Esempio per Tesina – 2015/2016

Sommario

PRIMA CONSEGNA

In questo capitolo sono dettagliati tutti i passi richiesti per la prima consegna della tesina

Il punto di partenza è il diagramma relazionale del DBO con la relativa documentazione

Per ognuno dei FATTI, i passi principali da svolgere sono

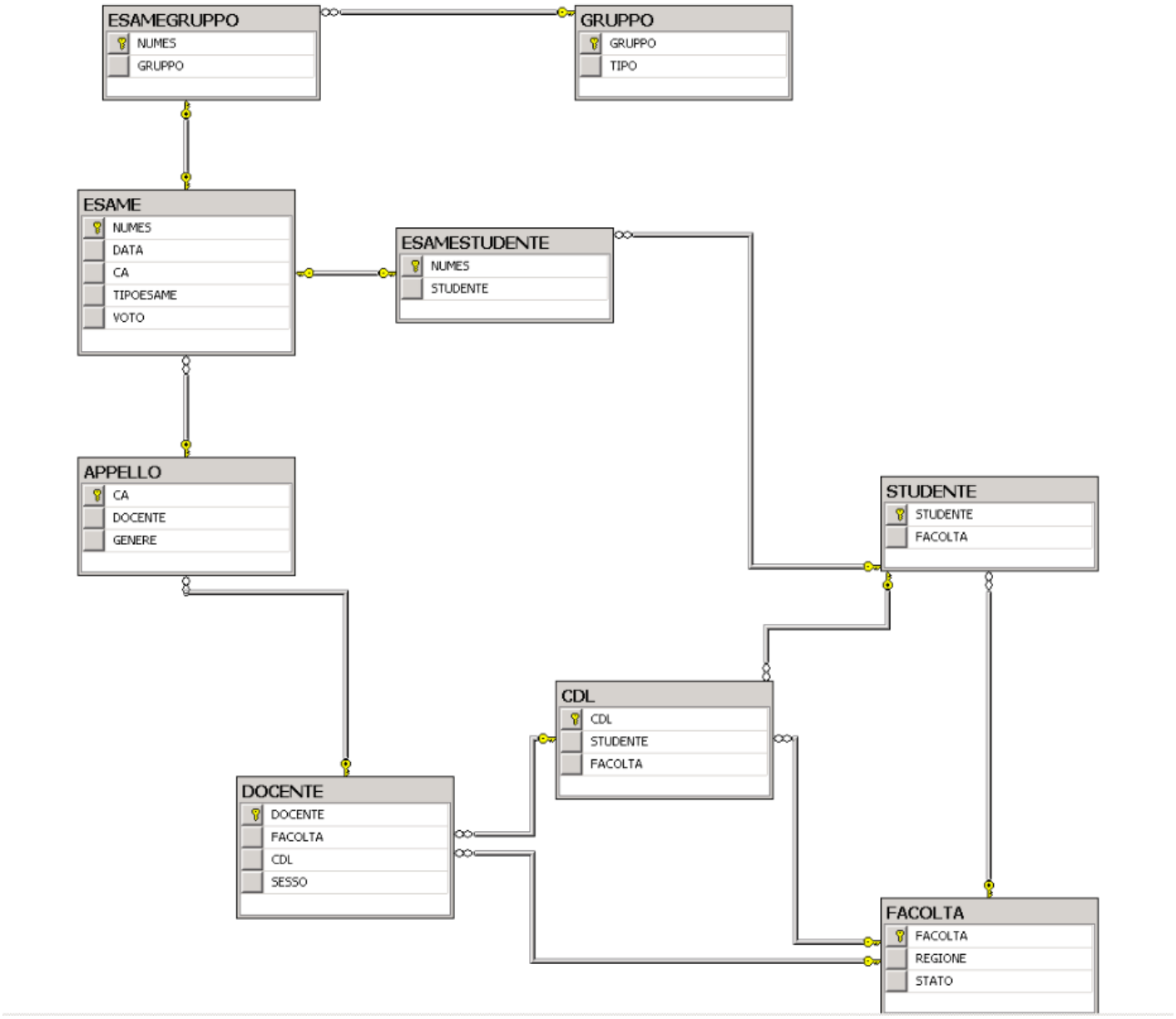
A) **DATA PROFILING**

B) **SCHEMA RELAZIONALE**

C) **SCHEMA ER**

D) **PROGETTO CONCETTUALE**
   a. **ALBERO DEGLI ATTRIBUTI**
   b. **SCHEMA DI FATTO,**
      indicando se TEMPORALE o TRANSAZIONALE e le FD tra le dimensioni

1.1 DIAGRAMMA RELAZIONALE e Documentazione del DBO



Documentazione:

FK: CDL → FACOLTA un CDL (CorsoDiLaurea) ha *sede* in una FACOLTA

FK: DOCENTE → CDL **:** un DOCENTE è di un CDL

FK: APPELLO → DOCENTE: un APPELLO è con il DOCENTE che lo tiene

FK: CDL→ STUDENTE: un CDL ha uno STUDENTE rappresentante

Attributo **TIPO_ESAME** di ESAME : assume il valore STUD se è un esame di uno studente ed il valore GRUP se è un esame di un gruppo

CDL è sinonimo di CDS

1.2 DATA PROFILING

Chiavi Alternative:

in CDS , ho AK: STUDENTE ? Si effettuano le query relative

```
SELECT *
FROM CDL
WHERE STUDENTE IS NULL
```

```
SELECT STUDENTE
FROM CDL
GROUP BY STUDENTE
HAVING COUNT(*) > 1
```

La seconda query ha un risultato non vuoto quindi la risposta è negativa, cioè non ho AK quindi uno studente può essere rappresentante in più CDL!!

Dipendenza funzionale:

in FACOLTA , ho FD: REGIONE → STATO ? Si effettua la query relativa

```
SELECT REGIONE
FROM FACOLTA
GROUP BY REGIONE
HAVING COUNT(DISTINCT STATO) > 1
```

e la risposta è positiva in quanto la query non restituisce niente.

1.3 SCHEMA RELAZIONALE

Viene riportato lo schema relazionale, comprensivo di key (primary Key ed eventuali Alternative Key) , Foreing Key e Dipendenze Funzionali

FACOLTA(FACOLTA,REGIONE, STATO)

FD: REGIONE → STATO

CDS(CDS,FACOLTA:FACOLTA,STUDENTE:STUDENTE)

DOCENTE(DOCENTE,FACOLTA:FACOLTA,CDL:CDL,SESSO)

STUDENTE(STUDENTE, FACOLTA:FACOLTA)

APPELLO(APPELLO,DOCENTE:DOCENTE, GENERE)

ESAME(NUMES, APPELLO:APPELLO, DATA, TIPO_ESAME, VOTO)

ESAMEGRUPPO(NUMES:ESAME, GRUPPO:GRUPPO)

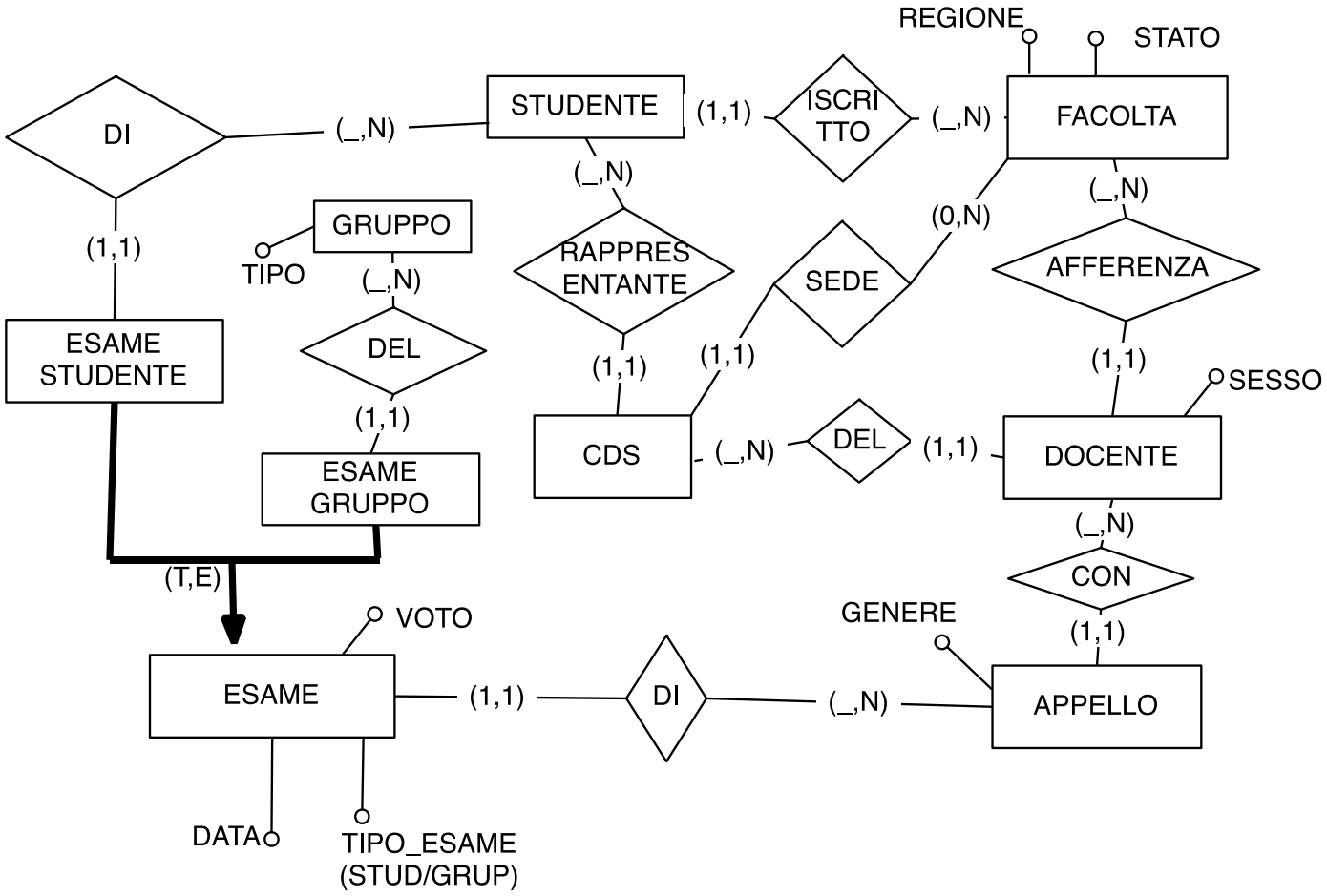
ESAMESTUDENTE(NUMES:ESAME, STUDENTE:STUDENTE)

GRUPPO(GRUPPO,TIPO)

1.4 SCHEMA ER

Effettuare il reverse-engineering ed ottenere il relativo schema ER.

Vedere http://www.dbgroup.unimore.it/SIA/EsempioReverseEngineeringDaSchemiER.pdf



NOTE

1) Le chiavi primarie, cioè gli identificatori primari delle entità, non sono indicati, in quanto *sottintesi*: ogni entità ha come identificatore il singolo attributo che costituisce la chiave primaria

2) Sono stati messi anche i nomi alle associzioni binarie uno-a-molti: i nomi non sono *importanti,* servono per indicare l'*interpretazione* del progettista (un DOCENTE è di una FACOLTA viene specificato dal progettista come AFFERENZA, cioè un DOCENTE afferisce ad una FACOLTA)

3) Le cardinalità minime nel caso (_,N) sono lasciate indefinite; nella tesina occorre riportarle (vedere esempio http://www.dbgroup.unimore.it/SIA/EsempioReverseEngineeringDaSchemiER.pdf

4) Si suppone sia stato verificato sui dati che la gerarchia su ESAME sia totale/esclusiva: vedi discussione in sezione 1.7

1.5 PROGETTO CONCETTUALE

FATTO: ESAME
Con Dimensioni

GRUPPO, STUDENTE, **TIPOESAME**,**DATA, DOCENTE**

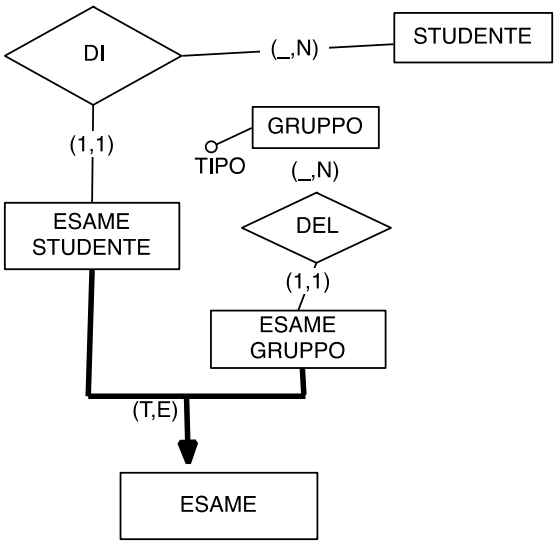
Cioè si vogliono **analizzare gli esami** senza scendere nel dettaglio del singolo APPELLO ma considerando solo il DOCENTE che ha tenuto l'APPELLO. Si vuole effettuare l'analisi anche rispetto allo studente e/o gruppo che ha sostenuto l'esame, al tipo di esame e alla data.

MISURE: per la prima consegna limitarsi a definire le misure indicandone solo il nome, una breve descrizione e se eventalmente prevedete che siano delle medie, dei MAX, etc. Non servono per la prima consegna altri dettagli sulle misure; le misure verranno definite in modo dettagliato nella seconda parte.
In questo esempio:

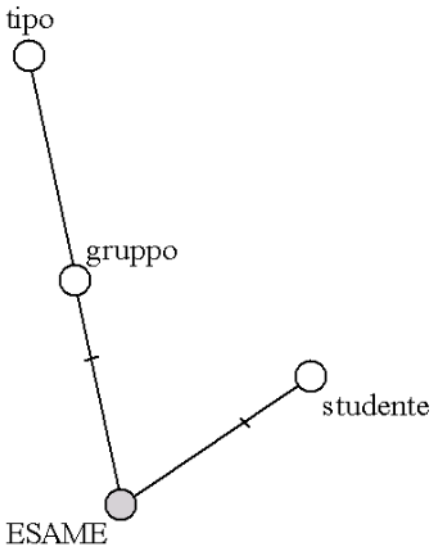
NUMERO ESAMI, è il numero degli esami fatti
VOTO ESAME MEDIO, è il voto medio riportato all'esame

E' importante osservare che una misura, ad esempio NUMERO ESAMI verrà poi calcolata rispetto alle dimensioni (in base al suo operatore di aggregazione, in questo caso la SUM, cioè misura addittiva): pertanto è sbagliato riportare anche altre misure quali NUMERO ESAMI GRUPPO intendendo il numero di esami che un gruppo ha fatto, in quanto questa è la misura NUMERO ESAMI valutata per un certo gruppo

Cominciamo la progettazione:
ESAME ha una gerarchia composta da ESAME_STUDENTE (con lo STUDENTE ASSOCIATO) ed ESAME_GRUPPO (con il GRUPPO ASSOCIATO)

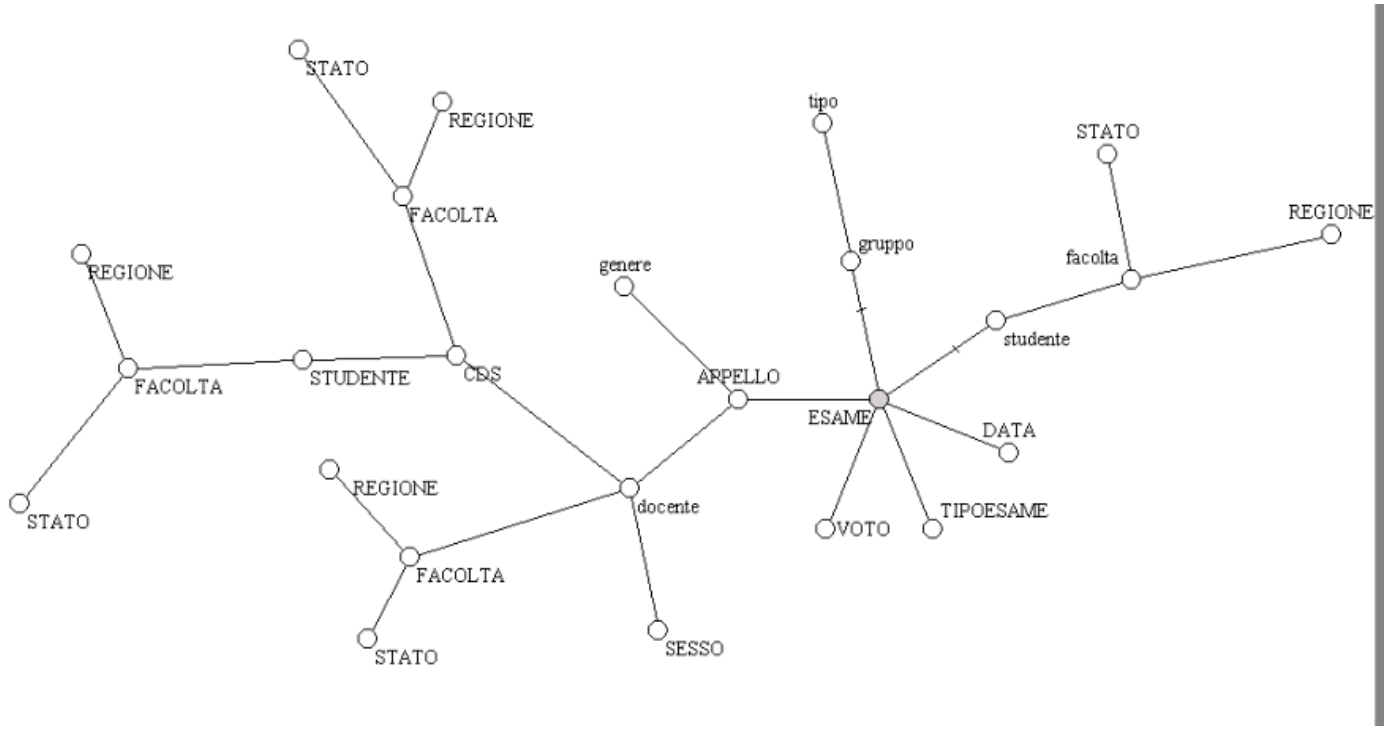


Come spiegato nell'esempio di pag. 25 delle dispense
http://www.dbgroup.unimo.it/SIA/SIA_2015_IlModelloConcettualeDFM.pdf
ed in maniera analoga a quanto fatto nell'esempio del BIGLIETTO con CHECK_IN, considerando come radice ESAME, la gerarchia viene rappresentata nell'albero degli attributi nel seguente modo

Quando un ESAME è un ESAME_STUDENTE avrà associato uno STUDENTE (che risulta quindi opzionale) Quando un ESAME è un ESAME_GRUPPO avrà associato uno GRUPPO (che risulta quindi opzionale); al gruppo è associato il TIPO Si può aggiungere nell'albero degli attributi (oppure nella documentazione allegata) il fatto che GRUPPO e STUDENTE rispetto ad ESAME sono Totale ed Esclusivo. Questo è un dettaglio che verrà considerato nel seguito.	

A questo punto si costruisce il resto dell'albero degli attributi in modo ***praticamente automatico***:

- per ogni associazione uno-a-molti si aggiunge il relativo ramo e si continua a sviluppare l'albero
- per ogni attributo si aggiunge il relativo ramo e ci si ferma



L'albero degli attributi si può disegnare su carta e quindi includero (foto o scanner) nella tesina, non è necessario usare un editor

IMPORTANTE

Nella progettazione (e quindi nell'albero) riportare solo gli attributi *significativi* dello schema relazionale: con riferimento alla tesina, attributi quali **rowguid**, **Modified Date** o altri attibuti che avete già deciso di non usare (ad esempio attributi che non hanno valore nel DB) non dovrebbero essere utilizzati

L'albero degli attributi evidenzia che nel database per gli esami sono riportate **quattro facoltà**

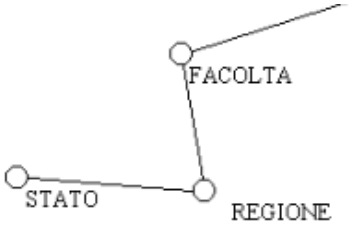
1) la facoltà del docente dell'appello dell'esame
2) la facoltà del CDS del docente dell'appello dell'esame
3) la facoltà dello studente rappresentante del CDS
4) la facoltà dello studente (eventuale) che ha sostenuto l'esame

E' ovviamente un caso limite, però non sono rari i casi in cui uno stesso concetto (la FACOLTA nel nostro esempio) compare due o più volte nell'analisi di un dato fatto.

Inizialmente si considera il caso generale di CONDIVISIONE: queste quattro facoltà possono essere quattro facoltà distinte. Poi successivamente si effettua l'enalisi di eventuali convergenze, cioè si analizza se alcune di queste facoltà sono coincidenti

Editing dell'albero: SI AGGIUNGONO LE FD individuate (e non riportate nello schema E/R). Nel nostro esempio la FD in FACOLTA : REGIONE → STATO

Grazie alla condivisione questa operazione viene fatta una sola volta



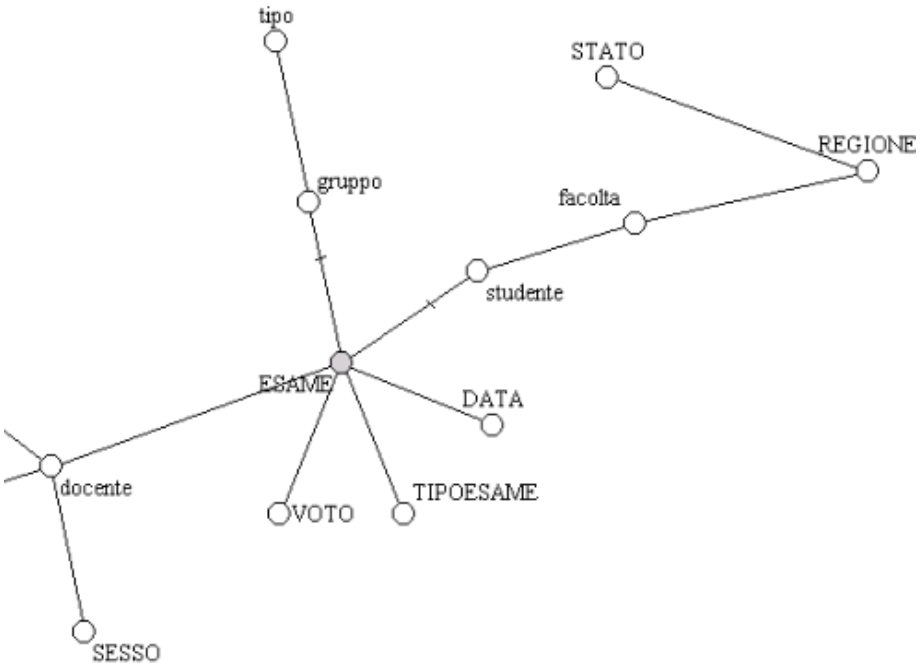
Consideriamo ora le dimensioni

GRUPPO, STUDENTE, **TIPOESAME**,**DATA, DOCENTE**

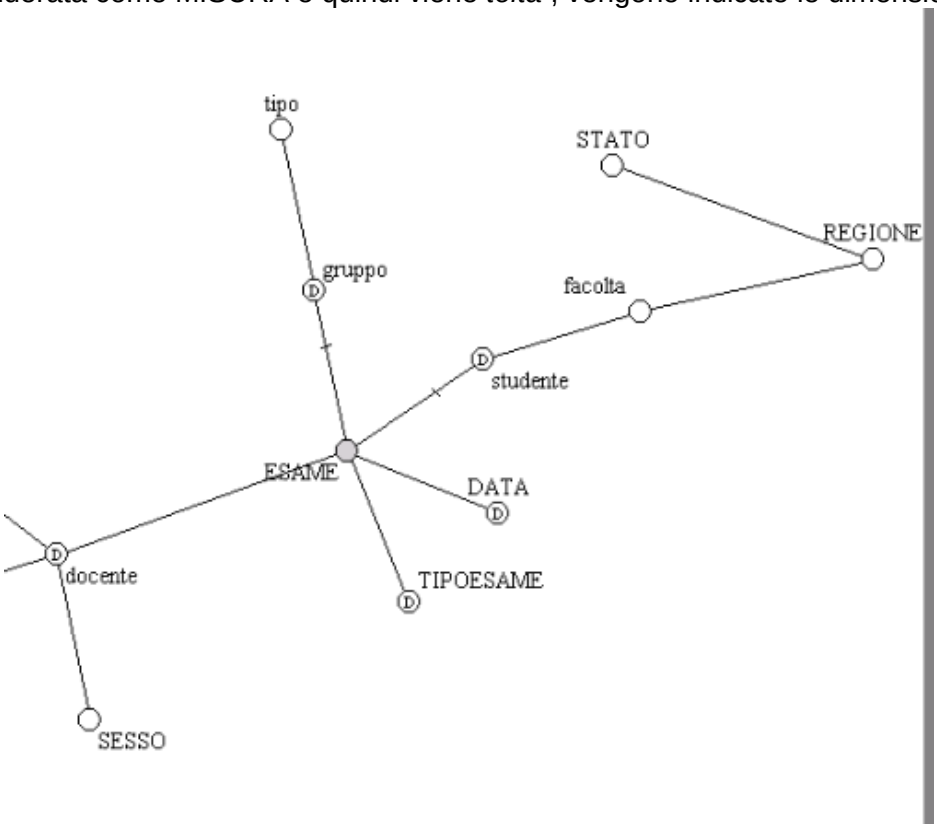
Le dimensioni **GRUPPO, STUDENTE**, **TIPOESAME** sono figli diretti della radice e quindi possono essere presi come dimensioni senza ulteriori elaborazioni; in particolare le due dimensioni **GRUPPO e STUDENTE** risultano opzionali

La dimensione DOCENTE non è figlio diretto della radice: questo significa che devo *togliere* qualcosa, cioè l'APPELLO: devo fare un innesto su APPELLO, portare DOCENTE come figlio della radice (il GENERE quindi scompare, non mi interessa)

La parte di albero modificata risulterà quindi:



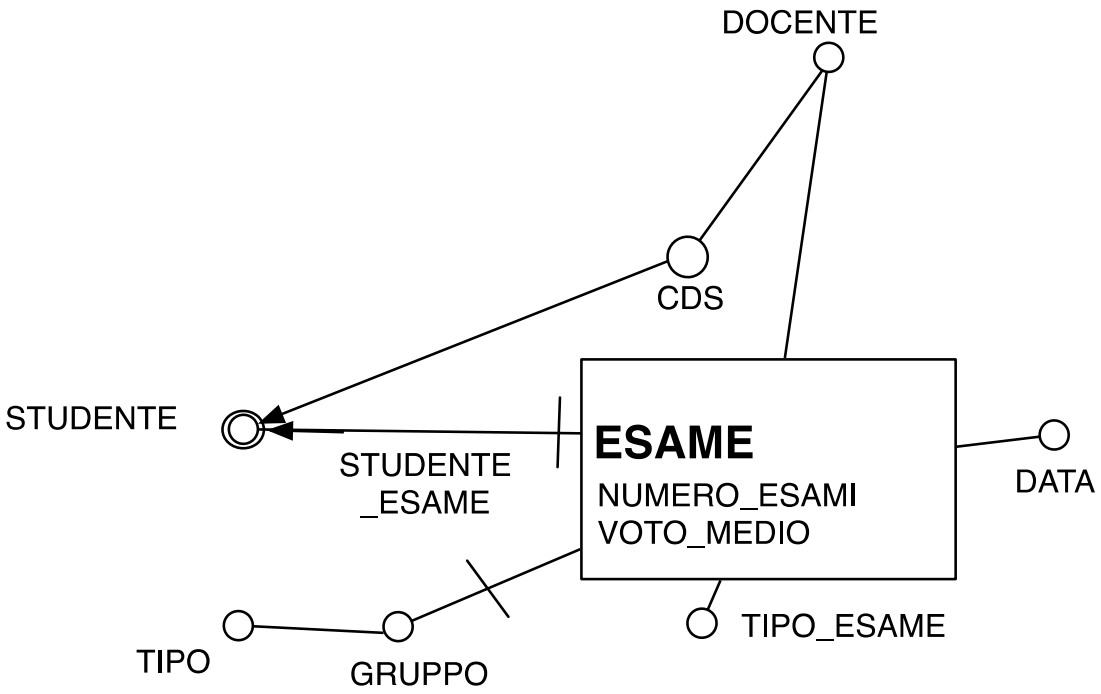
VOTO viene considerata come MISURA e quindi viene *tolta* ; vengono indicate le dimensioni



Schema di Fatto ESAME

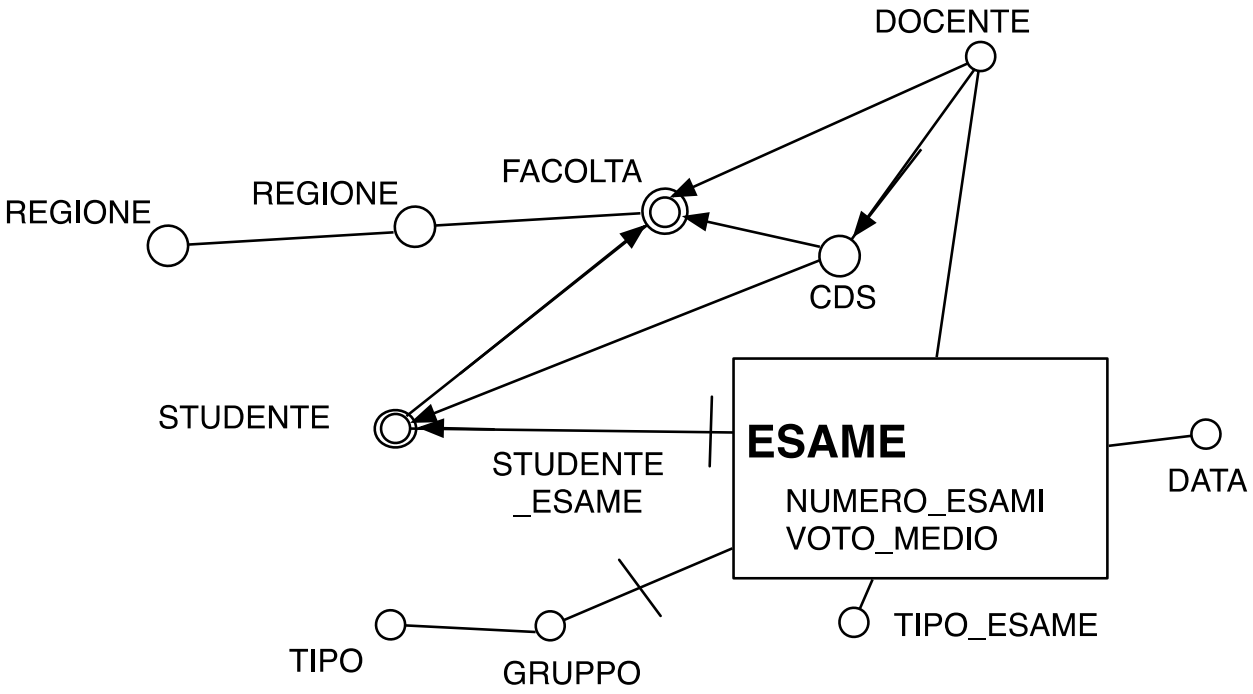
Si costruisce lo Schema di Fatto ESAME con le quattro dimensioni; vengono indicate anche le misure NUMERO ESAMI e VOTO ESAME MEDIO.

Per semplicità riportiamo prima la parte senza FACOLTA



Si noti condivisione su STUDENTE e la necessità di dare un nome al ruolo STUDENTE_ESAME : è appunto la dimensione STUDENTE richiesta nelle specifiche (in altre parole, visto che STUDENTE è condiviso la dimensione richiesta “STUDENTE dell’esame “ è stata chiamata **STUDENTE_ESAME**

Ora aggiungiamo FACOLTA ottenendo lo **Schema di Fatto** finale per **ESAME**



Controlliamo che ci siano tutte e quattro le facoltà richieste

1) la facoltà del docente dell'appello dell'esame
   DOCENTE.FACOLTA

2) la facoltà del CDS del docente dell'appello dell'esame
   DOCENTE.CDS.FACOLTA

3) la facoltà dello studente rappresentante del CDS
   DOCENTE.CDS.STUDENTE.FACOLTA

4) la facoltà dello studente (eventuale) che ha sostenuto l'esame
   STUDENTE_ESAME.FACOLTA

Lo schema di Fatto ESAME è **TEMPORALE.**

Il controllo è molto semplice

L'insieme delle dimensioni

{ GRUPPO, STUDENTE, **TIPOESAME**,**DATA, DOCENTE }**

contiene **almeno una** delle chiavi della relazione dello schema di partenza che ho scelto come fatto, cioè della relazione ESAME?

> **Se** la risposta è negativa (come in questo caso) allora lo schema di fatto è TEMPORALE altrimenti è TRANSAZIONALE

Dipendenze Funzionali tra le dimensioni.

Considerato che le dimensioni sono

GRUPPO, STUDENTE, **TIPOESAME**, **DATA, DOCENTE**

e considerato il significato dell' attributo **TIPO_ESAME** di ESAME (dato nelle specifiche):

> assume il valore STUD se è un esame di uno studente ed il valore GRUP se è un esame di un gruppo

una possibile dipendenza funzionale tra le dimensioni può essere GRUPPO → TIPO_ESAME: infatti se GRUPPO ha un certo valore diverso da NULL, significa che è un esame di gruppo e che quindi TIPOESAME vale GRUP, invece se GRUPPO ha valore NULL, significa che è un esame di uno studente e che quindi TIPOESAME vale STUD.
Per controllare GRUPPO → TIPO_ESAME si deve fare un join in quanto gli attributi sono su due tabelle, ESAME e ESAMEGRUPPO; il join deve essere un left join in quanto non tutti gli ESAME hanno un ESAMEGRUPPO associato (nella query usare i nomi dello schema, quindi TIPOESAME e non TIPO_ESAME)

```
SELECT GRUPPO
FROM ESAME  E LEFT JOIN ESAMEGRUPPO EG ON(E.NUMES = EG.NUMES)
GROUP BY  GRUPPO
HAVING  COUNT(DISTINCT TIPOESAME)>1
```

Il risultato è vuoto quindi la FD è valida!

> Stesso discorso per STUD → TIPO_ESAME.

In definitiva, FD tra le dimensioni:

GRUPPO → TIPO_ESAME
STUD → TIPO_ESAME

SECONDA CONSEGNA

Quello svolto finora è la **parte minima** indispensabile **per la prima consegna**

Le misure verranno considerate nella **seconda consegna**
Così come alcuni punti relativi alla progettazione concettuale con **COSTRUTTI AVANZATI** quali

1) Analisi delle Convergeneze
2) Attributi cross-dimensionali, archi opzionali e coperture

Questi aspetti vengono discussi nel seguito di questo capitolo.

Nella seconda consegna verranno anche considerati

A) Progettazione logica del Data Mart
B) Proge/azione dell'alimentazione del DM

1.6 ANALISI CONVERGENZE

Un altro aspetto importante da discutere sullo schema di fatto sono le **eventuali convergenze**
Cioè si devono considerare le varie condivisioni individuate e si deve discutere se sono invece delle convergenze

1) Ci sono dei casi il cui la risposta è immediata: la condivisione su STUDENTE non è (non ha senso che sia) una convergenza, altrimenti andrei a considerare solo gli esami degli studenti che sono rappresentanti nel CDL del docente che ha fatto l'esame …

2) In altri casi si devono fare delle indagini sui datia disposizione, come nel caso di FACOLTA

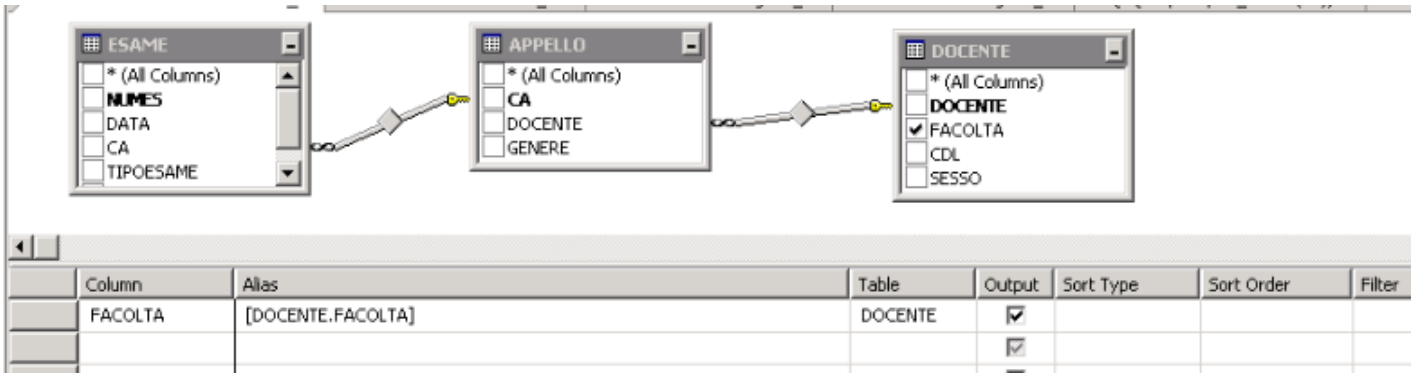
Si considerano le coppie di FACOLTAControlliamo che ci siano tutte e quattro le facoltà richieste

1) la facoltà del docente dell'appello dell'esame DOCENTE.FACOLTA
2) la facoltà del CDS del docente dell'appello dell'esame DOCENTE.CDS.FACOLTA

Convergono ??
Effettuo il controllo sui dati tramite una query che mi restituisce queste due facolta:
Sarà una query con vari join quindi la costruisco tramite l'interfaccia delle viste

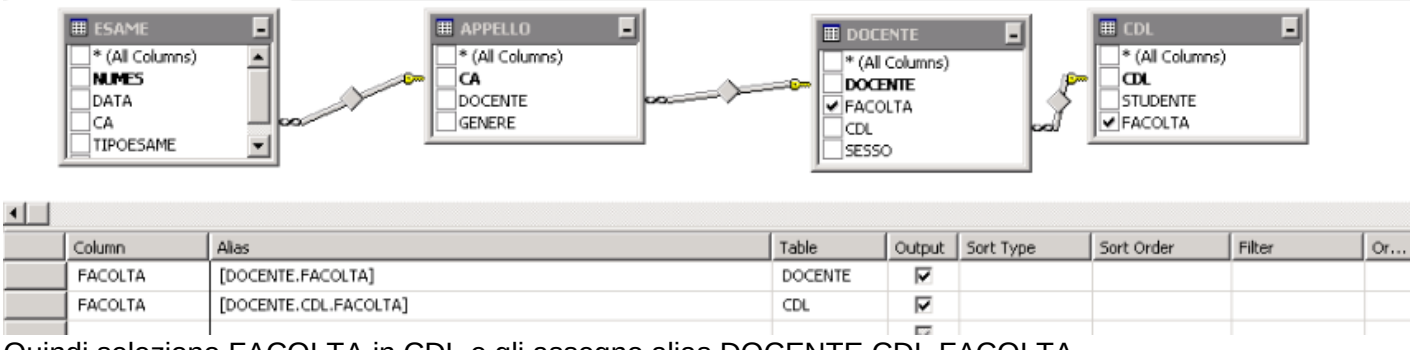
Guardo il Diagramma relazionale per capire dove sono queste due FACOLTA

1) la facoltà del docente dell'appello dell'esame DOCENTE.FACOLTA
è in DOCENTE associato all'APPELLO dell'ESAME



Quindi seleziono FACOLTA e gli assegno alias DOCENTE.FACOLTA

2) la facoltà del CDS del docente dell'appello dell'esame DOCENTE.CDS.FACOLTA:
è in CDL (ricordiamo che CDS è sinonimo di CDL) del DOCENTE associato all'APPELLO dell'ESAME : è importante notare che è sempre lo stesso DOCENTE, quindi non devo aggiungere un'altra volta la tabella DOCENTE, ma devo aggiungere solo la tabella CDL



Quindi seleziono FACOLTA in CDL e gli assegno alias DOCENTE.CDL.FACOLTA

Eseguo query e vedo che sono diverse

	DOCENTE.FACOLTA	DOCENTE.CDL.FACOLTA
	MUSICA	LETTERE
	LETTERE	MUSICA
	LETTERE	MUSICA
	MUSICA	LETTERE
	LETTERE	MUSICA
	LETTERE	MUSICA

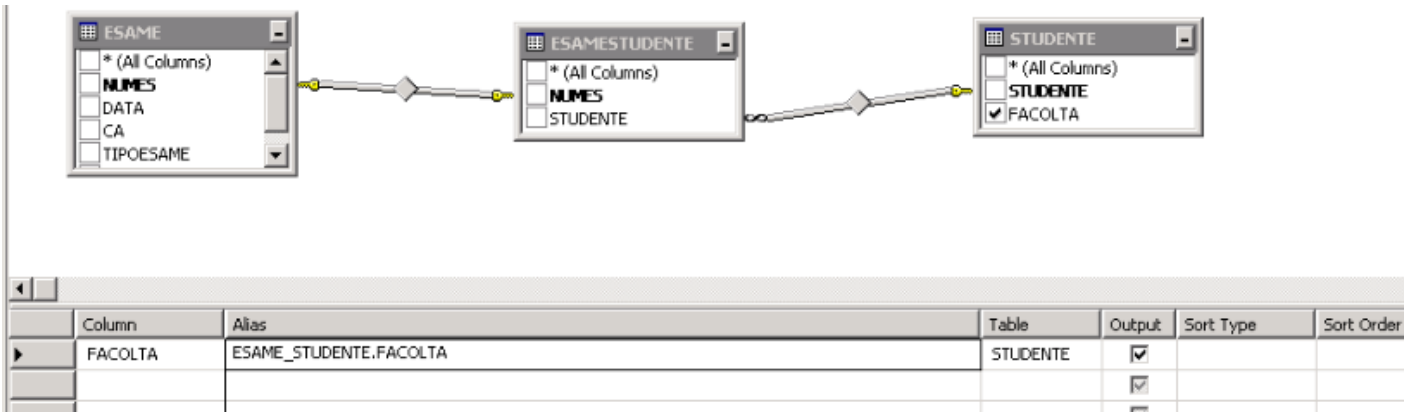
Per fare il controllo effettivo, alla query generata aggiungo la condizione WHERE

```
WHERE  dbo.DOCENTE.FACOLTA <> dbo.CDL.FACOLTA
```

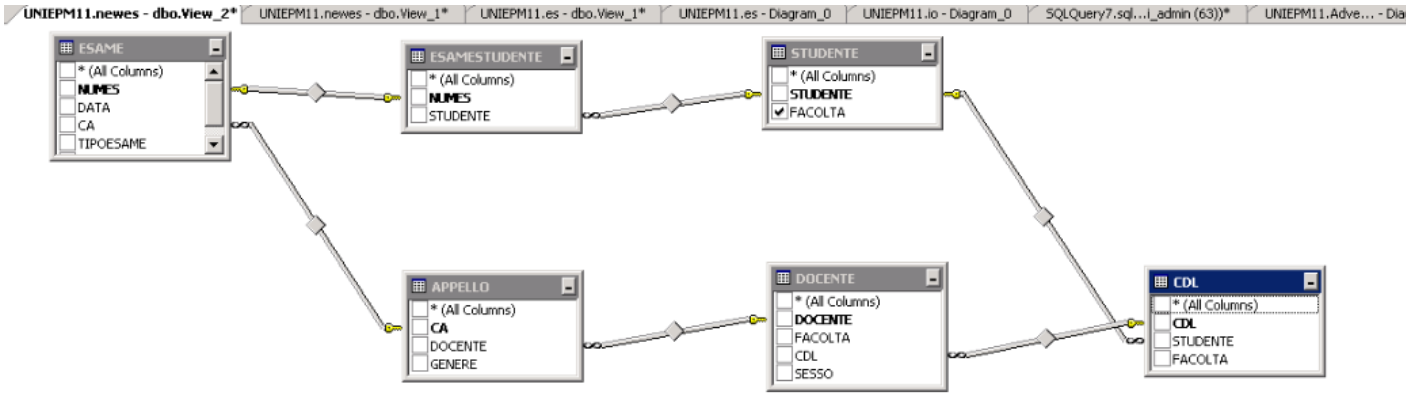
Eseguo la query: se il **risultato è vuoto** allora c'è **convergenza**, altrimenti resta condivisione

Stessa cosa per le altre coppie (questo è un caso **molto, molto** particolare …)

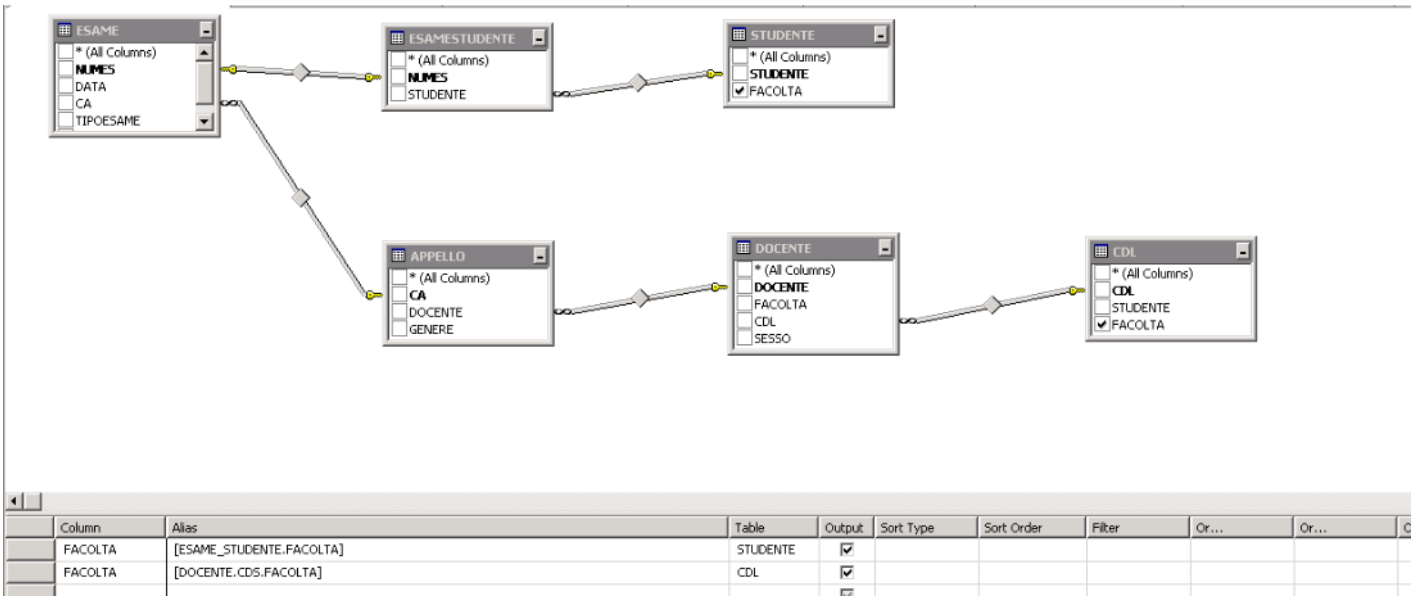
1) la facoltà dello studente (eventuale) che ha sostenuto l'esame **STUDENTE_ESAME.FACOLTA**



2) la facoltà del CDS del docente dell'appello dell'esame DOCENTE.CDS.FACOLTA:



IMPORTANTE: il join automaticamente aggiunto dal sistema tra STUDENTE e CDL deve ESSERE ELIMINATO (altrimenti andrei a consierare solo il caso degli esami degli studenti che sono rappresentanti nel CDL del docente che ha fatto l'esame …)



e procedo come in precedenza.

Concludiamo con il caso più difficile, ma che si presenta spesso in pratica …

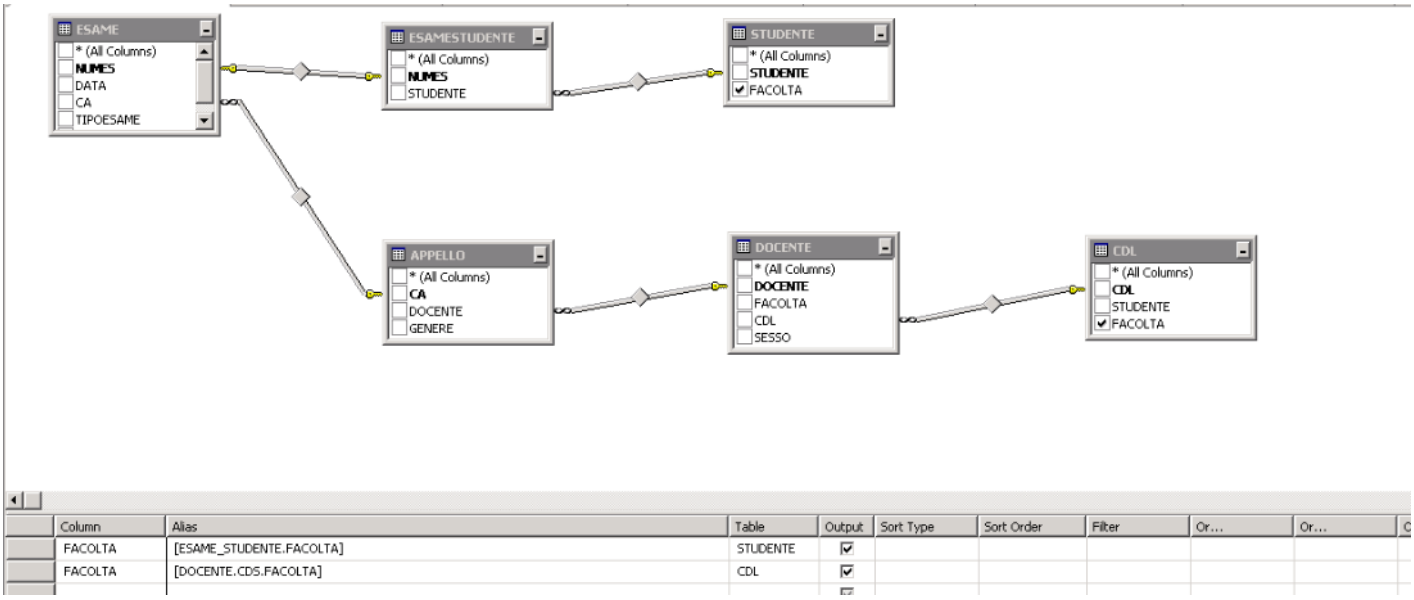
1) la facoltà dello ***studente*** (eventuale) che ha sostenuto l'esame**STUDENTE_ESAME.FACOLTA**

2) la facoltà dello ***studente*** rappresentante del CDS **DOCENTE.CDS.STUDENTE.FACOLTA**

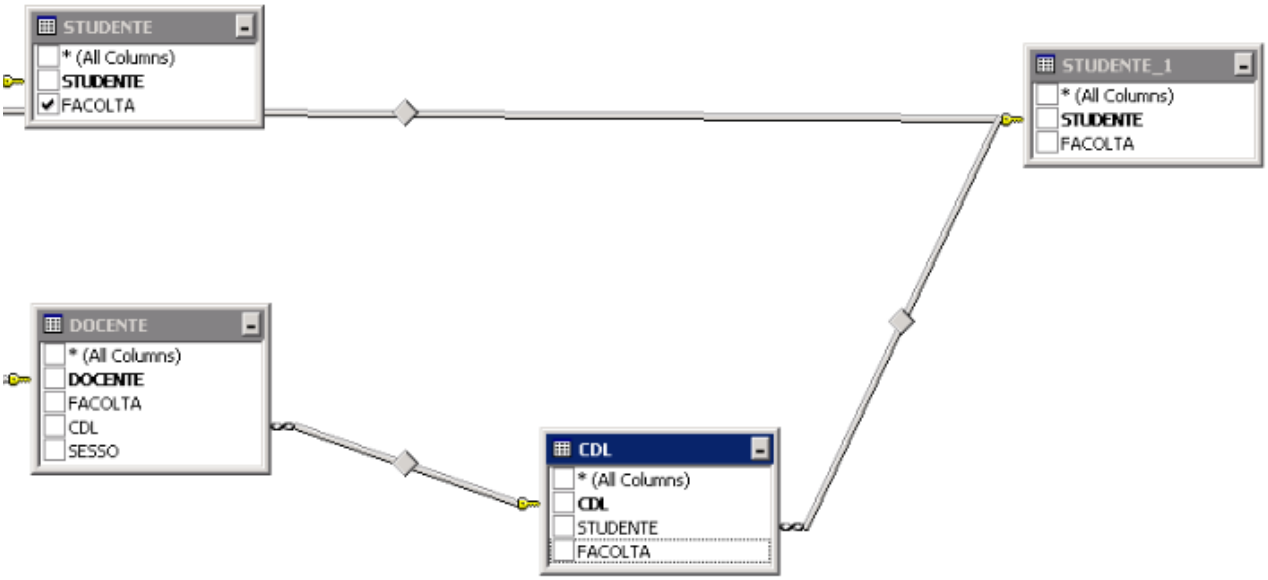
IMPORTANTE:

ho **due studenti**, che abbiamo detto **non necessariamente coincidenti**
QUINDI
Devo mettere nella query **due volte** la tabella **STUDENTE**

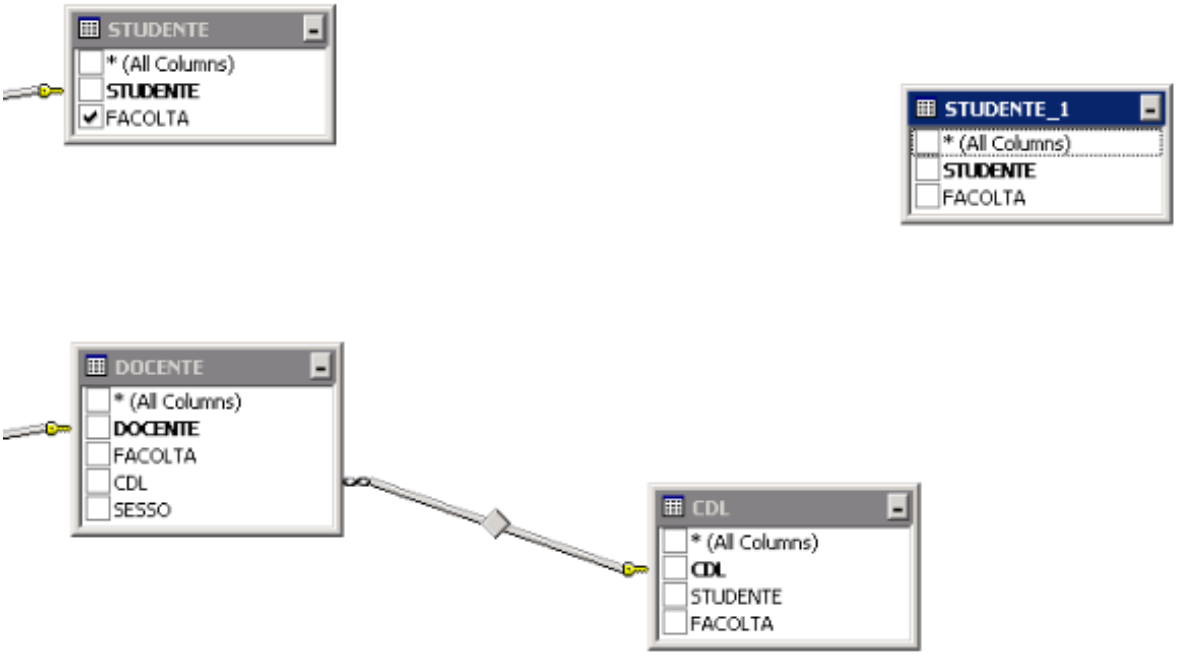
Si parte dalla precedente query, de-selezionando DOCENTE.CDS.FACOLTA



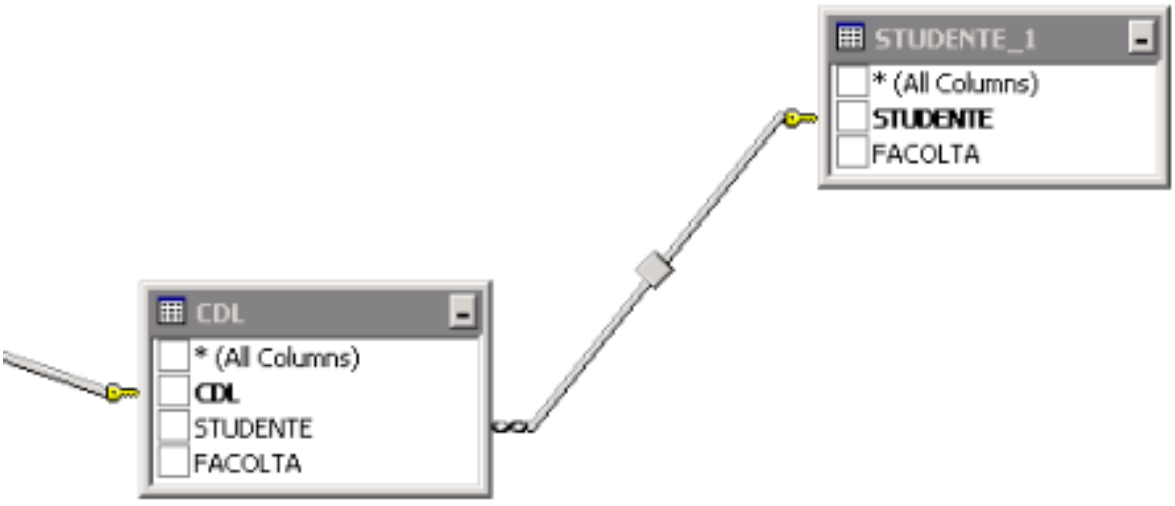
Si aggiunge un'altra tabella STUDENTE (il sistema gli assegna l'alias STUDENTE_1)



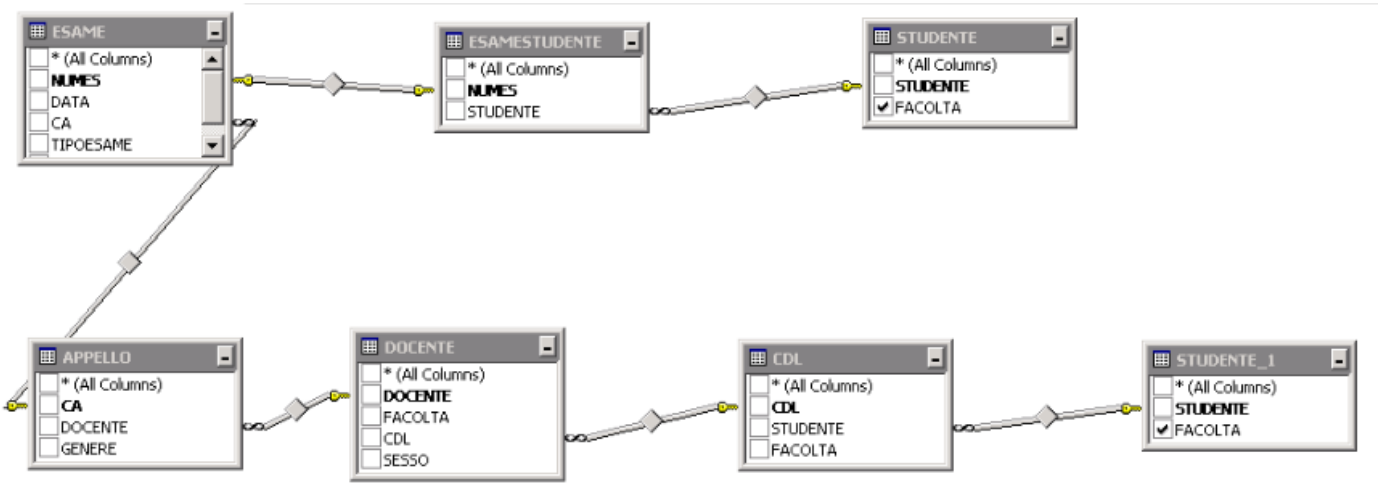
tolgo tutti i collegamenti automaticamente generati con STUDENTE 1



e definisco quello che mi interessa, cioè tra CDL e STUDENTE



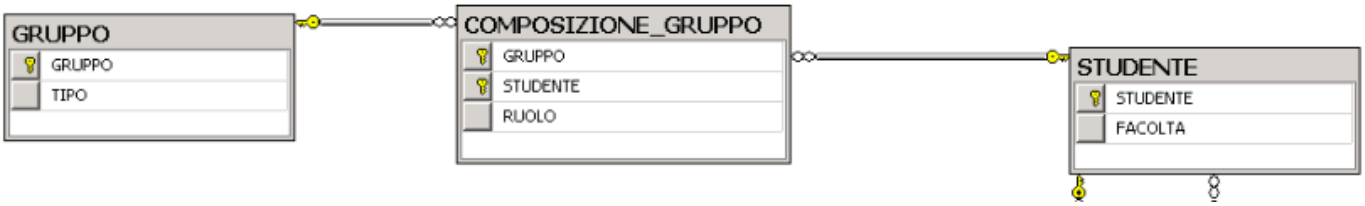
seleziono quindi la FACOLTA di STUDeNTE_1 che sarà quindi **DOCENTE.CDS.STUDENTE.FACOLTA**



Column	Alias	Table	Output	Sort Type	Sort Order	Filter
FACOLTA	[ESAME_STUDENTE.FACOLTA]	STUDENTE	☑			
FACOLTA	[DOCENTE.CDS.STUDENTE.FACOLTA]	STUDENTE_1	☑			

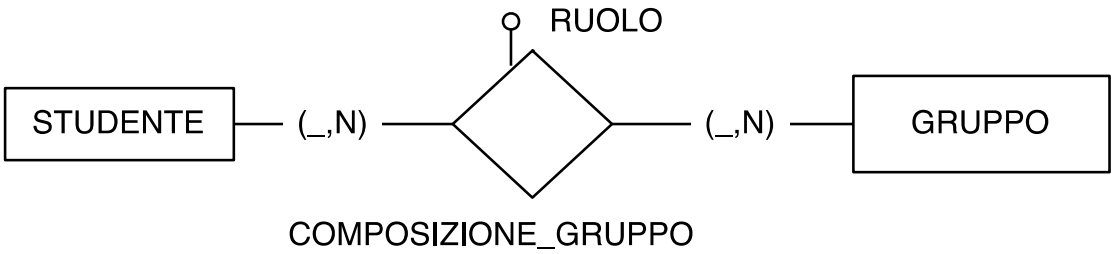
e procedo come in precedenza

Si considera anche COMPOSIZIONE_GRUPPO



COMPOSIZIONE_GRUPPO(GRUPPO:GRUPPO, STUDENTE:STUDENTE, RUOLO)

In E_R



NOTE:

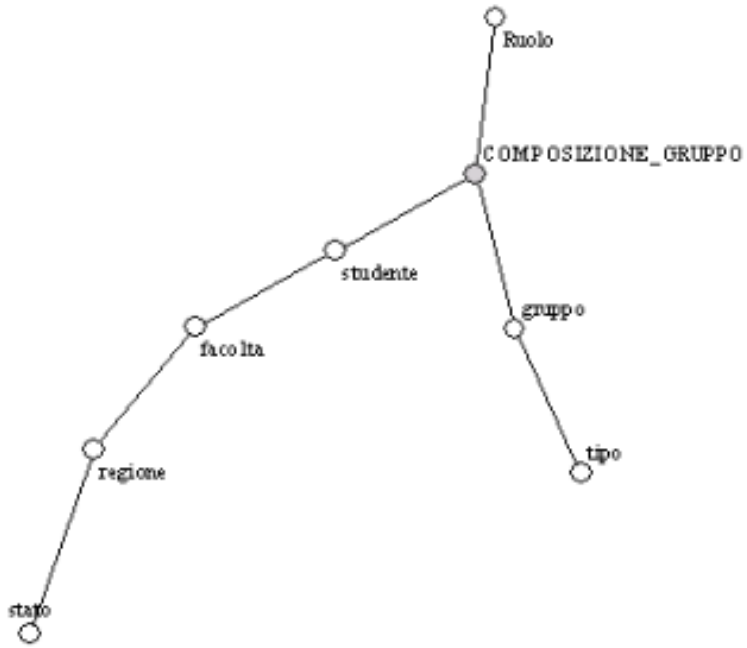
1) L'ER può essere disegnato *a pezzi*

2) Quando si ripete una entità già fatta in precedenza (STUDENTE e GRUPPO) non si deve ripetere tutto, basta il loro nome …

FATTO: COMPOSIZIONE_GRUPPO

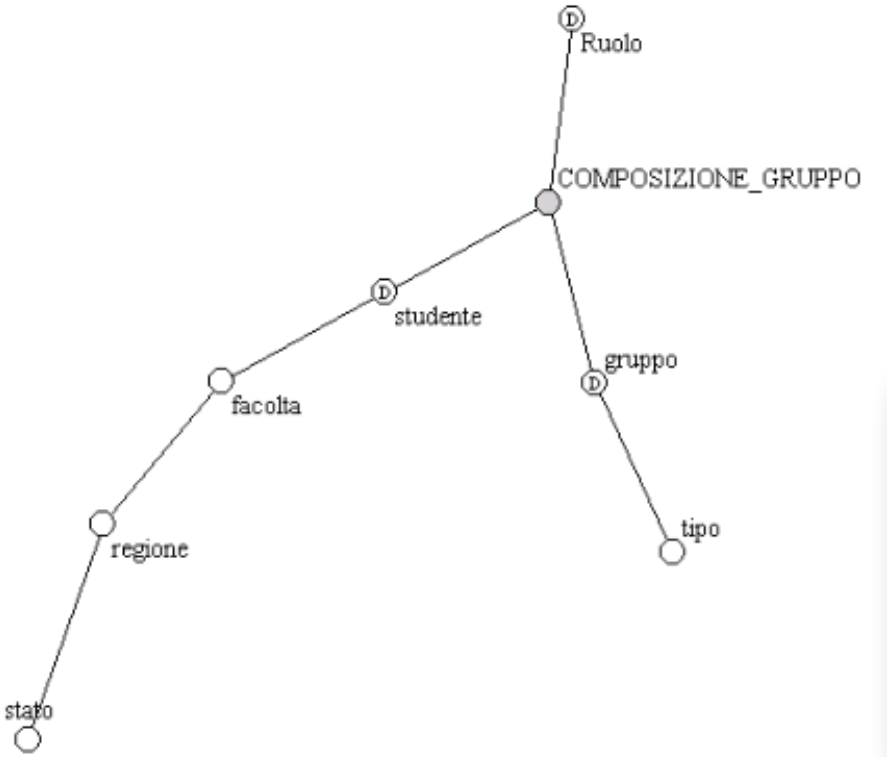
1) Si costruisce l'albero degli attributi come spiegato nelle dispense
   http://www.dbgroup.unimo.it/SIA/SIA_2015_ProgettazioneConcettuale.pdf
   (in particolare vedere attentamente l'esempio delle vendite che inizia a pag. 15)

2) Editing dell'albero: SI AGGIUNGONO LE FD individuate (e non riportate nello schema E/R). Nel nostro esempio la FD in FACOLTA : REGIONE → STATO

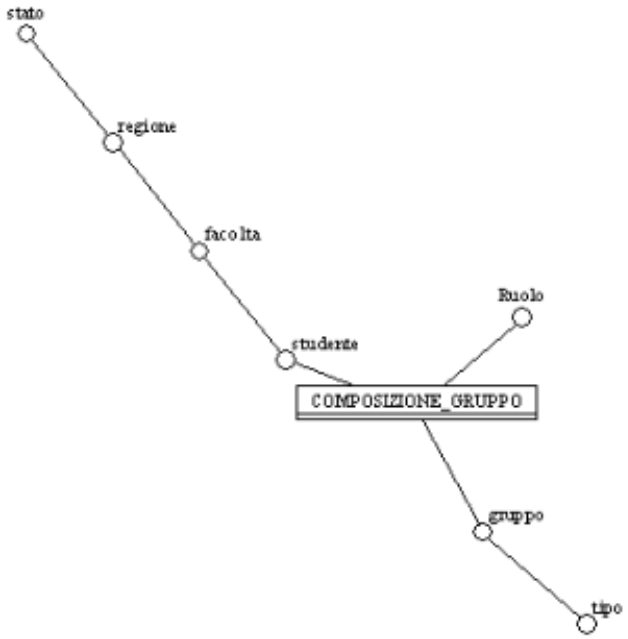


Si decide di non fare semplificazioni (innesti e potature)

DIMENSIONI: STUDENTE, GRUPPO e RUOLO



SI crea lo Schema di Fatto



1.7 COPERTURA DI ARCHI OPZIONALI

Nello schema ER era stato supposto che che la gerarchia su ESAME sia totale/esclusiva.
Per verificarlo sui dati, si considerano le tabelle

```
ESAME(NUMES, APPELLO:APPELLO, DATA, TIPO_ESAME, VOTO)
ESAMEGRUPPO(NUMES:ESAME, GRUPPO:GRUPPO)
ESAMESTUDENTE(NUMES:ESAME, STUDENTE:STUDENTE)
```

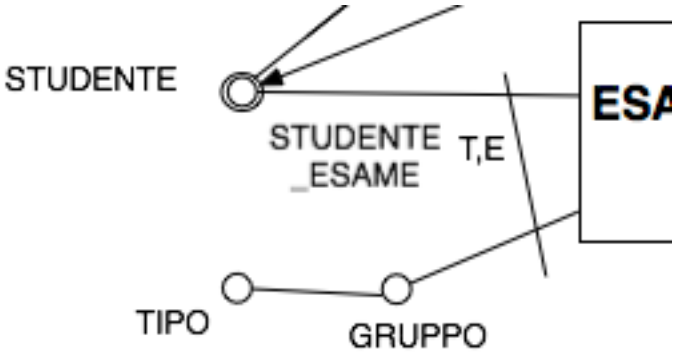
Esclusiva: non ci sono ESAMEGRUPPO che siano anche ESAMESTUDENTE, e viceversa, in quanto la seguente query è vuota

```
SELECT *
FROM ESAMEGRUPPO  EG JOIN ESAMESTUDENTE ES ON(ES.NUMES = EG.NUMES)
```

Totale: non ci sono ESAME che **NON** siano anche ESAMESTUDENTE oppure ESAMEGRUPPO, in quanto la seguente query è vuota

```
SELECT *
FROM ESAME
WHERE NUMES NOT IN (SELECT NUMES FROM ESAMEGRUPPO
                    UNION
                    SELECT NUMES FROM ESAMESTUDENTE)
```

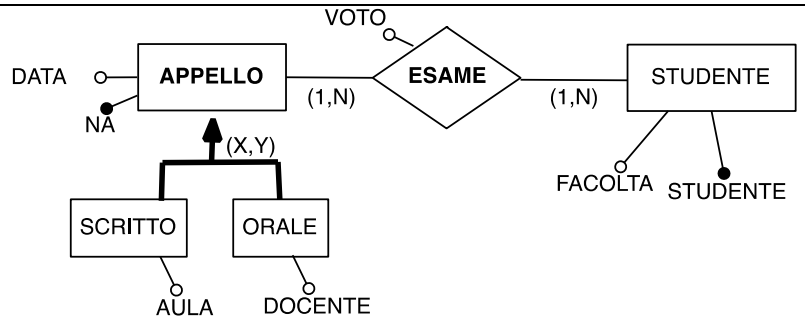
Nello schema di fatto si indica che tra dimensione GRUPPO e la dimensione STUDENTE_ESAME c'è una **Copertura di Arco Opzionale** di tipo Totale Esclusiva: cioè le due dimensioni sono opzionali, non posso avere sia lo STUDENTE_ESAME che il gruppo (Esclusiva) ed inoltre un ESAME avrà necessariamente o uno STUDENTE_ESAME o un GRUPPO



1.8 ESEMPIO COMPLETO DI COPERTURA DI ARCHI OPZIONALI

Esempio completo per illustrare e discutere la Copertura di un arco opzionale (PAG. 24 dei lucidi MODELLAZIONE CONCETTUALE); vengono considerate anche l'alimentazione, l'aggregabilità delle misure.

A tale scopo si considera un DBO relativo agli ESAMI con il relativo schema ER contenente una *gerarchia di generalizzazione* sull'entità APPELLO (da non confondere con il concetto di gerarchia di una dimensione del modello DFM!) e si effettua la progettazione concettuale e logica e l'alimentazione.

Schema ER del DBO	**Schema relazionale del DBO**
	STUDENTE(<u>STUDENTE</u>,FACOLTA) APPELLO(<u>NA,</u>DATA) SCRITTO(<u>NA:APPELLO</u>,AULA) ORALE(<u>NA:APPELLO</u>,DOCENTE) ESAME(<u>NA:APPELLO</u>, <u>STUDENTE:STUDENTE</u>, VOTO)

Requisiti del DW

Fatto: ESAME

Dimensioni : APPELLO, DATA e FACOLTA

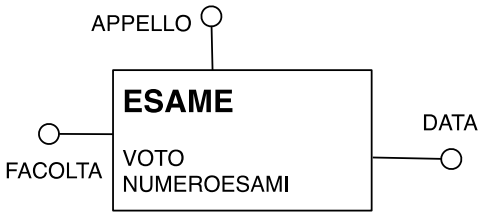
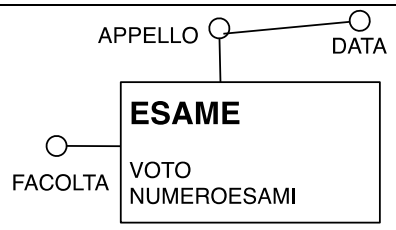
Misure: NUMEROESAMI, VOTO

1.8.1 Soluzione: Progettazione Concettuale

E' evidente dallo schema del DBO che esiste una ***dipendenza funzionale tra le dimensioni***:

APPELLO → DATA

Si hanno due possibilità, *equivalenti* tra di loro:

Dimensioni : **APPELLO**, **DATA** e **FACOLTA** Con FD: APPELLO → DATA	Dimensioni : APPELLO e FACOLTA
	

Questi due schemi di fatto sono tra loro equivalenti. La scelta di quale usare, cioè se tenere DATA come dimensione oppure metterla nella gerarchia di APPELLO, verrà fatta nel seguito durante la progettazione logica e realizzazione OLAP.

In questo esercizio, teniamo DATA come dimensione, motivando la scelta in base al fatto che era stata esplicitamente richiesta come tale nei requisiti.

Consideriamo ora l'aspetto principale di questo esempio, cioè la ***gerarchia di generalizzazione*** sull'entità APPELLO (schema ER).

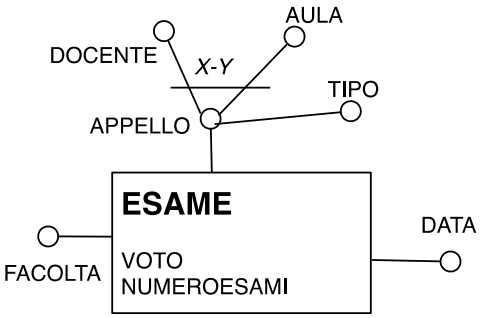
Quali attributi dimensionali inserire nella ***gerarchia della dimensione*** APPELLO (schema DFM)?

Un primo attributo è il **TIPO** dell'APPELLO, i cui valori dipendono dal tipo di copertura della *gerarchia di generalizzazione* sull'entità APPELLO, valori scelti dal progettista.

TIPO COPERTURA *X,Y*	**Valori di TIPO**	**Dipendenze Funzionali** tra AULA,DOCENTE,TIPO
T,E (totale esclusiva)	'Scritto','Orale'	AULA → TIPO DOCENTE → TIPO
T,S (totale sovrapposta)	'Scritto','Orale', 'Scritto-Orale'	{AULA,DOCENTE} → TIPO
P,E (parziale esclusiva)	'Scritto','Orale', 'Progetto'	{AULA,DOCENTE} → TIPO
P,S (parziale sovrapposta.)	'Scritto','Orale', 'Scritto-Orale', 'Progetto'	{AULA,DOCENTE} → TIPO

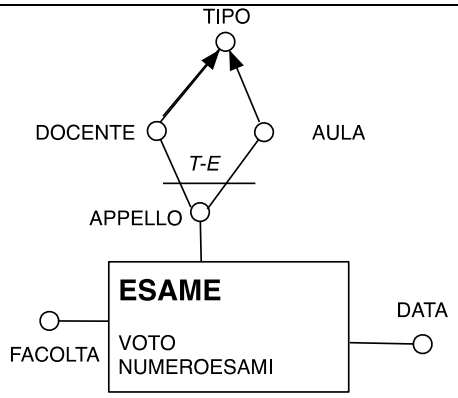
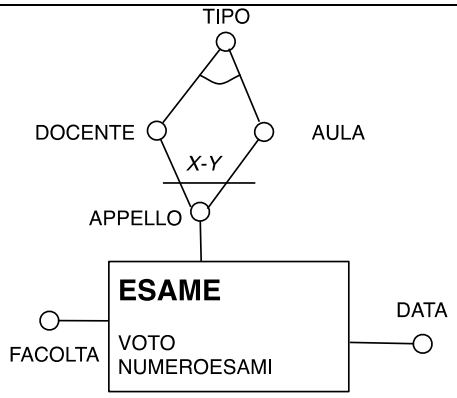
Quindi APPELLO viene completata con l'informazione su DOCENTE e AULA: ogni APPELLO che è uno scritto ha un DOCENTE, quindi ogni APPELLO può avere un docente ➔ l'***arco opzionale*** tra APPELLO e DOCENTE. Ricordiamo che un APPELLO che non ha associato il DOCENTE, viene associato ad un valore particolare di DOCENTE, diciamo 'NO_DOCENTE'. Stesso discorso per AULA.

Oltre al simbolo di opzionalità, nello schema DFM si indica anche la copertura, ottenendo quindi



E' ovvio che il valore del TIPO *dipende* dal fatto che l'appello sia uno scritto (e quindi con un'aula) e/o un orale (e quindi con un docente): di conseguenza TIPO è determinato funzionalmente da AULA e DOCENTE, come riportato nella terza colonna della tabella.

Queste dipendenze possono essere esplicitate/dichiarate sullo schema di fatto:

AULA → TIPO e DOCENTE → TIPO (**convergenza su TIPO**)	{AULA,DOCENTE} → TIPO **attributo cross-dimensionale**
	

1.8.2 Considerazioni preliminari per le query di alimentazione

Prima di passare alla progettazione logica e l'alimentazione facciamo alcune considerazioni preliminari.

Consideriamo lo schema relazionale del DBO relativo ad APPELLO

```
APPELLO(NA,DATA)
SCRITTO(NA:APPELLO,AULA)
ORALE(NA:APPELLO,DOCENTE)
```

Creiamo tale schema su un DB relazionale con il seguente script

```
CREATE TABLE APPELLO(NA INT PRIMARY KEY, DATA DATE)
CREATE TABLE SCRITTO(NA INT PRIMARY KEY REFERENCES APPELLO, AULA VARCHAR(12))
CREATE TABLE ORALE(NA INT PRIMARY KEY REFERENCES APPELLO, DOCENTE VARCHAR(12))
```

Quindi, supponendo di essere nel caso più generale di gerarchia P,S (parziale sovrapposta) inseriamo esattamente quattro appelli

```
INSERT INTO APPELLO  VALUES (1,'10-14-2013')
INSERT INTO APPELLO  VALUES (2,'10-14-2013')
INSERT INTO APPELLO  VALUES (3,'10-14-2013')
INSERT INTO APPELLO  VALUES (4,'10-14-2013')
```

Ciascuno relativo ad una delle *quattro tipologie di appelli*

```
Appello 1 è  un 'Progetto'
```

Quindi non viene inserito né in ORALE né in SCRITTO:

```
Appello 2 è  uno 'Scritto-Orale',
```

Quindi viene inserito sia in ORALE che SCRITTO:

```
        INSERT INTO ORALE VALUES (2,'DOC_A')
        INSERT INTO SCRITTO   VALUES (2,'AULA_5')
Appello 3 è  uno 'Scritto',
```

Quindi viene inserito in SCRITTO:

```
        INSERT INTO SCRITTO   VALUES (3, 'AULA_6')
Appello 4 è  un 'Orale', ,
```

Quindi viene inserito in ORALE:

```
        INSERT INTO ORALE VALUES (4,'DOC_A')
```

Quindi il DBO risulta essere

APPELLO

NA	DATA
1	2013-10-14
2	2013-10-14
3	2013-10-14
4	2013-10-14

SCRITTO

NA	AULA
2	AULA_5
3	AULA_6

ORALE

NA	DOCENTE
2	DOC_A
4	DOC_A

Per semplificare i calcoli conviene definire una vista **APPELLI_SCRITTI_ORALI_TIPO** con tutte tutte le informazioni relative ad appello, cioè che contenga già un attributo TIPO opportunamente calcolato e in essa siano già codificati i valori 'NO_DOCENTE' e 'NO_AULA'; cioè vogliamo ottenere:

NA	DATA	DOCENTE	AULA	TIPO
1	2013-10-14	NO_DOCENTE	NO_AULA	PROGETTO
2	2013-10-14	DOC_A	AULA_5	SCRITTO_ORALE
3	2013-10-14	NO_DOCENTE	AULA_6	SCRITTO
4	2013-10-14	DOC_A	NO_AULA	ORALE

SOLUZIONE

La soluzione *è ispirata* alla dipendenza funzionale {AULA,DOCENTE} → TIPO :

Il valore di TIPO viene stabilito in base ai valori di AULA e DOCENTE.

Per prima cosa si codificano i valori 'NO_DOCENTE' e 'NO_AULA'

Codifichiamo 'NO_DOCENTE'

```
CREATE  VIEW  APPELLI_ORALI AS
      SELECT A.*, DOCENTE=COALESCE(DOCENTE,'NO_DOCENTE')
      FROM APPELLO A LEFT JOIN ORALE O ON (A.NA=O.NA)
```

Codifichiamo 'NO_AULA'

```
CREATE VIEW  APPELLI_SCRITTI_ORALI AS
      SELECT A.*, AULA=COALESCE(AULA,'NO_AULA')
      FROM APPELLI_ORALI A LEFT JOIN SCRITTO S ON (A.NA=S.NA)
```

Quindi calcoliamo il valore di TIPO in base ai valori di AULA e DOCENTE:

```
CREATE VIEW  APPELLI_SCRITTI_ORALI_TIPO AS
SELECT A.*,
      TIPO=CASE
            WHEN DOCENTE='NO_DOCENTE' AND AULA='NO_AULA' THEN 'PROGETTO'
            WHEN DOCENTE='NO_DOCENTE' AND AULA<>'NO_AULA' THEN 'SCRITTO'
            WHEN DOCENTE<>'NO_DOCENTE' AND AULA='NO_AULA' THEN 'ORALE'
            ELSE 'SCRITTO_ORALE'
      END
FROM APPELLI_SCRITTI_ORALI A
```

Questo modo di procedere funziona se uno scritto ha necessariamente un'aula e un orale ha necessariamente un docente, ovvero se nello schema era stato dichiarato esplicitamente il NOT NULL

```
CREATE TABLE SCRITTO(... AULA VARCHAR(12) NOT NULL)
CREATE TABLE ORALE(...  DOCENTE VARCHAR(12) NOT NULL)
```

Consideriamo il caso in cui AULA (DOCENTE) possa assumere un valore NULL:

APPELLO

NA	DATA
1	2013-10-14
2	2013-10-14
3	2013-10-14
4	2013-10-14
5	2013-10-15
6	2013-10-15
7	2013-10-16

SCRITTO

NA	AULA
2	AULA_5
3	AULA_6
6	NULL
7	NULL

ORALE

NA	DOCENTE
2	DOC_A
4	DOC_A
5	NULL
7	NULL

Per inserire questi tre nuovi appelli

```
INSERT INTO APPELLO  VALUES (5,'10-15-2013')
INSERT INTO APPELLO  VALUES (6,'10-15-2013')
INSERT INTO APPELLO  VALUES (7,'10-16-2013')


INSERT INTO ORALE VALUES (5,NULL)
INSERT INTO SCRITTO VALUES (6,NULL)
INSERT INTO ORALE VALUES (7,NULL)
INSERT INTO SCRITTO VALUES (7,NULL)
```

Si deve decidere come *codificare* questi due casi:
l'appello 5 è codificato come TIPO='ORALE' con il nuovo valore DOCENTE=**'DOCENTE_INDEFINITO'**;

l'appello 6 è codificato come TIPO='SCRITTO' con il nuovo valore AULA=**'AULA_INDEFINITA'**;

Quindi la vista **APPELLI_SCRITTI_ORALI_TIPO** si deve modificare per restituire il seguente risultato:

NA	DATA	DOCENTE	AULA	TIPO
1	2013-10-14	NO_DOCENTE	NO_AULA	PROGETTO
2	2013-10-14	DOC_A	AULA_5	SCRITTO_ORALE
3	2013-10-14	NO_DOCENTE	AULA_6	SCRITTO
4	2013-10-14	DOC_A	NO_AULA	ORALE
5	2013-10-15	DOC_INDEFINITO	NO_AULA	ORALE
6	2013-10-15	NO_DOCENTE	AULA_INDEFINITA	SCRITTO
7	2013-10-16	DOC_INDEFINITO	AULA_INDEFINITA	SCRITTO_ORALE

SOLUZIONE

Consideriamo

```
CREATE  VIEW  APPELLI_ORALI AS
      SELECT A.*, DOCENTE=COALESCE(DOCENTE,'NO_DOCENTE')
      FROM APPELLO A LEFT JOIN ORALE O ON (A.NA=O.NA)
```

Non *funziona più* in quanto restituisce 'NO_DOCENTE' anche per l'appello 5!!

Allora viene modificata (tramite l'istruzione ALTER VIEW) come segue

```
ALTER VIEW  APPELLI_ORALI AS
      SELECT A.*,
            DOCENTE=CASE    WHEN O.NA IS NULL  THEN 'NO_DOCENTE'
                            WHEN DOCENTE IS NULL THEN 'DOC_INDEFINITO'
                            ELSE DOCENTE
                        END
      FROM APPELLO A LEFT JOIN ORALE O ON (A.NA=O.NA)
```

Stesso discorso per l'altra vista `APPELLI_SCRITTI_ORALI:`

```
ALTER VIEW  APPELLI_SCRITTI_ORALI AS
      SELECT A.*,
            AULA=CASE  WHEN  S.NA IS NULL  THEN 'NO_AULA'
                       WHEN AULA IS NULL THEN 'AULA_INDEFINITA'
                       ELSE AULA
                  END
      FROM APPELLI_ORALI A LEFT JOIN SCRITTO S ON (A.NA=S.NA)
```

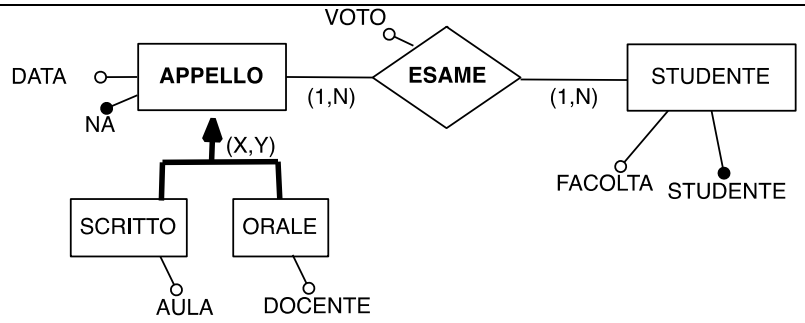
Quindi si modifica la vista **APPELLI_SCRITTI_ORALI_TIPO**

(si noti che la dipendenza funzionale {AULA,DOCENTE} → TIPO resta valida)

```
ALTER VIEW  APPELLI_SCRITTI_ORALI_TIPO AS
SELECT A.*,
      TIPO=CASE
            WHEN DOCENTE='NO_DOCENTE' AND AULA='NO_AULA' THEN 'PROGETTO'
            WHEN (AULA<>'NO_AULA' OR AULA='AULA_INDEFINITA')
                  AND DOCENTE='NO_DOCENTE' THEN 'SCRITTO'
            WHEN (DOCENTE<>'NO_DOCENTE' OR DOCENTE='DOC_INDEFINITO')
                  AND AULA='NO_AULA' THEN 'ORALE'
            ELSE 'SCRITTO_ORALE'
      END
FROM APPELLI_SCRITTI_ORALI A
```

1.8.3 Progettazione Logica e Alimentazione

Viene completato il precedente esercizio, considerando anche la progettazione logica e l'alimentazione; inoltre vengono aggiunte altre **due misure**:

Schema ER del DBO	**Schema relazionale del DBO**
	STUDENTE(<u>STUDENTE</u>,FACOLTA) APPELLO(<u>NA,</u>DATA) SCRITTO(<u>NA:APPELLO</u>,AULA) ORALE(<u>NA:APPELLO</u>,DOCENTE) ESAME(<u>NA:APPELLO</u>, <u>STUDENTE:STUDENTE</u>, VOTO)

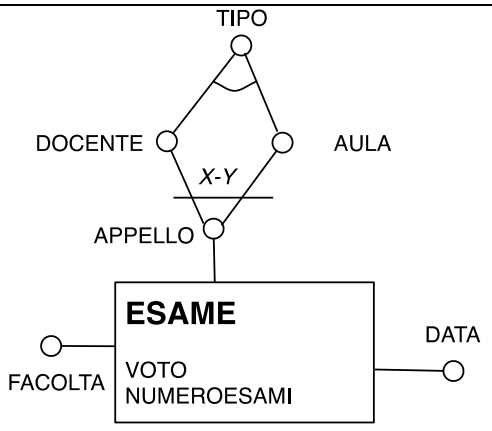
Requisiti del DW

Fatto: ESAME

Dimensioni : APPELLO, DATA e FACOLTA

Misure: NUMEROESAMI, VOTO, NUMERO_STUDENTI, NUMERO_APPELLI_CON_AULA

TIPO COPERTURA *X,Y*	**Valori di TIPO**	**Dipendenze Funzionali** tra AULA,DOCENTE,TIPO
P,S (parziale sovrapposta.)	'Scritto','Orale', 'Scritto-Orale', 'Progetto'	{AULA,DOCENTE} → TIPO

{AULA,DOCENTE} → TIPO **attributo cross-dimensionale**	

Schema di fatto **temporale**, con dipendenza tra le dimensioni APPELLO → DATA

GLOSSARIO DELLE MISURE

1. NUMERO_ESAMI= **COUNT**(*)
2. VOTO_MEDIO = **AVG** (VOTO)
3. NUMERO_STUDENTI = **COUNT**(**DISTINCT** STUDENTE)
4. NUMERO_APPELLI_CON_AULA

A differenza delle altre misure, NUMERO_APPELLI_CON_AULA non si riesce a calcolare *facilmente* e *direttamente* sul DBO attraverso un operatore classico di aggregazione SQL.

Per definirne il significato consideriamo l'istanza del DBO:

APPELLO

NA	DATA
1	2013-10-14
2	2013-10-14
3	2013-10-14
4	2013-10-14
5	2013-10-15
6	2013-10-15
7	2013-10-16

SCRITTO

NA	AULA
2	AULA_5
3	AULA_6
6	NULL
7	NULL

ORALE

NA	DOCENTE
2	DOC_A
4	DOC_A
5	NULL
7	NULL

Un modo semplice di procedere è quello di definire una variabile booleana CON_AULA_S_N che associ ad ogni appello il valore 1 (appello con aula) oppure il valore 0: questo verrà fatto definendo una vista

APPELLO_CON_AULA(NA, CON_AULA_S_N)

Occorre innanzitutto stabilire se gli appelli 6 e 7 devono essere contati come APPELLI_CON_AULA. Decidiamo di **non considerarli**.

Supponendo di aver già creato le viste della soluzione di pagina 6, ed in particolare la vista APPELLI_SCRITTI_ORALI**,** la vista APPELLO_CON_AULA è immediata

```
CREATE VIEW  APPELLO_CON_AULA AS
SELECT NA,
      CON_AULA_S_N =CASE
            WHEN (AULA='NO_AULA' OR AULA='AULA_INDEFINITA')
                  THEN 0
            ELSE 1
      END
FROM APPELLI_SCRITTI_ORALI
```

SI VERIFICA QUESTA VISTA SELEZIONANDONE GLI ELEMENTI

NA	CON_AULA_S_N
1	0
2	1
3	1
4	0
5	0
6	0
7	0

A questo punto è facile constatare che NUMERO_APPELLI_CON_AULA è una semplice somma dell'attributo CON_AULA_S_N

NUMERO_APPELLI_CON_AULA = **SUM**(CON_AULA_S_N)

Possiamo ora definire per ogni misura

1) **TIPOLOGIA** della misura: Normale, calcolata, derivata, misura vuota
2) **ALIMENTAZIONE**: Cosa deve essere messo nella FACT_TABLE e come deve essere calcolato
3) **AGGREGAZIONE**: L'operatore di aggregazione da usare nel DATAMART; per le misure calcolate occorre definire l'operatore di aggregazione delle misure componenti
4) **NON AGGREGABILITA**: Eventuali non aggregabilità

Consideriamo le quattro misure :

NUMERO_ESAMI= **COUNT**(*)

TIPOLOGIA : E' una misura normale (quindi da riportare nella FACT_TABLE)

ALIMENTAZIONE: COUNT(*)

AGGREGAZIONE: ADDITIVA

NON AGGREGABILITA : nessuna (additiva rispetto a tutte le dimensioni)

VOTO_MEDIO = **AVG(**VOTO**)**

TIPOLOGIA : E' una misura calcolata, VOTO_SUM/ VOTO_COUNT

ALIMENTAZIONE: VOTO_SUM=SUM(VOTO),
VOTO_COUNT=COUNT(VOTO)

AGGREGAZIONE: VOTO_SUM e VOTO_COUNT sono additive

NON AGGREGABILITA : nessuna

NUMERO_APPELLI_CON_AULA = **SUM**(CON_AULA_S_N)

TIPOLOGIA : E' una misura normale

ALIMENTAZIONE: SUM(CON_AULA_S_N)

AGGREGAZIONE: ADDITIVA

NON AGGREGABILITA : nessuna

NUMERO_STUDENTI = **COUNT**(**DISTINCT** STUDENTE)

TIPOLOGIA : E' una misura normale

ALIMENTAZIONE: COUNT(**DISTINCT** STUDENTE)

AGGREGAZIONE:

ADDITIVA rispetto a FACOLTA, in quanto STUDENTE→ FACOLTA

ADDITIVA rispetto a DATA, in quanto
per dipendenza funzionale tra le dimensioni APPELLO → DATA
abbiamo che esiste X = {APPELLO} tale che
$X \cup$ STUDENTE → DATA

NON AGGREGABILITA : NA = {APPELLO}

Supponendo di aver verificato che VOTO assume anche dei valori NULL, allora VOTO_COUNT è diverso da NUMERO_ESAMI e quindi verrà esplicitamente inserito nella FACT_TABLE

2.a) PROGETTAZIONE LOGICA :

La particolarità di questo **schema logico** è la presenza di un attributo cross-dimensionale:

```
FACT_TABLE(FACOLTA, DATA,APPELLO:DT_APPELLO,
      NUMERO_ESAMI,VOTO_SUM,VOTO_COUNT,NUMERO_APPELLI_CON_AULA,NUMERO_STUDENTI)

      DT_TIPO(APPELLO,DOCENTE,AULA)

      DT_TIPO(DOCENTE,AULA,TIPO)
```

In questo semplice caso lo star schema e lo snowflake coincidono.

2.b) ALIMENTAZIONE:

Si riporta lo script (codice SQL) della creazione delle **viste corrispondenti** alla **FACT_TABLE** e alle **DIMENSION TABLE**; queste viste devono essere create sul DBO.

```
CREATE VIEW  FACT_TABLE
AS
      SELECT FACOLTA,DATA,APPELLO=A.NA,
            NUMERO_ESAMI= COUNT(*),
            VOTO_SUM = SUM (VOTO),
            VOTO_COUN = COUNT(VOTO),
            NUMERO_STUDENTI  = COUNT(DISTINCT E.STUDENTE),
            NUMERO_APPELLI_CON_AULA = SUM(CON_AULA_S_N)
      FROM ESAME E JOIN APPELLO A ON (E.NA=A.NA)
                  JOIN STUDENTE S ON (E.STUDENTE=S.STUDENTE)
                  JOIN APPELLO_CON_AULA AA ON (A.NA=AA.NA)
      GROUP BY
            A.NA, DATA,FACOLTA


CREATE VIEW DT_APPELLO
AS
SELECT
APPELLO=NA,DOCENTE,AULA
FROM APPELLI_SCRITTI_ORALI_TIPO
```

Si noti che nella precedente query non è necessario mettere DISTINCT in quanto la SELECT contiene NA che è la chiave di APPELLO e quindi è chiave nella view `APPELLI_SCRITTI_ORALI_TIPO`.

Il DISTINCT è necessario invece nella view:

```
CREATE VIEW DT_DOCENTE
AS
SELECT DISTINCT DOCENTE,AULA,TIPO
FROM APPELLI_SCRITTI_ORALI_TIPO
```

A questo punto il progetto delle viste di alimentazione è terminato; possiamo quindi usare tale view per verificare la correttezza nei calcoli delle misure

Misura VOTO_MEDIO per il pattern {}

```
-- NEL DM
SELECT VOTO_MEDIO =SUM(VOTO_SUM)/SUM(VOTO_COUN) FROM FACT_TABLE

--NEL DBO
SELECT VOTO_MEDIO =AVG(VOTO) FROM ESAME

-- VERIFICHIAMO NON AGGREGABILITA' DI NUMERO_STUDENTI RISPETTO AD APPELLO
-- CONSIDERANDO IL PATTERN {FACOLTA} --> VALORE NON CORRETTO

SELECT FACOLTA,
 COUNT(DISTINCT ESAME.STUDENTE) AS NUMERO_STUDENTI
 FROM ESAME JOIN STUDENTE ON ESAME.STUDENTE=STUDENTE.STUDENTE
GROUP BY FACOLTA


SELECT SUM(NUMERO_STUDENTI) AS NUMERO_STUDENTI
   FROM FACT_TABLE
GROUP BY FACOLTA

-- CONSIDERANDO ORA IL PATTERN {APPELLO}

SELECT APPELLO=NA,
 COUNT(DISTINCT ESAME.STUDENTE) AS NUMERO_STUDENTI
 FROM ESAME
 GROUP BY NA

SELECT APPELLO, SUM(NUMERO_STUDENTI) AS NUMERO_STUDENTI
   FROM FACT_TABLE
GROUP BY APPELLO

-- IL PATTERN APPELLO RESTITUISCE VALORI CORRETTI : QUESTO CONFERMA
--- L'ADDITIVITA' DI NUMERO_STUDENTI RISPETTO A DATA E RISPETTO A APPELLO


-- COME ESEMPIO RELATIVO ALL'ATTRIBUTO CROSS-DIMENSIONALE
-- CONSIDERIAMO IL PATTERN {TIPO, DOCENTE}

SELECT TIPO,A.DOCENTE,
NUMERO_ESAMI=SUM(NUMERO_ESAMI),
VOTO = SUM(VOTO_SUM)/SUM(VOTO_COUN)
FROM FACT_TABLE F JOIN DT_APPELLO A ON (F.APPELLO=A.APPELLO)
JOIN DT_TIPO T ON (A.AULA=T.AULA AND A.DOCENTE=T.DOCENTE)
GROUP BY TIPO,A.DOCENTE
-- SI NOTI IL JOIN PER LA DT_TIPO
```

Backup del DB

http://www.dbgroup.unimo.it/SIA/EsempioAppelli.bak

1.8.4 Nota sulle Misure

Le misure sono riferite alle dimensioni nel senso che quando parliamo di NUMERO_ESAMI è sottinteso che ci stiamo riferendo al numero di esami effettuati per un certo APPELLO, in una certa DATA, per una certa FACOLTA (quindi per studenti di una certa FACOLTA).

In effetti, in presenza di dipendenze funzionali tra le dimensioni è più preciso dire: numero di esami effettuati per un certo APPELLO e per una certa FACOLTA.

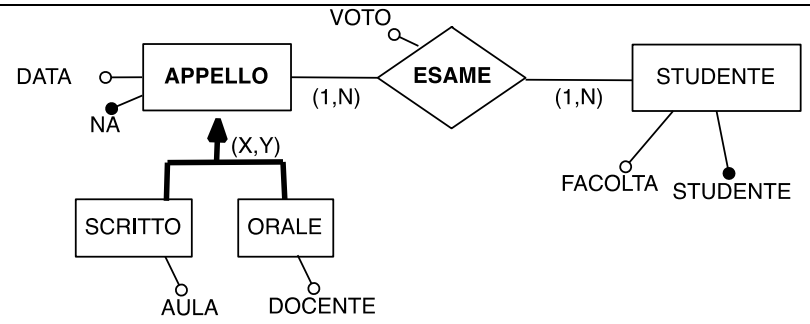
Quindi nello schema di fatto non vengono normalmente inserite misure riferite ad un certo attributo dimensionale, quale ad esempio DOCENTE per definire `NUMERO_ESAMI_PER_DOCENTE`

Per ogni FACOLTA, il `NUMERO_ESAMI_PER_DOCENTE` inteso come numero medio di esami per docente

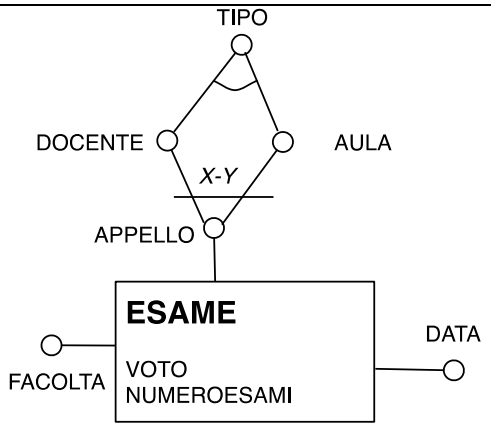
```
SELECT FACOLTA,
NUMERO_ESAMI=SUM(NUMERO_ESAMI),
NUMERO_ESAMI_PER_DOCENTE=SUM(NUMERO_ESAMI)/COUNT(DISTINCT DOCENTE)
FROM FACT_TABLE F JOIN DT_APPELLO A ON (F.APPELLO=A.APPELLO)
GROUP BY FACOLTA
```

1.8.5 Variante

Partendo dal progetto realizzato, di seguito sintetizzato

Schema ER del DBO	**Schema relazionale del DBO**
	STUDENTE(<u>STUDENTE</u>,FACOLTA) APPELLO(<u>NA,</u>DATA) SCRITTO(<u>NA:APPELLO</u>,AULA) ORALE(<u>NA:APPELLO</u>,DOCENTE) ESAME(<u>NA:APPELLO</u>, <u>STUDENTE:STUDENTE</u>, VOTO)

{AULA,DOCENTE} → TIPO
attributo cross-dimensionale



Discutere cosa cambia <u>se la DATA non è più un attributo di APPELLO ma è un attributo chiave di ESAME</u>

APPELLO(<u>NA,</u>~~<u>DATA</u>~~)

ESAME(<u>DATA,NA:APPELLO</u>,
<u>STUDENTE:STUDENTE</u>,
VOTO)

(nel backup dato la nuova relazione ESAME è ESAME2)

SOLUZIONE

Lo Schema di fatto rimane **temporale**, ma ovviamente non vale più la dipendenza funzionale tra le dimensioni APPELLO → DATA.

Quali sono i punti che erano influenzati da questa dipendenza funzionale?

La dipendenza funzionale tra le dimensioni APPELLO → DATA

Era stata usata per valutare l'aggregabilità di

NUMERO_STUDENTI = **COUNT**(**DISTINCT** STUDENTE)

In assenza di tale dipendenza funzionale NUMERO_STUDENTI non risulta più additiva rispetto a DATA, quindi l'insieme NA delle sue non aggregabilità è NA = {APPELLO,DATA}.

Questa è la principale *modifica* al progetto realizzato.

Le altre modifiche riguardano le viste per l'alimentazione che usano DATA: quindi si analizzano le query per decidere se è necessario apportare delle modifiche.

Nell'esempio in questione DATA è utilizzata solo da `FACT_TABLE`, ed è facile verificare che essendo ora DATA su ESAME non è più necessario il JOIN con APPELLO

```
CREATE VIEW  FACT_TABLE
AS
SELECT FACOLTA,DATA,APPELLO=A.NA,
NUMERO_ESAMI= COUNT(*),
VOTO_SUM = SUM (VOTO),
VOTO_COUN = COUNT(VOTO),
NUMERO_STUDENTI  = COUNT(DISTINCT E.STUDENTE),
NUMERO_APPELLI_CON_AULA = SUM(CON_AULA_S_N)
FROM ESAME E ~~JOIN APPELLO A ON (E.NA=A.NA)~~
             JOIN STUDENTE S ON (E.STUDENTE=S.STUDENTE)
             JOIN APPELLO_CON_AULA AA ON (A.NA=AA.NA)
GROUP BY  A.NA, DATA,FACOLTA
```