Schema.

physical schema: student.excel.

logical schema/conceptual schema: Entities. Students(SID:string, name:string). Courses(CID:string, dept:string). Relationships. A student can enroll in courses.

external schema/views: EnrolledCourses(SID, CID, Coursename)

ER Diagram.

A movie can have at most 1 director.

A director must direct at least 1 movie.

keys: A minimum set of one or more attributes that has a unique value for each record

candidate keys: potential keys. primary key: one of the candidate keys.

strong entity: can exist independently.

ISA: overlap/disjoint: >=2 subclasses contain same entity

total/partial: all entities in subclasses include the super class.

SQL - CREATE TABLE

```
CREATE TABLE Athlete(
    aid INTEGER PRIMARY KEY,
    name CHAR(30) NOT NULL,
    country CHAR(30),
    UNIQUE(name, country));
CREATE TABLE Compete(
    aid INTEGER, oid INTEGER,
    PRIMARY KEY(aid, oid)
    FOREIGN KEY(aid) REFERENCES Athlete,
    FOREIGN KEY(oid) REFERENCES Olympics);
```

ON DELETE CASADE: also delete all referencing tuples (用于删除 weak entity. 被删除 strong entity 的 Foreign key Ref to it).

ON UPDATE SET NULL: sets foreign key of referencing tuple to NULL

DROP TABLE: DROP TABLE Olympics

ALTER TABLE: ALTER TABLE Athlete ADD COLUMN age: INTEGER.

GENERALCONSTRAINT: CHECK(age>=18).

SELECT TABLE

```
SELECT DISTINCT S.sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid
AND R.bid = 103;
```

```
SELECT S.sname AS name1
FROM Sailors S, Reserves R1, Reserves R2,
WHERE S.sid = R1.sid AND S.sid = R2.sid
AND R1.bid <> R2.bid AND R2.day = R1.day
ORDER BY S.age ASC, S.rating DESC.
```

Select sailors who have reserved a red boat OR/AND / but not a green boat:

```
SELECT DISTINCT S.name
FROM Sailors S, Reserve R, Boat B
WHERE S.sid = R.sid AND R.bid = B.bid
AND B.color = 'red'
UNION / INTERSECT / MINUS ... 'green'
```

```
JOIN: SELECT S.name
FROM Sailors S (INNER) JOIN Reserves R on S.sid = R.sid
WHERE R.bid = 103;
```

INNER JOIN / OUTER JOIN:

Sailors

sid	sname
22	Dustin
58	Tamato

Reserves

sid	bid	day
22	101	1/1
22	103	1/2

Select sid, bid

inner: 22, 101

left: 22 101
58 NULL

right: NULL 103
22 101

Full: 22 101
58 NULL
NULL 103

not in Sailor

NESTED Queries

```
SELECT S.sid FROM Sailors S
WHERE S.rating > Any/All (SELECT S2.rating
WHERE S2.name = 'John') AND S.sid IN (
FROM Sailors S2 SELECT R.sid FROM Reserve R WHERE R.bid = 103);
```

AGGREGATE Operators

```
SELECT S.rating, MIN(S.age).
FROM Sailors S WHERE S.age >= 18
GROUP BY S.rating Having COUNT(*) >= 2.
```

Sailors Having exactly 1 boat:

① MINUS

② GROUP BY S.sid HAVING COUNT(*) = 1.

CREATE VIEWS:

```
CREATE VIEW View_Event AS
SELECT
    ue.event_id,
    c.city_name AS event_city
FROM
    User_Events ue
    JOIN Cities c on ue.cid = c.cid
```

Relational Algebra

σ: Selects a subset of rows from relation

π: Selects a subset of columns.

×: combine two relations

−: in rel1 but not in 2.

∪: in rel1 or in rel2

condition, relation

$\sigma_{sport='gymnastics' \wedge country='USA'}(Athlete)$

column, relation

$\pi_{sport, country}(Athlete)$

output: relation

Union ∪, Intersection ∩ set-difference −

兩個必須是 same type

Cross product column 數直接相加. ×

Rename $\rho(AFTER, BEFORE), \rho_{AFTER}(BEFORE)$

conditional join

Join ⋈ $R \bowtie_C S = \sigma_C(R \times S)$.

Equi-join Join conditions consists of only equalities

Natural join Drops duplicates $R \bowtie S$.

$\pi_{sname}((\sigma_{bid=103}(Reserves)) \bowtie Sailors)$

$= \pi_{sname}(\sigma_{bid=103}(Reserves \bowtie Sailors))$

Division: $A/B = \{<x> \mid \forall <y> \in B, <x,y> \in A\}$

Staff Member (Uname, Role, CityID).

City (CityID, Cname)

Fruit (FruitID, Fname)

Enjoys (Uname, FruitID)

$\pi_{x,y}A / \pi_y B = (\pi_x A)$

Question: staff member not in Michigan enjoy all fruits.

Ans: $\rho(EnjoyAllFruits, C\pi_{uname, FruitID}(Enjoys)/(\pi_{FruitID}Fruit))$

$\pi_{FN}(\sigma_{city!='Michigan'} EnjoyAllFruits \bowtie StaffMember \bowtie City)$

Question: Names of sailors who reserved a red and green boat

$\rho(Tempred, \pi_{sid}((\sigma_{color='red'}(Boats)) \bowtie Reserves) - \cdots$

$\pi_{name}((Tempred \cup Tempgreen) \bowtie Sailors)$

NF3

We want the attributes depend on the primary key,

on the whole key and nothing but the key.

first Armstrong Axioms (A.B) -> A. A -> A.

Y ⊆ X, then X -> Y; If X -> Y, XZ -> YZ.

X -> Y and Y -> Z. then X -> Z.

X -> Y, X -> Z then X -> YZ; X -> YZ. then X -> Y, X -> Z

F^*: All FD implied by F. (F 本体)

F^+: All FD that can be generated by F using AA.

Lossless: Join 回去时关系是否会缺失 (BCNF 一定 lossless)

Dependency Preserving: 用两个各有的部分关系, 是否包含 F^+.

3NF 和 BCNF ① 如果 X -> Y. X 是 candidate key:

both 3NF 和 BCNF

② X -> Y, Y 是某一个 candidate key part

3NF

③ 都不是 neither.