Entrega II

Definição do esquema relacional

Relation reference	Relation Compact Notation
R01	users(id, name **NN**, email **NN UK**, wallet **DF** 0.0, country)
R02	studio(id, name **NN UK** , description)
R03	game(id, title **NN**, price **NN**, rating **DF** 50, publisher -> Studio, developer -> Studio)
R04	achievement(id, name **NN**, description **NN**, game -> Game)
R05	users_achievement(users -> users, achievement -> achievement)
R06	review(users -> users, game -> game, body, rating **NN**)
R07	ownership(users -> users, game -> game)
R08	wishlist(users -> users, game -> game)
R09	follow_studio(users -> users, studio -> studio)
R10	friendship(user1 -> users, user2 -> users **CK** user2 != user1)
R11	communities(id, name **NN**)
R12	post(id, title **NN**, body **NN**, rating **DF** 50, author -> users, community -> communities)
R13	comment(id, body **NN**, rating **DF** 50, author -> users, post -> post)
R14	privileges(users -> users, community -> communities, is_administrator **DF** false)

where **UK** means **UNIQUE KEY**, **NN** means **NOT NULL**, **DF** means **DEFAULT** and **CK** means **CHECK**.

Análise de Dependências Funcionais e Formas Normais

Table R01	users
Keys	{id, email}
Functional Dependencies:	
FD0101	{id} -> {name, email, wallet, country}
FD0102	{email} -> {id, name, wallet, country}
Normal Form	BCNF

Table R02	studio
Keys	{id, name}
Functional Dependencies:	
FD0201	{id} -> {name, description}
FD0202	{name} -> {id, description}
Normal Form	BCNF

Table R03	game
Keys	{id}
Functional Dependencies:	
FD0301	{id} -> {title, price, rating, publisher, developer}
Normal Form	BCNF

Table R04	achievement
Keys	{id}
Functional Dependencies:	
FD0401	{id} -> {name, description, game}
Normal Form	BCNF

Table R05	users_achievement
Keys	{users, achievement}
Functional Dependencies:	
(None)	
Normal Form	BCNF

Table R06	review
Keys	{users, game}
Functional Dependencies:	
FD0601	{users, game} -> {body, rating}
Normal Form	BCNF

Table R07	ownership
Keys	{users, game}
Functional Dependencies:	
(None)	
Normal Form	BCNF

Table R08	wishlist
Keys	{users, game}
Functional Dependencies:	
(None)	
Normal Form	BCNF

Table R09	follow_studio
Keys	{users, studio}
Functional Dependencies:	
(None)	
Normal Form	BCNF

Table R10	friendship
Keys	{user1, user2}
Functional Dependencies:	
(None)	
Normal Form	BCNF

Table R11	communities
Keys	{id}
Functional Dependencies:	
FD1101	{id} -> {name}
Normal Form	BCNF

Table R12	post
Keys	{id}
Functional Dependencies:	
FD1201	{id} -> {title, body, rating, author, community}
Normal Form	BCNF

Table R13	comment
Keys	{id}
Functional Dependencies:	
FD1301	{id} -> {body, rating, author, post}
Normal Form	BCNF

Table R14	privileges
Keys	{users, community}
Functional Dependencies:	
FD1401	{users, community} -> {is_administrator}
Normal Form	BCNF

Interrogações

Seguem, em baixo, as 10 interrogações por ordem dos ficheiros entregues, descritas em linguagem corrente e com a respetiva tradução em álgebra relacional.

Int1: Selecionar as *reviews* de um jogo em específico, neste caso o jogo de id = 1.

$$\sigma \text{ game\_id} = 1 \text{ (review) } \tau \text{ rating, DESC}$$

Int2: Selecionar todos os *achivements* de um dado jogo de um dado *user*, neste caso o *user* de id = 2 e o jogo com id = 1.

$$\pi \text{ achievement.name } (\sigma \text{ users\_achivement.user\_id} = 2 \wedge \text{game.id} = 1 \text{ ((users\_achivement} \bowtie \text{achievement)} \bowtie \text{game))}$$

Int3: Ordenar a *wishlist* de um dado user por preço descendente, neste caso o *user* de id = 6.

$$\pi \text{ game.title, game.price } (\sigma \text{ user\_id} = 6 \text{ (game} \bowtie \text{wishlist))} \tau \text{ game.price, DESC}$$

Int4: Selecionar todos os jogos de um dado estúdio, ou seja, todos os jogos que esse estúdio desenvolveu ou publicou. Neste caso, utilizamos o estúdio com id = 1.

$$\sigma \text{ game.publisher\_id} = 1 \vee \text{game.developer\_id} = 1 \text{ (game)}$$

Int5: Atualizar o preço de um dado jogo, neste caso o jogo de id = 3.

$$\text{game} \leftarrow \pi \text{ game.price} = 30 \text{ (}\sigma \text{ game.id} = 3 \text{ (game))}$$

Int6: Colocar um dado *user* como administrador de uma comunidade, neste caso o *user* de id = 4 e a comunidade com id = 1.

$$\text{privileges} \leftarrow \pi \text{ is\_administrator} = \text{true } (\sigma \text{ user\_id} = 4 \wedge \text{community\_id} = 1\text{(privileges))}$$

Int7: Apagar as *reviews* de um dado jogo escritas por um dado *user*, neste caso o *user* de id = 5 e o jogo de id = 1.

$$\text{review} \leftarrow \text{review} - \sigma \text{ game\_id} = 1 \wedge \text{user\_id} = 5 \text{ (review)}$$

Int8: Selecionar todos os comentários de um *post* com pontuação superior a 33, neste caso do *post* com id = 1.

$$\sigma \text{ post\_id} = 1 \wedge \text{reting} > 33 \text{ (comment) } \tau \text{ rating, ASC}$$

Int9: Selecionar todos os jogos que os amigos de um têm, ordenando-os depois de forma decrescente. Neste caso, foi utilizado o *user* de id = 5.

$$\pi \text{ game.title, users.name } ((\pi \text{ F1.user1 } (\sigma \text{ F1.user2 = 5 } (\text{friendship F1})) \cup \pi \text{ F2.user2 } (\sigma \text{ F2.user1 = 5 } (\text{friendship F2})) \bowtie \text{ownership}) \bowtie \text{game}) \tau \text{ game.id, DESC}$$

Int10: Selecionar os jogos de um certo estúdio que um *user* não tem mas os seus amigos têm. Neste caso, utilizamos o *user* de id = 5 e o estúdio de id = 1.

$$\pi \text{ id } ((\pi \text{ F1.user1 } (\sigma \text{ F1.user2 = 5 } (\text{friendship F1})) \cup \pi \text{ F2.user2 } (\sigma \text{ F2.user1 = 5 } (\text{friendship F2})) \bowtie \text{ownsership}) \bowtie (\pi \text{ GAME.id } (\sigma \text{ publisher\_id = 1 } (\text{game GAME})))) - \pi \text{ game\_id } (\sigma \text{ user\_id = 5 } (\text{ownership}))$$

Gatilhos

Seguem, em baixo, os 3 gatilhos por ordem dos ficheiros entregues, descritos em linguagem corrente.

Gatilho1: Após a compra de um determinado jogo, caso este se encontrasse previamente na *wishlist* do *user*, será removido da mesma.

Gatilho2: Em todas as comunidades, um *user* pode ser administrador. Como tal, concebemos este gatilho de forma a preservar a consistência da tabela. Assim, caso alguém tente colocar o *user* como administrador e este já tenha esse papel, ou retira-lo de administrador e este já tenha sido retirado, ocurrerá um erro.

Gatilho3: Este gatilho apaga todos os comentários de um *post* quando este é apagado, de modo a garantir que a tabela não terá nenhum comentário que não esteja associado a um *post* ainda existente.