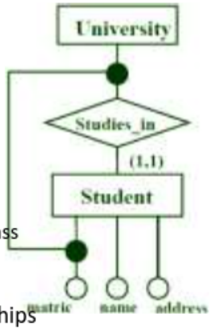
Term	Definition
Key	**Minimal** set of attributes that uniquely identifies the entity
Superkey	A subset of attributes that uniquely identifies its tuples
Candidate Key	Relation could have multiple keys called candidate keys
Primary Key	One of the candidate key chosen
Cardinality	
Degree	

Entity-Relationship Model



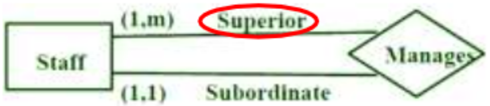
Weak Entities

Relationship must exist and be unique for each entity in the set. Weak entities can only be defined for a participation constrained by (1,1) cardinality

Hierarchies:

- Subclass and Superclass e.g. Person ➔ Student
- Inheritance: subclass inherits attributes of superclass
- Specialization: subclass as its own attributes

Recursive Relationship Sets and Roles in Relationships



The Relational Model

No. of columns = Degree

No. of rows = Cardinality

A subset of attributes relation A is a foreign key if it is the primary key relation B

Options if tuple *t* in Courses is to be deleted:

- Disallow deletion if some rows in Enrolls refers to *t*
- Delete all rows in Enrolls that refer to *t*
- For each row in Enrolls that refer to *t,* replace *cid* value with DEFAULT
- For each row in Enrolls that refer to *t,* replace *cid* value with NULL, provided *cid* is not a primary key attribute in Enrolls

Structured Query Language

DDL – Data Definition Language		
CREATE TABLE	**CREATE TABLE** *name (…)*	Creates table for *name*
DEFAULT	*name* **CHAR(24) DEFAULT** *'some name'*	Default value
PRIMARY KEY	**PRIMARY KEY** (*name, name*) <u>*OR*</u> *name* CHAR(24) **PRIMARY KEY**	
REFERENCES	**FOREIGN KEY** (*attribute*)	Where to get
FOREIGN KEY	**REFERENCES** *Staff(name)*	the attribute
DROP TABLE	**DROP TABLE** *name*	Delete table
ALTER TABLE	**ALTER TABLE** *name* **ADD** *attribute+domain* <u>*OR*</u> **ALTER TABLE** *name* **DROP** *attribute*	Edit columns
UNIQUE	*name* CHAR(24) **UNIQUE**	For Candidate
NOT NULL	*name* CHAR(24) **NOT NULL**	key
CHECK	*age* **NUMERIC CHECK** (*age > < = value)*	Check value
CREATE VIEW	**CREATE VIEW** *NewName* **AS** *(some query you want in NewName)*	

DML – Data Definition Language		
INSERT INTO… VALUES…	**INSERT INTO** *name (relation_attributes)* **VALUES** *('value1', 'value2', …)*	
DELETE FROM	**DELETE FROM** *name [WHERE…]*	
UPDATE… SET…	**UPDATE** *name* **SET** *attribute = value* [WHERE …]	
ORDER BY	*name* **DESC** <u>*OR*</u> *name* **ASC**	
SELECT… AS…	**SELECT** *name* **AS** *newName*	Rename
GROUP BY	Usually models the 'each' noun e.g. *each* director, in the qn and is similar to what you are SELECT-ing	
HAVING	**SELECT** *a* **FROM** *Acts a* **GROUP BY** *a.Actor* **HAVING COUNT** (*) > (**SELECT COUNT** *(m.title)* **FROM** *Movies m* **WHERE** *qualifier*) Col appearing in HAVING clause must either be in GROUP BY clause or be an argument of an aggregation operator	
Arithmetic	**WHERE** *assets * 1.7 < 17* <u>*OR*</u> **SELECT** *(rating + 0.2) * 10*	
Logical Connectors	**WHERE** *qualifier1* **AND** *qualifier 2*	AND, OR, NOT
LIKE	**WHERE** *title* **LIKE** *'W_%S'* Symbol '_' stands for single arbitrary char Symbol '%' stands for 0 or more arbitrary char	
UNION	**SELECT** *a* **FROM** *Acts a* **WHERE** *a.b > 0*	
INTERSECT	**UNION / INTERSECT / EXCEPT**	
EXCEPT	**SELECT** *b* **FROM** *Acts b* **WHERE** *b.a > 0*	
Aggregation operators	**COUNT** ([DISTINCT] *A)* ➔Number of [unique] values in *A* **col** **COUNT** ([DISTINCT] *) ➔ Number of (unique) **rows** **SUM** ([DISTINCT] *A*) ➔ Sum of all (unique) values in *A* **column** **AVG** ([DISTINCT] *A*) ➔ Average of all (unique) values in A **col** SUM, AVG, MIN, MAX often appear in SELECT statement	
Set Comparison in WHERE clause	— v **IN** Q is true iff value v is in the set returned by Q — v **NOT** IN Q is true iff value v is not in the set returned by Q — **EXISTS** Q is true iff the result of Q is non-empty — **NOT EXISTS** Q is true iff the result of Q is empty — **UNIQUE** Q is true iff the result of Q has no duplicates — v op **ANY** Q is true iff there exists some v' in result of Q s.t. v op v' is true — v op **ALL** Q is true iff for each v' in result of Q, v op v' is true — op ∈ { = , <> , < , <= , > , >=}	

Relational Algebra

$\sigma_c(R)$	Select tuples of relation R that satisfy condition c
$\pi_L(R)$	List attributes L of relation R
$\rho(R'(N_1 \to N_1', \dots), R)$	Rename. Can also do with $\rho(R', R)$
$R \cup S \,/\, R \cap S \,/\, R - S$	
$R_1 \times R_2 \,/\, R \otimes_c S$	
R/S	R/S contains all A tuples s.t. for every B tuple in S there is a AB tuple in R

Armstrong Axioms

Reflexivity: if $Y \subseteq X$, then $X \to Y$

Augmentation: if $X \to Y$, then $XZ \to YZ$

Transitivity: if $X \to Y$ and $Y \to Z$, then $X \to Z$

Union: if $X \to Y$ and $X \to Z$, then $X \to YZ$

Decomposition: if $X \to YZ$, then $X \to Y$ and $X \to Z$

Functional Dependency

Define $X \to Y$ $\forall t_1, t_2 \in R$ $t_1.X = t_2.X \Longrightarrow t_1.Y = t_2.Y$	Trivial FD: sid, sname → sid Non-trivial: completely non-trivial: totally do NOT share attributes; non-completely non-trivial: share SOME attributes

Minimal Cover

Given a relation R(A,B,C,D,E,F). The following set F of FDs hold for this table.

$$F = \{AB \to CD, C \to CE, C \to F, F \to E, CDF \to E, DFE \to A\}$$

Step 1: Decompose FDs

$$F = \{AB \to C, AB \to D, \cancel{C \to C}, C \to E, C \to F, F \to E, CDF \to E, DFE \to A\}$$

Step 2: Eliminate redundant attributes from LHS of FDs: CHECK CLOSURE

If we replaced $DFE \to A$ with $DF \to A$, then we get $DF^+ = \{D, F, A, E\}$

$$F = \{AB \to C, AB \to D, C \to F, F \to E, DF \to A\}$$

Step 3: Eliminate redundant FDs

Remove $C \to E$ since $C^* = \{C, F, E\}$. Hence, we are left with:

$$F = \{AB \to C, AB \to D, C \to F, F \to E, CDF \to E, DFE \to A\}$$

$CDF^+ = \{C, D, F, E, A\}$ If we removed $CDF \to E$, $CDF^+ = \{C, D, F, E, A\}$.

So, we can remove this f.d.

$$F = \{AB \to C, AB \to D, C \to F, F \to E, DFE \to A\}$$

Decompositions

Decompositions have to be:

- Lossless: When you ⊗ them back, the original is subset of the result
  To be lossless, attributes common between two relations must functionally determine all attributes in ONE of the two relations
- Dependency preserving: $\{A \to B, B \to C, A \to C\}$ ➔ $\{A \to B, B \to C\}$

Computing FD Projections

For F_{XY}

- Compute $X^+ = X..$, we have $X \to X.. \cap XY$
- Compute $Y^+ = Y..$, we have $Y \to Y.. \cap XY$
- So, $F_{XY} = \{X \to xyz, Y \to xyz\}$
- xyz is the common attribute of $X.. \cap XY$

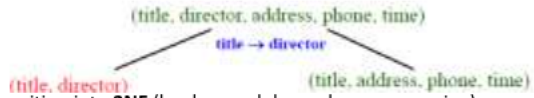
Normal Forms

3NF	BCNF
1. Trivial 2. LHS is a superkey 3. RHS is a prime attribute (appear in at least one key)	1. Trivial 2. LHS is a superkey

Decomposition into **BCNF** (only guarantees lossless)

- Let $X \to A$ be an FD in F that violates BCNF
- Decompose R into
  $R_1 = XA$
  $R_2 = R - A$
- If R_1 or R_2 is not in BCNF, decompose further

E.g.



Decomposition into **3NF** (lossless and dependency preserving)

- Compute minimal cover of R
- Create schema for each FD in minimal cover
- Choose a key and create $i + 1^{th}$ schema
- Remove redundant schema if one is a subset of another

E.g. Minimal cover of $F = \{AC \to E, E \to D, A \to B\}$, key is AC

Schema created: $R_1(A, C, E), R_2(E, D), R_3(A, B), \cancel{R_4(A, C)}$

3NF decomposition is $R_1(A, C, E),\ R_2(E, D),\ R_3(A, B)$

SQL

Find names of suppliers that supplies at least two parts and average cost of it

```
SELECT P.sname, AVG(P.cost)
FROM Part P
GROUP BY P.sname
HAVING COUNT(*) >1
```

Find names of suppliers who supply at least 5 parts with price > 1000

```
SELECT S.name
FROM Part P, Supplier S, PartSupp PS
WHERE P.price > 1000 AND P.partkey = PS.partkey AND PS.suppkey = S.suppkey
GROUP BY S.name
HAVING COUNT ( DISTINCT P.partkey) > 4
```

Find Dept_no where the avg salary of emp in that dept is > the avg emp

```
SELECT E.dept_no
FROM Employee E
GROUP BY E.dept_no
HAVING AVG(E.salary) > ( SELECT AVG ( T1.salary) FROM Employee T1)
```

Find names of required courses for 'CS' curriculum that 'Smith' did not take

```
SELECT C.course_name
FROM Couse C, Required R
WHERE R.curriculum = 'CS' AND R.CID = C.CID
AND C.CID NOT IN (   SELECT T.CID
                     FROM Student S, Take T
                     WHERE S.student_name = 'Smith' AND S.SID = T.SID)
```

Find identifier of all students who never took the course 101 offered by Dept 11

```
SELECT S.SID
FROM Student S
WHERE NOT EXISTS (   SELECT *
                     FROM Transcript T, Section SE
                     WHERE SE.dept_id = 11 AND SE.course_no = 101
                     AND S.SID = T.SID AND T.SEID = SE.SEID)
```

Find course number and dept_id of all course where no student ever got an 'F'

```
SELECT C.course_no, D.dept_id
FROM Course C
WHERE NOT EXISTS (   SELECT *
                     FROM Transcript T, Section S
                     WHERE T.grade = 'F' AND T.SID = S.SID
                     AND C.course_no = S.course_no )
```

Find names of all students who are enrolled two classes at the same timing

```
SELECT DISTINCT S.name
FROM Student S
WHERE S.snum IN (    SELECT E.snum
                     FROM Enrolled E1, Enrolled E2, Class C1, Class C2
                     WHERE E1.enum = E2.enum AND E1.cname <> E2.cname
                     AND E1.cname = C1.cname AND E2.cname = C2.cname
                     AND C1.meets_at = C2.meets_at )
```

TRC / DRC

Find the names of pizzas that come in a 10 inch or a 12 inch size.

TRC: $\{T \mid \exists P \in Pizza\ ((P.size = 10 \vee P.size = 12) \wedge T.name = P.name)\}$

DRC: $\{< N > | \exists C, S(< C, N, S > \in Pizza\ \wedge (S = 10 \vee S = 12)$

Find codes of the most expensive pizzas

$$\{T | \exists P1 \in Pizza \forall P2 \in Pizza(P1.price \geq P2.price \wedge P1.code = P2.code)\}$$

$$\{< C1 > | \exists C1, P1 \forall C2, P2 \left(< C1, P1 > \in Pizza \wedge \left(< C2, P2 > \in Pizza \rightarrow (P1 \geq P2)\right)\right)\}$$

Find sids of Suppliers who supply every red part

$$\{T | \exists C \in Catalog\ \forall P \in Parts(C.pid = P.pid \wedge P.color = red \wedge T.sid = C.sid)\}$$

$$\{< X > | < X, Y, Z > \in Catalog \wedge \forall < A, B, C > \in Parts\left(C = red \vee \exists < P, Q, R > \in Catalog(Q = A \wedge P = X)\right)\}$$

Find sids of Suppliers who supply some red part

$$\{T | \exists C \in Catalog \exists P \in Parts\ (C.pid = P.pid \wedge P.color = red \wedge T.sid = C.sid$$

$$\{< X > | < X, Y, Z > \in Catalog \wedge \exists P, Q, R\ \left(< P, Q, R > \in Parts\ (Y = P \wedge R = red)\right)\}$$

Find the pids of parts supplied by at least two different suppliers

$$\{T | \exists C1 \in Catalog\ \exists C1 \in Catalog(C1.sid <> Cs.sid \wedge C1.pid = C2.pid \wedge C1.pid = T.pid)\}$$

$$\{< Y > | < X, Y, Z > \in Catalog \wedge \exists A, B, C(< A, B, C > \in Catalog \wedge A <> X \wedge Y = B)\}$$

Relational Algebra

Find sids of Suppliers who supply every red part

$$(\pi_{sid,\, pid} Catalog)/(\pi_{pid} \sigma_{color\ =\ red} Parts)$$

Find sids of Suppliers who supply some red part

$$\pi_{sid}(Catalog \otimes_{pid=pid} (\sigma_{color=red} Parts))$$

List names of suppliers who supply at least two parts

$$\rho(T1, Part)$$
$$\rho(T2, Part)$$
$$\pi_{sname}(\sigma_{pno<>pno \wedge sname=sname} (T1 \times T2))$$

List names of suppliers who supply ALL complex parts whose labor cost is > 100

$$\rho(T1, \pi_{pno}(\sigma_{labor>100}(ComplexPart))$$
$$\pi_{sname,pno}\ (Part\ /\ T1)$$

Find the employment numbers of pilots who can fly ALL MD planes

$$\rho\left(B, \pi_{Model_{No}}(\sigma_{Maker=MD}(Plane))\right)$$
$$\rho(A, Can\_Fly)$$
$$\pi_{Emp\_No}(A) - \pi_{Emp\_no}((\pi_{Emp\_no}(A) \times B) - A)$$