Team 09, Progetto di Basi di Dati 2021

Parte I

1. Requisiti Disambiguati:

- Le figure coach e caposquadra sono modellate dalla gerarchia utente:
    - il coach è solo un moderatore.
    - il caposquadra è sia moderatore che giocatore.

- Abbiamo interpretato la figura dell'admin come un'entità esterna agli utenti del gioco, quindi abbiamo evitato di inserirlo nella gerarchia.

- Un dado è associato ad un solo gioco, poiché dadi associati a più giochi potrebbero risultare problematici e confusionali.

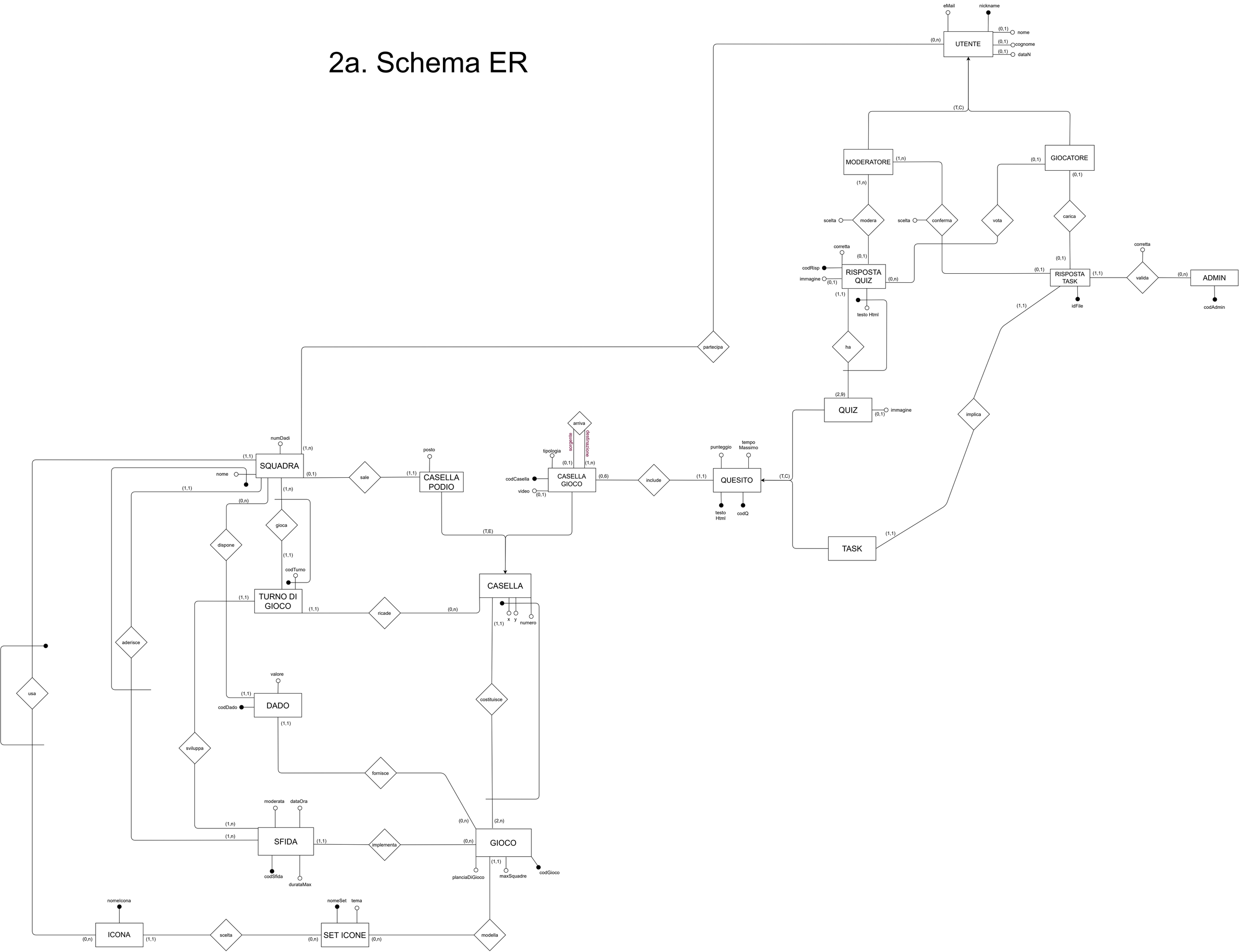
- Siccome non viene specificato che se in una casella ci sono uno o più quiz allora non ci sono task, abbiamo inteso che una cosa non escluda l'altra; per noi ci possono essere quindi in una casella sia quiz (al massimo 5), sia task (al massimo uno); da questa interpretazione deriva la gerarchia condivisa e il vincolo di cardinalità (0,6) da casella di gioco a quesito.

- Abbiamo seguito l'esempio nella specifica e abbiamo introdotto le caselle "scala" e "serpente", generalizzandole a caselle che permettano al giocatore di avanzare o indietreggiare ad altre caselle; in questo modo anche se un gioco non usa esplicitamente i nomi "scala" e "serpente" può comunque possedere delle caselle che hanno quel comportamento.

- Abbiamo ipotizzato che le caselle necessarie in un gioco fossero almeno 5: tre di esse per il podio, due di esse perché sono inizio e fine delle caselle di gioco.

- Riguardo ai punteggi dei quiz e dei task, se si risponde correttamente ad essi viene assegnato un punteggio positivo pari ai punti che valgono (attributo "punteggio" di Quesito); in caso di risposta sbagliata invece viene detratto alla squadra lo stesso punteggio.

2a. Schema ER


eMail
nickname
UTENTE
(0,1)
nome
(0,1)
cognome
(0,1)
dataN
(0,n)
(T,C)
MODERATORE
(1,n)
GIOCATORE
(0,1)
(1,n)
(0,1)
scelta
modera
scelta
conferma
vota
carica
corretta
(0,1)
codRisp
immagine
(0,1)
RISPOSTA QUIZ
(0,n)
(0,1)
RISPOSTA TASK
(1,1)
corretta
valida
(0,n)
ADMIN
codAdmin
idFile
(1,1)
testo Html
(1,1)
ha
(2,9)
QUIZ
(0,1)
immagine
implica
partecipa
numDadi
(1,n)
(1,1)
SQUADRA
nome
(1,1)
(0,1)
sale
(1,1)
posto
CASELLA PODIO
(1,n)
(0,n)
gioca
dispone
(1,1)
codTurno
(1,1)
TURNO DI GIOCO
(1,1)
ricade
(0,n)
CASELLA
(1,1)
x
y
numero
arriva
sorgente
destinazione
tipologia
(0,1)
(1,n)
codCasella
video
(0,1)
CASELLA GIOCO
(0,6)
include
(1,1)
punteggio
tempo Massimo
QUESITO
(T,C)
testo Html
codQ
(T,E)
TASK
(1,1)
aderisce
usa
valore
(1,1)
codDado
DADO
(1,1)
costituisce
sviluppa
fornisce
moderata
dataOra
(1,n)
(1,n)
SFIDA
(1,1)
codSfida
durataMax
implementa
(0,n)
(0,n)
(2,n)
GIOCO
(1,1)
planciaDiGioco
maxSquadre
codGioco
nomeIcona
(0,n)
ICONA
(1,1)
scelta
nomeSet
tema
(0,n)
SET ICONE
(0,n)
modella

2b. Documentazione Relativa ai Domini degli Attributi:

1. dom(CASELLA PODIO, posto) = {Primo, Secondo, Terzo}
2. dom(CASELLA, x) = numero che rappresenta l'ascissa rispetto all'immagine di sfondo del gioco (ovvero alla plancia).
3. dom(CASELLA, y) = numero che rappresenta l'ordinata rispetto all'immagine di sfondo del gioco (ovvero alla plancia).
4. dom(CASELLA DI GIOCO, tipologia) = {Inizio, Fine, Normale, CasellaAvanza, CasellaIndietreggia}
5. dom(DADO, valore) = {1, …, 6}; corrisponde all'ultimo lancio del dado.
6. dom(TURNO DI GIOCO, codTurno) = SERIAL
7. dom(SFIDA, dataOra) = TIMESTAMP
8. dom(SFIDA, moderata) = BOOLEAN
9. dom(SFIDA, durataMax) = INTEGER (espressa in minuti)
10. dom(RISPOSTA_QUIZ, corretta) = BOOLEAN
11. dom(valida, corretta) = BOOLEAN
12. dom(modera, scelta) = BOOLEAN
13. dom(conferma, scelta) = BOOLEAN

2c. Vincoli non Esprimibili nel Diagramma:

1. La casella start (punto di partenza) è quella con numero 1 e la casella arrivo è quella con numero massimo.
2. Ogni risposta ha un punteggio positivo se giusta, nullo o negativo se sbagliata.
3. Se la squadra non fornisce risposta alla domanda e al task entro il tempo massimo queste scadono e il punteggio ottenuto è nullo.
4. La casella destinazione potrebbe anche essere condizionata dal punteggio che si ottiene rispondendo alle domande/al task posti sulla casella: a diversi punteggi potrebbero essere associate caselle destinazione diverse.
5. Il comportamento della casella può includere il lancio dei dadi.
6. Il punteggio che si ottiene rispondendo alle domande/al task posti sulla casella può modificare il numero di dadi a disposizione della squadra.
7. Nel caso la casella preveda una casella destinazione (eventualmente condizionata dal punteggio) allora non prevedrà il lancio dei dadi.
8. Se non è specificata casella successiva ci sarà lancio dei dadi e avanzamento di un numero di caselle pari al punteggio ottenuto con i dadi.
9. Le icone delle squadre devono essere tutte distinte e tutte della stessa dimensione, derivanti dal set delle icone scelto per quel gioco.
10. L'utente appartiene a un'unica squadra tra quelle che partecipano alla stessa sfida.
11. Allo scadere del tempo vengono raccolte tutte le risposte.
12. Una risposta quiz ha un testo html, che per quel determinato quiz non si può ripetere; ad esempio, posso avere due risposte quiz con testo "sì" oppure "no" ma solo se sono associate a quiz diversi.

2d. Specifica dei Tipi di Gerarchie di Generalizzazione:

1. UTENTE è totale e condivisa.
2. CASELLA è totale ed esclusiva.
3. QUESITO è totale e condivisa.

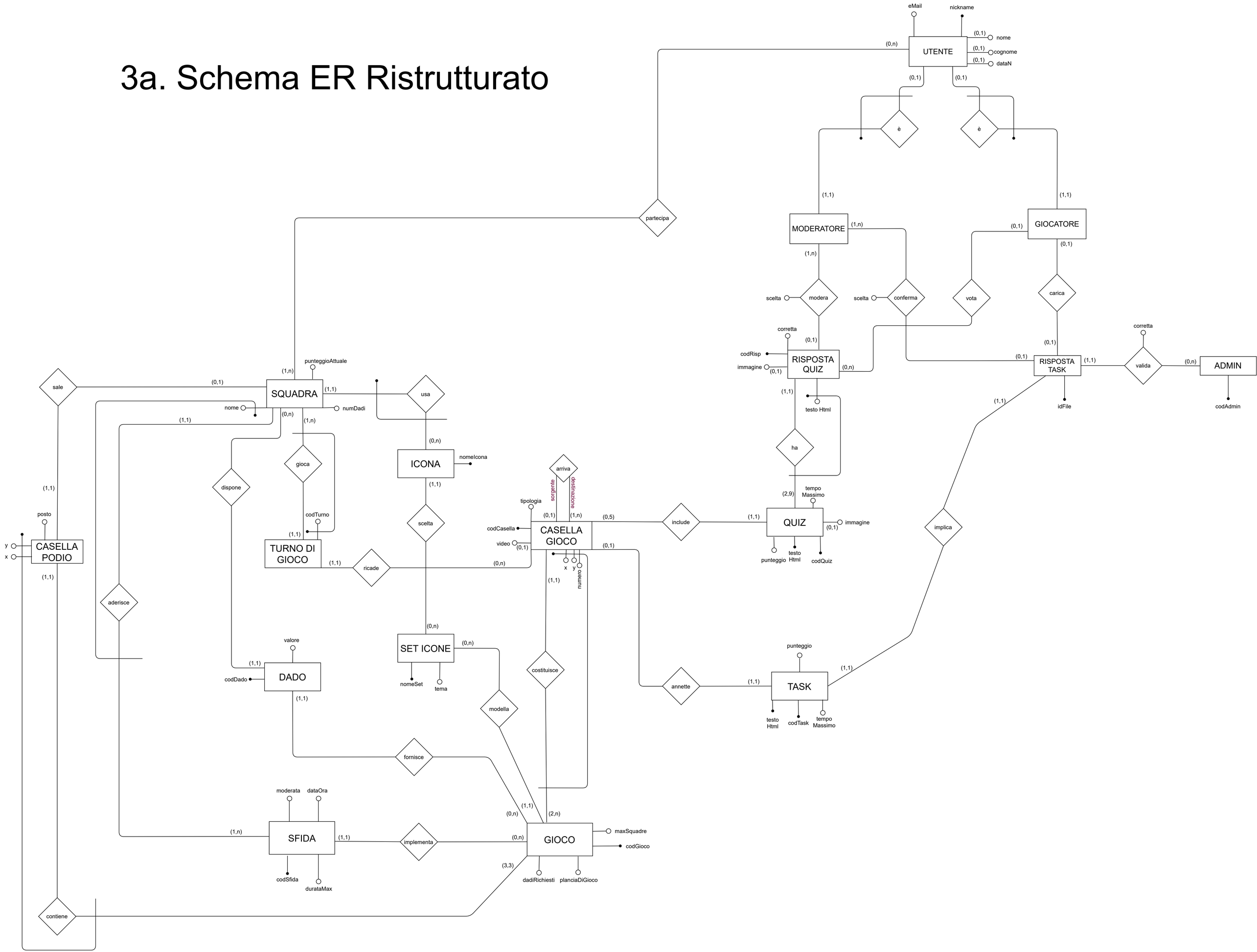
Annotazioni:

1. Icona non è inclusa nell'identificazione mista, poiché in quel caso due squadre potrebbero avere la stessa icona, a patto di avere nomi diversi, oppure potrebbero avere lo stesso nome, a patto di

avere icone diverse. Noi invece vogliamo che tutte le squadre abbiano sia icone che nomi diversi; per questo abbiamo incluso il nome nell'identificazione mista e abbiamo definito l'univocità dell'icona tramite vincolo unique (pallino nero) che deriva dall'identificazione esterna e abbiamo ribadito la sua univocità nei vincoli.

2. Abbiamo evitato un'associazione tra dado e turno di gioco, esplicitando il fatto che il valore del dado in un qualsiasi momento deve essere il valore dell'ultimo lancio di quel dado; questo quindi dovrebbe essere gestito tramite un trigger che aggiorni ogni volta l'attributo valore.

3a. Schema ER Ristrutturato



3b. Eventuali modifiche dei domini degli attributi e informazioni sui domini di eventuali attributi introdotti:

1. Abbiamo cambiato il dominio del seguente attributo:
   dom(CASELLA PODIO, posto) = {1, 2, 3}
   Cambiando il tipo dell'attributo per semplificare il dominio (che a questo punto può essere SMALLINT) ed il vincolo CHECK che andremo ad inserire nelle fasi successive.
2. Nella traduzione della gerarchia di CASELLA, l'entità figlia CASELLA_PODIO avrebbe dovuto ereditare l'attributo "numero", che però è solo ridondante essendoci già "posto", e quindi è stato eliminato e lasciato solo in CASELLA_GIOCO.
3. Nello schema concettuale è presente l'attributo "CodQ" in Quesito che viene tradotto come "CodQuiz" in Quiz e "CodTask" in Task.

Sono stati fatti altri cambiamenti, in termini di aggiunta e modifica di attributi, al fine di ottimizzare le fasi successive; essi sono descritti nel punto 3f (bis).

3c. Modifiche all'elenco di Vincoli del Modello Concettuale:

Abbiamo inserito i seguenti vincoli:

1. Se due squadre hanno lo stesso punteggio viene visualizzata per prima sul podio la squadra che precede l'altra in ordine alfabetico.
2. A ogni casella può essere associato un solo task.
3. Un utente deve essere o un moderatore o un giocatore oppure entrambe le cose (non può non essere né un giocatore né un moderatore).

3d. Scelte Fatte per eliminare le Gerarchie di Generalizzazione:

1. Per quanto riguarda UTENTE abbiamo sostituito la gerarchia con delle associazioni che collegano UTENTE a GIOCATORE e UTENTE a MODERATORE; questa scelta è stata condizionata dal tipo di gerarchia (condivisa), che impedisce l'eliminazione dell'entità padre; inoltre, le operazioni fanno distinzione tra le varie sotto-entità, quindi la scelta di eliminare le entità figlie sarebbe stata problematica.
2. La gerarchia su CASELLA è stata cancellata eliminando l'entità padre in quanto è la scelta più conveniente, essendo nel caso in cui esistono operazioni che si riferiscono alle istanze delle specifiche entità figlie, ovvero CASELLA GIOCO e CASELLA PODIO; queste ultime sono infatti associate a entità diverse e risultano quindi molto differenti nelle operazioni che svolgono.
3. Analogamente a quella di CASELLA, la gerarchia di QUESITO è stata cancellata eliminando l'entità padre, in quanto le entità figlie hanno ambiti diversi: una riguarda i QUIZ e tutto ciò che è associato ad essi, e l'altra riguarda i TASK.

Non erano presenti attributi multi-valori o attributi composti nello schema.

3e. Schema Logico

SQUADRA(<u>nome, codSfida</u>^{SFIDA}, numDadi, punteggioAttuale, *nomeIcona*^{ICONA})

ICONA(<u>nomeIcona</u>, nomeSet^{SET_ICONE})

PARTECIPA(<u>nome^{SQUADRA}, codSfida^{SQUADRA}, nickname^{UTENTE}</u>)

CASELLA_PODIO(<u>x, y, codGioco^{GIOCO}</u>, nome^{SQUADRA}, codSfida^{SQUADRA}, posto)

TURNO_DI_GIOCO(<u>codTurno, nome^{SQUADRA}, codSfida^{SQUADRA},</u> codCasella^{CASELLA_GIOCO})

DADO(<u>codDado</u>, nome^{SQUADRA}, codSfida^{SQUADRA}, valore, codGioco^{GIOCO})

SFIDA(<u>codSfida</u>, dataOra, durataMax, codGioco^{GIOCO}, moderata)

CASELLA_GIOCO(<u>codCasella</u>, *x, y, numero, codGioco*^{GIOCO}, video$_O$, tipologia, codCasellaArrivo$_O$^{CASELLA_GIOCO})

GIOCO(<u>codGioco</u>, dadiRichiesti, maxSquadre, planciaDiGioco, nomeSet^{SET_ICONE})

SET_ICONE(<u>nomeSet</u>, tema)

QUIZ(<u>codQuiz</u>, *testoHtml*, punteggio, tempoMassimo, immagine$_O$, codCasella^{CASELLA_GIOCO})

TASK(<u>codTask</u>, *testoHtml*, punteggio, tempoMassimo, codCasella^{CASELLA_GIOCO})

* RISPOSTA_QUIZ(<u>codRisp</u>, *testoHtml*, immagine$_O$, *codQuiz*^{QUIZ}, nicknameModeratore$^{MODERATORE}{}_O$, corretta, scelta$_O$)

RISPOSTA_TASK(<u>idFile</u>, codTask^{TASK}, nicknameModeratore$^{MODERATORE}{}_O$, codAdmin^{ADMIN}, corretta, scelta$_O$)

ADMIN(<u>codAdmin</u>)

UTENTE(<u>nickname</u>, eMail, nome$_O$, cognome$_O$, dataN_O)

MODERATORE(<u>nickname^{UTENTE}</u>)

GIOCATORE(<u>nickname^{UTENTE}</u>, codRisp$^{RISPOSTA_QUIZ}{}_O$, idFile$^{RISPOSTA_TASK}{}_O$)

* la coppia di attributi (testoHtml, codQuiz^{QUIZ}) è una chiave candidata (unique); gli attributi presi singolarmente invece non lo sono.

3f. Verifica di Qualità dello Schema:

SQUADRA(nome, codSfida[SFIDA], numDadi, punteggioAttuale, nomeIcona[ICONA])

nome, codSfida -> numDadi, punteggioAttuale, nomeIcona

ICONA(nomeIcona, nomeSet[SET_ICONE])

nomeIcona -> nomeSet

PARTECIPA(nome[SQUADRA], codSfida[SFIDA], nickname[UTENTE])

CASELLA_PODIO(x, y, codGioco[GIOCO], nome[SQUADRA], codSfida[SFIDA], posto)

x, y, codGioco -> nome, codSfida, posto

TURNO_DI_GIOCO(codTurno, nome[SQUADRA], codSfida[SQUADRA], codCasella[CASELLA_GIOCO])

codTurno, nome, codSfida -> codCasella

DADO(codDado, nome[SQUADRA], codSfida[SQUADRA], valore, codGioco[GIOCO])

codDado -> nome, codSfida, valore, codGioco

SFIDA(codSfida, dataOra, durataMax, codGioco[GIOCO], moderata)

codSfida -> dataOra, durataMax, codGioco, moderata

CASELLA_GIOCO(codCasella, *x, y, numero, codGioco*[*GIOCO*], video$_O$, tipologia, codCasellaArrivo$_O$[CASELLA_GIOCO])

codCasella -> x, y, numero, codGioco, video, tipologia, codCasellaArrivo

x, y, numero, codGioco -> codCasella, video, tipologia, codCasellaArrivo

GIOCO(codGioco, dadiRichiesti, maxSquadre, planciaDiGioco, nomeSet[SET_ICONE])

codGioco -> dadiRichiesti, maxSquadre, planciaDiGioco, nomeSet

SET_ICONE(nomeSet, tema)

nomeSet -> tema

QUIZ(codQuiz, *testoHtml*, punteggio, tempoMassimo, immagine$_O$, codCasella[CASELLA_GIOCO])

codQuiz -> testoHtml, punteggio, tempoMassimo, immagine, codCasella

testoHtml -> codQuiz, punteggio, tempoMassimo, immagine, codCasella

TASK(codTask, *testoHtml*, punteggio, tempoMassimo, codCasella^{CASELLA_GIOCO})

codTask -> testoHtml, punteggio, tempoMassimo, codCasella

testoHtml -> codTask, punteggio, tempoMassimo, codCasella

RISPOSTA_QUIZ(codRisp, *testoHtml*, immagine$_O$, *codQuiz*^{QUIZ}, nicknameModeratore$^{MODERATORE}{}_O$, corretta, scelta$_O$)

codRisp -> testoHtml, immagine, codQuiz, nicknameModeratore, corretta, scelta

testoHtml, codQuiz -> codRisp, immagine, nicknameModeratore, corretta, scelta

RISPOSTA_TASK(idFile, codTask^{TASK}, nicknameModeratore$^{MODERATORE}{}_O$, codAdmin^{ADMIN}, corretta, scelta$_O$)

idFile -> codTask, nicknameModeratore, codAdmin, corretta, scelta

ADMIN(codAdmin)

UTENTE(nickname, eMail, nome$_O$, cognome$_O$, dataN$_O$)

nickname -> eMail, nome, cognome, dataN

MODERATORE(nickname^{UTENTE})

GIOCATORE(nickname^{UTENTE}, codRisp$^{RISPOSTA_QUIZ}{}_O$, idFile$^{RISPOSTA_TASK}{}_O$)

nickname -> codRisp, idFile

Tutte le parti sinistre delle dipendenze funzionali sono chiavi candidate per le relazioni.

Lo schema è quindi già normalizzato secondo la forma normale di Boyce-Codd.

3f (bis). Eventuali ottimizzazioni applicate tenendo in considerazione il carico di lavoro:

1. Abbiamo inserito l'attributo "punteggioAttuale" in Squadra, inserendo ridondanza, ma semplificando le operazioni di calcolo dei punteggi nei vari turni di gioco; questo ha semplificato di molto il nostro lavoro nel trigger 4b.

2. Abbiamo inserito l'attributo "dadiRichiesti" in Gioco per semplificare un'interrogazione nel carico di lavoro.
3. Abbiamo posto codTurno come SERIAL per gestire il secondo trigger con più facilità; questo ci dà infatti la possibilità di riferirci al turno precedente.

4. Progetto Fisico

Svolgiamo l'attività di tuning fisico, che si preoccupa di progettare uno schema fisico, in termini di insieme di indici creati; lo schema deve essere realizzato al fine di rendere possibile un'esecuzione efficiente delle interrogazioni contenute nel **workload**.

Consideriamo quindi le interrogazioni relative al workload, andandole ad analizzare una alla volta; siccome esse hanno la stessa frequenza le consideriamo in ordine numerico:

1) Determinare l'identificatore dei giochi che coinvolgono al più quattro squadre e richiedono l'uso di due dadi.

Gli attributi che appaiono in una clausola WHERE sono candidati come chiavi di ricerca per un indice; in questo caso gli attributi che ci interessano sono maxSquadre e dadiRichiesti.

Proviamo quindi a ipotizzare un potenziale piano di esecuzione ottimale per l'interrogazione:

- Cerchiamo di eseguire la selezione sulla condizione del WHERE nel modo più efficiente possibile; *maxSquadre = 4 AND dadiRichiesti = 2* è un **fattore booleano**, ovvero una condizione che, se falsa, rende falsa tutta l'interrogazione; sfruttiamo quindi questa condizione per creare un piano di esecuzione ottimale.

Individuiamo gli indici che permettono al sistema di prendere in considerazione il piano individuato nello spazio dei piani; per la selezione possiamo utilizzare vari indici:

- Indici su singolo attributo, ad albero o hash, su maxSquadre e/o su dadiRichiesti; entrambi i tipi di indici sono possibili, poiché entrambi gli attributi sono coinvolti in ricerche per uguaglianza;
- Multi-attributo ad albero o hash su maxSquadre e dadiRichiesti,
- Multi-attributo ad albero o hash su dadiRichiesti e maxSquadre.

Valutiamo, dopo aver considerato le altre interrogazione, quale opzione scegliere ed anche se **clusterizzare** o meno questo/i indice/i.

2) Determinare l'identificatore delle sfide relative a un gioco A di vostra scelta (ovvero 'OCA') che, in alternativa:
   - hanno avuto luogo a gennaio 2021 e durata massima superiore a 2 ore, o
   - hanno avuto luogo a marzo 2021 e durata massima pari a 30 minuti.

Gli attributi che appaiono in una clausola WHERE sono candidati come chiavi di ricerca per un indice; in questo caso sono codGioco, dataOra e durataMax.

Proviamo quindi a ipotizzare un potenziale piano di esecuzione ottimale per l'interrogazione e individuiamo gli indici necessari:

- Per la selezione consideriamo tre indici su codGioco, dataOra e durataMax: quelli su codGioco e dataOra possono essere indici hash, quello su durataMax è per forza un indice ordinato.
  Purtroppo, in questo caso non vi è un fattore booleano a causa dell'OR all'interno della condizione; si può quindi cercare di creare vari indici sugli attributi interessati per poi unire i loro risultati.

3) Determinare le sfide, di durata massima superiore a due ore, dei giochi che richiedono almeno due dadi. Restituire sia l'identificatore della sfida sia l'identificatore del gioco.

Gli attributi che appaiono in una clausola WHERE sono candidati come chiavi di ricerca per un indice; in questo caso l'unico attributo in questione è durataMax.

Inoltre, dobbiamo considerare un piano efficiente per effettuare il join.

Proviamo quindi a ipotizzare un potenziale piano di esecuzione ottimale per l'interrogazione:

- Per eseguire il join nel modo più efficiente possibile consideriamo l'algoritmo di merge join;
- Per la selezione consideriamo un indice ordinato su durataMax, che viene creato ed utilizzato nel secondo punto.

Individuiamo gli indici che permettono al sistema di prendere in considerazione il piano individuato nello spazio dei piani:

- Per eseguire il join mediante merge join servono vari indici ordinati clusterizzati: quello su Sfida.codGioco esiste già (dovremmo clusterizzarlo) mentre ne serve uno su Gioco.codGioco ed un terzo su Dado.codGioco.
- Per la selezione ci serve l'indice ad albero su durataMax, già creato nel punto precedente.

Indici Scelti:

Andiamo quindi a riassumere quali potrebbero essere degli indici efficienti per i vari piani di esecuzione delle tre interrogazioni nel workload:

- Per la prima interrogazione creiamo un indice multi-attributo ordinato su maxSquadre e dadiRichiesti.
- Per la seconda interrogazione creiamo indici su codGioco, dataOra e durataMax; quello su codGioco lo clusterizziamo poiché ci servirà anche nella terza interrogazione.
- Per la terza interrogazione creiamo due indici ordinati clusterizzati rispettivamente su Dado.codGioco e Gioco.codGioco (per il merge join); il terzo indice clusterizzato, su Sfida.codGioco esiste già.
  Per la selezione abbiamo già l'indice ad albero su durataMax creato nel punto precedente.