin S SELECT Starid FROM MovieM, Starsin S SELECT Starid equivalent to using FROM MovieM, Starsin S SELECT starid equivalent to using FROM MovieM, Starsin S SELECT starid equivalent to using FROM MovieM, Starsin S SELECT starid from Have approximation and the name & Age of the oldest student(s) SELECT stard from sever 1974 WHERE M.MovielD = s.movielD AND Year = 1944 WHERE M.MovielD = s.movielD AND Year = 1974 WHERE M.MovielD = s.movielD AND Year = 1944 WHERE M.MovielD = s.movielD AND Year = 1974 WHERE M	Mikeword         Orthold is         Thick is         Thick is           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Hand Count         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Michaeword         Halke - Michaeword           Halke - Michaeword         Halke - Micha	LIKE is used for string matching •stands for any one character • % is 0 or more arbitrar characters or and but didn't B ave not been in' • ************************************	SELECT DISTINCT c.profname FROM Committee c WHERE NOT EXISTS (SELECT commname FROM Committee a WHERE profname = 'Piper') EXCEPT (SELECT commname FROM committee a WHERE a profname=c profname) Find all enclosures that were never visited by the visitors born before 2000-01-03 SELECT E.id Visitor(visitorID, firstName, lastName, dateofBirth) FROM Employee E Visits(passID, enclosureID) EXCEPT Enclosure(d, latitude, longitude) (SELECT E.id FROM Visitor V, visits vs, Enclosure EZ WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id AND V.bateOfBirth < Date('2001-01-01') New WHERE v.visitorID = vs.passID AND vs.enclosureID = ez.id AND v.bateOfBirth < Date('2001-01-01') SELECT EZ.id FROM TakesCareOf NATURAL INNER JOIN Employee FROM TakesCareOf NATURAL INNER JOIN Employee FROM TakesCareOf NATURAL INNER JOIN Employee FROM Sailors s, Reserves r1, Reserves r2 WHERE r1.sid=s.sid AND r2.sid=s.sid AND r1.bid <> r2.bid Find the names of sailors who have reserved at least 2 boats SELECT s.sname FROM Sailors s WHERE stating >> (SELECT MAX(s2.rating) FROM Sailors s2) WHERE stating >> (SELECT b.bid FROM Boats b WHERE name LIKE "%typhoon%' EXCEPT (SELECT r.bid FROM Reserves r WHERE r.sid = s.sid)) Find the name & age of the oldest sailor

Conceptual Design	Normalization Removing redundancy from data.	Decomposing (a) π JobNum, DriverENum (Job) / π DriverENum (σ JobNum = 301 (Job) )
<ul> <li>Entity Set: Collection of entities (ex. All cats)</li> <li>All entities in entity set have same set of attributes</li> </ul>	First Normal Form (1NF)	Decomposing into BCNF
All entity sets have <u>keys</u>	Each attribute in a tuple has only one value (can't be an array).	Relation: R(ABCD) FD: B→C, D→A where not exists( (select DriverENum from Job
Domain: value type (ex. float, date, int)	E.g., for "postal code" you can't have both V6T 1Z4 and V6S 1W6	Keys? (select DriverENum from Job A <sup>+</sup> = {A} where JobNum = 301)
Key: minimal set of attributes which can identify an entity in entity set.	Codd's original version allowed multi-valued attributes.	$B^+ = \{B,C\}$ except
Primary Key: main key to identify entity in entity set. has to be as <u>minimal</u> as possible. (ex. cat_id). MUST BE UNIQUE, and NON-NULL.	1 INCORRECT. V6T 1Z4, V6S 1W6	$C^+ = \{C\}$ (select DriverENum from Job
Candidate key: One more kore keys in a relation	Client ID Postal Code	D+ = {A,D} where Job.JobNum = A.JobNum)
Super key: A key + zero or more additional attributes. (Ex. {sid, name}, or {CWL, major},	1 In 1NF. V6T 124 1 V6S 1W6	$BD^+ = \{B,D,C,A\}$
or {name, major, age})	Second Normal Form (2NF)	BD is the only key $X \rightarrow b$
Minimal Key: Smallest set of keys to identify entity in entity set Cardinality ratio: the number of relationships in the set that an entity can participate	No partial key dependencies.	Look at FD $B \rightarrow C$ . Is B a superkey? No. Decompose $AD(B)C$
in. These are called cardinality constraints:	A relation is in 2NF if it is in 1NF and for every FD, X -> Y where X	No. Decompose AD B C (R1(B,C),R2(A,B,D)
······································	is a (minimal) key and Y is a non-key attribute, then no proper	Look at ED D A Lis D a suparkay for P22
	subset of X determines Y.	No. Decompose
And a state of the	e.g., the address relation is not in 2NF:	(B (D) A)
	<ul> <li>House#, street, postal_code is a (minimal) key</li> </ul>	Final answer: R1(B,C), R3(D,A), R4(D,B)
Donnars Lead Market Market Mar	X Y	Consider the relation $R = (A, B, C, D, E, F)$ with the following functional dependencies
CON CALINA CONTRACT OF THE STATE	- Heurett street sector and > Desidence	• (FD1) $C, D \rightarrow A$
Planticipi availand 2 in Anthone in Anthone	<ul> <li>House#, street, postal_code → Province</li> <li>Subset of X Y</li> </ul>	• (FD2) $C, E \rightarrow B$
LANS CONTRACTOR		• (FD3) $D, F \rightarrow E$
Anno Constants of	Postal_code → Province	• (FD4) $B \rightarrow D$ • (FD5) $C \rightarrow F$
A Contraction of the contraction	Not in 2NF	Candidate keys is where you can get every letter.
NOT THE REPORT OF THE REPORT O	Boyce-Codd Normal Form (BCNF)	STEP 1: Find all candidate keys.
	For all non-trivial functional dependencies X->b, X must be a superkey for a relation to be in BCNF.	L M K CHICK AND K
	Ex. Whenever a set of attributes R determine another attribute,	D COT C CDARME B3
A Control of the second	it should determine all the attributes of R.	E CB SCPABEE3 V SC,D3
Man And And And And And And And And And An	Check if all the left parts of the FD are a superkey. If they are,	E CF=5 CF 3 {C, B 3
	then it is in BCNF, and if not, they need to be decomposed.	STEP 2: Identify all the FDs that violate the BCNF condition.
Filong	Third Normal Form (3NF)	$D_{F} \rightarrow E$ BCNF is that for X -> B, it has
Key Constraints	A Relation R is in 3NF if: If X -> b is a non-trivial dependency in R, then X is a superkey for	to be a supreduce. Cinese
Key constraints are shown with arrow in ER diagram: (one)	R, or B is part of a (minimal) key.	(C E) {C D) {C B} are keys
one-to-one: entity in A is associated <u>at most once</u> with one entity in B	Note: b must be part of a key not part of a superkey (if a key	C -7 F FD1 and FD2 are ruled out,
(ex. A: Student, B: student ID #)	exists, all attributes are part of a superkey)	violates BCNF meaning only 3 options left.
The primary key of R (relation) is Primary Key of A <u>OR</u> Primary Key of B	Example:	STEP 3: Decompose R so that everything is in BCNF. Choose FDs in ascending
one-to-many: entity in A is associated <u>with any number</u> (ex. A: Mom, B: children)	Each CK has 3 attributes but each FD has 2, so it's not in BCNF.	R(AB, C, D, E, F)
One to many (A is one, B is many) is primary key of B (one) (many)	3NF? AB → C AB is not a CK but C is part of a CK; so, 3NF is good so far.	ABLIDE E BENT COES. TOT
many-to-one: opposite of one-to-many	$CD \rightarrow E CD$ is not a CK, but E is part of a CK; therefore, 3NF, so far	RI(A,B,C,D,E) RZ(B,E,E) E have to preserve functional dependences
One to many (B is one, A is many) is primary key of A	$DE \rightarrow B DE$ is not a CK, but B is part of a CK	
many-to-many: entities in A can be associated with m Employees Works In Departments	Conclusion: 3NF	$R^{3}(A, C, F, D) R^{4}(B, D)$ (CF(B)D)
Many to many is Primary Key of <u>AND</u> Primary Key of (many) (many)	Rule: for all non-trivial functional dependencies in a relation R of the form X->b, it must be the case that X is	FD1 and
Overlapping Constraints Total + Overlap Total + Disjoint	a superkey of R or <b>b is part of a key.</b>	R5(ADC) R6(C,C)
Overlap constraints: specializations can be	Entity Relationship Diagrams	key. make sure to
Disjoint: a superclass entity belongs to no more     Musician Actor     then a single sublass	Name Symbol	Finel. underline middle item.
Has to EITHER be musician or actor, cannot be     Actor & Musician	Entity Entity	
both Partial + Overlap Partial + Disjoint		$R(A,\underline{D},\underline{c}), R(\underline{c},f), R(\underline{B},\underline{D}), R(\underline{D},\underline{f},\underline{c})$
Overlapping: subclasses may overlap     MoviePerson     MoviePerson	Attribute Attribute	Decompose into minimal cover
Can be BOTH musician or actor Covering Constraints: Specializations can be	Relationship Relationship	Given the relation A(P, Q, R, S, T, U, V, W) and set of FDS:
Total: A superclass entity must belong to some	Generalization/Specialization	$P, Q \rightarrow T$ Cross or cartesian product is denoted by X * Y and returns a
subclass Actor & Musician	IsA	$U \rightarrow V, W$ relation on tuples, whose schema contains all fields of X $R \rightarrow S$ followed by all fields of Y.
<u>Movieperson be musician or actor</u>	Weak Entity	$R \rightarrow S$ followed by all fields of Y. $O \rightarrow U, V$ Natural join is denoted by 'a'. It is applicable only when there is
<ul> <li>Partial: some superclass entity may not be in any subclass</li> </ul>	Aggregation	$V, U \rightarrow W, V$ at least one attribute common to R and S.
<ul> <li>Some movie people don't have to be a part of a</li> </ul>	Participation Constraints	STEP 1: Put FDs into standard form.
subclass		Standard form: split output into individual FDs.
Participation Constraints Whether a relationship has to be populated or not.	ISA relationships If we declare A tion	$P, Q \rightarrow T$ $Q \rightarrow U$ $V, U \rightarrow A, B$ gets turned into $V, U \rightarrow A$ and $V, U \rightarrow B$ .
Important for updates.	ISA B, every A entity is a B entity	P, Q -> V P, Q -> T gets turned into P -> T, Q -> T.
Employees>[Manages] <departments< td=""><td>Reasons for using ISA:</td><td>V-&gt;V VV</td></departments<>	Reasons for using ISA:	V->V VV
Ex When deleting a new department must still have a manager		
Ex. When deleting, a new department must still have a manager	Add descriptive attributes     MoviePeople	V, U-JW
Ex. When deleting, a new department must still have a manager Aggregation Having a relationship between relationships is forbidden.		U->W VU->W
Aggregation	Add descriptive attributes specific to subclass	U-> W R -> S V, U -> V
Aggregation Having a relationship between relationships is forbidden.	Add descriptive attributes specific to subclass     Restrict entities that participate in relationship	$\begin{array}{c} V, V \rightarrow W \\ V, V \rightarrow W \\ R \rightarrow S \\ \end{array}$ STEP 2: Minimize the LHS (left-hand-side) of FDs.
Aggregation Having a relationship between relationships is forbidden.	Add descriptive attributes specific to subclass     Restrict entities that	$V \rightarrow W$ $R \rightarrow S$ STEP 2: Minimize the LHS (left-hand-side) of FDs. Since $V \rightarrow V$ , $V \rightarrow W$ con become $V \rightarrow W$
Aggregation Having a relationship between relationships is forbidden.	Add descriptive attributes specific to subclass     Restrict entities that participate in relationship Key constraints Students the Vaccination Totel rearticination     det # title	$\begin{array}{c} V, V \rightarrow W \\ V, V \rightarrow W \\ R \rightarrow S \\ \end{array}$ STEP 2: Minimize the LHS (left-hand-side) of FDs.
Aggregation Having a relationship between relationships is forbidden.	Add descriptive attributes specific to subclass     Restrict entities that participate in relationship Key constraints Students     Vaccination     Status     Musiclass	$\begin{array}{cccc} V, V \rightarrow V \\ R \rightarrow S \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ Since U \rightarrow V, V, U \rightarrow W \ con \ become U \rightarrow W \\ Since U \rightarrow W, V, V \rightarrow V \ con \ become U \rightarrow V, \ since \ oltraidy \ exists \ make \ HS \\ \hline P \cap \rightarrow T \\ \hline Q \cap \rightarrow V \\ \hline Q \cap \rightarrow T \\ \hline Q \cap \rightarrow V \\ \hline Q \cap \rightarrow T \\ \hline Q \cap D \\ \hline D \cap D \\ \hline $
Aggregation Having a relationship between relationships is forbidden. C B Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. Closure is that given a set, what Closure is that given a set, what Closu	Add descriptive attributes specific to subclass     Restrict entities that participate in relationship Key constraints Students the Vaccination Totel rearticination     det # title	$\begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ Since U \rightarrow V, V, U \rightarrow W \ con \ become U \rightarrow W \\ since U \rightarrow W, V, U \rightarrow W \ con \ become U \rightarrow V, \ since \ oltraidy \ exists \ make \ results \\ P, Q \rightarrow T \\ Q \rightarrow V \\ \hline Q \rightarrow V \\ \hline Small as possible. the "V" in V, U \rightarrow W \ is useless. \end{array}$
Aggregation Having a relationship between relationships is forbidden. C B Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. Closure is that given a set, what else can it get? Xercise Evaluates The (minimal) key of Evaluates is iid + course# + term.	Add descriptive attributes specific to subclass     Restrict entities that participate in relationship Key constraints Students     Constraints     Students     Constraints     Constrai	$\begin{array}{cccc} V, V \rightarrow V \\ R \rightarrow S \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ Since U \rightarrow V, V, U \rightarrow W \ con \ become U \rightarrow W \\ Since U \rightarrow W, V, V \rightarrow V \ con \ become U \rightarrow V, \ since \ oltraidy \ exists \ make \ HS \\ \hline P \cap \rightarrow T \\ \hline Q \cap \rightarrow V \\ \hline Q \cap \rightarrow T \\ \hline Q \cap \rightarrow V \\ \hline Q \cap \rightarrow T \\ \hline Q \cap D \\ \hline D \cap D \\ \hline $
Aggregation Having a relationship between relationships is forbidden. C B Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. Closure is that given a set, what Closure is that given a set, what Closu	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Student - Top Visconation Tolel participation</li> <li>Weak Entities</li> <li>Gept &amp; Uildo Returns (onc)</li> <li>A weak entity is identified by considering the</li> </ul>	$V, U \rightarrow W$ $R \rightarrow S$ $V, U \rightarrow V$ $STEP 2: Minimize the LHS (left-hand-side) of FDs.$ Since $U \rightarrow V$ , $V, U \rightarrow W$ can become $U \rightarrow W$ Since $U \rightarrow W$ , $V, U \rightarrow W$ can become $U \rightarrow W$ Since $U \rightarrow W$ , $V, U \rightarrow V$ can become $U \rightarrow V$ , since already exists and le move $P, Q \rightarrow T$ $Q \rightarrow U$ Look for areas that can be changed to make LHS where $V \rightarrow V$ $Q \rightarrow V$ Small as possible. the "V" in $V, U \rightarrow W$ is useless.
Aggregation Having a relationship between relationships is forbidden. C B Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. Closure is that given a set, what else can it get? Xercise Evaluates The (minimal) key of Evaluates is iid + course# + term.	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints</li> <li>Students</li> <li>Vaccination</li> <li>Vaccina</li></ul>	$\begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline Since \\ V \rightarrow V \\ \hline V, V \rightarrow V \\ \hline Since \\ V \rightarrow V \\ \hline V, V \rightarrow V \\ \hline Since \\ V \rightarrow V \\ \hline V, V \rightarrow V \\ \hline Since \\ \hline V \rightarrow V \\ \hline V, V \rightarrow V \\ \hline Since \\ \hline V \rightarrow V \\ \hline V, V \rightarrow V \\ \hline Since \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \rightarrow V \\ \hline V \rightarrow V \rightarrow$
Aggregation Having a relationship between relationships is forbidden. Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. tid TAG Employee name else can it get? Xercise Evaluates evaluation The (minimal) key of Evaluates is iid + course# + term.	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints</li> <li>Students</li> <li>Totel cardination</li> <li>Gept</li> <li>Have</li> <li>Sections</li> <li>(many)</li> <li>A weak entity set and weak entity set participate in a one-</li> </ul>	$\begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline Since U \rightarrow V \\ \hline Since U \rightarrow V \\ \hline V, V \rightarrow W \\ \hline con become U \rightarrow W \\ \hline Since U \rightarrow V \\ V, V \rightarrow V \\ con become U \rightarrow V \\ \hline Since U \rightarrow V \\ \hline V, V \rightarrow V \\ con become U \rightarrow V \\ \hline Since U \rightarrow V \\ \hline V, V \rightarrow V \\ con become U \rightarrow V \\ \hline Since U \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \rightarrow$
Aggregation Having a relationship between relationships is forbidden. C B Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. td TAG Employee frame else can it get? Xercise Evaluates evaluation The (minimal) key of Evaluates is iid + course# + term. td france frame franslixting Entity Set: Mapping frame	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints</li> <li>Students</li> <li>Actors</li> <li>Musicians</li>     &lt;</ul>	$\begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline \\ STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ \hline \\ Since U \rightarrow V, V, U \rightarrow W \\ con become U \rightarrow W \\ \hline \\ Since U \rightarrow W, V, U \rightarrow V \\ con become U \rightarrow W \\ \hline \\ Since U \rightarrow W, V, U \rightarrow V \\ con become U \rightarrow V \\ \hline \\ \\ Since U \rightarrow V, V, U \rightarrow V \\ con become U \rightarrow W \\ \hline \\ \\ Since U \rightarrow V, V, U \rightarrow V \\ con become U \rightarrow W \\ \hline \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\$
Aggregation Having a relationship between relationships is forbidden. C B Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. tid TAG Employee name else can it get? Xercise evaluates evaluation The (minimal) key of Evaluates is iid + course# + term. tid name course# term Instructors Teaches courses Translating Entity Set Mapped of the attraction of the attr	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints</li> <li>Students</li> <li>GuernHame</li> <li>Weak Entities</li> <li>(onc)</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>Owner entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set has total participation.</li> </ul>	$\begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline Since U \rightarrow V, V, U \rightarrow W \\ \hline Since U \rightarrow V, V, U \rightarrow W \\ \hline Since U \rightarrow V, V, U \rightarrow W \\ \hline Since U \rightarrow W, V, U \rightarrow V \\ \hline Since U \rightarrow W, V, U \rightarrow V \\ \hline Since U \rightarrow W, V, U \rightarrow V \\ \hline Since U \rightarrow W, V, U \rightarrow V \\ \hline Since U \rightarrow W, V, U \rightarrow V \\ \hline Since U \rightarrow W, V, U \rightarrow V \\ \hline Since U \rightarrow V, U \rightarrow W \\ \hline Since U \rightarrow V, U \rightarrow V \\ \hline Since U \rightarrow V \\ \hline V \rightarrow V \\ \hline Since U \rightarrow V \\ \hline V \rightarrow V \\ \hline Since V, V \rightarrow V \\ \hline Since Q \rightarrow V \\$
Aggregation Having a relationship between relationships is forbidden. C B Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Xercise Evaluates evaluation The (minimal) key of Evaluates is iid + course# + term. Iid name Course# term Instructors Translations for a take the effect Courses Translations for a take the effect Translations for a take the eff	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints <u>Students</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston Vacconston <u>Vacconston Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston Vacconston <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Vacconston</u> <u>Va</u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></u></li></ul>	$\begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline \\ STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ \hline \\ Since U \rightarrow V, V, U \rightarrow W \\ con become U \rightarrow W \\ \hline \\ Since U \rightarrow W, V, U \rightarrow V \\ con become U \rightarrow W \\ \hline \\ Since U \rightarrow W, V, U \rightarrow V \\ con become U \rightarrow V \\ \hline \\ \\ Since U \rightarrow V, V, U \rightarrow V \\ con become U \rightarrow W \\ \hline \\ \\ Since U \rightarrow V, V, U \rightarrow V \\ con become U \rightarrow W \\ \hline \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\$
Aggregation Having a relationship between relationships is forbidden. C B Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Xercise Evaluates evaluation The (minimal) key of Evaluates is iid + course# + term. iid name Course# term Instructors Teaches Courses Translating Entity Set: Mapped to closure and the attraction of th	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Sudents to visit attributed attribute attr</li></ul>	$\begin{array}{c} V, V \rightarrow W\\ R \rightarrow S \\ \hline V, V \rightarrow V\\ \hline R \rightarrow S \\ \hline V, V \rightarrow V\\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs.\\ \hline Since V \rightarrow V, V, V \rightarrow V\\ \hline con become V \rightarrow W\\ \hline Since V \rightarrow V, V, V \rightarrow V\\ \hline con become V \rightarrow V\\ \hline Since V \rightarrow V, V, V \rightarrow V\\ \hline con become V \rightarrow V\\ \hline Since V \rightarrow V, V \rightarrow V\\ \hline Con become V \rightarrow V\\ \hline Since V, V \rightarrow V\\ \hline Q \rightarrow V\\ \hline V, V \rightarrow V\\ \hline Since V, V \rightarrow V\\ \hline Since Q \rightarrow V \ ond V \rightarrow V, Q \rightarrow V$ is redundent and can be removed
Aggregation Having a relationship between relationships is forbidden. C B Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. closure is that given a set, what tid TAG Employee name else can it get? XerCise Evaluates evaluation The (minimal) key of Evaluates is iid + course# + term. tid name course# term Instructors Teaches courses Translating Entry Set, Mapped, to course and the set of	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Sudents - the Vaccination Vactors - Vaccination Vactors - Vaccination (one)</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>Owner entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> </ul>	$\begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline Since U \rightarrow V, V, U \rightarrow W \\ \hline Since U \rightarrow V, V, U \rightarrow W \\ \hline Since U \rightarrow V, V, U \rightarrow W \\ \hline Since U \rightarrow V, V, U \rightarrow W \\ \hline Since U \rightarrow W, V, V \rightarrow V \\ \hline Since U \rightarrow W, V, V \rightarrow V \\ \hline Since U \rightarrow W, V, V \rightarrow V \\ \hline Since U \rightarrow V, V, U \rightarrow V \\ \hline Since U \rightarrow V, V, U \rightarrow V \\ \hline Since U \rightarrow V, V, U \rightarrow V \\ \hline Since U \rightarrow V, V, U \rightarrow V \\ \hline Since U \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \hline V \rightarrow V \hline V \rightarrow V \rightarrow V \hline V \rightarrow V \rightarrow$
Aggregation Having a relationship between relationships is forbidden. C Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Xercise Evaluates ovaluation The (minimal) key of Evaluates is iid + course# + term. iid name Courses Translating Entity Set: Mapped to c. false the attri- instructors Translating Entity Set: Mapped to c. false the attri- tion of the course of the course of the attri- tion of the course of the attri- tion of the course of the course of the attri- tion of the course of the course of the attri- tion of the course of the course of the attri- tion of the course of the course of the attri- tion of the course of the course of the course of the attri- tion of the course of	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Sudents - Vector attributed subus</li> <li>Weak Entities</li> <li>Gertham</li> <li>Gourses - Have - Sections (many)</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>Owner entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set has total participation.</li> <li>Entits of Redundancy is need to charge &gt; 1 Update Anomaly: Need to charge &gt; 1 Update</li></ul>	$\begin{array}{c} V, V \rightarrow V \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ \hline con become U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ \hline con become U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ \hline con become U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ \hline con become U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ \hline con become U \rightarrow V \\ \hline Since U \rightarrow V, V \rightarrow V \\ \hline con become U \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \hline V \rightarrow V \\ \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \hline V \rightarrow V \hline V \rightarrow V \rightarrow V \hline V \rightarrow V \rightarrow$
Aggregation Having a relationship between relationships is forbidden. C B Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. Closure is that given a set, what else can it get? Courses Course	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Sudents (a constraint)</li> <li>Sudents (a constraint)</li> <li>Weak Entities</li> <li>(one)</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>Owner entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one- to-many identifying relationship set.</li> <li>Weak entity set and weak entity set and</li></ul>	$\begin{array}{c} V, V \rightarrow W\\ R \rightarrow S \\ \hline V, V \rightarrow V\\ \hline R \rightarrow S \\ \hline V, V \rightarrow V \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs.\\ \hline Since U \rightarrow V, V, V \rightarrow V \\ \hline con become U \rightarrow V\\ \hline Since U \rightarrow W, V, V \rightarrow V \\ \hline con become U \rightarrow V, \\ \hline Since U \rightarrow W, V, V \rightarrow V \\ \hline con become U \rightarrow V, \\ \hline Since U \rightarrow W, V, V \rightarrow V \\ \hline con become U \rightarrow V, \\ \hline Since U \rightarrow W, V, V \rightarrow V \\ \hline con become U \rightarrow V, \\ \hline Since U \rightarrow W, \\ \hline V, V \rightarrow V \\ \hline Q \rightarrow V \\ \hline V \rightarrow$
Aggregation Having a relationship between relationships is forbidden. C Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Courses	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Sudents - Constraints - Conste</li></ul>	$V, U \rightarrow W$ $R \rightarrow S$ $V, U \rightarrow V$ $R \rightarrow S$ $V, U \rightarrow V$ $Since U \rightarrow V, V, U \rightarrow W$ $Con become U \rightarrow W$ $Since U \rightarrow W, V, U \rightarrow V$ $Con become U \rightarrow W$ $Since U \rightarrow W, V, U \rightarrow V$ $Con become U \rightarrow V, Since already exists manked to make LHS V \rightarrow V Q \rightarrow V V, U \rightarrow V Q \rightarrow V V, U \rightarrow V V V, U \rightarrow V V \rightarrow V $
Aggregation Having a relationship between relationships is forbidden. C Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Courses	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Suscender of the owner (strong) entities (one) (many)</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>Owner entity set has total participate in a one-to-many identifying relationship set.</li> <li>Weak entity is that and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity is the total participation.</li> </ul>	$V, U \rightarrow W$ $R \rightarrow S$ $V, U \rightarrow V$ $STEP 2: Minimize the LHS (left-hand-side) of FDs.$ Since $U \rightarrow V$ , $V, U \rightarrow W$ can become $U \rightarrow W$ since $U \rightarrow V$ , $V, U \rightarrow W$ can become $U \rightarrow W$ since $U \rightarrow W$ , $V, V \rightarrow V$ can become $U \rightarrow W$ . $P, Q \rightarrow T$ $Q \rightarrow V$ Since $U \rightarrow V$ , $V, V \rightarrow V$ can be changed to make LHS $V \rightarrow V$ $Q \rightarrow V$ $V = V  V = V  V = V  V = V  W = V + V = V = V = V$ $V = V  V = V  V = V  V = V  W = V = V = V = V = V = V = V = V = V =$
Aggregation Having a relationship between relationships is forbidden. C Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Courses	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Suscend - To Vacconston - Vaccons</li></ul>	$V, U \rightarrow W$ $R \rightarrow S$ $V, U \rightarrow V$ STEP 2: Minimize the LHS (left-hand-side) of FDs. Since $U \rightarrow V$ , $V, U \rightarrow W$ can be one $U \rightarrow W$ Since $U \rightarrow W$ , $V, U \rightarrow W$ can be one $U \rightarrow W$ Since $U \rightarrow W$ , $V, U \rightarrow V$ can be care to $U \rightarrow W$ Since $U \rightarrow W$ , $V, U \rightarrow V$ can be care $U \rightarrow W$ , since already exists and $P, Q \rightarrow T$ $Q \rightarrow V$ Look for areas that can be changed to make LHS small as possible. the "V" in $V, U \rightarrow W$ is useless. $U \rightarrow V$ $Q \rightarrow V$ $V, U \rightarrow V$ $V \rightarrow V$ $V, U \rightarrow V$ $V, U \rightarrow V$ $V, U \rightarrow V$ $V, U \rightarrow V$ $V \rightarrow V \rightarrow V$ $V \rightarrow V \rightarrow V$ $V \rightarrow $
Aggregation Having a relationship between relationships is forbidden. C Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Xercise Evaluates evaluation The (minimal) key of Evaluates is iid + course# + term. Id name Courses Immediate and the course # + term. Id name Courses Immediate and the course # + term. Id name Courses Immediate and the course # + term. Immediate and the	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Suscent of the status</li> <li>Totel nerticipation</li> <li>Weak Entities</li> <li>Owner entity is identified by considering the primary key of the owner (strong) entity.</li> <li>Owner entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set has total participation.</li> <li>Earls of Vectoration of the owner (strong) entity.</li> <li>Owner entity set has total participate in a one-to-many identifying relationship set.</li> <li>Weak entity set has total participation.</li> <li>Earls of Vectoration of the owner (strong) entity.</li> <li>Deducte Anomality: Need to change 7 1</li> <li>Update Anomality: Need to change 7 1</li> <li>Update Anomality: Need to change 7 1</li> <li>Update Anomality: Attention of the primary was the participation.</li> <li>Earls of Vectoration of the owner (strong) entity.</li> <li>Other entity set has total participation.</li> <li>Earls of Vectoration of the owner (strong) entity.</li> <li>Opdate Anomality: Need to change 7 1</li> <li>Update Anomality: Attention of the owner owner (strong) entity.</li> <li>Optical context of the presence of other attributes cont whet hase</li> <li>Optical context of the presence of other attributes of delete all additions of the owner (strong owner for a context owner for a con</li></ul>	$V, U \rightarrow W$ $R \rightarrow S$ $V, U \rightarrow V$ STEP 2: Minimize the LHS (left-hand-side) of FDs. Since $U \rightarrow V$ , $V, U \rightarrow W$ can be one $U \rightarrow W$ Since $U \rightarrow V$ , $V, U \rightarrow W$ can be one $U \rightarrow W$ Since $U \rightarrow V$ , $V, U \rightarrow V$ can be called $U \rightarrow W$ Since $U \rightarrow V$ , $V, U \rightarrow V$ can be called $U \rightarrow W$ Since $U \rightarrow V$ , $V, U \rightarrow V$ can be called $U \rightarrow W$ Since $U \rightarrow V$ , $V, U \rightarrow V$ can be called $U \rightarrow W$ $R \rightarrow S$ $U \rightarrow V$ $V, U \rightarrow V$ $V \rightarrow $
Aggregation Having a relationship between relationships is forbidden. C Aggregation allows us to treat relationships set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Course is that given a set, what else can be be course is that given a set, what course is that given a set, what is the course is that given a set, what course is that given a set, what is the course is that given a set, what course is that given a set, what is the course is that given a set, what else can be course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Suscent of the owner (strong) entities (one) (many)</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>Owner entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity is under the owner (strong) entity.</li> <li>Owner entity set has total participation.</li> </ul>	$ \begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow W \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ con become U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ con become U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ Q \rightarrow V \\ V \rightarrow V \\ U \rightarrow V \\ V \rightarrow V $
Aggregation Having a relationship between relationships is forbidden. C Aggregation allows us to treat relationships set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Course is that given a set, what else can be be course is that given a set, what course is that given a set, what is the course is that given a set, what course is that given a set, what is the course is that given a set, what course is that given a set, what is the course is that given a set, what else can be course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given a set, what is the course is that given	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Sudents to visco and the subclass</li> <li>Restrict not subclass<td><math display="block"> \begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow W \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V, \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ Q \rightarrow V \\ V \rightarrow V \\ U \rightarrow V \\ V \rightarrow</math></td></li></ul>	$ \begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow W \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V, \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ Q \rightarrow V \\ V \rightarrow V \\ U \rightarrow V \\ V \rightarrow$
Aggregation Having a relationship between relationships is forbidden. Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Xercise Evaluates Translating Entity Set, Mapped, to c. Take The Insert Inser	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Subcets</li> <li>Guernet in relationship</li> <li>Key constraints Subcets</li> <li>Guernet in relationship</li> <li>Meak Entities</li> <li>Guernet in relationship</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>Owner entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participation.</li> <li>Evils of keduncancy is the storal participation.</li> <li>Evils of keduncancy is the provide the presence of other attributes out the marked without the presence of other attributes.</li> <li>Deletion Anomaly : Attributes are lost bloc of attention of other attributes.</li> <li>Deletion Anomaly : Attributes are lost bloc of attention of other attributes.</li> <li>Deletion Anomaly : Attributes are lost bloc of attention of other attributes.</li> </ul>	$ \begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow W \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V, \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ con become U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ \hline Q \rightarrow V \\ \hline V \rightarrow V \\ V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \hline V \rightarrow V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \hline V \rightarrow V \rightarrow V \\ \hline V \rightarrow V \hline V \rightarrow V \hline V \rightarrow V \rightarrow V \hline V \rightarrow V \rightarrow V$
Aggregation Having a relationship between relationships is forbidden. C Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Xercise Exclusion Toaches Courses Translating Entity Set, Mapped, 10 d. Take The anti- translating Entity Set, Mapped, 10 d. Take The anti- Exc (EEATE TAKE Directed - Mapped, 10 d. Take The anti- Converting Figure The Anti- Converting Figure The Anti- Participate The Anti- Relation for the figure the anti- for Converting Figure The Anti- Anti- for Converting Figure The Anti- Anti- Anti- Anti- Anti- Anti- Converting Figure The Set Nucl. ON PEDICANK (UTD) Anti- Anti- Anti- Anti- Anti- Converting Figure The Set Nucl. ON PEDICANK (UTD) Anti- Ant	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Subcets</li> <li>Guernet in relationship</li> <li>Key constraints Subcets</li> <li>Guernet in relationship</li> <li>Meak Entities</li> <li>Guernet in relationship</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>Owner entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participation.</li> <li>Evils of keduncancy is the storal participation.</li> <li>Evils of keduncancy is the provide the presence of other attributes out the marked without the presence of other attributes.</li> <li>Deletion Anomaly : Attributes are lost bloc of attention of other attributes.</li> <li>Deletion Anomaly : Attributes are lost bloc of attention of other attributes.</li> <li>Deletion Anomaly : Attributes are lost bloc of attention of other attributes.</li> </ul>	$ \begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ con before U \rightarrow W \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con before U \rightarrow W \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con before U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ con before U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ con before U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ Q \rightarrow V \\ V \rightarrow V \\ U \rightarrow V \\ V \rightarrow V \rightarrow$
Aggregation Having a relationship between relationships is forbidden. C Aggregation allows us to treat relationships set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Xercise Evaluates evaluation The (minimal) key of Evaluates is iid + course# + term. Courses forms letting Set: Mapping for a field for the first instructors Translating Entity Set: Mapping for a field for the first forms letting set: Mapping for a field for the first Have been better and the field for the first set of th	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Subcets</li> <li>Guernet in relationship</li> <li>Key constraints Subcets</li> <li>Guernet in relationship</li> <li>Meak Entities</li> <li>Guernet in relationship</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>Owner entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participation.</li> <li>Evils of keduncancy is the storal participation.</li> <li>Evils of keduncancy is the provide the presence of other attributes out the marked without the presence of other attributes.</li> <li>Deletion Anomaly : Attributes are lost bloc of attention of other attributes.</li> <li>Deletion Anomaly : Attributes are lost bloc of attention of other attributes.</li> <li>Deletion Anomaly : Attributes are lost bloc of attention of other attributes.</li> </ul>	$ \begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ construction to the end of the second to the end of the second to the end of t$
Aggregation Having a relationship between relationships is forbidden. C C C C C C C C C C C C C C C C C C C	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Suscent of the value of the value</li></ul>	$ \begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ con before U \rightarrow W \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con before U \rightarrow W \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con before U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ con before U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ con before U \rightarrow V, \\ \hline Since U \rightarrow V, \\ V, V \rightarrow V \\ Q \rightarrow V \\ V \rightarrow V \\ U \rightarrow V \\ V \rightarrow V \rightarrow$
Aggregation Having a relationship between relationships is forbidden. C C C C C C C C C C C C C C C C C C C	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Sudents to visual attributes to the subclass</li> <li>Weak Entities</li> <li>Owner entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set and weak entity set participation.</li> <li>Entits of Kedunation in the intervention in a one-to-many identifying relationship set.</li> <li>Weak entity set has total participation.</li> <li>Entits of kedunation in a one-to-many identifying relationship set.</li> <li>Weak entity set on a store of other attributes can't be included without the presence of other attributes in delete and other attributes in the intervention of the intervention of the interventing in the intervention intervention in the intervention inten</li></ul>	$ \begin{array}{c} V, V \rightarrow W \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ can be converted U \rightarrow W \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ can be converted U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ can be converted U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ can be converted U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ can be converted U \rightarrow V \\ \hline Since U \rightarrow V \\ \hline V, V \rightarrow V \\ \hline V \rightarrow V $
Aggregation Having a relationship between relationships is forbidden.  Aggregation allows us to treat relationships set as entity set, letting us participate in other relationships.  Closure is that given a set, what else can it get?  Xercise  Evaluates  availation  The (minimal) key of Evaluates is iid + course# + term.  and and and and and and and and and an	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Suscent of the owner (strong) entity.</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>A weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set has total participation.</li> </ul>	V, U = 7 WSTEP 2: Minimize the LHS (left-hand-side) of FDs.Since U = V, V, U = W can be come U = WSince U = V, V, U = W can be come U = WSince U = V, V, U = V can be come U = WP, Q = TQ = VCook for areas that can be changed to make LHSP, Q = TQ = VSincle U = VV, U = VV is useless.V = VQ = VW to = VQ = VW is useless.V = VQ = VV, U = VV, U = VW is useless.V = SU = VV, U = VV, U = VMinimized 40U = VSince V, U = VV = VV, U = VMinimized 40V = VSince V, U = VStatements which are proven by otherU = VW = VQ = VStatements can be removed.U = VU = VW = VM = X > YStatements can be removed.Ex Department determines AddressU = W = W = M = M = X > ZM = X > YStatements which are proven by otherU = W = W = M = X > M = X > ZMinimize M = X > ZMinimize M = X > ZReflexivity: If Y < X, then X > YFunctional dependency: one attributedetermines another.Ex Department determines AddressU = W + M = M > X > ZMinimize M = X > ZShow hat (sname, pH) is a superkey ofSuppler Particument (determines address ButDepartment is NOT a key.Sname > citySname, pH = sname, pHa. Show: sname > citySname, pHSname, pHi. sname > citySname, pHZ, aug </td
Aggregation Having a relationship between relationships is forbidden. C Aggregation allows us to treat relationships set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Xercise Evaluates evaluation The (minimal) key of Evaluates is iid + course# + term. Courses Translating Entity Set: mapping for a field for the field fo	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Suscent of the owner (strong) entity.</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>A weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set has total participation.</li> </ul>	V, U = 7 WSTEP 2: Minimize the LHS (left-hand-side) of FDs.Since U = V, V, U = W can be come U = WSince U = V, V, U = W can be come U = WSince U = V, V, U = W can be come U = WSince U = V, V, U = V can be come U = WP, Q = TQ = VLe comoveQ = VLook for areas that can be changed to make LHSSince U = VV, U = VV, U = VV, U = VV TO THE U = VV, U = VV, U = VV = VSince V, U = VV, U = VV, U = VV, U = VSTEP 3: Delete Redundant FDs.Since V, U = V and U = V, Q = VStatements which are proven by otherColspan="2">Statements which are proven by otherU = VStatements can be removed.U = VStatements which are proven by otherU = VStatements which are proven by otherU = VStatements which are proven by otherU = VStatements can be removed.U = VStatements can be removed.U = VStatements which are proven by otherU = VStatements which are proven by otherU = VStatements which are proven by otherU = VStatements can be removed.U = VStatements can be removed.U = VStatements can be removed.U = VSid 3
Aggregation Having a relationship between relationships is forbidden. Aggregation allows us to treat relationship set as entity set, letting us participate in other relationships. closure is that given a set, what else can it get? Xercise Excellent and the participate of the minimal key of Evaluates is iid + course# + term. Insert	<ul> <li>Add descriptive attributes specific to subclass</li> <li>Restrict entities that participate in relationship</li> <li>Key constraints Suscent of the owner (strong) entity.</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>A weak entity is identified by considering the primary key of the owner (strong) entity.</li> <li>A weak entity set and weak entity set participate in a one-to-many identifying relationship set.</li> <li>Weak entity set has total participation.</li> </ul>	$ \begin{array}{c} V, V \rightarrow V \\ R \rightarrow S \\ \hline V, V \rightarrow V \\ \hline STEP 2: Minimize the LHS (left-hand-side) of FDs. \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ con become U \rightarrow V \\ \hline Since U \rightarrow V, V, V \rightarrow V \\ \hline C \rightarrow V \\ R \rightarrow S \\ U \rightarrow V \\ \hline V, U \rightarrow V \\ \hline V \rightarrow V \\$