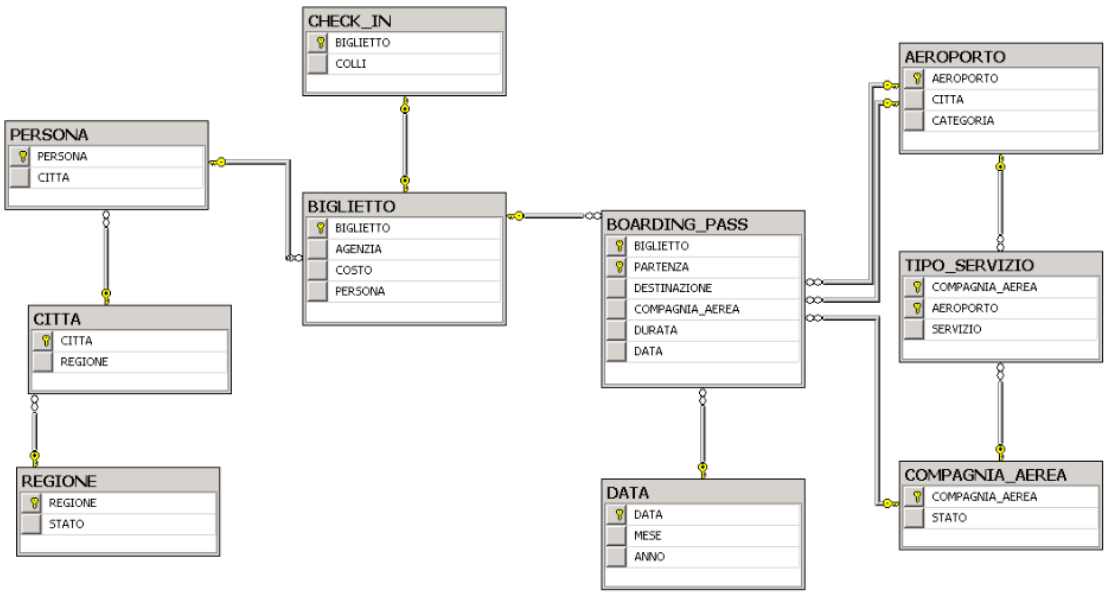
Sistemi Informativi Avanzati
Anno Accademico 2015/2016
Prof. Domenico Beneventano

Reverse engineering
di schemi relazionali
in schemi E/R

Diagramma Relazionale



Data Profiling

- Relazione AEROPORTO: CITTA è AK ?
  Si in quanto entrambe le seguenti query hanno un risultato vuoto

```
SELECT CITTA
FROM AEROPORTO
WHERE CITTA IS NULL
```

```
SELECT CITTA
FROM AEROPORTO
GROUP BY CITTA
HAVING COUNT(*)>1
```

- Relazione DATA: MESE → ANNO?
  Si in quanto la seguente query ha un risultato vuoto

```
SELECT MESE
FROM DATA
GROUP BY MESE
HAVING COUNT(DISTINCT ANNO)>1
```

Schema Relazionale

REGIONE(REGIONE, STATO)

CITTA(CITTA, REGIONE:REGIONE)

PERSONA(PERSONA, CITTA:CITTA)

AEROPORTO(AEROPORTO, CITTA:CITTA, CATEGORIA)
AK : CITTA

DATA(DATA, MESE,ANNO)
FD: MESE → ANNO

BIGLIETTO(BIGLIETTO, DATA:DATA, AGENZIA, COSTO, PERSONA:PERSONA)

CHECK-IN(BIGLIETTO :BIGLETTO, COLLI)

COMPAGNIA_AEREA(COMPAGNIA_AEREA, STATO)

BOARDING_PASS(BIGLIETTO:BIGLIETTO, PARTENZA:AEROPORTO,
DESTINAZIONE: AEROPORTO,
COMPAGNIA_AEREA:COMPAGNIA_AEREA, DURATA)

TIPO_SERVIZIO(COMPAGNIA_AEREA:COMPAGNIA_AEREA,
AEROPORTO: AEROPORTO, SERVIZIO)

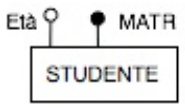
Reverse engineering di schemi relazionali in E/R

- Dallo schema relazionale ottenere lo schema E/R *equivalente*
- Procedimento inverso della Progettazione logico-relazionale: dato uno schema E/R tradurlo in uno schema relazionale
  1. *Regole di semplificazione dello schema E/R* per eliminare identificatori esterni, gerarchie, …
  2. *Regole di traduzione logico-relazionale* per tradurre uno schema E/R in uno schema relazionale normalizzato
- Il Reverse Engineering di schemi relazionali in schemi E/R si basa sull'applicazione in **senso inverso** di queste regole di semplificazione e traduzione.
- Per rendere efficace il processo si deve partire da uno schema relazionale completo con le indicazioni di chiavi e di integrità referenziale (chiavi esterne), valori nulli, dipendenze funzionali.

Individuazione di gerarchie di generalizzazione (*is-a*)

- Si inizia considerando le relazioni senza chiavi esterne, che corrispondono sicuramente a entita' nello schema E/R

STUDENTE(MATR, ETA)

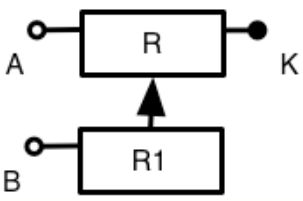


- Uno schema relazionale con

```
R(K, A, ...)

R1(K1, B, ...)
  FK: K1 REFERENCES R
```

corrisponde in E/R ad una gerarchia di generalizzazione (subset) R1 is-a R



- Non e' possibile individuare in modo automatico gerarchie con piu' entita' figlie in quanto questa informazione non e' presente nello schema relazionale
  - Per individuare tali gerarchie occorre far riferimento ad altre informazioni aggiuntive

Individuazione di associazioni uno-a-molti

- Dato uno schema relazionale con

```
R(K,...  )

R1(K1, ... KR, ...)
  FK: KR REFERENCES R
```

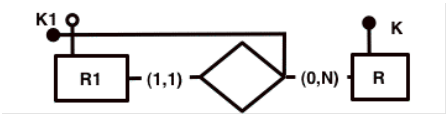
- Se KR non è chiave in R2, allora la FK traduce una associazione uno-a-molti

- Se KR è una chiave in R1, allora l'associazione A è uno-a-uno : N=1

- Se KR è una parte della chiave di R1:

```
R1(K1,KR, ...)
```

allora la FK traduce un identificatore esterno

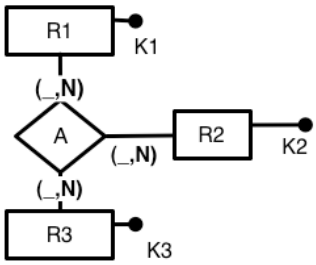


Individuazione di associazioni *n-arie*

- In generale, una relazione R con *n* chiavi esterne *RKi* riferite ad *n* relazioni *Ri* individua una associazione tra le Ri (dalla regola di traduzione-standard)

```
R (RK1,..., RKn, ...)
  FK: KR1 REFERENCES R1
  ...
  FK: KRn REFERENCES Rn
```

- Se la chiave di R è l'insieme di tutte le KRi:
  `R (RK1,..., RKn, ...)`
  allora tutte le entità partecipano con cardinalità max N.
  In questo caso non c'e' nessuna dipendenza funzionale tra le KRi.

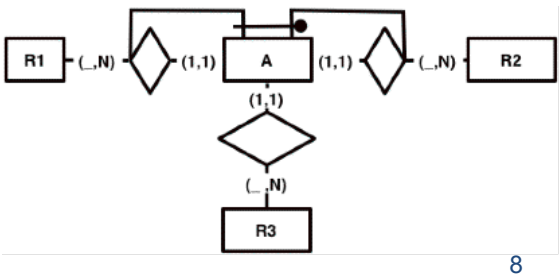


- Se la chiave di R è un sottoinsieme delle FK allora si esprime tramite reificazione.
  Esempio:

```
R (RK1,RK2,RK3, ...)
 FK: KR1 REFERENCES R1
     FK: KR2 REFERENCES R2
     FK: KR3 REFERENCES R3
```

esprime la dipendenza funzionale

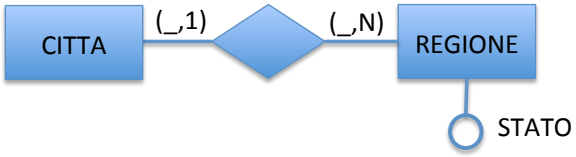
`RK1,RK2 → RK3`



ASSOCIAZIONI **UNO-A-MOLTI**

REGIONE(<u>REGIONE</u>, STATO)
CITTA(<u>CITTA</u>, REGIONE:REGIONE)

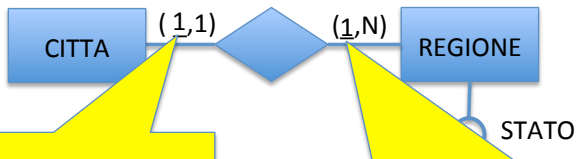
REGIONE è **Foreign** key



ASSOCIAZIONI **UNO-A-MOLTI**

REGIONE(<u>REGIONE</u>, STATO)
CITTA(<u>CITTA</u>, REGIONE:REGIONE)

REGIONE è **Foreign** key



1 se
REGIONE **NOT NULL**
oppure se la query è vuota:

```
SELECT *
FROM CITTA
WHERE REGIONE IS NULL
```

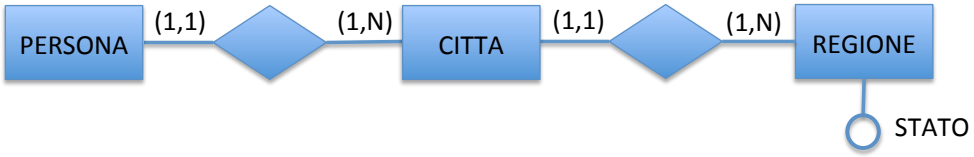
1 se **la query è vuota:**

```
SELECT *
FROM CITTA C RIGHT JOIN REGIONE R
      on (R.REGIONE=C.REGIONE)
WHERE C.REGIONE IS NULL
```

ASSOCIAZIONI **UNO-A-MOLTI**

REGIONE(<u>REGIONE</u>, STATO)
CITTA(<u>CITTA</u>, REGIONE:REGIONE)
PERSONA(<u>PERSONA</u>, CITTA:CITTA)

CITTA è
Foreign key

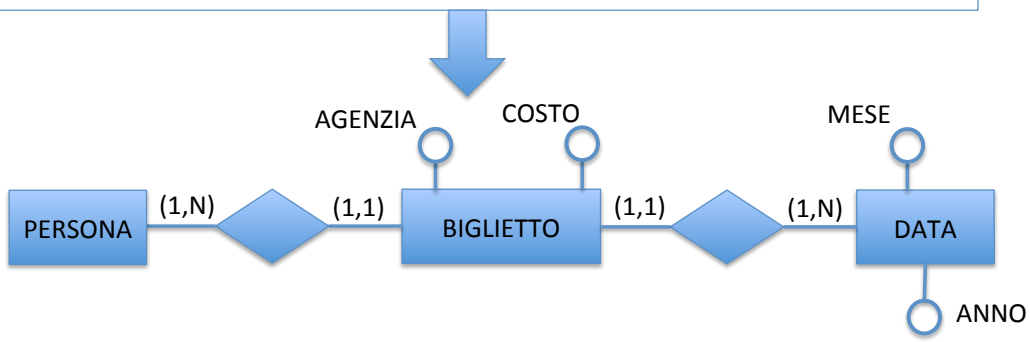


In questo esempio, tutte le cardinalità minime si ipotizzano =1

ASSOCIAZIONI **UNO-A-MOLTI**

DATA(<u>DATA</u>, MESE,ANNO)
FD: MESE → ANNO

BIGLIETTO(<u>BIGLIETTO</u>, DATA:DATA, AGENZIA, COSTO, PERSONA:PERSONA)

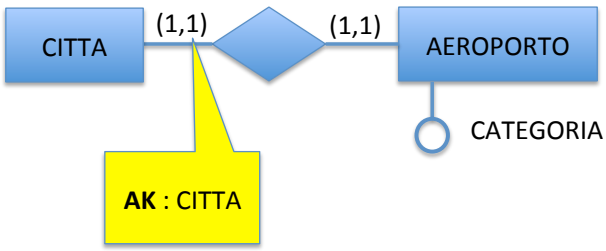


ASSOCIAZIONI **UNO-A-UNO**

CITTA(CITTA, REGIONE:REGIONE)

AEROPORTO(AEROPORTO, CITTA:CITTA, CATEGORIA)
AK : CITTA

CITTA è
sia **Foreign** key che **Alternative** key



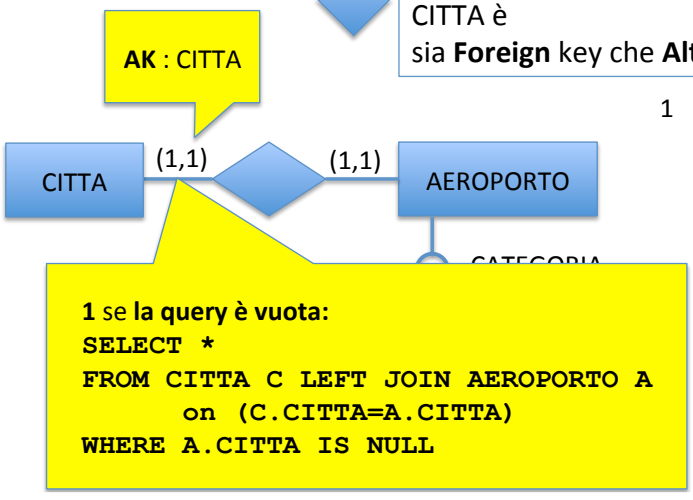
ASSOCIAZIONI **UNO-A-UNO**

CITTA(CITTA, REGIONE:REGIONE)

AEROPORTO(AEROPORTO, CITTA:CITTA, CATEGORIA)
AK : CITTA

CITTA è
sia **Foreign** key che **Alternative** key

1

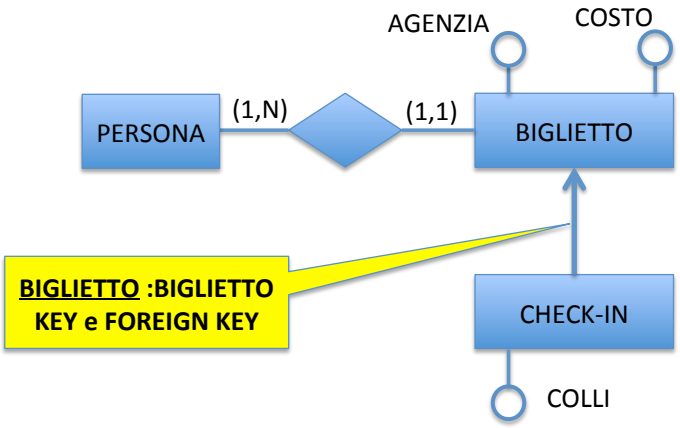


GERARCHIE

BIGLIETTO(<u>BIGLIETTO</u>, AGENZIA, COSTO, PERSONA:PERSONA)

CHECK-IN(<u>BIGLIETTO :BIGLETTO</u>, COLLI)

BIGLIETTO è
sia **Foreign** key che **Primary** key



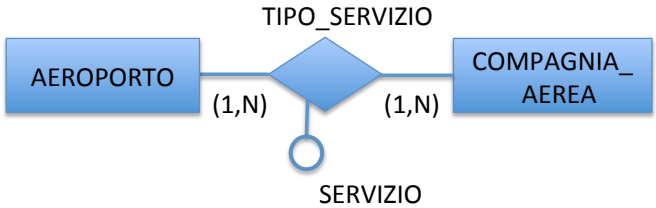
ASSOCIAZIONI **MOLTI-A-MOLTI**

TIPO_SERVIZIO(<u>COMPAGNIA_AEREA:COMPAGNIA_AEREA</u>,
<u>AEROPORTO: AEROPORTO,</u>
SERVIZIO)

AEROPORTO è
sia **Foreign** key che ***parte della*** **Primary** key

COMPAGNIA_AEREA è
sia **Foreign** key che ***parte della*** **Primary** key

Non ci sono altre **Foreign Key**



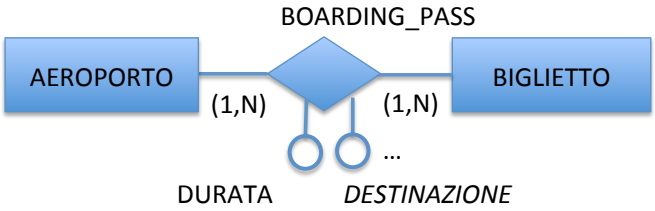
ASSOCIAZIONI **MOLTI-A-MOLTI**

BOARDING_PASS(<u>BIGLIETTO:BIGLIETTO, PARTENZA:AEROPORTO</u>,
DESTINAZIONE: AEROPORTO,
COMPAGNIA_AEREA:COMPAGNIA_AEREA, DURATA)

BIGLIETTO è
sia **Foreign** key che ***parte della*** **Primary** key

PARTENZA è
sia **Foreign** key che ***parte della*** **Primary** key

NO : Ci sono altre **Foreign Key**



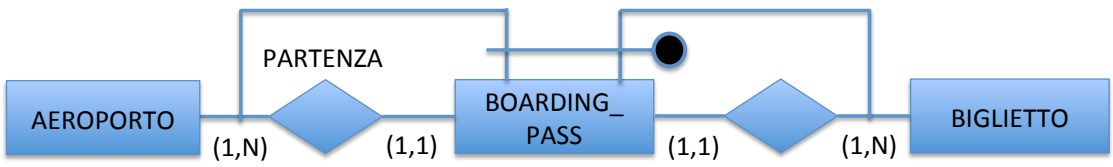
ASSOCIAZIONI **REIFICATE**

BOARDING_PASS(<u>BIGLIETTO:BIGLIETTO, PARTENZA:AEROPORTO</u>,
DESTINAZIONE: AEROPORTO,
COMPAGNIA_AEREA:COMPAGNIA_AEREA, DURATA)

BIGLIETTO è
sia **Foreign** key che ***parte della*** **Primary** key

PARTENZA è
sia **Foreign** key che ***parte della*** **Primary** key

IDENTIFICATORE ESTERNO

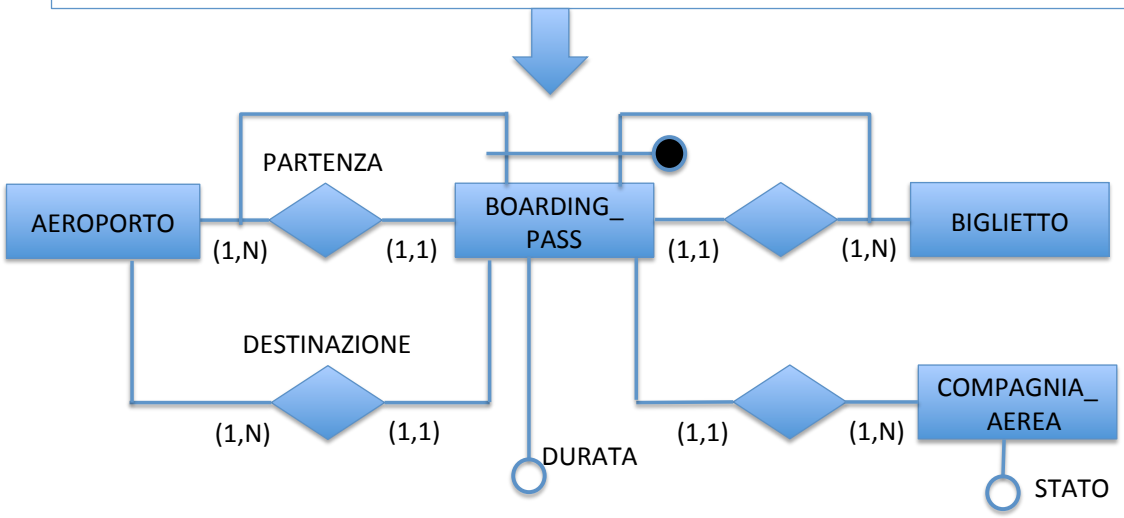


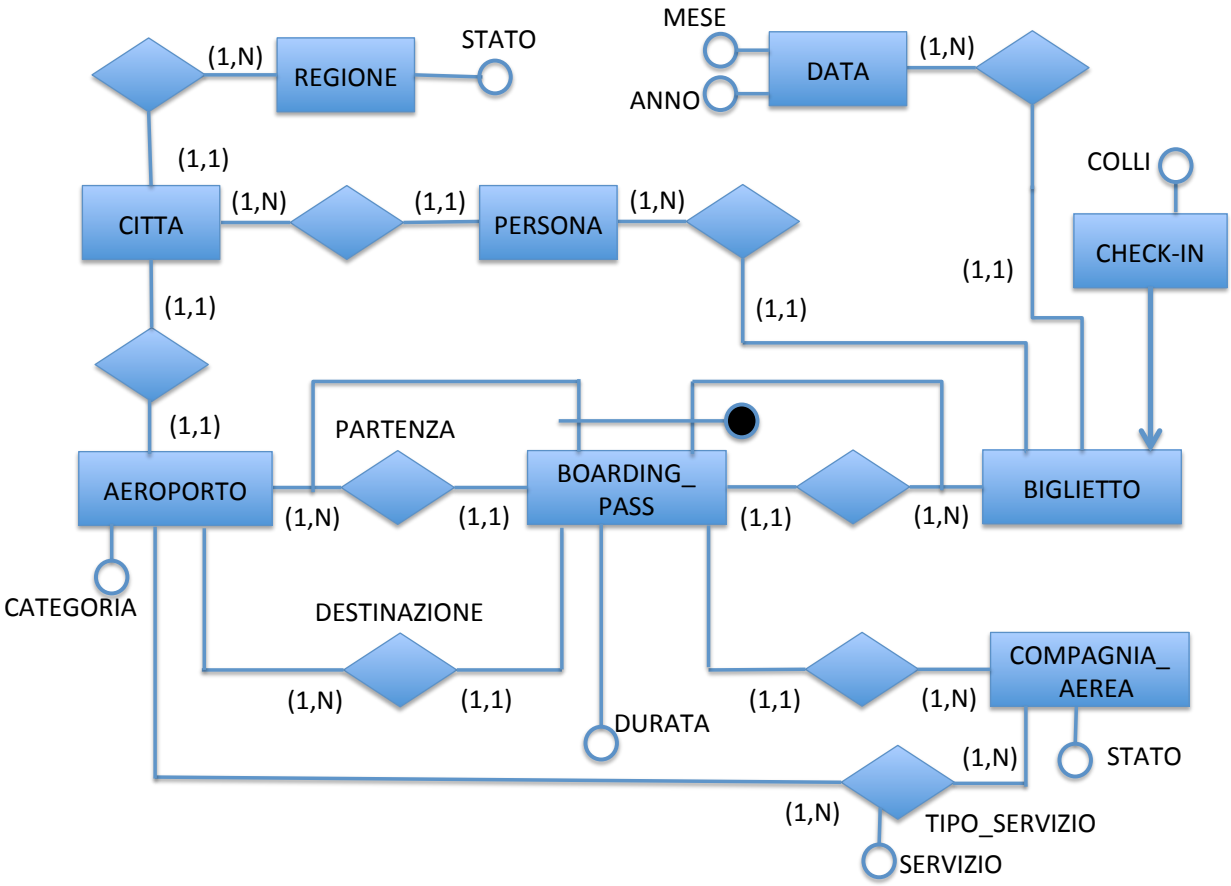
Le altre **Foreign Key** ➔ **Associazioni UNO-A-MOLTI**

ASSOCIAZIONI **REIFICATE**

BOARDING_PASS(<u>BIGLIETTO:BIGLIETTO, PARTENZA:AEROPORTO</u>,
DESTINAZIONE: AEROPORTO,
COMPAGNIA_AEREA:COMPAGNIA_AEREA, DURATA)

COMPAGNIA_AEREA(<u>COMPAGNIA_AEREA</u>, STATO)





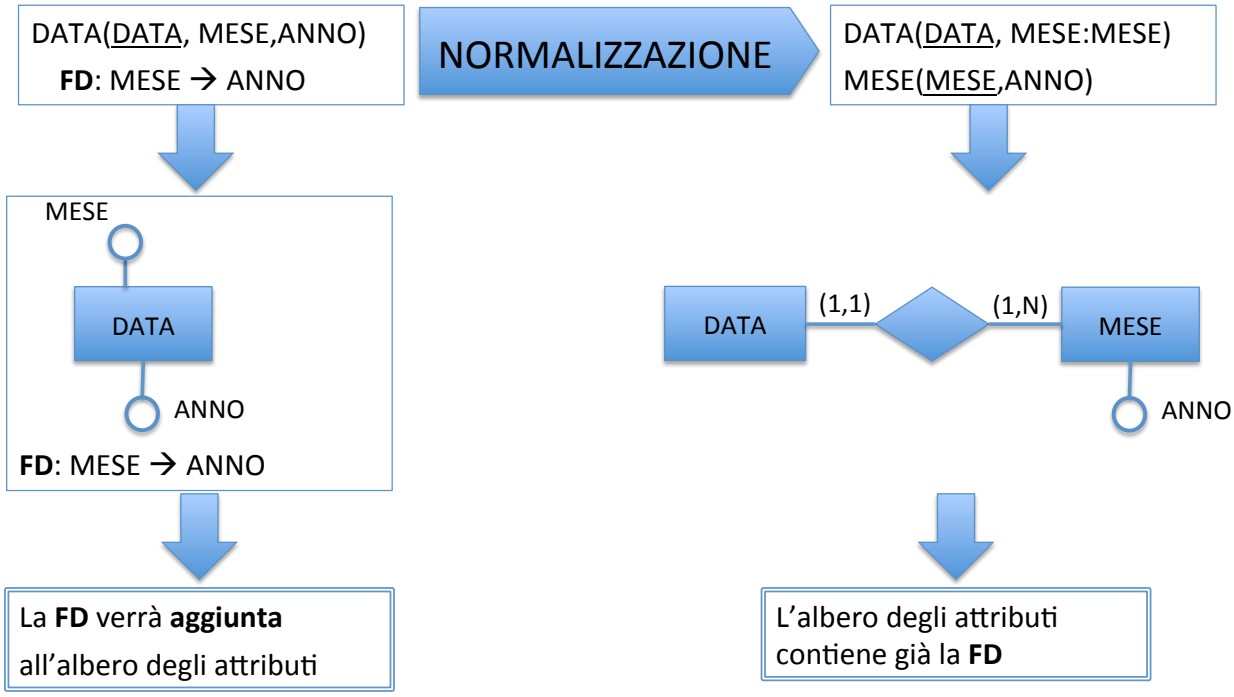
Requisiti

- **Progettazione concettuale**:
  schema di fatto BOARDING_PASS con 5 **dimensioni:**
  1. AEROPORTO_ARRIVO
  2. BIGLIETTO
  3. REGIONE_PARTENZA
  4. COMPAGNIA_AEREA
  5. DATA

  e misure
  1. NUMERO: è il numero di *viaggi*, cioè il numero di boarding_pass
  2. DURATA: è la durata media dei viaggi (riportata nei boarding_pass)

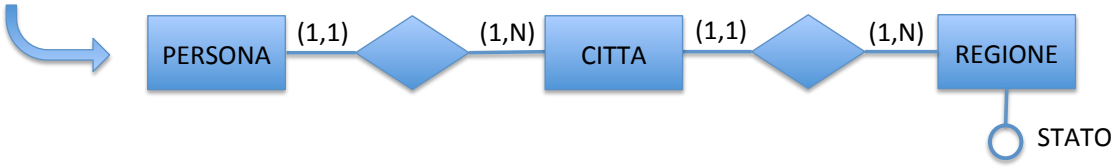
- **Cosa cambia** considerando come dimensione
  CITTA_PARTENZA al posto di REGIONE_PARTENZA?

DIPENDENZE FUNZIONALI



IDENTIFICATORE **ESTERNO**

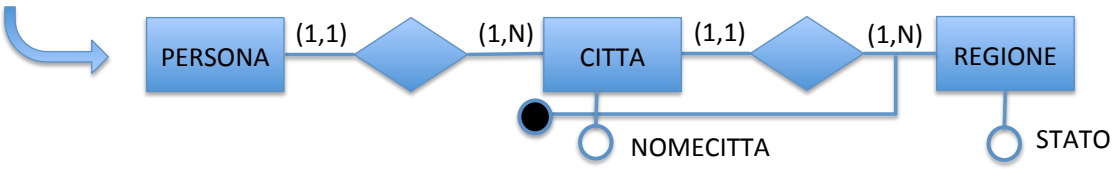
REGIONE(<u>REGIONE</u>, STATO)
CITTA(<u>CITTA</u>, REGIONE:REGIONE)
PERSONA(<u>PERSONA</u>, CITTA:CITTA)



➢ Se nella chiave di CITTA si include anche la Foreign Key

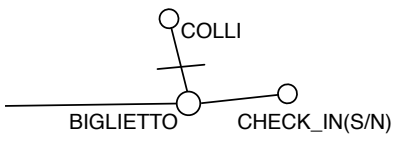
REGIONE(<u>REGIONE</u>, STATO)
CITTA(<u>NOMECITTA, REGIONE:REGIONE</u>)

PERSONA(<u>PERSONA</u>, [NOMECITTA,REGIONE]:CITTA)
oppure **FK** NOMECITTA,REGIONE **REFERENCES** CITTA

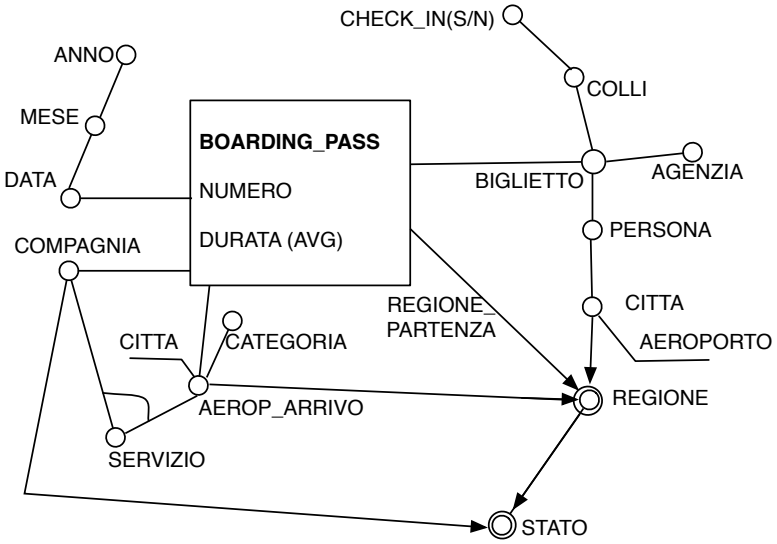


SOLUZIONE – Schema di Fatto

- Dell'AEROPORTO_PARTENZA viene tenuta solo la REGIONE_PARTENZA con il relativo STATO
- SERVIZIO deriva dall'associazione molti-a-molti tra AEROPORTO e COMPAGNIA_AEREA (nel seguito COMPAGNIA): è un attributo cross-dimensionale tra AEROPORTO_ARRIVO (è l'unico concetto di aeroporto presente nello schema e COMPAGNIA)
- In AEROPORTO_ARRIVO, CITTA è in associazione uno-a-uno, quindi viene riportato come attributo descrittivo
- Il COSTO del biglietto non viene richiesto come misura; non viene considerato come attributo dimensionale
- Il subset CHECK_IN viene riportato tramite un attributo booleano CHECK_IN(S/N) ed un attributo opzionale COLLI.
  Se il valore opzionale viene codificato con 0, si può togliere l'opzionalità e aggingere la FD : COLLI → CHECK_IN(S/N)



SOLUZIONE – Schema di Fatto



- In CITTA, AEROPORTO è in associazione uno-a-uno, quindi viene riportato come attributo descrittivo

SOLUZIONE – Schema di Fatto

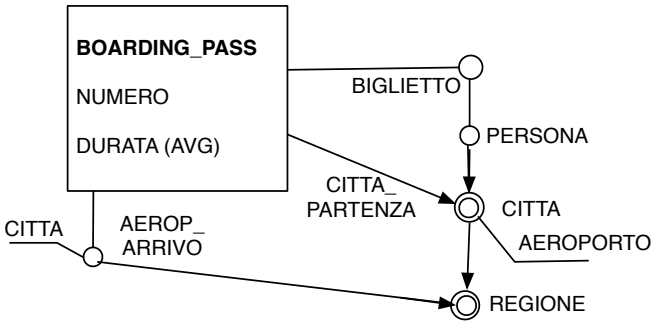
- Dimensioni **D** = {DATA,COMPAGNIA,BIGLIETTO,REGIONE_PARTENZA,AEROP_ARRIVO}
- FD tra le dimensioni : BIGLIETTO → DATA
- Fatto BOARDING_PASS(<u>BIGLIETTO, AEROP_ARRIVO</u>, …
- **D** non contiene alcuna chiave del fatto ➔ schema **temporale**

- **Glossario delle misure**
  1. DURATA = **AVG**(BOARDING_PASS.DURATA)
  2. NUMERO = **COUNT**(*)

➢ Nello studio dell'aggregabilità delle misure vedremo che DURATA sarà una **<u>misura calcolata</u>** dalle componenti DURATA_SUM e DURATA_COUNT.
In questo esempio useremo solo la misura additiva **NUMERO**.

SOLUZIONE – variante

- Considerando come dimensione CITTA_PARTENZA al posto di REGIONE_PARTENZA, lo schema risulta (solo le parti modificate):



- Dimensioni **D** ora contiene {BIGLIETTO,CITTA_PARTENZA} che costituisce una chiave del fatto BOARDING_PASS(BIGLIETTO, AEROP_ARRIVO, …
  ➔ schema **transazionale**
- **Glossario delle misure : cambia** la definizione, **non** si deve più usare un operatore di aggregazione
  1. DURATA = ~~**AVG**~~(BOARDING_PASS.DURATA)
  2. NUMERO = ~~**COUNT(*)**~~ **= 1**

DBO: Schema Relazionale

REGIONE(REGIONE, STATO)

CITTA(CITTA, REGIONE:REGIONE)

PERSONA(PERSONA, CITTA:CITTA)

AEROPORTO(AEROPORTO, CITTA:CITTA, CATEGORIA)
AK : CITTA

DATA(DATA, MESE,ANNO)
FD: MESE → ANNO

BIGLIETTO(BIGLIETTO, DATA:DATA, AGENZIA, COSTO, PERSONA:PERSONA)

CHECK-IN(BIGLIETTO :BIGLETTO, COLLI)

COMPAGNIA_AEREA(COMPAGNIA_AEREA, STATO)

BOARDING_PASS(BIGLIETTO:BIGLIETTO, PARTENZA:AEROPORTO,
DESTINAZIONE: AEROPORTO,
COMPAGNIA_AEREA:COMPAGNIA_AEREA, DURATA)

TIPO_SERVIZIO(COMPAGNIA_AEREA:COMPAGNIA_AEREA,
AEROPORTO: AEROPORTO, SERVIZIO)

DW: Schema di Fatto