CS3402 – Chapter 3
Integrity Constraints

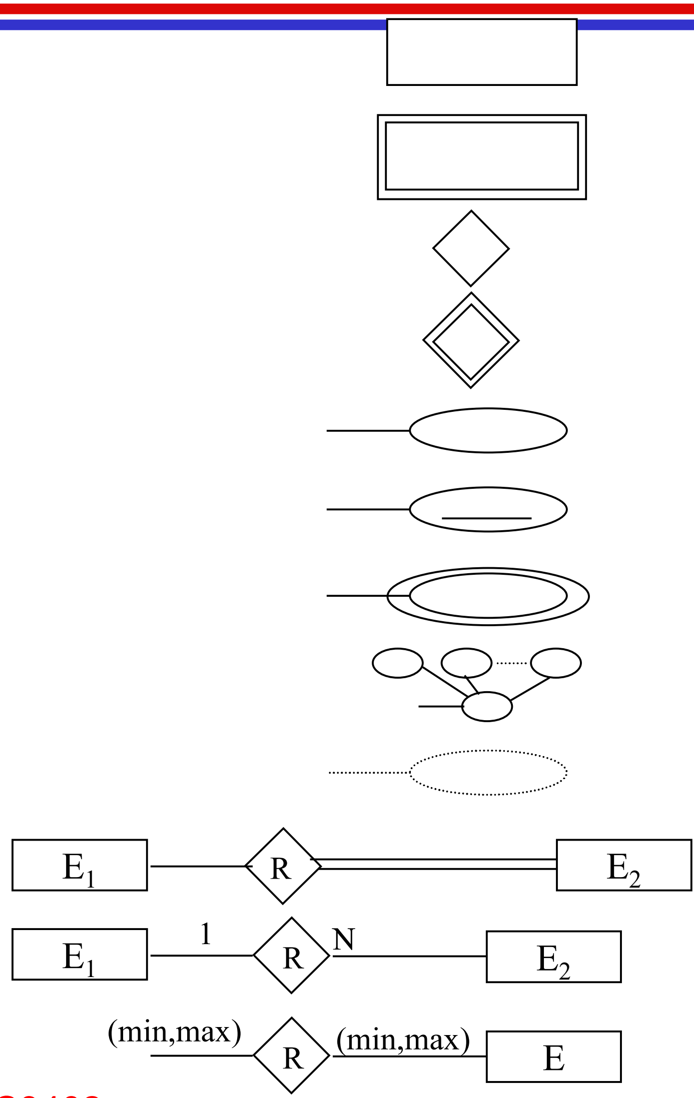
Database Modelling and Implementation

Ideas/requirements $\longrightarrow$ E/R design $\longrightarrow$ Relational schema $\longrightarrow$ Relational database

Summary of ER-Diagram Notation

Symbol

Meaning



ENTITY TYPE

WEAK ENTITY TYPE

RELATIONSHIP TYPE

IDENTIFYING RELATIONSHIP TYPE

ATTRIBUTE

(Primary) KEY ATTRIBUTE

MULTIVALUED ATTRIBUTE

COMPOSITE ATTRIBUTE

DERIVED ATTRIBUTE

TOTAL PARTICIPATION OF E_2 IN R

CARDINALITY RATIO 1:N FOR E_1:E_2 IN R

STRUCTURAL CONSTRAINT (min, max) ON PARTICIPATION OF E IN R

Database Modelling and Implementation

Ideas/requirements $\longrightarrow$ E/R design $\longrightarrow$ Relational schema $\longrightarrow$ Relational database

Relation Name

Attributes

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Tuples

Informal Terms	Formal Terms
Table	Relation
Column Header	Attribute
All possible Column Values	Domain
Row	Tuple
Table Definition	Schema of a Relation
Populated Table	State of the Relation

Summary of Mapping for ER Model Constructs

Table 9.1 Correspondence between ER and Relational Models

ER MODEL	RELATIONAL MODEL
Entity type	*Entity* relation
1:1 or 1:N relationship type	Foreign key (or *relationship* relation)
M:N relationship type	*Relationship* relation and *two* foreign keys
n-ary relationship type	*Relationship* relation and *n* foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

Integrity Constraints

- A relational database schema is a set of relation scheme S = {R1, R2, …, Rn} and a set of integrity constraints IC
- Integrity Constraints determine which values are permissible and which are not in the database (table)
  - Constraints are conditions that must hold on all valid relation states
- Valid state Vs. invalid state
  - Invalid state: A database state that does not obey all the integrity constraints
  - Valid state: a state that satisfies all the constraints in the defined set of integrity constraints

Relational Integrity Constraints

- They are of three main types of constraints:

  - Inherent or Implicit Constraints: These are based on the data model itself. (E.g., relational model does not allow multiple values for any attribute)

  - Schema-based or Explicit Constraints: They are expressed in the schema by using the facilities provided by the model. (E.g., max. cardinality ratio constraint in the ER model)

  - Application-based or Semantic constraints: These are beyond the expressive power of the model and must be specified and enforced by the application programs. (e.g. the salary of an employee should not exceed the salary of the employee's supervisor)

Relational Integrity Constraints

- There are four *main types* of schema-based constraints that can be expressed in the relational model:
  - Domain constraints
  - Key constraints
  - Entity integrity constraints
  - Referential integrity constraints

Domain Constraints

- Domain constraint: Every value in a tuple must be an atomic value from the domain of its attribute (or it could be null, if NULL is allowed for that attribute)

- E.g.,

  C-Name: string of char (30)

  Balance: Number (6,2) …

Key Constraints

- Superkey of R:
  - A set of attributes that can uniquely identify a tuple
  - It is a set of attributes SK, e.g., {A1, A2} of R with the following condition (Key constraint):
    - No two tuples in any valid relation state r(R) will have the same value for SK
    - For any distinct tuples t1 and t2 in r(R), $t1[SK] \neq t2[SK]$

- (Candidate) Key of R:
  - A "minimal" superkey
  - A key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)

Key Constraints

- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, <u>SerialNo</u>, Make, Model, Year)
  - CAR has two keys:
    - Key1 = {State, Reg#}
    - Key2 = {SerialNo}
  - Both are also superkeys of CAR
  - {SerialNo, Make} is a superkey but *not* a key

- In general:
  - Any *key* is a *superkey* (but not vice versa)
  - Any set of attributes that *includes a key* is a *superkey*
  - A *minimal* superkey is a key

Key Constraints

- If a relation has several candidate keys, one is chosen arbitrarily to be the primary key
  - The primary key attributes are underlined
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - We chose SerialNo as the primary key
- The primary key value is used to *uniquely identify* each tuple in a relation
- General rule: Choose as primary key the smallest of the candidate keys (in terms of size)

Keys of Relations

Movies(title, year, length, type, studioName, starName)
Title, year, starName -> length, type, studioName

- Attributes {title, year, starName} form a key for the relation Movie
- Suppose two tuples agrees on these three attributes: title, year, starName
- They must agree on the other attributes, length, type and studioName
- No proper subset of {title, year, starName} functionally determines all other attributes
- {title, year} does not determine starName since many movies have more than one star
- {year, starName} is not a key because we could have a star in two movies in the same year

Entity Integrity Constraints

- Entity Integrity:
  - The *primary key attributes* PK of each relation schema R cannot have null values in any tuple of R
    - Primary key values are used to *identify* the individual tuples
    - $t[PK] \neq$ null for any tuple t in R
    - If PK has several attributes, null is not allowed in any of these attributes
  - Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity

- Key, domain, and entity integrity constraints are specified on individual relations.

- Referential integrity is a constraint involving two relations
  - To specify a relationship among tuples in two relations
  - The referencing relation and the referenced relation (R1 -> R2)

- Tuples in the referencing relation R1 have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation R2 if it satisfies:
  - The attributes in FK have the same domain(s) as the primary key attributes PK of R2
  - A value of FK in a tuple t1 of the current state r1(R1) either occurs as a value of PK for some tuple t2 in the current state r2(R2) or is NULL.

Referential Integrity

- For example, we designate Dno to be a foreign key of EMPLOYEE referencing the DEPARTMENT's Dnumber
- A value of Dno in any tuple t1 of the EMPLOYEE relation must match a value of the primary key Dnumber, in the tuple t2 of the DEPARTMENT relation, or the value of Dno can be NULL if the employee does not belong to a department or will be assigned to a department later.

DEPARTMENT

<u>Dnumber</u>	Dname .....
1	
2	
3	
...	

EMPLOYEE

<u>SSN</u>	...	DNO
123	...	1
456	...	2
789	...	2
129	...	NULL

- Referential integrity constraints typically arise from the relationships among the entities represented by the relation

Referential Integrity

DEPARTMENT

<u>Dnumber</u>	Dname .....
1	
2	
3	
...	

EMPLOYEE

<u>SSN</u>	…	DNO
123	…	1
456	…	2
789	…	2
129	…	NULL

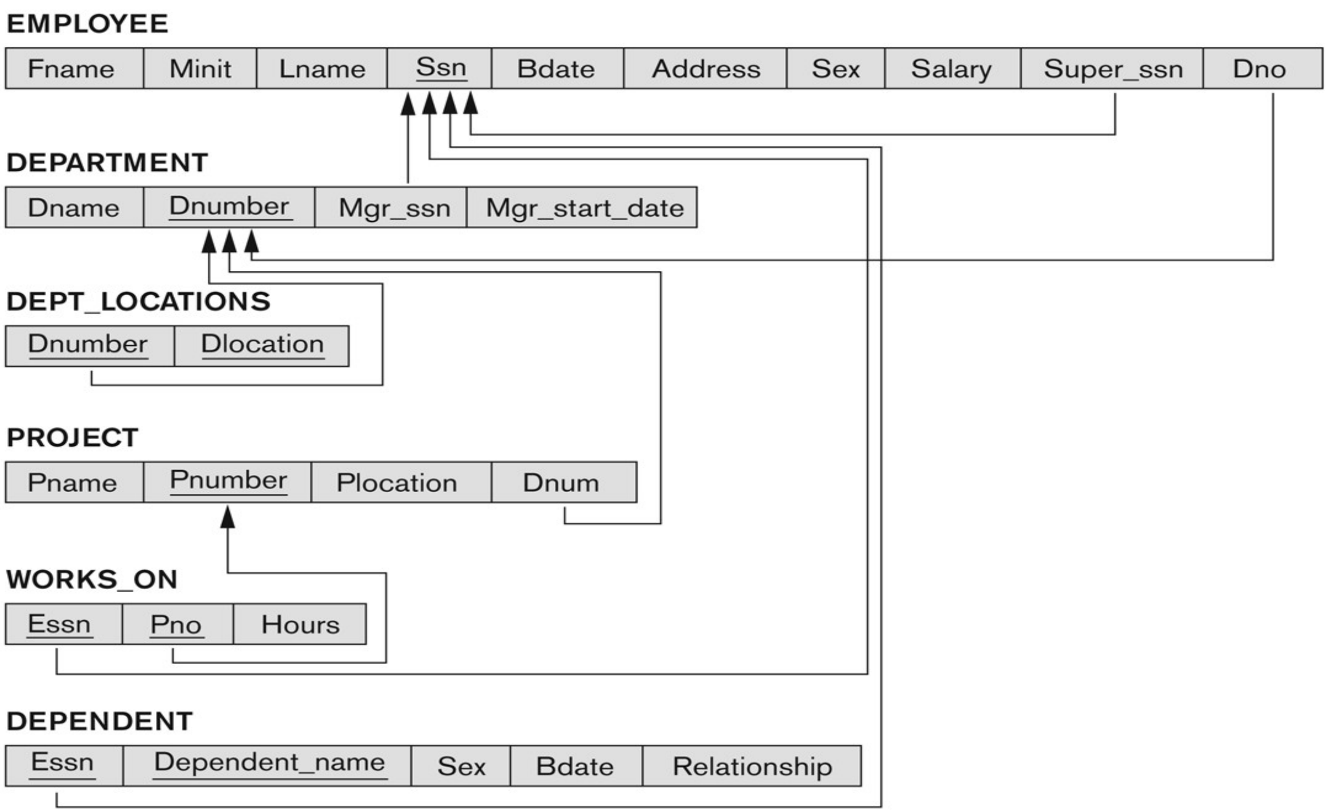
Displaying a relational database schema and its constraints

- Each relation schema can be displayed as a row of attribute names
- The name of the relation is written above the attribute names
- The primary key attribute (or attributes) will be underlined
- A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the primary key of the referenced relation.
- Next slide shows the COMPANY relational schema diagram with referential integrity constraints

Database State for COMPANY

- All examples discussed below refer to the COMPANY database shown here

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



Populated database state for COMPANY

Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Update Operations on Relations

- INSERT a tuple
- DELETE a tuple
- MODIFY a tuple

- Integrity constraints should not be violated by the update operations
- Several update operations may have to be grouped together
- Updates may propagate to cause other updates automatically. This may be necessary to maintain integrity constraints

Possible violations for each operation

- INSERT may violate any of the constraints:
  - Domain constraint: if one of the attribute values provided for the new tuple is not of the specified attribute domain
  - Key constraint: if the value of a key attribute in the new tuple already exists in another tuple in the relation
  - Entity integrity: if the primary key value is null in the new tuple
  - Referential integrity: if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation

Possible violations for each operation

- Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, NULL, ‘1960-04-05’, ‘6357 Windy Lane, Katy,TX’, F, 28000, NULL, 4> into EMPLOYEE.

  --violates the entity integrity constraint (NULL for the primary key Ssn)

- Insert <‘Alicia’, ‘J’, ‘Zelaya’, ‘999887777’, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, ‘987654321’, 4> into EMPLOYEE.

  --violates the key constraint because another tuple with the same Ssn value already exists in the EMPLOYEE relation

- Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’, ‘1960-04-05’, ‘6357 Windswept, Katy, TX’, F, 28000, ‘987654321’, 7> into EMPLOYEE.

  --violates the referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7

Possible violations for each operation

- MODIFY(UPDATE) may violate any of the constraints:
  - Key constraint: if the value of a key attribute in the modified tuple already exists in another tuple in the relation
  - Domain constraint: if one of the attribute values provided for the modified tuple is not of the specified attribute domain
  - Entity integrity: if the primary key value is null in the modified tuple
  - Referential integrity: if a foreign key value in the modified tuple references a primary key value that does not exist in the referenced relation.

Possible violations for each operation

- Update the Sex of the EMPLOYEE tuple with Ssn = ‘999887777’ to 55.

  --violates domain constraint, because domain of sex is character

- Update the Dno of the EMPLOYEE tuple with Ssn = ‘999887777’ to 7.

  --violates referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7

- Update the Ssn of the EMPLOYEE tuple with Ssn = ‘987654321’ to ‘999887777’.

-- violates primary key constraint by repeating a value that already exists as a primary key in another tuple; violates referential integrity constraints because there are other relations that refer to the existing value of Ssn.

Possible violations for each operation

- DELETE may violate only referential integrity:
  - If the primary key value of the tuple being deleted is referenced from other tuples in the database

- Delete the EMPLOYEE tuple with Ssn = ‘999887777’.

--violates the referential integrity constraints, because there are tuples in WORKS_ON that refer to this tuple, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

Integrity Constraints

- In case of integrity violation, several actions can be taken:
  - cancel the operation that causes the violation
  - perform the operation but inform the user of the violation (e.g. ask the user to provide a valid value)
  - trigger additional updates so the violation is corrected (e.g. cascade the deletion by deleting tuples that reference the tuple being deleted)
  - execute a user-specified error-correction routine

Adding Constraints in SQL

```
CREATE TABLE EMPLOYEE
        ( Fname           VARCHAR(15)       NOT NULL,
          Minit           CHAR,
          Lname           VARCHAR(15)       NOT NULL,
          Ssn             CHAR(9)           NOT NULL,
          Bdate           DATE,
          Address         VARCHAR(30),
          Sex             CHAR,
          Salary          DECIMAL(10,2),
          Super_ssn       CHAR(9),
          Dno             INT               NOT NULL,
        PRIMARY KEY (Ssn),
CREATE TABLE DEPARTMENT
        ( Dname           VARCHAR(15)       NOT NULL,
          Dnumber         INT               NOT NULL,
          Mgr_ssn         CHAR(9)           NOT NULL,
          Mgr_start_date  DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber         INT               NOT NULL,
          Dlocation       VARCHAR(15)       NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
```

Adding Constraints in SQL

```
CREATE TABLE PROJECT
        ( Pname                VARCHAR(15)        NOT NULL,
          Pnumber              INT                NOT NULL,
          Plocation            VARCHAR(15),
          Dnum                 INT                NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn                 CHAR(9)            NOT NULL,
          Pno                  INT                NOT NULL,
          Hours                DECIMAL(3,1)       NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn                 CHAR(9)            NOT NULL,
          Dependent_name       VARCHAR(15)        NOT NULL,
          Sex                  CHAR,
          Bdate                DATE,
          Relationship         VARCHAR(8),
        PRIMARY KEY (Essn, Dependent_name),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

Reference Constraints in SQL

- Add constraints after creation

```
ALTER TABLE COUNTRY  ADD CONSTRAINT PK_country PRIMARY KEY (cntry_cd);

ALTER TABLE EXCHANGE ADD CONSTRAINT FK_exchg_cntry FOREIGN KEY
(cntry_cd) REFERENCES COUNTRY (cntry_cd)  ;
```

Functional Dependency

- Functional dependency is a constraint between two sets of attributes from the database
- E.g., In DEPARTMENT, *Dnumber* and *Dname*
  - If you know the department number, you know the department name, so we have *Dnumber* → *Dname*

- A functional dependency denotes by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of a relation schema R, specifies a constraint on the possible tuples that can form a relation state r of R.
- The constraint is that, for any two tuples t1 and t2 in r that have $t1[X] = t2[X]$, they must also have $t1[Y] = t2[Y]$
- The values of the Y component of a tuple in r depend on, or are determined by the values of the X component

Functional Dependency

- Formal definition:

  Let R be a relational schema, and $\alpha \subseteq R$, $\beta \subseteq R$ (ie, α and β are sets of R's attribuets). We say:

$$\alpha \rightarrow \beta$$

  if in any legal relation instance r(R), for all pairs of tuples t1 and t2 in r, we have:

$$(t1[\alpha] = t2[\alpha]) \Rightarrow (t1[\beta] = t2[\beta])$$

Functional Dependency

- Some usages of FDs:

  (1) to set constraints on legal relations (e.g. key constraints)

  (2) to test relations to see if they are "legal" under a given set of FDs.

  (3) to test the goodness of a database schema design (normalization).

Functional Dependency: keys

- A set of one or more attributes {A1, A2, …, An} is a (candidate) key for a relation if:
  - The attributes functionally determine all other attributes of the relation (superkey definition)
  - No proper subset of {A1, A2, …, An} functionally determines all other attributes of R, i.e., a key must be minimal

Functional Dependency: keys

- key
  - If a constraint on R states X is a key of R, then X->Y for any subset of attributes Y of R
  - A key uniquely identifies a tuple
  - The values of all remaining attributes are determined

- A functional dependency is property of the semantics or meaning of the attributes

Functional Dependency: properties

- If $X \rightarrow Y$ in R, this does not say whether or not $Y \rightarrow X$ in R

  Ssn $\rightarrow$ Fname, can it be Fname $\rightarrow$ Ssn? No

- If $X \rightarrow Y$, then $XZ \rightarrow Y$

  Ssn$\rightarrow$ Birthdate, can it be {Ssn, Fname} $\rightarrow$ Birthdate? Yes

- Some FDs are "trivial", since they are always satisfied by all relations:
  - E.g., $A \rightarrow A$, $AB \rightarrow A$, (the right-hand side is a subset of the left-hand side)
  - E.g., {Fname, Sex} $\rightarrow$ Fname

Inference Rules for FDs

- Given a set of FDs F, we can infer additional FDs that hold whenever the FDs in F hold

- Armstrong's inference rules:
  - IR1. (Reflexive) If Y is a *subset of* X, then X → Y
  - IR2. (Augmentation) If X → Y, then XZ → YZ
    - ✓ (Notation: XZ stands for X U Z)
  - IR3. (Transitive) If X → Y and Y → Z, then X → Z

- IR1, IR2, IR3 form a sound and complete set of inference rules
  - Sound: These rules are true
  - Complete: All the other rules that are true can be deduced from these rules

Inference Rules for FDs

- Some additional inference rules that are useful:
  - Decomposition: If X → YZ, then X → Y and X → Z
  - Union: If X → Y and X → Z, then X → YZ
  - Pseudotransitivity: If X → Y and WY → Z, then WX → Z

Inference Rules for FDs

IR1 (reflective rule)	If X is a subset of Y, then X → Y
IR2 (augmentation rule)	If X → Y, then XZ → YZ
IR3 (transitive rule)	If X → Y and Y → Z, then X → Z
IR4 (decomposition rule)	If X →YZ, then X → Y and X → Z
IR5 (union rule)	If X → Y and X → Z, then X → YZ
IR6 (pseudotransitive rule)	If X → Y and WY → Z, then WX → Z

Example

- Suppose we are given a schema R with attributes A, B, C, D, E, F and the FDs are:
  - $A \rightarrow BC$
  - $B \rightarrow E$
  - $CD \rightarrow EF$
  - Show that the FD $AD \rightarrow F$ holds

1. $A \rightarrow BC$ (given)
2. $A \rightarrow C$ (1, decomposition)
3. $AD \rightarrow CD$ (2, augmentation)
4. $CD \rightarrow EF$ (given)
5. $AD \rightarrow EF$ (3 and 4, transitivity)
6. $AD \rightarrow F$ (5, decomposition)

Inference Rules for FDs

- Closure of a set F of FDs is the set F^+ of all FDs that can be inferred from F

- E.g., suppose we specify the following set F of obvious functional dependencies
  - F = {Ssn →{Ename, Bdate, Address,Dnumber}, Dnumber→{Dname, Dmgr_ssn}}
  - Then,
    - Ssn → {Dname, Dmgr_ssn}
    - Ssn → Ssn
    - Dnumber → Dname
    - ….. ….
    - F^+ Including all FDs which can be inferred from F

Equivalence of Sets of FDs

- A set of functional dependencies F is said to cover another set of functional dependency G if every FD in G is also in F^+ (F is a subset of F^+)

- Two sets of FDs F and G are equivalent if:
  - Every FD in F can be inferred from G, and
  - Every FD in G can be inferred from F
  - Hence, F and G are equivalent if $F^+ = G^+$

- Example:
  - F: $A \rightarrow BC$;
  - G: $A \rightarrow B$, $A \rightarrow C$
  - $F^+ = G^+$

Inference Rules for FDs

- Closure of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X
  - Note both X and X^+ are a set of attributes
- If X^+ consists of all attributes of R, X is a superkey for R
  - From the value of X, we can determine the values the whole tuple
- X^+ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F
- From X to find out X^+

Example

- Suppose we are given a schema R with attributes A, B, C, D, E, F, and FDs
- $A \rightarrow BC$
- $E \rightarrow CF$
- $B \rightarrow E$
- $CD \rightarrow EF$
- $\{A\}^+$ =>

1. $A \rightarrow BC$ (given)=> B&C are in A+
2. $A \rightarrow B \rightarrow E$ (decomposition & transitivity) =>E is in A+
3. $E \rightarrow CF$ (decomposition)
4. $A \rightarrow CF$ (transitivity)
5. $A \rightarrow F$ (decomposition, transitivity) => F is in A+
6. A -> A (trival) => A is in A+

$\{A\}^+ = \{A,B,C,E,F\}$
Not a superkey or a key

Example

- Suppose we are given a schema R with attributes Ssn, Ename, Pname, Plocation, Pnumber and Hours, and a FD set F.

  F = {Ssn → Ename

  Pnumber → {Pname, Plocation}

  {Snn, Pnumber} → Hours}

- The following closure sets with respect to F
  - $\{Ssn\}^+$ = {Ssn, Ename}
  - $\{Pnumber\}^+$ = {Pnumber, Pname, Plocation}
  - $\{Ssn, Pnumber\}^+$ = {Ssn, Pnumber, Ename, Pname, Plocation, Hours}
  - {Ssn, Pnumber} is a (candidate) key.

Summary

- Closure of a set F of FDs
  - The set F^+ of all FDs that can be inferred from F
- Closure of a set of attributes X with respect to F
  - The set X^+ of all attributes that are functionally determined by X
- A set of functional dependencies F is said to cover another set of functional dependency G
  - If every FD in G is also in F^+
- Two sets of FDs F and G are equivalent
  - if and only if $F^+ = G^+$

References

- 6e
  - Ch. 3, p. 63 – 70