Person(

- bigint personId
- String firstName NOT NULL
- String lastName NOT NULL
- String gender
- Date birthday NOT NULL
- timestamp creationDate NOT NULL
- String locationIp NOT NULL
- String browserUsed
- int city //Fremdschlüssel auf City.placeId on Update CASCADE on Delete SET NULL
- language String[]

)

HasInterest(

- bigint personId //Fremdschlüssel auf Person.personId_on Update CASCADE on Delete CASCADE
- int TagId /Fremdschlüssel auf Tag.tagId on Update CASCADE on Delete CASCADE

)

Email(

- bigint personId //Fremdschlüssel auf Person.personId on Update CASCADE on Delete CASCADE
- String email

)

Knows(

- bigint person1Id //Fremdschlüssel auf Person.personId on Update CASCADE on Delete CASCADE
- bigint person2Id //Fremdschlüssel auf Person.personId on Update CASCADE on Delete CASCADE
- timestamp creationDate NOT NULL

)

Forum(

- bigint forumId
- String title NOT NULL
- timestamp creationDate NOT NULL
- bigint_moderator //Fremdschlüssel auf Person.personId on Update CASCADE on Delete SET NULL

)

ForumHasTag(

- bigint ForumId //Fremdschlüssel auf Forum.forumId_on Update CASCADE on Delete CASCADE
- int TagId //Fremdschlüssel auf Tag.tagId on Update CASCADE on Delete CASCADE

)

HasMember(

- bigint forumId //Fremdschlüssel auf Forum.forumId on Update CASCADE on Delete SET NULL

- bigint personId //Fremdschlüssel auf Person.personId on Update CASCADE on Delete CASCADE
- timestamp joinDate NOT NULL

)

Tag(

- int tagId
- String name NOT NULL
- String url

)

HasType(

- int TagId /Fremdschlüssel auf Tag.tagId on Update CASCADE on Delete CASCADE
- int tagClassId //Fremdschlüssel auf TagClass.TagClassId on Update CASCADE on Delete CASCADE

)

TagClass(

- int tagClassId
- String name NOT NULL
- String url

)

TagSubClasses(

- int parentClass //Frendschlüssel auf TagClass.tagClassId on Update CASCADE on Delete CASCADE
- int childClass //Frendschlüssel auf TagClass.tagClassId on Update CASCADE on Delete CASCADE

)

Message (//extends Interface //Für JoinedTable wichtig

- bigint messageId
- timestamp creationDate NOT NULL
- String locationIp //bekommt man vllt durch Person zugewiesen
- String browserUsed //bekommt man durch Person zugewiesen
- Text content
- 32bitInteger length
- bigint creator //Fremdschlüssel auf Person.personId on Update CASCADE on Delete SET NULL
- int country //Fremdschlüssel auf Country.placeId on Update CASCADE on Delete SET NULL

)

MessageHasTag(

- bigint messageId //Fremdschlüssel auf Message.messageId_on Update CASCADE on Delete CASCADE
- int tagId /Fremdschlüssel auf Tag.tagId on Update CASCADE on Delete CASCADE

)

Comment(//implements Message

- *int messageId //Fremdschlüssel auf Message.messageId on Update CASCADE on Delete CASCADE*

- Message replyOn NOT NULL //Fremdschlüssel auf Message.messageId on Update CASCADE on Delete CASCADE

)

Post(//implements Message

- bigint *messageId //Fremdschlüssel auf Message.messageId on Update CASCADE on Delete CASCADE*
- String language
- String imageFile
- bigint forumId //Fremdschlüssel auf Forum.forumId on Update CASCADE on Delete SET NULL

)

Likes(

- bigint personId //Fremdschlüssel auf Person.personId on Update CASCADE on Delete CASCADE
- bigint messageId //Fremdschlüssel auf Message.messageId on Update CASCADE on Delete CASCADE
- timestamp creationDate NOT NULL

)

University(//extends Organisation

- *int organisationId //Fremdschlüssel auf Organisation.organistationId on Update CASCADE on Delete CASCADE*
- int city //Fremdschlüssel auf City.placeId on Update CASCADE on Delete SET NULL

)

Company(//extends Organisation

- *int organisationId //Fremdschlüssel auf Organisation.organistationId on Update CASCADE on Delete CASCADE*
- int country //Fremdschlüssel auf Country.placeId on Update CASCADE on Delete SET NULL

)

Organisation(

- *int organisationId*
- *String name* NOT NULL

)

Country(//extends Place

- *int placeId //Fremdschlüssel auf Place.placeID on Update CASCADE on Delete CASCADE*
- int continent //Fremdschlüssel auf Continent.placeId on Update CASCADE on Delete NO ACTION

)

Place(

- *int placeId*
- *String name* NOT NULL

)

City(//extends Place

- *int placeId //Fremdschlüssel auf Place.placeID on Update CASCADE on Delete CASCADE*
- int country //Fremdschlüssel auf Country.placeId on Update CASCADE on Delete NO ACTION

)

Continent(//extends Place

- *int placeId //Fremdschlüssel auf Place.placeID on Update CASCADE on Delete CASCADE*

)

StudyAt(

- bigint personId //Fremdschlüssel auf Person.personId on Update CASCADE on Delete CASCADE
- int universityId //Fremdschlüssel auf University.organisationId on Update CASCADE on Delete CASCADE
- 32bitInteger classYear

)

WorkAt(

- bigint personId //Fremdschlüssel auf Person.personId on Update CASCADE on Delete CASCADE
- int companyId //Fremdschlüssel auf Company.organisationId on Update CASCADE on Delete CASCADE
- 32bitInteger workSince

)

WorkHistory(

- int workHistoryId (SERIAL)
- int personId
- int companyId
- 32bitInteger workSince
- timestamp endDate

)

Pirate(

- String gamertag
- int personId //Fremdschlüssel auf Person.personId on Update CASCADE on Delete SET NULL
- int gold
- int doubloons
- String bodyType

)

Fleet(

- String captain //Fremdschlüssel auf Pirate.gamertag on Update CASCADE on Delete CASCADE
- int shipID //Fremdschlüssel auf Ship.shipId on Update CASCADE on Delete CASCADE

)

Ship(

- int shipId

- String type NOT NULL
- String name
- Skin JSON

)

FactionRelation(

- String gamertag //Fremdschlüssel auf Pirate.gamertag on Update CASCADE on Delete CASCADE
- String faction //Fremdschlüssel auf Faction.name on Update CASCADE on Delete CASCADE
- float level

)

Faction(

- String name
- String description
- int maxLvl NOT NULL

)

Die Primärschlüssel und die kombinierten Primärschlüssel müssen immer eindeutig und (auch die Teile der Primärschlüssel) dürfen nicht NULL sein

Mehr Bedingungen:

Bedingung	Erklärung
FactionRelation.level >= 0 FactionRelation.level <= Faction.maxLvl	Fraktionslevel: Der Wert von level in der Tabelle FactionRelation muss zwischen 0 und dem maximalen Level (maxLvl) der Fraktion liegen.
Pirate.gold >= 0 Pirate.doubloons >= 0	Gold und Dublonen: In der Tabelle Pirate müssen gold und doubloons mindestens 0 sein.
UNIQUE Email.email	E-Mail-Adresse: Jede email in der Tabelle Email muss eindeutig sein
Person.birthday <= CURRENT_DATE	Geburtstag: Der birthday darf nicht in der Zukunft liegen.
email ~* '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$	E-Mail-Adresse: Die email-Adresse muss ein gültiges Format haben.

2.d)

```
 CREATE OR REPLACE FUNCTION log_work_history()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
        INSERT INTO WorkHistory (personId, companyId, workSince, endDate)
        VALUES (OLD.personId, OLD.companyId, OLD.workSince, CURRENT_TIMESTAMP);
        RETURN OLD;
END;
$$;


CREATE TRIGGER after_delete_workAt
AFTER DELETE ON WorkAt
FOR EACH ROW
EXECUTE FUNCTION log_work_history();
```